1

SET UP AND CONFIGURATION HADOOP USING CLOUDERA CREATING A HDFS SYSTEM WITH MINIMUM 1 NAME NODE AND 1 DATA NODES HDFS COMMANDS

Unit Structure :

1.1 Objectives

1.2 Prerequisite

1.3 GUI Configuration

1.4 Command Line Configuration

1.5 Summary

1.6 Sample Questions

1.7 References

1.1 OBJECTIVES

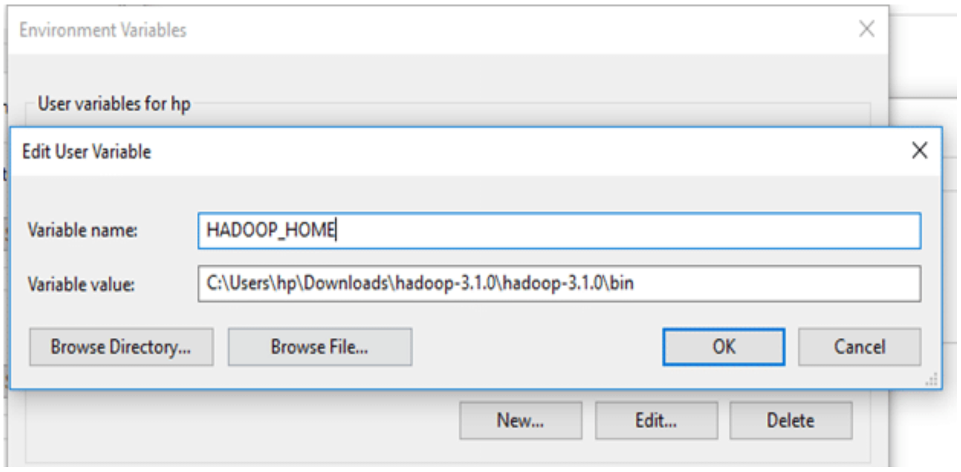
Hadoop file system stores the data in multiple copies. Also, it's a cost-effective solution for any business to store their data efficiently. HDFS Operations acts as the key to open the vaults in which you store the data to be available from remote locations. This chapter describes how to set up and edit the deployment configuration files for HDFS

1.2 PREREQUISITE: TO INSTALL HADOOP, YOU SHOULD HAVE JAVA VERSION 1.8 IN YOUR SYSTEM.

Check your java version through this command on command prompt

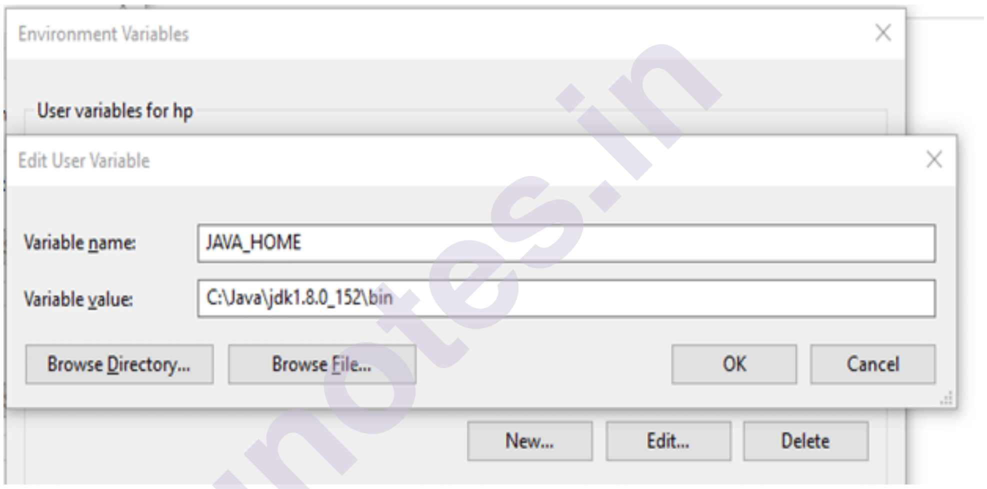
Java -version

Create a new user variable. Put the Variable_name as HADOOP_HOME and Variable_value as the path of the bin folder where you extracted hadoop.



Variable name: HADOOP_HOME
Variable value: C:\Users\hp\Downloads\hadoop-3.1.0\hadoop-3.1.0\bin

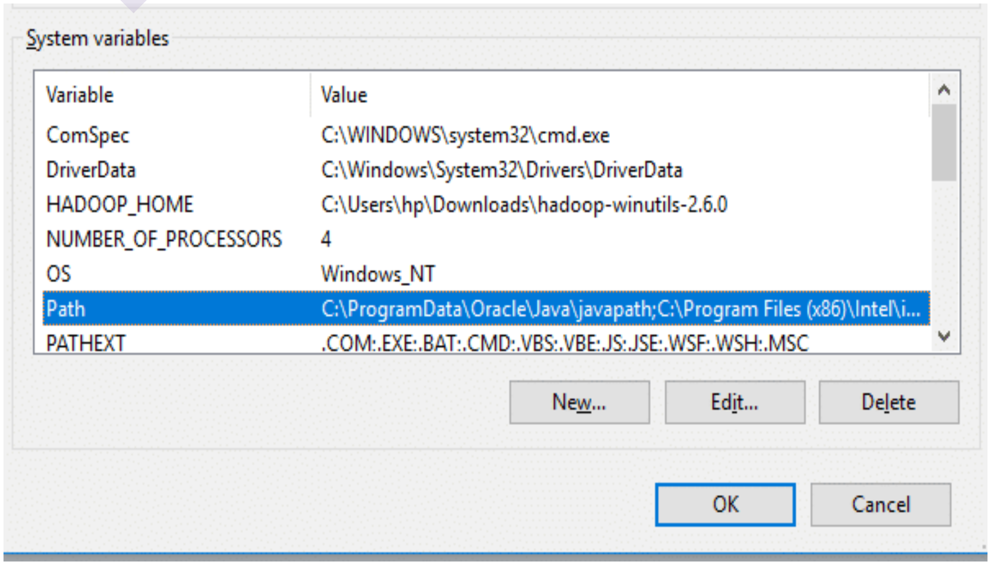
Likewise, create a new user variable with variable name as JAVA_HOME and variable value as the path of the bin folder in the Java directory.



Variable name: JAVA_HOME
Variable value: C:\Java\jdk1.8.0_152\bin

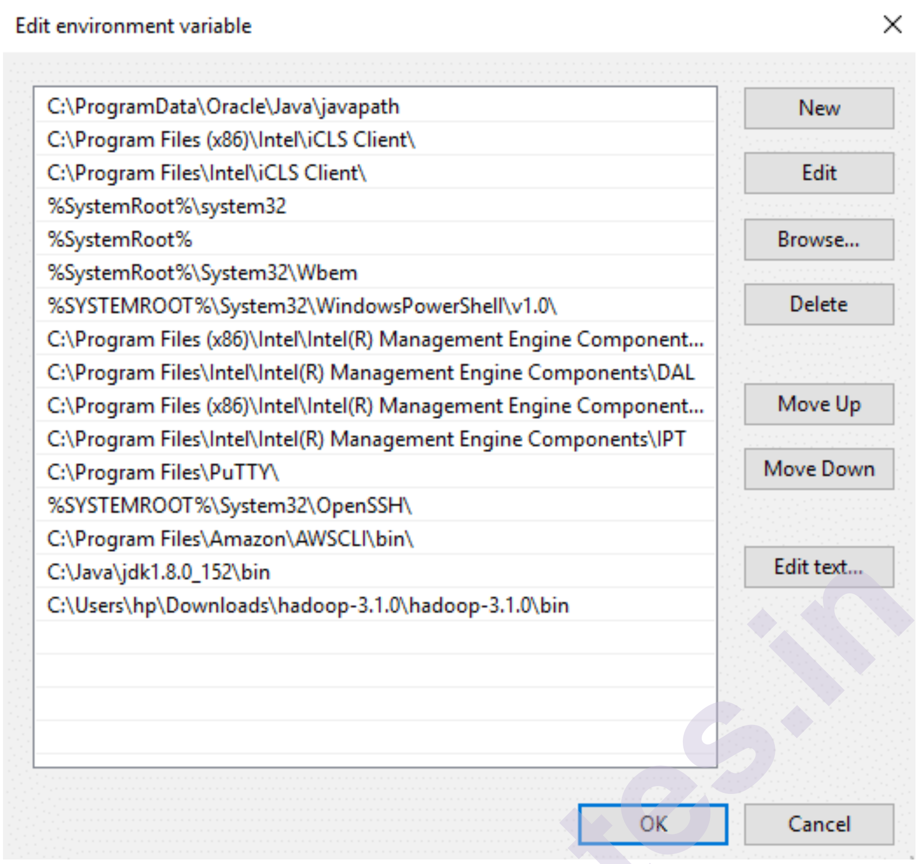
Now we need to set Hadoop bin directory and Java bin directory path in system variable path.

Edit Path in system variable



Variable	Value
ComSpec	C:\WINDOWS\system32\cmd.exe
DriverData	C:\Windows\System32\Drivers\DriverData
HADOOP_HOME	C:\Users\hp\Downloads\hadoop-winutils-2.6.0
NUMBER_OF_PROCESSORS	4
OS	Windows_NT
Path	C:\ProgramData\Oracle\Java\javapath;C:\Program Files (x86)\Intel\i...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC

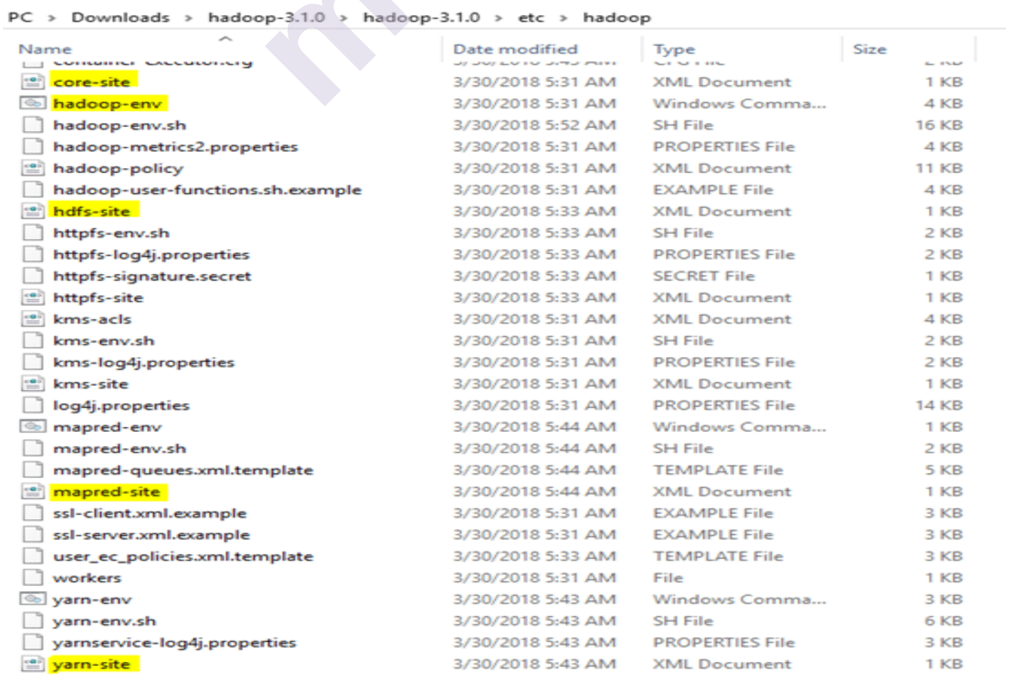
Click on New and add the bin directory path of Hadoop and Java in it.



```
C:\ProgramData\Oracle\Java\javapath
C:\Program Files (x86)\Intel\iCLS Client\
C:\Program Files\Intel\iCLS Client\
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
C:\Program Files (x86)\Intel\Intel(R) Management Engine Component...
C:\Program Files\Intel\Intel(R) Management Engine Components\DAL
C:\Program Files (x86)\Intel\Intel(R) Management Engine Component...
C:\Program Files\Intel\Intel(R) Management Engine Components\IPT
C:\Program Files\PuTTY\
%SYSTEMROOT%\System32\OpenSSH\
C:\Program Files\Amazon\AWSCLI\bin\
C:\Java\jdk1.8.0_152\bin
C:\Users\hp\Downloads\hadoop-3.1.0\hadoop-3.1.0\bin
```

1.3 GUI CONFIGURATIONS

Now we need to edit some files located in the hadoop directory of the etc folder where we installed hadoop. The files that need to be edited have been highlighted.



PC > Downloads > hadoop-3.1.0 > hadoop-3.1.0 > etc > hadoop

Name	Date modified	Type	Size
core-site	3/30/2018 5:31 AM	XML Document	1 KB
hadoop-env	3/30/2018 5:31 AM	Windows Comma...	4 KB
hadoop-env.sh	3/30/2018 5:52 AM	SH File	16 KB
hadoop-metrics2.properties	3/30/2018 5:31 AM	PROPERTIES File	4 KB
hadoop-policy	3/30/2018 5:31 AM	XML Document	11 KB
hadoop-user-functions.sh.example	3/30/2018 5:31 AM	EXAMPLE File	4 KB
hdfs-site	3/30/2018 5:33 AM	XML Document	1 KB
httpfs-env.sh	3/30/2018 5:33 AM	SH File	2 KB
httpfs-log4j.properties	3/30/2018 5:33 AM	PROPERTIES File	2 KB
httpfs-signature.secret	3/30/2018 5:33 AM	SECRET File	1 KB
httpfs-site	3/30/2018 5:33 AM	XML Document	1 KB
kms-acls	3/30/2018 5:31 AM	XML Document	4 KB
kms-env.sh	3/30/2018 5:31 AM	SH File	2 KB
kms-log4j.properties	3/30/2018 5:31 AM	PROPERTIES File	2 KB
kms-site	3/30/2018 5:31 AM	XML Document	1 KB
log4j.properties	3/30/2018 5:31 AM	PROPERTIES File	14 KB
mapred-env	3/30/2018 5:44 AM	Windows Comma...	1 KB
mapred-env.sh	3/30/2018 5:44 AM	SH File	2 KB
mapred-queues.xml.template	3/30/2018 5:44 AM	TEMPLATE File	5 KB
mapred-site	3/30/2018 5:44 AM	XML Document	1 KB
ssl-client.xml.example	3/30/2018 5:31 AM	EXAMPLE File	3 KB
ssl-server.xml.example	3/30/2018 5:31 AM	EXAMPLE File	3 KB
user_ec_policies.xml.template	3/30/2018 5:33 AM	TEMPLATE File	3 KB
workers	3/30/2018 5:31 AM	File	1 KB
yarn-env	3/30/2018 5:43 AM	Windows Comma...	3 KB
yarn-env.sh	3/30/2018 5:43 AM	SH File	6 KB
yarnservice-log4j.properties	3/30/2018 5:43 AM	PROPERTIES File	3 KB
yarn-site	3/30/2018 5:43 AM	XML Document	1 KB

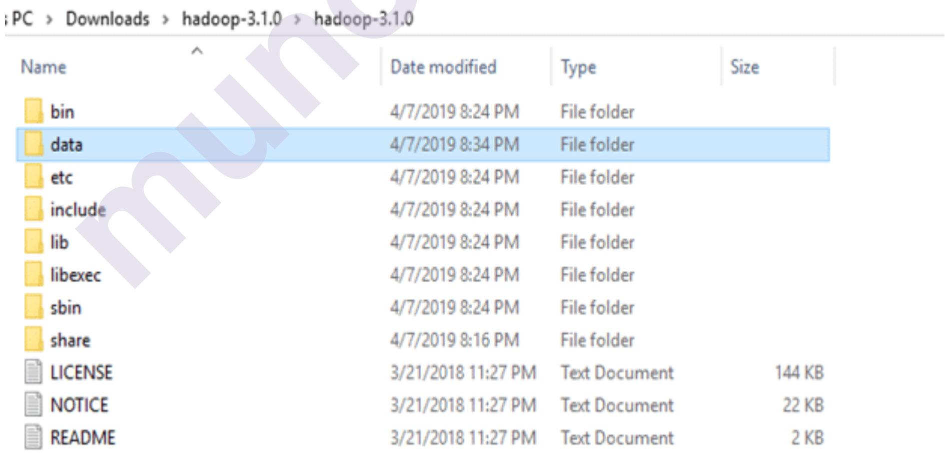
1. **Edit the file core-site.xml in the hadoop directory. Copy this xml property in the configuration in the file**

```
<configuration>
  <property>
     <name>fs.defaultFS</name>
     <value>hdfs://localhost:9000</value>
  </property>
</configuration>
```

2. **Edit mapred-site.xml and copy this property in the configuration**

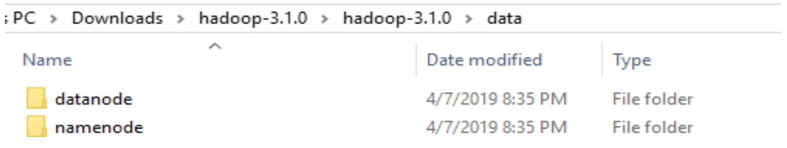
```
<configuration>
  <property>
     <name>mapreduce.framework.name</name>
     <value>yarn</value>
  </property>
</configuration>
```

3. **Create a folder 'data' in the hadoop directory**


PC › Downloads › hadoop-3.1.0 › hadoop-3.1.0

Name	Date modified	Type	Size
bin	4/7/2019 8:24 PM	File folder	
data	4/7/2019 8:34 PM	File folder	
etc	4/7/2019 8:24 PM	File folder	
include	4/7/2019 8:24 PM	File folder	
lib	4/7/2019 8:24 PM	File folder	
libexec	4/7/2019 8:24 PM	File folder	
sbin	4/7/2019 8:24 PM	File folder	
share	4/7/2019 8:16 PM	File folder	
LICENSE	3/21/2018 11:27 PM	Text Document	144 KB
NOTICE	3/21/2018 11:27 PM	Text Document	22 KB
README	3/21/2018 11:27 PM	Text Document	2 KB

4. **Create a folder with the name 'datanode' and a folder 'namenode' in this data directory**


PC › Downloads › hadoop-3.1.0 › hadoop-3.1.0 › data

Name	Date modified	Type
datanode	4/7/2019 8:35 PM	File folder
namenode	4/7/2019 8:35 PM	File folder

5. **Edit the file hdfs-site.xml and add below property in the configuration**

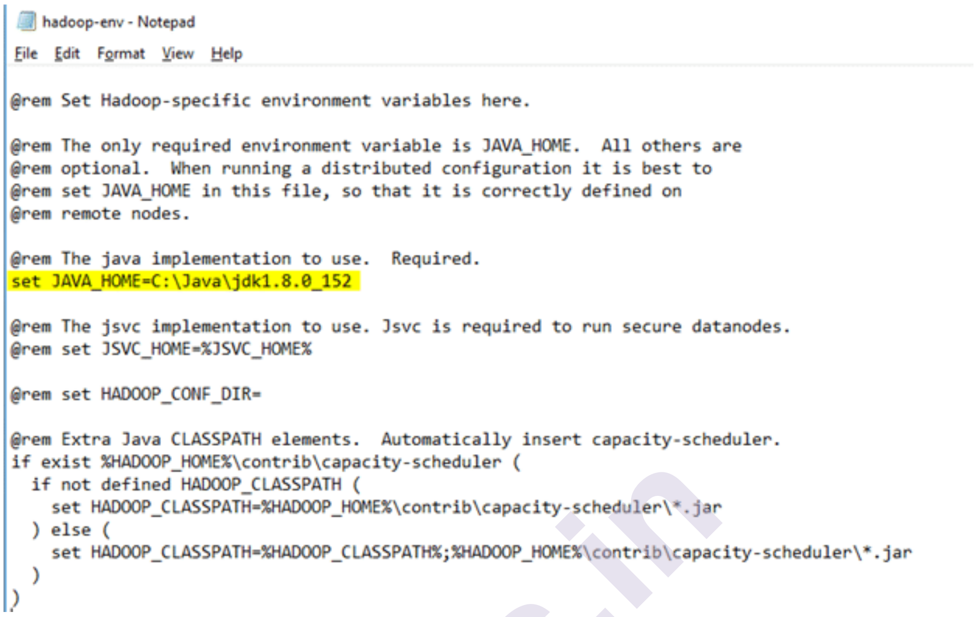
Note: The path of namenode and datanode across value would be the path of the datanode and namenode folders you just created.

```
<configuration>

  <property>

     <name>dfs.replication</name>

     <value>1</value>

  </property>

  <property>

     <name>dfs.namenode.name.dir</name>

     <value>C:\Users\hp\Downloads\hadoop-3.1.0\hadoop-
       3.1.0\data\namenode</value>

  </property>

  <property>

     <name>dfs.datanode.data.dir</name>

     <value>                    C:\Users\hp\Downloads\hadoop-3.1.0\hadoop-
       3.1.0\data\datanode</value>

  </property>

</configuration>
```

6. **Edit the file yarn-site.xml and add below property in the configuration**

```
<configuration>

  <property>

       <name>yarn.nodemanager.aux-services</name>

       <value>mapreduce_shuffle</value>

  </property>

  <property>


               <name>yarn.nodemanager.auxservices.mapreduce.shuffle.clas
       s</name>

       <value>org.apache.hadoop.mapred.ShuffleHandler</value>

  </property>

</configuration>
```

7. Edit hadoop-env.cmd and replace %JAVA_HOME% with the path of the java folder where your jdk 1.8 is installed



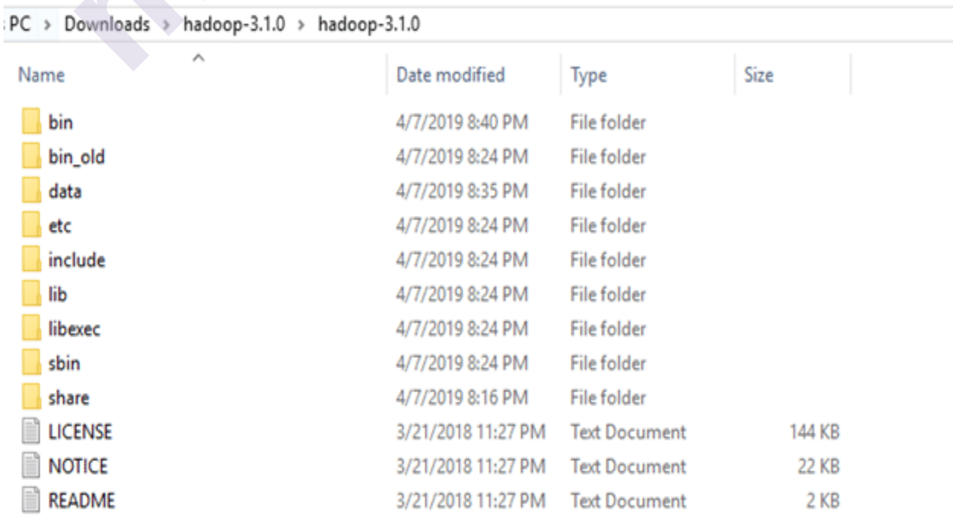
```
@rem Set Hadoop-specific environment variables here.

@rem The only required environment variable is JAVA_HOME.  All others are
@rem optional.  When running a distributed configuration it is best to
@rem set JAVA_HOME in this file, so that it is correctly defined on
@rem remote nodes.

@rem The java implementation to use.  Required.
set JAVA_HOME=C:\Java\jdk1.8.0_152

@rem The jsvc implementation to use. Jsvc is required to run secure datanodes.
@rem set JSVC_HOME=%JSVC_HOME%

@rem set HADOOP_CONF_DIR=

@rem Extra Java CLASSPATH elements.  Automatically insert capacity-scheduler.
if exist %HADOOP_HOME%\contrib\capacity-scheduler (
  if not defined HADOOP_CLASSPATH (
    set HADOOP_CLASSPATH=%HADOOP_HOME%\contrib\capacity-scheduler\*.jar
  ) else (
    set HADOOP_CLASSPATH=%HADOOP_CLASSPATH%;%HADOOP_HOME%\contrib\capacity-scheduler\*.jar
  )
)
```

8. Hadoop needs windows OS specific files which does not come with default download of hadoop.

To include those files, replace the bin folder in hadoop directory with the bin folder provided in this github link.

https://github.com/s911415/apache-hadoop-3.1.0-winutils

Download it as zip file. Extract it and copy the bin folder in it. If you want to save the old bin folder, rename it like bin_old and paste the copied bin folder in that directory.



Name	Date modified	Type	Size
bin	4/7/2019 8:40 PM	File folder	
bin_old	4/7/2019 8:24 PM	File folder	
data	4/7/2019 8:35 PM	File folder	
etc	4/7/2019 8:24 PM	File folder	
include	4/7/2019 8:24 PM	File folder	
lib	4/7/2019 8:24 PM	File folder	
libexec	4/7/2019 8:24 PM	File folder	
sbin	4/7/2019 8:24 PM	File folder	
share	4/7/2019 8:16 PM	File folder	
LICENSE	3/21/2018 11:27 PM	Text Document	144 KB
NOTICE	3/21/2018 11:27 PM	Text Document	22 KB
README	3/21/2018 11:27 PM	Text Document	2 KB

Check whether hadoop is successfully installed by running this command on cmd-

hadoop –version

Format the NameNode

Formatting the NameNode is done once when hadoop is installed and not for running hadoop filesystem, else it will delete all the data inside HDFS. Run this command-

hdfs namenode –format

Now change the directory in cmd to sbin folder of hadoop directory with this command,

Start namenode and datanode with this command –

start-dfs.cmd

Two more cmd windows will open for NameNode and DataNode

Now start yarn through this command-

start-yarn.cmd

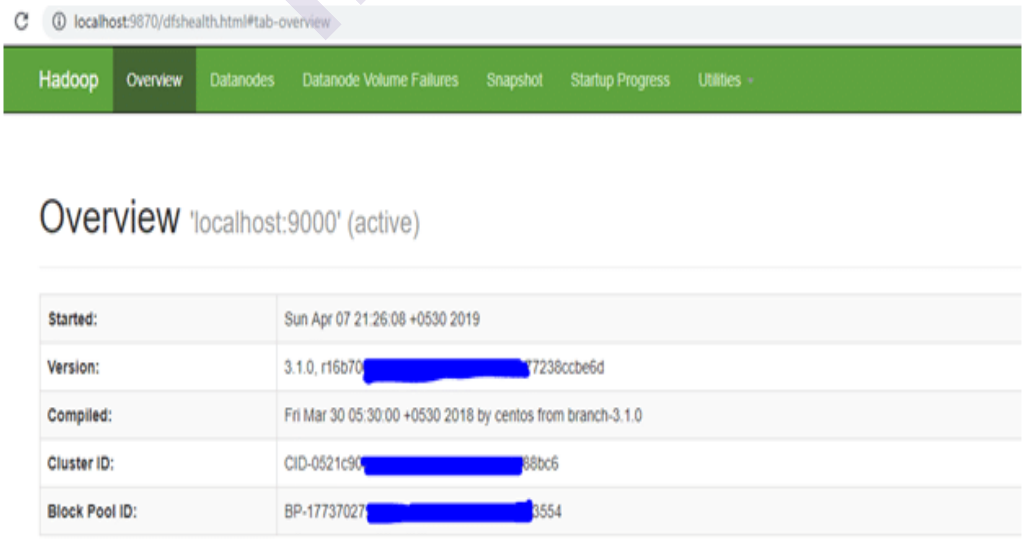
Note: Make sure all the 4 Apache Hadoop Distribution windows are up n running. If they are not running, you will see an error or a shutdown message. In that case, you need to debug the error.

To access information about resource manager current jobs, successful and failed jobs, go to this link in browser-

http://localhost:8088/cluster

To check the details about the hdfs (namenode and datanode),

http://localhost:9870/



Overview 'localhost:9000' (active)

Started:	Sun Apr 07 21:26:08 +0530 2019
Version:	3.1.0, r16b70[redacted]7238ccbe6d
Compiled:	Fri Mar 30 05:30:00 +0530 2018 by centos from branch-3.1.0
Cluster ID:	CID-0521c90[redacted]88bc6
Block Pool ID:	BP-17737027[redacted]3554

1.4 COMMAND LINE CONFIGURATION

Starting HDFS

Format the configured HDFS file system and then open the namenode (HDFS server) and execute the following command.

```
$ hadoop namenode -format
```

Start the distributed file system and follow the command listed below to start the namenode as well as the data nodes in cluster.

```
$ start-dfs.sh
```

Read & Write Operations in HDFS

You can execute almost all operations on Hadoop Distributed File Systems that can be executed on the local file system. You can execute various reading, writing operations such as creating a directory, providing permissions, copying files, updating files, deleting, etc. You can add access rights and browse the file system to get the cluster information like the number of dead nodes, live nodes, spaces used, etc.

HDFS Operations to Read the file

To read any file from the HDFS, you have to interact with the NameNode as it stores the metadata about the DataNodes. The user gets a token from the NameNode and that specifies the address where the data is stored.

You can put a read request to NameNode for a particular block location through distributed file systems. The NameNode will then check your privilege to access the DataNode and allows you to read the address block if the access is valid.

```
$ hadoop fs -cat <file>
```

HDFS Operations to write in file

Similar to the read operation, the HDFS Write operation is used to write the file on a particular address through the NameNode. This NameNode provides the slave address where the client/user can write or add data. After writing on the block location, the slave replicates that block and copies to another slave location using the factor 3 replication. The salve is then reverted back to the client for authentication.

The process for accessing a NameNode is pretty similar to that of a reading operation. Below is the HDFS write commence:

```
bin/hdfs dfs -ls  <path>
```

Listing Files in HDFS

Finding the list of files in a directory and the status of a file using 'ls' command in the terminal. Syntax of ls can be passed to a directory or a filename as an argument which are displayed as follows:

```
$ $HADOOP_HOME/bin/hadoop fs -ls <args>
```

Inserting Data into HDFS

Below mentioned steps are followed to insert the required file in the Hadoop file system.

Step1: Create an input directory

```
$ $HADOOP_HOME/bin/hadoop fs -mkdir /user/input
```

Step2: Use the put command transfer and store the data file from the local systems to the HDFS using the following commands in the terminal.

```
$ $HADOOP_HOME/bin/hadoop fs -put /home/intellipaat.txt /user/input
```

Step3: Verify the file using ls command.

```
$ $HADOOP_HOME/bin/hadoop fs -ls /user/input
```

Retrieving Data from HDFS

For instance, if you have a file in HDFS called Intellipaat. Then retrieve the required file from the Hadoop file system by carrying out:

Step1: View the data from HDFS using the cat command.

```
$ $HADOOP_HOME/bin/hadoop fs -cat /user/output/intellipaat
```

Step2: Gets the file from HDFS to the local file system using get command as shown below

```
$ $HADOOP_HOME/bin/hadoop fs -get /user/output/ /home/hadoop_tp/
```

Shutting Down the HDFS

Shut down the HDFS files by following the below command

```
$ stop-dfs.sh
```

Multi-Node Cluster

Installing Java

Syntax of java version command

```
$ java -version
```

Following output is presented.

```
java version "1.7.0_71"

Java(TM) SE Runtime Environment (build 1.7.0_71-b13)

Java HotSpot(TM) Client VM (build 25.0-b02, mixed mode)
```

Bottom of Form

Creating User Account

System user account is used on both master and slave systems for the Hadoop installation.

```
# useradd hadoop

# passwd hadoop
```

Mapping the nodes

Hosts files should be edited in /etc/ folder on each and every nodes and IP address of each system followed by their host names must be specified mandatorily.

```
# vi /etc/hosts
```

Enter the following lines in the /etc/hosts file.

```
192.168.1.109 hadoop-master

192.168.1.145 hadoop-slave-1

192.168.56.1 hadoop-slave-2
```

Configuring Key Based Login

Ssh should be set up in each node so they can easily converse with one another without any prompt for a password.

```
# su hadoop

$ ssh-keygen -t rsa

$ ssh-copy-id -i ~/.ssh/id_rsa.pub tutorialspoint@hadoop-master

$ ssh-copy-id -i ~/.ssh/id_rsa.pub hadoop_tp1@hadoop-slave-1

$ ssh-copy-id -i ~/.ssh/id_rsa.pub hadoop_tp2@hadoop-slave-2

$ chmod 0600 ~/.ssh/authorized_keys

$ exit
```

Installation of Hadoop

Hadoop should be downloaded in the master server using the following procedure.

```
# mkdir /opt/hadoop

# cd /opt/hadoop/

# wget http://apache.mesi.com.ar/hadoop/common/hadoop-1.2.1/hadoop-1.2.0.tar.gz

# tar -xzf hadoop-1.2.0.tar.gz

# mv hadoop-1.2.0 hadoop

# chown -R hadoop /opt/hadoop

# cd /opt/hadoop/hadoop/
```

Configuring Hadoop

Hadoop server must be configured in core-site.xml and should be edited wherever required.

```
<configuration>

<property>

<name>fs.default.name</name><value>hdfs://hadoop-master:9000/</value>

</property>

<property>

<name>dfs.permissions</name>

<value>false</value>

</property>

</configuration>
```

hdfs-site.xml file should be editted.

```
<configuration>

<property>

<name>dfs.data.dir</name>

<value>/opt/hadoop/hadoop/dfs/name/data</value>

<final>true</final>
```

```
</property>
<property>
<name>dfs.name.dir</name>
<value>/opt/hadoop/hadoop/dfs/name</value>
<final>true</final>
</property>
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
</configuration>
```

mapred-site.xml file should be edited as per the requirement example is being shown.

```
<configuration>
<property>
<name>mapred.job.tracker</name><value>hadoop-master:9001</value>
</property>
</configuration>
```

JAVA_HOME, HADOOP_CONF_DIR, and HADOOP_OPTS should be edited as follows:

```
export JAVA_HOME=/opt/jdk1.7.0_17
export HADOOP_OPTS=-Djava.net.preferIPv4Stack=true
export HADOOP_CONF_DIR=/opt/hadoop/hadoop/conf
```

Installing Hadoop on Slave Servers

Hadoop should be installed on all the slave servers

```
# su hadoop
$ cd /opt/hadoop
$ scp -r hadoop hadoop-slave-1:/opt/hadoop
$ scp -r hadoop hadoop-slave-2:/opt/hadoop
```

Configuring Hadoop on Master Server

Master server configuration

```
# su hadoop

$ cd /opt/hadoop/hadoop
```

Master Node Configuration

```
$ vi etc/hadoop/masters

hadoop-master
```

Slave Node Configuration

```
$ vi etc/hadoop/slaves

hadoop-slave-1

hadoop-slave-2
```

Name Node format on Hadoop Master

```
# su hadoop

$ cd /opt/hadoop/hadoop

$ bin/hadoop namenode –format

25/05/22 10:58:07 INFO namenode.NameNode: STARTUP_MSG:

************************************************************

STARTUP_MSG: Starting NameNode

STARTUP_MSG: host = hadoop-master/192.168.1.109

STARTUP_MSG: args = [-format]

STARTUP_MSG: version = 1.2.0

STARTUP_MSG: build =

https://svn.apache.org/repos/asf/hadoop/common/branches/branch-1.2 -r
1479473; compiled by 'hortonfo' on Monday May 23 06:59:37 UTC 2022

STARTUP_MSG: java = 1.7.0_71

************************************************************

25/05/22 10:58:08 INFO util.GSet: Computing capacity for map
BlocksMap editlog=/opt/hadoop/hadoop/dfs/name/current/edits

…………………………………………………….
```

```
25/05/22 10:58:08 INFO common.Storage: Storage directory /opt/hadoop/hadoop/dfs/name has been successfully formatted.

25/05/22 10:58:08 INFO namenode.NameNode: SHUTDOWN_MSG:

************************************************************

SHUTDOWN_MSG: Shutting down NameNode at hadoop-master/192.168.1.15

************************************************************
```

Hadoop Services

Starting Hadoop services on the Hadoop-Master procedure explains its setup.

```
$ cd $HADOOP_HOME/sbin

$ start-all.sh
```

Addition of a New DataNode in the Hadoop Cluster is as follows:

Networking

Add new nodes to an existing Hadoop cluster with some suitable network configuration. Consider the following network configuration for new node Configuration:

IP address : 192.168.1.103

netmask : 255.255.255.0

hostname : slave3.in

Adding a User and SSH Access

Add a user working under "hadoop" domain and the user must have the access added and password of Hadoop user can be set to anything one wants.

```
useradd hadoop

passwd hadoop
```

To be executed on master

```
mkdir -p $HOME/.ssh

chmod 700 $HOME/.ssh

ssh-keygen -t rsa -P '' -f $HOME/.ssh/id_rsa

cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
```

chmod 644 $HOME/.ssh/authorized_keys

Copy the public key to new slave node in hadoop user $HOME directory

scp $HOME/.ssh/id_rsa.pub hadoop@192.168.1.103:/home/hadoop/

Execution done on slaves

su hadoop ssh -X hadoop@192.168.1.103

Content of public key must be copied into file "$HOME/.ssh/authorized_keys" and then the permission for the same must be changed as per the requirement.

cd $HOME

mkdir -p $HOME/.ssh

chmod 700 $HOME/.ssh

cat id_rsa.pub >>$HOME/.ssh/authorized_keys

chmod 644 $HOME/.ssh/authorized_keys

ssh login must be changed from the master machine. It is possible that the ssh to the new node without a password from the master must be verified.

ssh hadoop@192.168.1.103 or hadoop@slave3

Setting Hostname for New Node

Hostname is setup in the file directory /etc/sysconfig/network

On new slave3 machine

NETWORKING=yes

HOSTNAME=slave3.in

Machine must be restarted again or hostname command should be run under new machine with the corresponding hostname to make changes effectively.

On slave3 node machine:

hostname slave3.in

/etc/hosts must be updated on all machines of the cluster

192.168.1.102 slave3.in slave3

ping the machine with hostnames to check whether it is resolving to IP address.

ping master.in

Start the DataNode on New Node

Datanode daemon should be started manually using $HADOOP_HOME/bin/hadoop-daemon.sh script. Master (NameNode) should correspondingly join the cluster after automatically contacted. New node should be added to the configuration/slaves file in the master server. New node will be identified by script-based commands.

Login to new node

```
su hadoop or ssh -X hadoop@192.168.1.103
```

HDFS is started on a newly added slave node

```
./bin/hadoop-daemon.sh start datanode
```

jps command output must be checked on a new node.

```
$ jps

7141 DataNode

10312 Jps
```

Removing a DataNode

Node can be removed from a cluster while it is running, without any worries of data loss. A decommissioning feature is made available by HDFS which ensures that removing a node is performed securely.

Step 1

Login to master machine so that the user can check Hadoop is being installed.

```
$ su hadoop
```

Step 2

Before starting the cluster an exclude file must be configured where a key named dfs.hosts.exclude should be added to our$HADOOP_HOME/etc/hadoop/hdfs-site.xmlfile.

NameNode's local file system contains a list of machines which are not permitted to connect to HDFS receives full path by this key and the value associated with it as follows.

```
<property>

<name>dfs.hosts.exclude</name><value>/home/hadoop/hadoop-
1.2.1/hdfs_exclude.txt</value><description>>DFS exclude</description>

</property>
```

Step 3

Hosts with respect to decommission are determined.

File reorganization by the hdfs_exclude.txt for each and every machine to be decommissioned which will results in preventing them from connecting to the NameNode.

slave2.in

Step 4

Force configuration reloads.

"$HADOOP_HOME/bin/hadoop dfsadmin -refreshNodes" should be run

```
$ $HADOOP_HOME/bin/hadoop dfsadmin -refreshNodes
```

NameNode will be forced made to re-read its configuration, as this is inclusive for the newly updated 'excludes' file. Nodes will be decommissioned over a period of time intervals, and allowing time for each node's blocks to be replicated onto machines which are scheduled to be active.jps command output should be checked on slave2.in. Once the work is done DataNode process will shutdown automatically.

Step 5

Shutdown nodes.

The decommissioned hardware can be carefully shut down for maintenance purpose after the decommission process has been finished.

```
$ $HADOOP_HOME/bin/hadoop dfsadmin -report
```

Step 6

Excludes are edited again and once the machines have been decommissioned, they are removed from the 'excludes' file. "$HADOOP_HOME/bin/hadoop dfsadmin -refreshNodes" will read the excludes file back into the NameNode.

Data Nodes will rejoin the cluster after the maintenance has been completed, or if additional capacity is needed in the cluster again is being informed.

To run/shutdown tasktracker

```
$ $HADOOP_HOME/bin/hadoop-daemon.sh stop tasktracker
```

```
$ $HADOOP_HOME/bin/hadoop-daemon.sh start tasktracker
```

Add a new node with the following steps

1) Take a new system which gives access to create a new username and password

2) Install the SSH and with master node setup ssh connections

3) Add sshpublic_rsa id key having an authorized keys file

4) Add the new data node hostname, IP address and other informative details in /etc/hosts slaves file192.168.1.102 slave3.in slave3

5) Start the DataNode on the New Node

6) Login to the new node command like suhadoop or Ssh -X hadoop@192.168.1.103

7) Start HDFS of newly added in the slave node by using the following command ./bin/hadoop-daemon.sh start data node

8) Check the output of jps command on a new node.

1.5 SUMMARY

Hadoop Distributed File System is a highly scalable, flexible, fault-tolerant, and reliable system that stores the data across multiple nodes on different servers. It follows a master-slave architecture, where the NameNode acts as a master, and the DataNode as the slave. HDFS Operations are used to access these NameNodes and interact with the data. The files are broken down into blocks where the client can store the data, read, write, and perform various operations by completing the authentication process.

1.6 SAMPLE QUESTIONS

1. What are the different vendor-specific distributions of Hadoop?
2. What are the different Hadoop configuration files?
3. What are the three modes in which Hadoop can run?
4. What are the differences between regular FileSystem and HDFS?
5. Why is HDFS fault-tolerant?
6. Explain the architecture of HDFS.
7. What are the two types of metadata that a NameNode server holds?
8. What is the difference between a federation and high availability?
9. If you have an input file of 350 MB, how many input splits would HDFS create and what would be the size of each input split?
10. How does rack awareness work in HDFS?
11. How can you restart NameNode and all the daemons in Hadoop?
12. Which command will help you find the status of blocks and FileSystem health?

13. What would happen if you store too many small files in a cluster on HDFS?

14. How do you copy data from the local system onto HDFS?

1.7 REFERENCES

1 Tom White, "HADOOP: The definitive Guide" O Reilly 2012, Third Edition

2 Chuck Lam, "Hadoop in Action", Dreamtech Press 2016, First Edition

3 Shiva Achari," Hadoop Essential "PACKT Publications

4 Radha Shankarmani and M. Vijayalakshmi," Big Data Analytics "Wiley Textbook Series, Second Edition

5 Jeffrey Aven,"Apache Spark in 24 Hours" Sam's Publication, First Edition

6 Bill Chambers and MateiZaharia,"Spark: The Definitive Guide: Big Data Processing Made Simple "O'Reilly Media; First edition

7 James D. Miller," Big Data Visualization" PACKT Publications.

8 https://hadoop.apache.org/docs/stable/

9 https://pig.apache.org/

10 https://hive.apache.org/

11 https://spark.apache.org/documentation.html

12 https://help.tableau.com/current/pro/desktop/en-us/default.htm

13 https://www.youtube.com/watch?v=HloGuAzP_H8

❄❄❄❄❄❄❄

EXPERIMENT 1

AIM

Write a simple program for Word Count Using Map Reduce Programming

OBJECTIVE

In MapReduce word count example, we find out the frequency of each word. Here, the role of Mapper is to map the keys to the existing values and the role of Reducer is to aggregate the keys of common values. So, everything is represented in the form of Key-value pair.

THEORY

In Hadoop, Map Reduce is a computation that decomposes large manipulation jobs into individual tasks that can be executed in parallel across a cluster of servers. The results of tasks can be joined together to compute final results.

Pre-requisite

Make sure that Hadoop is installed on your system with the Java SDK, ECLIPSE editor. For all Experiment.

Steps

1. Open Eclipse> File > New > Java Project >(Name it – MRProgramsDemo) > Finish.
2. Right Click > New > Package (Name it - PackageDemo) > Finish.
3. Right Click on Package > New > Class (Name it - WordCount).
4. Add Following Reference Libraries:
    1. Right Click on Project > Build Path> Add External
    2. /usr/lib/hadoop-0.20/**hadoop-core.jar**
    3. Usr/lib/hadoop-0.20/lib/**Commons-cli-1.2.jar**
5. Type the following code:

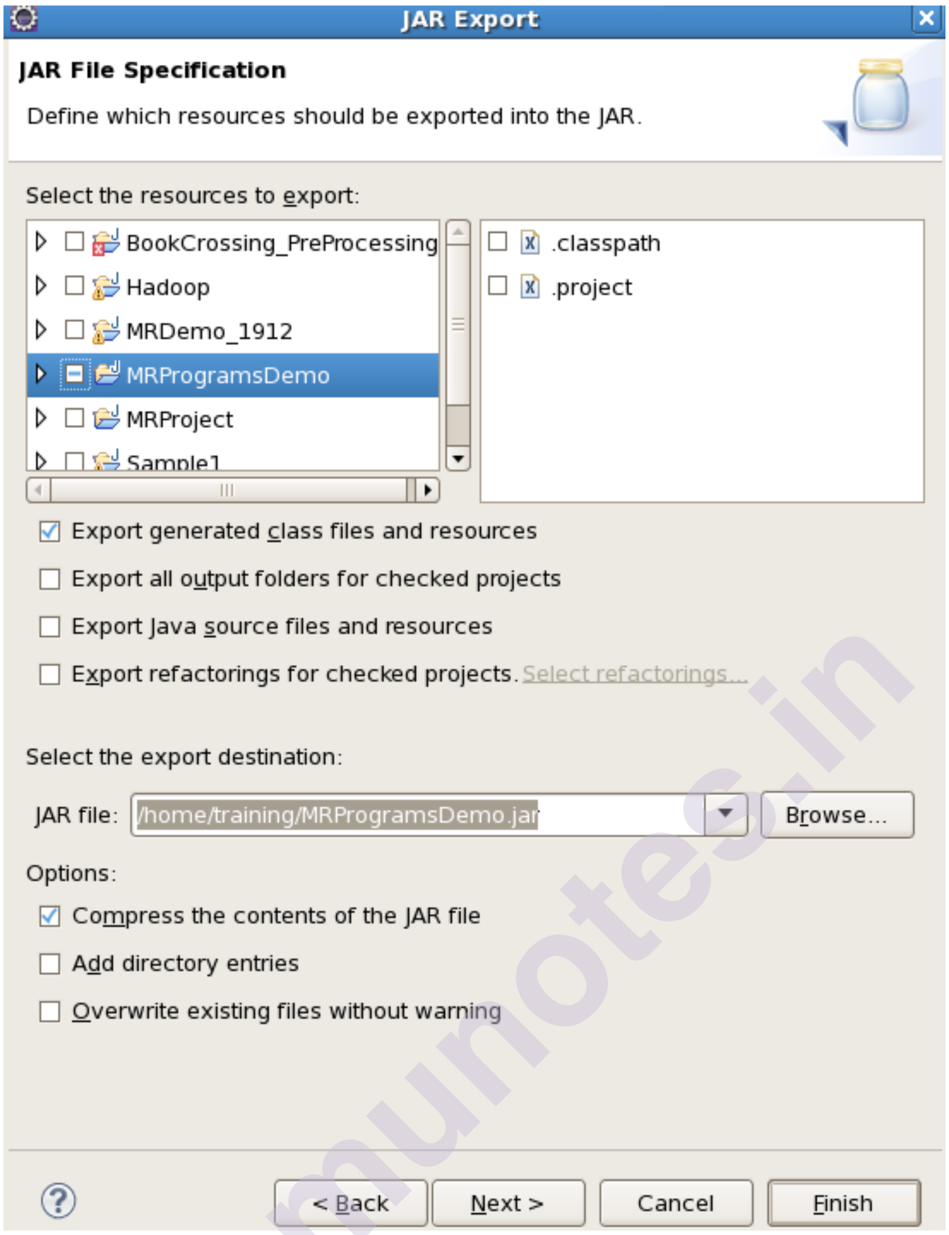
```
package PackageDemo;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;

public class WordCount {
public static void main(String [] args) throws Exception
{
Configuration c=new Configuration();
String[] files=new
GenericOptionsParser(c,args).getRemainingArgs();
Path input=new Path(files[0]);
Path output=new Path(files[1]);
Job j=new Job(c,"wordcount");
j.setJarByClass(WordCount.class);
j.setMapperClass(MapForWordCount.class);
j.setReducerClass(ReduceForWordCount.class);
j.setOutputKeyClass(Text.class);
j.setOutputValueClass(IntWritable.class);
FileInputFormat.addInputPath(j, input);
FileOutputFormat.setOutputPath(j, output);
System.exit(j.waitForCompletion(true)?0:1);
}
public static class MapForWordCount extends
Mapper<LongWritable, Text, Text, IntWritable>{
public void map(LongWritable key, Text value, Context con) throws
IOException, InterruptedException
{
String line = value.toString();
String[] words=line.split(",");
for(String word: words )
{
     Text outputKey = new Text(word.toUpperCase().trim());
  IntWritable outputValue = new IntWritable(1);
  con.write(outputKey, outputValue);
}
}
}


public static class ReduceForWordCount extends Reducer<Text,
IntWritable, Text, IntWritable>
{
public void reduce(Text word, Iterable<IntWritable> values,
Context con) throws IOException, InterruptedException
{
int sum = 0;
   for(IntWritable value : values)
   {
   sum += value.get();
   }
   con.write(word, new IntWritable(sum));
}


}
}
```

The above program consists of three classes:

- Driver class (Public, void, static, or main; this is the entry point).
- The Map class which **extends** the public class Mapper<KEYIN,VALUEIN,KEYOUT,VALUEOUT> and implements the Map function.
- The Reduce class which extends the public class Reducer<KEYIN,VALUEIN,KEYOUT,VALUEOUT> and implements the Reduce function.

