

RAMNIRANJAN JHUNJHUNWALA COLLEGE OF ARTS, SCIENCE &
COMMERCE

Big Data Analytics with Hadoop and Spark

Name- **Ekta Yadav**

Roll No- **627**



Hindi Vidya Prachar Samiti's
RAMNIRANJAN JHUNJHUNWALA COLLEGE
OF ARTS, SCIENCE & COMMERCE
(Empowered Autonomous College)
DEPARTMENT OF INFORMATION TECHNOLOGY
2023 – 2024

M.Sc. (I.T.) Part-1 SEM 2
RJSPGITE202P

Big Data Analytics with Hadoop and Spark

Name: Ekta Yadav

Roll No: 627

Index

Practical No.	Practical Aim	Date
1	Using basic Commands of Linux and Hadoop	18/12/2023
2	Using HBase Tool	08/01/2024
3	Using Hive	15/01/2024
4	Hive Partitioning, Bucketing, Joining and Sorting	05/02/2024
5	Using Pig and PigLatin	12/02/2024
6	RDD Actions and Transformations	26/02/2024
7	Creating PySpark Application	04/03/2024
8	Spark Dataframes and SQL	06/03/2024
9	Spark MLlib : Regression, Classification and Clustering	07/03/2024
10	Using Sqoop	11/03/2024

Practical No. 1: Using basic Commands of Linux and Hadoop

Community Edition DataBricks- <https://community.cloud.databricks.com/login.html>

Click this Link for getting Community edition of Databricks

The screenshot shows a browser window with the Databricks landing page. The page title is "Try Databricks free". It features a section for "Choose a cloud provider" with options for AWS, Microsoft Azure, and Google Cloud Platform. Below this is a "Continue" button. To the right, there's a note about agreeing to Privacy Policy and Terms of Service. At the bottom, there are logos for Nielsen, Grab, hyperloop, and edmunds.

- Create a dataframe for the specified data.

Code-

```
data = [[295, "South Bend", "Indiana", "IN", 101190, 112.9]]
columns = ["rank", "city", "state", "code", "population", "price"]
df1 = spark.createDataFrame(data, schema="rank LONG, city STRING, state STRING, code STRING, population LONG, price DOUBLE")
display(df1)
```

Output-

The screenshot shows a Jupyter Notebook cell with the following content:

```
1 data = [[295, "South Bend", "Indiana", "IN", 101190, 112.9]]
2 columns = ["rank", "city", "state", "code", "population", "price"]
3 df1 = spark.createDataFrame(data, schema="rank LONG, city STRING, state STRING, code STRING, population LONG, price DOUBLE")
4 display(df1)
```

The output pane shows the resulting DataFrame:

	rank	city	state	code	population	price
1	295	South Bend	Indiana	IN	101190	112.9

Below the table, it says "1 row | 17.06 seconds runtime" and "Refreshed 2 minutes ago". The command took 17.06 seconds and was run by ekta.yadav1015@rjcollege.edu.in at 12/16/2023, 9:19:56 PM on My Cluster2.

- Create and load a dataframe for the databricks sample dataset

/databricks-datasets/samples/population-vs-price/data_geo.csv

Code-

```
df2 = (spark.read  
    .format("csv")  
    .option("header", "true")  
    .option("inferSchema", "true")  
    .load("/databricks-datasets/samples/population-vs-price/data_geo.csv"))
```

Output-

Cmd 2

```
1 #Create and load a dataframe for the databricks sample dataset
2 #/databricks-datasets/samples/population-vs-price/data_geo.csv
3
4 df2 = (spark.read
5     .format("csv")
6     .option("header", "true")
7     .option("inferSchema", "true")
8     .load("/databricks-datasets/samples/population-vs-price/data_geo.csv")
9 )
```

▶ (2) Spark Jobs

▼ df2: pyspark.sql.DataFrame

- 2014 rank: integer
- City: string
- State: string
- State Code: string
- 2014 Population estimate: integer
- 2015 median sales price: double

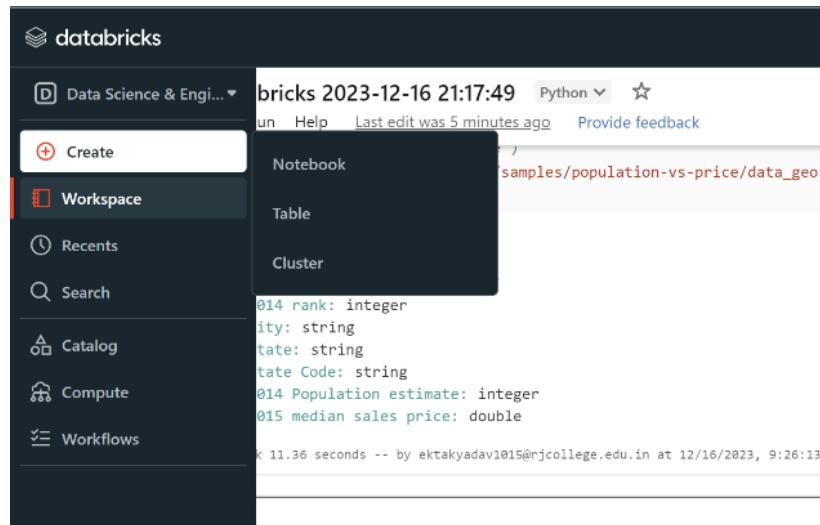
Command took 11.36 seconds -- by ekta.yadav1015@rjcollege.edu.in at 12/16/2023, 9:26:13 PM on My Cluster2

- Upload any sample dataset.csv file into Databricks and load the Data into dataframe.

Download data from github-

<https://github.com/mwaskom/seaborn-data/blob/master/iris.csv>

Step 1- Upload Data -Click Create

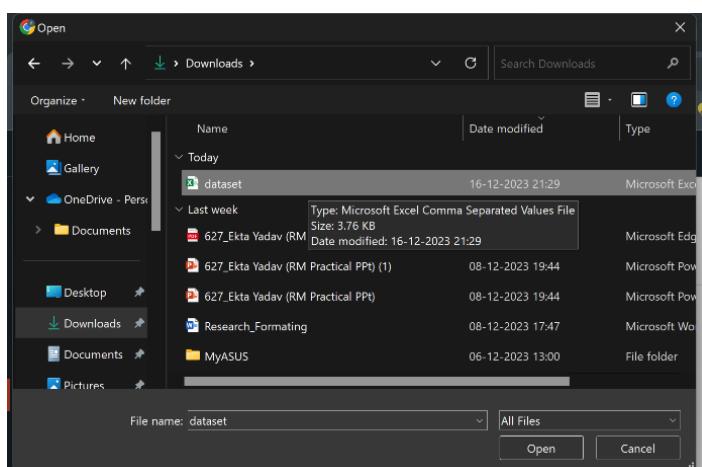


Step 2-Select Table Option

The screenshot shows the Databricks web interface. On the left, there's a sidebar with various options: Create, Workspace (which is currently selected and highlighted in red), Recents, Search, Catalog, Compute, and Workflows. A dropdown menu is open over the 'Create' button, showing three options: Notebook, Table, and Cluster. Below the sidebar, there's a code editor window displaying some JSON or YAML-like configuration. At the top right, there's a status bar showing the date and time: 'bricks 2023-12-16 21:17:49'.

Step 3- Drag file to upload or click to browser

The screenshot shows the 'Create New Table' page in Databricks. The top navigation bar has 'Data source' with a question mark icon, 'Upload File' (which is selected and highlighted in blue), 'S3', and 'Other Data Sources'. Below this, there's a section for 'DBFS Target Directory' with a dropdown set to '/FileStore/tables/'. There's also a note about files being accessible to everyone in the workspace. At the bottom, there's a large input field labeled 'Files' with the placeholder 'Drop files to upload, or click to browse'.



Step 4-File is Uploaded and click Create Table with UI Button

The screenshot shows the 'Create New Table' interface in Databricks. On the left is a sidebar with icons for Databricks, Data Source, DBFS, Cluster, Notebook, and Help. The main area has a dark header 'Create New Table'. Below it, there's a 'Data source' section with tabs for 'Upload File' (selected), 'S3', and 'Other Data Sources'. Under 'DBFS Target Directory', there's a text input field with '/FileStore/tables/' and '(optional)' next to a 'Select' button. A message below says 'Files uploaded to DBFS are accessible by everyone who has access to this workspace. Learn more'. In the 'Files' section, a single file 'dataset.csv' is listed with a green checkmark icon, a size of '3.9 KB', and a 'Remove file' link. Below this, a message says '✓File uploaded to /FileStore/tables/dataset.csv'. At the bottom are two buttons: 'Create Table with UI' (highlighted in blue) and 'Create Table in Notebook'.

Step 5- Create Cluster and select cluster – Attach and Run

This screenshot continues from the previous one, showing the 'Create New Table' interface. The 'Create Table with UI' button is still highlighted. Below it, a section titled 'Select a Cluster to Preview the Table' asks 'Choose a cluster with which you will read and preview the data.' A dropdown menu labeled 'Cluster' with the placeholder 'Select a cluster' is shown. At the bottom is a 'Preview Table' button.

The screenshot shows the 'Create New Table' interface in Databricks. A file named 'dataset.csv' has been uploaded to the DBFS target directory '/FileStore/tables/'. The file is 3.9 KB in size. Below the file list, there are two buttons: 'Create Table with UI' and 'Create Table in Notebook'. A note says 'Select a Cluster to Preview the Table' and 'Choose a cluster with which you will read and preview the data.' A dropdown menu for 'Cluster' shows 'My Cluster2' selected. A 'Preview Table' button is also visible.

Step 6- After Attach Cluster click on Preview Table Button

The screenshot shows the 'Create New Table' interface in Databricks, specifically the 'Specify Table Attributes' step. The 'Table Name' is set to 'dataset_csv'. The 'File Type' is set to 'CSV'. In the 'Table Preview' section, there is a note: 'Specify the Table Name, Database and Schema to add this to the data UI for other users to access'. Below this, there are fields for 'Create in Database' (set to 'Loading databases...'), 'Column Delimiter' (empty), and checkboxes for 'First row is header', 'Infer schema', and 'Multi-line'. There is also a 'Create Table' button. A 'Preview Table' button is visible on the left. The 'Table Preview' area is currently loading, as indicated by a circular progress icon.

Step 7- View the Your loaded Dataset

Create New Table

Preview Table

Specify Table Attributes

Table Name: dataset_csv

Create in Database: default

File Type: CSV

Column Delimiter: ,

First row is header:

Infer schema:

Multi-line:

Create Table

Create Table in Notebook

Table Preview

c0	c1	c2	c3	c4
sepal_length	sepal_width	petal_length	petal_width	species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa

Step 8- Copy the path of loaded data

Create New Table

Files

dataset.csv ✓
3.9 KB Remove file

✓ File uploaded to /FileStore/tables/dataset.csv

Create Table with UI **Create Table in Notebook**

Select a Cluster to Preview the Table

Choose a cluster with which you will read and preview the data.

Cluster: My Cluster2

Preview Table

Specify Table Attributes

Specify the Table Name, Database and Schema to add this to the data UI for other users to access

Table Name: dataset

Table Preview

After copy the path Write the code in Notebook-

```
#Upload any sample dataset.csv file into Databricks and load the Data into dataframe
```

```
df2 = (spark.read
    .format("csv")
    .option("header", "true")
    .option("inferSchema", "true")
    .load("/FileStore/tables/dataset.csv")
)
```

Output-

```
1 #Upload any sample dataset.csv file into Databricks and load the Data into dataframe
2 df2 = (spark.read
3     .format("csv")
4     .option("header", "true")
5     .option("inferSchema", "true")
6     .load("/FileStore/tables/dataset.csv")
7 )

▶ (2) Spark Jobs
└─ df2: pyspark.sql.dataframe.DataFrame
    sepal_length: double
    sepal_width: double
    petal_length: double
    petal_width: double
    species: string

Command took 1.70 seconds -- by ekta'yadav1015@rjcollege.edu.in at 12/16/2023, 9:47:13 PM on My Cluster2

Shift+Enter to run
Shift+Ctrl+Enter to run selected text
```

- Upload any sample dataset.json file into Databricks and load the data into the dataframe.