6. Make a jar file

Right Click on Project> Export> Select export destination as **Jar File** > next> Finish.



JAR Export

JAR File Specification

Define which resources should be exported into the JAR.

Select the resources to export:

- BookCrossing_PreProcessing
- Hadoop
- MRDemo_1912
- MRProgramsDemo
- MRProject
- Sample1

.classpath
.project

[✓] Export generated class files and resources
[ ] Export all output folders for checked projects
[ ] Export Java source files and resources
[ ] Export refactorings for checked projects. Select refactorings...

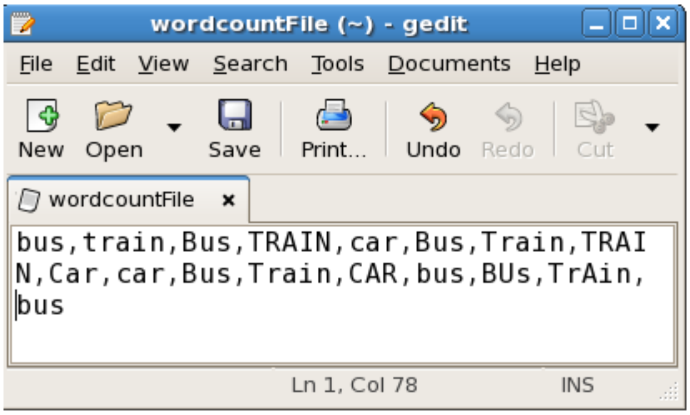
Select the export destination:

JAR file: /home/training/MRProgramsDemo.jar

Options:

[✓] Compress the contents of the JAR file
[ ] Add directory entries
[ ] Overwrite existing files without warning

7. Take a text file and move it into HDFS format:



wordcountFile (~) - gedit

```
bus,train,Bus,TRAIN,car,Bus,Train,TRAI
N,Car,car,Bus,Train,CAR,bus,BUs,TrAin,
bus
```

File Edit View Search Tools Documents Help
New Open Save Print... Undo Redo Cut
wordcountFile
Ln 1, Col 78 INS

To move this into Hadoop directly, open the terminal and enter the following commands:

```
[training@localhost ~]$ hadoop fs -put wordcountFile
wordCountFile
```

9. Open the result:

```
[training@localhost ~]$ hadoop fs -ls MRDir1
```

Found 3 items

```
-rw-r--r--   1 training supergroup          0 2022-02-23 03:36
/user/training/MRDir1/_SUCCESS

drwxr-xr-x   - training supergroup          0 2022-02-23 03:36
/user/training/MRDir1/_logs

-rw-r--r--   1 training supergroup         20 2022-02-23 03:36
/user/training/MRDir1/part-r-00000

[training@localhost ~]$ hadoop fs -cat MRDir1/part-r-00000

BUS     7

CAR     4

TRAIN   6
```

EXPERIMENT 2

AIM :

Write a program in Map Reduce for Union operation.

OBJECTIVE

In MapReduce word count example, we find out the frequency of each word. Here, the role of Mapper is to map the keys to the existing values and the role of Reducer is to aggregate the keys of common values. So, everything is represented in the form of Key-value pair.

THEORY:

In Hadoop, Map Reduce is a computation that decomposes large manipulation jobs into individual tasks that can be executed in parallel across a cluster of servers. The results of tasks can be joined together to compute final results.

Pre-requisite

Make sure that Hadoop is installed on your system with the Java SDK, ECLIPSE editor. For all Experiment.

Steps

1. Open Eclipse> File > New > Java Project >(Name it – MRProgramsDemo) > Finish.
2. Right Click > New > Package (Name it - PackageDemo) > Finish.
3. Right Click on Package > New > Class (Name it - Union).
4. Add Following Reference Libraries:

1. Right Click on Project > Build Path> Add External

1. /usr/lib/hadoop-0.20/**hadoop-core.jar**
2. Usr/lib/hadoop-0.20/lib/**Commons-cli-1.2.jar**

5. Type the following code:

```
import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import java.io.IOException;

public class Union {

        private static Text emptyWord = new Text("");
        public static class Mapper
                extends org.apache.hadoop.mapreduce.Mapper<Object, Text,
Text, Text> {

                public void map(Object key, Text value, Context context
                ) throws IOException, InterruptedException {
                context.write(value, emptyWord);
                }
        }

        public static class Reducer
                extends org.apache.hadoop.mapreduce.Reducer<Text, Text,
Text, Text> {

                public void reduce(Text key, Iterable<Text> _values,
                                  Context context
                ) throws IOException, InterruptedException {
                        context.write(key, key);
                }
        }

        public static void main(String[] args) throws Exception {
                Configuration conf = new Configuration();

                Job job = Job.getInstance(conf, "Word sum");
                job.setJarByClass(Union.class);
                job.setMapperClass(Mapper.class);
                job.setCombinerClass(Reducer.class);
                job.setReducerClass(Reducer.class);
                job.setOutputKeyClass(Text.class);
                job.setOutputValueClass(Text.class);

                Path input = new Path( args[0]);
                Path output = new Path(args[1]);

                FileInputFormat.addInputPath(job, input);
                FileOutputFormat.setOutputPath(job, output);
                System.exit(job.waitForCompletion(true) ? 0 : 1);
        }
}
```

Run Above code similar to first experiment.

EXPERIMENT 3

AIM :

Write a program in Map Reduce for Intersection operation.

OBJECTIVE

In MapReduce word count example, we find out the frequency of each word. Here, the role of Mapper is to map the keys to the existing values and the role of Reducer is to aggregate the keys of common values. So, everything is represented in the form of Key-value pair.

THEORY

In Hadoop, Map Reduce is a computation that decomposes large manipulation jobs into individual tasks that can be executed in parallel across a cluster of servers. The results of tasks can be joined together to compute final results.

Pre-requisite

Make sure that Hadoop is installed on your system with the Java SDK, ECLIPSE editor. For all Experiment.

Steps

1. Open Eclipse> File > New > Java Project >(Name it – MRProgramsDemo) > Finish.
2. Right Click > New > Package (Name it - PackageDemo) > Finish.
3. Right Click on Package > New > Class (Name it - Intersection).
4. Add Following Reference Libraries:
   1. Right Click on Project > Build Path> Add External
      1. /usr/lib/hadoop-0.20/**hadoop-core.jar**
      2. Usr/lib/hadoop-0.20/lib/**Commons-cli-1.2.jar**

Type the following code:

```
import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import java.io.IOException;

public class Intersection {

        public static class Mapper
                extends org.apache.hadoop.mapreduce.Mapper<Object, Text,
Text, IntWritable> {

                private final static IntWritable one = new IntWritable(1);
                private Text word = new Text();

                public void map(Object key, Text value, Context context
                ) throws IOException, InterruptedException {
                context.write(value, one);
                }
        }

        public static class Combiner
                extends org.apache.hadoop.mapreduce.Reducer<Text,
IntWritable, Text, IntWritable> {
                private IntWritable result = new IntWritable();

                public void reduce(Text key, Iterable<IntWritable> values,
                                        Context context
                ) throws IOException, InterruptedException {
                int sum = 0;
                for (IntWritable val : values) {
                        sum ++;
                }
                result.set(sum);
                context.write(key, result);
                }
        }
        public static class Reducer
                extends org.apache.hadoop.mapreduce.Reducer<Text,
IntWritable, Text, Text> {

                public void reduce(Text key, Iterable<IntWritable> values,
                                        Context context
                ) throws IOException, InterruptedException {
                int sum = 0;
                for (IntWritable val : values) {
                        sum ++;
                }
                if (sum > 1) {
                        context.write(key, key);
                }
                }
        }

        public static void main(String[] args) throws Exception {
                Configuration conf = new Configuration();

                Job job = Job.getInstance(conf, "word count");
                job.setJarByClass(Intersection.class);
                job.setMapperClass(Mapper.class);
                job.setCombinerClass(Combiner.class);
                job.setReducerClass(Reducer.class);
                job.setOutputKeyClass(Text.class);
                job.setOutputValueClass(IntWritable.class);
                FileInputFormat.addInputPath(job, new Path(args[0]));
                FileOutputFormat.setOutputPath(job, new Path(args[1]));
                System.exit(job.waitForCompletion(true) ? 0 : 1);
        }
}
```

Run Above code similar to first experiment.

EXPERIMENT 4

AIM :

Write a program in Map Reduce for GroupSum operation.

OBJECTIVE

In MapReduce word count example, we find out the frequency of each word. Here, the role of Mapper is to map the keys to the existing values and the role of Reducer is to aggregate the keys of common values. So, everything is represented in the form of Key-value pair.

THEORY:

In Hadoop, Map Reduce is a computation that decomposes large manipulation jobs into individual tasks that can be executed in parallel across a cluster of servers. The results of tasks can be joined together to compute final results.

Pre-requisite

Make sure that Hadoop is installed on your system with the Java SDK, ECLIPSE editor. For all Experiment.

Steps

1. Open Eclipse> File > New > Java Project >(Name it – MRProgramsDemo) > Finish.
2. Right Click > New > Package (Name it - PackageDemo) > Finish.
3. Right Click on Package > New > Class (Name it - GroupSum).
4. Add Following Reference Libraries:

1. Right Click on Project > Build Path> Add External

1. /usr/lib/hadoop-0.20/**hadoop-core.jar**
2. Usr/lib/hadoop-0.20/lib/**Commons-cli-1.2.jar**

Type the following code:

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import java.io.IOException;

public class GroupSum {

        public static class MyMapper
                extends org.apache.hadoop.mapreduce.Mapper<Object, Text,
Text, IntWritable> {

                public void map(Object key, Text value, Context context
                ) throws IOException, InterruptedException {
                String[] splitArr = value.toString().split("\t");
                if ( splitArr.length > 1 ) {
                        IntWritable intWritable = new
IntWritable(Integer.parseInt(splitArr[1]));
                        context.write(new Text(splitArr[0]), intWritable);
                }
                }
        }

        public static class MyReducer
                extends org.apache.hadoop.mapreduce.Reducer<Text,
IntWritable, Text, IntWritable> {

                public void reduce(Text key, Iterable<IntWritable> values,
                                        Context context
                ) throws IOException, InterruptedException {
                int sum = 0;
                for(IntWritable intWritable : values) {
                        sum += intWritable.get();
                }
                context.write(key, new IntWritable(sum));
                }
        }

        public static void main(String[] args) throws Exception {
                Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "Word sum");
        job.setJarByClass(GroupSum.class);
        job.setMapperClass(MyMapper.class);
        job.setCombinerClass(MyReducer.class);
        job.setReducerClass(MyReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);

        Path input = new Path( args[0]);
        Path output = new Path(args[1]);

        FileInputFormat.addInputPath(job, input);
        FileOutputFormat.setOutputPath(job, output);
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

Run Above code similar to first experiment.

EXPERIMENT 5

AIM

Write a program in Map Reduce for Matrix Multiplication

OBJECTIVE

In MapReduce word count example, we find out the frequency of each word. Here, the role of Mapper is to map the keys to the existing values and the role of Reducer is to aggregate the keys of common values. So, everything is represented in the form of Key-value pair.

THEORY

In Hadoop, Map Reduce is a computation that decomposes large manipulation jobs into individual tasks that can be executed in parallel across a cluster of servers. The results of tasks can be joined together to compute final results.

Pre-requisite

Make sure that Hadoop is installed on your system with the Java SDK, ECLIPSE editor. For all Experiment.

Steps

1. Open Eclipse> File > New > Java Project >(Name it – MRProgramsDemo) > Finish.
2. Right Click > New > Package (Name it - PackageDemo) > Finish.
3. Right Click on Package > New > Class (Name it - MatrixMultiplication).
4. Add Following Reference Libraries:
    1. Right Click on Project > Build Path> Add External
        1. /usr/lib/hadoop-0.20/**hadoop-core.jar**
        2. Usr/lib/hadoop-0.20/lib/**Commons-cli-1.2.jar**

Type the following code:

```
Java.Lang.ArrayIndexOutOfBoundsException: 2
   at MatrixMult$mapper.map(MatrixMult.java:44)
   at Matrix$mapper.map(MatrixMult.java:1)
   at org.apache.hadoop.mapreduce.mapper.run(Mapper.java:145)
   at
org.apache.hadoop.mapred.Maptask.runNewMapper(mapTask.java:793)
   at org.apache.hadoop.mapred.maptask.run(maptask.java:341)
   at org.apache.hadoop.mapred.yarnChild$2.run(YarnChild.java:164)
   at java.security.accesscontroller.dopriviledged(Native Method)
   at javax.security.auth.subject.doAs(Subject.Java:415)
   at
org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInfor
mation.java:1917)
   at org.apache.hadoop.mapred.YarnChild.main(YarnChild.java:158)
```

mapreduce worked when I used a smaller matrix. I will post the code for mapper and reducer below:

```
public class Map
  extends org.apache.hadoop.mapreduce.Mapper<LongWritable, Text,
Text, Text> {
    @Override
    public void map(LongWritable key, Text value, Context context)
            throws IOException, InterruptedException {
        Configuration conf = context.getConfiguration();
        int m = Integer.parseInt(conf.get("m"));
        int p = Integer.parseInt(conf.get("p"));
        String line = value.toString();
        // (M, i, j, Mij);
        String[] indicesAndValue = line.split(",");
        Text outputKey = new Text();
        Text outputValue = new Text();
        if (indicesAndValue[0].equals("M")) {
            for (int k = 0; k < p; k++) {
                outputKey.set(indicesAndValue[1] + "," + k);
                // outputKey.set(i,k);
                outputValue.set(indicesAndValue[0] + "," +
indicesAndValue[2]
                        + "," + indicesAndValue[3]);
                // outputValue.set(M,j,Mij);
                context.write(outputKey, outputValue);
            }
        } else {
            // (N, j, k, Njk);
            for (int i = 0; i < m; i++) {
                outputKey.set(i + "," + indicesAndValue[2]);
                outputValue.set("N," + indicesAndValue[1] + ","
                        + indicesAndValue[3]);
                context.write(outputKey, outputValue);
            }
        }
        }
}
```

```
public class Reduce
  extends org.apache.hadoop.mapreduce.Reducer<Text, Text, Text, Text>
{
        @Override
        public void reduce(Text key, Iterable<Text> values, Context context)
                        throws IOException, InterruptedException {
                String[] value;
                //key=(i,k),
                //Values = [(M/N,j,V/W),..]
                HashMap<Integer, Float> hashA = new HashMap<Integer,
Float>();
                HashMap<Integer, Float> hashB = new HashMap<Integer,
Float>();
                for (Text val : values) {
                        value = val.toString().split(",");
                        if (value[0].equals("M")) {
                                hashA.put(Integer.parseInt(value[1]),
Float.parseFloat(value[2]));
                        } else {
                                hashB.put(Integer.parseInt(value[1]),
Float.parseFloat(value[2]));
                        }
                }
                int n = Integer.parseInt(context.getConfiguration().get("n"));
                float result = 0.0f;
                float m_ij;
                float n_jk;
                for (int j = 0; j < n; j++) {
                        m_ij = hashA.containsKey(j) ? hashA.get(j) : 0.0f;
                        n_jk = hashB.containsKey(j) ? hashB.get(j) : 0.0f;
                        result += m_ij * n_jk;
                }
                if (result != 0.0f) {
                        context.write(null,
                              new Text(key.toString() + "," +
Float.toString(result)));
            }
        }
}



public class MatrixMultiply {

    public static void main(String[] args) throws Exception {
        if (args.length != 2) {
            System.err.println("Usage: MatrixMultiply <in_dir> <out_dir>");
            System.exit(2);
        }
        Configuration conf = new Configuration();
        // M is an m-by-n matrix; N is an n-by-p matrix.
        conf.set("m", "1000");
        conf.set("n", "100");
        conf.set("p", "1000");
        @SuppressWarnings("deprecation")
                Job job = new Job(conf, "MatrixMultiply");
        job.setJarByClass(MatrixMultiply.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);

        job.setMapperClass(Map.class);
        job.setReducerClass(Reduce.class);

        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        job.waitForCompletion(true);
    }
}
```

Run Above code similar to first experiment.

www.geeksforgeeks.org

https://github.com/wosiu/Hadoop-map-reduce-relational-algebra/blob/master/src/main/java/WordCount.java

❄❄❄❄❄❄❄

3

MONGO DB

Unit Structure :

3.0 Objectives

3.1 Introduction

3.1.1 How it Works

3.1.2 How MongoDB is different from RDBMS

3.1.3 Features of MongoDB

3.1.4 Advantages of MongoDB

3.1.5 Disadvantages of MongoDB

3.2 MongoDB Environment

3.2.1 Install MongoDB in Windows

3.3 Creating a Database Using MongoDB

3.4 Creating Collections in MongoDB

3.5 CRUD Document

3.5.1 Create Operations

3.5.2 Read Operations

3.5.3 Update Operations

3.5.4 Delete Operations

3.6 Let’s Sum up

3.7 Website references

3.0 OBJECTIVES

After going through this unit, you will be able to:

- Have a solid understanding of basics of MongoDB
- Learn to run queries against a MongoDB instance
- Manipulate and retrieve data
- Different techniques and algorithms used in association rules.
- Understand the difference between Data Mining and KDD in Databases.

3.1 INTRODUCTION

MongoDB is a free, open-source document-oriented database that can hold a big amount of data while also allowing you to deal with it quickly. Because MongoDB does not store and retrieve data in the form of tables, it is classified as a NoSQL (Not Only SQL) database.

MongoDB is a database developed and managed by MongoDB.Inc, which was first released in February 2009 under the SSPL (Server Side Public License). It also includes certified driver support for all major programming languages, including C, C++, C#, etc. Net, Go, Java, Node.js, Perl, PHP, Python, Motor, Ruby, Scala, Swift, and Mongoid are examples of programming languages. As a result, any of these languages can be used to develop an application. MongoDB is now used by a large number of firms, including Face book and Google.

3.1.1 How it works

Now we'll see how things really work behind the scenes. As we all know, MongoDB is a database server that stores data in databases. In other words, the MongoDB environment provides you with a server on which you may install MongoDB and then construct several databases.

The data is saved in collections and documents thanks to the NoSQL database. As a result, the database, collection, and documents are linked as follows:

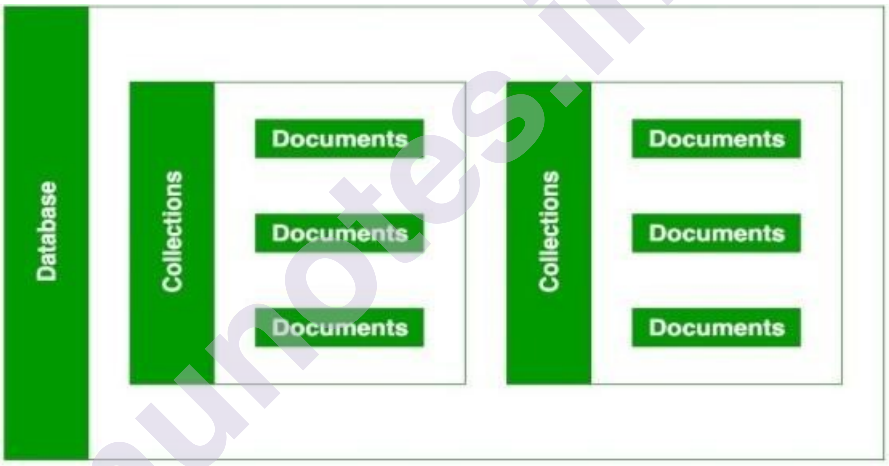


Figure: 4.1 Mongo DB Structure

- Collections exist in the MongoDB database, exactly as tables do in the MYSQL database. You have the option of creating several databases and collections.
- We now have papers within the collection. These documents hold the data we want to store in the MongoDB database, and a single collection can contain several documents. You are schema-less, which means that each document does not have to be identical to the others.
- The fields are used to build the documents. Fields in documents are key-value pairs, similar to columns in a relation database. The fields' values can be of any BSON data type, such as double, string, Boolean, and so on.
- The data in MongoDB is saved in the form of BSON documents. BSON stands for Binary representation of JSON documents in this

context. To put it another way, the MongoDB server translates JSON data into a binary format called as BSON, which can then be stored and searched more effectively.

- It is possible to store nested data in MongoDB documents. When opposed to SQL, layering data allows you to construct complicated relationships between data and store them in the same document, making data working and retrieval extremely efficient. To retrieve data from tables 1 and 2, you must use complicated SQL joins. The BSON document can be up to 16MB in size. In Mongo DB users are allowed to run multiple databases.

3.1.2 How Mongo DB is different from RDBMS

Some major differences between Mongo DB and RDBMS are as follows.

Mongo DB	RDBMS
It is a non-relational and document-oriented database.	It is a relational database
It is suitable for hierarchical data storage.	It is not suitable for hierarchical data storage.
It has a dynamic schema.	It has a predefined schema.
It centers on the CAP theorem (Consistency, Availability, and Partition tolerance).	It centers on ACID properties (Atomicity, Consistency, Isolation, and Durability).
In terms of performance, it is much faster than RDBMS.	In terms of performance, it is slower than MongoDB.

3.1.3 Features of MongoDB

- **Schema-less Database:** MongoDB has a fantastic feature called schema-less database. A schema-less database means that different types of documents can be stored in the same collection. In other words, a single collection in the MongoDB database can hold numerous documents, each of which may have a different amount of fields, content, and size. In contrast to relational databases, it is not required that one document be similar to another. MongoDB gives databases a lot of flexibility because to this amazing feature.

- **Document Oriented**: Unlike RDBMS, MongoDB stores all data in documents rather than tables. Data is kept in fields (key-value pair) rather than rows and columns in these documents, making the data far more flexible than in RDBMS. Each document also has its own unique object id.

- **Indexing:** Every field in the documents in the MongoDB database is indexed with primary and secondary indices, making it easier and faster to get or search data from the pool of data. If the data isn't indexed, the database will have to search each document individually for the query, which takes a long time and is inefficient.

- **Scalability:** Sharding is used by MongoDB to achieve horizontal scalability. Sharding is a method of distributing data across numerous servers. A significant quantity of data is partitioned into data chunks using the shard key, and these data pieces are evenly dispersed across shards located on multiple physical servers. It can also be used to add additional machines to an existing database.

- **Replication:** MongoDB delivers high availability and redundancy through replication, which produces multiple copies of data and sends them to a different server so that if one fails, the data may be retrieved from another.

- **Aggregation:** It enables you to conduct operations on grouped data and obtain a single or computed result. It's analogous to the GROUPBY clause in SQL. Aggregation pipelines map-reduce functions, and single-purpose aggregation methods are among the three types of aggregations available.

- **High Performance:** Due to characteristics such as scalability, indexing, replication, and others, MongoDB has a very high performance and data persistence when compared to other databases.

3.1.4 Advantages of MongoDB

- It's a NoSQL database with no schema. When working with MongoDB, you don't have to worry about designing the database schema.

- The operation of joining is not supported.

- It gives the fields in the papers a lot more flexibility.

- It contains a variety of data.

- It offers excellent availability, scalability, and performance.

- It effectively supports geospatial.

- The data is stored in BSON documents in this document-oriented database.

- It also allows ACID transitions between multiple documents (string from MongoDB 4.0).

- There is no need for SQL injection.

- It is simple to integrate with Hadoop Big Data

3.1.5 Disadvantages of MongoDB

- It stores data in a large amount of memory.
- You may not keep more than 16MB of data in your documents.
- The nesting of data in BSON is likewise limited; you can only nest data up to 100 levels.

3.2 MONGO DB ENVIRONMENT

To get started with MongoDB, you have to install it in your system. You need to find and download the latest version of MongoDB, which will be compatible with your computer system. You can use this (http://www.mongodb.org/downloads) link and follow the instruction to install MongoDB in your PC.

The process of setting up MongoDB in different operating systems is also different, here various installation steps have been mentioned and according to your convenience, you can select it and follow it.

3.2.1 Install Mongo DB in Windows

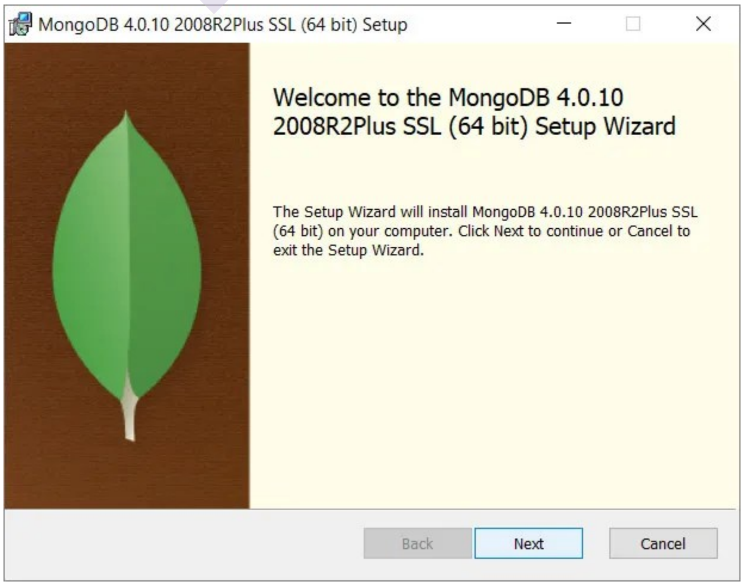
The website of MongoDB provides all the installation instructions, and MongoDB is supported by Windows, Linux as well as Mac OS.

It is to be noted that, MongoDB will not run in Windows XP; so you need to install higher versions of windows to use this database.

Once you visit the link (http://www.mongodb.org/downloads), click the download button.

1. Once the download is complete, double click this setup file to install it. Follow the steps:

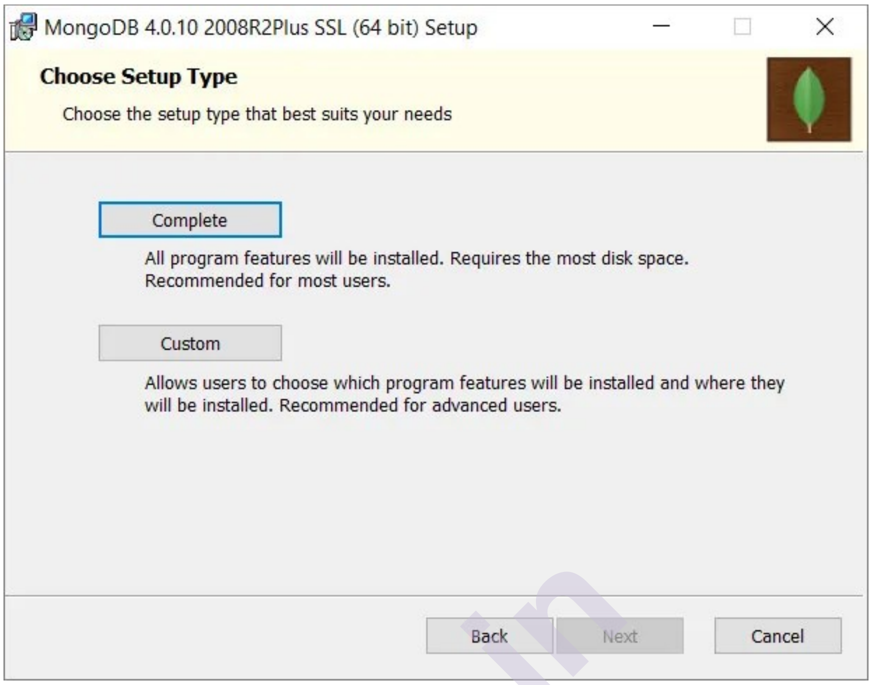
2.


MongoDB 4.0.10 2008R2Plus SSL (64 bit) Setup

Welcome to the MongoDB 4.0.10 2008R2Plus SSL (64 bit) Setup Wizard

The Setup Wizard will install MongoDB 4.0.10 2008R2Plus SSL (64 bit) on your computer. Click Next to continue or Cancel to exit the Setup Wizard.

3. Now, choose Complete to install MongoDB completely.



MongoDB 4.0.10 2008R2Plus SSL (64 bit) Setup

Choose Setup Type

Choose the setup type that best suits your needs

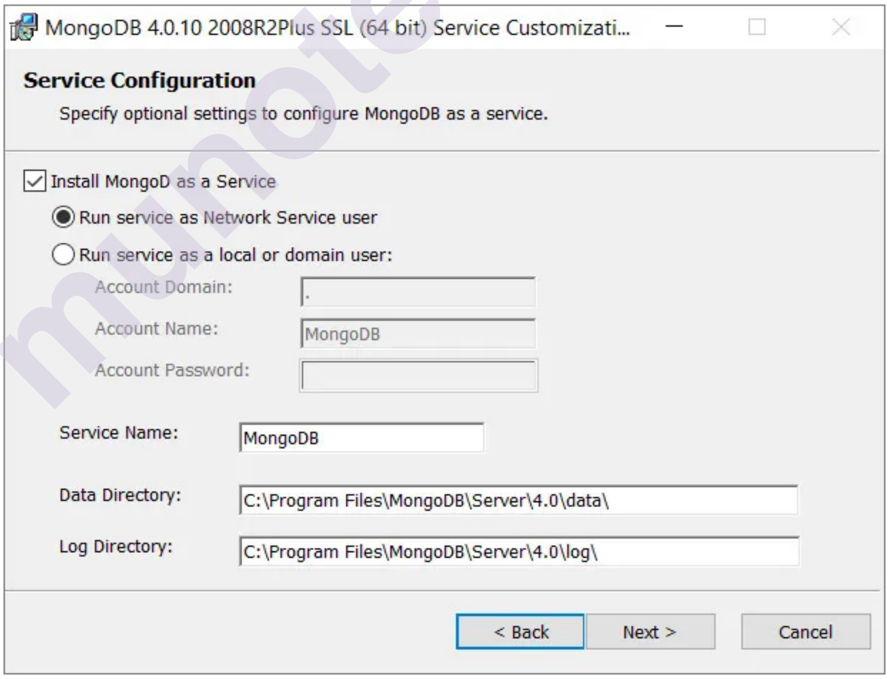
Complete

All program features will be installed. Requires the most disk space. Recommended for most users.

Custom

Allows users to choose which program features will be installed and where they will be installed. Recommended for advanced users.

4. Then, select the radio button "Run services as Network service user."



MongoDB 4.0.10 2008R2Plus SSL (64 bit) Service Customizati...

Service Configuration

Specify optional settings to configure MongoDB as a service.

☑ Install MongoD as a Service

● Run service as Network Service user

○ Run service as a local or domain user:

Account Domain: .

Account Name: MongoDB

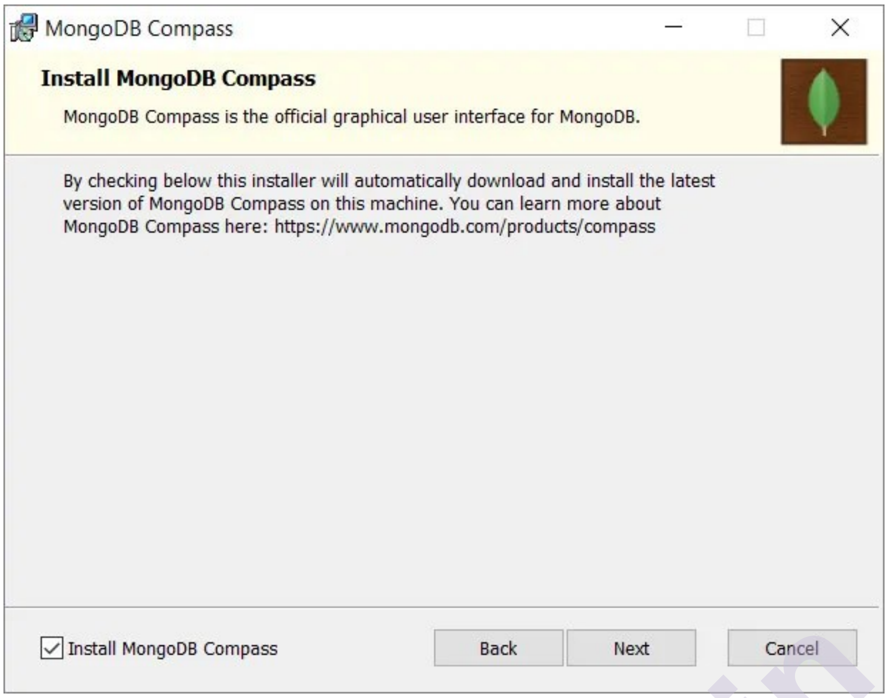
Account Password:

Service Name: MongoDB

Data Directory: C:\Program Files\MongoDB\Server\4.0\data\

Log Directory: C:\Program Files\MongoDB\Server\4.0\log\

5. The setup system will also prompt you to install MongoDB Compass, which is MongoDB official graphical user interface (GUI). You can tick the checkbox to install that as well.



Install MongoDB Compass

MongoDB Compass is the official graphical user interface for MongoDB.

By checking below this installer will automatically download and install the latest version of MongoDB Compass on this machine. You can learn more about MongoDB Compass here: https://www.mongodb.com/products/compass

[✓] Install MongoDB Compass

Once the installation is done completely, you need to start MongoDB and to do so follow the process:

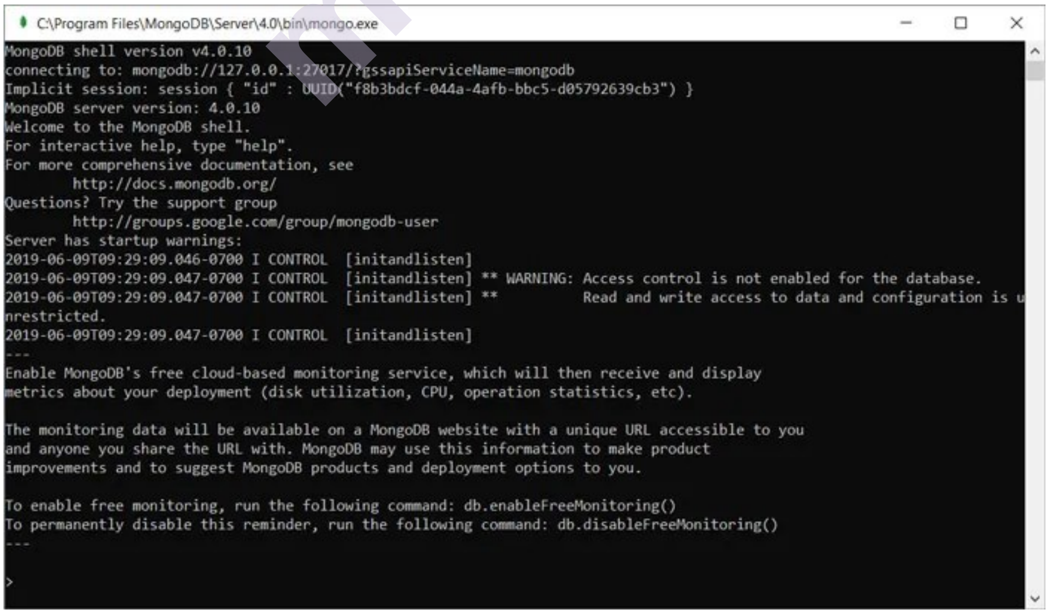
1. Open Command Prompt.

2. Type: C:\Program Files\MongoDB\Server\4.0\bin

3. Now type the command simply: **mongodb** to run the server.

In this way, you can start your MongoDB database. Now, for running MongoDB primary client system, you have to use the command:

C:\Program Files\MongoDB\Server\4.0\bin>**mongo.exe**



```
MongoDB shell version v4.0.10
connecting to: mongodb://127.0.0.1:27017/?gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("f8b3bdcf-044a-4afb-bbc5-d05792639cb3") }
MongoDB server version: 4.0.10
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        http://docs.mongodb.org/
Questions? Try the support group
        http://groups.google.com/group/mongodb-user
Server has startup warnings:
2019-06-09T09:29:09.046-0700 I CONTROL  [initandlisten]
2019-06-09T09:29:09.047-0700 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2019-06-09T09:29:09.047-0700 I CONTROL  [initandlisten] **          Read and write access to data and configuration is u
nrestricted.
2019-06-09T09:29:09.047-0700 I CONTROL  [initandlisten]
---
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---

>
```

3.3 CREATING A DATABASE USING MONGODB

To build a database in MongoDB, first construct a MongoClient object, and then supply a connection URL with the right IP address and the database name. If the database does not already exist, MongoDB will create it and connect to it.

Example: Create a database called "mydb"

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/mydb";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  console.log("Database created!");
  db.close();
});
```

Save the code above in a file called "demo_create_mongo_db.js" and run the file:

Run "demo_create_mongo_db.js"

```
C:\Users\Your Name>node demo_create_mongo_db.js
```

This will give you this result:

Database created!

Note: MongoDB waits until you have created a collection (table), with at least one document (record) before it actually creates the database (and collection).

The use Command

MongoDB use DATABASE_NAME is used to create database. The command will create a new database if it doesn't exist, otherwise it will return the existing database.

Syntax

Basic syntax of use DATABASE statement is as follows –

```
use DATABASE_NAME
```

Example

If you want to use a database with name **<mydb>**, then **use DATABASE** statement would be as follows −

```
>use mydb
switched to db mydb
```

To check your currently selected database, use the command **db**

```
>db
Mydb
```

If you want to check your databases list, use the command **show dbs**.

```
>show dbs
local     0.78125GB
test     0.23012GB
```

Your created database (mydb) is not present in list. To display database, you need to insert at least one document into it.

```
>db.movie.insert({"name":"tutorials point"})
>show dbs
local      0.78125GB
mydb       0.23012GB
test      0.23012GB
```

In MongoDB default database is test. If you didn't create any database, then collections will be stored in test database.

DROP DATABASE

The **dropDatabase()** Method

MongoDB db.dropDatabase() command is used to drop a existing database.

Syntax

Basic syntax of dropDatabase() command is as follows –

```
db.dropDatabase()
```

This will delete the selected database. If you have not selected any database, then it will delete default 'test' database.

Example

First, check the list of available databases by using the command, show dbs.

```
>show dbs

local      0.78125GB

mydb       0.23012GB

test      0.23012GB

>
```

If you want to delete new database <mydb>, then dropDatabase() command would be as follows −

```
>use mydb

switched to db mydb

>db.dropDatabase()

>{ "dropped" : "mydb", "ok" : 1 }

>
```

Now check list of databases.

```
>show dbs

local      0.78125GB

test      0.23012GB

>
```

3.4 CREATING COLLECTIONS IN MONGO DB

The createCollection() Method

MongoDB db.createCollection(name, options) is used to create collection.

Syntax

Basic syntax of createCollection() command is as follows −

```
db.createCollection(name, options)
```

In the command, **name** is name of collection to be created. **Options** is a document and is used to specify configuration of collection.

Parameter	Type	Description
Name	String	Name of the collection to be created
Options	Document	(Optional) Specify options about memory size and indexing

Options parameter is optional, so you need to specify only the name of the collection. Following is the list of options you can use –

Field	Type	Description
capped	Boolean	(Optional) If true, enables a capped collection. Capped collection is a fixed size collection that automatically overwrites its oldest entries when it reaches its maximum size. **If you specify true, you need to specify size parameter also.**
autoIndexId	Boolean	(Optional) If true, automatically create index on _id field.s Default value is false.
size	number	(Optional) Specifies a maximum size in bytes for a capped collection. **If capped is true, then you need to specify this field also.**
max	number	(Optional) Specifies the maximum number of documents allowed in the capped collection.

While inserting the document, MongoDB first checks size field of capped collection, then it checks max field.

Examples

Basic syntax of **createCollection()** method without options is as follows −

```
>use test
switched to db test
>db.createCollection("mycollection")
{ "ok" : 1 }
>
```

You can check the created collection by using the command show collections.

```
>show collections
mycollection
system.indexes
```

The following example shows the syntax of **createCollection()** method with few important options −

```
> db.createCollection("mycol", { capped : true, autoIndexID : true, size :
6142800, max : 10000 } ){
```

```
"ok" : 0,
"errmsg" : "BSON field 'create.autoIndexID' is an unknown field.",
"code" : 40415,
"codeName" : "Location40415"
}
>
```

In MongoDB, you don't need to create collection. MongoDB creates collection automatically, when you insert some document.

```
>db.tutorialspoint.insert({"name" : "tutorialspoint"}),
WriteResult({ "nInserted" : 1 })
>show collections
mycol
mycollection
system.indexes
tutorialspoint
>
```

The drop() Method

MongoDB's **db.collection.drop()** is used to drop a collection from the database.

Syntax

Basic syntax of **drop()** command is as follows −

```
db.COLLECTION_NAME.drop()
```

Example

First, check the available collections into your database **mydb**.

```
>use mydb
switched to db mydb
>show collections
mycol
mycollection
system.indexes
tutorialspoint
>
```

Now drop the collection with the name **mycollection**.

```
>db.mycollection.drop()
true
>
```

Again check the list of collections into database.

```
>show collections
mycol
system.indexes
tutorialspoint
>
```

drop() method will return true, if the selected collection is dropped successfully, otherwise it will return false.

3.5 CRUD DOCUMENT

As we know, we can use MongoDB for a variety of purposes such as building an application (including web and mobile), data analysis, or as an administrator of a MongoDB database. In all of these cases, we must interact with the MongoDB server to perform specific operations such as entering new data into the application, updating data in the application, deleting data from the application, and reading the data of the application.

MongoDB provides a set of simple yet fundamental operations known as CRUD operations that will allow you to quickly interact with the MongoDB server.

C → Create
R → Read
U → Update
D → Delete

3.5.1 **Create Operations** — these operations are used to insert or add new documents to the collection. If a collection does not exist, a new collection will be created in the database. MongoDB provides the following methods for performing and creating operations:

Method	Description
db.collection.insertOne()	It is used to insert a single document in the collection.
db.collection.insertMany()	It is used to insert multiple documents in the collection.

insertOne()

As the namesake, insertOne() allows you to insert one document into the collection. For this example, we're going to work with a collection called

RecordsDB. We can insert a single entry into our collection by calling the insertOne() method on RecordsDB. We then provide the information we want to insert in the form of key-value pairs, establishing the Schema.

Example:

```
db.RecordsDB.insertOne({

   name: "Marsh",

   age: "6 years",

   species: "Dog",

   ownerAddress: "380 W. Fir Ave",

   chipped: true

})
```

If the create operation is successful, a new document is created. The function will return an object where "acknowledged" is "true" and "insertID" is the newly created "ObjectId."

```
> db.RecordsDB.insertOne({

... name: "Marsh",

... age: "6 years",

... species: "Dog",

... ownerAddress: "380 W. Fir Ave",

... chipped: true

... })

{

      "acknowledged" : true,

      "insertedId" : ObjectId("5fd989674e6b9ceb8665c57d")

}
```

insertMany()

It's possible to insert multiple items at one time by calling the insertMany() method on the desired collection. In this case, we pass multiple items into our chosen collection (RecordsDB) and separate them by commas. Within the parentheses, we use brackets to indicate that we are passing in a list of multiple entries. This is commonly referred to as a nested method.

Example:

```
db.RecordsDB.insertMany([{
   name: "Marsh",
   age: "6 years",
   species: "Dog",
   ownerAddress: "380 W. Fir Ave",
   chipped: true},
     {name: "Kitana",
     age: "4 years",
     species: "Cat",
     ownerAddress: "521 E. Cortland",
     chipped: true}])
```

```
db.RecordsDB.insertMany([{ name: "Marsh", age: "6 years", species: "Dog",
ownerAddress: "380 W. Fir Ave", chipped: true}, {name: "Kitana", age: "4 years",
species: "Cat", ownerAddress: "521 E. Cortland", chipped: true}])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("5fd98ea9ce6e8850d88270b4"),
                ObjectId("5fd98ea9ce6e8850d88270b5")
        ]
}
```

3.5.2 Read Operations

You can give specific query filters and criteria to the read operations to indicate the documents you desire. More information on the possible query filters can be found in the MongoDB manual. Query modifiers can also be used to vary the number of results returned.

MongoDB offers two ways to read documents from a collection:

- db.collection.find()
- db.collection.findOne()

find()

In order to get all the documents from a collection, we can simply use the find() method on our chosen collection. Executing just the find() method with no arguments will return all records currently in the collection.

db.RecordsDB.find

findOne()

In order to get one document that satisfies the search criteria, we can simply use the findOne() method on our chosen collection. If multiple documents satisfy the query, this method returns the first document according to the natural order which reflects the order of documents on the disk. If no documents satisfy the search criteria, the function returns null. The function takes the following form of syntax.

db.{collection}.findOne({query}, {projection})

3.5.3 Update Operations

Update operations, like create operations, work on a single collection and are atomic at the document level. Filters and criteria are used to choose the documents to be updated during an update procedure.

You should be cautious while altering documents since alterations are permanent and cannot be reversed. This also applies to remove operations.

There are three techniques for updating documents in MongoDB CRUD**:**

- db.collection.updateOne()
- db.collection.updateMany()
- db.collection.replaceOne()

updateOne()

With an update procedure, we may edit a single document and update an existing record. To do this, we use the updateOne() function on a specified collection, in this case "RecordsDB." To update a document, we pass two arguments to the method: an update filter and an update action.

The update filter specifies which items should be updated, and the update action specifies how those items should be updated. We start with the update filter. Then we utilise the "$set" key and supply the values for the fields we wish to change. This function updates the first record that matches the specified filter.

updateMany()

updateMany() allows us to update multiple items by passing in a list of items, just as we did when inserting multiple items. This update operation uses the same syntax for updating a single document.

replaceOne()

The replaceOne() method is used to replace a single document in the specified collection. replaceOne() replaces the entire document, meaning fields in the old document not contained in the new will be lost.

3.5.4 Delete Operations

Delete operations, like update and create operations, work on a single collection. For a single document, delete actions are similarly atomic. You can provide delete actions with filters and criteria to indicate which documents from a collection you want to delete. The filter options use the same syntax as the read operations.

MongoDB provides two ways for removing records from a collection:

- db.collection.deleteOne()
- db.collection.deleteMany()

deleteOne()

deleteOne() is used to remove a document from a specified collection on the MongoDB server. A filter criterion is used to specify the item to delete. It deletes the first record that matches the provided filter.

deleteMany()

deleteMany() is a method used to delete multiple documents from a desired collection with a single delete operation. A list is passed into the method and the individual items are defined with filter criteria as in deleteOne()

3.6 Let us sum up

- MongoDB is an open-source database that uses a document-oriented data model and a non-structured query language
- It is one of the most powerful NoSQL systems and databases around, today.
- The data model that MongoDB follows is a highly elastic one that lets you combine and store data of multivariate types without having to compromise on the powerful indexing options, data access, and validation rules.
- A group of database documents can be called a collection. The RDBMS equivalent to a collection is a table. The entire collection exists within a single database. There are no schemas when it comes

to collections. Inside the collection, various documents can have varied fields, but mostly the documents within a collection are meant for the same purpose or for serving the same end goal.

- A set of key-value pairs can be designated as a document. Documents are associated with dynamic schemas. The benefit of having dynamic schemas is that a document in a single collection does not have to possess the same structure or fields. Also, the common fields in a collection document can have varied types of data.

3.7 WEBSITE REFERENCES

1. https://www.mongodb.com
2. https://www.tutorialspoint.com
3. https://www.w3schools.com
4. https://www.educba.com

❄❄❄❄❄❄❄

4

HIVE

Unit Structure :

4.0 Objectives
4.1 Introduction
4.2 Summary
4.3 References
4.4 Unit End Exercises

4.0 OBJECTIVE

Hive **allows users to read, write, and manage petabytes of data using SQL**. Hive is built on top of Apache Hadoop, which is an open-source framework used to efficiently store and process large datasets. As a result, Hive is closely integrated with Hadoop, and is designed to work quickly on petabytes of data.

4.1 INTRODUCTION

Hive is a data warehouse system which is used to analyze structured data. It is built on the top of Hadoop. It was developed by Facebook.

Hive provides the functionality of reading, writing, and managing large datasets residing in distributed storage. It runs SQL like queries called HQL (Hive query language) which gets internally converted to MapReduce jobs.

Using Hive, we can skip the requirement of the traditional approach of writing complex MapReduce programs. Hive supports Data Definition Language (DDL), Data Manipulation Language (DML), and User Defined Functions (UDF).

Features of Hive

These are the following features of Hive:

- Hive is fast and scalable.
- It provides SQL-like queries (i.e., HQL) that are implicitly transformed to MapReduce or Spark jobs.
- It is capable of analyzing large datasets stored in HDFS.
- It allows different storage types such as plain text, RCFile, and HBase.
- It uses indexing to accelerate queries.
- It can operate on compressed data stored in the Hadoop ecosystem.

- It supports user-defined functions (UDFs) where user can provide its functionality.

Limitations of Hive

- Hive is not capable of handling real-time data.
- It is not designed for online transaction processing.
- Hive queries contain high latency.
- Hive is a database technology that can define databases and tables to analyze structured data. The theme for structured data analysis is to store the data in a tabular manner, and pass queries to analyze it. This chapter explains how to create Hive database. Hive contains a default database named **default**.

Create Database Statement

Create Database is a statement used to create a database in Hive. A database in Hive is a **namespace** or a collection of tables. The **syntax** for this statement is as follows:

```
o    CREATE DATABASE|SCHEMA [IF NOT EXISTS] <database
name>
```

Here, IF NOT EXISTS is an optional clause, which notifies the user that a database with the same name already exists. We can use SCHEMA in place of DATABASE in this command. The following query is executed to create a database named **userdb**:

```
o    hive> CREATE DATABASE [IF NOT EXISTS] userdb;
```

or

```
o    hive> CREATE SCHEMA userdb;
```

The following query is used to verify a databases list:

```
o    hive> SHOW DATABASES;

o    default

o    userdb
```

JDBC Program

The JDBC program to create a database is given below.