Code-

```
#Upload any sample dataset.json file into Databricks and load the data into the dataframe
```

```
data = [[295, "South Bend", "Indiana", "IN", 101190, 112.9]]
columns = ["rank", "city", "state", "code", "population", "price"]
```

```
df1 = spark.createDataFrame(data, schema="rank LONG, city STRING, state STRING, code
STRING, population LONG, price DOUBLE")
#display(df1)
```

```
df1.write.format("json").save("/tmp/json_data3")
```

Output-

```
1 #Upload any sample dataset.json file into Databricks and load the data into the dataframe
2 data = [[295, "South Bend", "Indiana", "IN", 101190, 112.9]]
3 columns = ["rank", "city", "state", "code", "population", "price"]
4
5 df1 = spark.createDataFrame(data, schema="rank LONG, city STRING, state STRING, code STRING, population LONG, price DOUBLE")
6 #display(df1)
7
8 df1.write.format("json").save("./tmp/json_data3")

▼ (1) Spark Jobs
  ▶ Job 16  View (Stages: 1/1)
  └─ df1: pyspark.sql.dataframe.DataFrame
      rank: long
      city: string
      state: string
      code: string
      population: long
      price: double

Command took 3.38 seconds -- by ekta'yadav1015@rjcollege.edu.in at 12/16/2023, 9:51:07 PM on My Cluster2

Shift+Enter to run
Shift+Ctrl+Enter to run selected text
```

- Save the Dataframes data into the .csv file/.json file.

Code-

```
#Save the Dataframes data into the .csv file/.json file.
df3 = spark.read.format("json").json("/tmp/json_data3")
display(df3)
```

Output-

The screenshot shows a Jupyter Notebook cell with the following code:

```
1 #Save the Dataframes data into the .csv file/.json file.
2 df3 = spark.read.format("json").json("/tmp/json_data3")
3 display(df3)
```

Below the code, the output shows:

- (2) Spark Jobs
- df3: pyspark.sql.dataframe.DataFrame = [city: string, code: string ... 4 more fields]

A table view is displayed with the following data:

	city	code	population	price	rank	state
1	South Bend	IN	101190	112.9	295	Indiana

Runtime: 1.88 seconds

Command took 1.88 seconds -- by ekta.yadav1015@rjcollege.edu.in at 12/16/2023, 9:53:46 PM on My Cluster2

Shift+Enter to run
Shift+Ctrl+Enter to run selected text

Printing the Schema of dataframe-

```
df.printSchema()
```

Output-

The screenshot shows a Jupyter Notebook cell with the following code:

```
1 #Save the Dataframes data into the .csv file/.json file.
2 df3 = spark.read.format("json").json("/tmp/json_data3")
3 display(df3)
4 #Print the schema
5 df3.printSchema()
6
```

Below the code, the output shows:

- (2) Spark Jobs
- df3: pyspark.sql.dataframe.DataFrame = [city: string, code: string ... 4 more fields]

A table view is displayed with the following data:

	city	code	population	price	rank	state
1	South Bend	IN	101190	112.9	295	Indiana

Runtime: 1.41 seconds

root
|-- city: string (nullable = true)
|-- code: string (nullable = true)
|-- population: long (nullable = true)
|-- price: double (nullable = true)
|-- rank: long (nullable = true)
|-- state: string (nullable = true)

Command took 1.41 seconds -- by ekta.yadav1015@rjcollege.edu.in at 12/16/2023, 9:55:20 PM on My Cluster2

- Create dataframe to store the following data

OrderID	Product	SalesAmount	Country
1	Apple	12345	India
2	Grapes	43857	UK
3	PineApple	2345	US

Code-

```
#Creating DataFrame for Given Dataset
data = [[1, "Apple", "12345", "India"], [2, "Grapes", "43857", "UK"], [3, "Banana", "2345", "US"]]
columns = ["OrderID", "Product", "SalesAmount", "Country"]

df1 = spark.createDataFrame(data, schema="OrderID LONG, Product STRING, SalesAmount STRING, Country STRING")
display(df1)
```

Output-

```
> (3) Spark Jobs
  ▾ df1: pyspark.sql.dataframe.DataFrame
    OrderID: long
    Product: string
    SalesAmount: string
    Country: string

Table +
```

	OrderID	Product	SalesAmount	Country
1	1	Apple	12345	India
2	2	Grapes	43857	UK
3	3	Banana	2345	US

↓ 3 rows | 0.82 seconds runtime Refreshed now

```
root
|-- OrderID: long (nullable = true)
|  |-- Product: string (nullable = true)
|  |-- SalesAmount: string (nullable = true)
|  |-- Country: string (nullable = true)

Command took 0.82 seconds -- by ektayadav1015@rjcollege.edu.in at 12/16/2023, 10:12:28 PM on My Cluster2
```

Shift+Enter to run
Shift+Ctrl+Enter to run selected text

Printing the Dataframe schema-

```
#printing the Dataframe Schema
df1.printSchema()
```

Output-

```

1  #Creating Dataframe for given dataset
2  data = [[1, "Apple", "12345", "India"], [2, "Grapes", "2345", "UK"], [3, "Banana", "234", "US"]]
3  columns = ["OrderID", "Product", "SalesAmount", "Country"]
4
5  df1 = spark.createDataFrame(data, schema="OrderID LONG, Product STRING, SalesAmount STRING, Country STRING")
6  display(df1)
7
8  #printing the Dataframe Schema
9  df1.printSchema()
10

▶ (3) Spark Jobs
  ▶ df1: pyspark.sql.dataframe.DataFrame
    OrderID: long
    Product: string
    SalesAmount: string
    Country: string

Table + 



|   | OrderID | Product | SalesAmount | Country |
|---|---------|---------|-------------|---------|
| 1 | 1       | Apple   | 12345       | India   |
| 2 | 2       | Grapes  | 2345        | UK      |
| 3 | 3       | Banana  | 234         | US      |



↓ 3 rows | 0.98 seconds runtime
root

```

Finding the Union of Dataframe-

Syntax-

```
# Returns a DataFrame that combines the rows of df1 and df2
df = df1.union(df2)
```

Code-

```
# Returns a DataFrame that combines the rows of df1 and df2
data = [[1, "Apple", "12345", "India"], [2, "Grapes", "43857", "UK"], [3, "Banana", "2345", "US"]]
columns = ["OrderID", "Product", "SalesAmount", "Country"]

df1 = spark.createDataFrame(data, schema="OrderID LONG, Product STRING, SalesAmount STRING, Country STRING")
display(df1)

data = [[4, "Chiku", "12345", "India"], [5, "Grava", "43857", "UK"], [6, "Lichhi", "2345", "US"]]
columns = ["OrderID", "Product", "SalesAmount", "Country"]

df2 = spark.createDataFrame(data, schema="OrderID LONG, Product STRING, SalesAmount STRING, Country STRING")
display(df2)

dfu = df1.union(df2)
display(dfu)
```

Cmd 7

```

1  # Returns a DataFrame that combines the rows of df1 and df2
2  data = [[1, "Apple", "12345", "India"], [2, "Grapes", "43857", "UK"], [3, "Banana", "2345", "US"]]
3  columns = ["OrderID", "Product", "SalesAmount", "Country"]
4
5  df1 = spark.createDataFrame(data, schema="OrderID LONG, Product STRING, SalesAmount STRING, Country STRING")
6  display(df1)
7
8  data = [[4, "Chiku", "12345", "India"], [5, "Grava", "43857", "UK"], [6, "Lichhi", "2345", "US"]]
9  columns = ["OrderID", "Product", "SalesAmount", "Country"]
10
11 df2 = spark.createDataFrame(data, schema="OrderID LONG, Product STRING, SalesAmount STRING, Country STRING")
12 display(df2)
13
14 dfu = df1.union(df2)
15 display(dfu)

▶ (9) Spark Jobs
▶ df1: pyspark.sql.dataframe.DataFrame = [OrderID: long, Product: string ... 2 more fields]
▶ df2: pyspark.sql.dataframe.DataFrame = [OrderID: long, Product: string ... 2 more fields]
▶ dfu: pyspark.sql.dataframe.DataFrame = [OrderID: long, Product: string ... 2 more fields]

```

Output-

Table +

	OrderID	Product	SalesAmount	Country
1	1	Apple	12345	India
2	2	Grapes	43857	UK
3	3	Banana	2345	US

↓ 3 rows | 2.36 seconds runtime

Table +

	OrderID	Product	SalesAmount	Country
1	4	Chiku	12345	India
2	5	Grava	43857	UK
3	6	Lichhi	2345	US

↓ 3 rows | 2.36 seconds runtime

Table +

	OrderID	Product	SalesAmount	Country
1	1	Apple	12345	India
2	2	Grapes	43857	UK
3	3	Banana	2345	US
4	4	Chiku	12345	India
5	5	Grava	43857	UK
6	6	Lichhi	2345	US

↓ 6 rows | 2.36 seconds runtime

Command took 2.36 seconds -- by ektakyadav1015@rjcollege.edu.in at 12/16/2023, 10:18:40 PM on My Cluster2

Shift+Enter to run
Shift+Ctrl+Enter to run selected text

Save the Data to Table format-

Syntax-

```
df.write.saveAsTable("<table name>")
```

Code-

```
dfu.write.saveAsTable("tableORDER")
```

Output-

```
Cmd 8
1 dfu.write.saveAsTable("tableORDER")
▶ (6) Spark Jobs
Command took 26.36 seconds -- by ekta'yadav1015@rjcollege.edu.in at 12/16/2023, 10:21:10 PM on My Cluster2

Shift+Enter to run
Shift+Ctrl+Enter to run selected text
```

```
1 dfu.write.saveAsTable("tableORDER")
▼ (6) Spark Jobs
  ▶ Job 55 View (Stages: 1/1)
  ▶ Job 56 View (Stages: 1/1)
  ▶ Job 58 View (Stages: 1/1)
  ▶ Job 59 View (Stages: 1/1, 1 skipped)
  ▶ Job 60 View (Stages: 1/1, 1 skipped)
  ▶ Job 61 View (Stages: 1/1, 2 skipped)
Command took 26.36 seconds -- by ekta'yadav1015@rjcollege.edu.in at 12/16/2023, 10:21:10 PM on My Cluster2

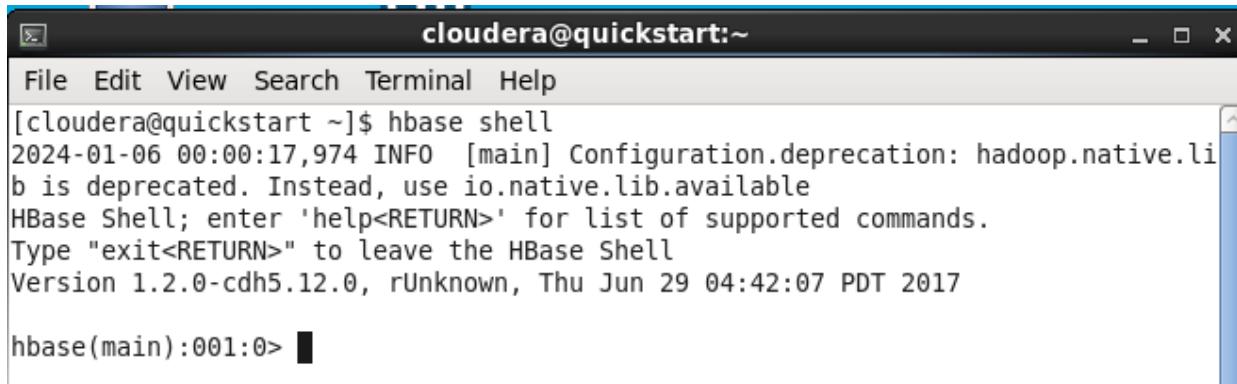
Shift+Enter to run
Shift+Ctrl+Enter to run selected text
```

Practical 2: Using HBase Tool

Aim: Using the HBase commands to store and query the columnar data.

1. Start with HBase

\$hbase shell



```
cloudera@quickstart:~
```

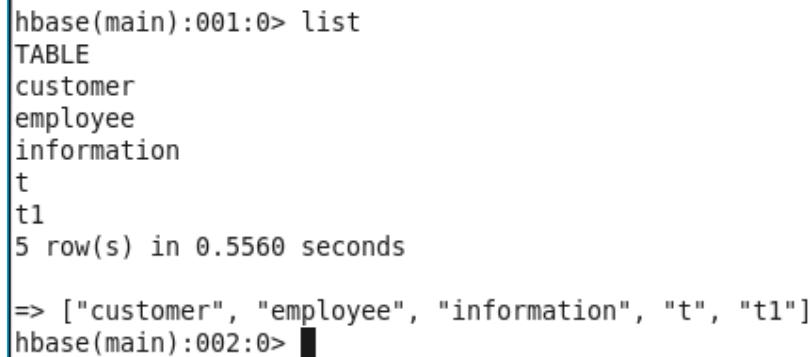
File Edit View Search Terminal Help

```
[cloudera@quickstart ~]$ hbase shell
2024-01-06 00:00:17,974 INFO [main] Configuration.deprecation: hadoop.native.lib is deprecated. Instead, use io.native.lib.available
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.0-cdh5.12.0, rUnknown, Thu Jun 29 04:42:07 PDT 2017
```

```
hbase(main):001:0> 
```

2. List the existing tables.

hbase>list



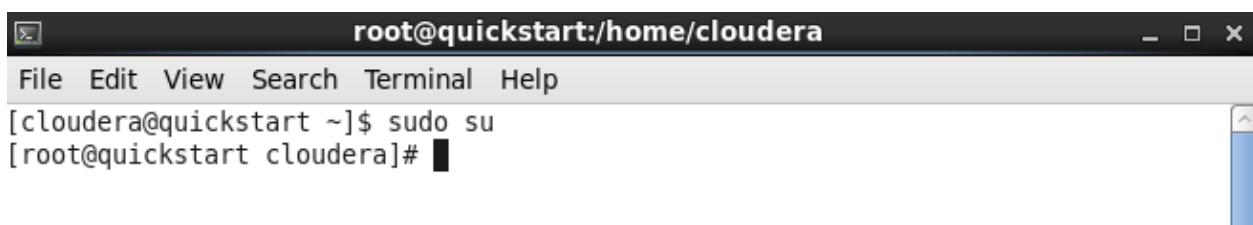
```
hbase(main):001:0> list
TABLE
customer
employee
information
t
t1
5 row(s) in 0.5560 seconds

=> ["customer", "employee", "information", "t", "t1"]
hbase(main):002:0> 
```

3. To restart the HBase services use the following commands

Get Logged in as a super user

\$sudo su



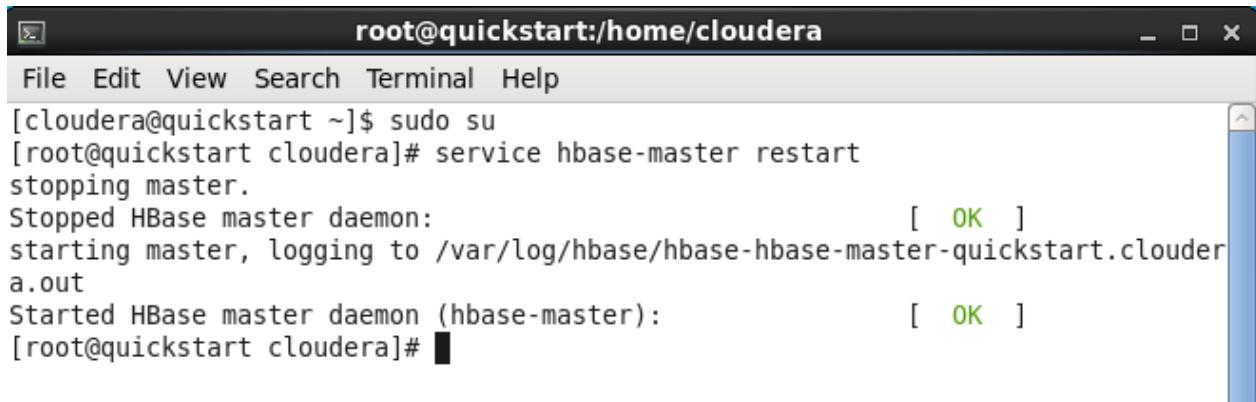
```
root@quickstart:/home/cloudera
```

File Edit View Search Terminal Help

```
[cloudera@quickstart ~]$ sudo su
[root@quickstart cloudera]# 
```

Restart the hbase-master service

```
$service hbase-master restart
```

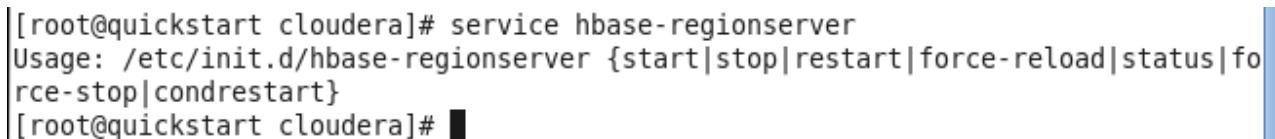


A terminal window titled "root@quickstart:/home/cloudera". The window shows the command "sudo su" followed by "service hbase-master restart". The output indicates the master is stopping, then a stopped HBase master daemon is started, and finally a new HBase master daemon (hbase-master) is started. Both the stopping and starting processes are marked with "[OK]".

```
[cloudera@quickstart ~]$ sudo su
[root@quickstart cloudera]# service hbase-master restart
stopping master.
Stopped HBase master daemon: [ OK ]
starting master, logging to /var/log/hbase/hbase-hbase-master-quickstart.cloudera.out
Started HBase master daemon (hbase-master): [ OK ]
[root@quickstart cloudera]#
```

Restart the hbase-regionserver service

```
$service hbase-regionserver
```

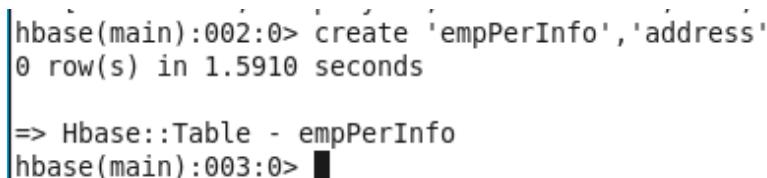


A terminal window titled "root@quickstart cloudera". The command "service hbase-regionserver" is run, which displays the usage information for the service. It includes options like start, stop, restart, force-reload, status, force-stop, and condrestart.

```
[root@quickstart cloudera]# service hbase-regionserver
Usage: /etc/init.d/hbase-regionserver {start|stop|restart|force-reload|status|force-stop|condrestart}
[root@quickstart cloudera]#
```

4. Design the schema for storing the employees personal information.

```
hbase>create 'empPerInfo','address'
```



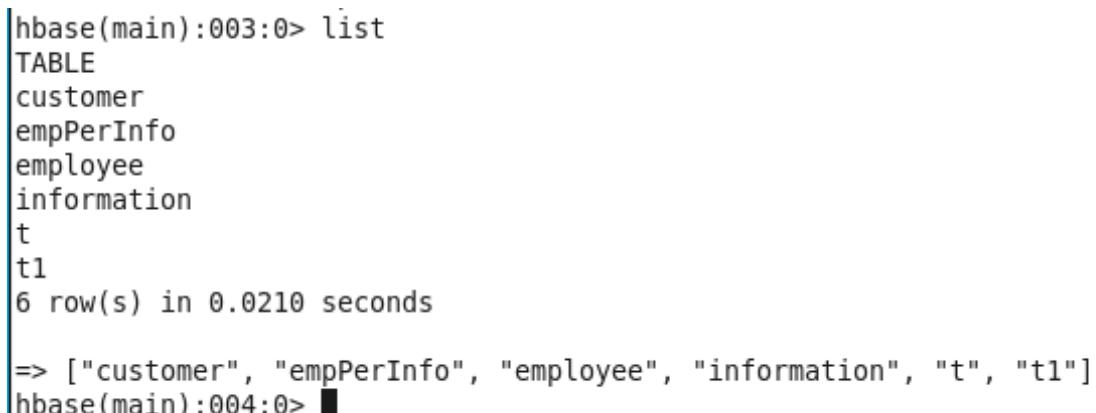
An Hbase shell session. The command "create 'empPerInfo','address'" is run, which creates a table named "empPerInfo" with a single column family "address". The response shows 0 row(s) created in 1.5910 seconds. The table is then listed as "=> Hbase::Table - empPerInfo".

```
hbase(main):002:0> create 'empPerInfo','address'
0 row(s) in 1.5910 seconds

=> Hbase::Table - empPerInfo
hbase(main):003:0>
```

5. Check for the creation of the table

```
hbase>list
```



An Hbase shell session. The command "list" is run, displaying a list of tables: customer, empPerInfo, employee, information, t, and t1. The response shows 6 row(s) in 0.0210 seconds. The list is then converted to a JSON array: ["customer", "empPerInfo", "employee", "information", "t", "t1"].

```
hbase(main):003:0> list
TABLE
customer
empPerInfo
employee
information
t
t1
6 row(s) in 0.0210 seconds

=> ["customer", "empPerInfo", "employee", "information", "t", "t1"]
hbase(main):004:0>
```

6. Check the schema of the ‘empPerInfo’ table.

hbase>describe ‘empPerInfo’

```
hbase(main):004:0> describe 'empPerInfo'
Table empPerInfo is ENABLED
empPerInfo
COLUMN FAMILIES DESCRIPTION
{NAME => 'address', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW', REPLICATION_SCOPE => '0', VERSIONS => '1', COMPRESSION => 'NONE', MIN_VERSIONS => '0', TTL => 'FOREVER', KEEP_DELETED_CELLS => 'FALSE', BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}
1 row(s) in 0.1820 seconds
```

hbase(main):005:0> █

7. Insert records into the ‘empPerInfo’ table

hbase>put ‘empPerInfo’, ‘Amit’,‘address:Building’, ‘D1045’

```
hbase(main):005:0> put 'empPerInfo', 'Amit', 'address:Building', 'D1045'
0 row(s) in 0.1090 seconds
```

hbase(main):006:0> █

hbase>put ‘empPerInfo’, ‘Amit’,‘address:street’,‘Karve Road’

```
hbase(main):006:0> put 'empPerInfo', 'Amit', 'address:street', 'Karve Road'
0 row(s) in 0.0150 seconds
```

hbase(main):007:0> █

hbase>put ‘empPerInfo’, ‘Amit’,‘address:city’,‘Pune’

```
hbase(main):007:0> put 'empPerInfo', 'Amit', 'address:city', 'Pune'
0 row(s) in 0.0140 seconds
```

hbase(main):008:0> █

8. Design the schema for storing the employees professional information like education, designation and package amount. (TableName: ‘empProfInfo’)

```
hbase(main):008:0> create 'empProfInfo', 'education', 'designation', 'package'
0 row(s) in 1.2750 seconds
```

=> Hbase::Table - empProfInfo

hbase(main):009:0> █

9. Insert the records in the ‘empProfInfo’ table.

```
=> Hbase::Table - empProfInfo
hbase(main):009:0> put 'empProfInfo','Amit','education','BSC'
0 row(s) in 0.0200 seconds

hbase(main):010:0> put 'empProfInfo','Amit','designation','Jr.developer'
0 row(s) in 0.0110 seconds

hbase(main):011:0> put 'empProfInfo','Amit','package','4L'
0 row(s) in 0.0090 seconds

hbase(main):012:0> put 'empProfInfo','Ridhi','education','BTech'
0 row(s) in 0.0080 seconds

hbase(main):013:0> put 'empProfInfo','Ridhi','designation','Sr.developer'
0 row(s) in 0.0100 seconds

hbase(main):014:0> put 'empProfInfo','Ridhi','package','8L'
0 row(s) in 0.0100 seconds

hbase(main):015:0> ■
```

10. Display all records of ‘empPerInfo’ and ‘empProfInfo’ tables.

hbase>scan ‘empPerInfo’

```
hbase(main):015:0> scan 'empPerInfo'
ROW           COLUMN+CELL
Amit          column=address:Building, timestamp=1704528611292, value=D1
              045
Amit          column=address:city, timestamp=1704528794293, value=Pune
Amit          column=address:street, timestamp=1704528747656, value=Karv
              e Road
1 row(s) in 0.0800 seconds

hbase(main):016:0> ■
```

hbase>scan ‘empProfInfo’

```
hbase(main):016:0> scan 'empProfInfo'
ROW           COLUMN+CELL
Amit          column=designation:, timestamp=1704529149605, value=Jr.dev
              eloper
Amit          column=education:, timestamp=1704529090890, value=BSC
Amit          column=package:, timestamp=1704529208629, value=4L
Ridhi         column=designation:, timestamp=1704529291120, value=Sr.dev
              eloper
Ridhi         column=education:, timestamp=1704529251820, value=BTech
Ridhi         column=package:, timestamp=1704529314111, value=8L
2 row(s) in 0.0420 seconds

hbase(main):017:0> ■
```