```
o       import java.sql.SQLException;
o       import java.sql.Connection;
o       import java.sql.ResultSet;
o       import java.sql.Statement;
o       import java.sql.DriverManager;
o
o       public class HiveCreateDb {
o         private static String driverName =
"org.apache.hadoop.hive.jdbc.HiveDriver";
o
o         public static void main(String[] args) throws SQLException {
o           // Register driver and create driver instance
o
o           Class.forName(driverName);
o           // get connection
o
o           Connection con =
DriverManager.getConnection("jdbc:hive://localhost:10000/default", "",
"");
o           Statement stmt = con.createStatement();
o
o           stmt.executeQuery("CREATE DATABASE userdb");
o           System.out.println("Database userdb created successfully.");
o
o           con.close();
o         }
o       }
```

Save the program in a file named HiveCreateDb.java. The following commands are used to compile and execute this program.

```
o     $ javac HiveCreateDb.java
o     $ java HiveCreateDb
```

Output:

```
Database userdb created successfully.
```

Hive - Partitioning

Hive organizes tables into partitions. It is a way of dividing a table into related parts based on the values of partitioned columns such as date, city, and department. Using partition, it is easy to query a portion of the data.

Tables or partitions are sub-divided into **buckets,** to provide extra structure to the data that may be used for more efficient querying. Bucketing works based on the value of hash function of some column of a table.

For example, a table named **Tab1** contains employee data such as id, name, dept, and yoj (i.e., year of joining). Suppose you need to retrieve the details of all employees who joined in 2012. A query searches the whole table for the required information. However, if you partition the employee data with the year and store it in a separate file, it reduces the query processing time. The following example shows how to partition a file and its data:

The following file contains employeedata table.

```
/tab1/employeedata/file1

id, name, dept, yoj

1, gopal, TP, 2012

2, kiran, HR, 2012

3, kaleel,SC, 2013

4, Prasanth, SC, 2013
```

The above data is partitioned into two files using year.

```
/tab1/employeedata/2012/file2

1, gopal, TP, 2012

2, kiran, HR, 2012

/tab1/employeedata/2013/file3

3, kaleel,SC, 2013

4, Prasanth, SC, 2013
```

Adding a Partition

We can add partitions to a table by altering the table. Let us assume we have a table called **employee** with fields such as Id, Name, Salary, Designation, Dept, and yoj.

Syntax:

```
ALTER TABLE table_name ADD [IF NOT EXISTS] PARTITION
partition_spec

[LOCATION 'location1'] partition_spec [LOCATION 'location2'] ...;



partition_spec:

: (p_column = p_col_value, p_column = p_col_value, ...)
```

The following query is used to add a partition to the employee table.

```
hive> ALTER TABLE employee

> ADD PARTITION (year='2012')

> location '/2012/part2012';
```

Renaming a Partition

The syntax of this command is as follows.

```
ALTER TABLE table_name PARTITION partition_spec RENAME TO
PARTITION partition_spec;
```

The following query is used to rename a partition:

```
hive> ALTER TABLE employee PARTITION (year='1203')

  > RENAME TO PARTITION (Yoj='1203');
```

Dropping a Partition

The following syntax is used to drop a partition:

```
ALTER TABLE table_name DROP [IF EXISTS] PARTITION
partition_spec, PARTITION partition_spec,...;
```

The following query is used to drop a partition:

```
hive> ALTER TABLE employee DROP [IF EXISTS]

  > PARTITION (year='1203');
```

4.2 SUMMARY

Hive - Built-in Functions

Here we are explaining the built-in functions available in Hive. The functions look quite similar to SQL functions, except for their usage.

Built-In Functions

Hive supports the following built-in functions:

Return Type	Signature	Description
BIGINT	round(double a)	It returns the rounded BIGINT value of the double.
BIGINT	floor(double a)	It returns the maximum BIGINT value that is equal or less than the double.
BIGINT	ceil(double a)	It returns the minimum BIGINT value that is equal or greater than the double.
Double	rand(), rand(int seed)	It returns a random number that changes from row to row.
String	concat(string A, string B,...)	It returns the string resulting from concatenating B after A.
String	substr(string A, int start)	It returns the substring of A starting from start position till the end of string A.
String	substr(string A, int start, int length)	It returns the substring of A starting from start position with the given length.
String	upper(string A)	It returns the string resulting from converting all characters of A to upper case.

String	ucase(string A)	Same as above.
String	lower(string A)	It returns the string resulting from converting all characters of B to lower case.
String	lcase(string A)	Same as above.
String	trim(string A)	It returns the string resulting from trimming spaces from both ends of A.
String	ltrim(string A)	It returns the string resulting from trimming spaces from the beginning (left hand side) of A.
String	rtrim(string A)	rtrim(string A) It returns the string resulting from trimming spaces from the end (right hand side) of A.
String	regexp_replace(string A, string B, string C)	It returns the string resulting from replacing all substrings in B that match the Java regular expression syntax with C.
Int	size(Map<K.V>)	It returns the number of elements in the map type.
Int	size(Array<T>)	It returns the number of elements in the array type.
value of <type>	cast(<expr> as <type>)	It converts the results of the expression expr to <type> e.g. cast('1' as BIGINT) converts the string '1' to it integral representation. A NULL is returned if the conversion does not succeed.
String	from_unixtime(int unixtime)	convert the number of seconds from Unix epoch (1970-01-01 00:00:00 UTC) to a string

		representing the timestamp of that moment in the current system time zone in the format of "1970-01-01 00:00:00"
String	to_date(string timestamp)	It returns the date part of a timestamp string: to_date("1970-01-01 00:00:00") = "1970-01-01"
Int	year(string date)	It returns the year part of a date or a timestamp string: year("1970-01-01 00:00:00") = 1970, year("1970-01-01") = 1970
Int	month(string date)	It returns the month part of a date or a timestamp string: month("1970-11-01 00:00:00") = 11, month("1970-11-01") = 11
Int	day(string date)	It returns the day part of a date or a timestamp string: day("1970-11-01 00:00:00") = 1, day("1970-11-01") = 1
String	get_json_object(string json_string, string path)	It extracts json object from a json string based on json path specified, and returns json string of the extracted json object. It returns NULL if the input json string is invalid.

Example

The following queries demonstrate some built-in functions:

round() function

```
hive> SELECT round(2.6) from temp;
```

On successful execution of query, you get to see the following response:

floor() function

```
hive> SELECT floor(2.6) from temp;
```

On successful execution of the query, you get to see the following response:

ceil() function

```
hive> SELECT ceil(2.6) from temp;
```

On successful execution of the query, you get to see the following response:

3.0

Aggregate Functions

Hive supports the following built-in **aggregate functions**. The usage of these functions is as same as the SQL aggregate functions.

Return Type	Signature	Description
BIGINT	count(*), count(expr),	count(*) - Returns the total number of retrieved rows.
DOUBLE	sum(col), sum(DISTINCT col)	It returns the sum of the elements in the group or the sum of the distinct values of the column in the group.
DOUBLE	avg(col), avg(DISTINCT col)	It returns the average of the elements in the group or the average of the distinct values of the column in the group.
DOUBLE	min(col)	It returns the minimum value of the column in the group.
DOUBLE	max(col)	It returns the maximum value of the column in the group.

Hive - Built-in Operators

Here we are explaining the operators available in Hive. There are five types of operators in Hive:

- Relational Operators
- Arithmetic Operators
- Logical Operators
- Complex Operators

Relational Operators

These operators are used to compare two operands. The following table describes the relational operators available in Hive:

Operator	Operand	Description
A = B	all primitive types	TRUE if expression A is equivalent to expression B otherwise FALSE.
A != B	all primitive types	TRUE if expression A is not equivalent to expression B otherwise FALSE.
A < B	all primitive types	TRUE if expression A is less than expression B otherwise FALSE.
A <= B	all primitive types	TRUE if expression A is less than or equal to expression B otherwise FALSE.
A > B	all primitive types	TRUE if expression A is greater than expression B otherwise FALSE.
A >= B	all primitive types	TRUE if expression A is greater than or equal to expression B otherwise FALSE.
A IS NULL	all types	TRUE if expression A evaluates to NULL otherwise FALSE.
A IS NOT NULL	all types	FALSE if expression A evaluates to NULL otherwise TRUE.
A LIKE B	Strings	TRUE if string pattern A matches to B otherwise FALSE.
A RLIKE B	Strings	NULL if A or B is NULL, TRUE if any substring of A matches the Java regular expression B , otherwise FALSE.
A REGEXP B	Strings	Same as RLIKE.

Example

Let us assume the **employee** table is composed of fields named Id, Name, Salary, Designation, and Dept as shown below. Generate a query to retrieve the employee details whose Id is 1205.

```
+-----+--------------+--------+--------------------------+------+
| Id | Name | Salary | Designation | Dept |
+-----+--------------+-----------------------------------+------+
|1201 | Gopal | 45000 | Technical manager | TP |
|1202 | Manisha | 45000 | Proofreader | PR |
|1203 | Masthanvali | 40000 | Technical writer | TP |
|1204 | Krian | 40000 | Hr Admin | HR |
|1205 | Kranthi | 30000 | Op Admin | Admin|
+-----+--------------+--------+--------------------------+------+
```

The following query is executed to retrieve the employee details using the above table:

```
hive> SELECT * FROM employee WHERE Id=1205;
```

On successful execution of query, you get to see the following response:

```
+-----+-----------+-----------+----------------------------------+
| ID | Name | Salary | Designation | Dept |
+-----+---------------+-------+----------------------------------+
|1205 | Kranthi | 30000 | Op Admin | Admin |
+-----+-----------+-----------+----------------------------------+
```

The following query is executed to retrieve the employee details whose salary is more than or equal to Rs 40000.

```
hive> SELECT * FROM employee WHERE Salary>=40000;
```

On successful execution of query, you get to see the following response:

```
+-----+------------+--------+----------------------------+------+
| ID | Name | Salary | Designation | Dept |
+-----+------------+--------+----------------------------+------+
|1201 | Gopal | 45000 | Technical manager | TP |
|1202 | Manisha | 45000 | Proofreader | PR |
|1203 | Masthanvali| 40000 | Technical writer | TP |
|1204 | Krian | 40000 | Hr Admin | HR |
+-----+------------+--------+----------------------------+------+
```

Arithmetic Operators

These operators support various common arithmetic operations on the operands. All of them return number types. The following table describes the arithmetic operators available in Hive:

Operators	Operand	Description
A + B	all number types	Gives the result of adding A and B.
A - B	all number types	Gives the result of subtracting B from A.
A * B	all number types	Gives the result of multiplying A and B.
A / B	all number types	Gives the result of dividing B from A.
A % B	all number types	Gives the reminder resulting from dividing A by B.
A & B	all number types	Gives the result of bitwise AND of A and B.
A | B	all number types	Gives the result of bitwise OR of A and B.
A ^ B	all number types	Gives the result of bitwise XOR of A and B.
~A	all number types	Gives the result of bitwise NOT of A.

Example

The following query adds two numbers, 20 and 30.

```
hive> SELECT 20+30 ADD FROM temp;
```

On successful execution of the query, you get to see the following response:

```
+--------+
| ADD |
+--------+
| 50 |
+--------+
```

Logical Operators

The operators are logical expressions. All of them return either TRUE or FALSE.

Operators	Operands	Description
A AND B	boolean	TRUE if both A and B are TRUE, otherwise FALSE.
A && B	boolean	Same as A AND B.
A OR B	boolean	TRUE if either A or B or both are TRUE, otherwise FALSE.
A || B	boolean	Same as A OR B.
NOT A	boolean	TRUE if A is FALSE, otherwise FALSE.
!A	boolean	Same as NOT A.

Example

The following query is used to retrieve employee details whose Department is TP and Salary is more than Rs 40000.

```
hive> SELECT * FROM employee WHERE Salary>40000 && Dept=TP;
```

On successful execution of the query, you get to see the following response:

```
+------+--------------+-------------+-------------------+--------+

| ID | Name | Salary | Designation | Dept |

+------+--------------+-------------+-------------------+--------+

|1201 | Gopal | 45000 | Technical manager | TP |

+------+--------------+-------------+-------------------+--------+
```

Complex Operators

These operators provide an expression to access the elements of Complex Types.

Operator	Operand	Description
A[n]	A is an Array and n is an int	It returns the nth element in the array A. The first element has index 0.
M[key]	M is a Map<K, V> and key has type K	It returns the value corresponding to the key in the map.
S.x	S is a struct	It returns the x field of S.

Hive - View and Indexes

Views are generated based on user requirements. You can save any result set data as a view. The usage of view in Hive is same as that of the view in SQL. It is a standard RDBMS concept. We can execute all DML operations on a view.

Creating a View

You can create a view at the time of executing a SELECT statement. The syntax is as follows:

```
CREATE VIEW [IF NOT EXISTS] view_name [(column_name
[COMMENT column_comment], ...) ]

[COMMENT table_comment]

AS SELECT ...
```

Example

Let us take an example for view. Assume employee table as given below, with the fields Id, Name, Salary, Designation, and Dept. Generate a query to retrieve the employee details who earn a salary of more than Rs 30000. We store the result in a view named **emp_30000.**

```
+------+--------------+-------------+-------------------+--------+
| ID | Name | Salary | Designation | Dept |
+------+--------------+-------------+-------------------+--------+
|1201 | Gopal | 45000 | Technical manager | TP |
|1202 | Manisha | 45000 | Proofreader | PR |
|1203 | Masthanvali | 40000 | Technical writer | TP |
|1204 | Krian | 40000 | Hr Admin | HR |
|1205 | Kranthi | 30000 | Op Admin | Admin |
+------+--------------+-------------+-------------------+--------+
```

The following query retrieves the employee details using the above scenario:

```
hive> CREATE VIEW emp_30000 AS

SELECT * FROM employee

WHERE salary>30000;
```

Dropping a View

Use the following syntax to drop a view:

```
DROP VIEW view_name
```

The following query drops a view named as emp_30000:

```
hive> DROP VIEW emp_30000;
```

Creating an Index

An Index is nothing but a pointer on a particular column of a table. Creating an index means creating a pointer on a particular column of a table. Its syntax is as follows:

```
CREATE INDEX index_name

ON TABLE base_table_name (col_name, ...)

AS 'index.handler.class.name'

[WITH DEFERRED REBUILD]

[IDXPROPERTIES (property_name=property_value, ...)]

[IN TABLE index_table_name]

[PARTITIONED BY (col_name, ...)]

[

[ ROW FORMAT ...] STORED AS ...

| STORED BY ...

]

[LOCATION hdfs_path]

[TBLPROPERTIES (...)]
```

Example

Let us take an example for index. Use the same employee table that we have used earlier with the fields Id, Name, Salary, Designation, and Dept. Create an index named index_salary on the salary column of the employee table.

The following query creates an index:

```
hive> CREATE INDEX inedx_salary ON TABLE employee(salary)

AS 'org.apache.hadoop.hive.ql.index.compact.CompactIndexHandler';
```

It is a pointer to the salary column. If the column is modified, the changes are stored using an index value.

Dropping an Index

The following syntax is used to drop an index:

```
DROP INDEX <index_name> ON <table_name>
```

The following query drops an index named index_salary:

```
hive> DROP INDEX index_salary ON employee;
```

4.3 REFERENCES

- "The Visual Display of Quantitative Information" by Edward R. ...
- "Storytelling With Data: A Data Visualization Guide for Business Professionals" by Cole Nussbaumer Knaflic.
- "Data Visualization – A Practical Introduction" by Kieran Healy.

4.4 UNIT END EXERCISES

Write a program to perform Table Operations such as **Creation, Altering, and Dropping** tables in Hive

❄❄❄❄❄❄❄

5

PIG

Unit Structure :

5.0 Objectives

5.1 Introduction

5.2 Summary

5.3 References

5.4 Unit End Exercises

5.0 OBJECTIVE

Pig is an open-source high level data flow system. It provides a simple language called Pig Latin, for queries and data manipulation, which are then compiled in to MapReduce jobs that run on Hadoop.

5.1 INTRODUCTION

Pig is important as companies like Yahoo, Google and Microsoft are collecting huge amounts of data sets in the form of click streams, search logs and web crawls. Pig is also used in some form of ad-hoc processing and analysis of all the information.

Why Do you Need Pig?

- It's easy to learn, especially if you're familiar with SQL.
- Pig's multi-query approach reduces the number of times data is scanned. This means 1/20th the lines of code and 1/16th the development time when compared to writing raw MapReduce.
- Performance of Pig is in par with raw MapReduce
- Pig provides data operations like filters, joins, ordering, etc. and nested data types like tuples, bags, and maps, that are missing from MapReduce.
- Pig Latin is easy to write and read.

Why was Pig Created?

Pig was originally developed by Yahoo in 2006, for researchers to have an ad-hoc way of creating and executing MapReduce jobs on very large data sets. It was created to reduce the development time through its multi-query approach. Pig is also created for professionals from non-Java background, to make their job easier.

Where Should Pig be Used?

Pig can be used under following scenarios:

When data loads are time sensitive.

- When processing various data sources.
- When analytical insights are required through sampling.

Pig Latin – Basics

Pig Latin is the language used to analyze data in Hadoop using Apache Pig. In this chapter, we are going to discuss the basics of Pig Latin such as Pig Latin statements, data types, general and relational operators, and Pig Latin UDF's.

Pig Latin – Data Model

As discussed in the previous chapters, the data model of Pig is fully nested. A **Relation** is the outermost structure of the Pig Latin data model. And it is a **bag** where −

- A bag is a collection of tuples.
- A tuple is an ordered set of fields.
- A field is a piece of data.

Pig Latin – Statemets

While processing data using Pig Latin, **statements** are the basic constructs.

- These statements work with **relations**. They include **expressions** and **schemas**.
- Every statement ends with a semicolon (;).
- We will perform various operations using operators provided by Pig Latin, through statements.
- Except LOAD and STORE, while performing all other operations, Pig Latin statements take a relation as input and produce another relation as output.
- As soon as you enter a **Load** statement in the Grunt shell, its semantic checking will be carried out. To see the contents of the schema, you need to use the **Dump** operator. Only after performing the **dump** operation, the MapReduce job for loading the data into the file system will be carried out.

Example

Given below is a Pig Latin statement, which loads data to Apache Pig.

```
grunt> Student_data = LOAD 'student_data.txt' USING PigStorage(',')as
   ( id:int, firstname:chararray, lastname:chararray, phone:chararray,
city:chararray );
```

Pig Latin – Data types

Given below table describes the Pig Latin data types.

S.N.	Data Type	Description & Example
1	int	Represents a signed 32-bit integer. **Example** : 8
2	long	Represents a signed 64-bit integer. **Example** : 5L
3	float	Represents a signed 32-bit floating point. **Example** : 5.5F
4	double	Represents a 64-bit floating point. **Example** : 10.5
5	chararray	Represents a character array (string) in Unicode UTF-8 format. **Example** : ‘tutorials point’
6	Bytearray	Represents a Byte array (blob).
7	Boolean	Represents a Boolean value. **Example** : true/ false.
8	Datetime	Represents a date-time. **Example** : 1970-01-01T00:00:00.000+00:00
9	Biginteger	Represents a Java BigInteger. **Example** : 60708090709
10	Bigdecimal	Represents a Java BigDecimal **Example** : 185.98376256272893883

Complex Types		
11	Tuple	A tuple is an ordered set of fields. **Example** : (raja, 30)
12	Bag	A bag is a collection of tuples. **Example** : {(raju,30),(Mohhammad,45)}
13	Map	A Map is a set of key-value pairs. **Example** : ['name'#'Raju', 'age'#30]

Null Values

Values for all the above data types can be NULL. Apache Pig treats null values in a similar way as SQL does.

A null can be an unknown value or a non-existent value. It is used as a placeholder for optional values. These nulls can occur naturally or can be the result of an operation.

Pig Latin – Arithmetic Operators

The following table describes the arithmetic operators of Pig Latin. Suppose a = 10 and b = 20.

Operator	Description	Example
+	**Addition** − Adds values on either side of the operator	a + b will give 30
−	**Subtraction** − Subtracts right hand operand from left hand operand	a − b will give −10
*	**Multiplication** − Multiplies values on either side of the operator	a * b will give 200
/	**Division** − Divides left hand operand by right hand operand	b / a will give 2
%	**Modulus** − Divides left hand operand by right hand operand and returns remainder	b % a will give 0

? :	**Bincond** − Evaluates the Boolean operators. It has three operands as shown below. variable **x** = (expression) ? **value1** *if true* : **value2** *if false*.	b = (a == 1)? 20: 30; if a = 1 the value of b is 20. if a!=1 the value of b is 30.

CASE
WHEN
THEN
ELSE END
	Case − The case operator is equivalent to nested bincond operator.	CASE f2 % 2
		WHEN 0 THEN 'even'
		WHEN 1 THEN 'odd'
		END

Pig Latin – Comparison Operators

The following table describes the comparison operators of Pig Latin.

Operator	Description	Example
==	**Equal** − Checks if the values of two operands are equal or not; if yes, then the condition becomes true.	(a = b) is not true
!=	**Not Equal** − Checks if the values of two operands are equal or not. If the values are not equal, then condition becomes true.	(a != b) is true.
>	**Greater than** − Checks if the value of the left operand is greater than the value of the right operand. If yes, then the condition becomes true.	(a > b) is not true.
<	**Less than** − Checks if the value of the left operand is less than the value of the right operand. If yes, then the condition becomes true.	(a < b) is true.
>=	**Greater than or equal to** − Checks if the value of the left operand is greater than or equal to the value of the right operand. If yes, then the condition becomes true.	(a >= b) is not true.

<=	**Less than or equal to** − Checks if the value of the left operand is less than or equal to the value of the right operand. If yes, then the condition becomes true.	(a <= b) is true.
matches	**Pattern matching** − Checks whether the string in the left-hand side matches with the constant in the right-hand side.	f1 matches '.*tutorial.*'

Pig Latin – Type Construction Operators

The following table describes the Type construction operators of Pig Latin.

Operator	Description	Example
()	**Tuple constructor operator** − This operator is used to construct a tuple.	(Raju, 30)
{}	**Bag constructor operator** − This operator is used to construct a bag.	{(Raju, 30), (Mohammad, 45)}
[]	**Map constructor operator** − This operator is used to construct a tuple.	[name#Raja, age#30]

Pig Latin – Relational Operations

The following table describes the relational operators of Pig Latin.

Operator	Description
Loading and Storing	
LOAD	To Load the data from the file system (local/HDFS) into a relation.
STORE	To save a relation to the file system (local/HDFS).
Filtering	
FILTER	To remove unwanted rows from a relation.
DISTINCT	To remove duplicate rows from a relation.

FOREACH, GENERATE	To generate data transformations based on columns of data.
STREAM	To transform a relation using an external program.
Grouping and Joining	
JOIN	To join two or more relations.
COGROUP	To group the data in two or more relations.
GROUP	To group the data in a single relation.
CROSS	To create the cross product of two or more relations.
Sorting	
ORDER	To arrange a relation in a sorted order based on one or more fields (ascending or descending).
LIMIT	To get a limited number of tuples from a relation.
Combining and Splitting	
UNION	To combine two or more relations into a single relation.
SPLIT	To split a single relation into two or more relations.
Diagnostic Operators	
DUMP	To print the contents of a relation on the console.
DESCRIBE	To describe the schema of a relation.
EXPLAIN	To view the logical, physical, or MapReduce execution plans to compute a relation.
ILLUSTRATE	To view the step-by-step execution of a series of statements.

Apache Pig - Grunt Shell

After invoking the Grunt shell, you can run your Pig scripts in the shell. In addition to that, there are certain useful shell and utility commands provided by the Grunt shell. This chapter explains the shell and utility commands provided by the Grunt shell.

Shell Commands

The Grunt shell of Apache Pig is mainly used to write Pig Latin scripts. Prior to that, we can invoke any shell commands using **sh** and **fs**.

sh Command

Using **sh** command, we can invoke any shell commands from the Grunt shell. Using **sh** command from the Grunt shell, we cannot execute the commands that are a part of the shell environment (**ex** − cd).

Syntax

Given below is the syntax of **sh** command.

```
grunt> sh shell command parameters
```

Example

We can invoke the **ls** command of Linux shell from the Grunt shell using the **sh** option as shown below. In this example, it lists out the files in the **/pig/bin/** directory.

```
grunt> sh ls
pig
pig_1444799121955.log
pig.cmd
pig.py
```

fs Command

Using the **fs** command, we can invoke any FsShell commands from the Grunt shell.

Syntax

Given below is the syntax of **fs** command.

```
grunt> sh File System command parameters
```

Example

We can invoke the ls command of HDFS from the Grunt shell using fs command. In the following example, it lists the files in the HDFS root directory.

```
grunt> fs –ls


Found 3 items
drwxrwxrwx - Hadoop supergroup 0 2015-09-08 14:13 Hbase
drwxr-xr-x - Hadoop supergroup 0 2015-09-09 14:52 seqgen_data
drwxr-xr-x - Hadoop supergroup 0 2015-09-08 11:30 twitter_data
```

In the same way, we can invoke all the other file system shell commands from the Grunt shell using the **fs** command.

Utility Commands

The Grunt shell provides a set of utility commands. These include utility commands such as **clear, help, history, quit,** and **set**; and commands such as **exec, kill,** and **run** to control Pig from the Grunt shell. Given below is the description of the utility commands provided by the Grunt shell.

clear Command

The **clear** command is used to clear the screen of the Grunt shell.

Syntax

You can clear the screen of the grunt shell using the **clear** command as shown below.

grunt> clear

help Command

The **help** command gives you a list of Pig commands or Pig properties.

Usage

You can get a list of Pig commands using the **help** command as shown below.

grunt> help

Commands: <pig latin statement>; - See the PigLatin manual for details:

http://hadoop.apache.org/pig

File system commands:fs <fs arguments> - Equivalent to Hadoop dfs command:

http://hadoop.apache.org/common/docs/current/hdfs_shell.html

Diagnostic Commands:describe <alias>[::<alias] - Show the schema for the alias.

Inner aliases can be described as A::B.

explain [-script <pigscript>] [-out <path>] [-brief] [-dot|-xml]

[-param <param_name>=<pCram_value>]

[-param_file <file_name>] [<alias>] -

Show the execution plan to compute the alias or for entire script.

-script - Explain the entire script.

-out - Store the output into directory rather than print to stdout.

-brief - Don't expand nested plans (presenting a smaller graph for overview).

-dot - Generate the output in .dot format. Default is text format.

-xml - Generate the output in .xml format. Default is text format.

-param <param_name - See parameter substitution for details.

-param_file <file_name> - See parameter substitution for details.

alias - Alias to explain.

dump <alias> - Compute the alias and writes the results to stdout.

Utility Commands: exec [-param <param_name>=param_value] [-param_file <file_name>] <script> -

Execute the script with access to grunt environment including aliases.

-param <param_name - See parameter substitution for details.

-param_file <file_name> - See parameter substitution for details.

script - Script to be executed.

run [-param <param_name>=param_value] [-param_file <file_name>] <script> -

Execute the script with access to grunt environment.

-param <param_name - See parameter substitution for details.

-param_file <file_name> - See parameter substitution for details.

script - Script to be executed.

sh <shell command> - Invoke a shell command.

kill <job_id> - Kill the hadoop job specified by the hadoop job id.

set <key> <value> - Provide execution parameters to Pig. Keys and values are case sensitive.

The following keys are supported:

default_parallel - Script-level reduce parallelism. Basic input size heuristics used

by default.

debug - Set debug on or off. Default is off.

job.name - Single-quoted name for jobs. Default is PigLatin:<script name>

job.priority - Priority for jobs. Values: very_low, low, normal, high, very_high.

Default is normal stream.skippath - String that contains the path.

This is used by streaming any hadoop property.

help - Display this message.

history [-n] - Display the list statements in cache.

-n Hide line numbers.

quit - Quit the grunt shell.

history Command

This command displays a list of statements executed / used so far since the Grunt sell is invoked.

Usage

Assume we have executed three statements since opening the Grunt shell.

```
grunt> customers = LOAD 'hdfs://localhost:9000/pig_data/customers.txt'
USING PigStorage(',');


grunt> orders = LOAD 'hdfs://localhost:9000/pig_data/orders.txt' USING
PigStorage(',');


grunt> student = LOAD 'hdfs://localhost:9000/pig_data/student.txt'
USING PigStorage(',');
```

Then, using the **history** command will produce the following output.

grunt> history

```
customers = LOAD 'hdfs://localhost:9000/pig_data/customers.txt' USING
PigStorage(',');

orders = LOAD 'hdfs://localhost:9000/pig_data/orders.txt' USING
PigStorage(',');

student = LOAD 'hdfs://localhost:9000/pig_data/student.txt' USING
PigStorage(',');
```

set Command

The **set** command is used to show/assign values to keys used in Pig.

Usage

Using this command, you can set values to the following keys.

Key	Description and values
default_parallel	You can set the number of reducers for a map job by passing any whole number as a value to this key.
debug	You can turn off or turn on the debugging freature in Pig by passing on/off to this key.
job.name	You can set the Job name to the required job by passing a string value to this key.

job.priority	You can set the job priority to a job by passing one of the following values to this key − • very_low • low • normal • high • very_high
stream.skippath	For streaming, you can set the path from where the data is not to be transferred, by passing the desired path in the form of a string to this key.

quit Command

You can quit from the Grunt shell using this command.

Usage

Quit from the Grunt shell as shown below.

```
grunt> quit
```

Let us now take a look at the commands using which you can control Apache Pig from the Grunt shell.

exec Command

Using the **exec** command, we can execute Pig scripts from the Grunt shell.

Syntax

Given below is the syntax of the utility command **exec**.

```
grunt> exec [–param param_name = param_value] [–param_file
file_name] [script]
```

Example

Let us assume there is a file named **student.txt** in the **/pig_data/** directory of HDFS with the following content.

Student.txt

```
001,Rajiv,Hyderabad

002,siddarth,Kolkata

003,Rajesh,Delhi
```

And, assume we have a script file named **sample_script.pig** in the **/pig_data/** directory of HDFS with the following content.

Sample_script.pig

```
student = LOAD 'hdfs://localhost:9000/pig_data/student.txt' USING
PigStorage(',')

   as (id:int,name:chararray,city:chararray);



Dump student;
```

Now, let us execute the above script from the Grunt shell using the **exec** command as shown below.

```
grunt> exec /sample_script.pig
```

Output

The **exec** command executes the script in the **sample_script.pig**. As directed in the script, it loads the **student.txt** file into Pig and gives you the result of the Dump operator displaying the following content.

```
(1,Rajiv,Hyderabad)

(2,siddarth,Kolkata)

(3,Rajesh,Delhi)
```

kill Command

You can kill a job from the Grunt shell using this command.

Syntax

Given below is the syntax of the **kill** command.

```
grunt> kill JobId
```

Example

Suppose there is a running Pig job having id **Id_0055**, you can kill it from the Grunt shell using the **kill** command, as shown below.

```
grunt> kill Id_0055
```

run Command

You can run a Pig script from the Grunt shell using the **run** command

Syntax

Given below is the syntax of the **run** command.

```
grunt> run [–param param_name = param_value] [–param_file file_name]
script
```

Example

Let us assume there is a file named **student.txt** in the **/pig_data/** directory of HDFS with the following content.

Student.txt

```
001,Rajiv,Hyderabad

002,siddarth,Kolkata

003,Rajesh,Delhi
```

And, assume we have a script file named **sample_script.pig** in the local filesystem with the following content.

Sample_script.pig

```
student = LOAD 'hdfs://localhost:9000/pig_data/student.txt' USING

  PigStorage(',') as (id:int,name:chararray,city:chararray);
```

Now, let us run the above script from the Grunt shell using the run command as shown below.

```
grunt> run /sample_script.pig
```

You can see the output of the script using the **Dump operator** as shown below.

```
grunt> Dump;

(1,Rajiv,Hyderabad)

(2,siddarth,Kolkata)

(3,Rajesh,Delhi)
```

Note − The difference between **exec** and the **run** command is that if we use **run**, the statements from the script are available in the command history.

5.2 SUMMARY

Pig Data Types

Apache Pig supports many data types. A list of Apache Pig Data Types with description and examples are given below.

Type	Description	Example
Int	Signed 32 bit integer	2
Long	Signed 64 bit integer	15L or 15l
Float	32 bit floating point	2.5f or 2.5F
Double	32 bit floating point	1.5 or 1.5e2 or 1.5E2
charArray	Character array	hello students
byteArray	BLOB(Byte array)	
Tuple	Ordered set of fields	(12,43)
Bag	Collection f tuples	{(12,43),(54,28)}
Map	collection of tuples	[open#apache]

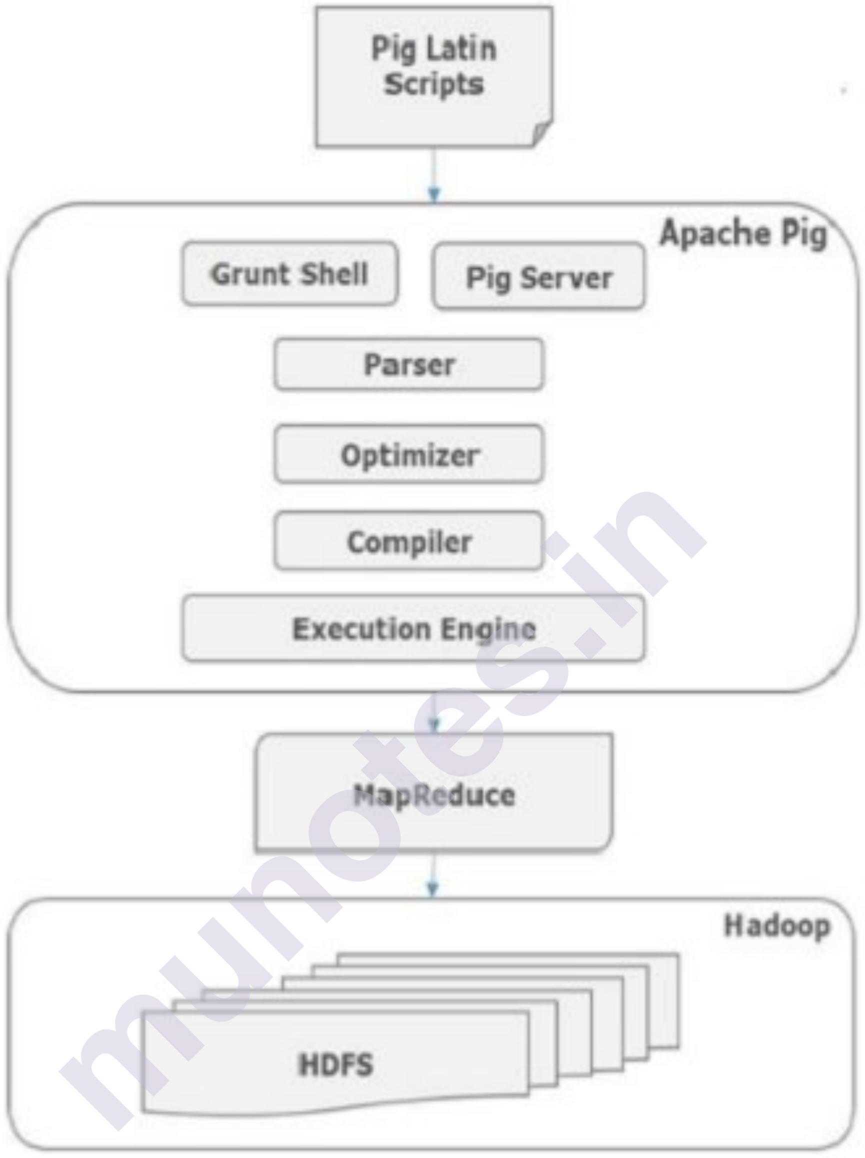
Apache Pig - Architecture

Creating a data model in pig:

The language used to analyze data in Hadoop using Pig is known as **Pig Latin**. It is a highlevel data processing language which provides a rich set of data types and operators to perform various operations on the data.

To perform a particular task Programmers using Pig, programmers need to write a Pig script using the Pig Latin language, and execute them using any of the execution mechanisms (Grunt Shell, UDFs, Embedded). After execution, these scripts will go through a series of transformations applied by the Pig Framework, to produce the desired output.

Internally, Apache Pig converts these scripts into a series of MapReduce jobs, and thus, it makes the programmer's job easy. The architecture of Apache Pig is shown below.



Apache Pig Components

As shown in the figure, there are various components in the Apache Pig framework. Let us take a look at the major components.

Parser

Initially the Pig Scripts are handled by the Parser. It checks the syntax of the script, does type checking, and other miscellaneous checks. The output of the parser will be a DAG (directed acyclic graph), which represents the Pig Latin statements and logical operators.

In the DAG, the logical operators of the script are represented as the nodes and the data flows are represented as edges.

Optimizer

The logical plan (DAG) is passed to the logical optimizer, which carries out the logical optimizations such as projection and pushdown.

Compiler

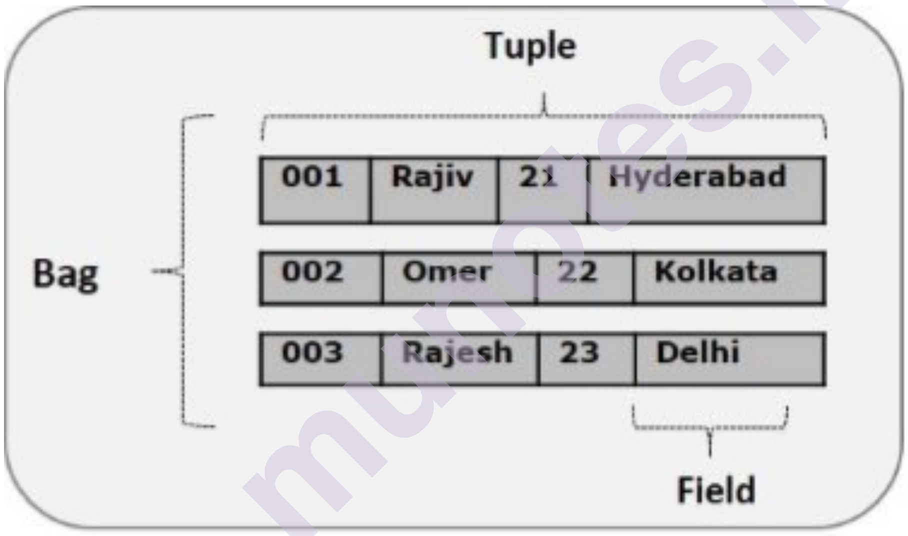
The compiler compiles the optimized logical plan into a series of MapReduce jobs.

Execution engine

Finally the MapReduce jobs are submitted to Hadoop in a sorted order. Finally, these MapReduce jobs are executed on Hadoop producing the desired results.

Pig Latin Data Model

The data model of Pig Latin is fully nested and it allows complex non-atomic datatypes such as **map** and **tuple**. Given below is the diagrammatical representation of Pig Latin's data model.



Atom

Any single value in Pig Latin, irrespective of their data, type is known as an **Atom**. It is stored as string and can be used as string and number. int, long, float, double, chararray, and bytearray are the atomic values of Pig. A piece of data or a simple atomic value is known as a **field**.

Example − 'raja' or '30'

Tuple

A record that is formed by an ordered set of fields is known as a tuple, the fields can be of any type. A tuple is similar to a row in a table of RDBMS.

Example − (Raja, 30)

Bag

A bag is an unordered set of tuples. In other words, a collection of tuples (non-unique) is known as a bag. Each tuple can have any number of fields (flexible schema). A bag is represented by '{}'. It is similar to a table in RDBMS, but unlike a table in RDBMS, it is not necessary that every tuple contain the same number of fields or that the fields in the same position (column) have the same type.

Example − {(Raja, 30), (Mohammad, 45)}

A bag can be a field in a relation; in that context, it is known as **inner bag**.

Example − {Raja, 30, **{9848022338, raja@gmail.com,}**}

Map

A map (or data map) is a set of key-value pairs. The **key** needs to be of type chararray and should be unique. The **value** might be of any type. It is represented by '[]'

Example − [name#Raja, age#30]

Relation

A relation is a bag of tuples. The relations in Pig Latin are unordered (there is no guarantee that tuples are processed in any particular order).

Apache Pig - Reading Data

In general, Apache Pig works on top of Hadoop. It is an analytical tool that analyzes large datasets that exist in the **H**adoop **F**ile **S**ystem. To analyze data using Apache Pig, we have to initially load the data into Apache Pig. This chapter explains how to load data to Apache Pig from HDFS.

Preparing HDFS

In MapReduce mode, Pig reads (loads) data from HDFS and stores the results back in HDFS. Therefore, let us start HDFS and create the following sample data in HDFS.

Student ID	First Name	Last Name	Phone	City
001	Rajiv	Reddy	9848022337	Hyderabad
002	siddarth	Battacharya	9848022338	Kolkata
003	Rajesh	Khanna	9848022339	Delhi
004	Preethi	Agarwal	9848022330	Pune
005	Trupthi	Mohanthy	9848022336	Bhuwaneshwar
006	Archana	Mishra	9848022335	Chennai

The above dataset contains personal details like id, first name, last name, phone number and city, of six students.

Step 1: Verifying Hadoop

First of all, verify the installation using Hadoop version command, as shown below.

```
$ hadoop version
```

If your system contains Hadoop, and if you have set the PATH variable, then you will get the following output −

```
Hadoop 2.6.0

Subversion https://git-wip-us.apache.org/repos/asf/hadoop.git -r

e3496499ecb8d220fba99dc5ed4c99c8f9e33bb1

Compiled by jenkins on 2014-11-13T21:10Z

Compiled with protoc 2.5.0

From source with checksum 18e43357c8f927c0695f1e9522859d6a

This command was run using
/home/Hadoop/hadoop/share/hadoop/common/hadoop

common-2.6.0.jar
```

Step 2: Starting HDFS

Browse through the **sbin** directory of Hadoop and start **yarn** and Hadoop dfs (distributed file system) as shown below.

```
cd /$Hadoop_Home/sbin/
```

$ start-dfs.sh

```
localhost: starting namenode, logging to
/home/Hadoop/hadoop/logs/hadoopHadoop-namenode-
localhost.localdomain.out

localhost: starting datanode, logging to
/home/Hadoop/hadoop/logs/hadoopHadoop-datanode-
localhost.localdomain.out

Starting secondary namenodes [0.0.0.0]

starting secondarynamenode, logging to
/home/Hadoop/hadoop/logs/hadoop-Hadoopsecondarynamenode-
localhost.localdomain.out
```

$ start-yarn.sh

```
starting yarn daemons

starting resourcemanager, logging to /home/Hadoop/hadoop/logs/yarn-
Hadoopresourcemanager-localhost.localdomain.out

localhost: starting nodemanager, logging to
/home/Hadoop/hadoop/logs/yarnHadoop-nodemanager-
localhost.localdomain.out
```

Step 3: Create a Directory in HDFS

In Hadoop DFS, you can create directories using the command **mkdir**. Create a new directory in HDFS with the name **Pig_Data** in the required path as shown below.

```
$cd /$Hadoop_Home/bin/

$ hdfs dfs -mkdir hdfs://localhost:9000/Pig_Data
```

Step 4: Placing the data in HDFS

The input file of Pig contains each tuple/record in individual lines. And the entities of the record are separated by a delimiter (In our example we used **","**).

In the local file system, create an input file **student_data.txt** containing data as shown below.

```
001,Rajiv,Reddy,9848022337,Hyderabad

002,siddarth,Battacharya,9848022338,Kolkata

003,Rajesh,Khanna,9848022339,Delhi

004,Preethi,Agarwal,9848022330,Pune
```

005,Trupthi,Mohanthy,9848022336,Bhuwaneshwar

006,Archana,Mishra,9848022335,Chennai.

Now, move the file from the local file system to HDFS using **put** command as shown below. (You can use **copyFromLocal** command as well.)

```
$ cd $HADOOP_HOME/bin

$ hdfs dfs -put /home/Hadoop/Pig/Pig_Data/student_data.txt
dfs://localhost:9000/pig_data/
```

Verifying the file

You can use the **cat** command to verify whether the file has been moved into the HDFS, as shown below.

```
$ cd $HADOOP_HOME/bin

$ hdfs dfs -cat hdfs://localhost:9000/pig_data/student_data.txt
```

Output

You can see the content of the file as shown below.

```
15/10/01 12:16:55 WARN util.NativeCodeLoader: Unable to load native-
hadoop

library for your platform... using builtin-java classes where applicable


001,Rajiv,Reddy,9848022337,Hyderabad

002,siddarth,Battacharya,9848022338,Kolkata

003,Rajesh,Khanna,9848022339,Delhi

004,Preethi,Agarwal,9848022330,Pune

005,Trupthi,Mohanthy,9848022336,Bhuwaneshwar

006,Archana,Mishra,9848022335,Chennai
```

The Load Operator

You can load data into Apache Pig from the file system (HDFS/ Local) using **LOAD** operator of **Pig Latin**.

Syntax

The load statement consists of two parts divided by the "=" operator. On the left-hand side, we need to mention the name of the relation **where** we want to store the data, and on the right-hand side, we have to define **how** we store the data. Given below is the syntax of the **Load** operator.

Relation_name = LOAD 'Input file path' USING function as schema;

Where,

- **relation_name** − We have to mention the relation in which we want to store the data.
- **Input file path** − We have to mention the HDFS directory where the file is stored. (In MapReduce mode)
- **function** − We have to choose a function from the set of load functions provided by Apache Pig (**BinStorage, JsonLoader, PigStorage, TextLoader**).
- **Schema** − We have to define the schema of the data. We can define the required schema as follows −

(column1 : data type, column2 : data type, column3 : data type);

Note − We load the data without specifying the schema. In that case, the columns will be addressed as $01, $02, etc… (check).

Example

As an example, let us load the data in **student_data.txt** in Pig under the schema named **Student** using the **LOAD** command.

Start the Pig Grunt Shell

First of all, open the Linux terminal. Start the Pig Grunt shell in MapReduce mode as shown below.

```
$ Pig –x mapreduce
```

It will start the Pig Grunt shell as shown below.

```
15/10/01 12:33:37 INFO pig.ExecTypeProvider: Trying ExecType :
LOCAL

15/10/01 12:33:37 INFO pig.ExecTypeProvider: Trying ExecType :
MAPREDUCE

15/10/01 12:33:37 INFO pig.ExecTypeProvider: Picked MAPREDUCE as
the ExecType


2015-10-01 12:33:38,080 [main] INFO org.apache.pig.Main - Apache Pig
version 0.15.0 (r1682971) compiled Jun 01 2015, 11:44:35

2015-10-01 12:33:38,080 [main] INFO org.apache.pig.Main - Logging
error messages to: /home/Hadoop/pig_1443683018078.log

2015-10-01 12:33:38,242 [main] INFO org.apache.pig.impl.util.Utils -
Default bootup file /home/Hadoop/.pigbootup not found


2015-10-01 12:33:39,630 [main]
```

```
INFO org.apache.pig.backend.hadoop.executionengine.HExecutionEngine
- Connecting to hadoop file system at: hdfs://localhost:9000

grunt>
```

Execute the Load Statement

Now load the data from the file **student_data.txt** into Pig by executing the following Pig Latin statement in the Grunt shell.