11. Display all the records of the employee ‘Amit’

```
hbase>get 'empPerInfo', 'Amit'
```

```
hbase(main):017:0> get 'empPerInfo', 'Amit'  
COLUMN          CELL  
address:Building    timestamp=1704528611292, value=D1045  
address:city       timestamp=1704528794293, value=Pune  
address:street      timestamp=1704528747656, value=Karve Road  
3 row(s) in 0.0200 seconds
```

```
hbase(main):018:0> █
```

12. Display address details of the employee ‘Amit’

```
hbase>get 'empPerInfo', 'Amit', 'address'
```

```
hbase(main):019:0> get 'empPerInfo', 'Amit', 'address'  
COLUMN          CELL  
address:Building    timestamp=1704528611292, value=D1045  
address:city       timestamp=1704528794293, value=Pune  
address:street      timestamp=1704528747656, value=Karve Road  
3 row(s) in 0.0340 seconds
```

```
hbase(main):020:0> █
```

13. Display building details of the employee ‘Amit’

```
hbase>get 'empPerInfo', 'Amit', 'address:Building'
```

```
hbase(main):020:0> get 'empPerInfo', 'Amit', 'address:Building'  
COLUMN          CELL  
address:Building    timestamp=1704528611292, value=D1045  
1 row(s) in 0.0110 seconds
```

```
hbase(main):021:0> █
```

14. Delete building details of the employee ‘Amit’

```
hbase>delete 'empPerInfo', 'Amit', 'address:Building'
```

```
hbase(main):021:0> delete 'empPerInfo', 'Amit', 'address:building'  
0 row(s) in 0.0350 seconds
```

```
hbase(main):022:0> █
```

hbase>get 'empPerInfo', 'Amit', 'address'

```
hbase(main):023:0> get 'empPerInfo','Amit','address'
COLUMN          CELL
address:Building timestamp=1704528611292, value=D1045
address:city      timestamp=1704528794293, value=Pune
address:street    timestamp=1704528747656, value=Karve Road
3 row(s) in 0.0150 seconds
```

hbase(main):024:0> █

15. Delete address details of the employee 'Amit'

hbase>deleteall 'empPerInfo', 'Amit'

```
hbase(main):024:0> deleteall 'empPerInfo','Amit'
0 row(s) in 0.0080 seconds
```

hbase(main):025:0> █

hbase>get 'empPerInfo', 'Amit'

```
hbase(main):025:0> get 'empPerInfo','Amit'
COLUMN          CELL
0 row(s) in 0.0090 seconds
```

hbase(main):026:0> █

16. Display the total number of records of "empProfInfo" table

hbase>count 'empProfInfo'

```
hbase(main):026:0> count 'empProfInfo'
2 row(s) in 0.0270 seconds
```

=> 2

hbase(main):027:0> █

17. Delete the education column family of the 'empProfInfo' table using the alter command.

hbase>alter 'empProfInfo', 'delete' =>'education'

```
hbase(main):027:0> alter 'empProfInfo','delete'=>'education'
Updating all regions with the new schema...
1/1 regions updated.
Done.
0 row(s) in 2.2550 seconds
```

```
hbase(main):028:0>
```

18. Disable ‘empProfInfo’ table.

```
hbase>disable 'empProfInfo'
```

```
hbase(main):028:0> disable 'empProfInfo'
0 row(s) in 2.3150 seconds
```

```
hbase(main):029:0>
```

19. Enable ‘empProfInfo’

```
hbase>enable 'empProfInfo'
```

```
hbase(main):029:0> enable 'empProfInfo'
0 row(s) in 1.3280 seconds
```

```
hbase(main):030:0>
```

20. Make the ‘empProfInfo’ table read only.

```
hbase>alter 'empProfInfo' 'READONLY'
```

```
hbase(main):030:0> alter 'empProfInfo' 'READONLY'
```

```
ERROR: Can't find a table: empProfInfoREADONLY
```

Alter a table. If the "hbase.online.schema.update.enable" property is set to false, then the table must be disabled (see help 'disable'). If the "hbase.online.schema.update.enable" property is set to true, tables can be altered without disabling them first. Altering enabled tables has caused problems in the past, so use caution and test it before using in production.

You can use the alter command to add, modify or delete column families or change table configuration options. Column families work in a similar way as the 'create' command. The column family specification can either be a name string, or a dictionary with the NAME attribute. Dictionaries are described in the output of the 'help' command, with no argument s.

For example, to change or add the 'f1' column family in table 't1' from current value to keep a maximum of 5 cell VERSIONS, do:

```
hbase> alter 't1', NAME => 'f1', VERSIONS => 5
```

You can operate on several column families:

```
hbase> alter 't1', 'f1', {NAME => 'f2', IN_MEMORY => true}, {NAME => 'f3', VERSIONS => 5}
```

To delete the 'f1' column family in table 'ns1:t1', use one of:

21. Drop table 'empProfInfo'

```
hbase>drop 'empProfInfo'
```

```
hbase(main):031:0> drop 'empProfInfo'
```

```
ERROR: Table empProfInfo is enabled. Disable it first.
```

```
Drop the named table. Table must first be disabled:
```

```
hbase> drop 't1'  
hbase> drop 'ns1:t1'
```

```
hbase(main):032:0> █
```

22. Create the second version of 'empPerInfo' table as 'empPerInfoVer2'

```
hbase>alter 'empPerInfo', NAME='empPerInfoVer2', VERSION=>2
```

```
hbase(main):032:0> alter 'empPerInfo',NAME='empPerInfoVer2',VERSION=>2  
Updating all regions with the new schema...  
1/1 regions updated.  
Done.  
Unknown argument ignored: 1.8.7  
Updating all regions with the new schema...  
1/1 regions updated.  
Done.  
0 row(s) in 4.1980 seconds
```

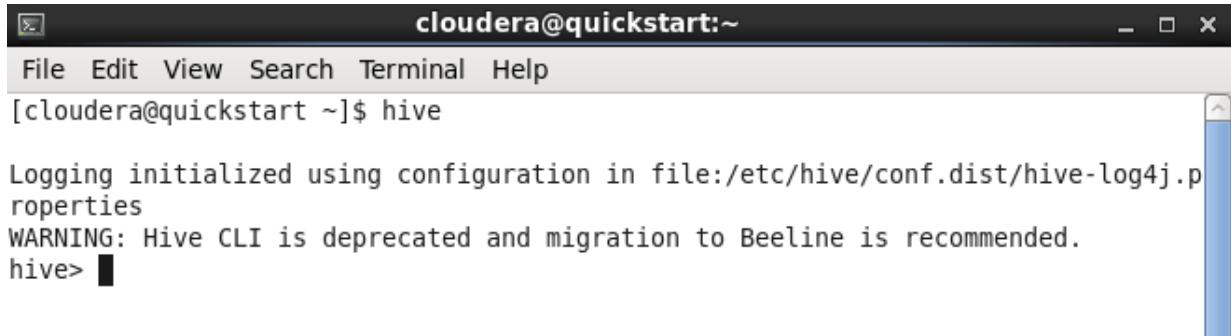
```
hbase(main):033:0> █
```

hbase>scan 'empPerInfoVer2'

```
hbase(main):033:0> scan 'empPerInfoVer2'  
ROW          COLUMN+CELL  
  
ERROR: Unknown table empPerInfoVer2!  
  
Scan a table; pass table name and optionally a dictionary of scanner  
specifications. Scanner specifications may include one or more of:  
TIMERANGE, FILTER, LIMIT, STARTROW, STOPROW, ROWPREFIXFILTER, TIMESTAMP,  
MAXLENGTH or COLUMNS, CACHE or RAW, VERSIONS, ALL_METRICS or METRICS  
  
If no columns are specified, all columns will be scanned.  
To scan all members of a column family, leave the qualifier empty as in  
'col_family'.  
  
The filter can be specified in two ways:  
1. Using a filterString - more information on this is available in the  
Filter Language document attached to the HBASE-4176 JIRA  
2. Using the entire package name of the filter.  
  
If you wish to see metrics regarding the execution of the scan, the  
ALL_METRICS boolean should be set to true. Alternatively, if you would  
prefer to see only a subset of the metrics, the METRICS array can be  
defined to include the names of only the metrics you care about.  
  
Some examples:  
  
hbase> scan 'hbase:meta'  
hbase> scan 'hbase:meta', {COLUMNS => 'info:regioninfo'}  
hbase> scan 'ns1:t1', {COLUMNS => ['c1', 'c2'], LIMIT => 10, STARTROW => 'xyz'  
}  
hbase> scan 't1', {COLUMNS => ['c1', 'c2'], LIMIT => 10, STARTROW => 'xyz'
```

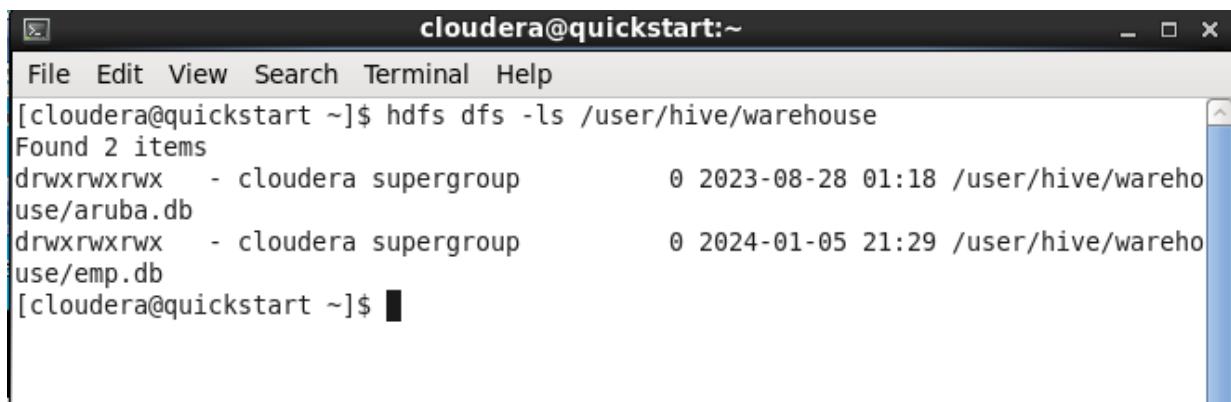
Practical 3: Using Hive

Q.1. Start with Hive



```
cloudera@quickstart:~$ hive
Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.properties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
hive> 
```

Q.2. List the existing Hive database



```
cloudera@quickstart:~$ hdfs dfs -ls /user/hive/warehouse
Found 2 items
drwxrwxrwx - cloudera supergroup          0 2023-08-28 01:18 /user/hive/warehouse/aruba.db
drwxrwxrwx - cloudera supergroup          0 2024-01-05 21:29 /user/hive/warehouse/emp.db
cloudera@quickstart:~$ 
```

Q.3. Create a database named “ABCCompany”. Before creating the database check the existence of the database.

```
hive> CREATE DATABASE IF NOT EXISTS ABCCompany;
OK
Time taken: 2.793 seconds
hive> 
```

Q.4. Create a table name ‘Employee’ in the above database with the following attributes.

id int, deptid int, name string, age int, and gender string

```
hive> CREATE TABLE IF NOT EXISTS ABCCompany.Employee (
    > id int,
    > deptid int,
    > name string,
    > age int,
    > gender string )
    > COMMENT 'Employee Table'
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ',';
OK
Time taken: 0.541 seconds
hive> 
```

Q.5. Create data.csv or data.txt file for ‘Employee’ data and store data for at least 5 employees from each department.

```
[cloudera@quickstart ~]$ gedit employee.txt
[cloudera@quickstart ~]$
```

```
cloudera-quickstart-vm-5.12.0-0-virtualbox [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System
File Edit View Search Tools Documents Help
Open Save Undo | | | | | | | |
employee.txt X
1,101,'Amit',20,'M'
2,102,'Sumit',21,'M'
3,103,'Priya',22,'F'
4,104,'Sanika',23,'F'
5,105,'Scott',24,'M'
```

Q.6. Load the data of the data file into the ‘Employee’ table.

```
[cloudera@quickstart ~]$ hdfs dfs -copyFromLocal /home/cloudera/employee.txt
[cloudera@quickstart ~]$ hdfs dfs -ls
Found 4 items
-rw-r--r-- 1 cloudera cloudera      105 2024-01-08 21:16 employee.txt
drwxr-xr-x - cloudera cloudera          0 2023-09-25 01:42 pig
drwxr-xr-x - cloudera cloudera          0 2023-09-25 22:13 train
drwxr-xr-x - cloudera cloudera          0 2023-08-28 00:32 zipcodes
[cloudera@quickstart ~]$
```

```
hive> LOAD DATA INPATH 'employee.txt' INTO TABLE ABCCompany.Employee;
Loading data to table abccompany.employee
Table abccompany.employee stats: [numFiles=1, totalSize=105]
OK
Time taken: 0.745 seconds
hive>
```

Q.7. Create the ‘Department’ table with the following fields.

deptid int, deptname string

```
hive> CREATE TABLE IF NOT EXISTS ABCCompany.Department (
    > deptid int,
    > deptname string )
    > COMMENT 'Department Table'
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ',';
OK
Time taken: 0.084 seconds
hive>
```

Q.8. Insert the data of the deptdata file into the ‘department’ table using INSERT command.

```
INSERT INTO emp.employee values(101,'Account');
```

```
INSERT INTO emp.employee values(102,IT);
```

```
hive> INSERT INTO ABCCompany.Department values(101,'Account');
Query ID = cloudera_20240108212525_aad6257e-0149-4e42-b9b7-5e341288b697
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1704774615894_0001, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1704774615894_0001/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1704774615894_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2024-01-08 21:25:28,015 Stage-1 map = 0%,  reduce = 0%
2024-01-08 21:25:40,893 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.35 sec
MapReduce Total cumulative CPU time: 2 seconds 350 msec
Ended Job = job_1704774615894_0001
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to: hdfs://quickstart.cloudera:8020/user/hive/warehouse/abccompany.db/department/.hive-staging_hive_2024-01-08_21-25-04_825_535281849686411828-1/-ext-10000
Loading data to table abccompany.department
Table abccompany.department stats: [numFiles=1, numRows=1, totalSize=12, rawDataSize=11]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Cumulative CPU: 2.35 sec  HDFS Read: 3584 HDFS Write: 8
9 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 350 msec
OK
Time taken: 37.583 seconds
hive> ■
```

```
hive> INSERT INTO ABCCompany.Department values(102,'IT');
Query ID = cloudera_20240108212828_ce8976a6-60ce-44a7-9fb8-aede74df75f3
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1704774615894_0002, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1704774615894_0002/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1704774615894_0002
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2024-01-08 21:29:09,143 Stage-1 map = 0%,  reduce = 0%
2024-01-08 21:29:22,151 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.4 sec
MapReduce Total cumulative CPU time: 2 seconds 400 msec
Ended Job = job_1704774615894_0002
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to: hdfs://quickstart.cloudera:8020/user/hive/warehouse/abccompany.db/department/.hive-staging_hive_2024-01-08_21-28-49_206_410121194036946658-1/-ext-10000
Loading data to table abccompany.department
Table abccompany.department stats: [numFiles=2, numRows=2, totalSize=19, rawDataSize=17]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Cumulative CPU: 2.4 sec  HDFS Read: 3663 HDFS Write: 84
SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 400 msec
OK
Time taken: 34.89 seconds
hive> ■
```

Q.9. List the records of both ‘Employee’ and ‘Department’ table.

```
hive> SELECT * FROM ABCCompany.Employee;
OK
1      101      'Amit'   20      'M'
2      102      'Sumit'   21      'M'
3      103      'Priya'   22      'F'
4      104      'Sanika'  23      'F'
5      105      'Scott'   24      'M'
Time taken: 1.715 seconds, Fetched: 5 row(s)
hive> SELECT * FROM ABCCompany.Department;
OK
101    Account
102    IT
Time taken: 0.105 seconds, Fetched: 2 row(s)
hive> █
```

Q.10. Display the schema of each table.

```
hive> DESCRIBE ABCCompany.Employee;
OK
id                  int
deptid              int
name                string
age                 int
gender              string
Time taken: 0.183 seconds, Fetched: 5 row(s)
hive> DESCRIBE ABCCompany.Department;
OK
deptid              int
deptname            string
Time taken: 0.085 seconds, Fetched: 2 row(s)
hive> █
```

Q. 11. Create the external table named ‘orders’ with the following fields in the prodorders folder in hdfs.

orderid, customerid, orderdate, productid, quantity

```
hive> CREATE EXTERNAL TABLE ABCCompany.Orders
    > orderid int,
    > customerid int,
    > orderdate string,
    > productid int,
    > quantity string )
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > LOCATION '/user/hive/prodorders/Orders';
OK
Time taken: 0.121 seconds
hive>
```

Q.12. Create the external table named ‘customers’ with the following fields in the prodorders folder in hdfs.Custid, custname

```
hive> CREATE EXTERNAL TABLE ABCCompany.Customers (
  > custid int,
  > custname string )
  > ROW FORMAT DELIMITED
  > FIELDS TERMINATED BY ','
  > LOCATION '/user/hive/prodorders/Customers';
OK
Time taken: 0.125 seconds
hive> █
```

Q. 13. Create the temporary table named ‘product’ with the following fields in the prodorders folder in hdfs.Productid, productname, price

```
hive> CREATE TEMPORARY TABLE ABCCompany.Product (
  > Productid int,
  > Productname string,
  > Price int )
  > ROW FORMAT DELIMITED
  > FIELDS TERMINATED BY ','
  > LOCATION '/user/hive/prodorders/Product';
OK
Time taken: 0.051 seconds
hive> █
```

Q. 14. Create the transaction table named ‘Transact’ with the following fields in the Prodorders folder in hdfs.

transid, transdate,custid,orderid,prodid, quantity,totalamount

```
hive> CREATE EXTERNAL TABLE IF NOT EXISTS Transact (
  > transid INT,transdate STRING,custid INT,orderid INT,prodid INT,quantity INT,totalamount DOUBLE)
  > ROW FORMAT DELIMITED
  > FIELDS TERMINATED BY ','
  > LOCATION '/home/cloudera/Desktop/prodorders';
OK
Time taken: 0.06 seconds
hive> █
```

Q. 15. Load or insert at least two related records in above tables and display the list of all records.

Inserting 2 records into Orders:

```
hive> INSERT INTO Orders VALUES(1,101,'1-1-2024',201,3);
```

```

hive> INSERT INTO Orders VALUES(1,101,'1-1-2024',201,3);
Query ID = cloudera_20240226012222_e1197fdc-6788-40c7-acae-e96cb01f8b66
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1708926577653_0004, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1708926577653_0004/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1708926577653_0004
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2024-02-26 01:23:04,706 Stage-1 map = 0%, reduce = 0%
2024-02-26 01:23:17,092 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 2.42 sec
MapReduce Total cumulative CPU time: 2 seconds 420 msec
Ended Job = job_1708926577653_0004
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to: hdfs://quickstart.cloudera:8020/home/cloudera/Desktop/prodorders/.hive-staging_hive_2024-02-26_01-22-51_484_1791816063088516483-1-ext-10000
Loading data to table default.orders
Table default.orders stats: [numFiles=1, numRows=1, totalSize=21, rawDataSize=20]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Cumulative CPU: 2.42 sec HDFS Read: 4534 HDFS Write: 91 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 420 msec
OK
Time taken: 26.966 seconds
hive> ■

```

```

hive> INSERT INTO Orders VALUES(2,102,'2-1-2024',202,5);
hive> INSERT INTO Orders VALUES(2,102,'2-1-2024',202,5);
Query ID = cloudera_20240226012525_f0d1729f-bf46-4016-bc58-22bfbb65b787
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1708926577653_0005, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1708926577653_0005/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1708926577653_0005
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2024-02-26 01:25:30,686 Stage-1 map = 0%, reduce = 0%
2024-02-26 01:25:42,829 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 2.36 sec
MapReduce Total cumulative CPU time: 2 seconds 360 msec
Ended Job = job_1708926577653_0005
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to: hdfs://quickstart.cloudera:8020/home/cloudera/Desktop/prodorders/.hive-staging_hive_2024-02-26_01-25-17_277_1825841825231098788-1-ext-10000
Loading data to table default.orders
Table default.orders stats: [numFiles=2, numRows=2, totalSize=42, rawDataSize=40]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Cumulative CPU: 2.36 sec HDFS Read: 4618 HDFS Write: 91 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 360 msec
OK
Time taken: 26.973 seconds
hive> ■

```

Displaying Data

```

hive> SELECT * FROM Orders;
OK
1      101      1-1-2024      201      3
2      102      2-1-2024      202      5
Time taken: 0.104 seconds, Fetched: 2 row(s)
hive> ■

```

Inserting 2 records into Customers:

```

hive> INSERT INTO Customers VALUES(101,'Jack');

```

```

hive> INSERT INTO Customers VALUES(101,'Jack');
Query ID = cloudera_20240226012929_fb56267f-46a4-4611-bfcd-d30592306e36
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1708926577653_0006, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1708926577653_0006/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1708926577653_0006
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2024-02-26 01:29:52,876 Stage-1 map = 0%, reduce = 0%
2024-02-26 01:30:05,177 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 2.33 sec
MapReduce Total cumulative CPU time: 2 seconds 330 msec
Ended Job = job_1708926577653_0006
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to: hdfs://quickstart.cloudera:8020/home/cloudera/Desktop/prodorders/.hive-staging_hive_2024-02-26_01-29-39_711_3791738435119846815-1/-ext-10000
Loading data to table default.customers
Table default.customers stats: [numFiles=3, numRows=1, totalSize=51, rawDataSize=8]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Cumulative CPU: 2.33 sec HDFS Read: 4002 HDFS Write: 82 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 330 msec
OK
Time taken: 26.935 seconds
hive> █

```

```

-----+
hive> INSERT INTO Customers VALUES(102,'Jill');
hive> INSERT INTO Customers VALUES(102,'Jill');
Query ID = cloudera_20240226013131_0139ed47-eb64-40c4-98b3-3d2f8812a469
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1708926577653_0007, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1708926577653_0007/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1708926577653_0007
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2024-02-26 01:31:24,352 Stage-1 map = 0%, reduce = 0%
2024-02-26 01:31:36,884 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 2.39 sec
MapReduce Total cumulative CPU time: 2 seconds 390 msec
Ended Job = job_1708926577653_0007
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to: hdfs://quickstart.cloudera:8020/home/cloudera/Desktop/prodorders/.hive-staging_hive_2024-02-26_01-31-11_126_5116937329139596967-1/-ext-10000
Loading data to table default.customers
Table default.customers stats: [numFiles=4, numRows=2, totalSize=60, rawDataSize=16]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Cumulative CPU: 2.39 sec HDFS Read: 4005 HDFS Write: 82 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 390 msec
OK
Time taken: 28.393 seconds
hive> █

```

Displaying Data

```

hive> SELECT * FROM Customers;
OK
1      101
2      102
101    Jack
102    Jill
Time taken: 0.061 seconds, Fetched: 4 row(s)
hive> █

```

Inserting 2 records into Product:

```

hive> INSERT INTO Product VALUES(201,"PEN",5.00);

```

```

hive> INSERT INTO product VALUES(201,"PEN",5.00);
Query ID = cloudera_20240226013434_a9f53a67-795b-43f9-a7da-307366db2bba
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1708926577653_0008, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1708926577653_0008/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1708926577653_0008
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2024-02-26 01:35:14,014 Stage-1 map = 0%, reduce = 0%
2024-02-26 01:35:25,285 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 2.45 sec
MapReduce Total cumulative CPU time: 2 seconds 450 msec
Ended Job = job_1708926577653_0008
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to: hdfs://quickstart.cloudera:8020/home/cloudera/Desktop/prodorders/.hive-staging_hive_2024-02-26_01-34-59_823_6928804279956113658-1/-ext-10000
Loading data to table default.product
Table default.product stats: [numFiles=5, numRows=1, totalSize=72, rawDataSize=11]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Cumulative CPU: 2.45 sec HDFS Read: 4364 HDFS Write: 83 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 450 msec
OK
Time taken: 28.013 seconds
hive>

```

```

-----+
hive> INSERT INTO Product VALUES(202,"PENCIL",3.00);+
-----+
hive> INSERT INTO product VALUES(202,"PENCIL",3.00);
Query ID = cloudera_20240226013838_c1fdb19e-cd5d-4a56-b494-4ff7a9e4817a
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1708926577653_0009, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1708926577653_0009/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1708926577653_0009
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2024-02-26 01:38:17,697 Stage-1 map = 0%, reduce = 0%
2024-02-26 01:38:30,034 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 2.46 sec
MapReduce Total cumulative CPU time: 2 seconds 460 msec
Ended Job = job_1708926577653_0009
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to: hdfs://quickstart.cloudera:8020/home/cloudera/Desktop/prodorders/.hive-staging_hive_2024-02-26_01-38-04_497_766063974842650476-1/-ext-10000
Loading data to table default.product
Table default.product stats: [numFiles=6, numRows=2, totalSize=87, rawDataSize=25]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Cumulative CPU: 2.46 sec HDFS Read: 4367 HDFS Write: 86 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 460 msec
OK
Time taken: 26.917 seconds
hive>