```
grunt> student = LOAD 'hdfs://localhost:9000/pig_data/student_data.txt'
  USING PigStorage(',')
  as ( id:int, firstname:chararray, lastname:chararray, phone:chararray,
  city:chararray );
```

Following is the description of the above statement.

Relation name	We have stored the data in the schema **student**.
Input file path	We are reading data from the file **student_data.txt,** which is in the /pig_data/ directory of HDFS.
Storage function	We have used the **PigStorage()** function. It loads and stores data as structured text files. It takes a delimiter using which each entity of a tuple is separated, as a parameter. By default, it takes '\t' as a parameter.
schema	We have stored the data using the following schema.

column	id	firstname	Lastname	phone
datatype	int	char array	char array	char array

Apache Pig - Storing Data

we learnt how to load data into Apache Pig. You can store the loaded data in the file system using the **store** operator. This chapter explains how to store data in Apache Pig using the **Store** operator.

Syntax

Given below is the syntax of the Store statement.

STORE Relation_name INTO ' required_directory_path ' [USING function];

Example

Assume we have a file **student_data.txt** in HDFS with the following content.

001,Rajiv,Reddy,9848022337,Hyderabad

002,siddarth,Battacharya,9848022338,Kolkata

003,Rajesh,Khanna,9848022339,Delhi

004,Preethi,Agarwal,9848022330,Pune

005,Trupthi,Mohanthy,9848022336,Bhuwaneshwar

006,Archana,Mishra,9848022335,Chennai.

And we have read it into a relation **student** using the LOAD operator as shown below.

```
grunt> student = LOAD 'hdfs://localhost:9000/pig_data/student_data.txt'

  USING PigStorage(',')

  as ( id:int, firstname:chararray, lastname:chararray, phone:chararray,

  city:chararray );
```

Now, let us store the relation in the HDFS directory **"/pig_Output/"** as shown below.

```
grunt> STORE student INTO ' hdfs://localhost:9000/pig_Output/ ' USING
PigStorage (',');
```

Output

After executing the **store** statement, you will get the following output. A directory is created with the specified name and the data will be stored in it.

```
2015-10-05 13:05:05,429 [main] INFO
org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.

MapReduceLau ncher - 100% complete

2015-10-05 13:05:05,429 [main] INFO
org.apache.pig.tools.pigstats.mapreduce.SimplePigStats -

Script Statistics:

HadoopVersion PigVersion UserId StartedAt FinishedAt Features

2.6.0 0.15.0 Hadoop 2015-10-0 13:03:03 2015-10-05 13:05:05
UNKNOWN
```

```
Success!

Job Stats (time in seconds):

JobId Maps Reduces MaxMapTime MinMapTime AvgMapTime
MedianMapTime

job_14459_06 1 0 n/a n/a n/a n/a

MaxReduceTime MinReduceTime AvgReduceTime MedianReducetime
Alias Feature

0 0 0 0 student MAP_ONLY

OutPut folder

hdfs://localhost:9000/pig_Output/


Input(s): Successfully read 0 records from:
"hdfs://localhost:9000/pig_data/student_data.txt"

Output(s): Successfully stored 0 records in:
"hdfs://localhost:9000/pig_Output"

Counters:

Total records written : 0

Total bytes written : 0

Spillable Memory Manager spill count : 0

Total bags proactively spilled: 0

Total records proactively spilled: 0


Job DAG: job_1443519499159_0006


2015-10-05 13:06:06,192 [main] INFO
org.apache.pig.backend.hadoop.executionengine

.mapReduceLayer.MapReduceLau ncher - Success!
```

Verification

You can verify the stored data as shown below.

Step 1

First of all, list out the files in the directory named **pig_output** using the **ls** command as shown below.

hdfs dfs -ls 'hdfs://localhost:9000/pig_Output/'

```
Found 2 items

rw-r--r- 1 Hadoop supergroup 0 2015-10-05 13:03
hdfs://localhost:9000/pig_Output/_SUCCESS

rw-r--r- 1 Hadoop supergroup 224 2015-10-05 13:03
hdfs://localhost:9000/pig_Output/part-m-00000
```

You can observe that two files were created after executing the **store** statement.

Step 2

Using **cat** command, list the contents of the file named **part-m-00000** as shown below.

$ hdfs dfs -cat 'hdfs://localhost:9000/pig_Output/part-m-00000'

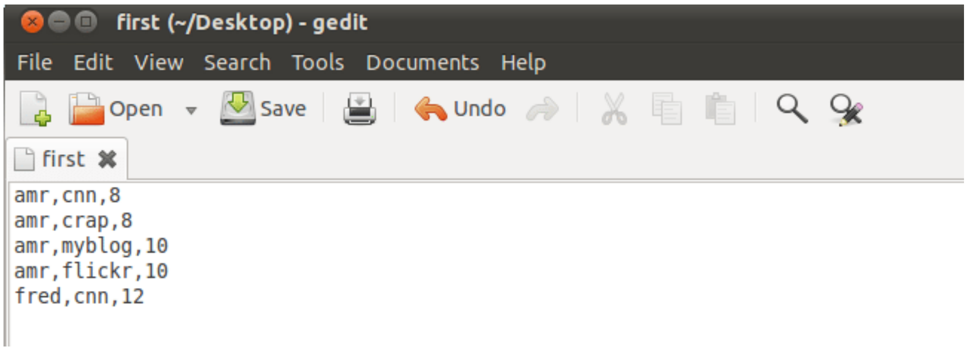
```
1,Rajiv,Reddy,9848022337,Hyderabad

2,siddarth,Battacharya,9848022338,Kolkata

3,Rajesh,Khanna,9848022339,Delhi

4,Preethi,Agarwal,9848022330,Pune

5,Trupthi,Mohanthy,9848022336,Bhuwaneshwar

6,Archana,Mishra,9848022335,Chennai
```

Apache Pig Operators

The Apache Pig Operators is a high-level procedural language for querying large data sets using Hadoop and the Map Reduce Platform. A Pig Latin statement is an operator that takes a relation as input and produces another relation as output. These operators are the main tools for Pig Latin provides to operate on the data. They allow you to transform it by sorting, grouping, joining, projecting, and filtering.

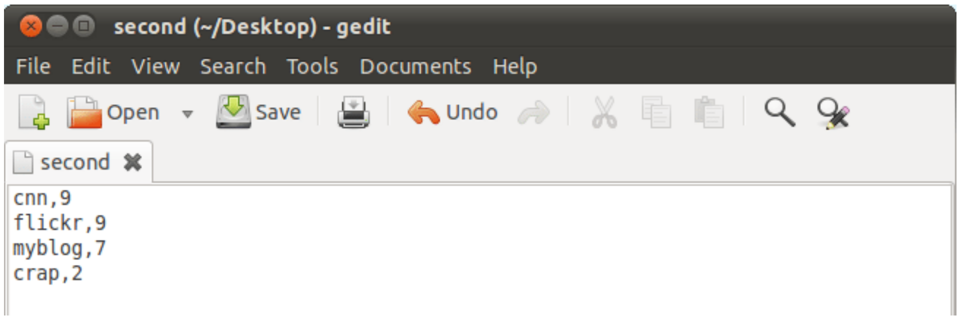
Let's create two files to run the commands:

We have two files with name 'first' and 'second.' The first file contain three fields: user, url & id.


first (~/Desktop) - gedit
File Edit View Search Tools Documents Help
Open
Save
Undo
first

```
amr,cnn,8
amr,crap,8
amr,myblog,10
amr,flickr,10
fred,cnn,12
```

The second file contain two fields: url & rating. These two files are CSV files.


second (~/Desktop) - gedit
File Edit View Search Tools Documents Help
Open
Save
Undo
second

```
cnn,9
flickr,9
myblog,7
crap,2
```

The Apache Pig operators can be classified as: *Relational and Diagnostic.*

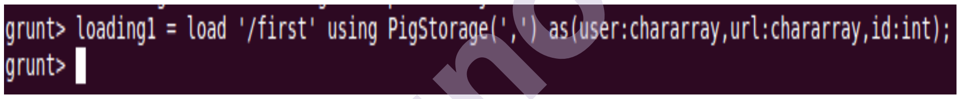
Relational Operators:

Relational operators are the main tools Pig Latin provides to operate on the data. It allows you to transform the data by sorting, grouping, joining, projecting and filtering. This section covers the basic relational operators.

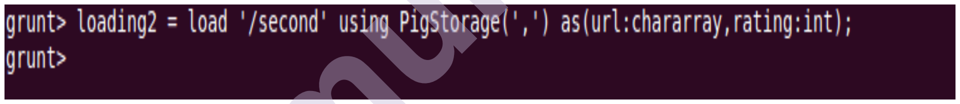
LOAD:

LOAD operator is used to load data from the file system or HDFS storage into a Pig relation.

In this example, the Load operator loads data from file 'first' to form relation 'loading1'. The field names are user, url, id.



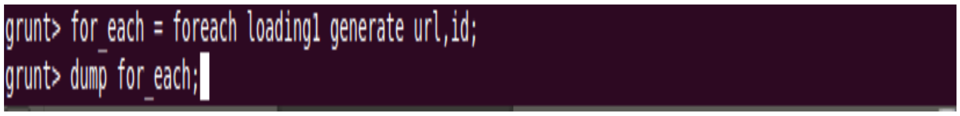
```
grunt> loading1 = load '/first' using PigStorage(',') as(user:chararray,url:chararray,id:int);
grunt>
```



```
grunt> loading2 = load '/second' using PigStorage(',') as(url:chararray,rating:int);
grunt>
```

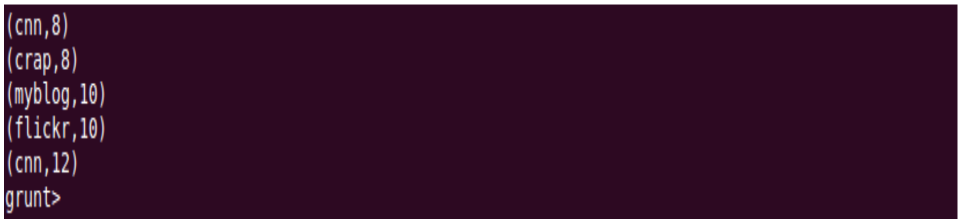
FOREACH:

This operator generates data transformations based on columns of data. It is used to add or remove fields from a relation. Use FOREACH-GENERATE operation to work with columns of data.



```
grunt> for_each = foreach loading1 generate url,id;
grunt> dump for_each;
```

FOREACH Result:

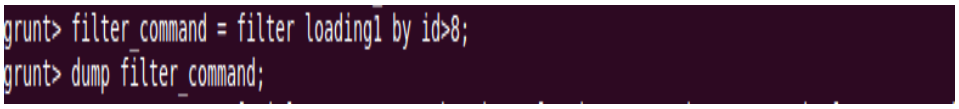


```
(cnn,8)
(crap,8)
(myblog,10)
(flickr,10)
(cnn,12)
grunt>
```

FILTER:

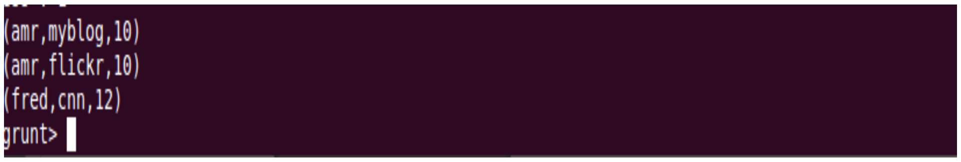
This operator selects tuples from a relation based on a condition.

In this example, we are filtering the record from 'loading1' when the condition 'id' is greater than 8.



```
grunt> filter_command = filter loading1 by id>8;
grunt> dump filter_command;
```

FILTER Result:

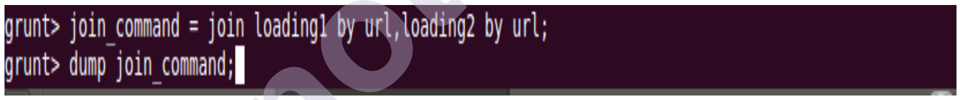


```
(amr,myblog,10)
(amr,flickr,10)
(fred,cnn,12)
grunt>
```

JOIN:

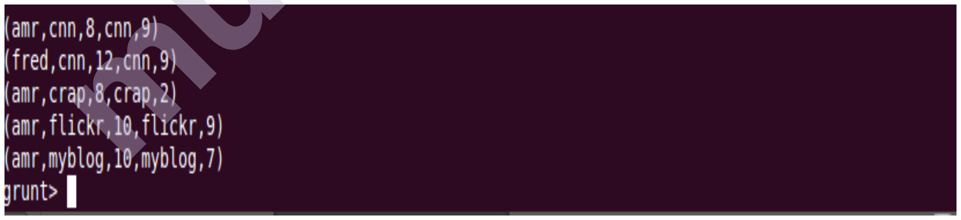
JOIN operator is used to perform an inner, equijoin join of two or more relations based on common field values. The JOIN operator always performs an inner join. Inner joins ignore null keys, so it makes sense to filter them out before the join.

In this example, join the two relations based on the column 'url' from 'loading1' and 'loading2'.



```
grunt> join_command = join loading1 by url,loading2 by url;
grunt> dump join_command;
```

JOIN Result:

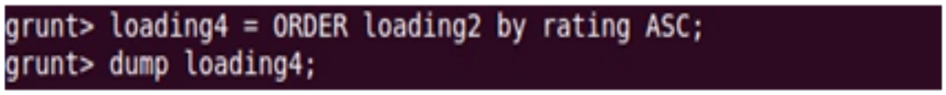


```
(amr,cnn,8,cnn,9)
(fred,cnn,12,cnn,9)
(amr,crap,8,crap,2)
(amr,flickr,10,flickr,9)
(amr,myblog,10,myblog,7)
grunt>
```

ORDER BY:

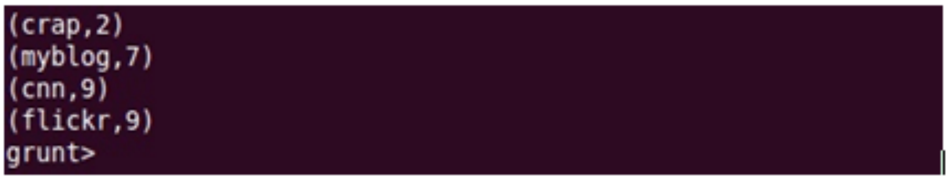
Order By is used to sort a relation based on one or more fields. You can do sorting in ascending or descending order using ASC and DESC keywords.

In below example, we are sorting data in loading2 in ascending order on ratings field.



```
grunt> loading4 = ORDER loading2 by rating ASC;
grunt> dump loading4;
```

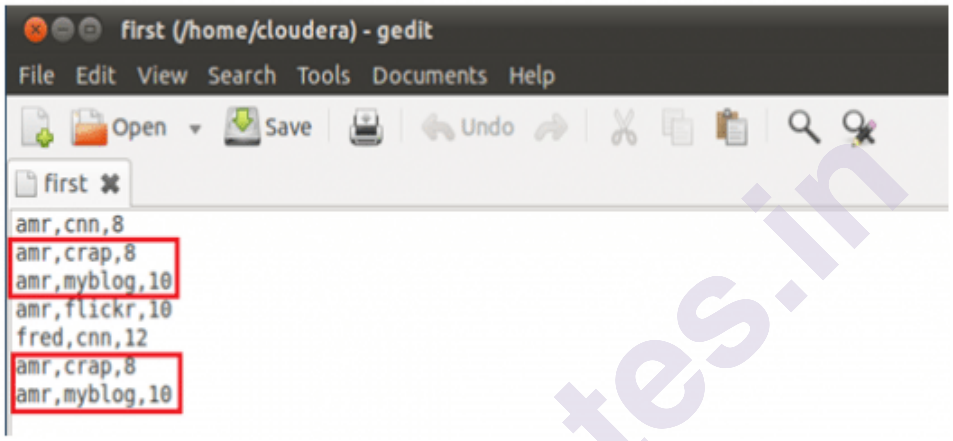
ORDER BY Result:



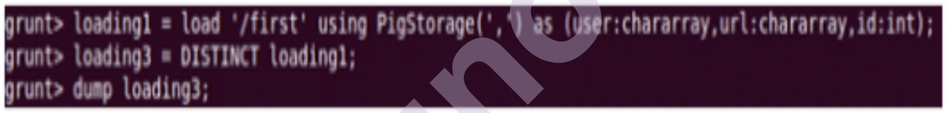
```
(crap,2)
(myblog,7)
(cnn,9)
(flickr,9)
grunt>
```

DISTINCT:

Distinct removes duplicate tuples in a relation.Lets take an input file as below, which has **amr,crap,8** and **amr,myblog,10** twice in the file. When we apply distinct on the data in this file, duplicate entries are removed.

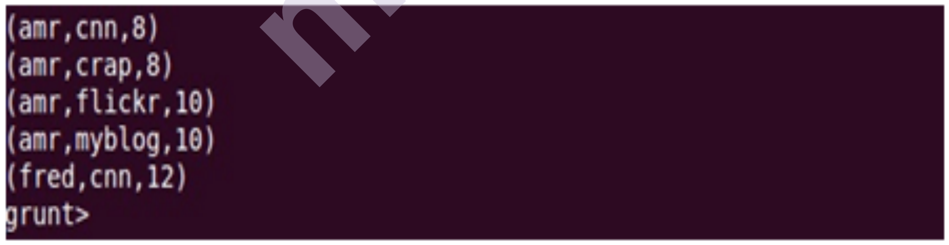

first (/home/cloudera) - gedit
File Edit View Search Tools Documents Help
Open
Save
Undo
first

```
amr,cnn,8
amr,crap,8
amr,myblog,10
amr,flickr,10
fred,cnn,12
amr,crap,8
amr,myblog,10
```



```
grunt> loading1 = load '/first' using PigStorage(',') as (user:chararray,url:chararray,id:int);
grunt> loading3 = DISTINCT loading1;
grunt> dump loading3;
```

DISTINCT Result:

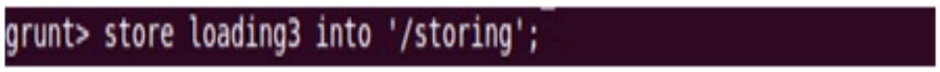


```
(amr,cnn,8)
(amr,crap,8)
(amr,flickr,10)
(amr,myblog,10)
(fred,cnn,12)
grunt>
```

STORE:

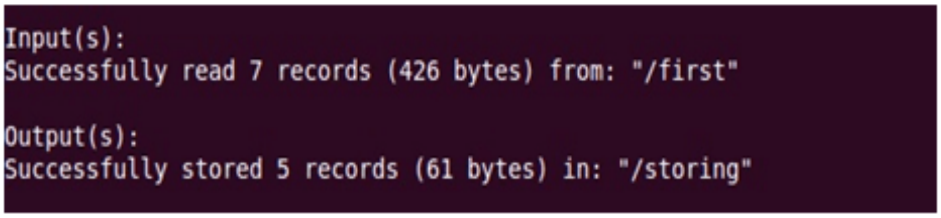
Store is used to save results to the file system.

Here we are saving **loading3** data into a file named **storing** on HDFS.

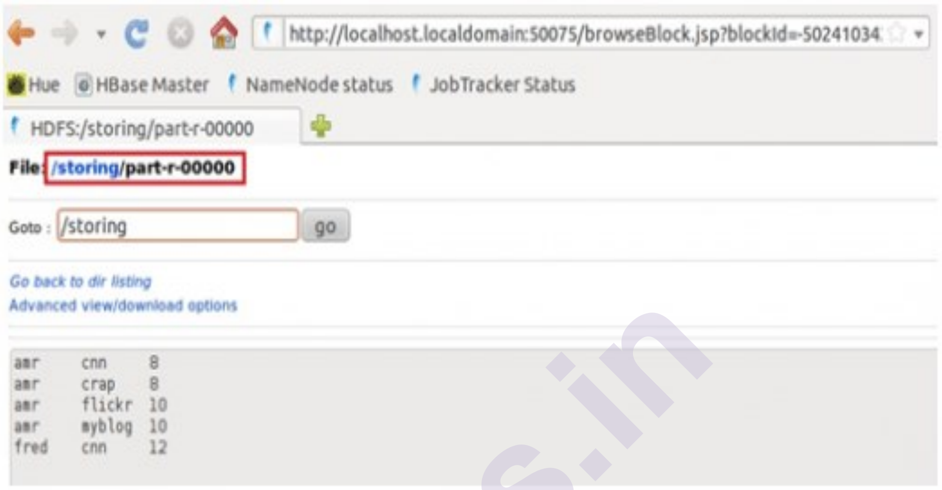


```
grunt> store loading3 into '/storing';
```

STORE Result:



```
Input(s):
Successfully read 7 records (426 bytes) from: "/first"

Output(s):
Successfully stored 5 records (61 bytes) in: "/storing"
```



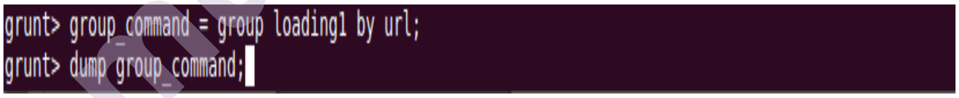
http://localhost.localdomain:50075/browseBlock.jsp?blockId=-50241034
Hue
HBase Master
NameNode status
JobTracker Status
HDFS:/storing/part-r-00000
File: /storing/part-r-00000
Goto : /storing
go
Go back to dir listing
Advanced view/download options

```
amr     cnn     8
amr     crap    8
amr     flickr  10
amr     myblog  10
fred    cnn     12
```

GROUP:

The GROUP operator groups together the tuples with the same group key (key field). The key field will be a tuple if the group key has more than one field, otherwise it will be the same type as that of the group key. The result of a GROUP operation is a relation that includes one tuple per group.

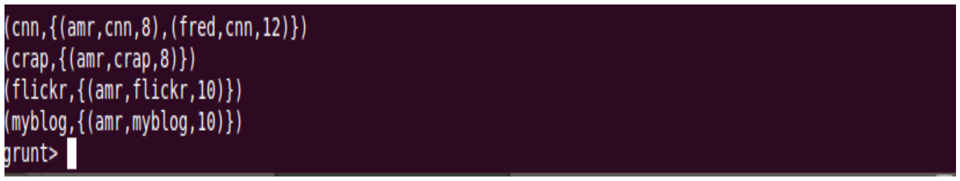
In this example, group th



```
grunt> group_command = group loading1 by url;
grunt> dump group_command;
```

e relation ‘loading1’ by column url.

GROUP Result:

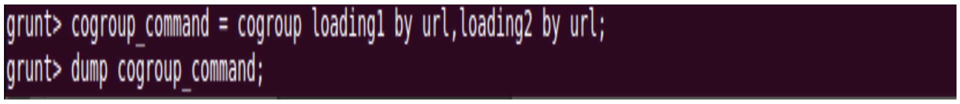


```
(cnn,{(amr,cnn,8),(fred,cnn,12)})
(crap,{(amr,crap,8)})
(flickr,{(amr,flickr,10)})
(myblog,{(amr,myblog,10)})
grunt>
```

COGROUP:

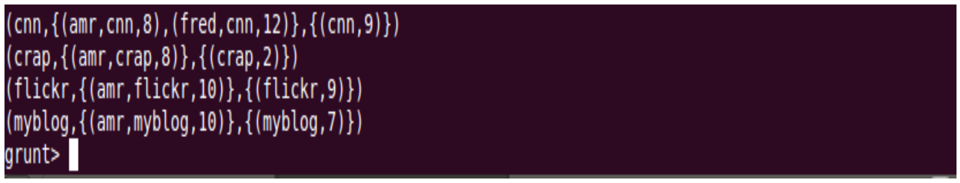
COGROUP is same as GROUP operator. For readability, programmers usually use GROUP when only one relation is involved and COGROUP when multiple relations re involved.

In this example group the ‘loading1’ and ‘loading2’ by url field in both relations.



```
grunt> cogroup_command = cogroup loading1 by url,loading2 by url;
grunt> dump cogroup_command;
```

COGROUP Result:

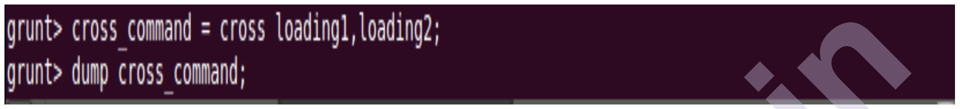


```
(cnn,{(amr,cnn,8),(fred,cnn,12)},{(cnn,9)})
(crap,{(amr,crap,8)},{(crap,2)})
(flickr,{(amr,flickr,10)},{(flickr,9)})
(myblog,{(amr,myblog,10)},{(myblog,7)})
grunt>
```

CROSS:

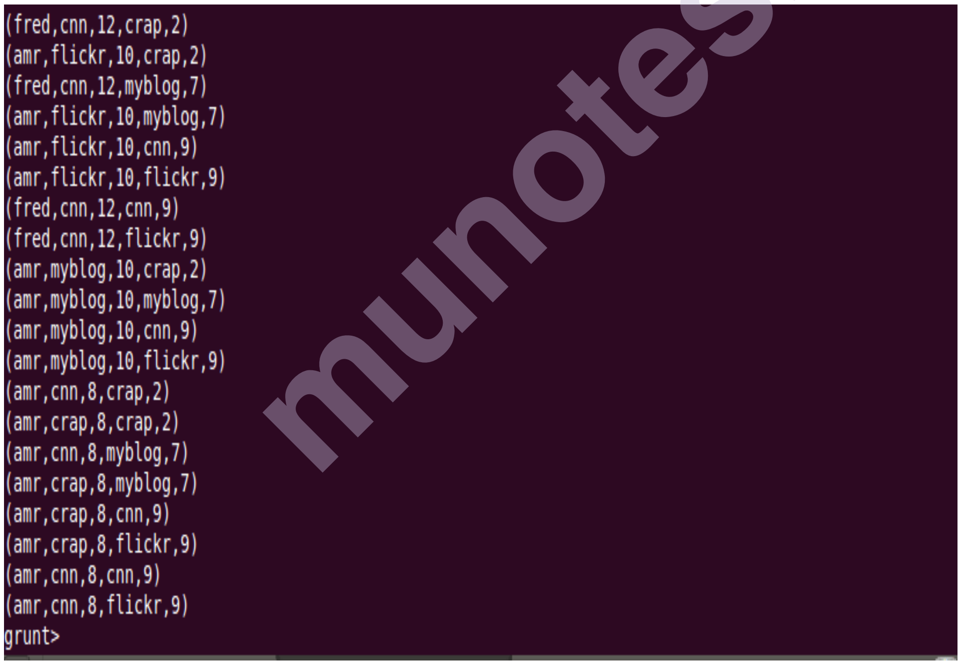
The CROSS operator is used to compute the cross product (Cartesian product) of two or more relations.

Applying cross product on loading1 and loading2.



```
grunt> cross_command = cross loading1,loading2;
grunt> dump cross_command;
```

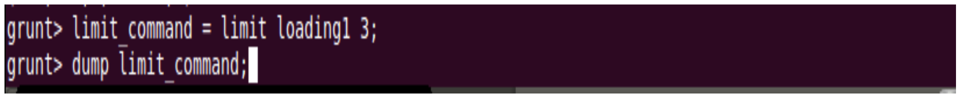
CROSS Result:



```
(fred,cnn,12,crap,2)
(amr,flickr,10,crap,2)
(fred,cnn,12,myblog,7)
(amr,flickr,10,myblog,7)
(amr,flickr,10,cnn,9)
(amr,flickr,10,flickr,9)
(fred,cnn,12,cnn,9)
(fred,cnn,12,flickr,9)
(amr,myblog,10,crap,2)
(amr,myblog,10,myblog,7)
(amr,myblog,10,cnn,9)
(amr,myblog,10,flickr,9)
(amr,cnn,8,crap,2)
(amr,crap,8,crap,2)
(amr,cnn,8,myblog,7)
(amr,crap,8,myblog,7)
(amr,crap,8,cnn,9)
(amr,crap,8,flickr,9)
(amr,cnn,8,cnn,9)
(amr,cnn,8,flickr,9)
grunt>
```

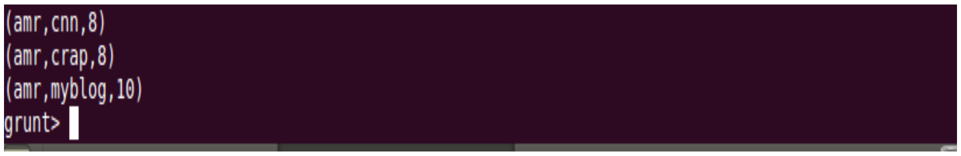
LIMIT:

LIMIT operator is used to limit the number of output tuples. If the specified number of output tuples is equal to or exceeds the number of tuples in the relation, the output will include all tuples in the relation.



```
grunt> limit_command = limit loading1 3;
grunt> dump limit_command;
```

LIMIT Result:

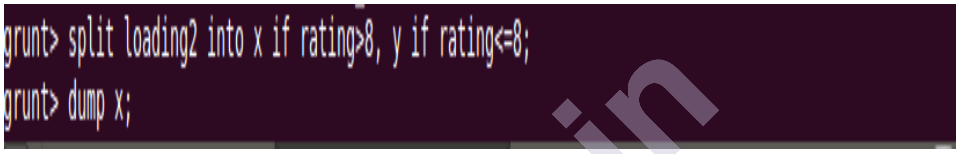


```
(amr,cnn,8)
(amr,crap,8)
(amr,myblog,10)
grunt>
```

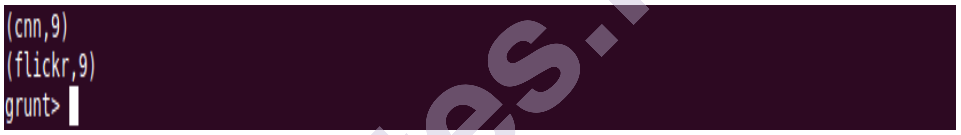
SPLIT:

SPLIT operator is used to partition the contents of a relation into two or more relations based on some expression. Depending on the conditions stated in the expression.

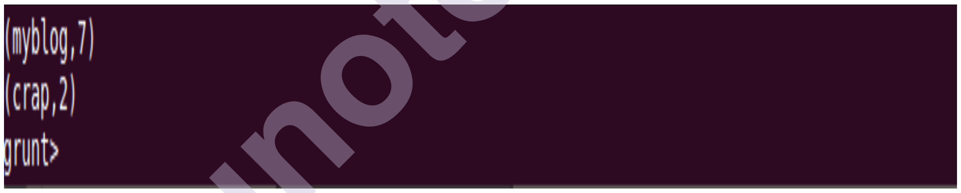
Split the loading2 into two relations x and y. x relation created by loading2 contain the fields that the rating is greater than 8 and y relation contain fields that rating is less than or equal to 8.



```
grunt> split loading2 into x if rating>8, y if rating<=8;
grunt> dump x;
```



```
(cnn,9)
(flickr,9)
grunt>
```



```
(myblog,7)
(crap,2)
grunt>
```

5.3 REFERENCES

- “The Visual Display of Quantitative Information” by Edward R. ...
- “Storytelling With Data: A Data Visualization Guide for Business Professionals” by Cole Nussbaumer Knaflic.
- “Data Visualization – A Practical Introduction” by Kieran Healy.

5.4 UNIT END EXERCISES

Create Your First Apache Pig Script.

❄❄❄❄❄❄❄

6

SPARK

Unit Structure :

6.0 Introduction

6.1 Spark

6.2 Spark RDD – Operations

6.3 Create an RDD in Apache Spark

6.4 SPARK IN-Memory Computing

6.5 Lazy Evaluation In Apache Spark

6.6 RDD Persistence and Caching in Spark

6.7 Lab Sessions

6.8 Exercises

6.9 Questions

6.10 Quiz

6.11 Video Lectures

6.12 Moocs

6.13 References

6.0 INTRODUCTION

Spark is a unified analytics engine for large-scale data processing including built-in modules for SQL, streaming, machine learning and graph processing.

6.1 SPARK

Apache Spark is an open-source cluster computing framework. Its primary purpose is to handle the real-time generated data. Spark was built on the top of the Hadoop MapReduce. It was optimized to run in memory whereas alternative approaches like Hadoop's MapReduce writes data to and from computer hard drives. Apache Spark RDD makes developer's work more efficient, as it divides cluster into nodes to compute parallel operations on each node. Before anything else, we will go through the brief introduction of Spark RDD.

Introduction to Apache Spark RDD

Apache Spark RDDs (Resilient Distributed Datasets) are a basic abstraction of spark which is immutable. These are logically partitioned that we can also apply parallel operations on them.

We can create RDD in three ways:

1. Parallelizing already existing collection in driver program.
2. Referencing a dataset in an external storage system (e.g. HDFS, Hbase, shared file system).
3. Creating RDD from already existing RDDs.

Prominent Features

There are following traits of Resilient distributed datasets. Those are list-up below:

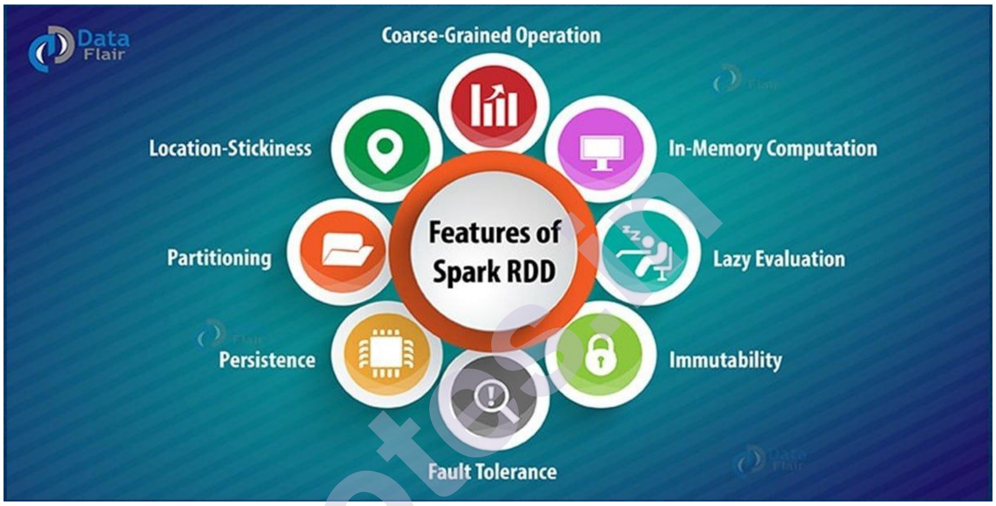


Fig 1. Spark Prominent Features

i. **In-memory computation**

The data inside RDD are stored in memory for as long as you want to store. Keeping the data in-memory improves the performance by an order of magnitudes.

ii. **Lazy Evaluation**

The data inside RDDs are not evaluated on the go. The changes or the computation is performed only after an action is triggered. Thus, it limits how much work it has to do.

iii. **Fault Tolerance**

Upon the failure of worker node, using lineage of operations we can re-compute the lost partition of RDD from the original one. Thus, we can easily recover the lost data.

iv. **Immutability**

RDDS are immutable in nature meaning once we create an RDD we can not manipulate it. And if we perform any transformation, it creates new RDD. We achieve consistency through immutability.

v. **Persistence**

We can store the frequently used RDD in in-memory and we can also retrieve them directly from memory without going to disk, this speedup the execution. We can perform Multiple operations on the same data, this happens by storing the data explicitly in memory by calling persist() or cache() function. Follow this guide for the detailed study of RDD persistence in Spark.

vi. Partitioning

RDD partition the records logically and distributes the data across various nodes in the cluster. The logical divisions are only for processing and internally it has no division. Thus, it provides parallelism.

vii. Parallel

Rdd, process the data parallelly over the cluster.

viii. Location-Stickiness

RDDs are capable of defining placement preference to compute partitions. Placement preference refers to information about the location of RDD. The DAGScheduler places the partitions in such a way that task is close to data as much as possible. Thus speed up computation.

ix. Coarse-grained Operation

We apply coarse-grained transformations to RDD. Coarse-grained meaning the operation applies to the whole dataset not on an individual element in the data set of RDD.

x. Typed

We can have RDD of various types like: RDD [int], RDD [long], RDD [string].

xi. No limitation

We can have any number of RDD. There is no limit to its number. The limit depends on the size of disk and memory.

Limitations of Apache Spark – Ways to Overcome Spark Drawbacks

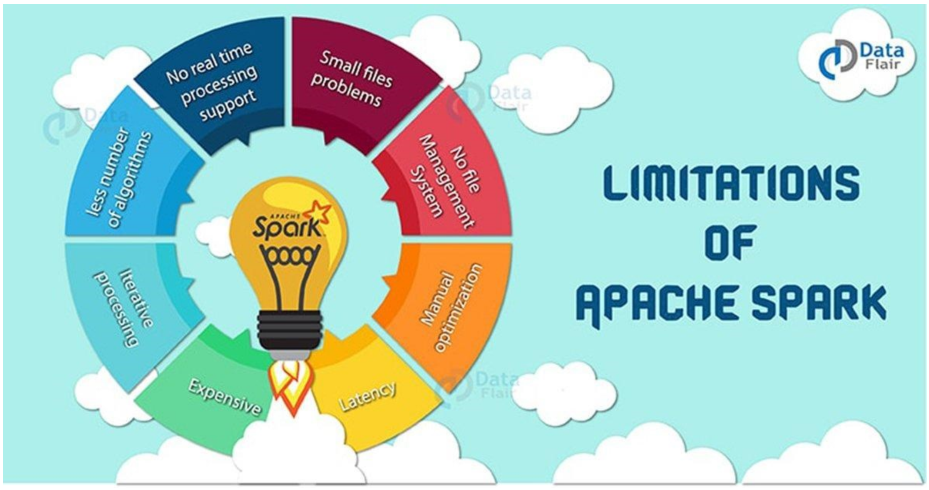

No real time processing support
Small files problems
No file Management System
Manual optimization
Latency
Expensive
Iterative processing
less number of algorithms
Apache Spark
Data Flair
LIMITATIONS OF APACHE SPARK

Fig 2. Limitations of Apache Spark

1. Objective

Some of the drawbacks of Apache Spark are there is no support for real-time processing, Problem with small file, no dedicated File management system, Expensive and much more due to these limitations of Apache Spark, industries have started shifting to Apache Flink– 4G of Big Data.

2. Limitations of Apache Spark

As we know Apache Spark is the next Gen Big data tool that is being widely used by industries but there are certain limitations of Apache Spark due to which industries have started shifting to Apache Flink– 4G of Big Data.

a. No Support for Real-time Processing

In Spark Streaming, the arriving live stream of data is divided into batches of the pre-defined interval, and each batch of data is treated like Spark Resilient Distributed Database (RDDs). Then these RDDs are processed using the operations like map, reduce, join etc. The result of these operations is returned in batches. Thus, it is not real time processing but Spark is near real-time processing of live data. Micro-batch processing takes place in Spark Streaming.

b. Problem with Small File

If we use Spark with Hadoop, we come across a problem of a small file. HDFS provides a limited number of large files rather than a large number of small files. Another place where Spark legs behind is we store the data gzipped in S3. This pattern is very nice except when there are lots of small gzipped files. Now the work of the Spark is to keep those files on network and uncompress them. The gzipped files can be uncompressed only if the entire file is on one core.

In the resulting RDD, each file will become a partition; hence there will be a large amount of tiny partition within an RDD. Now if we want efficiency in our processing, the RDDs should be repartitioned into some manageable format. This requires extensive shuffling over the network.

c. No File Management System

Apache Spark does not have its own file management system, thus it relies on some other platform like Hadoop or another cloud-based platform which is one of the Spark known issues.

d. Expensive

In-memory capability can become a bottleneck when we want cost-efficient processing of big data as keeping data in memory is quite expensive, the memory consumption is very high, and it is not handled in a user-friendly manner. Apache Spark requires lots of RAM to run in-memory, thus the cost of Spark is quite high.

e. Less number of Algorithms

Spark MLlib lags behind in terms of a number of available algorithms like Tanimoto distance.

f. Manual Optimization

The Spark job requires to be manually optimized and is adequate to specific datasets. If we want to partition and cache in Spark to be correct, it should be controlled manually.

g. Iterative Processing

In Spark, the data iterates in batches and each iteration is scheduled and executed separately.

h. Latency

Apache Spark has higher latency as compared to Apache Flink.

i. Window Criteria

Spark does not support record based window criteria. It only has time-based window criteria.

j. Back Pressure Handling

Back pressure is build up of data at an input-output when the buffer is full and not able to receive the additional incoming data. No data is transferred until the buffer is empty.

6.2 SPARK RDD – OPERATIONS

We can perform different operations on RDD as well as on data storage to form another RDDs from it. There are two different operations:

- Spark RDD - Spark RDD operation
- Spark RDD – Transformation and Action
- TRANSFORMATION
- ACTION

To modify the available datasets, we need to provide step by step instructions to the spark. Those steps must clearly explain what changes we want. That set of instructions is generally known as Transformations".

Transformations are operations on RDDs which results in new RDD such as Map, Filter.

Actions are operations, triggers the process by returning the result back to program. Transformation and actions work differently.

1. Transformation

Transformation is a process of forming new RDDs from the existing ones. Transformation is a user specific function. It is a process of changing the current dataset in the dataset we want to have.

Some common transformations supported by Spark are:

For example, Map(func), Filter(func), Mappartitions (func), Flatmap (func) etc.

All transformed RDDs are lazy in nature. As we are already familiar with the term "Lazy Evaluations". That means it does not produce their results instantly. However, we always require an action to complete the computation.

To trigger the execution, an action is a must. Up to that action data inside RDD is not transformed or available.

After transformation, you incrementally build the lineage. That lineage is which formed by the entire parent RDDs of final RDDs. As soon as the execution process ends, resultant RDDs will be completely different from their parent RDDs.

They can be smaller (e.g. filter, count, distinct, sample), bigger (e.g. flatMap, union, Cartesian) or the same size (e.g. map).

Transformation can categorize further as: Narrow Transformations, Wide Transformations.

a. Narrow Transformations

Narrow transformations are the result of map() and filter() functions and these compute data that live on a single partition meaning there will not be any data movement between partitions to execute narrow transformations.

An output RDD also has partitions with records. In the parent RDD, that output originates from a single partition. Additionally, to calculate the result Only a limited subset of partitions is used.

In Apache spark narrow transformations groups as a stage. That process is mainly known as pipelining. Pipelining is an implementation mechanism. Functions such as map(), mapPartition(), flatMap(), filter(), union() are some examples of narrow transformation.

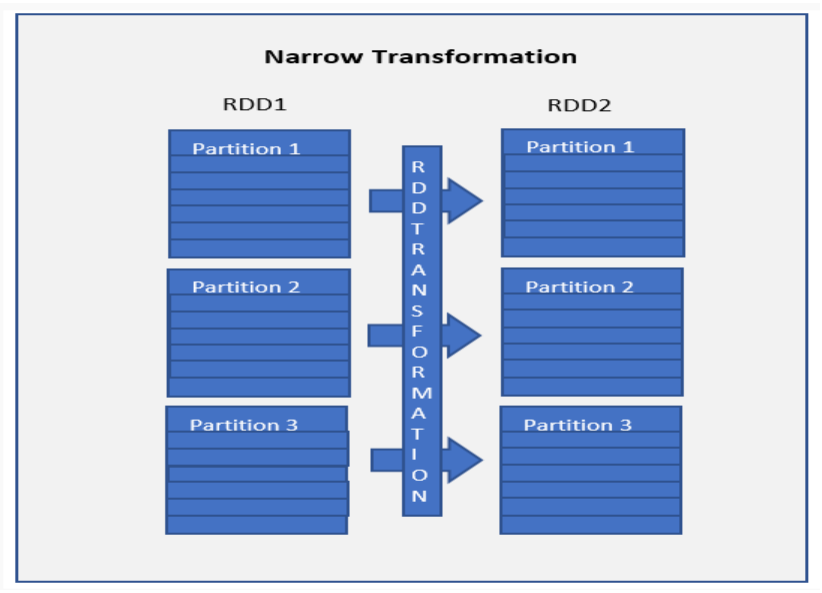


Fig 3. Narrow Transformation

b. Wide Transformations

Wide transformations are the result of groupByKey (func) and reduceByKey (func). As data may reside in many partitions of the parent RDD. These are used to compute the records by data in the single partition. Wide transformations may also know as shuffle transformations.

Functions such as groupByKey(), aggregateByKey(), aggregate(), join(), repartition() are some examples of a wider transformations.

Note: When compared to Narrow transformations, wider transformations are expensive operations due to shuffling.

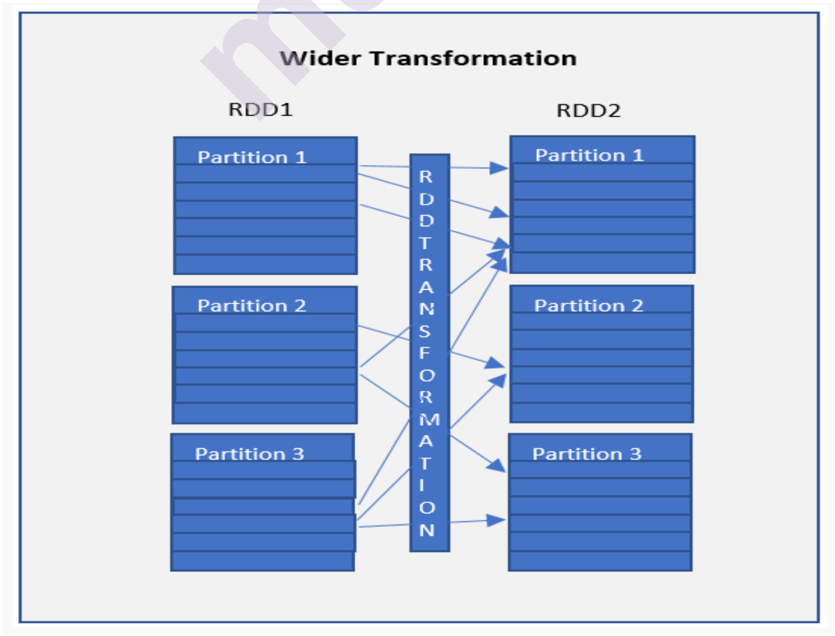


Fig 4. Wider Transformation

TRANSFORMATION METHODS	METHOD USAGE AND DESCRIPTION
cache()	Caches the RDD
filter()	Returns a new RDD after applying filter function on source dataset.
flatMap()	Returns flattern map meaning if you have a dataset with array, it converts each elements in a array as a row. In other words it return 0 or more items in output for each element in dataset.
map()	Applies transformation function on dataset and returns same number of elements in distributed dataset.
mapPartitions()	Similar to map, but executs transformation function on each partition, This gives better performance than map function
mapPartitionsWithIndex()	Similar to map Partitions, but also provides func with an integer value representing the index of the partition.
randomSplit()	Splits the RDD by the weights specified in the argument. For example rdd.randomSplit(0.7,0.3)
union()	Comines elements from source dataset and the argument and returns combined dataset. This is similar to union function in Math set operations.
sample()	Returns the sample dataset.
intersection()	Returns the dataset which contains elements in both source dataset and an argument
distinct()	Returns the dataset by eliminating all duplicated elements.
repartition()	Return a dataset with number of partition specified in the argument. This operation reshuffles the RDD randamly, It could either return lesser or more partioned RDD based on the input supplied.
coalesce()	Similar to repartition by operates better when we want to the decrease the partitions. Betterment acheives by reshuffling the data from fewer nodes compared with all nodes by repartition.

Spark RDD Transformations complete example

```
package com.sparkbyexamples.spark.rdd
import org.apache.spark.rdd.RDD
import org.apache.spark.sql.SparkSession
object WordCountExample {
  def main(args:Array[String]): Unit = {
    val spark:SparkSession = SparkSession.builder()
      .master("local[3]")
      .appName("SparkByExamples.com")
      .getOrCreate()
    val sc = spark.sparkContext
    val rdd:RDD[String] = sc.textFile("src/main/resources/test.txt")
    println("initial partition count:"+rdd.getNumPartitions)
    val reparRdd = rdd.repartition(4)
    println("re-partition count:"+reparRdd.getNumPartitions)
    //rdd.coalesce(3)
    rdd.collect().foreach(println)
    // rdd flatMap transformation
    val rdd2 = rdd.flatMap(f=>f.split(" "))
    rdd2.foreach(f=>println(f))
    //Create a Tuple by adding 1 to each word
    val rdd3:RDD[(String,Int)]= rdd2.map(m=>(m,1))
    rdd3.foreach(println)
    //Filter transformation
    val rdd4 = rdd3.filter(a=> a._1.startsWith("a"))
    rdd4.foreach(println)
    //ReduceBy transformation
    val rdd5 = rdd3.reduceByKey(_ + _)
    rdd5.foreach(println)
    //Swap word,count and sortByKey transformation
    val rdd6 = rdd5.map(a=>(a._2,a._1)).sortByKey()
    println("Final Result")
    //Action - foreach
    rdd6.foreach(println)
    //Action - count
    println("Count : "+rdd6.count())
    //Action - first
    val firstRec = rdd6.first()
    println("First Record : "+firstRec._1 + ","+ firstRec._2)
    //Action - max
    val datMax = rdd6.max()
    println("Max Record : "+datMax._1 + ","+ datMax._2)
    //Action - reduce
    val totalWordCount = rdd6.reduce((a,b) => (a._1+b._1,a._2))
    println("dataReduce Record : "+totalWordCount._1)
    //Action - take
    val data3 = rdd6.take(3)
    data3.foreach(f=>{
      println("data3 Key:"+ f._1 +", Value:"+f._2)
    })
    //Action - collect
    val data = rdd6.collect()
    data.foreach(f=>{
      println("Key:"+ f._1 +", Value:"+f._2)
    })
    //Action - saveAsTextFile
    rdd5.saveAsTextFile("c:/tmp/wordCount")
  }
}
```

2. Actions

An action is an operation, triggers execution of computations and RDD transformations. Also, returns the result back to the storage or its program. Transformation returns new RDDs and actions returns some other data types. Actions give non-RDD values to the RDD operations.

It forces the evaluation of the transformation process need for the RDD they may call on. Since they actually need to produce output. An action instructs Spark to compute a result from a series of transformations.

Actions are one of two ways to send data from executors to the driver. Executors are agents that are responsible for executing different tasks. While a driver coordinates execution of tasks.

Transformations create RDDs from each other, but when we want to work with the actual dataset, at that point action is performed. When the action is triggered after the result, new RDD is not formed like transformation. Thus, Actions are Spark RDD operations that give non-RDD values. The values of action are stored to drivers or to the external storage system. It brings laziness of RDD into motion.

An action is one of the ways of sending data from Executer to the driver. Executors are agents that are responsible for executing a task. While the driver is a JVM process that coordinates workers and execution of the task. Some of the actions of Spark are:

count()

Action count() returns the number of elements in RDD.

For example, RDD has values {1, 2, 2, 3, 4, 5, 5, 6} in this RDD "rdd.count()" will give the result 8.

Count() example:

```
[php]val data = spark.read.textFile("spark_test.txt").rdd

val mapFile = data.flatMap(lines => lines.split(" ")).filter(value => value=="spark")

println(mapFile.count())[/php]
```

Note – In above code flatMap() function maps line into words and count the word "Spark" using count() Action after filtering lines containing "Spark" from mapFile.

collect()

The action collect() is the common and simplest operation that returns our entire RDDs content to driver program. The application of collect() is unit testing where the entire RDD is expected to fit in memory. As a result, it makes easy to compare the result of RDD with the expected result.

Action Collect() had a constraint that all the data should fit in the machine, and copies to the driver.

Collect() example:

```
[php]val data =
spark.sparkContext.parallelize(Array(('A',1),('b',2),('c',3)))

val data2
=spark.sparkContext.parallelize(Array(('A',4),('A',6),('b',7),('c',3),('c',8)
))

val result = data.join(data2)

println(result.collect().mkString(","))[/php]
```

Note – join() transformation in above code will join two RDDs on the basis of same key(alphabet). After that collect() action will return all the elements to the dataset as an Array.

take(n)

The action take(n) returns n number of elements from RDD. It tries to cut the number of partition it accesses, so it represents a biased collection. We cannot presume the order of the elements.

For example, consider RDD {1, 2, 2, 3, 4, 5, 5, 6} in this RDD "take (4)" will give result { 2, 2, 3, 4}

Take() example:

[php]val data = spark.sparkContext.parallelize(Array(('k',5),('s',3),('s',4),('p',7),('p',5),('t',8),('k',6)),3)

val group = data.groupByKey().collect()

val twoRec = result.take(2)

twoRec.foreach(println)[/php]

Note – The take(2) Action will return an array with the first n elements of the data set defined in the taking argument.

top()

If ordering is present in our RDD, then we can extract top elements from our RDD using top(). Action top() use default ordering of data.

Top() example:

[php]val data = spark.read.textFile("spark_test.txt").rdd

val mapFile = data.map(line => (line,line.length))

val res = mapFile.top(3)

res.foreach(println)[/php]

Note – map() operation will map each line with its length. And top(3) will return 3 records from mapFile with default ordering.

countByValue()

The countByValue() returns, many times each element occur in RDD.

For example, RDD has values {1, 2, 2, 3, 4, 5, 5, 6} in this RDD "rdd.countByValue()" will give the result {(1,1), (2,2), (3,1), (4,1), (5,2), (6,1)}

countByValue() example:

[php]val data = spark.read.textFile("spark_test.txt").rdd

val result= data.map(line => (line,line.length)).countByValue()

result.foreach(println)[/php]

Note – The countByValue() action will return a hashmap of (K, Int) pairs with the count of each key.

reduce()

The reduce() function takes the two elements as input from the RDD and then produces the output of the same type as that of the input elements. The simple forms of such function are an addition. We can add the elements of RDD, count the number of words. It accepts commutative and associative operations as an argument.

Reduce() example:

[php]val rdd1 = spark.sparkContext.parallelize(List(20,32,45,62,8,5))

val sum = rdd1.reduce(_+_)

println(sum)[/php]

Note – The reduce() action in above code will add the elements of the source RDD.

fold()

The signature of the fold() is like reduce(). Besides, it takes "zero value" as input, which is used for the initial call on each partition. But, the condition with zero value is that it should be the identity element of that operation. The key difference between fold() and reduce() is that, reduce() throws an exception for empty collection, but fold() is defined for empty collection.

For example, zero is an identity for addition; one is identity element for multiplication. The return type of fold() is same as that of the element of RDD we are operating on.

For example, rdd.fold(0)((x, y) => x + y).

Fold() example:

[php]val rdd1 = spark.sparkContext.parallelize(List(("maths", 80),("science", 90)))

val additionalMarks = ("extra", 4)

val sum = rdd1.fold(additionalMarks){ (acc, marks) => val add = acc._2 + marks._2

("total", add)

}

println(sum)[/php]

Note – In above code additionalMarks is an initial value. This value will be added to the int value of each record in the source RDD.

aggregate()

It gives us the flexibility to get data type different from the input type. The aggregate() takes two functions to get the final result. Through one function we combine the element from our RDD with the accumulator, and the second, to combine the accumulator. Hence, in aggregate, we supply the initial zero value of the type which we want to return.

foreach()

When we have a situation where we want to apply operation on each element of RDD, but it should not return value to the driver. In this case, foreach() function is useful. For example, inserting a record into the database.

Foreach() example:

```
[php]val data =
spark.sparkContext.parallelize(Array((‘k’,5),(‘s’,3),(‘s’,4),(‘p’,7),(‘p’,5),(‘
t’,8),(‘k’,6)),3)

val group = data.groupByKey().collect()

group.foreach(println)[/php]
```

Note – The foreach() action run a function (println) on each element of the dataset group.

6.3 CREATE AN RDD IN APACHE SPARK

1. Spark create RDD from Seq or List (using Parallelize)
2. Creating an RDD from a text file
3. Creating from another RDD
4. Creating from existing DataFrames and DataSet

Spark Create RDD from Seq or List (using Parallelize)

RDD’s are generally created by parallelized collection i.e. by taking an existing collection from driver program (scala, python e.t.c) and passing it to SparkContext’s parallelize() method. This method is used only for testing but not in realtime as the entire data will reside on one node which is not ideal for production.