```

Displaying Data:

```

hive> SELECT * FROM Product;
OK
1      101    NULL
2      102    NULL
101    Jack   NULL
102    Jill   NULL
201    PEN    5.0
202    PENCIL 3.0

```

Inserting 2 records into Transact:

```
hive> INSERT INTO TABLE Transact VALUES
    > (1, '1-1-2024', 101, 1, 201, 3, 15),
    > (2, '2-1-2024', 102, 2, 202, 5, 15);
hive> INSERT INTO TABLE Transact VALUES
    > (1, '1-1-2024', 101, 1, 201, 3, 15),
    > (2, '2-1-2024', 102, 2, 202, 5, 15);
Query ID = cloudera_20240226014545_3382c479-a2e4-445d-a62f-de83a9240c67
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1708926577653_0012, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1708926577653_0012/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1708926577653_0012
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2024-02-26 01:45:59,511 Stage-1 map = 0%, reduce = 0%
2024-02-26 01:46:11,966 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 2.49 sec
MapReduce Total cumulative CPU time: 2 seconds 490 msec
Ended Job = job_1708926577653_0012
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to: hdfs://quickstart.cloudera:8020/home/cloudera/Desktop/prodorders/.hive-staging_hive_2024-02-26_01-45-46_516_5258006532080569069-1/-ext-10000
Loading data to table default.transact
Table default.transact stats: [numFiles=9, numRows=2, totalSize=170, rawDataSize=54]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1   Cumulative CPU: 2.49 sec   HDFS Read: 5202 HDFS Write: 128 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 490 msec
OK
Time taken: 26.886 seconds
hive>
```

Displaying Data:

```
hive> SELECT * FROM Transact;
1      1-1-2024      101      1      201      3      15.0
2      2-1-2024      102      2      202      5      15.0
...
```

Q. 16. Create the ‘productDummy’ table using the ‘product’ table and the SELECT command.

```
hive> CREATE TABLE IF NOT EXISTS productDummy As
    > SELECT * FROM Product;
hive> CREATE TABLE IF NOT EXISTS productDummy As
    > SELECT * FROM Product;
Query ID = cloudera_20240226015050_59579e9c-9769-470d-9b6d-e077aac0bf3f
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1708926577653_0013, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1708926577653_0013/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1708926577653_0013
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2024-02-26 01:50:32,877 Stage-1 map = 0%, reduce = 0%
2024-02-26 01:50:45,106 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 1.92 sec
MapReduce Total cumulative CPU time: 1 seconds 920 msec
Ended Job = job_1708926577653_0013
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to: hdfs://quickstart.cloudera:8020/user/hive/warehouse/.hive-staging_hive_2024-02-26_01-50-20_054_5496292111288149089-1/-ext-10001
Moving data to: hdfs://quickstart.cloudera:8020/user/hive/warehouse/productdummy
Table default.productdummy stats: [numFiles=1, numRows=10, totalSize=130, rawDataSize=120]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1   Cumulative CPU: 1.92 sec   HDFS Read: 4604 HDFS Write: 207 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 920 msec
OK
Time taken: 26.361 seconds
hive>
```

Q. 17. Create the table named ‘empDup’ using the ‘LIKE’ clause.

```
hive> CREATE TABLE IF NOT EXISTS empDup LIKE Employee;
OK
Time taken: 0.162 seconds
hive>
```

Q. 18. Describe the table structure of internal/managed table, external table, temporary table, and transactional table and observe the output details.

Orders:

```
hive> DESCRIBE FORMATTED Orders;
OK
# col_name          data_type          comment
orderid            int
customerid        int
orderdate          string
productid          int
quantity           int

# Detailed Table Information
Database:          default
Owner:             cloudera
CreateTime:        Mon Feb 26 01:07:46 PST 2024
LastAccessTime:    UNKNOWN
Protect Mode:     None
Retention:         0
Location:          hdfs://quickstart.cloudera:8020/home/cloudera/Desktop/prodorders
Table Type:        EXTERNAL_TABLE
Table Parameters:
  COLUMN_STATS_ACCURATE  true
  EXTERNAL                TRUE
  numFiles                 2
  numRows                  2
  rawDataSize              40
  totalSize                 42
  transient_lastDdlTime   1708939544
```

Customers:

```
hive> DESCRIBE FORMATTED Customers;
OK
# col_name          data_type          comment
custid            int
custname          string

# Detailed Table Information
Database:          default
Owner:             cloudera
CreateTime:        Mon Feb 26 01:09:52 PST 2024
LastAccessTime:    UNKNOWN
Protect Mode:     None
Retention:         0
Location:          hdfs://quickstart.cloudera:8020/home/cloudera/Desktop/prodorders
Table Type:        EXTERNAL_TABLE
Table Parameters:
  COLUMN_STATS_ACCURATE  true
  EXTERNAL                TRUE
  numFiles                 4
  numRows                  2
  rawDataSize              16
  totalSize                 60
  transient_lastDdlTime   1708939899
```

Product:

```

hive> DESCRIBE FORMATTED Product;
OK
# col_name          data_type        comment
productid          int
productname        string
price              double

# Detailed Table Information
Database:          default
Owner:             cloudera
CreateTime:        Mon Feb 26 01:13:30 PST 2024
LastAccessTime:    UNKNOWN
Protect Mode:     None
Retention:         0
Location:          hdfs://quickstart.cloudera:8020/home/cloudera/Desktop/prodorders
Table Type:        EXTERNAL_TABLE
Table Parameters:
  COLUMN_STATS_ACCURATE  true
  EXTERNAL                TRUE
  numFiles                 8
  numRows                  4
  rawDataSize               50
  totalSize                  114
  transient_lastDdlTime   1708940550

```

Transact:

```

hive> DESCRIBE FORMATTED Transact;
OK
# col_name          data_type        comment
transid            int
transdate          string
custid             int
orderid            int
prodid             int
quantity           int
totalamount        double

# Detailed Table Information
Database:          default
Owner:             cloudera
CreateTime:        Mon Feb 26 01:16:51 PST 2024
LastAccessTime:    UNKNOWN
Protect Mode:     None
Retention:         0
Location:          hdfs://quickstart.cloudera:8020/home/cloudera/Desktop/prodorders
Table Type:        EXTERNAL_TABLE
Table Parameters:
  COLUMN_STATS_ACCURATE  true
  EXTERNAL                TRUE
  numFiles                 9
  numRows                  2
  rawDataSize               54
  totalSize                  170
  transient_lastDdlTime   1708940773

```

Q. 19. Drop table 'empDup' using drop command.

```
hive> drop table empDup;
OK
Time taken: 0.312 seconds
hive> █
```

Q. 20. Drop table ‘productDummy’ using drop command and PURGE clause.

```
hive> DROP TABLE IF EXISTS productDummy PURGE;
OK
Time taken: 0.091 seconds
hive> █
```

Q. 21. Alter table ‘orders’ to change its name to ‘ordersTable’

```
hive> alter table orders RENAME to ordersTable
hive> alter table orders RENAME to ordersTable;
OK
Time taken: 0.12 seconds
hive> █
```

```
hive> describe ordersTable;
OK
orderid          int
customerid       int
orderdate        string
productid        int
quantity         int
Time taken: 0.114 seconds, Fetched: 5 row(s)
hive> █
```

Q. 22. Add the address column to the ‘Employee’ table using the alter command.

```
hive> alter table Employee add columns(empAddress string);
hive> alter table Employee add columns(empAddress string);
OK
Time taken: 0.138 seconds
hive> █
hive> describe Employee;
OK
id              int
deptid          int
name            string
age             int
gender          string
empaddress      string
Time taken: 0.068 seconds, Fetched: 6 row(s)
hive> █
```

Q. 23. Create and Drop database named “RJC”.

```
hive> create database rjc;
OK
Time taken: 0.072 seconds
hive> drop database rjc;
OK
Time taken: 0.155 seconds
hive> show databases;
OK
abccompany
default
emp
xyzcompany
Time taken: 0.026 seconds, Fetched: 4 row(s)
hive> ■
```

Practical 4: Hive Partitioning, Bucketing, Joining and Sorting

I. PARTITIONED Tables

Q1. Create a partitioned table named ‘supplier’ with the following attributes and also partition the suppliers data based on the city.

Supplierid, suppliername, supplieraddress,city,state,country

```
CREATE TABLE supplier (supplierid int, suppliername string, supplieraddress string, city
string, state string, country string)
PARTITIONED BY (city string)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
```

```
cloudera@quickstart:~$ hive
Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.properties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
hive> CREATE TABLE supplier (supplierid int,suppliername string, supplieraddress string,city string,state string,country string)
> PARTITIONED BY (city string)
> ROW FORMAT DELIMITED
> FIELDS TERMINATED BY ',';
FAILED: SemanticException [Error 10035]: Column repeated in partitioning columns
hive> CREATE TABLE supplier (supplierid int,suppliername string, supplieraddress string,state string,country string)
> PARTITIONED BY (city string)
> ROW FORMAT DELIMITED
> FIELDS TERMINATED BY ',';
OK
Time taken: 33.056 seconds
hive> 
```

Q2. Load the data from supplierData.txt file into the supplies table. Also try insert command to insert data into the partitioned table.

```
cloudera@quickstart:~$ gedit supplierData.txt
```

```
*supplierData.txt (~) - gedit
File Edit View Search Tools Documents Help
Open Save Undo Redo Cut Copy Find Select All
*supplierData.txt
101,'Amit','Sion','Maharashtra','India','Amravati'
102,'Tannu','Ghatkopar','Maharashtra','India','Mumbai'
103,'Ekta','Kurla','Maharashtra','India','Mumbai'
104,'Manjali','Vikhroli','Maharashtra','India','Nagpur'
105,'Anjali','Dadar','Maharashtra','India','Nashik'
```

```

cloudera@quickstart:~ 
File Edit View Search Terminal Help
[cloudera@quickstart ~]$ gedit supplierData.txt
[cloudera@quickstart ~]$ hdfs dfs -copyFromLocal /home/cloudera/supplierData.txt

[cloudera@quickstart ~]$ 
[cloudera@quickstart ~]$ hdfs dfs -ls
Found 1 items
-rw-r--r-- 1 cloudera cloudera      258 2024-03-14 10:43 supplierData.txt
[cloudera@quickstart ~]$ 

hive> LOAD DATA INPATH 'supplierData.txt' INTO TABLE supplier PARTITION (city='Mumbai');
Loading data to table default.supplier partition (city=Mumbai)
chgrp: changing ownership of 'hdfs://quickstart.cloudera:8020/user/hive/warehouse/supplier/city=Mumbai/supplierData.txt': User does not belong to supergroup
Partition default.supplier{city=Mumbai} stats: [numFiles=1, numRows=0, totalSize=258, rawDataSize=0]
OK
Time taken: 173.461 seconds
hive> 
hive> INSERT INTO supplier PARTITION (city='Nagpur') values(106,'Sudha','Ghatkopar','Maharashtra','India');
Query ID = cloudera_20240314113838_4c884f77-1721-40e8-8085-a2adc454c573
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1710187546367_0001, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1710187546367_0001/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1710187546367_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2024-03-14 12:02:30,509 Stage-1 map = 0%, reduce = 0%
2024-03-14 12:03:31,619 Stage-1 map = 0%, reduce = 0%
2024-03-14 12:04:32,597 Stage-1 map = 0%, reduce = 0%
2024-03-14 12:05:33,754 Stage-1 map = 0%, reduce = 0%
2024-03-14 12:06:35,068 Stage-1 map = 0%, reduce = 0%
2024-03-14 12:08:19,785 Stage-1 map = 0%, reduce = 0%
2024-03-14 12:09:41,055 Stage-1 map = 0%, reduce = 0%
2024-03-14 12:10:44,926 Stage-1 map = 0%, reduce = 0%, Cumulative CPU 4.24 sec
2024-03-14 12:11:32,425 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 5.58 sec
2024-03-14 12:13:10,002 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 6.71 sec
MapReduce Total cumulative CPU time: 6 seconds 710 msec
Ended Job = job_1710187546367_0001
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to: hdfs://quickstart.cloudera:8020/user/hive/warehouse/supplier/city=Nagpur/.hive-staging_hive_2024-03-14_11-38-44_4
27_1721226349256848177-1/-ext-10000
Loading data to table default.supplier partition (city=Nagpur)
Partition default.supplier{city=Nagpur} stats: [numFiles=1, numRows=1, totalSize=38, rawDataSize=37]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1   Cumulative CPU: 6.71 sec   HDFS Read: 4564 HDFS Write: 122 SUCCESS
Total MapReduce CPU Time Spent: 6 seconds 710 msec
OK
Time taken: 2103.084 seconds

```

Q3. Display all partitions of the table.

```

hive> SHOW PARTITIONS supplier;
OK
city=Mumbai
city=Nagpur
Time taken: 1.549 seconds, Fetched: 2 row(s)
hive>

```

Q4. Display the data of the ‘supplier’ table.

```
hive> SELECT * FROM supplier;
OK
101   'Amit'  'Sionaharashtra'      'India' 'Amravati'      Mumbai
102   'Tannu'  'Ghatkopar'        'Maharashtra' 'India' Mumbai
103   'Ekta'   'Kurla'          'Maharashtra' 'India' Mumbai
104   'Manjali' 'Vikhroli'       'Maharashtra' 'India' Mumbai
105   'Anjali'  'Dadar'          'Maharashtra' 'India' Mumbai
106   Sudha    Ghatkopar       Maharashtra  India Nagpur
Time taken: 4.613 seconds, Fetched: 6 row(s)
hive> Display all 478 possibilities? (y or n)
hive> █
```

Q5. Display the schema of the Hive partitioned table and check for partitions information.

```
hive> DESCRIBE FORMATTED supplier;
OK
# col_name          data_type          comment
supplierid          int
suppliername         string
supplieraddress      string
state                string
country              string

# Partition Information
# col_name          data_type          comment
city                string

# Detailed Table Information
Database:           default
Owner:               cloudera
CreateTime:          Wed Mar 13 14:16:35 PDT 2024
LastAccessTime:      UNKNOWN
Protect Mode:        None
Retention:           0
Location:            hdfs://quickstart.cloudera:8020/user/hive/warehouse/supplier
Table Type:          MANAGED_TABLE
Table Parameters:
    numPartitions     2
    transient_lastDdlTime 1710364595

# Storage Information
SerDe Library:      org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
InputFormat:          org.apache.hadoop.mapred.TextInputFormat
OutputFormat:         org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
Compressed:          No
Num Buckets:         -1
Bucket Columns:       []
Sort Columns:         []
Storage Desc Params:
    field.delim      ,
    serialization.format  ,
Time taken: 2.044 seconds, Fetched: 37 row(s)
hive> █
```

Q6. Update the partition location of any one partition using the ALTER command.

```
hive> ALTER TABLE supplier PARTITION (city='Mumbai') SET LOCATION '/home/cloudera/supplierData2.txt';
OK
Time taken: 3.694 seconds
hive> █
```

Q7. Rename any one partition using the ALTER command.

```
hive> ALTER TABLE supplier PARTITION (city='Nagpur') RENAME TO PARTITION (city='Nashik');
OK
Time taken: 3.047 seconds
```

Q8. Display the ‘supplier’ data for the specific partition.

```
hive> SELECT * FROM supplier WHERE city='Mumbai';
OK
Time taken: 10.164 seconds
hive> █
```

Q9. Demonstrate the DROP command on the hive partition.

```
hive> ALTER TABLE supplier DROP PARTITION (city='Mumbai');
Dropped the partition city=Mumbai
OK
Time taken: 7.588 seconds
hive> █
```

Q10. Get the suppliers list in upper case format.

```
hive> SELECT UPPER(suppliername) AS suppliername_upper FROM supplier;
OK
SUDHA
Time taken: 0.522 seconds, Fetched: 1 row(s)
hive> █
```

II. Hive Bucketing

Q1. Create a partitioned table named ‘supplierBucket’ with the following attributes and also partition and bucket the suppliers data based on the city.

```
hive> CREATE TABLE supplierBucket (
    > supplier_id INT,
    > supplier_name STRING
    > )
    > PARTITIONED BY (city STRING)
    > CLUSTERED BY (supplier_id) INTO 4 BUCKETS;
OK
Time taken: 0.401 seconds
hive> _
```

```

hive> INSERT INTO supplierBucket PARTITION (city='MUmbai') VALUES
    > (1, 'Tannu'),
    > (2, 'Ekta');
Query ID = cloudera_20240314130707_3df85bff-1df0-4397-8154-fc02accfdc10
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1710187546367_0002, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1710187546367_0002/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1710187546367_0002
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2024-03-14 13:10:32,659 Stage-1 map = 0%, reduce = 0%
2024-03-14 13:11:33,499 Stage-1 map = 0%, reduce = 0%
2024-03-14 13:12:36,088 Stage-1 map = 0%, reduce = 0%, Cumulative CPU 1.8 sec
2024-03-14 13:12:43,221 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 4.36 sec
MapReduce Total cumulative CPU time: 4 seconds 600 msec
Ended Job = job_1710187546367_0002
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to: hdfs://quickstart.cloudera:8020/user/hive/warehouse/supplierbucket/city=MUmbai/.hive-staging_hive_2024-03-14_13-0
7-22 964 8696354398050949245-1/-ext-10000
Loading data to table default.supplierbucket partition (city=MUmbai)
Partition default.supplierbucket{city=MUmbai} stats: [numFiles=1, numRows=2, totalSize=15, rawDataSize=13]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Cumulative CPU: 4.6 sec HDFS Read: 4060 HDFS Write: 105 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 600 msec
OK
Time taken: 337.108 seconds
hive>
hive> INSERT INTO supplierBucket PARTITION (city='Delhi') VALUES
    > (3, 'Sudha'),
    > (4, 'Radhika');
Query ID = cloudera_20240314132121_2c280208-f61f-4b0e-8388-91b76878ee6e
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1710187546367_0003, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1710187546367_0003/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1710187546367_0003
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2024-03-14 13:21:58,534 Stage-1 map = 0%, reduce = 0%
2024-03-14 13:22:58,949 Stage-1 map = 0%, reduce = 0%
2024-03-14 13:23:06,387 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 3.98 sec
MapReduce Total cumulative CPU time: 3 seconds 980 msec
Ended Job = job_1710187546367_0003
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to: hdfs://quickstart.cloudera:8020/user/hive/warehouse/supplierbucket/city=Delhi/.hive-staging_hive_2024-03-14_13-21
-17 615 484502699107440214-1/-ext-10000
Loading data to table default.supplierbucket partition (city=Delhi)
Partition default.supplierbucket{city=Delhi} stats: [numFiles=1, numRows=2, totalSize=18, rawDataSize=16]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Cumulative CPU: 3.98 sec HDFS Read: 4055 HDFS Write: 107 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 980 msec
OK
Time taken: 120.261 seconds
hive>
hive> INSERT INTO supplierBucket PARTITION (city='Kolkata') VALUES
    > (5, 'Manjali'),
    > (6, 'Anjali');
Query ID = cloudera_20240314132525_26a5447d-9351-4618-8f12-52ea6bfa3b3e
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1710187546367_0004, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1710187546367_0004/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1710187546367_0004
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2024-03-14 13:26:28,957 Stage-1 map = 0%, reduce = 0%
2024-03-14 13:27:30,840 Stage-1 map = 0%, reduce = 0%
2024-03-14 13:27:44,623 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 3.21 sec
MapReduce Total cumulative CPU time: 3 seconds 210 msec
Ended Job = job_1710187546367_0004
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to: hdfs://quickstart.cloudera:8020/user/hive/warehouse/supplierbucket/city=Kolkata/.hive-staging_hive_2024-03-14_13-
25-28 371 6588925951170246562-1/-ext-10000
Loading data to table default.supplierbucket partition (city=Kolkata)
Partition default.supplierbucket{city=Kolkata} stats: [numFiles=1, numRows=2, totalSize=19, rawDataSize=17]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Cumulative CPU: 3.21 sec HDFS Read: 4069 HDFS Write: 110 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 210 msec
OK
Time taken: 144.588 seconds
hive>
hive> INSERT INTO supplierBucket PARTITION (city='Chennai') VALUES
    > (7, 'Ankit'),
    > (8, 'Amit');
Query ID = cloudera_20240314133030_b0f6b58f-e1f3-4adf-8403-b3f56f9ed96a
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1710187546367_0005, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1710187546367_0005/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1710187546367_0005
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2024-03-14 13:31:08,277 Stage-1 map = 0%, reduce = 0%
2024-03-14 13:31:55,424 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 2.79 sec
MapReduce Total cumulative CPU time: 3 seconds 610 msec
Ended Job = job_1710187546367_0005
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to: hdfs://quickstart.cloudera:8020/user/hive/warehouse/supplierbucket/city=Chennai/.hive-staging_hive_2024-03-14_13-
30-20 271 2988018460549787699-1/-ext-10000
Loading data to table default.supplierbucket partition (city=Chennai)
Partition default.supplierbucket{city=Chennai} stats: [numFiles=1, numRows=2, totalSize=15, rawDataSize=13]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Cumulative CPU: 3.61 sec HDFS Read: 4065 HDFS Write: 106 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 610 msec
OK
Time taken: 114.917 seconds
hive>

```

Q2. Display the data of ‘supplierBucket’ tables.

```
hive> SELECT * FROM supplierBucket;
OK
7      Ankit    Chennai
8      Amit     Chennai
3      Sudha    Delhi
4      Radhika   Delhi
5      Manjali   Kolkata
6      Anjali   Kolkata
1      Tannu    MUmbai
2      Ekta    MUmbai
Time taken: 0.373 seconds, Fetched: 8 row(s)
hive> █
```

Q3. Display the ‘supplier’ data for the specific bucket.

```
hive> SELECT * FROM supplierBucket TABLESAMPLE(BUCKET 2 OUT OF 4);
OK
5      Manjali   Kolkata
1      Tannu    MUmbai
Time taken: 0.607 seconds, Fetched: 2 row(s)
hive> █
```

Q4. Display total number of records of a specific bucket.

```
hive> SELECT COUNT(*) FROM supplierBucket WHERE city='MUmbai';
Query ID = cloudera_20240314133737_875c4d3c-f865-4cf8-8d04-26310f556c6c
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1710187546367_0007, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1710187546367_0007/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1710187546367_0007
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2024-03-14 13:38:11,943 Stage-1 map = 0%,  reduce = 0%
2024-03-14 13:39:28,670 Stage-1 map = 0%,  reduce = 0%
2024-03-14 13:39:36,681 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 4.46 sec
2024-03-14 13:40:12,289 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 6.38 sec
MapReduce Total cumulative CPU time: 6 seconds 380 msec
Ended Job = job_1710187546367_0007
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1  Cumulative CPU: 6.38 sec  HDFS Read: 8020 HDFS Write: 2 SUCCESS
Total MapReduce CPU Time Spent: 6 seconds 380 msec
OK
2
Time taken: 137.47 seconds, Fetched: 1 row(s)
hive> █
```

III. Hive Joining

Q1. create a database named “HiveJoins” and Create following external tables, data files and load data of data files into the tables.

Example:

```
create table customers(ID int, Name string, Age int, Address string, Salary float)
row format delimited
fields terminated by ','
tblproperties("skip.header.line.count" ="1");
```

```

hive> CREATE TABLE customers(
    > ID int,
    > Name string,
    > Age int,
    > Address string,
    > Salary float
    > )
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > tblproperties("skip.header.line.count"="1");
OK
Time taken: 10.415 seconds
hive>

```

Customer.csv

customer.csv (~) - gedit

```

File Edit View Search Tools Documents Help
Open Save Undo
customer.csv X
Id,Name,Age,Address,Salary
1,Tannu,32,New york,2000
2,Ekta,25,Florida,1500
3,Kim,23,Seattle,2000
4,Clay,25,Boston,6500
5,Henry,27,California,8500
6,Kit,22,Chicago,4500
7,Muffy,24,New York,10000

```

```

hive> LOAD DATA LOCAL INPATH '/home/cloudera/customer.csv' INTO TABLE customers;
Loading data to table default.customers
Table default.customers stats: [numFiles=1, totalSize=194]
OK
Time taken: 1.432 seconds
hive>

```

```

create table orders(oid int, odate date, cid int, amount float)
row format delimited
fields terminated by ','
tblproperties("skip.header.line.count"="1");
hive> CREATE TABLE orders(
    > oid int,
    > odate date,
    > cid int,
    > amount float
    > )
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > tblproperties("skip.header.line.count"="1");
OK
Time taken: 0.227 seconds
hive>

```

Orders.csv

```
cloudera@quickstart:~
```

```
[cloudera@quickstart ~]$ gedit customer.csv
[cloudera@quickstart ~]$ gedit orders.csv
```

orders.csv (~) - gedit

```
File Edit View Search Tools Documents Help
Open Save Undo
orders.csv X
```

```
Oid,Date,Customer_id,Amount
102,2018-08-08,3,3000
100,2018-08-03,3,1500
101,2018-11-02,2,1560
103,2018-11-08,4,2060
```

```
hive> LOAD DATA LOCAL INPATH '/home/cloudera/orders.csv' INTO TABLE orders;
Loading data to table default.orders
Table default.orders stats: [numFiles=1, totalSize=116]
OK
Time taken: 1.38 seconds
hive>
```

Q2. Write a query to retrieve customer id, name, age from customers table and amount from the orders table by using various join on id of the customers and orders table.