```
val rdd=spark.sparkContext.parallelize(Seq(("Java", 20000),
  ("Python", 100000), ("Scala", 3000)))
rdd.foreach(println)
```

Outputs:

```
(Python,100000)
(Scala,3000)
(Java,20000)
```

Create an RDD from a text file

Mostly for production systems, we create RDD’s from files. here will see how to create an RDD by reading data from a file.

```
val rdd = spark.sparkContext.textFile("/path/textFile.txt")
```

This creates an RDD for which each record represents a line in a file.

If you want to read the entire content of a file as a single record use wholeTextFiles() method on sparkContext.

```
val rdd2 = spark.sparkContext.wholeTextFiles("/path/textFile.txt")

rdd2.foreach(record=>println("FileName : "+record._1+", FileContents :"+record._2))
```

In this case, each text file is a single record. In this, the name of the file is the first column and the value of the text file is the second column.

Creating from another RDD

You can use transformations like map, flatmap, filter to create a new RDD from an existing one.

```
val rdd3 = rdd.map(row=>{(row._1,row._2+100)})
```

Above, creates a new RDD “rdd3” by adding 100 to each record on RDD. this example outputs below.

```
(Python,100100)

(Scala,3100)

(Java,20100)
```

From existing DataFrames and DataSet

To convert DataSet or DataFrame to RDD just use rdd() method on any of these data types.

```
val myRdd2 = spark.range(20).toDF().rdd
```

toDF() creates a DataFrame and by calling rdd on DataFrame returns back RDD.

6.4 SPARK IN-MEMORY COMPUTING

The data is kept in random access memory(RAM) instead of some slow disk drives and is processed in parallel. Using this we can detect a pattern, analyze large data. This has become popular because it reduces the cost of memory. So, in-memory processing is economic for applications. The two main columns of in-memory computation are-

- ➢ RAM storage
- ➢ Parallel distributed processing.

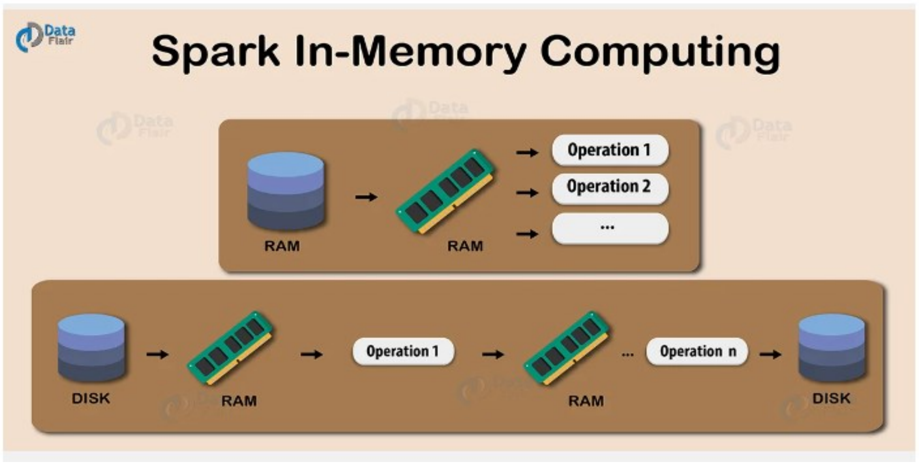


Fig 5. Spark In-Memory Computing

Keeping the data in-memory improves the performance by an order of magnitudes. The main abstraction of Spark is its RDDs. And the RDDs are cached using the cache() or persist() method.

When we use cache() method, all the RDD stores in-memory. When RDD stores the value in memory, the data that does not fit in memory is either recalculated or the excess data is sent to disk. Whenever we want RDD, it can be extracted without going to disk. This reduces the space-time complexity and overhead of disk storage.

The in-memory capability of Spark is good for machine learning and micro-batch processing. It provides faster execution for iterative jobs.

When we use persist() method the RDDs can also be stored in-memory, we can use it across parallel operations. The difference between cache() and persist() is that using cache() the default storage level is MEMORY_ONLY while using persist() we can use various storage levels.

Storage levels of RDD Persist() in Spark

The various storage level of persist() method in Apache Spark RDD are:

- ❖ MEMORY_ONLY
- ❖ MEMORY_AND_DISK
- ❖ MEMORY_ONLY_SER
- ❖ MEMORY_AND_DISK_SER
- ❖ DISK_ONLY
- ❖ MEMORY_ONLY_2 and MEMORY_AND_DISK_2

MEMORY_ONLY

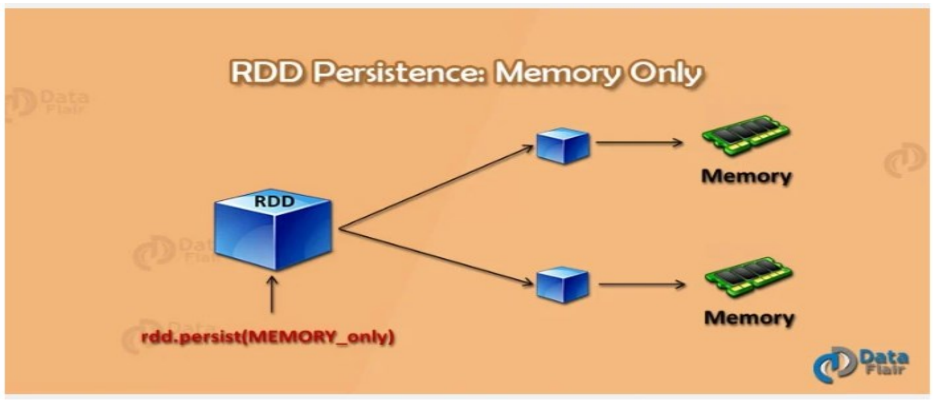


Fig 6. Spark storage level – memory only

In this storage level Spark, RDD store as deserialized JAVA object in JVM. If RDD does not fit in memory, then the remaining will re-compute each time they are needed.

MEMORY_AND_DISK

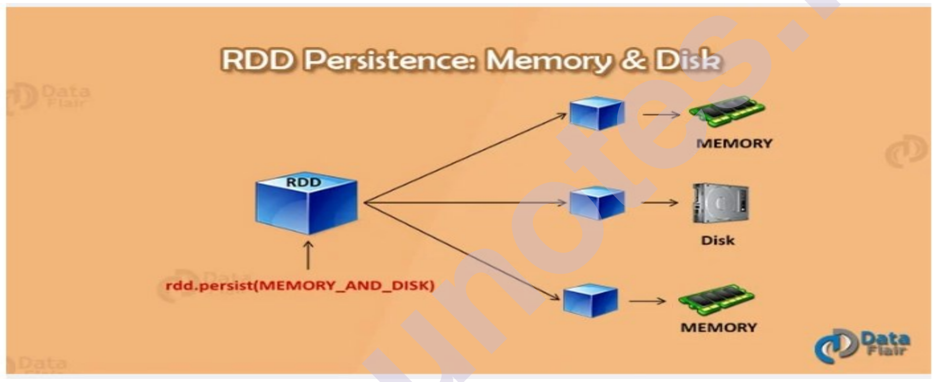


Fig 7. Spark storage level-memory and disk

In this level, RDD is stored as deserialized JAVA object in JVM. If the full RDD does not fit in memory then the remaining partition is stored on disk, instead of recomputing it every time when it is needed.

MEMORY_ONLY_SER

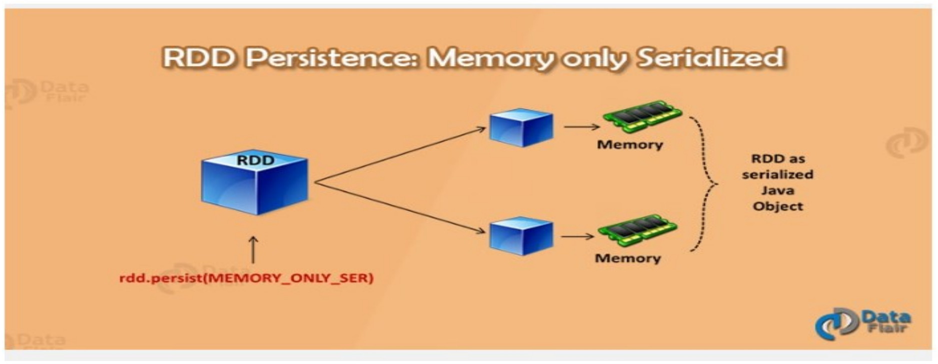

RDD Persistence: Memory only Serialized
RDD
rdd.persist(MEMORY_ONLY_SER)
Memory
Memory
RDD as serialized Java Object

Fig 8. Spark storage level – memory only serialized

This level stores RDDs as serialized JAVA object. It stores one-byte array per partition. It is like MEMORY_ONLY but is more space efficient especially when we use fast serializer.

MEMORY_AND_DISK_SER

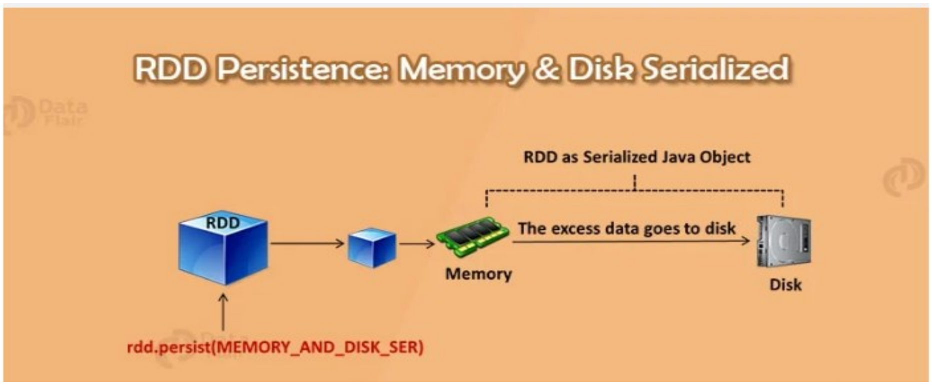

RDD Persistence: Memory & Disk Serialized
RDD as Serialized Java Object
RDD
The excess data goes to disk
Memory
Disk
rdd.persist(MEMORY_AND_DISK_SER)

Fig 9. Spark storage level – memory and disk serialized

This level stores RDD as serialized JAVA object. If the full RDD does not fit in the memory then it stores the remaining partition on the disk, instead of recomputing it every time when we need.

DISK_ONLY

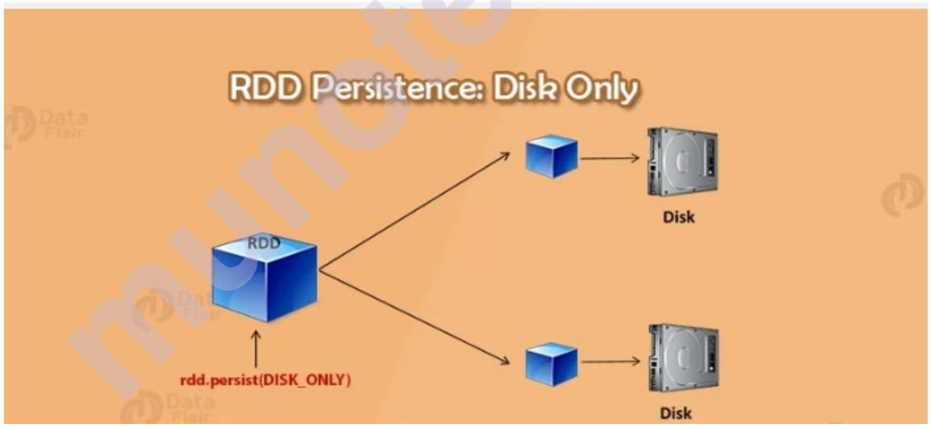


Fig 10. Spark storage level-disk-only

This storage level stores the RDD partitions only on disk.

MEMORY_ONLY_2 and MEMORY_AND_DISK_2

It is like MEMORY_ONLY and MEMORY_AND_DISK. The only difference is that each partition gets replicate on two nodes in the cluster.

Advantages of In-memory Processing

After studying Spark in-memory computing introduction and various storage levels in detail, let's discuss the advantages of in-memory computation-

1. When we need a data to analyze it is already available on the go or we can retrieve it easily.
2. It is good for real-time risk management and fraud detection.
3. The data becomes highly accessible.
4. The computation speed of the system increases.
5. Improves complex event processing.
6. Cached a large amount of data.
7. It is economic, as the cost of RAM has fallen over a period of time.

6.5 LAZY EVALUATION IN APACHE SPARK

Llazy evaluation in Spark means that the execution will not start until an action is triggered. In Spark, the picture of lazy evaluation comes when Spark transformations occur.

Transformations are lazy in nature meaning when we call some operation in RDD, it does not execute immediately. Spark maintains the record of which operation is being called(Through DAG).

Apache Spark Lazy Evaluation Feature.

Apache Spark Lazy Evaluation Explanation.

In MapReduce, much time of developer wastes in minimizing the number of MapReduce passes. It happens by clubbing the operations together. While in Spark we do not create the single execution graph, rather we club many simple operations. Thus it creates the difference between Hadoop MapReduce vs Apache Spark.

In Spark, driver program loads the code to the cluster. When the code executes after every operation, the task will be time and memory consuming. Since each time data goes to the cluster for evaluation.

Advantages of Lazy Evaluation in Spark Transformation

There are some benefits of Lazy evaluation in Apache Spark-

a. Increases Manageability

By lazy evaluation, users can organize their Apache Spark program into smaller operations. It reduces the number of passes on data by grouping operations.

b. Saves Computation and increases Speed

Spark Lazy Evaluation plays a key role in saving calculation overhead. Since only necessary values get compute. It saves the trip between driver and cluster, thus speeds up the process.

c. Reduces Complexities

The two main complexities of any operation are time and space complexity. Using Apache Spark lazy evaluation we can overcome both. Since we do not execute every operation, Hence, the time gets saved. It let us work with an infinite data structure. The action is triggered only when the data is required, it reduces overhead.

d. Optimization

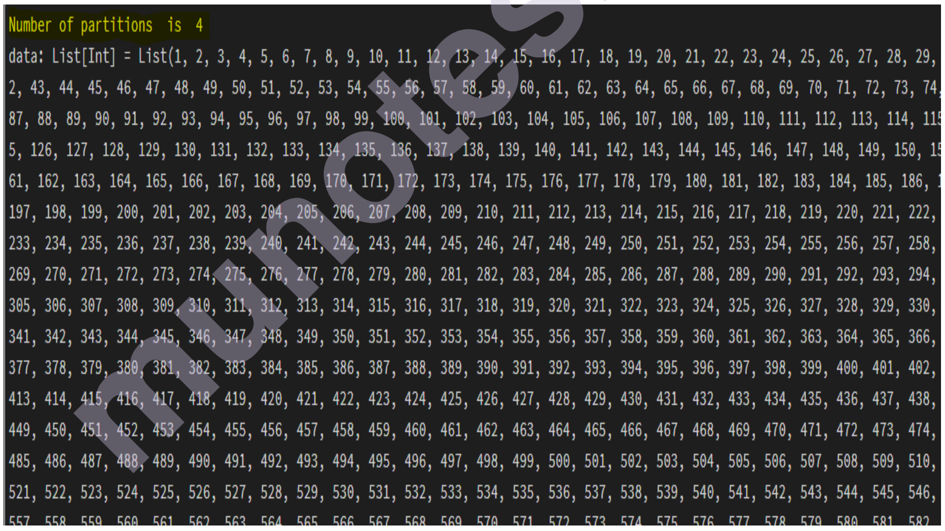
It provides optimization by reducing the number of queries. Learn more about Apache Spark Optimization.

Spark performs lazy evaluation

Example 1

In the first step, we created a list of 10 million numbers and made an RDD with four partitions below. And we can see the result in the below output image.

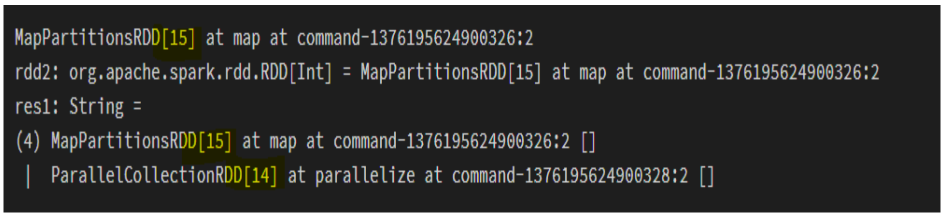
val data = (1 to 100000).toList val rdd = sc.parallelize(data,4) println("Number of partitions is "+rdd.getNumPartitions)



```
Number of partitions  is  4
data: List[Int] = List(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29,
2, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74,
87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115
5, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 15
61, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 1
197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222,
233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258,
269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294,
305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330,
341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366,
377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402,
413, 414, 415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438,
449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472, 473, 474,
485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510,
521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546,
557, 558, 559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 582,
```

Next, we will perform a fundamental transformation, like adding 4 to each number. Note that Spark, at this point, has not started any transformation. It only records a series of transformations in the form of RDD Lineage. You can see that RDD lineage using the function toDebugString

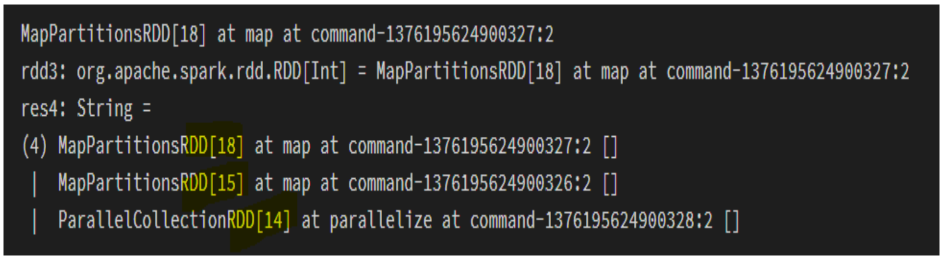
//Adding 5 to each value in rdd val rdd2 = rdd.map(x => x+5) //rdd2 objetc println(rdd2) //getting rdd lineage rdd2.toDebugString



```
MapPartitionsRDD[15] at map at command-1376195624900326:2
rdd2: org.apache.spark.rdd.RDD[Int] = MapPartitionsRDD[15] at map at command-1376195624900326:2
res1: String =
(4) MapPartitionsRDD[15] at map at command-1376195624900326:2 []
 |  ParallelCollectionRDD[14] at parallelize at command-1376195624900328:2 []
```

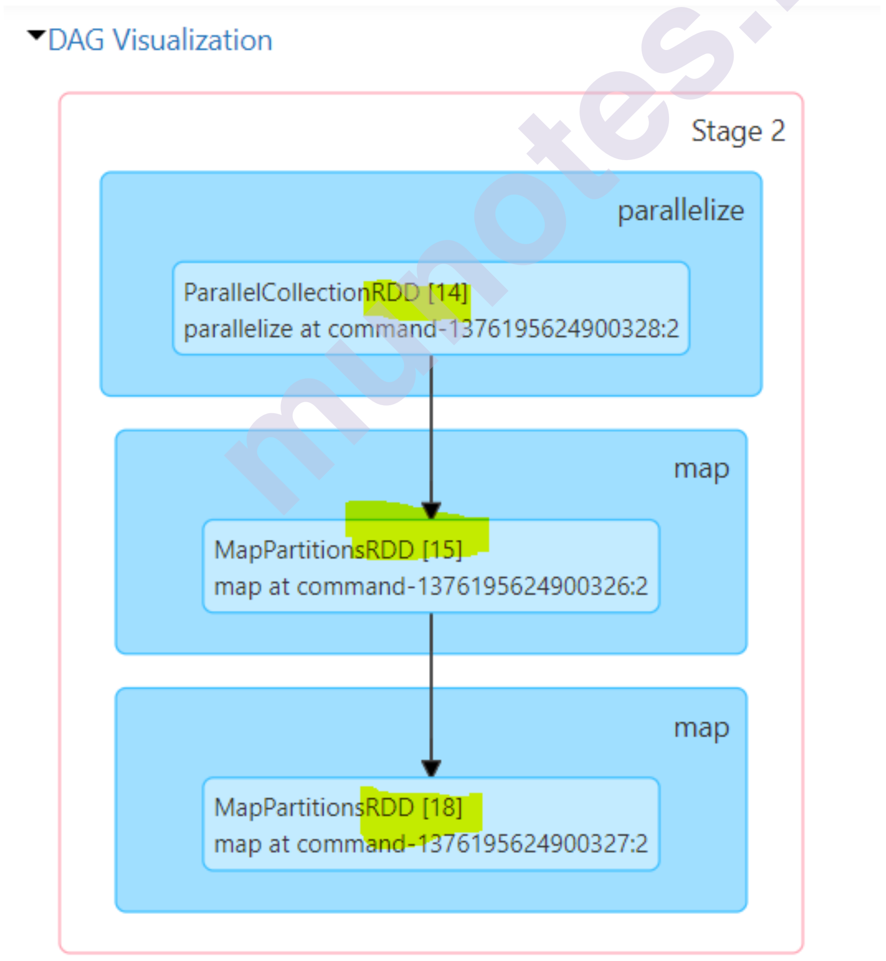
Now if you observe MapPartitionsRDD[15] at map is dependent on ParallelCollectionRDD[14]. Now, let's go ahead and add one more transformation to add 20 to all the elements of the list.

//Adding 5 to each value in rdd val rdd3 = rdd2.map(x => x+20) //rdd2 objetc println(rdd3) //getting rdd lineage rdd3.toDebugString rdd3.collect



```
MapPartitionsRDD[18] at map at command-1376195624900327:2
rdd3: org.apache.spark.rdd.RDD[Int] = MapPartitionsRDD[18] at map at command-1376195624900327:2
res4: String =
(4) MapPartitionsRDD[18] at map at command-1376195624900327:2 []
 |  MapPartitionsRDD[15] at map at command-1376195624900326:2 []
 |  ParallelCollectionRDD[14] at parallelize at command-1376195624900328:2 []
```

Now if you observe MapPartitionsRDD[18] at map is dependent on MapPartitionsRDD[15] and ParallelCollectionRDD[14]. Now, let's go ahead and add one more transformation to add 20 to all the elements of the list. After calling a action using collect we see that three stages of DAG lineage at ParallelCollectionRDD[14], MapPartitionsRDD[15] and MapPartitionsRDD[18].


DAG Visualization
Stage 2
parallelize
ParallelCollectionRDD [14]
parallelize at command-1376195624900328:2
map
MapPartitionsRDD [15]
map at command-1376195624900326:2
map
MapPartitionsRDD [18]
map at command-1376195624900327:2

From the above examples, we can able to understand that spark lineage is maintained using DAG.

Example 2

Here we see how unnecessary steps or transformations are skipped due to spark Lazy evaluation during the execution process.

import org.apache.spark.sql.functions.

_ val df1 = (1 to 100000).toList.toDF("col1")

//scenario 1

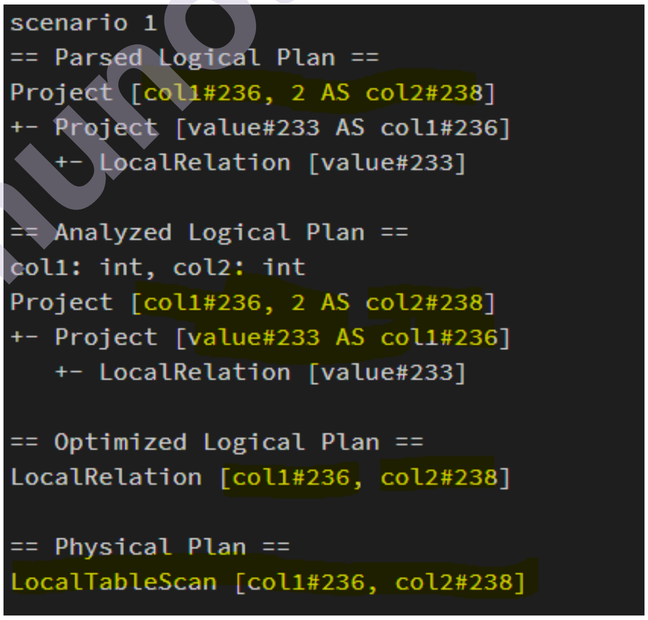
println("scenario 1")

df1.withColumn("col2",lit(2)).explain(true);

//scenario 2 println("scenario 2")

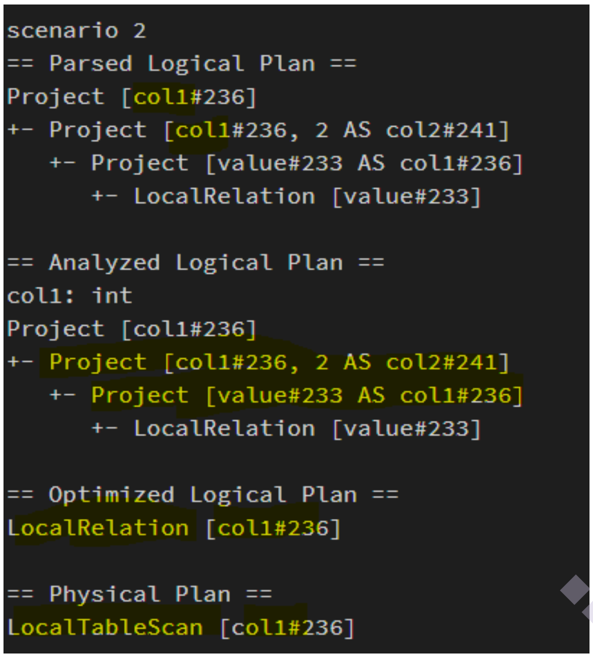
df1.withColumn("col2",lit(2)).drop("col2").explain(true);

In this, we created a dataframe with column "col1" at the very first step. If you observe Scenario-1, I have created a column "col2" using withColumn() function and after that applied explain() function to analyze the physical execution plan. The below image contains a logical plan, analyzed logical plan, optimized logical plan, and physical plan. In the analyzed logical plan, if you observe there is only one projection stage, Projection main indicates the columns moving forward for further execution.



```
scenario 1
== Parsed Logical Plan ==
Project [col1#236, 2 AS col2#238]
+- Project [value#233 AS col1#236]
   +- LocalRelation [value#233]

== Analyzed Logical Plan ==
col1: int, col2: int
Project [col1#236, 2 AS col2#238]
+- Project [value#233 AS col1#236]
   +- LocalRelation [value#233]

== Optimized Logical Plan ==
LocalRelation [col1#236, col2#238]

== Physical Plan ==
LocalTableScan [col1#236, col2#238]
```

If you observe Scenario-1, I have created a column "col2" using the withColumn() function, and we are dropping that column and after that applied explain() function to analyze the physical execution plan. The below image contains a logical plan, analyzed logical plan, optimized logical plan, and physical plan. In the analyzed logical plan, if you observe there are only two projection stages, Projection main indicates the columns moving forward for further execution. In finalized physical plan, there is no task of creation and of "col2".



```
scenario 2
== Parsed Logical Plan ==
Project [col1#236]
+- Project [col1#236, 2 AS col2#241]
   +- Project [value#233 AS col1#236]
      +- LocalRelation [value#233]

== Analyzed Logical Plan ==
col1: int
Project [col1#236]
+- Project [col1#236, 2 AS col2#241]
   +- Project [value#233 AS col1#236]
      +- LocalRelation [value#233]

== Optimized Logical Plan ==
LocalRelation [col1#236]

== Physical Plan ==
LocalTableScan [col1#236]
```

Advantages of Spark Lazy Evaluation

Users can divide the entire work into smaller operations for easy readability and management. But Spark internally groups the transformations reducing the number of passes on data. What this means is, if Spark could group two transformations into one, then it had to read the data only once to apply the transformations rather than reading twice. In one of the above examples[Scenario 2], we saw that only the necessary computation is done by Spark, which increases Speed.

As the action is triggered only when data is required, this reduces unnecessary overhead. Let's say a program does the following steps (i)Read a file, (ii)Does a function call unrelated to the file (iii)Loads the file into a table. If Spark read the file as soon as it met the first transformation, it had to wait for the function call to finish before it loads the data, and all this while the data had to sit in memory.

6.6 RDD PERSISTENCE AND CACHING IN SPARK

Spark RDD persistence is an optimization technique in which saves the result of RDD evaluation. Using this we save the intermediate result so that we can use it further if required. It reduces the computation overhead.

We can make persisted RDD through cache() and persist() methods. When we use the cache() method we can store all the RDD in-memory. We can persist the RDD in memory and use it efficiently across parallel operations.

The difference between cache() and persist() is that using cache() the default storage level is MEMORY_ONLY while using persist() we can use various storage levels (described below). It is a key tool for an interactive algorithm.

Because, when we persist RDD each node stores any partition of it that it computes in memory and makes it reusable for future use. This process speeds up the further computation ten times.

When the RDD is computed for the first time, it is kept in memory on the node. The cache memory of the Spark is fault tolerant so whenever any partition of RDD is lost, it can be recovered by transformation Operation that originally created it.

Persistence in Apache Spark

In Spark, we can use some RDD's multiple times. If honestly, we repeat the same process of RDD evaluation each time it required or brought into action. This task can be time and memory consuming, especially for iterative algorithms that look at data multiple times. To solve the problem of repeated computation the technique of persistence came into the picture.

Benefits of RDD Persistence in Spark

There are some advantages of RDD caching and persistence mechanism in spark. It makes the whole system

- Time efficient
- Cost efficient
- Lessen the execution time.

6.7 LAB SESSIONS

Installing Spark

To install Spark on your computer, perform the following steps:

- First ensure you have a JVM installed on your machine by typing java -version at a command line
- Go to http://spark.apache.org and click on "Download"
- Select release 1.3.0
- Select "Prebuilt for Hadoop 2.4 and later"
- Select "Direct download"
- Click on the Download Spark link

Unpacking Spark

- In your home directory, create a subdirectory spark in which to store Spark
- Move the downloaded file into this directory
- Unpack the file. If you are on a Unix system (Linux, Mac OS) type the following at the command

line: tar xvf spark-1.3.0-bin-hadoop2.4.tgz. If you are on Windows, you can download 7zip

from http://www.7zip.org and use it to extract the contents.

Downloading a sample dataset

As part of today's lab, we will be processing a large file containing approximately 4 million words.

- Download a compressed version of that file from http://jmg3.web.rice.edu/32big.zip using either wget or by navigating to that URL in your browser.
- Extract the 32big.txt file from that ZIP using 7zip or the unzip utility. You should store 32big.txt under the new directory you just created beneath your home directory, spark/spark-1.3.0-bin-hadoop2.4.

Running Spark

- cd into the new directory: cd spark/spark-1.3.0-bin-hadoop2.4
- You can now start an interactive session with Spark by calling the Spark Shell ./bin/spark-shell.

This will start a read-eval-print loop for Scala, similar to what you might have seen for many scripting languages, such as Python. You can try it out by evaluating some simple Scala expressions:

```
scala> 1 + 2

res0: Int = 3
```

As you type expressions at the prompt, the resulting values are displayed, along with new variables that they are bound to (in this case, res0), which you can use in subsequent expressions.

When invoking spark-shell, you can alter the number of cores and bytes of memory that Spark uses with the keyword local. For example, to invoke spark-shell with one core and 2GB of memory, we write:

```
./bin/spark-shell --master local[1] --driver-memory 2G
```

Interacting with Spark

If you are still in a Spark session, close it using Ctrl-D and create a new one with 1 CPU core and 2GB of memory:

```
./bin/spark-shell --master local[1] --driver-memory 2G
```

In your interactive session, you can interact with Spark via the global "spark context" variable sc. Try this out by creating a simple RDD from the text in the large 32big.txt file you downloaded:

```
scala> val textFile = sc.textFile("32big.txt")
```

You now have a handle on an RDD. The elements in the RDD correspond to the lines in the 32big.txt file.

We can find out the number of lines using count:

```
scala> textFile.count()
```

As shown in class, we can also use the map/reduce pattern to count the number of occurences of each word in the RDD:

```
scala> val wordCounts = textFile.flatMap(line => line.split(" ")).

map(word => (word, 1)).

reduceByKey((a, b) => a + b)
```

We can then view the result of running our word count via the collect operation:

```
scala> wordCounts.collect()
```

How long does this operation take when you only use one core? Note that Spark reports execution times for alloperations, there should be a line at the end of collect's output that is labeled with "INFO DAGSchedule" and starts with "Job 0 finished" which reports a time. Take note of this time, as we will compare it to an execution with more than 1 core later.

If you run this collect operation repeatedly, i.e.:

```
scala> wordCounts.collect()

scala> wordCounts.collect()

scala> wordCounts.collect()
```

does the execution time change after the first execution? Can you explain this change?

You can also test the speedup with more cores. Kill your current Spark session by pressing Ctrl+D, and launch a new one that uses 4 cores (assuming your laptop has 4 or more cores):

```
./bin/spark-shell --master local[4] --driver-memory 2G
```

Now, rerun the previous commands:

```
scala> val textFile = sc.textFile("32big.txt")

scala> val wordCounts = textFile.flatMap(line => line.split(" ")).

map(word => (word, 1)).

reduceByKey((a, b) => a + b)

scala> wordCounts.collect()
```

How does execution time change, compared to the time you noted for a single core?

Selective Wordcount

Your next task is to alter our map/reduce operation on textFile so that only words of length 5 are counted,

and then display the counts of all (and only) words of length 5.

Hints:

- Scala syntax for if expressions is:

  if testExpr thenExpr else elseExpr

  An if expression can be used in any context that an expression can be used, and returns the value returned by whichever branch of the if expression is executed.

- The length of a string s in Scala can be found by using the methods. length

- The elements of a pair can be retrieved using the accessors _1 and _2. For example:

  scala> (1,2)._1

  res2: Int = 1

- Collections in Scala (including RDDs) have a method filter that takes a boolean test and returns a new RDD that contains only the elements for which the testing function passed. For example, the following application of filter to a list of ints returns a new list containing only the even elements:

  scala> List(1,2,3,4).filter(n => n % 2 == 0)

  res1: List[Int] = List(2, 4)

Estimating π

Exit the Spark Shell using Ctrl+D and then restart it with just one core:

./bin/spark-shell --master local[1] --driver-memory 2G

We can now walk through a Spark program to estimate π in parallel from random trials. We will alter the

number of cores that Spark makes use of and observe the impact on performance.

At your read-eval-print loop, first define the number of random trials:

scala> val NUM_SAMPLES = 1000000000

Now we can estimate π with the following code snippet:

```
scala> val count = sc.parallelize(1 to NUM_SAMPLES).map{i =>

val x = Math.random()

val y = Math.random()

if (x*x + y*y < 1) 1 else 0

}.reduce(_ + _)
```

You can then print out an estimate of π as follows:

```
println("Pi is roughly " + 4.0 * count / NUM_SAMPLES)
```

Now exit the Spark shell and restart with 2 cores, and then 4 cores. Do you observe a speedup?

6.8 EXERCISES

Implement a Spark program that does the following:

1. When running the provided example code, you will observe that there might be several entries that are equivalent. More specifically, you will see two entries for "British" with the only difference that one has an extra white space in the name. Propose a new version of the computation that will consider these two entries as the same one.
2. Count the total number of observations included in the dataset (each line corresponds to one observation)
3. Count the number of years over which observations have been made (Column "Year" should be used)
4. Display the oldest and the newest year of observation
5. Display the years with the minimum and the maximum number of observations (and the corresponding number of observations)
6. Count the distinct departure places (column "VoyageFrom") using two methods (i.e., using the function distinct() or reduceByKey()) and compare the execution time.
7. Display the 10 most popular departure places
8. Display the 10 roads (defined by a pair "VoyageFrom" and "VoyageTo") the most often taken.

- Here you can start by implementing a version where a pair "VoyageFrom"-"VoyageTo"

  A-B and a pair B-A correspond to different roads.

- Implement then a second version where A-B and B-A are considered as the same road.

8. Compute the hottest month (defined by column "Month") on average over the years considering all temperatures (column "ProbTair") reported in the dataset.

6.9 QUESTIONS

1. How is Apache Spark different from MapReduce?
2. What are the important components of the Spark ecosystem?
3. Explain how Spark runs applications with the help of its architecture.
4. What are the different cluster managers available in Apache Spark?
5. What is the significance of Resilient Distributed Datasets in Spark?
6. What is a lazy evaluation in Spark?
7. What makes Spark good at low latency workloads like graph processing and Machine Learning?
8. How can you trigger automatic clean-ups in Spark to handle accumulated metadata?
9. How can you connect Spark to Apache Mesos?
10. What is a Parquet file and what are its advantages?
11. What is shuffling in Spark? When does it occur?
12. What is the use of coalesce in Spark?
13. How can you calculate the executor memory?
14. What are the various functionalities supported by Spark Core?
15. How do you convert a Spark RDD into a DataFrame?
16. Explain the types of operations supported by RDDs.
17. What is a Lineage Graph?
18. What do you understand about DStreams in Spark?
19. Explain Caching in Spark Streaming.
20. What is the need for broadcast variables in Spark?
21. How to programmatically specify a schema for DataFrame?
22. Which transformation returns a new DStream by selecting only those records of the source DStream for which the function returns true?
23. Does Apache Spark provide checkpoints?
24. What do you mean by sliding window operation?
25. What are the different levels of persistence in Spark?
26. What is the difference between map and flatMap transformation in Spark Streaming?
27. How would you compute the total count of unique words in Spark?

28. Suppose you have a huge text file. How will you check if a particular keyword exists using Spark?
29. What is the role of accumulators in Spark?
30. What are the different MLlib tools available in Spark?
31. What are the different data types supported by Spark MLlib?
32. What is a Sparse Vector?
33. Describe how model creation works with MLlib and how the model is applied.
34. What are the functions of Spark SQL?
35. How can you connect Hive to Spark SQL?
36. What is the role of Catalyst Optimizer in Spark SQL?
37. How can you manipulate structured data using domain-specific language in Spark SQL?
38. What are the different types of operators provided by the Apache GraphX library?
39. What are the analytic algorithms provided in Apache Spark GraphX?
40. What is the PageRank algorithm in Apache Spark GraphX?

6.10 QUIZ

1. Spark was initially started by ____________ at UC Berkeley AMPLab in 2009.

   a) Mahek Zaharia

   b) Matei Zaharia

   c) Doug Cutting

   d) Stonebraker

2. Point out the correct statement.

   a) RSS abstraction provides distributed task dispatching, scheduling, and basic I/O functionalities

   b) For cluster manager, Spark supports standalone Hadoop YARN

   c) Hive SQL is a component on top of Spark Core

   d) None of the mentioned

3. ____________ is a component on top of Spark Core.

   a) Spark Streaming

   b) Spark SQL

   c) RDDs

   d) All of the mentioned

4. Spark SQL provides a domain-specific language to manipulate ___________ in Scala, Java, or Python.

   a) Spark Streaming

   b) Spark SQL

   c) RDDs

   d) All of the mentioned

5. Point out the wrong statement.

   a) For distributed storage, Spark can interface with a wide variety, including Hadoop Distributed File System (HDFS)

   b) Spark also supports a pseudo-distributed mode, usually used only for development or testing purposes

   c) Spark has over 465 contributors in 2014

   d) All of the mentioned

6. ______________ leverages Spark Core fast scheduling capability to perform streaming analytics.

   a) MLlib

   b) Spark Streaming

   c) GraphX

   d) RDDs

7. ____________ is a distributed machine learning framework on top of Spark.

   a) MLlib

   b) Spark Streaming

   c) GraphX

   d) RDDs

8. ________ is a distributed graph processing framework on top of Spark.

   a) MLlib

   b) Spark Streaming

   c) GraphX

   d) All of the mentioned

9. GraphX provides an API for expressing graph computation that can model the __________ abstraction.

   a) GaAdt

   b) Spark Core

   c) Pregel

   d) None of the mentioned

10. Spark architecture is ___________ times as fast as Hadoop disk-based Apache Mahout and even scales better than Vowpal Wabbit.

    a) 10
    b) 20
    c) 50
    d) 100

11. Users can easily run Spark on top of Amazon's __________

    a) Infosphere
    b) EC2
    c) EMR
    d) None of the mentioned

12. Point out the correct statement.

    a) Spark enables Apache Hive users to run their unmodified queries much faster
    b) Spark interoperates only with Hadoop
    c) Spark is a popular data warehouse solution running on top of Hadoop
    d) None of the mentioned

13. Spark runs on top of ___________ a cluster manager system which provides efficient resource isolation across distributed applications.

    a) Mesjs
    b) Mesos
    c) Mesus
    d) All of the mentioned

14. Which of the following can be used to launch Spark jobs inside MapReduce?

    a) SIM
    b) SIMR
    c) SIR
    d) RIS

15. Point out the wrong statement.

    a) Spark is intended to replace, the Hadoop stack
    b) Spark was designed to read and write data from and to HDFS, as well as other storage systems
    c) Hadoop users who have already deployed or are planning to deploy Hadoop Yarn can simply run Spark on YARN
    d) None of the mentioned

16. Which of the following language is not supported by Spark?

    a) Java

    b) Pascal

    c) Scala

    d) Python

17. Spark is packaged with higher level libraries, including support for _________ queries.

    a) SQL

    b) C

    c) C++

    d) None of the mentioned

18. Spark includes a collection over ________ operators for transforming data and familiar data frame APIs for manipulating semi-structured data.

    a) 50

    b) 60

    c) 70

    d) 80

19. Spark is engineered from the bottom-up for performance, running ___________ faster than Hadoop by exploiting in memory computing and other optimizations.

    a) 100x

    b) 150x

    c) 200x

    d) None of the mentioned

20. Spark powers a stack of high-level tools including Spark SQL, MLlib for _________

    a) regression models

    b) statistics

    c) machine learning

    d) reproductive research

6.11 VIDEO LECTURES

1. What Is Apache Spark.
   https://www.youtube.com/watch?v=znBa13Earms

2. Spark Tutorial | Spark Tutorial for Beginners.
   https://www.youtube.com/watch?v=zC9cnh8rJd0

3. Apache Spark Full Course - Learn Apache Spark in 8 Hours.
   https://www.youtube.com/watch?v=F8pyaR4uQ2g

4. Apache Spark Tutorial | Spark Tutorial for Beginners.
   https://www.youtube.com/watch?v=9mELEARcxJo

5. Spark Tutorial For Beginners | Big Data Spark Tutorial.
   https://www.youtube.com/watch?v=QaoJNXW6SQo

6. Apache Spark Full Course | Apache Spark Tutorial For Beginners.
   https://www.youtube.com/watch?v=vqEF9F7pH40

7. Apache Spark Tutorial | Spark Tutorial for Beginners | Spark Big Data | Intellipaat.
   https://www.youtube.com/watch?v=GFC2gOL1p9k

8. Big Data on Spark | Tutorial for Beginners [Part 1] | Introduction to Spark | Great Learning.
   https://www.youtube.com/watch?v=HSC4XYW0KrA

9. Apache Spark Internal architecture jobs stages and tasks.
   https://www.youtube.com/watch?v=P1knn8i1Ijs

6.12 MOOCS

1. Apache Spark Course. Intellipaat. https://intellipaat.com/apache-spark-scala-training/?utm_source=google&utm_medium=search&utm_term=apache%20spark%20online%20course&utm_campaign=s_apache_spark_in_state&gclid=CjwKCAjw77WVBhBuEiwAJ-YoJInJN3VDTkf1Dz2Ikw_Xstic-JpWLlweFRQDp59nQrbvM0FVMpgOGBoCEl4QAvD_BwE

2. Apache Spark and Scala Training. The Knowledge Academy. https://www.theknowledgeacademy.com/in/offers/apache-spark-and-scala-certification-training-courses/?utm_term=apache%20spark%20courses&utm_campaign=%5BAttention+Management%5D&utm_source=adwords&utm_medium=ppc&hsa_acc=8156085647&hsa_cam=16215108706&hsa_grp=131206939897&hsa_ad=582094255411&hsa_src=g&hsa_tgt=kwd385642089417&hsa_kw=apache%20spark%20courses&hsa_mt=p&hsa_net=adwords&hsa_ver=3&gclid=CjwKCAjw77WVBhBuEiwAJ-YoJOyMetejzXy0mGePV7noZU2s3lV7eNzTsP4Dqs-WXUbOBJGmos5KChoCev0QAvD_BwE

3. Post Graduate Program In Data Engineering. Purdue University. https://www.simplilearn.com/pgp-data-engineering-certification-training-course?utm_source=google&utm_medium=cpc&utm_term=apache%20spark%20course&utm_content=11233548673-106960986141-473291238433&utm_device=c&utm_campaign=Search-DataCluster-PG-BigData-CDE-Purdue-IN-Main-AllDevice-adgroup-Apache-Spark-Course-phrase&gclid=CjwKCAjw77WVBhBuEiwAJ-YoJA_zbAqjr244nB2x1Z6t11N21AN5G-cKVGWhX6uZ3zMC0LTcXx9i7BoCJdoQAvD_BwE

4. Apache Spark. EdX. https://www.edx.org/learn/apache-spark

5. Data Science on Apache Spark. Databricks. https://databricks.com/blog/2015/06/01/databricks-launches-mooc-data-science-on-spark.html

REFERENCES

1. https://techvidvan.com/tutorials/spark-rdd-features/

2. https://data-flair.training/blogs/apache-spark-rdd-features/

3. https://sparkbyexamples.com/apache-spark-rdd/spark-rdd-transformations/

4. https://data-flair.training/blogs/spark-rdd-operations-transformations-actions/

5. https://sparkbyexamples.com/apache-spark-rdd/different-ways-to-create-spark-rdd/

6. https://data-flair.training/blogs/spark-in-memory-computing/

7. https://www.projectpro.io/recipes/explain-spark-lazy-evaluation-detail

8. https://data-flair.training/blogs/apache-spark-rdd-persistence-caching/

HTTPS://WWW.SIMPLILEARN.COM/TOP-APACHE-SPARK-INTERVIEW-QUESTIONS-AND-ANSWERS-ARTICLE#APACHE_SPARK_INTERVIEW_QUESTIONS

❄❄❄❄❄❄❄

7

DATA VISUALIZATION

Unit Structure :

7.0 Introduction
7.1 Data Visualization
7.2 Data Visualization Snd Big Data
7.3 Types of Data Visualizations
7.4 Common Data Visualization Use Cases
7.5 Basic Data Visualization Principles
7.6 Connecting to Data
7.7 Tableau
7.8 Custom SQL Queries Tableau
7.9 Visualizing Treemaps Using Tableau
7.10 Creating A Story with Tableau Public
7.11 Saving and Publishing Your Tableau Public Workbook
7.12 A Dashboard with Tableau
7.13 Questions
7.14 Quiz
7.15 Video Lectures
7.16 MOOCS
7.17 References

7.0 INTRODUCTION

There is an increased recognition that effectively visualizing data is important to anyone who works with and analyzes data. To that end, there has been an explosion in data analysis and data visualization tools over the past few years. Microsoft Excel continues to the be the workhorse for their data visualization needs. There are certainly different strategies to creating some of these graphs, the approach presented here allow you to not only create those graphs, but also give you the techniques you can use elsewhere to create your own graphs.

7.1 DATA VISUALIZATION

Data visualization is the representation of data through use of common graphics, such as charts, plots, infographics, and even animations. These visual displays of information communicate complex data relationships and data-driven insights in a way that is easy to understand.

Data visualization can be utilized for a variety of purposes, and it’s important to note that is not only reserved for use by data teams. Management also leverages it to convey organizational structure and hierarchy while data analysts and data scientists use it to discover and explain patterns and trends. Harvard Business Review categorizes data visualization into four key purposes: idea generation, idea illustration, visual discovery, and everyday dataviz. We’ll delve deeper into these below:

Data visualization is commonly used to spur idea generation across teams. They are frequently leveraged during brainstorming or Design Thinking sessions at the start of a project by supporting the collection of different perspectives and highlighting the common concerns of the collective. While these visualizations are usually unpolished and unrefined, they help set the foundation within the project to ensure that the team is aligned on the problem that they’re looking to address for key stakeholders.

Data visualization for idea illustration assists in conveying an idea, such as a tactic or process. It is commonly used in learning settings, such as tutorials, certification courses, centers of excellence, but it can also be used to represent organization structures or processes, facilitating communication between the right individuals for specific tasks. Project managers frequently use Gantt charts and waterfall charts to illustrate workflows.

Benefits of data visualization include the following:

- The ability to absorb information quickly, improve insights and make faster decisions;
- An increased understanding of the next steps that must be taken to improve the organization;
- An improved ability to maintain the audience's interest with information they can understand;
- An easy distribution of information that increases the opportunity to share insights with everyone involved;
- Eliminate the need for data scientists since data is more accessible and understandable; and
- An increased ability to act on findings quickly and, therefore, achieve success with greater speed and less mistakes.

7.2 DATA VISUALIZATION AND BIG DATA

The increased popularity of big data and data analysis projects have made visualization more important than ever. Companies are increasingly using machine learning to gather massive amounts of data that can be difficult and slow to sort through, comprehend and explain. Visualization offers a means to speed this up and present information to business owners and stakeholders in ways they can understand.

Big data visualization often goes beyond the typical techniques used in normal visualization, such as pie charts, histograms and corporate graphs. It instead uses more complex representations, such as heat maps and fever charts. Big data visualization requires powerful computer systems to collect raw data, process it and turn it into graphical representations that humans can use to quickly draw insights.

While big data visualization can be beneficial, it can pose several disadvantages to organizations. They are as follows:

- To get the most out of big data visualization tools, a visualization specialist must be hired. This specialist must be able to identify the best data sets and visualization styles to guarantee organizations are optimizing the use of their data.
- Big data visualization projects often require involvement from IT, as well as management, since the visualization of big data requires powerful computer hardware, efficient storage systems and even a move to the cloud.
- The insights provided by big data visualization will only be as accurate as the information being visualized. Therefore, it is essential to have people and processes in place to govern and control the quality of corporate data, metadata and data sources.

7.3 TYPES OF DATA VISUALIZATIONS

Individuals could utilize data visualization to present data in a more effective manner and companies turn to dashboards to report their performance metrics in real-time. Dashboards are effective data visualization tools for tracking and visualizing data from multiple data sources, providing visibility into the effects of specific behaviors by a team or an adjacent one on performance. Dashboards include common visualization techniques, such as:

Tables: This consists of rows and columns used to compare variables.

Pie charts and stacked bar charts: These graphs are divided into sections that represent parts of a whole.

Line graphs and area charts: These visuals show change in one or more quantities by plotting a series of data points over time. Line graphs utilize lines to demonstrate these changes while area charts connect data points with line segments, stacking variables on top of one another and using color to distinguish between variables.

Histograms: This graph plots a distribution of numbers using a bar chart representing the quantity of data that falls within a particular range.

Scatter plots: These visuals are beneficial in reveling the relationship between two variables, and they are commonly used within regression data analysis.

Heat maps: These graphical displays are helpful in visualizing behavioral data by location.

Tree maps: Display hierarchical data as a set of nested shapes, typically rectangles. Treemaps are great for comparing the proportions between categories via their area size.

Population pyramids: This technique uses a stacked bar graph to display the complex social narrative of a population.

7.4 COMMON DATA VISUALIZATION USE CASES

Sales and marketing. Research from the media agency Magna predicts that half of all global advertising dollars will be spent online by 2020. Data visualization makes it easy to see traffic trends over time as a result of marketing efforts.

Politics. A common use of data visualization in politics is a geographic map that displays the party each state or district voted for.

Healthcare. Healthcare professionals frequently use choropleth maps to visualize important health data. A choropleth map displays divided geographical areas or regions that are assigned a certain color in relation to a numeric variable.

Scientists. Scientific visualization, sometimes referred to in shorthand as SciVis, allows scientists and researchers to gain greater insight from their experimental data than ever before.

Finance. Finance professionals must track the performance of their investment decisions when choosing to buy or sell an asset.

Logistics. Shipping companies can use visualization tools to determine the best global shipping routes.

Data scientists and researchers. Visualizations built by data scientists are typically for the scientist's own use, or for presenting the information to a select audience.

DATA VISUALIZATION TOOLS AND VENDORS

Data visualization tools can be used in a variety of ways. The most common use today is as a business intelligence (BI) reporting tool. Users can set up visualization tools to generate automatic dashboards that track company performance across key performance indicators (KPIs) and visually interpret the results.

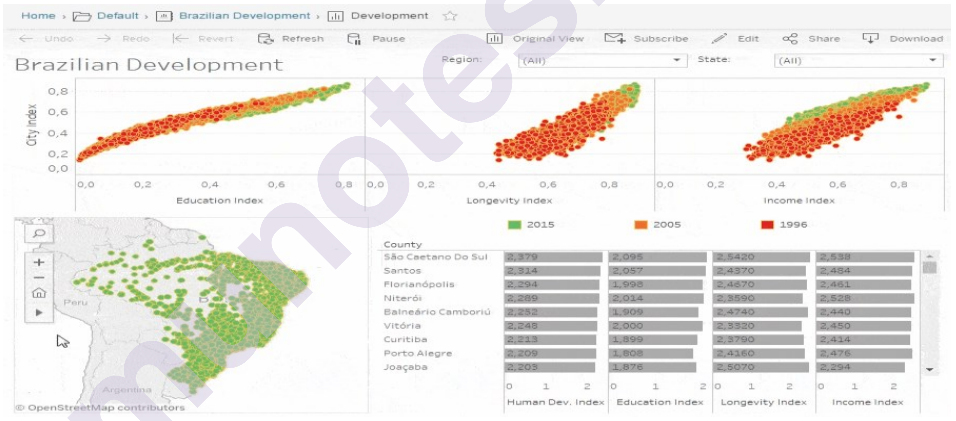
While Microsoft Excel continues to be a popular tool for data visualization, others have been created that provide more sophisticated abilities:

- IBM Cognos Analytics
- Qlik Sense and QlikView

- Microsoft Power BI
- Oracle Visual Analyzer
- SAP Lumira
- SAS Visual Analytics
- Tibco Spotfire
- Zoho Analytics
- D3.js
- Jupyter
- MicroStrategy
- Google Charts

DATA VISUALIZATION BEST PRACTICES

- Set the context:
- Know your audience(s)
- Choose an effective visual
- Keep it simple



County	Human Dev. Index	Education Index	Longevity Index	Income Index
São Caetano Do Sul	2,379	2,095	2,5420	2,538
Santos	2,314	2,057	2,4370	2,484
Florianópolis	2,294	1,998	2,4670	2,461
Niterói	2,289	2,014	2,3590	2,528
Balneário Camboriú	2,252	1,909	2,4740	2,440
Vitória	2,248	2,000	2,3320	2,450
Curitiba	2,213	1,899	2,3790	2,414
Porto Alegre	2,209	1,808	2,4160	2,476
Joaçaba	2,203	1,876	2,5070	2,294

Fig 1: Good data visualization

7.5 BASIC DATA VISUALIZATION PRINCIPLES

Three basic principles seem especially useful to guide your creation of better, more effective visualizations.

1. **Show the Data**

   People read will read the graphs in your report, article, or blog post to better understand your argument.

2. **Reduce the Clutter**

   Cart clutter, the use of unnecessary or distracting visual elements, tends to reduce effectiveness of the graph. Clutter comes in many forms: dark or heavy gridlines; unnecessary tick marks, labels, or text;

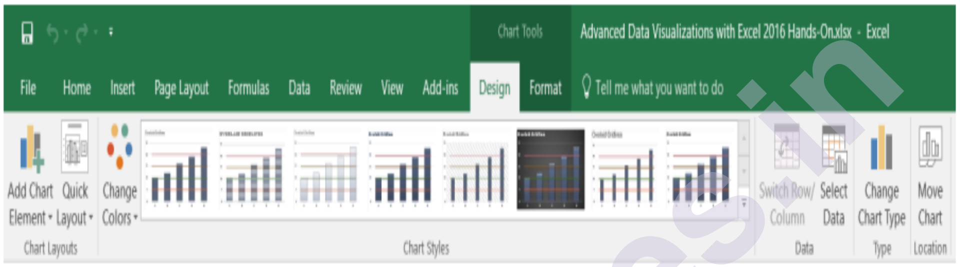
unnecessary icons or pictures; ornamental shading and gradients; and unnecessary dimensions.

3. **Integrate the Text and the Graph**

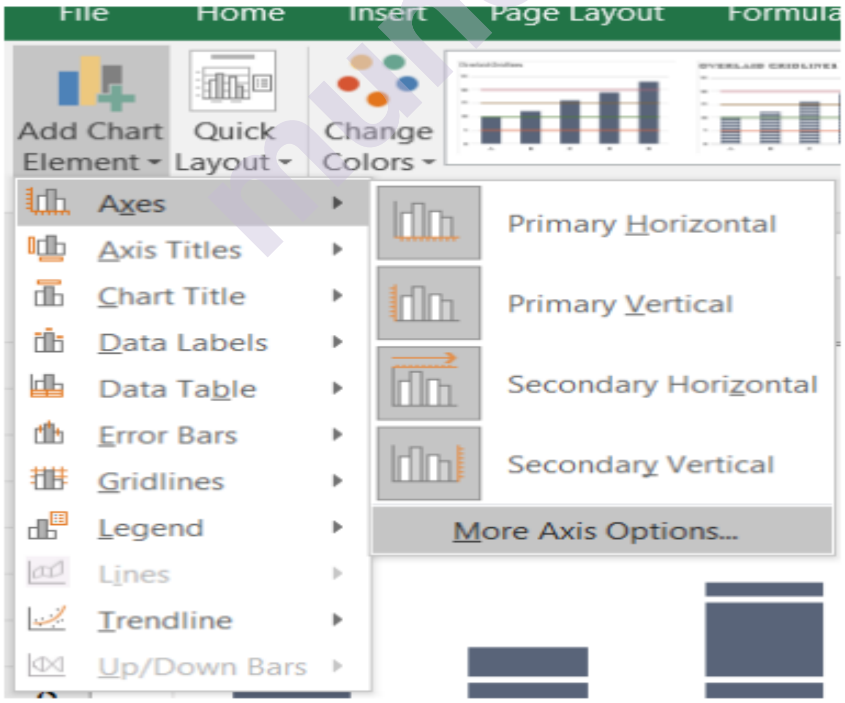
Legends define or explain a series on a graph are often placed far away from the content—off to the right or below the graph.

Design Tab

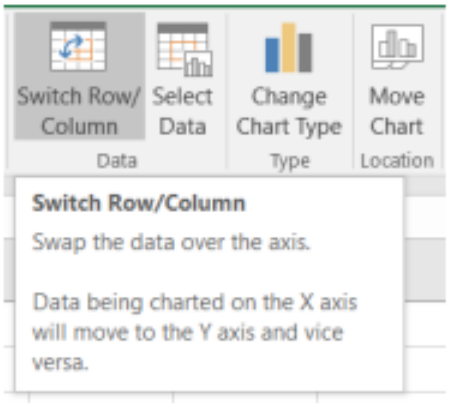
Chart Tools tab will appear at the top of your ribbon consisting of two tabs: Design and Format. The Design tab contains options that allow you to apply different default 'Chart Layouts' and 'Chart Styles'. The options available under the 'Add Chart Element' button replaces the Layout tab on previous versions of Excel and allows you to modify the appearance of axes, titles, gridlines, and more.



Each of the options in the 'Add Chart Element' menu allows you to choose from a set of pre-populated options, or to open a menu with more options.



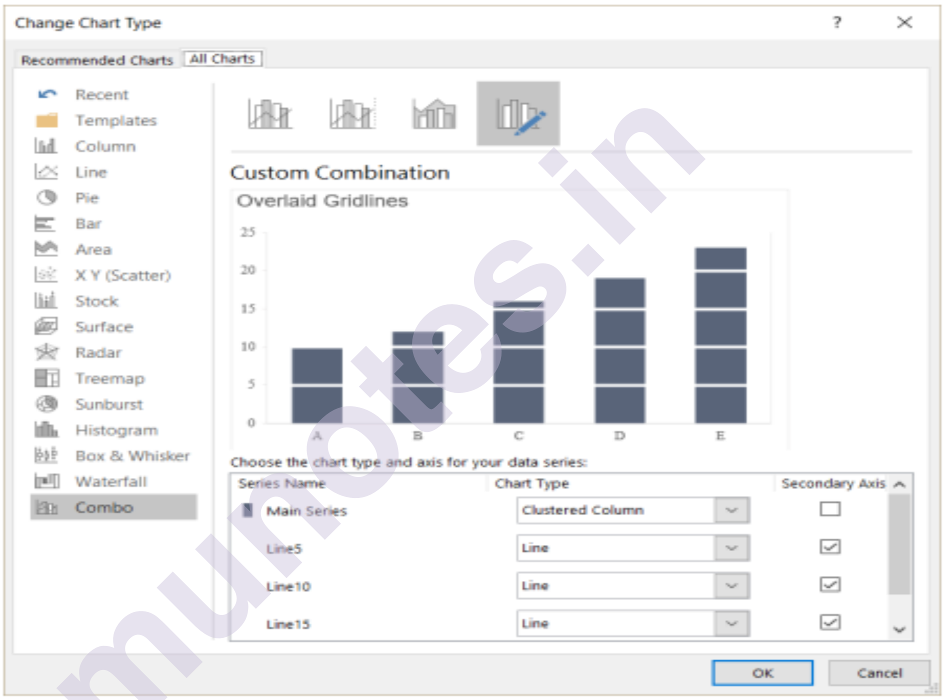
The 'Change Chart Type' button will allow you to change the type of chart for all the data on the chart, or a selected series.



Switch Row/Column

Swap the data over the axis.

Data being charted on the X axis will move to the Y axis and vice versa.

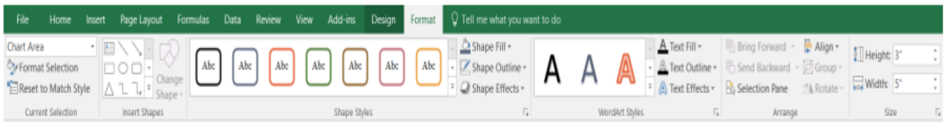


Choose the chart type and axis for your data series:

Series Name	Chart Type	Secondary Axis
Main Series	Clustered Column	☐
Line5	Line	☑
Line10	Line	☑
Line15	Line	☑

Format Tab

The Format tab contains the standard outline and fill color options.



In the very top-left section of the Format tab is the ‘Chart Elements’ drop-down menu. The list in this drop-down menu consists of everything in your chart including titles, axes, error bars, and every series. If you have a lot of objects on your chart, this drop-down menu will help you to easily find and select what you need.

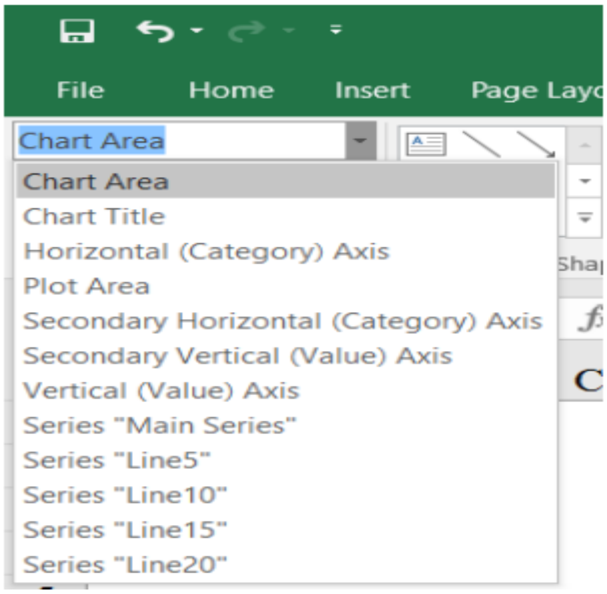
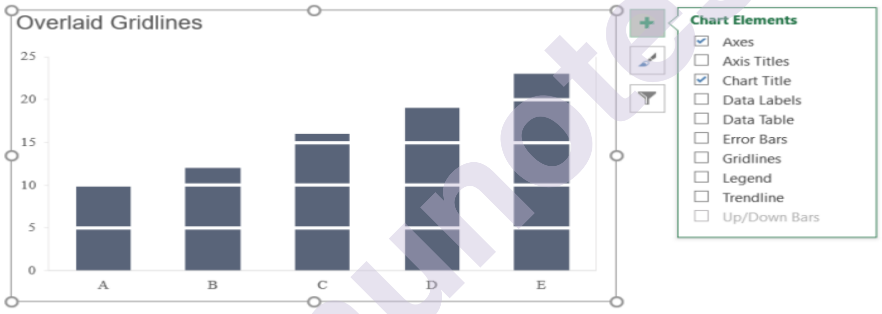
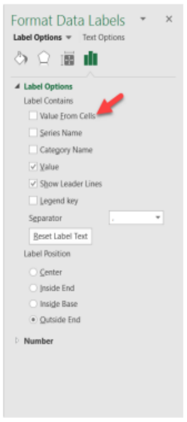


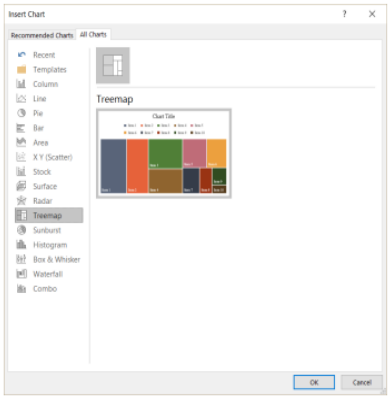
Chart Elements Menu

‘Chart Elements Menu’ that appears just outside the top-right part of the chart when you select it. Appearing as a ‘plus’ symbol, the menu is identical to the ‘Add Chart Elements’ button in the Design tab.



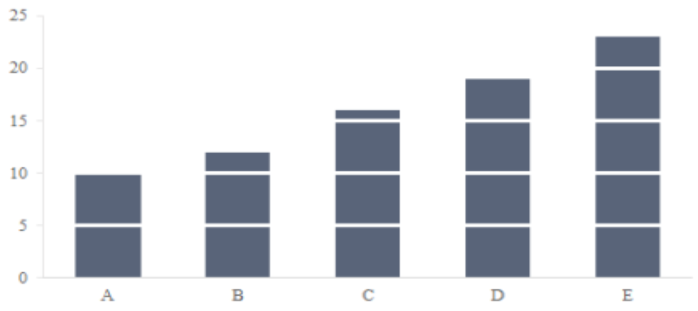
Select a specific data range to use as labels in your chart. This comes in quite handy when, for example, you want to add custom labels to a scatterplot. Instead of having to do the labeling manually, you can select the data labels series in the spreadsheet. Among the new chart types is a Treemap, Histogram, Box & Whisker, and Waterfall chart.



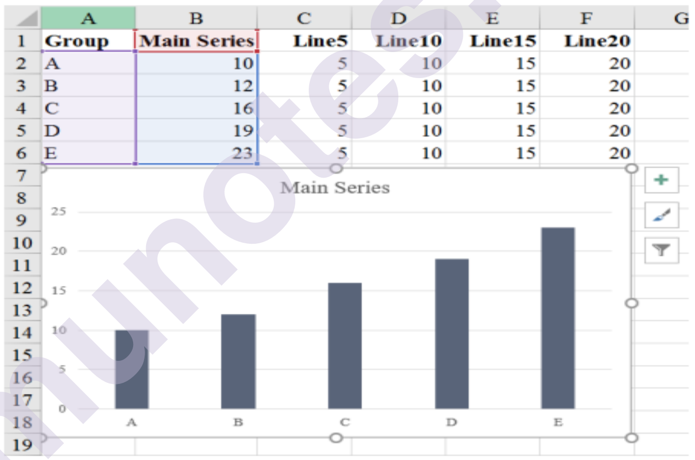


Overlaid Gridlines

The Overlaid Gridline chart is a column chart with gridlines on top of the columns. This type of chart allows viewers to absorb the column data as segments rather than single columns. Use the OverlaidGridline tab in the Advanced Data Visualizations with Excel 2016 Hands-On.xlsx spreadsheet to create the chart.

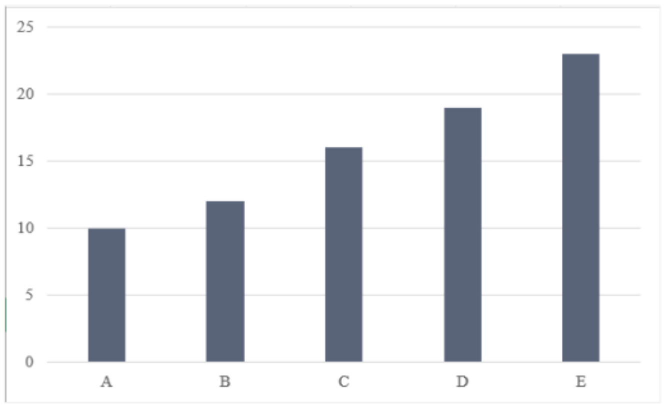


1. Begin by creating a column chart from columns A ("Group") and B ("Main Series").

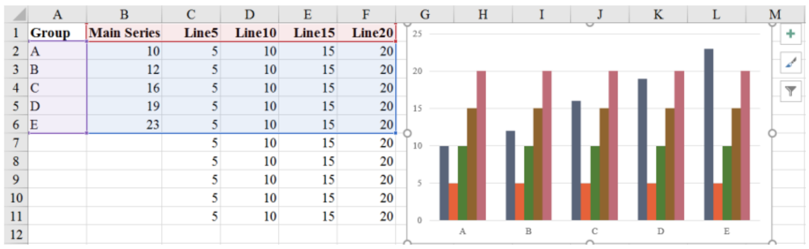


	A	B	C	D	E	F
1	Group	Main Series	Line5	Line10	Line15	Line20
2	A	10	5	10	15	20
3	B	12	5	10	15	20
4	C	16	5	10	15	20
5	D	19	5	10	15	20
6	E	23	5	10	15	20

2. Remove the title.

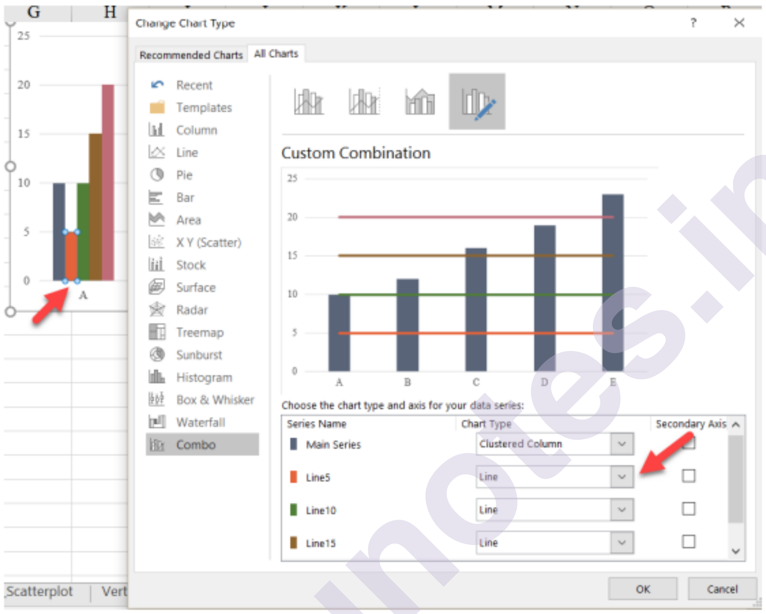


3. We're now going to add the four "Line" series to the chart.



Group	Main Series	Line5	Line10	Line15	Line20
A	10	5	10	15	20
B	12	5	10	15	20
C	16	5	10	15	20
D	19	5	10	15	20
E	23	5	10	15	20
		5	10	15	20
		5	10	15	20
		5	10	15	20
		5	10	15	20
		5	10	15	20

4. You will now have a clustered column chart, five series for each group.



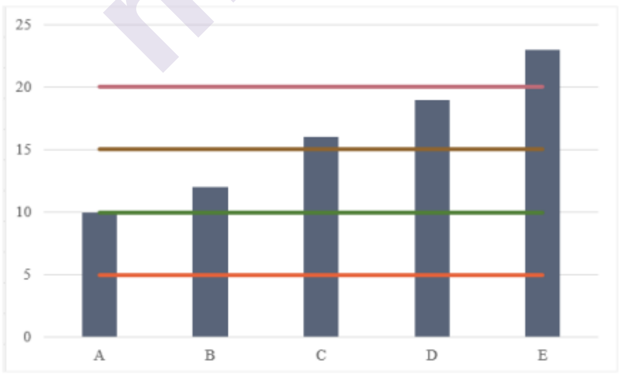
Change Chart Type

Custom Combination

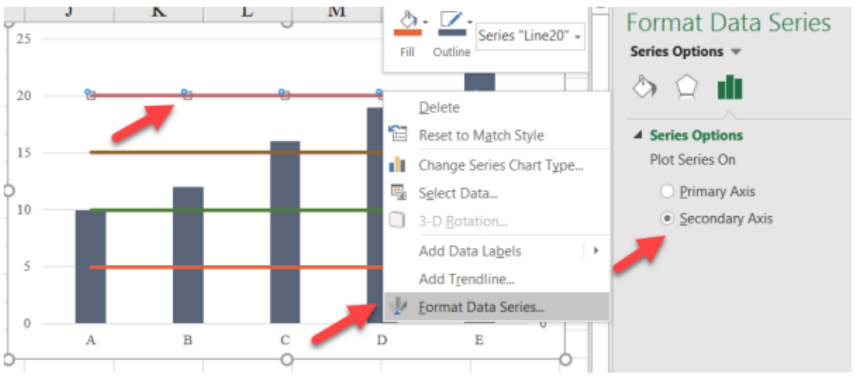
Choose the chart type and axis for your data series:

Series Name	Chart Type	Secondary Axis
Main Series	Clustered Column	☐
Line5	Line	☐
Line10	Line	☐
Line15	Line	☐

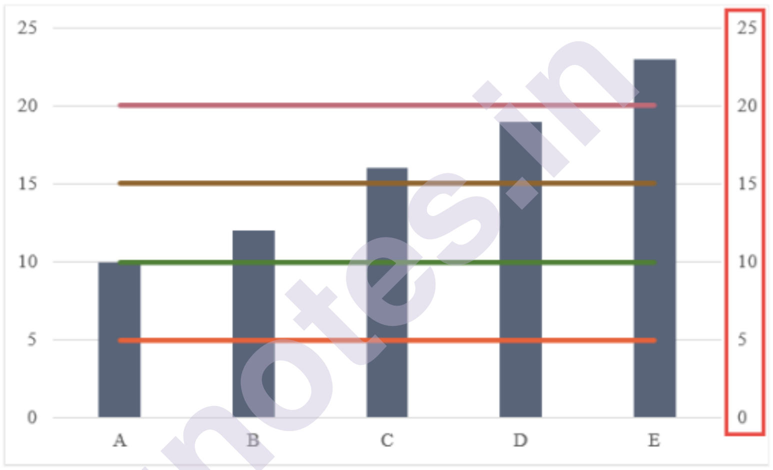
Each series except the "Main Series" now become lines.



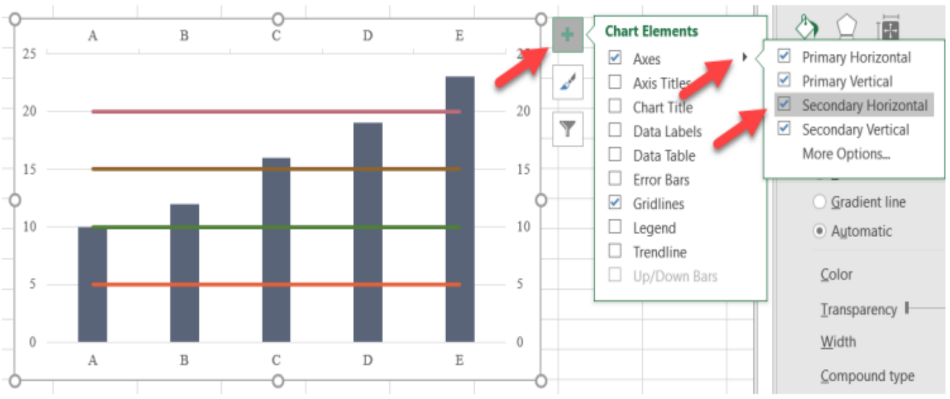
5. If we were to simply change the lines to white, they would end in the middle of the bars of the A and E groups. We now move each of those four lines to the "secondary axis" so we can get them to stretch through the bars. To do so, first select a line, right-click, and select "Format Data Series". Go to the "Series Options" tab and select the "Secondary Axis" option.



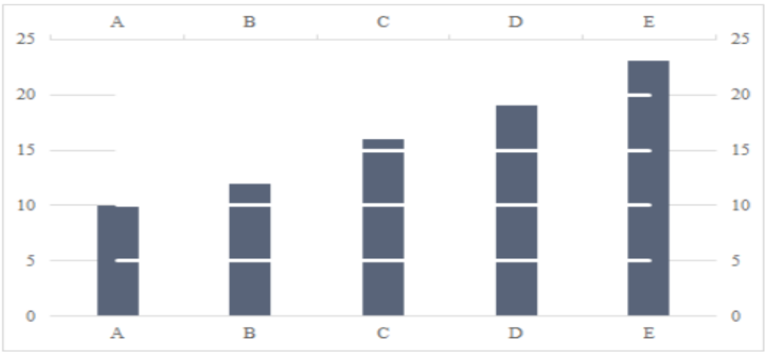
6. You'll notice that a new y-axis has appeared on the right side of the graph. When you're done moving all four series to the secondary axis, this new y-axis should go from 0 to 25.



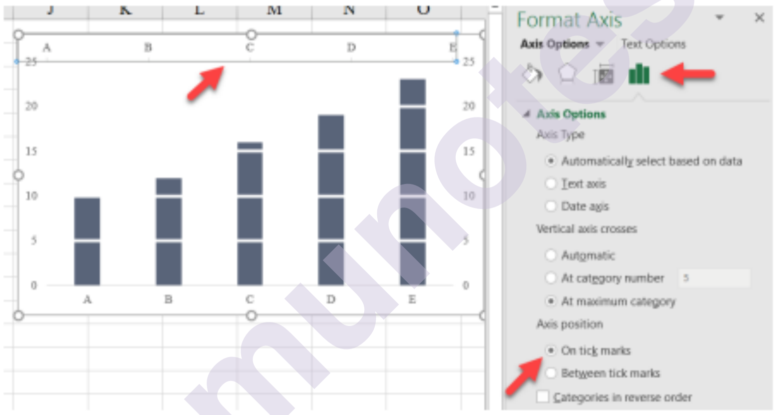
7. There is also now a secondary x-axis, but we need to turn it on. To do so, select the "Axes" option in the "Chart Elements" menu by pressing the "plus" button that will appear when you select the chart. By hovering over the "Axes" menu, three of the boxes will have checkmarks next to them. Turn on the "Secondary Horizontal" axis by selecting the checkbox.



8. Change the colors of the lines to white using the "Format" tab option.



9. We fix that by changing how the data points line up with the tick marks. In a default line graph in Excel, the data markers line up between the tick marks; notice how the line begins in the middle of the A bar, between the y-axis and the tick mark between the A and B groups. By placing the data markers on the tick marks, we can extend the lines through the bars. To do so, we'll format the secondary x-axis (by rightclicking and navigating to the "Axis position" options under "Axis Options" in the "Format axis" menu.


Format Axis

Axis Options Text Options

Axis Options

Axis Type

- [✓ Automatically select based on data]
- Text axis
- Date axis

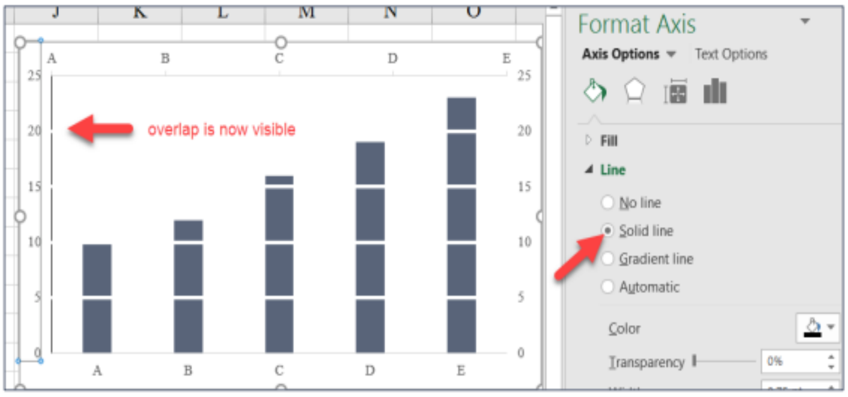
Vertical axis crosses

- Automatic
- At category number 5
- [✓ At maximum category]

Axis position

- [✓ On tick marks]
- Between tick marks
- Categories in reverse order

10. Add your vertical primary axis line, select the axis and add the line under the "Format Axis" menu.



Format Axis

Axis Options Text Options

Fill

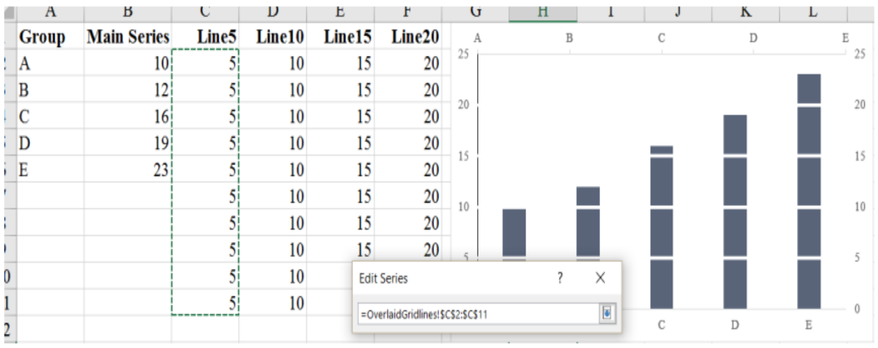
Line

- No line
- [✓ Solid line]
- Gradient line
- Automatic

Color

Transparency 0%

11. We want to extend the data series for each "Line" series through row 11. One way to do this is to right-click on the graph, select the "Select Data" option, and edit each of the 4 "Line" series to extend the data series.

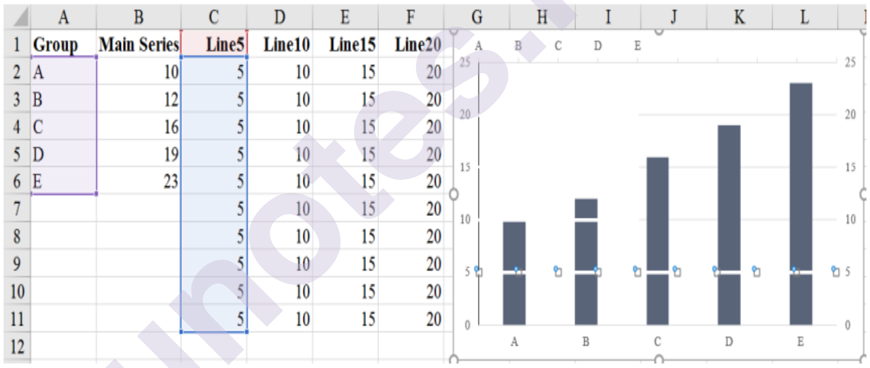


Group	Main Series	Line5	Line10	Line15	Line20
A	10	5	10	15	20
B	12	5	10	15	20
C	16	5	10	15	20
D	19	5	10	15	20
E	23	5	10	15	20
		5	10	15	20
		5	10	15	20
		5	10	15	20
		5	10		
		5	10		

Edit Series

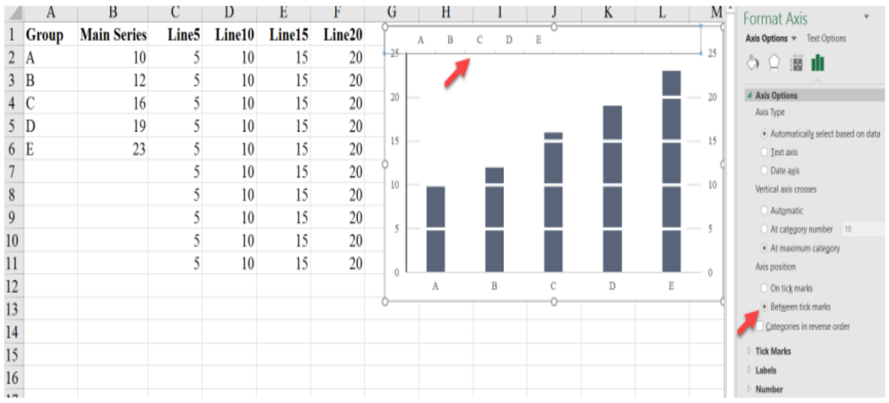
=OverlaidGridlines!$C$2:$C$11

Alternatively, you can select the line on the chart and you'll notice that your data are selected in the spreadsheet. You can then drag the selection box to extend the data series.



Group	Main Series	Line5	Line10	Line15	Line20
A	10	5	10	15	20
B	12	5	10	15	20
C	16	5	10	15	20
D	19	5	10	15	20
E	23	5	10	15	20
		5	10	15	20
		5	10	15	20
		5	10	15	20
		5	10	15	20
		5	10	15	20

12. We need to now change where the data markers line up with the tick marks. Once again, format the secondary x-axis and change the "Position Axis" back to "Between tick marks".



Group	Main Series	Line5	Line10	Line15	Line20
A	10	5	10	15	20
B	12	5	10	15	20
C	16	5	10	15	20
D	19	5	10	15	20
E	23	5	10	15	20
		5	10	15	20
		5	10	15	20
		5	10	15	20
		5	10	15	20
		5	10	15	20

Format Axis

Axis Options ▾ Text Options

Axis Options

Axis Type

● Automatically select based on data

○ Text axis

○ Date axis

Vertical axis crosses

○ Automatic

○ At category number 10

● At maximum category

Axis position

○ On tick marks

● Between tick marks

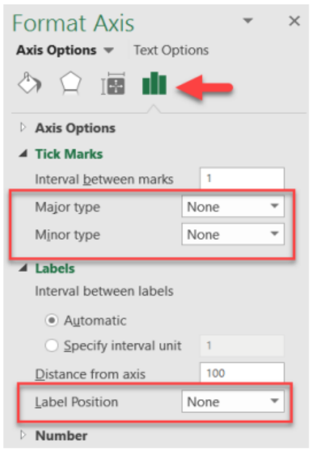
☐ Categories in reverse order

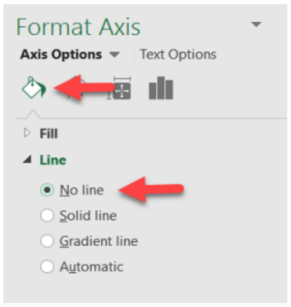
Tick Marks

Labels

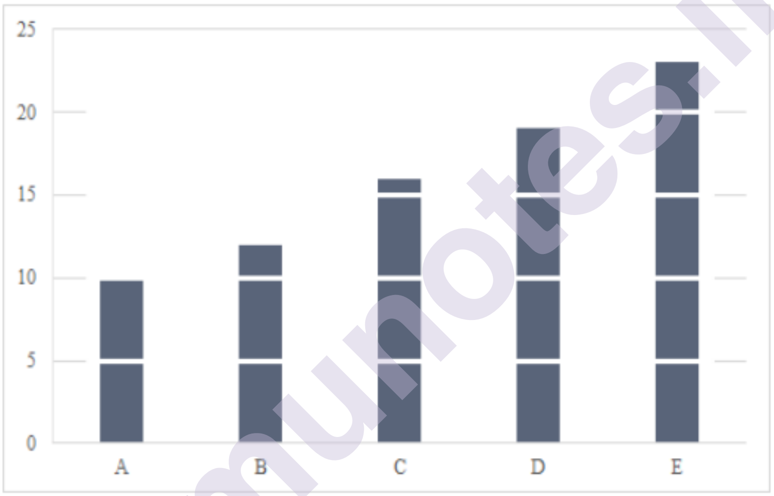
Number

13. We also want to turn off the secondary horizontal axis and set the "Line Color" to "No line".

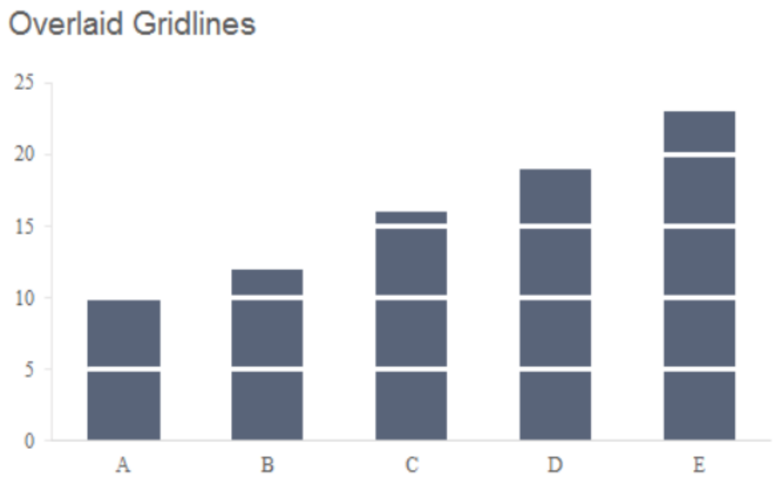

Format Axis
Axis Options Text Options
Axis Options
Tick Marks
Interval between marks 1
Major type None
Minor type None
Labels
Interval between labels
Automatic
Specify interval unit 1
Distance from axis 100
Label Position None
Number


Format Axis
Axis Options Text Options
Fill
Line
No line
Solid line
Gradient line
Automatic

14. Repeat the process in Step 13 for the secondary vertical axis, remove the gridlines and style the rest as you see fit.

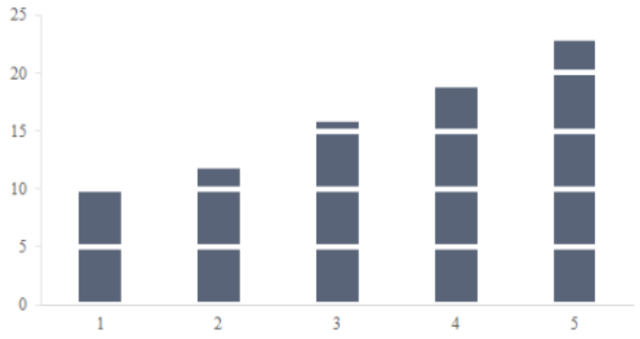


Final Version with Styling

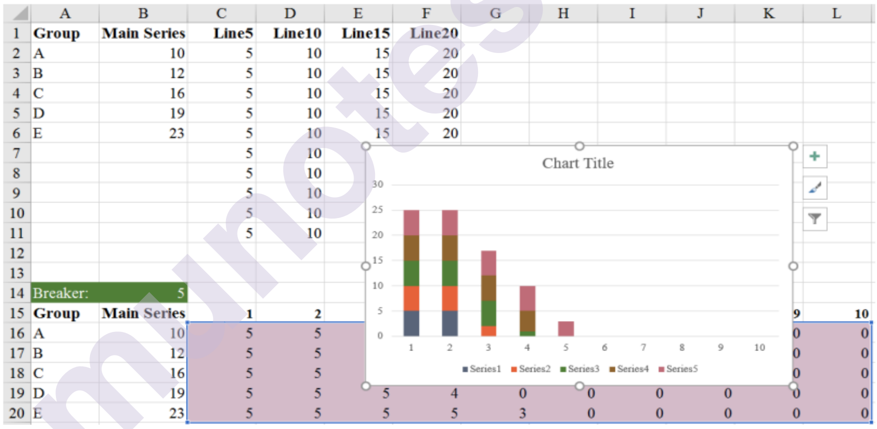


Overlaid Gridlines with a Formula

In this version of the Overlaid Gridlines chart, I create a stacked column chart. Each section of the chart is given a white outline so that it appears like there are gridlines. Use the OverlaidGridlines_Formula tab in the Advanced Data Visualizations with Excel 2016 Hands-On.xlsx spreadsheet to create the chart.

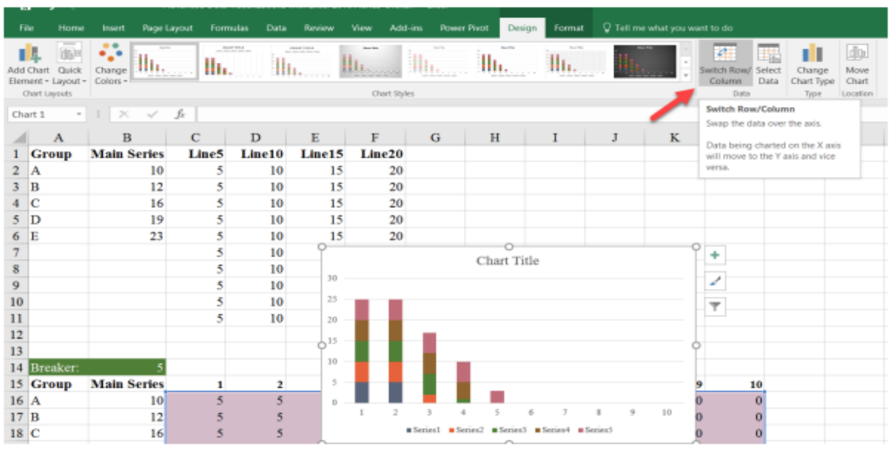


Create a stacked column chart from cells C16:M20. These are the cells that contain the formula.



	A	B	C	D	E	F	G	H	I	J	K	L
1	Group	Main Series	Line5	Line10	Line15	Line20						
2	A	10	5	10	15	20						
3	B	12	5	10	15	20						
4	C	16	5	10	15	20						
5	D	19	5	10	15	20						
6	E	23	5	10	15	20						
7			5	10								
8			5	10								
9			5	10								
10			5	10								
11			5	10								
12												
13												
14	Breaker:	5										
15	Group	Main Series	1	2							9	10
16	A	10	5	5							0	0
17	B	12	5	5							0	0
18	C	16	5	5							0	0
19	D	19	5	5	5	4	0	0	0	0	0	0
20	E	23	5	5	5	5	3	0	0	0	0	0

To plot the rows, select the chart and the "Switch Row/Column" button in the "Design" tab of the ribbon.



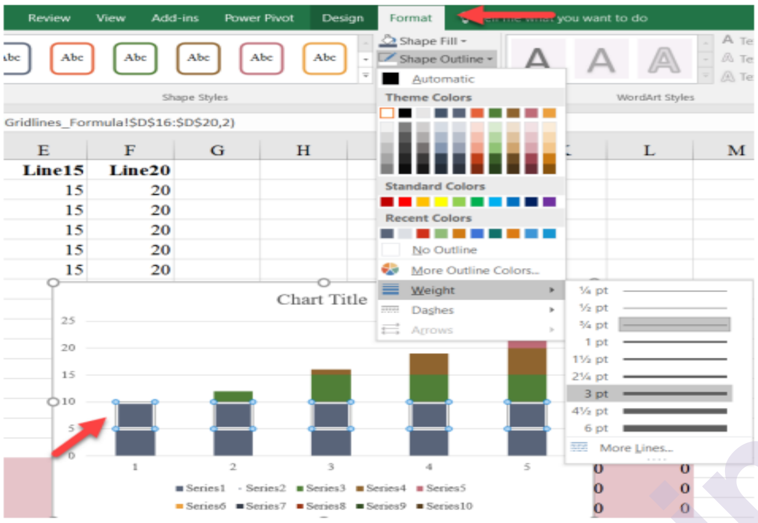
Switch Row/Column

Swap the data over the axis.

Data being charted on the X axis will move to the Y axis and vice versa.

	A	B	C	D	E	F	G	H	I	J	K
1	Group	Main Series	Line5	Line10	Line15	Line20					
2	A	10	5	10	15	20					
3	B	12	5	10	15	20					
4	C	16	5	10	15	20					
5	D	19	5	10	15	20					
6	E	23	5	10	15	20					
7			5	10							
8			5	10							
9			5	10							
10			5	10							
11			5	10							
12											
13											
14	Breaker:	5									
15	Group	Main Series	1	2							9
16	A	10	5	5							0
17	B	12	5	5							0
18	C	16	5	5							0

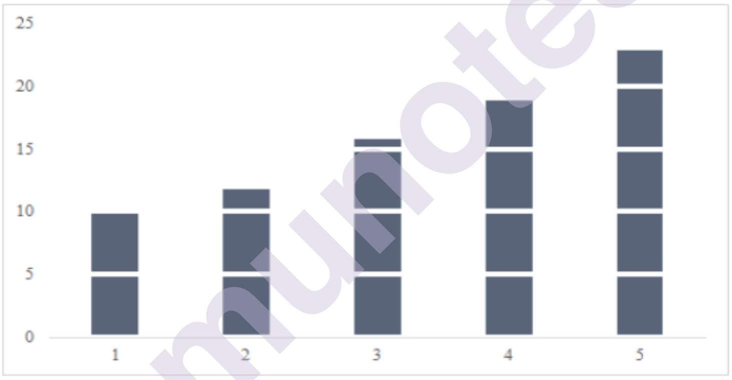
Change the fill of each shape under the “Shape Fill” dropdown in the “Format” menu to the same color. Similarly, change the color of the “Shape Outline” to white and increase the thickness to your desired weight. Of course, delete the existing (default) gridlines, legend, etc.



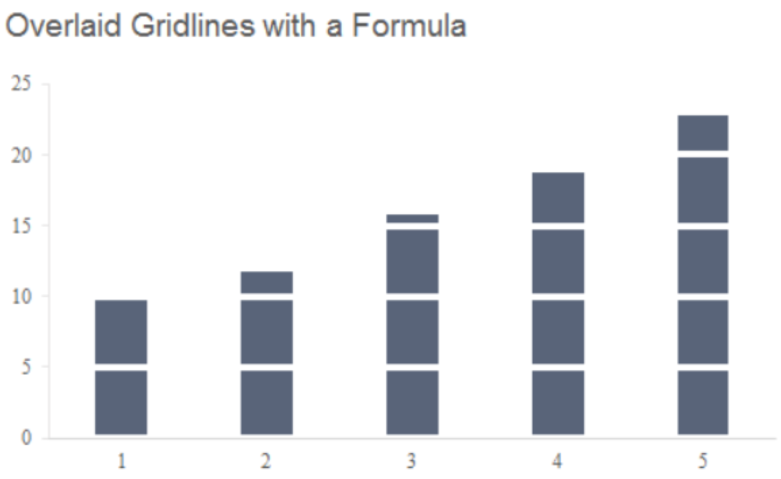
Gridlines_Formula!$D$16:$D$20,2)

E	F
Line15	Line20
15	20
15	20
15	20
15	20
15	20

Repeat for all 5 series



Final Version with Styling



7.6 CONNECTING TO DATA

Visualizations depend on the data in them, and however aesthetically pleasing your visualization might be, it may be misleading or even wrong, unless the data has been formatted, aggregated, and properly represented.

This session discusses the major elements of finding, cleansing, understanding, formatting, and aggregating data that you will need to understand in order to produce accurate visualizations that tell compelling stories, including the following elements:

- Where you can get publicly available data and how to use it
- What tables and databases are
- The data formats that Tableau Public connects to
- Databases, tables, dimensions, facts, and field formats and conventions
- Preparing data to load it into Tableau
- Connecting to the data from Tableau Public
- Using the data interpreter
- Pivoting fields

Public data

The data sets that are publicly available or the ones that you have compiled on your own, are ideal for Tableau Public. Public data is readily available online. Tableau Public maintains a catalog of publicly available data. Much of this data is produced by various governments, economic groups, and sports fans, along with a link to, and a rating for each source. You can find it at http://public.tableau.com/s/resources. The Google Public Data Explorer has a large collection of public data, including economic forecasts and global public health data. This tool is unique because it allows users to make simple visualizations from all the original data sources without having to investigate the source data, though most of it is available by linking available resources.

Tables and databases

Data is stored in tables. A table is an array of items, and it can be as simple as a single word, letter, or number, or as complicated as millions (or more) of rows of transactions with timestamps, qualitative attributes (such as size or color), and numeric facts, such as the quantity of the purchased goods.

Both a single text file of data and a worksheet in an excel workbook are tables. When grouped together in a method that has been designed to enable a user to retrieve data from them, they constitute a database. Typically, when we think of databases, we think of the Database Management Systems (DBMS) and languages that we use to make sense of the data in tables, such as Oracle, Teradata, or Microsoft's SQL Server. Currently, the Hadoop and NoSQL platforms are very popular because they are comparatively low-cost and can store very large sets of data, but Tableau Public does not enable a connection to these platforms.

Tableau Public is designed in such a way that it allows users in a single data connection to join tables of data, which may or may not have been previously related to each other, as long as they are in the same format.

The most common format of publicly available data is in a text file or a Character-Separated Values (CSV) file. CSV files are useful because they are simple. Many public data sources do allow data to be downloaded as Excel documents. The World Bank has a comprehensive collection, and we will demonstrate the connective capabilities of Tableau Public using one of its data products. Tables can be joined in Tableau Public by manually identifying the common field among the tables. Tableau Public connects to four different data sources, namely Access, Excel, text file (CSV or TXT), and OData; the first two data sources are bundled with Microsoft Office (in most cases), and the second two are freely available to everyone, regardless of the operating system that they are using.

Connecting to the data in Tableau Public

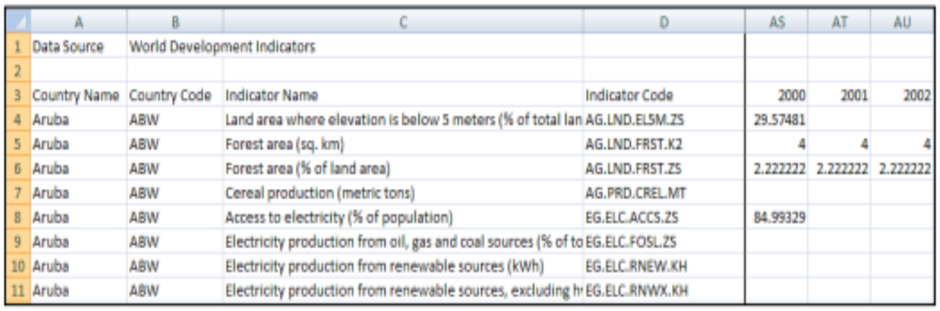
Tableau Public has a graphical user interface (GUI) that was designed to enable users to load data sources without having to write code. Since the only place to save Tableau Public documents is in Tableau's Cloud, data sources are automatically extracted and packaged with the workbook.

Connecting to data from a local file is illustrated as follows with detailed screenshots:

1. Click on the Connect to Data Link option from the Data menu.
2. Select the data source type.
3. Select the file or website to which you want to connect.
4. For a Microsoft Access, Microsoft Excel, or a text file, determine whether the connection is to one table or multiple tables or it requires a custom SQL connection:
    - If the connection is to one table, select the table.
    - If the connection is to multiple tables, select the option for the connection to multiple tables and identify the join conditions. We will discuss this in detail in the next section.
    - Alternatively, you can type or paste a custom SQL.

5. When all the selections have been made, click on Ok.

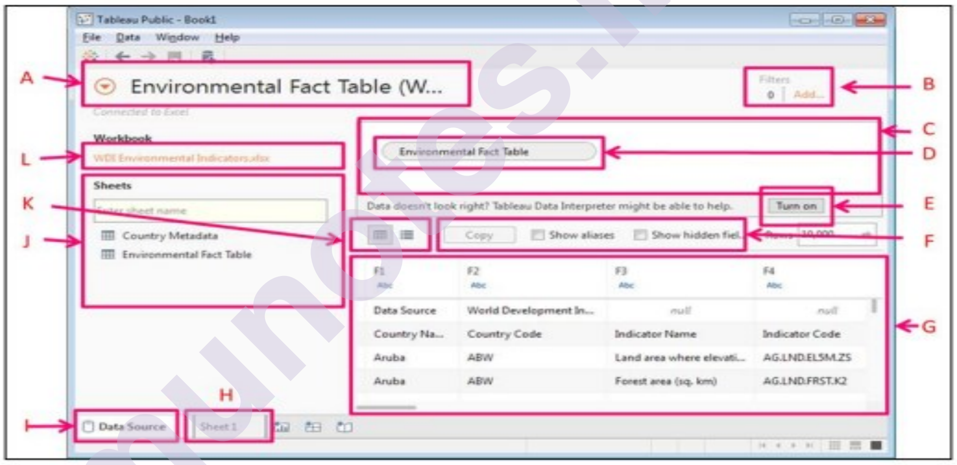
We will connect to the World Bank's environment indicators. You can download this data, which is formatted either for Microsoft Excel or as a text file, at www.worldbank.org.



	A	B	C	D	AS	AT	AU
1	Data Source	World Development Indicators					
2							
3	Country Name	Country Code	Indicator Name	Indicator Code	2000	2001	2002
4	Aruba	ABW	Land area where elevation is below 5 meters (% of total lan	AG.LND.EL5M.ZS	29.57481		
5	Aruba	ABW	Forest area (sq. km)	AG.LND.FRST.K2	4	4	4
6	Aruba	ABW	Forest area (% of land area)	AG.LND.FRST.ZS	2.222222	2.222222	2.222222
7	Aruba	ABW	Cereal production (metric tons)	AG.PRD.CREL.MT			
8	Aruba	ABW	Access to electricity (% of population)	EG.ELC.ACCS.ZS	84.99329		
9	Aruba	ABW	Electricity production from oil, gas and coal sources (% of to	EG.ELC.FOSL.ZS			
10	Aruba	ABW	Electricity production from renewable sources (kWh)	EG.ELC.RNEW.KH			
11	Aruba	ABW	Electricity production from renewable sources, excluding h	EG.ELC.RNWX.KH			

The data source user interface

Before loading data, let's know the different parts of the data connection user interface.



F1 Abc	F2 Abc	F3 Abc	F4 Abc
Data Source	World Development In...	null	null
Country Na...	Country Code	Indicator Name	Indicator Code
Aruba	ABW	Land area where elevati...	AG.LND.EL5M.ZS
Aruba	ABW	Forest area (sq. km)	AG.LND.FRST.K2

References	Description
A	This is the data source name, which will be modified in subsequent exercises
B	These are the data source filters, which can be used to limit the data that you load
C	This is the workspace, where you can add and join tables
D	These are individual tables
E	This is the Data Interpreter, which is available for Microsoft Excel files; we will learn how to turn it on and use it in subsequent exercises
F	Edit data source display by showing/hiding fields
G	This is the data
H	This is a link to sheets; you can click on this to go back to your worksheets
I	This is the Data Source button, which can be clicked on from any worksheet to get back to the data source
J	These are the tables within the data source, which can be dragged to the workspace to join to other workspace
K	This is the pivot or view grid of the data, which will be used in subsequent exercises
L	This is the data source, which can be changed by clicking on the orange link and then browsing to a new file

To load the file into Tableau Public, you can download the Tableau Public workbook by visiting https://public.tableau.com/profile/tableau.data.stories#!/.

1. Open a new instance of Tableau Public.
2. From the Connect pane, click on the data file type to which you'd like to connect. In this case, we are using an excel file.
3. Browse to the file to which you would like to connect.
4. Drag a table from the list of tables, which is a list of different worksheets in this case, along with the workbook onto the workspace.
5. Note that the values in the data source are now populating the space below the workspace, but at least with this data set, there is no complete set of field headers. We will edit the data source by using the data interpreter in the next exercise.

Connecting to web-based data sources

The steps required to connect to OData are different from the steps required to connect to the previously mentioned sources because they involve web servers and network security. These steps are a subset of the steps in Desktop Professional that are used to connect to a server:

1. Enter the URL of the website.
2. Select the authentication method.
3. Establish the connection.
4. Name the data source.

In order to refresh a web-based data source, perform the following steps:

1. Right-click on the data source name in the data pane.
2. Click on Edit Connection.
3. In the previous dialog box, which will be populated with the connection parameters, click on the Connect button in step 3 of the preceding list.

7.7 TABLEAU

Tableau is a visual analysis solution that allows people to explore and analyze data with simple drag and drop operations. It has a user-friendly interface that creates reports that look great right at the beginning. Tableau is a rapid BI Software.

Great Visualizations:

Allows to connect to the data, visualize and creative interactive, sharable dashboards in a few clicks.

Ease of use:

It's easy enough that any excel user can learn it, but powerful enough to satisfy even the most complex analytical problems

Fast:

We can create interactive dashboards, quick filters and calculations.

WHY TABLEAU?

As per Gartner survey "Tableau was chosen more often for functionality than any other vendor in the survey, with the highest product functionality scores". "Tableau's products often fill an unmet need in organizations that already have a BI standard "and more frequently deployed than other interactive visualization vendors as a complementary capability to an existing BI platform.

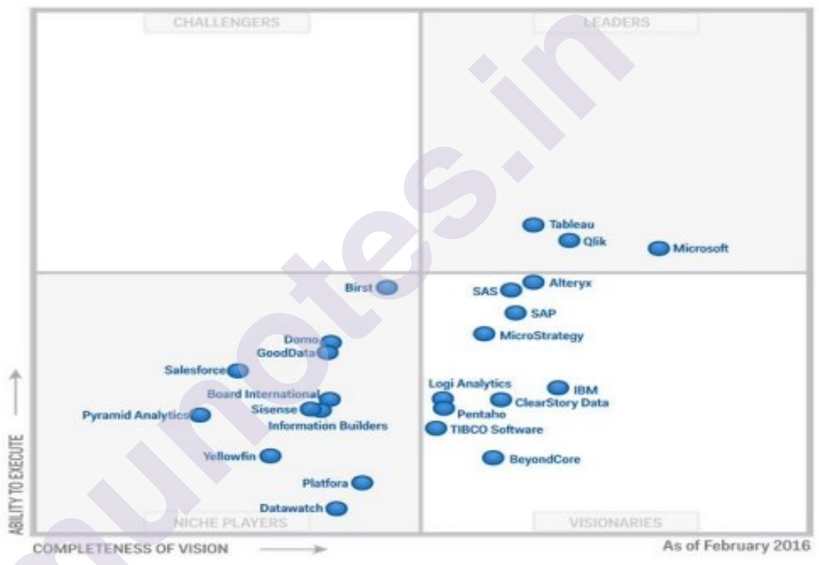
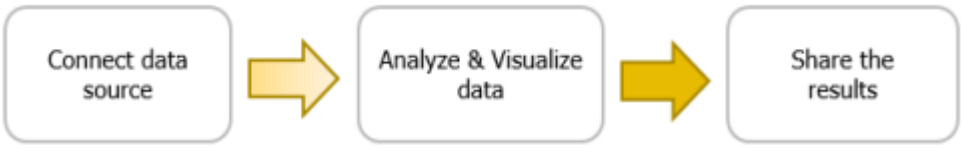


Fig 2. Gartner Chart 2016

Gartner positions tableau as "Leader". Companies frequently cite that breadth and ease of use along with high business benefits as the primary reasons that they choose Tableau.

Three main stages in Tableau



Connect data source: Connect Tableau to any data source like MS-Excel, MySQL, and Oracle. Tableau connects data in two ways Live connect and Extract.

Analyze & Visualize: Analyze the data by filtering, sorting and Visualize the data using the relevant chart provided. Tableau automatically analyzes the data as follows: Dimensions: Tableau treats any field containing qualitative, categorical information as dimension. All dimensions are indicated by "Blue" color.

Measures: Tableau treats any field containing numeric information as a measure. All measures are indicated by "Green" color Tableau suggests the recommended charts based on the dimensions and measures.

Share: Tableau Dashboards can be shared as word documents, pdf files, images.

Maps in Tableau

Tableau automatically assigns geographic roles with common geographically names, such as Country, State/Province, City etc. Fields with geographic role will automatically generates longitude and latitude coordinates on a map view. Fields with assigned geographic roles will have a globe icon next to them. In Tableau, there are two types of maps to choose from when creating a visualization with geographic data:

- ❖ Symbol Maps and
- ❖ Filled Maps.

Symbol Maps: These are simple maps that use a type of mark to represent a data point, such as a filled circle.

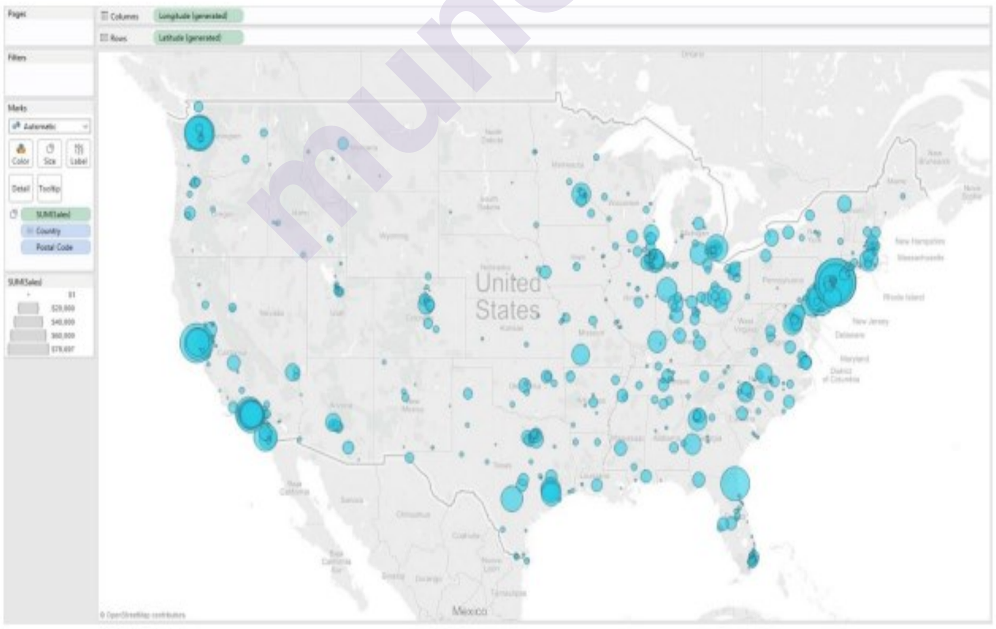


Fig 3. Symbol Map

Filled Maps: Instead of displaying data points, filled maps uses shading on a country or state basis to indicate relationships.

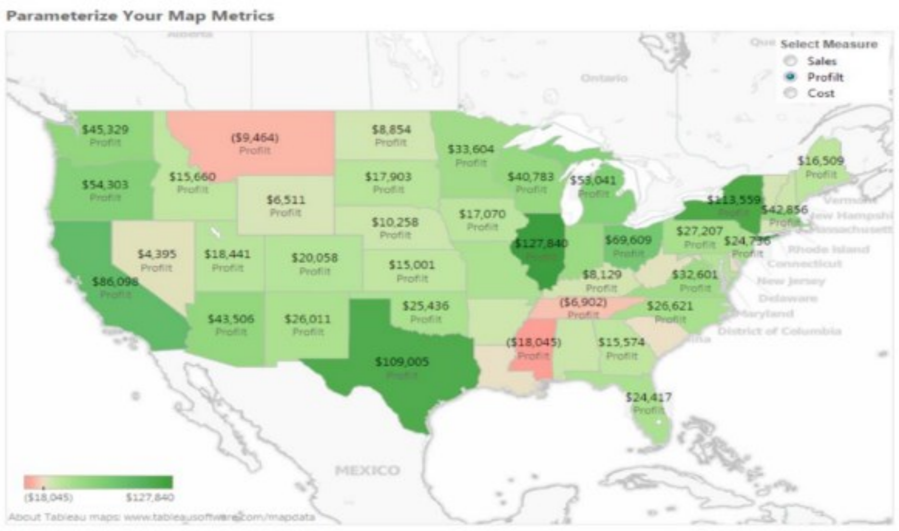


Fig 4. Filled Map

7.8 CUSTOM SQL QUERIES TABLEAU

What is Custom SQL Query?

Tableau queries each data source using SQL that's specific to the data type. Tableau allows the SQL used to query a data source to be customized in order to manipulate the joins, filters, and field lengths and types produce a more accurate output.

Sample SQL query:

```
SELECT ['us states data 1$'].[State] AS [State],

['us states data 1$'].[Population] AS [Population],

['us states data 1$'].[Region] AS [Region]

FROM ['us states data 1$']

UNION

SELECT ['us states data 2$'].[State] AS [State],

['us states data 2$'].[Population] AS [Population],

['us states data 2$'].[Region] AS [Region]

FROM ['us states data 2$']

UNION

SELECT ['us states data 3$'].[State] AS [State],

['us states data 3$'].[Population] AS [Population],

['us states data 3$'].[Region] AS [Region]
```

FROM ['us states data 3$']

UNION

SELECT ['us states data 4$'].[State] AS [State],

['us states data 4$'].[Population] AS [Population],

['us states data 4$'].[Region] AS [Region]

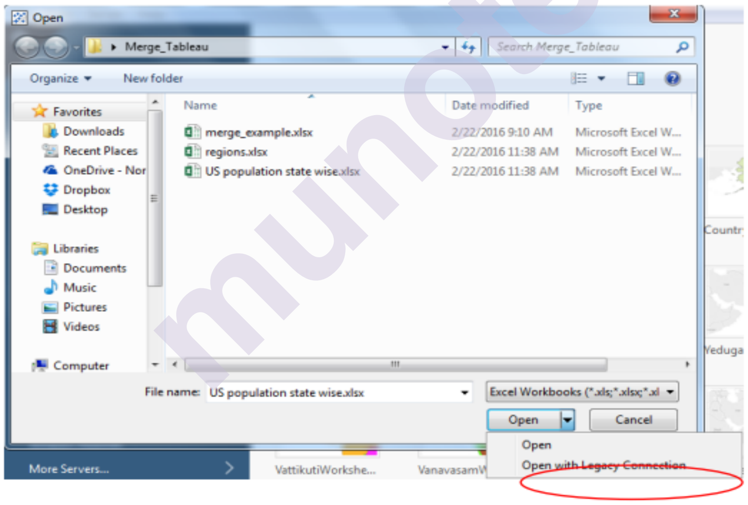
FROM ['us states data 4$']

Steps to visualize using SQL in Tableau:

1. The following excel workbook consists of 4 sheets consisting of US population state wise.

US population state wise.xlsx

2. Open Tableau and connect the Excel data as Open with Legacy Connection.

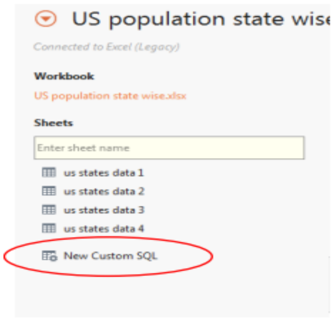


Name	Date modified	Type
merge_example.xlsx	2/22/2016 9:10 AM	Microsoft Excel W...
regions.xlsx	2/22/2016 11:38 AM	Microsoft Excel W...
US population state wise.xlsx	2/22/2016 11:38 AM	Microsoft Excel W...

File name: US population state wise.xlsx

Excel Workbooks (*.xls;*.xlsx;*.xl

3. Now your Tableau window looks like this with an additional sheet named as New Custom SQL.



US population state wis

Connected to Excel (Legacy)

Workbook

US population state wise.xlsx

Sheets

Enter sheet name

us states data 1

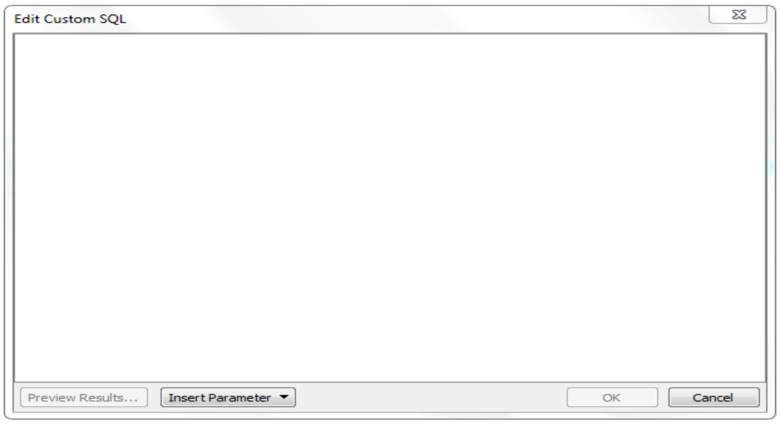
us states data 2

us states data 3

us states data 4

New Custom SQL

4. Click on New Custom SQL then you will be prompted to Edit Custom SQL as shown.



5. Now we need to write SQL query to perform union operation on the excel sheets and generate a combined sheet.

Ex:

```
SELECT [Sheet1$].[ID] AS [ID],

[Sheet1$].[Type] AS [Type],

[Sheet1$].[Value] AS [Value]

FROM [Sheet1$]

UNION ALL

SELECT [Sheet2$].[ID] AS [ID],

[Sheet2$].[Type] AS [Type],

[Sheet2$].[Value] AS [Value]

FROM

[Sheet2$]
```

The above SQL query performs union operation on two sheets namely Sheet1 and Sheet2 with ID, Type, and Value as their columns.

6. After writing the query click on Preview Results on the Edit Custom SQL window pane to check whether the query worked fine or not.

7. The resulting data should contain all the states, population and region name to which it belong to.

8. Click on the Tableau sheet to visualize the data.

9. Now you will see State, Region as Dimensions and Population as Measures.

10. Now visualize the data derived.

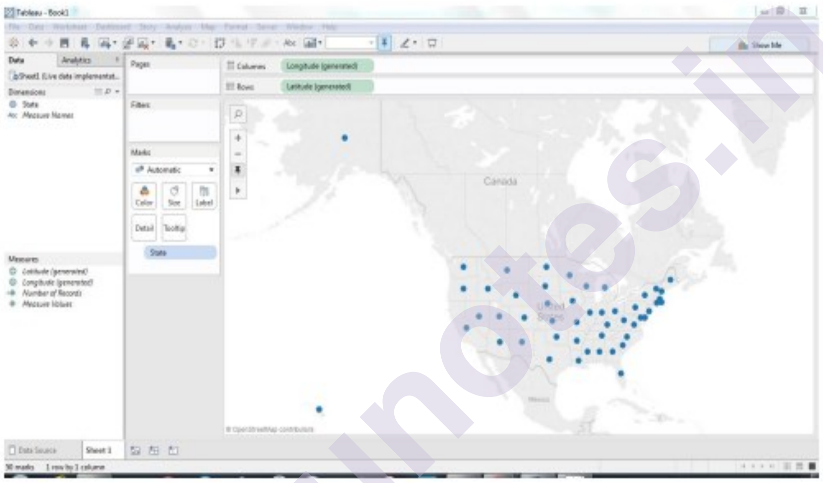
Live data implementation Tableau

Live Data or Real-time data (RTD) denotes information that is delivered immediately after collection. There is no delay in the timeliness of the information provided. Real-time data is often used for navigation or tracking. A Web Map Service (WMS) is a standard protocol for serving georeferenced map images which a map server generates using data from a GIS database. A Web Map Service (WMS) produces an image (e.g. GIF, JPG) of geospatial data. Steps to implement live map using Tableau:

1. Open Tableau and connect the Excel data.

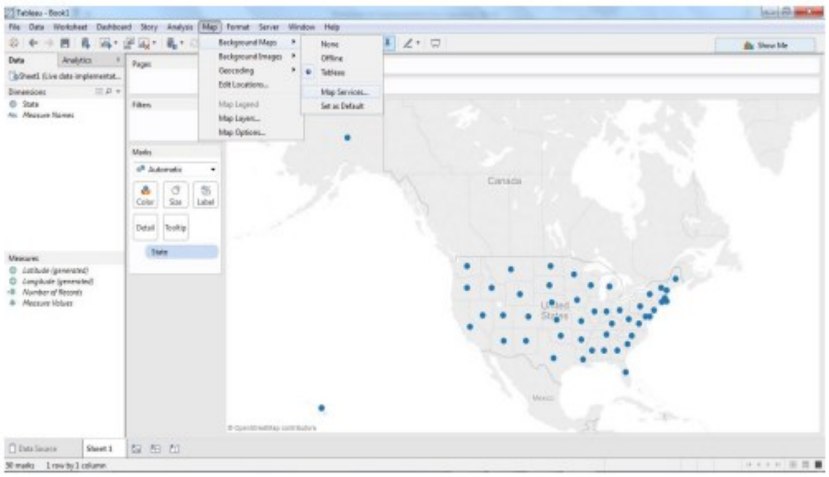
2. Drag and drop state Dimension in Detail menu provided in Marks window pane.

3. Now you can see Symbol map representing the states of United States. It looks as shown below

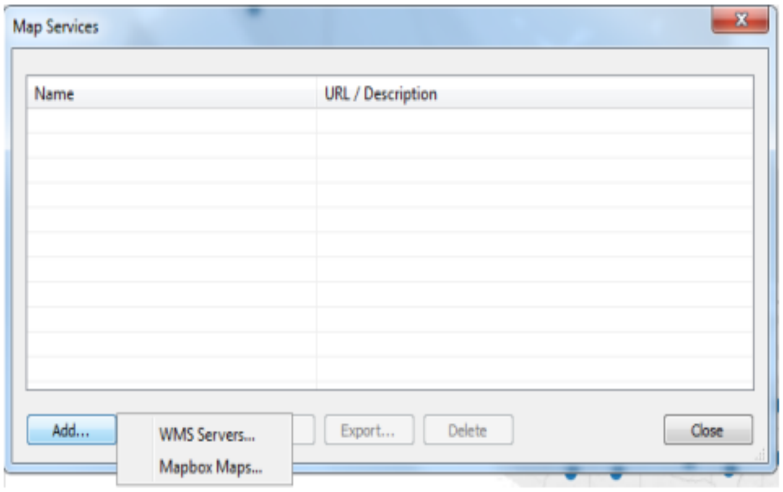


4. In this we will be using OGCWMS server to get the live geospatial data.

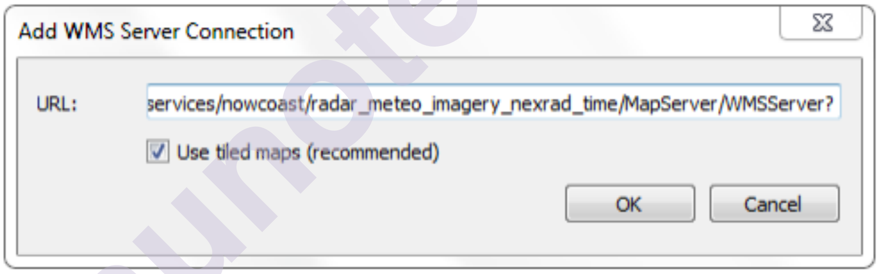
5. Click on Map item in the menu bar, then select Background Maps and select WMS Server as shown



6. After clicking on Map Services you will be prompted a window as shown



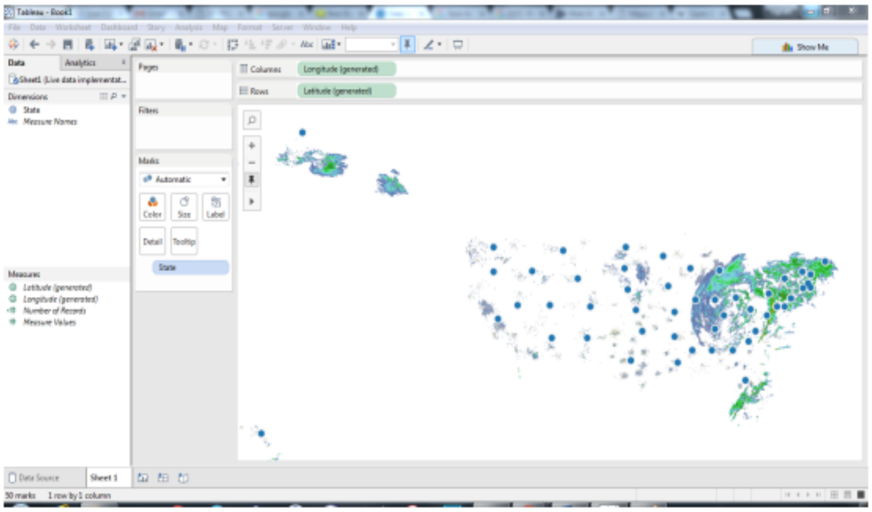
7. Click on Add and select WMS Servers. After clicking on WMS Servers a window pane as shown below will be prompted to enter the URL of the server. http://nowcoast.noaa.gov/arcgis/services/nowcoast/radar_meteo_imagery_nexrad_time/M apServer/WMSServer?


Add WMS Server Connection

URL: services/nowcoast/radar_meteo_imagery_nexrad_time/MapServer/WMSServer?

[✓] Use tiled maps (recommended)

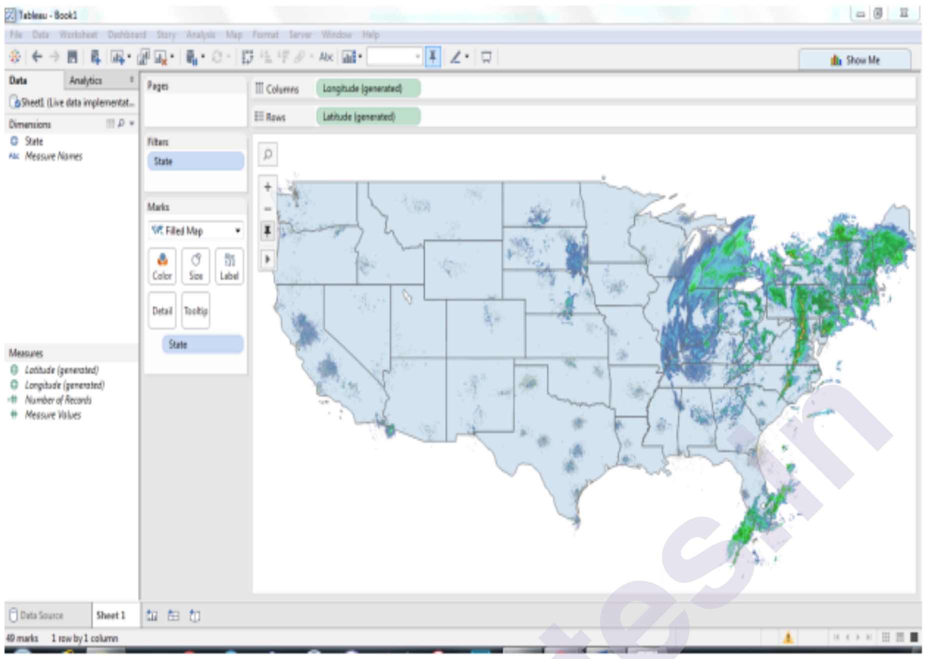
8. Now click on OK and then close the WMS server window pane and a map can be seen as shown



9. Change the map type to Filled map.

10. Change the color transparency to less than 20% so that the required map is visible.

11. T hen required map looks like this



7.9 VISUALIZING TREEMAPS USING TABLEAU

Tree map displays hierarchical data by using nested rectangles which together represent a whole. They were invented by Ben Shneiderman in 1992 as a way to visualize tree structures in a space-constrained layout. In a treemaps, each branch of a tree is a rectangle, which is tiled with smaller rectangles to represent sub-branches. This allows the viewer to easily see patterns that would be hard to spot in bar charts or area charts. The main benefits of tree maps are, they make efficient use of compact space. Treemaps are becoming popular in infographics and presentations, especially for data analysis that requires detailed views of many items. With this feature one can explore the data hierarchy effortlessly and simultaneously decent level of estimation is also possible for quantitative aspects of information.

Steps to visualize Treemaps in Tableau:

1. The data is about the various products sold at a fashion store in the cities of different states of the united states

2. Start with opening tableau and connect the data, which is of type excel to Tableau

3. Click on sheet1 which is at the bottom of the page to visualize the data

4. Now to the left of the screen you can see a small window showing Dimensions and Measures. The dimensions of the data are state, city, store name, product, SKU number, year, quarter, month and week. The measures of the data are Quantity sold and sales revenue

5. You can also see rows and columns at the top of the page, now drag sales revenue from the measures and drop it in the rows section. You can see a bar chart with single bar showing sales revenues

6. Now drag states from the dimension to the column section, you can see a bar chart showing the sales revenues of different states

7. Now click on Show Me button on the top right corner of the screen, you can see various maps suggested by the Tableau tool to visualize the data. The maps which are bright in color shows us that those maps are suitable to visualize the given data

8. Select the Tree Map by clicking it, you will see a Tree Map representing various states in green color with different shades

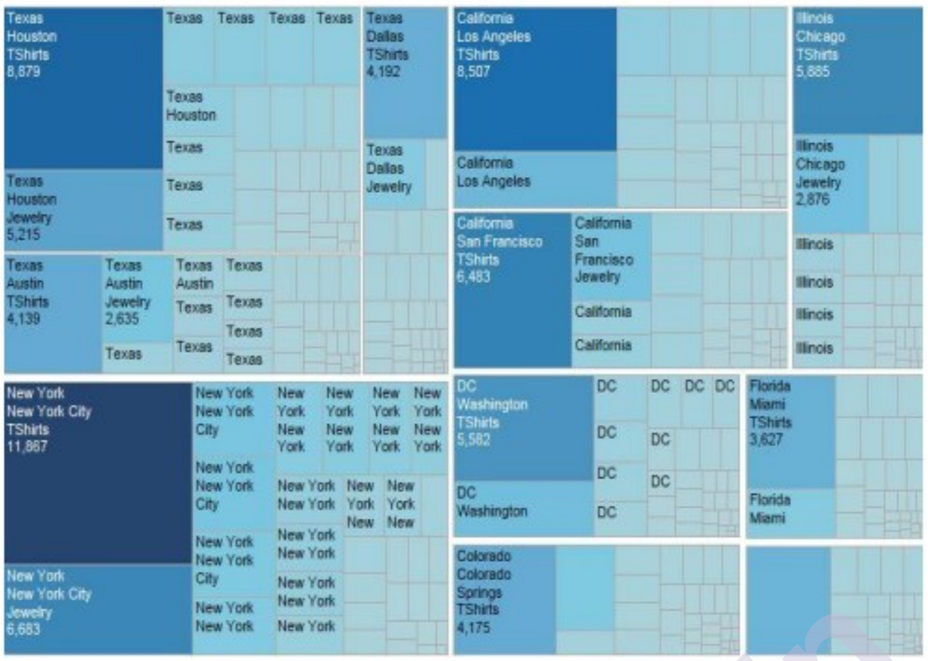
9. The rectangle which is large and bright represents the highest quantity according to sales revenue and similarly the smaller and lighter shade of the rectangles show less sales revenue

10. Now select the city tab from the dimension section and drop it on the Label in the Marks section. You can observe that the rectangles are split into even smaller rectangles according to the cities in each state

11. Now select the product from the dimension section and drop it on the Label in the Marks section. You can observe that the rectangles are split into even smaller rectangles according to the products.

12. Now select the Quantity sold tab from the dimension section and drop it on the Label in the Marks section. You can see that the actual quantity sold of each product.

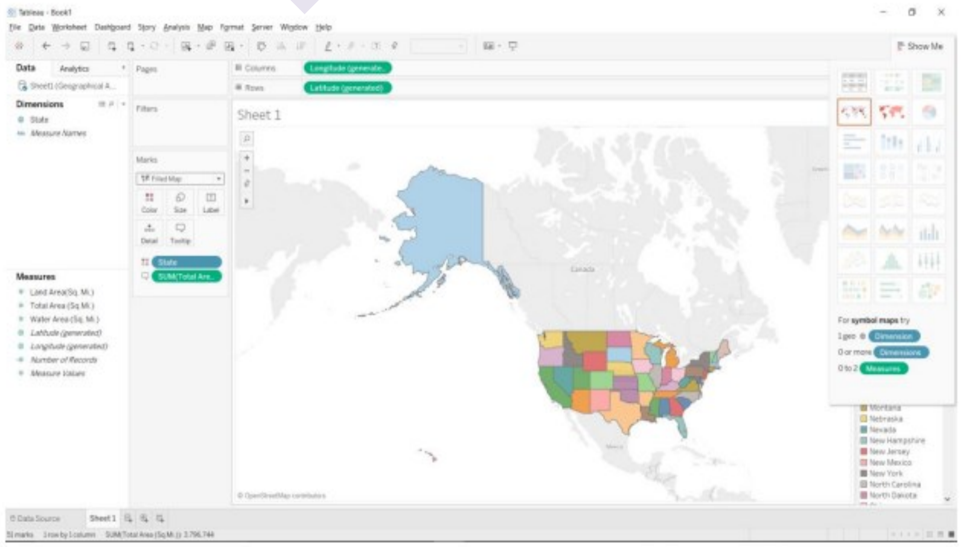
DatatreeMap.xlsx



Geographical map visualization using Tableau:

Steps to visualize Filled Maps in Tableau:

1. Open Tableau and connect excel sheet.
2. The data with states and their geographical data is shown in Tableau.
3. Drag and drop states onto columns and Total Area onto rows.
4. Drag and drop state onto color and total area onto tool tip.
5. Now select the filled map. Then the map is represented with the geographical data of each state as shown below.

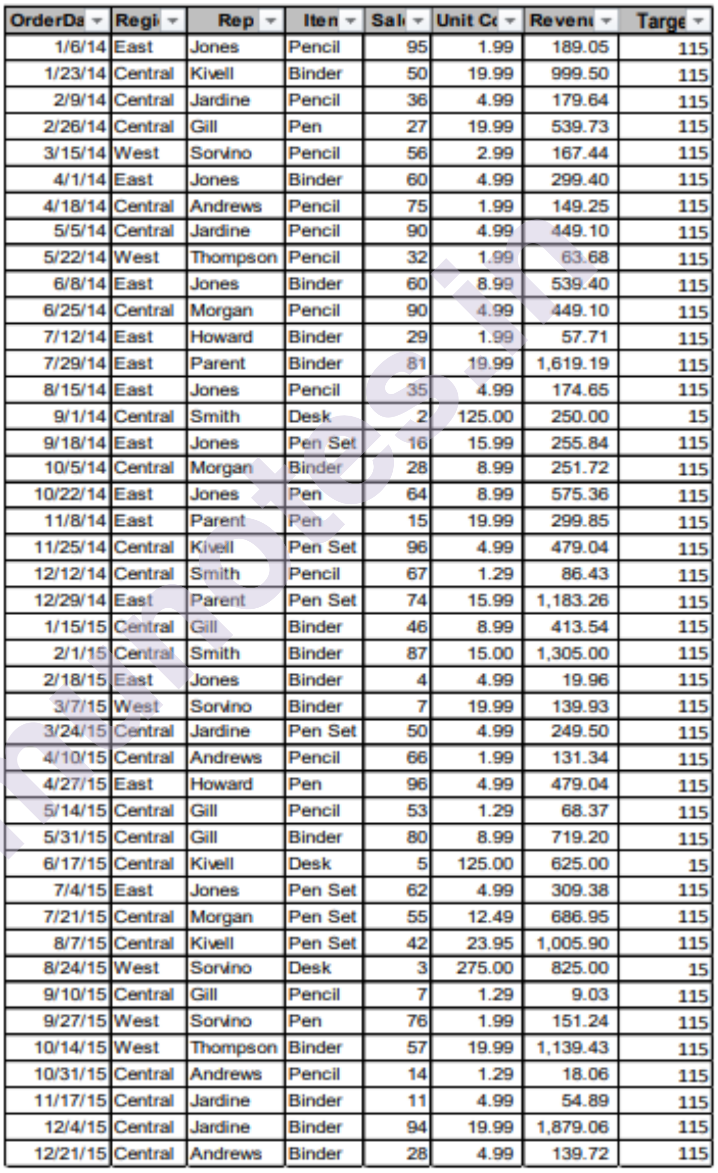


Visualization of Combo and Gauge Charts using QlikView

1. For this we have taken two different datasets.

- Dataset -1 (Cost_Student). Contains data which representing amount spend by the countries, on a student per year for primary to secondary education and postsecondary education.

- Dataset - 2 (Sales_Rep). Contains data, which represents the sales made by the representatives of a stationary chain around USA over particular dates.

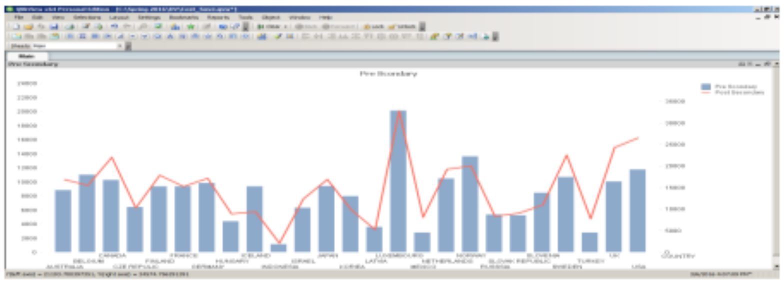


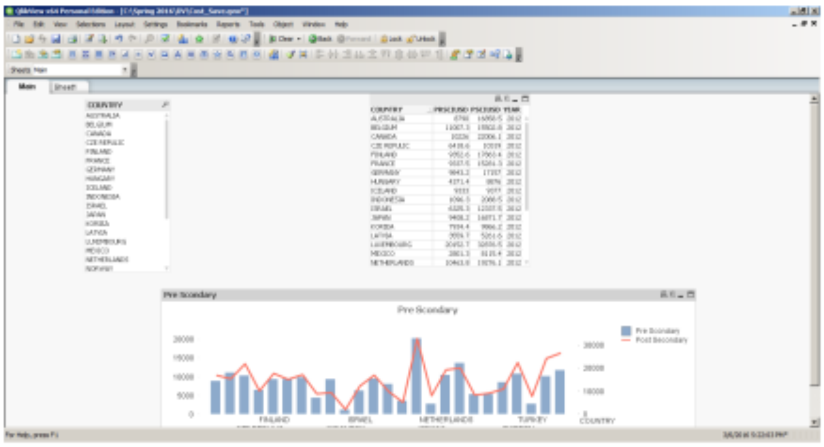
OrderDa	Regi	Rep	Iten	Sal	Unit Cc	Revenı	Targe
1/6/14	East	Jones	Pencil	95	1.99	189.05	115
1/23/14	Central	Kivell	Binder	50	19.99	999.50	115
2/9/14	Central	Jardine	Pencil	36	4.99	179.64	115
2/26/14	Central	Gill	Pen	27	19.99	539.73	115
3/15/14	West	Sorvino	Pencil	56	2.99	167.44	115
4/1/14	East	Jones	Binder	60	4.99	299.40	115
4/18/14	Central	Andrews	Pencil	75	1.99	149.25	115
5/5/14	Central	Jardine	Pencil	90	4.99	449.10	115
5/22/14	West	Thompson	Pencil	32	1.99	63.68	115
6/8/14	East	Jones	Binder	60	8.99	539.40	115
6/25/14	Central	Morgan	Pencil	90	4.99	449.10	115
7/12/14	East	Howard	Binder	29	1.99	57.71	115
7/29/14	East	Parent	Binder	81	19.99	1,619.19	115
8/15/14	East	Jones	Pencil	35	4.99	174.65	115
9/1/14	Central	Smith	Desk	2	125.00	250.00	15
9/18/14	East	Jones	Pen Set	16	15.99	255.84	115
10/5/14	Central	Morgan	Binder	28	8.99	251.72	115
10/22/14	East	Jones	Pen	64	8.99	575.36	115
11/8/14	East	Parent	Pen	15	19.99	299.85	115
11/25/14	Central	Kivell	Pen Set	96	4.99	479.04	115
12/12/14	Central	Smith	Pencil	67	1.29	86.43	115
12/29/14	East	Parent	Pen Set	74	15.99	1,183.26	115
1/15/15	Central	Gill	Binder	46	8.99	413.54	115
2/1/15	Central	Smith	Binder	87	15.00	1,305.00	115
2/18/15	East	Jones	Binder	4	4.99	19.96	115
3/7/15	West	Sorvino	Binder	7	19.99	139.93	115
3/24/15	Central	Jardine	Pen Set	50	4.99	249.50	115
4/10/15	Central	Andrews	Pencil	66	1.99	131.34	115
4/27/15	East	Howard	Pen	96	4.99	479.04	115
5/14/15	Central	Gill	Pencil	53	1.29	68.37	115
5/31/15	Central	Gill	Binder	80	8.99	719.20	115
6/17/15	Central	Kivell	Desk	5	125.00	625.00	15
7/4/15	East	Jones	Pen Set	62	4.99	309.38	115
7/21/15	Central	Morgan	Pen Set	55	12.49	686.95	115
8/7/15	Central	Kivell	Pen Set	42	23.95	1,005.90	115
8/24/15	West	Sorvino	Desk	3	275.00	825.00	15
9/10/15	Central	Gill	Pencil	7	1.29	9.03	115
9/27/15	West	Sorvino	Pen	76	1.99	151.24	115
10/14/15	West	Thompson	Binder	57	19.99	1,139.43	115
10/31/15	Central	Andrews	Pencil	14	1.29	18.06	115
11/17/15	Central	Jardine	Binder	11	4.99	54.89	115
12/4/15	Central	Jardine	Binder	94	19.99	1,879.06	115
12/21/15	Central	Andrews	Binder	28	4.99	139.72	115

2. Open QlikView. A QlikView getting started screen opens, now click on the New Document button on right below corner of the Wizard.

3. A new Window opens up (Kind of pop up) close the pop up window. (Since it is Auto chart generator, we do not want that wizard).

4. On closing the pop up, user will be able to see a Main blank sheet. Now have quick look on to the toolbar.
5. Over the tool bar below to the selections labe1, click on the Edit Script icon (Icon looks like pen on a paper).
6. A new Edit Script window opens up (We use this window to load an Excel file), bottom of the window under the Data from files Click Table Files button, and choose the saved Excel file, then click open.
7. A new File Wizard Type window opens up. Now select the Tables drop down and choose Cost_Student if it has not defaulted and click the Labels drop down and choose Embedded Labels. Click next until, next button disables (If it disables you are on final screen). Select check box load all and click finish.
8. On Finish you will be able to see Edit Script screen , with script added to load excel file(Script shown below) On Edit Script window tool bar below file , click the reload icon.It asks you save the file. LOAD * FROM [C:\Manusha\DataViz\DataViz-Qlik\QlikView_Dataset.xls] (biff, embedded labels, table is Cost_Per_Student$);
9. Now a sheet property window opens, please choose Country (Which works as a filter).
10. Right Click anywhere on the sheet, a window opens choose New Sheet Object and select Table box. New window open again, add all the available fields and click apply. You will be able to see a table box with content same as to excel sheet.
11. Right Click anywhere on the sheet, a window opens choose New Sheet Object and select Chart. New window open again, Select Combo Chart from chart type and click next.
12. Choose Country as dimension and click next now choose the expressions. • Expression 1 for bar chart Sum (PRSCIUSD), now click Add button to add one more expression Sum (PSCIUSD) and click finish.
13. Now we have seen the Combo Chart.
14. If you observe both pre-Secondary and post-secondary are on same axis. Let us split them. Right click on the chart and choose the properties, and navigate to the Axes tab, Select Post-Secondary Expression and choose the position Right (Top) click apply and ok. You will see charts similar to below.





COUNTRY	PRESCIUSD	PSCIUSD	YEAR
AUSTRALIA	6760	16950.5	2012
BELGIUM	11007.1	15502.6	2012
CANADA	10226	22006.1	2012
CZE REPUBLIC	6418.6	10019	2012
FINLAND	9952.6	17960.4	2012
FRANCE	9037.5	15264.5	2012
GERMANY	9843.2	17107	2012
HUNGARY	4171.4	8876	2012
ICELAND	9333	9077	2012
INDONESIA	1096.5	2066.5	2012
ISRAEL	4328.3	12337.5	2012
JAPAN	9408.2	16071.7	2012
KOREA	7934.4	9966.2	2012
LATVIA	3559.7	5261.6	2012
LUXEMBOURG	20452.7	32606.5	2012
MEXICO	2801.3	8115.4	2012
NETHERLANDS	10463.8	13276.1	2012

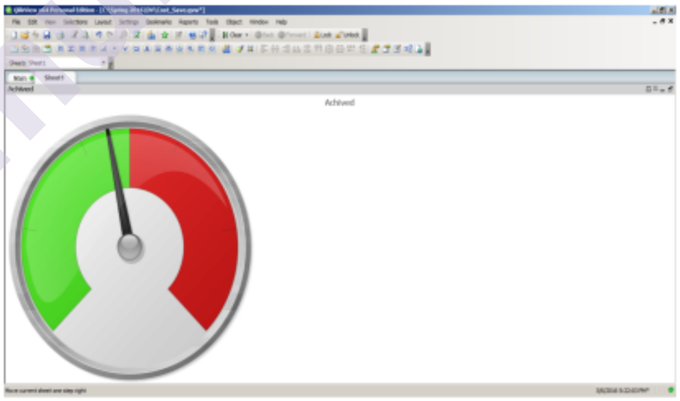
15. Now click on layout shown in the tool bar and add new sheet. A new sheet will open.

16. Repeat step 7 to step 12, in step 9 choose sales_Rep sheet and in step 11 Choose Rep to the filed display in list boxes.

17. Right Click anywhere on the sheet, a window opens choose New Sheet Object and select Chart. New window open again, Select Gauge from chart type and click next.

18. Gauge charts have no dimensions, so we do not select any dimension, will be moving directly to expression.

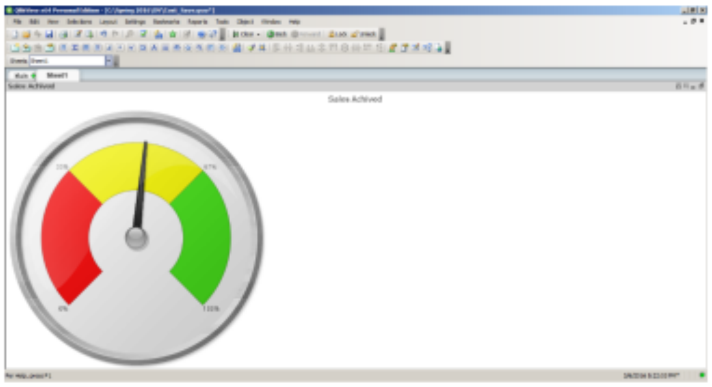
19. Since we are calculating the performance, we add a division between sales and targets (Sum (sales)/sum (targets)), add label for expression (Sales Achieved) finish you will see a chart like below.



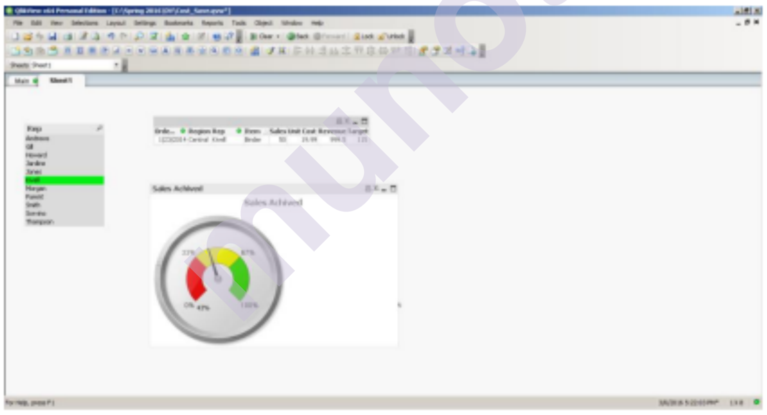
20. Let us make it more readable, right click, choose the properties, navigate to presentation add a segment in segment setup (we can go with 2 segments as well for clear user readability we choose 3) and change the colors of each segment by choosing color band (Red, Yellow, Green).

21. Now look for Show scale section left –below to segment setup add margin units a 4 and show labels on every major entry margin units 1

(Select the check box Show scale if it is not selected). 22. Now Switch to Number tab since we are representing it in percentage select the radio button Fixed to and select the check box Show in percentage and click apply.

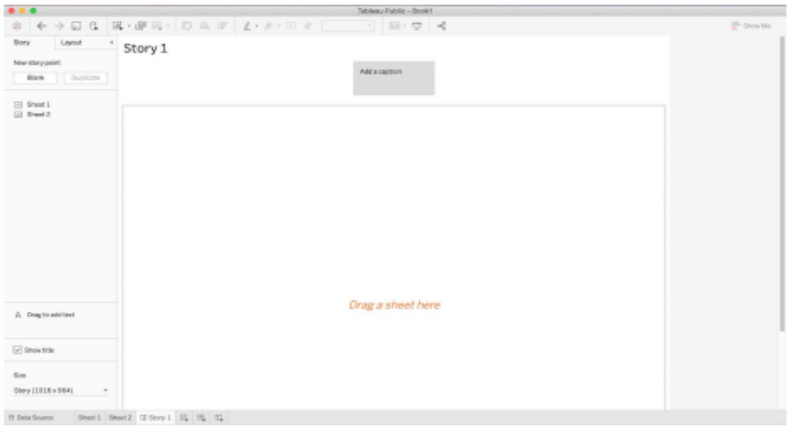


23. In order to see the needle value (Speedometer current value). Right click on the chart and select properties move to presentation tab and select Add button next to Text in Chart (located below right corner of the window) and add this formula(=num(Sum(Sales)/sum(Target),'##%')), and click apply, value appears on the top left corner of the chart. Press CTL +SHIFT to drag the text anywhere in the chart. Final Gauge chart with data along with chart looks like below.

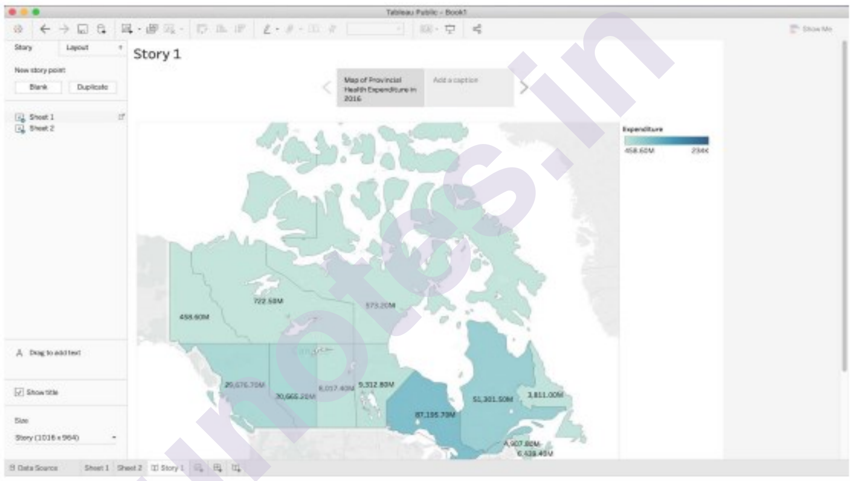


7.10 CREATING A STORY WITH TABLEAU PUBLIC

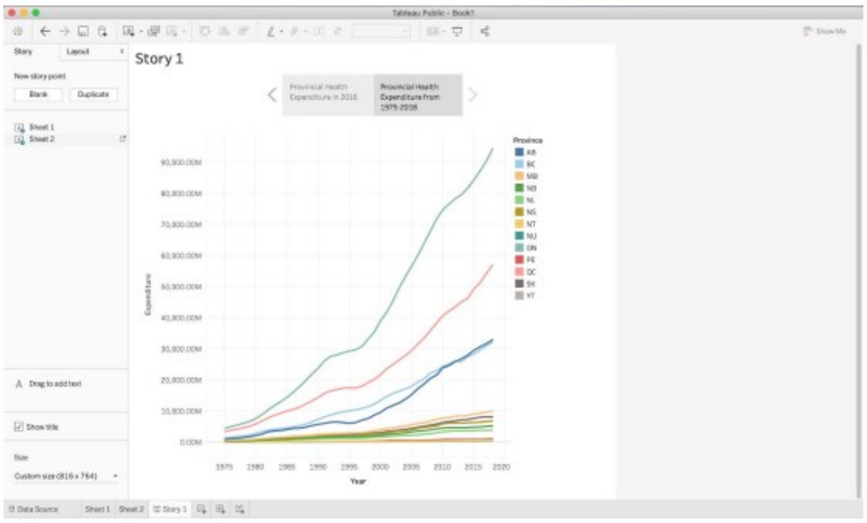
With Tableau public, you are able to organize your data in order to tell a meaningful story. This is beneficial when you are doing a presentation, creating an article, or uploading to a website, as it helps your audience understand your data. Stories are created through assembling the different worksheets and dashboards. We can highlight important data points, add text box and pictures to help convey our story. We will use our health expenditure worksheets to create a tailoring in story and illustrate the changes in Canada's spending in a meaningful way. To begin, select "New Story" at the bottom right of your screen



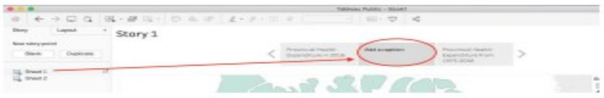
Drag "Sheet 1" and "Sheet 2" on to "Drag a sheet here". We can rename each storyboard by clicking "Add a caption". Rename Sheet 1 to "Provincial Health Expenditure in 2016".

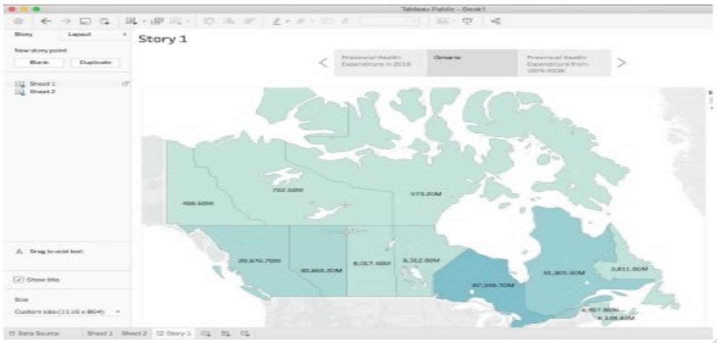


Use the arrows located on the side of the caption field to navigate to Sheet 2. Click on "Add a caption" and rename Sheet 2 to "Provincial Health Expenditure from 1975-2018".

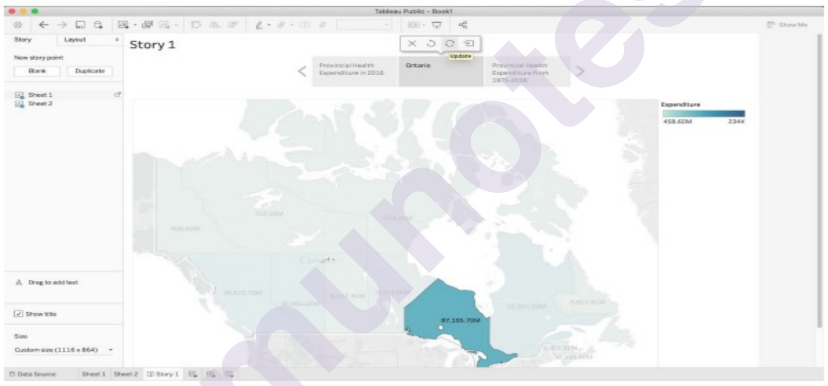

Story 1
Provincial Health Expenditure in 2016
Provincial Health Expenditure from 1975-2018
Expenditure
Year
Province

In this story, we are going to narrow in and draw attention to the province or territory that is spending the most amount of money on health. Drag an additional copy of "Sheet 1" and drop it between the two existing sheets. Select "Add a caption" and rename it to "Ontario".

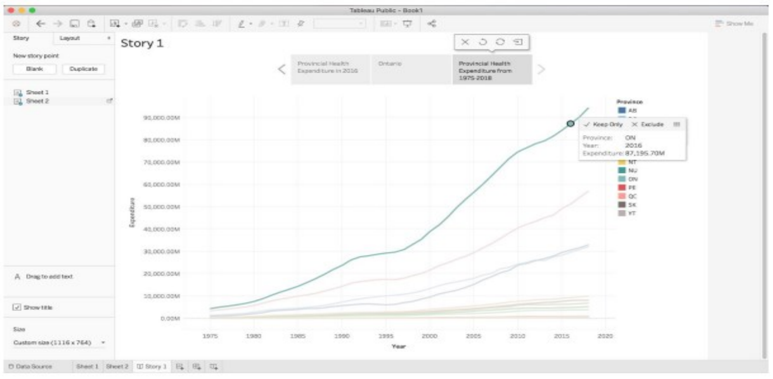

Story 1
Provincial Health Expenditure in 2016
Add a caption
Provincial Health Expenditure from 1975-2018
Sheet 1
Sheet 2


Story 1
Provincial Health Expenditure in 2016
Ontario
Provincial Health Expenditure from 1975-2018
87,195.70M

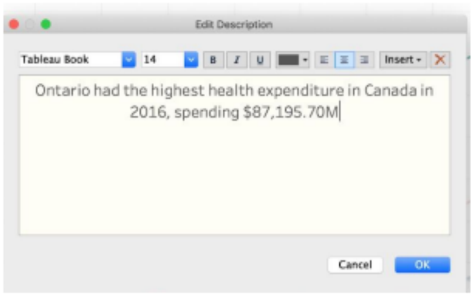
On the map, click on the province Ontario and then navigate to the caption field and select "Update". Your screen will show Ontario highlighted from the rest of Canada.


Story 1
Update
Provincial Health Expenditure in 2016
Ontario
Provincial Health Expenditure from 1975-2018
Expenditure
458.60M
87,195.70M

Select the right arrow to navigate to "Provincial Health Expenditure from 1975-2018". Hover over the line representing Ontario and select the data point representing health expenditure during the year 2016. Then click "Update".



We can add a textbox to label the highlighted pointed by dragging "Drag to add text" on to the line graph. Write a key message in the textbox, such as "Ontario had the highest health expenditure in Canada in 2016, spending $87,195.70M". Select "OK".



Ontario had the highest health expenditure in Canada in 2016, spending $87,195.70M

You can the edit the text box by selecting "More options" which will open a drop-down menu. Expand the text box by dragging the borders in order to show the full message.

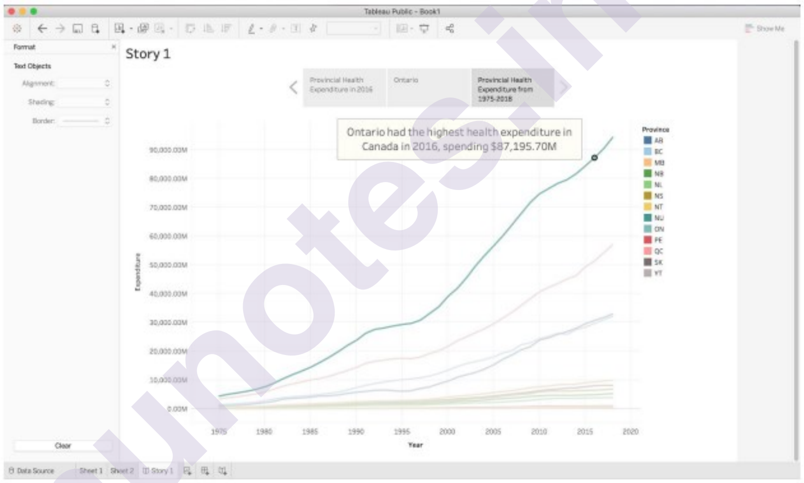
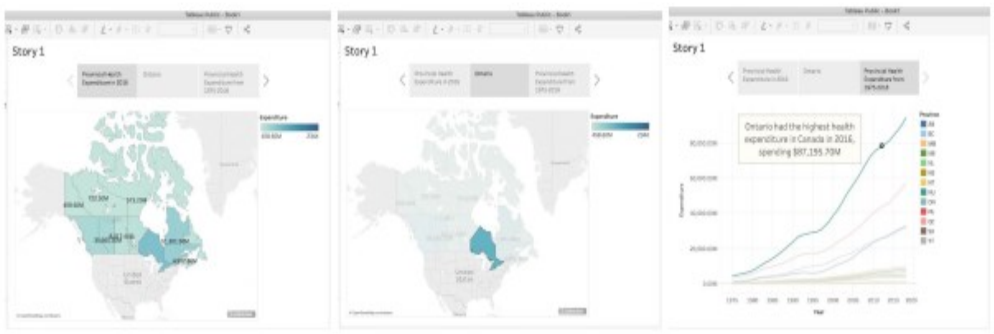


Tableau Public - Book1
Format
Text Objects
Story 1
Provincial Health Expenditure in 2016
Ontario
Provincial Health Expenditure from 1975-2016
Expenditure
Year
Province

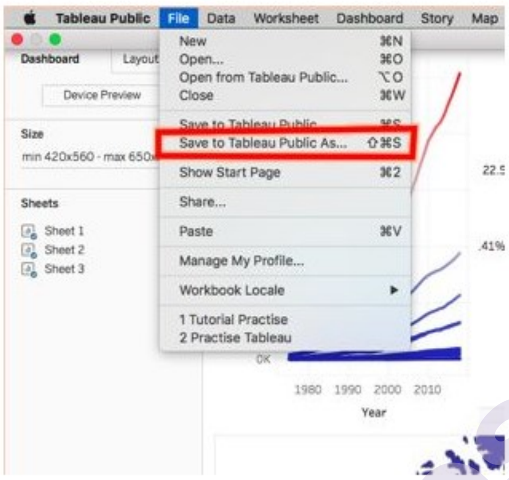
Ontario had the highest health expenditure in Canada in 2016, spending $87,195.70M

We have now created a story with three sheets of how Ontario had the highest health expenditure in the year 2016. If you choose to add a dashboard, it will allow your audience to play with data. You can navigate between the story as shown below:

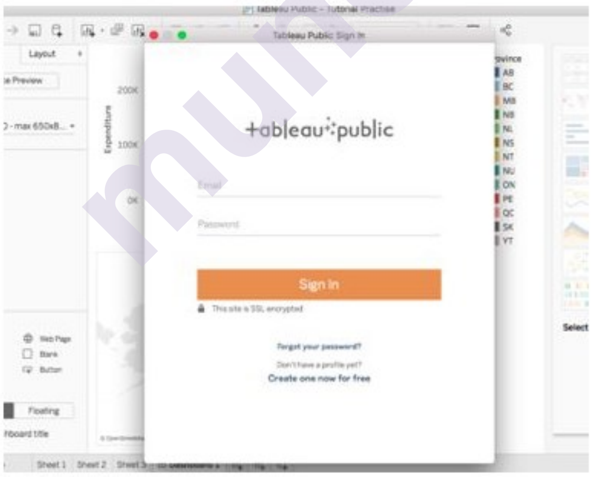


7.11 SAVING AND PUBLISHING YOUR TABLEAU PUBLIC WORKBOOK

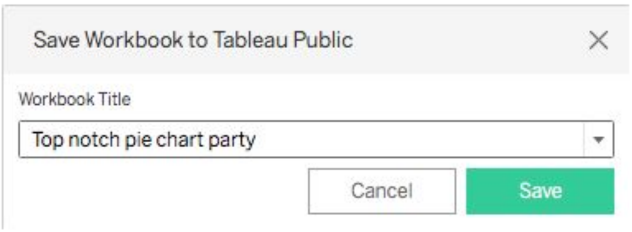
Once satisfied with your workbook, which includes sheets, dashboards, and stories, you can publish it to the Tableau Public website. This is the only way to save your work when using Tableau Public, so make sure to do it if you wish to return to the workbook in the future. Once ready to publish, select the "Save to Tableau Public As…" option under the "File" tab.



From here, you will likely be prompted to log in to the Tableau Public website. You can create an account for free if you have not already.



Then, create a title for your workbook.

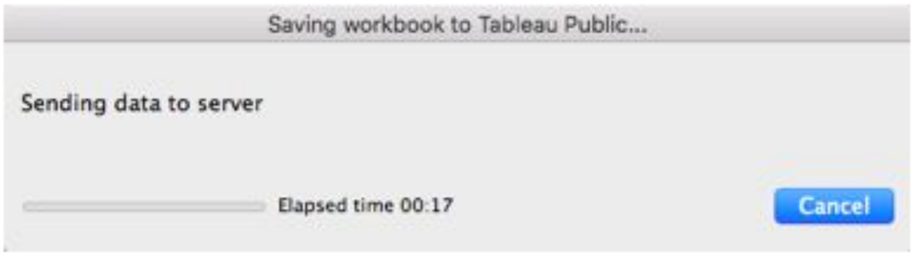

Save Workbook to Tableau Public

Workbook Title

Top notch pie chart party

Cancel Save

Your workbook will then be uploaded to the Tableau Public server. This may take a couple minutes.

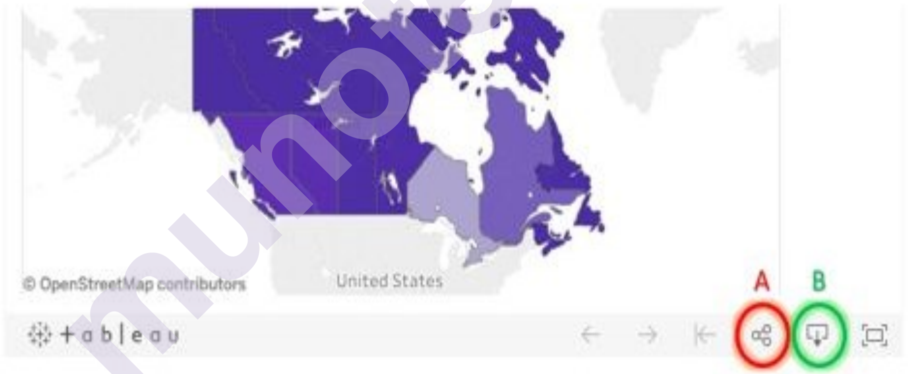


Saving workbook to Tableau Public...

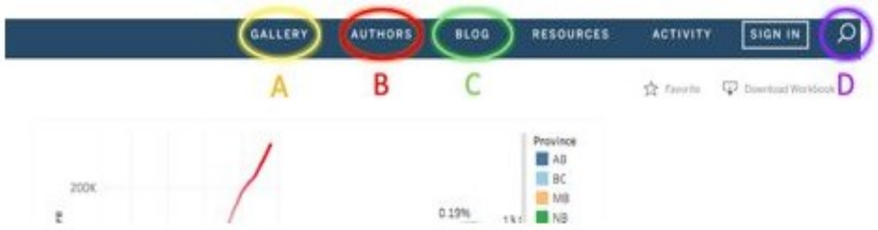
Sending data to server

Elapsed time 00:17

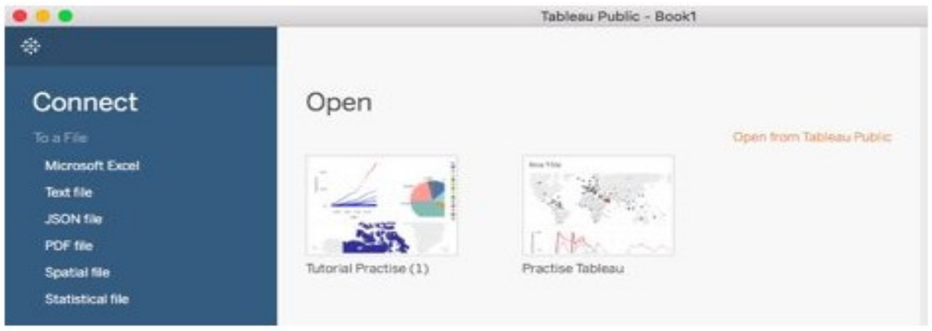
You will then be directed to the webpage on which your workbook is publicly available for download. Your workbook has now been published and saved! This page includes information on you, including a link to your Tableau Public profile, as well as additional information on the workbook. There are several options for making use of your work data visualizations that can be found at the bottom right of the workbook image. By clicking Share (A), you can get the code for embedding the workbook into a website or the link to find the workbook on tableau public. The Share button also gives you the option to share your work over email, Facebook, or Twitter. By clicking Download (B), you or any other Tableau Public users can download your workbook and make use of the data themselves.



On Tableau Public, all saved data visualizations are uploaded and available to all other users. This means you can use other users' work for your website, presentations, or research as well. To search for interesting data, there is a search function (D) at the top right corner of the webpage, along with highlighted visualizations (A), authors (B), and blogs (C).



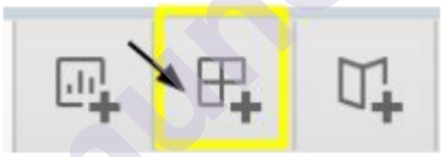
When you wish to access your published workbook, or any public workbooks you've downloaded, in the future, simply open the Tableau Public application and your workbooks will be there waiting for you!



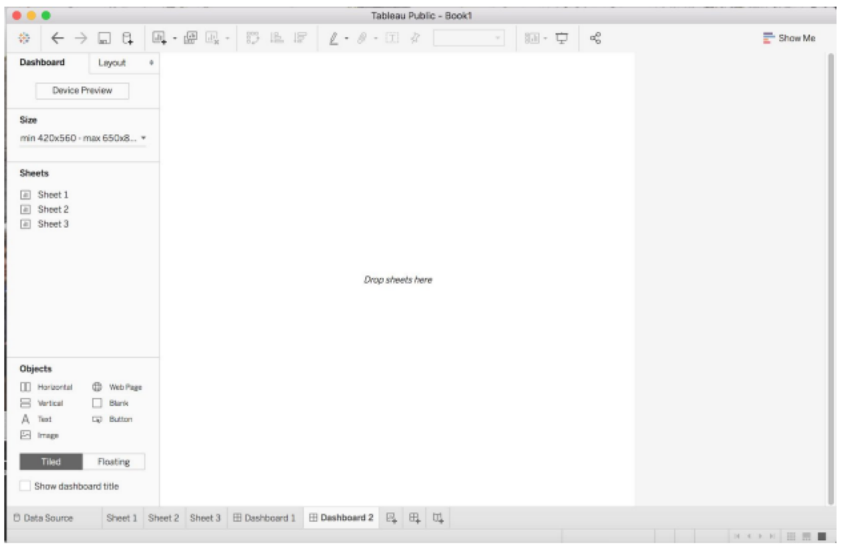
7.12 A DASHBOARD WITH TABLEAU

Dashboards are a great way to combine your data visualizations and have them interact with one another. A lot of businesses use dashboards to keep up-to-date in real time about key performance indicators at a glance. In this example, we will combine just two of our data visualizations, the map and the line graph from the first section of the tutorial, but in reality, it can be used to combine much visualization at once.

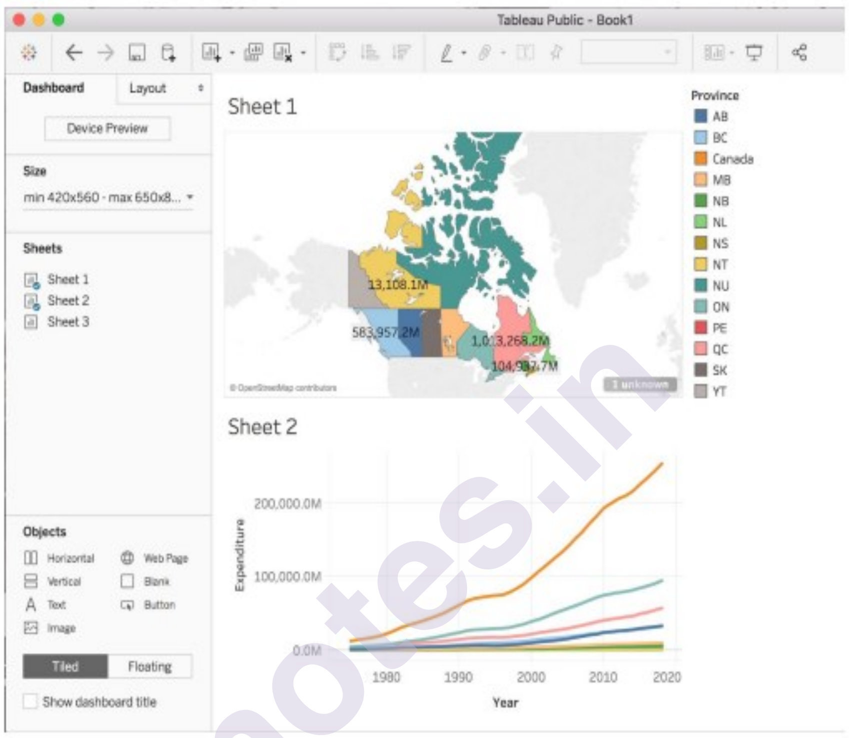
The first step in creating your dashboard is to open up the Dashboard tab at the bottom of the screen:



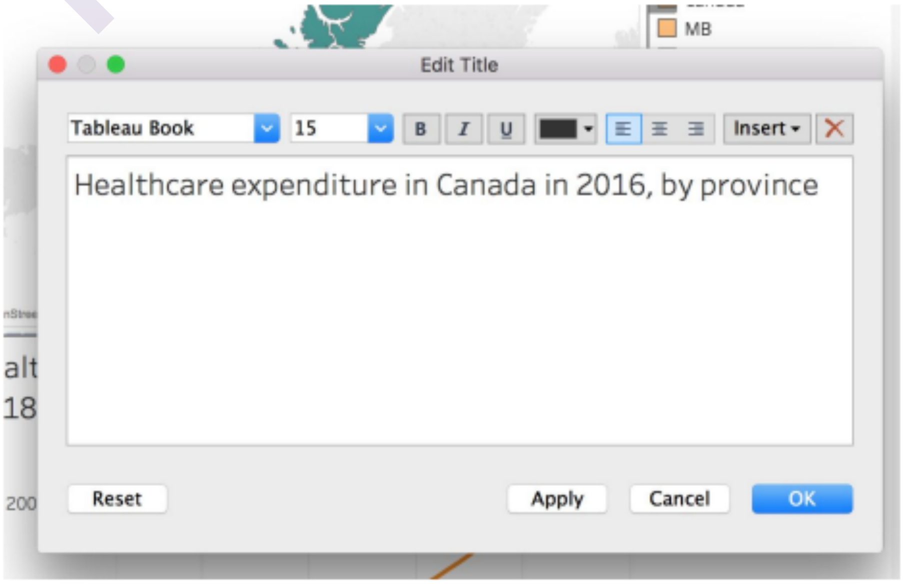
After clicking this icon, your screen should open to this:



This is your Dashboard Sheet. On the left side you can see that there is a list of the sheets you have made from your current data source. To build your dashboard, drag the sheet you want in to the centre where it says Drop sheets here. For our purposes, we will need to drag Sheet 1 and Sheet 2 where the map and line graph are saved. When you drag, you will notice an area of your screen will shade over where your graph will drop when you put it down. Organize your dashboard to look like the following:

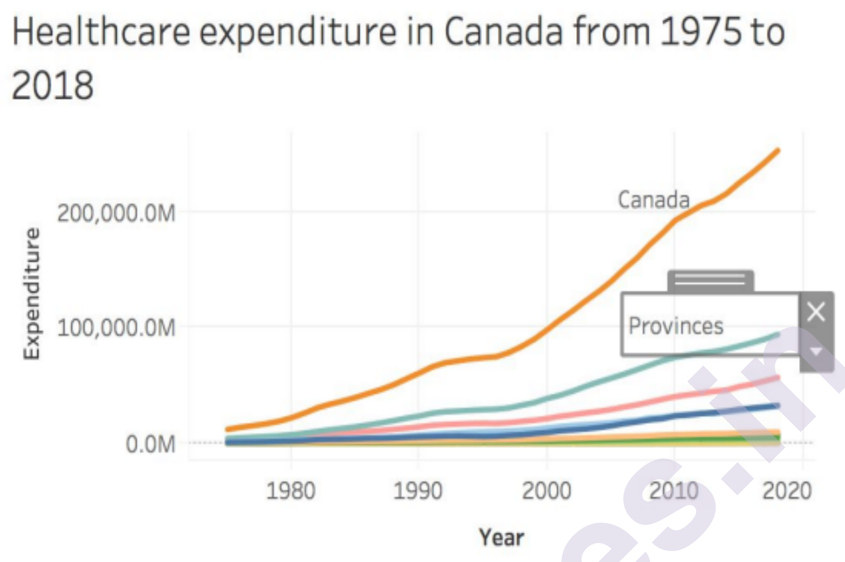


Now to add titles to the graphs that were chosen, double click on the automatic titles generated based on the sheet name, and a new window should appear, type in a title that describes the graph like so:

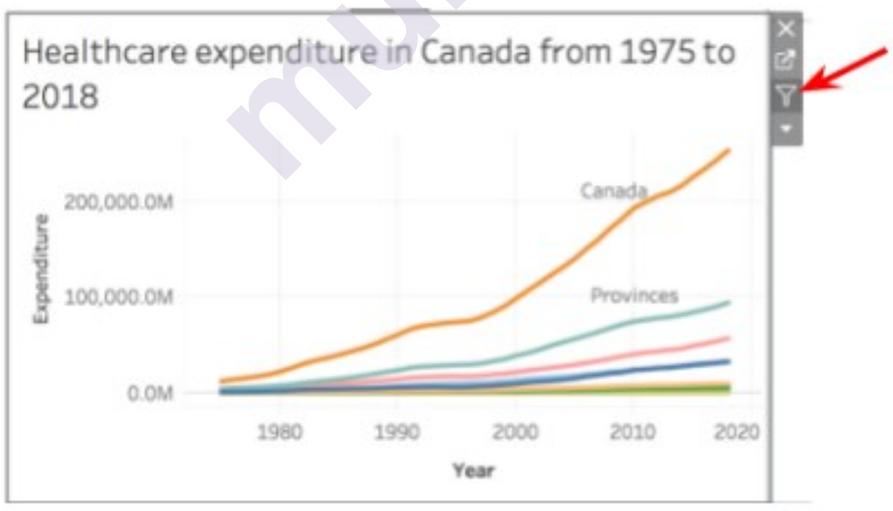


Healthcare expenditure in Canada in 2016, by province

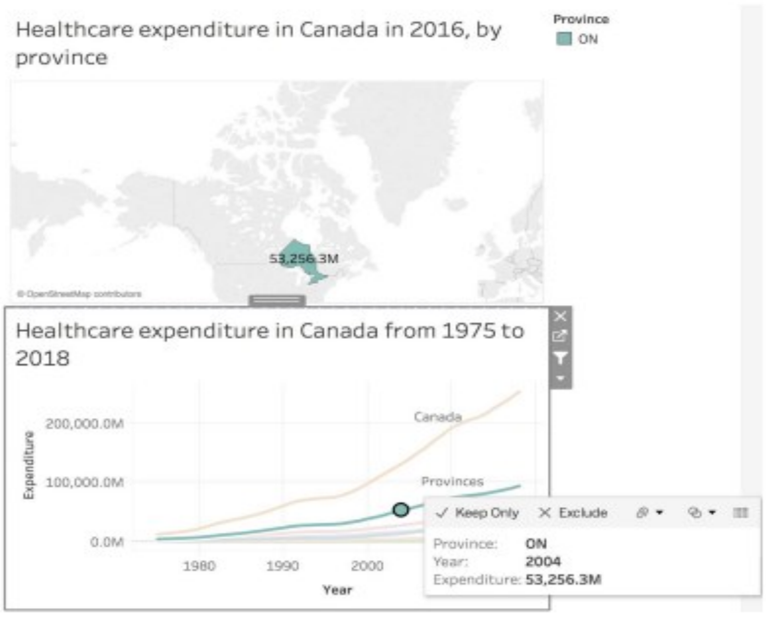
We can also add additional titles and objects to the dashboard by choosing an object from the Objects side panel and dragging it to the dashboard. We are going to add titles to the bottom line graph to differentiate between the Canada line and the provinces. To do this, drag A Text to the area near the orange line that corresponds to the sum of all provinces expenditure throughout the years. Type in "Canada". Drag A Text once more to label the remaining provinces. Your bottom graph should look like this:



Now, to add an interactive layer between the graphs, we can choose a graph that can act as a filter to the other. We will choose the line graph to act as a filter to the map. To do this, click on the line graph and a grey sidebar should appear. From this bar, click the filter icon to use this graph as a filter:



Now, when you click a given line, it will be highlighted on the above map:



Congrats, now you have an interactive dashboard that is ready to be published or saved!

7.13 QUESTIONS

1. What is data visualization in Tableau?
2. What is the difference between various BI tools and Tableau?
3. What are different Tableau products?
4. What is a parameter in Tableau?
5. Tell me something about measures and dimensions?
6. What are continuous and discrete field types?
7. What is aggregation and disaggregation of data?
8. What are the different types of joins in Tableau?
9. Tell me the different connections to make with a dataset?
10. What are the supported file extensions in Tableau?
11. What are the supported data types in Tableau?
12. What are sets?
13. What are groups in Tableau?
14. What are shelves?
15. Tell me something about Data blending in Tableau?
16. How do you generally perform load testing in Tableau?
17. Why would someone not use Tableau?
18. What is Tableau data engine?

19. What are the various types of filters in Tableau?
20. What are dual axes?
21. What is the difference between a tree and heat map?
22. What are extracts and schedules in Tableau server?
23. What are the components in a dashboard?
24. What is a TDE file?
25. What is the story in Tableau?
26. What are different Tableau files?
27. How do you embed views into webpages?

7.14 QUIZ

1. Data Values Available For The Visualization
   - **a. Table Calculation**
   - b. Basic Expression
   - c. LoD Expression
   - d. None of the above
2. Can we Perform All Kinds Of Joins Using Data Blending?
   - **a. Yes**
   - b. No
   - c. Sometimes
   - d. May Be
3. Current Version Of Tableau Is
   - a. 2020.1
   - b. 2020.2
   - **c. 2020.3**
   - d. 2020.4
4. Dimensions Can Be Aggregated
   - a. True
   - **b. False**
5. Which Of The Following Is NOT A Tableau Field Data Type?
   - a. Number (whole)
   - b. String
   - c. Boolean
   - **d. Float**

6. Default Aggregation Used For Tree Map
   a. Sum
   b. Avg
   c. Count
   d. Countd
7. What Are The File Extensions In Tableau ?
   a. Tableau Workbook (.twb)
   b. Tableau Packaged Workbook (.twbx)
   c. Tableau Data Source(.tds)
   d. All of the above
8. What Percent Of Total Profits Do The Top 10 Customer By Sales Represent?
   a. 3.50%
   b. 5.03%
   c. 17.54%
   d. None of the Above
9. What Type Of Join Is Used In Blending?
   a. Right join
   b. Left join
   c. Full join
   d. Inner join
10. What Is The Possible Cause Of The Data Not Being Updated?
    a. The data source configuration of Data Extrac needs to be refreshed
    b. Data Extrac needs an update
    c. The workers of Data Extrac are taking a timeout
    d. Services on Data Extrac are not running
11. Power BI Is A Product Of
    a. Oracle
    b. Facebook
    c. Microsoft
    d. SAP

12. What Are The Components Of A Dashboard?
    - a. Horizontal
    - b. Vertical
    - c. Image Extract
    - **d. All of the above**
13. How Many Maximum Tables Can You Join In Tableau?
    - a. 2
    - b. 8
    - c. 16
    - **d. 32**
14. Which of the accompanying isn't a Pattern Line display?
    - a. Straight Pattern Line
    - **b. Binomial Pattern Line**
    - c. Exponential Pattern Line
    - d. Logarithmic Pattern Line
15. For creating variable size bins we use ______
    - a. Sets
    - b. Groups
    - **c. Calculated fields**
    - d. Table Calculations
16. Can Tableau be installed on MacOS?
    - **a. True**
    - b. False
17. Default aggregation used for tree map
    - a. Avg
    - **b. Sum**
    - c. Count
    - d. Countd
18. What are different Tableau products?
    - a. Server
    - b. Reader
    - c. Online
    - **d. All of above**

19. Which one is not the best fitment for Tableau
    a. Self Service BI
    b. Mobile enablement
    c. Big Data connectivity
    d. Enterprise Deployment

20. Which type of chart is not available in Tableau V8
    a. Pie Chart
    b. Speed Dial
    c. Bullet Chart
    d. Bubble Chart

7.15 VIDEO LECTURES

1. Tableau Full Course - Learn Tableau in 6 Hours | Tableau Training for Beginners | Edureka. https://www.youtube.com/watch?v=aHaOIvR00So
2. Introduction to Tableau | How Tableau Works. https://www.youtube.com/watch?v=gWZtNdMko1k
3. Tableau Vs Excel. https://www.youtube.com/watch?v=ozGYSLrAc5o
4. Tableau Tutorial | Tableau Full Course - Learn Tableau In 6 Hours | Great Learning. https://www.youtube.com/watch?v=6mBtTNggkUk
5. Tableau Expert Full Course | Tableau Training | Tableau Tutorial | Tableau Full Course | Simplilearn. https://www.youtube.com/watch?v=DlBIhpDjhbY
6. Data Visualization with Tableau | Tableau Tutorial for Beginners in 2022 | Great Learning. https://www.youtube.com/watch?v=WAXfRAI96YA
7. Tableau Projects For Practice With Examples | Tableau Training For Beginners | Simplilearn. https://www.youtube.com/watch?v=5uzB4z4iN0g

7.16 MOOCS

❖ Data Visualization in Tableau. https://www.udacity.com/course/data-visualization-in-tableau--ud1006

❖ Tableau Tutorial for Beginners. https://www.udemy.com/course/tableau-tutorial-for-beginners/

❖ Tableau Certification Training Course. https://www.edureka.co/tableau-certification-training?

7.17 REFERENCES

1. https://www.techtarget.com/searchbusinessanalytics/definition/data-visualization

2. https://www.ibm.com/cloud/learn/data-visualization

3. https://www.tableau.com/learn/articles/data-visualization

4. https://policyviz.com/wp-content/uploads/woocommerce_uploads/2017/07/A-Guide-to-Advanced-Data-Visualization-in-Excel-2016-Final.pdf

❄❄❄❄❄❄❄