Example:

Normal Join

```
select c.id, c.name, c.age, o.amount
from customers c JOIN orders o
on (c.id = o.cid);
```

```
hive> SELECT c.ID, c.Name, c.Age, o.Amount
> FROM customers c
> JOIN orders o ON (c.ID = o.cid);
Query ID = cloudera_20240314141313_c62eadf5-d6e9-43ef-afb3-f52d9b0fdc62
Total jobs = 1
Execution log at: /tmp/cloudera/cloudera_20240314141313_c62eadf5-d6e9-43ef-afb3-f52d9b0fdc62.log
2024-03-14 02:13:27 Starting to launch local task to process map join; maximum memory = 932184064
2024-03-14 02:13:30 Dump the side-table for tag: 1 with group count: 3 into file: file:/tmp/cloudera/559da3f7-255e-4b95-95f0-d9931cc11991/hive_2024-03-14_14-13-19_575_9068084455476992788-1-local-10003/HashTable-Stage-3/MapJoin-mapfile01--.hashtable
2024-03-14 02:13:30 Uploaded 1 File to: file:/tmp/cloudera/559da3f7-255e-4b95-95f0-d9931cc11991/hive_2024-03-14_14-13-19_575_9068084455476992788-1-local-10003/HashTable-Stage-3/MapJoin-mapfile01--.hashtable (338 bytes)
2024-03-14 02:13:30 End of local task; Time Taken: 2.197 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1710187546367_0008, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1710187546367_0008/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1710187546367_0008
Hadoop job information for Stage-3: number of mappers: 1; number of reducers: 0
2024-03-14 14:13:48,706 Stage-3 map = 0%, reduce = 0%
2024-03-14 14:14:04,439 Stage-3 map = 100%, reduce = 0%, Cumulative CPU 1.92 sec
MapReduce Total cumulative CPU time: 1 seconds 920 msec
Ended Job = job_1710187546367_0008
MapReduce Jobs Launched:
Stage-Stage-3: Map: 1 Cumulative CPU: 1.92 sec HDFS Read: 6960 HDFS Write: 66 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 920 msec
OK
2      Ekta    25      1560.0
3      Kim     23      3000.0
3      Kim     23      1500.0
4      Clay    25      2060.0
Time taken: 61.126 seconds, Fetched: 4 row(s)
```

Left Outer Join

```
select c.id, c.name, o.amount, o.odate
from customers c LEFT OUTER JOIN orders o
on (c.id = o.cid);
hive> SELECT c.ID, c.Name, o.Amount, o.odate
> FROM customers c LEFT OUTER JOIN orders o
> on (c.ID = o.cid);
Query ID = cloudera_20240314141616_23589d5b-ecb9-4944-b274-23a78ea79f71
Total jobs = 1
Execution log at: /tmp/cloudera/cloudera_20240314141616_23589d5b-ecb9-4944-b274-23a78ea79f71.log
2024-03-14 02:16:36 Starting to launch local task to process map join; maximum memory = 932184064
2024-03-14 02:16:40 Dump the side-table for tag: 1 with group count: 3 into file: file:/tmp/cloudera/559da3f7-255e-4b95-95f0-d9931cc11991/hive_2024-03-14_14-16-30_008_7223135153109691991-1-local-10003/HashTable-Stage-3/MapJoin-mapfile11--.hashtable
2024-03-14 02:16:40 Uploaded 1 File to: file:/tmp/cloudera/559da3f7-255e-4b95-95f0-d9931cc11991/hive_2024-03-14_14-16-30_008_7223135153109691991-1-local-10003/HashTable-Stage-3/MapJoin-mapfile11--.hashtable (350 bytes)
2024-03-14 02:16:40 End of local task; Time Taken: 3.658 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1710187546367_0009, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1710187546367_0009/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1710187546367_0009
Hadoop job information for Stage-3: number of mappers: 1; number of reducers: 0
2024-03-14 14:16:52,776 Stage-3 map = 0%, reduce = 0%
2024-03-14 14:17:00,096 Stage-3 map = 100%, reduce = 0%, Cumulative CPU 1.11 sec
MapReduce Total cumulative CPU time: 1 seconds 110 msec
Ended Job = job_1710187546367_0009
MapReduce Jobs Launched:
Stage-Stage-3: Map: 1 Cumulative CPU: 1.11 sec HDFS Read: 7038 HDFS Write: 152 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 110 msec
OK
1      Tannu    NULL    NULL
2      Ekta     1560.0  2018-11-02
3      Kim      3000.0  2018-08-08
3      Kim      1500.0  2018-08-03
4      Clay     2060.0  2018-11-08
5      Henry    NULL    NULL
6      Kit      NULL    NULL
7      Muffy    NULL    NULL
Time taken: 32.293 seconds, Fetched: 8 row(s)
hive>
```

Right Outer Join

```
select c.id, c.name, o.amount, o.odate
from customers c RIGHT OUTER JOIN orders o
on (c.id = o.cid);
hive> SELECT c.ID, c.Name, o.Amount, o.odate
> FROM customers c
> RIGHT OUTER JOIN orders o ON (c.id = o.cid);
Query ID = cloudera_20240314141818_08154012-51f8-41dd-8931-e886eb3f1420
Total jobs = 1
Execution log at: /tmp/cloudera/cloudera_20240314141818_08154012-51f8-41dd-8931-e886eb3f1420.log
2024-03-14 02:18:37 Starting to launch local task to process map join; maximum memory = 932184064
2024-03-14 02:18:38 Dump the side-table for tag: 0 with group count: 7 into file: file:/tmp/cloudera/559da3f7-255e-4b95-95f0-d9931cc11991/hive_2024-03-14_14-18-34_034_9134433556291938991-1-local-10003/HashTable-Stage-3/MapJoin-mapfile20--.hashtable
2024-03-14 02:18:38 Uploaded 1 File to: file:/tmp/cloudera/559da3f7-255e-4b95-95f0-d9931cc11991/hive_2024-03-14_14-18-34_034_9134433556291938991-1-local-10003/HashTable-Stage-3/MapJoin-mapfile20--.hashtable (429 bytes)
2024-03-14 02:18:38 End of local task; Time Taken: 1.015 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1710187546367_0010, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1710187546367_0010/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1710187546367_0010
Hadoop job information for Stage-3: number of mappers: 1; number of reducers: 0
2024-03-14 14:18:47,887 Stage-3 map = 0%, reduce = 0%
2024-03-14 14:18:55,268 Stage-3 map = 100%, reduce = 0%, Cumulative CPU 1.4 sec
MapReduce Total cumulative CPU time: 1 seconds 400 msec
Ended Job = job_1710187546367_0010
MapReduce Jobs Launched:
Stage-Stage-3: Map: 1 Cumulative CPU: 1.4 sec HDFS Read: 6950 HDFS Write: 98 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 400 msec
OK
3      Kim      3000.0  2018-08-08
3      Kim      1500.0  2018-08-03
2      Ekta    1560.0  2018-11-02
4      Clay     2060.0  2018-11-08
Time taken: 23.38 seconds, Fetched: 4 row(s)
hive>
```

IV. Hive Sorting

Q1. Display the first four customers' salaries from the customers table in descending order.
select salary from customers sort by salary desc limit 4;

```
hive> SELECT salary FROM customers ORDER BY salary DESC LIMIT 4;
Query ID = cloudera_20240314142020_dd30a8db-7ffb-4955-87d8-d3099e9d6be2
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1710187546367_0011, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1710187546367_0011/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1710187546367_0011
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2024-03-14 14:20:56,834 Stage-1 map = 0%,  reduce = 0%
2024-03-14 14:21:11,742 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.62 sec
2024-03-14 14:21:41,052 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 4.18 sec
MapReduce Total cumulative CPU time: 4 seconds 180 msec
Ended Job = job_1710187546367_0011
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1  Cumulative CPU: 4.18 sec  HDFS Read: 7149 HDFS Write: 29 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 180 msec
OK
10000.0
8500.0
6500.0
4500.0
Time taken: 82.783 seconds, Fetched: 4 row(s)
hive> ■
```

Q2. Get the orders list in the ascending order of date.

Select * from orders Order By odate ASC;

```
hive> SELECT * FROM orders ORDER BY odate ASC;
Query ID = cloudera_20240314142222_3d2b37be-93bc-4eb9-8b75-d05eb274d72f
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1710187546367_0012, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1710187546367_0012/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1710187546367_0012
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2024-03-14 14:23:09,683 Stage-1 map = 0%,  reduce = 0%
2024-03-14 14:23:17,367 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.08 sec
2024-03-14 14:23:34,422 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 2.57 sec
MapReduce Total cumulative CPU time: 2 seconds 570 msec
Ended Job = job_1710187546367_0012
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1  Cumulative CPU: 2.57 sec  HDFS Read: 7917 HDFS Write: 96 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 570 msec
OK
100  2018-08-03    3      1500.0
102  2018-08-08    3      3000.0
101  2018-11-02    2      1560.0
103  2018-11-08    4      2060.0
Time taken: 43.226 seconds, Fetched: 4 row(s)
hive> ■
```

Q3. Get the list of customers in the ascending order of the customer's name.

```
hive> SELECT * FROM orders ORDER BY Name ASC;
FAILED: SemanticException [Error 10004]: Line 1:30 Invalid table alias or column reference 'Name': (possible column names are: oid, odate, cid, amount)
hive> SELECT * FROM customers ORDER BY Name ASC;
Query ID = cloudera_20240314142525_9795a545-5ad6-48c5-97b6-99af42118533
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1710187546367_0013, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1710187546367_0013/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1710187546367_0013
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2024-03-14 14:26:01,044 Stage-1 map = 0%, reduce = 0%
2024-03-14 14:26:15,378 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 3.41 sec
2024-03-14 14:26:48,443 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 5.08 sec
MapReduce Total cumulative CPU time: 5 seconds 80 msec
Ended Job = job_1710187546367_0013
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 5.08 sec HDFS Read: 8232 HDFS Write: 181 SUCCESS
Total MapReduce CPU Time Spent: 5 seconds 80 msec
OK
4      Clay    25      Boston   6500.0
2      Ekta    25      Florida  1500.0
5      Henry   27      California 8500.0
3      Kim     23      Seattle   2000.0
6      Kit     22      Chicago   4500.0
7      Muffy   24      New York  10000.0
1      Tannu   32      New york  2000.0
Time taken: 73.364 seconds, Fetched: 7 row(s)
hive>
```

Practical 5: Using Pig and PigLatin

What is Apache Pig

Apache Pig is a high-level data flow platform for executing MapReduce programs of Hadoop. The language used for Pig is Pig Latin.

The Pig scripts get internally converted to Map Reduce jobs and get executed on data stored in HDFS. Apart from that, Pig can also execute its job in Apache Tez or Apache Spark.

Pig can handle any type of data, i.e., structured, semi-structured or unstructured and stores the corresponding results into Hadoop Data File System. Every task which can be achieved using PIG can also be achieved using java used in MapReduce.

Features of Apache Pig

The various uses of Pig technology.

1) Ease of programming

Writing complex java programs for map reduce is quite tough for non-programmers. Pig makes this process easy. In the Pig, the queries are converted to MapReduce internally.

2) Optimization opportunities

It is how tasks are encoded permits the system to optimize their execution automatically, allowing the user to focus on semantics rather than efficiency.

3) Extensibility

A user-defined function is written in which the user can write their logic to execute over the data set.

4) Flexible

It can easily handle structured as well as unstructured data.

5) In-built operators

It contains various type of operators such as sort, filter and joins.

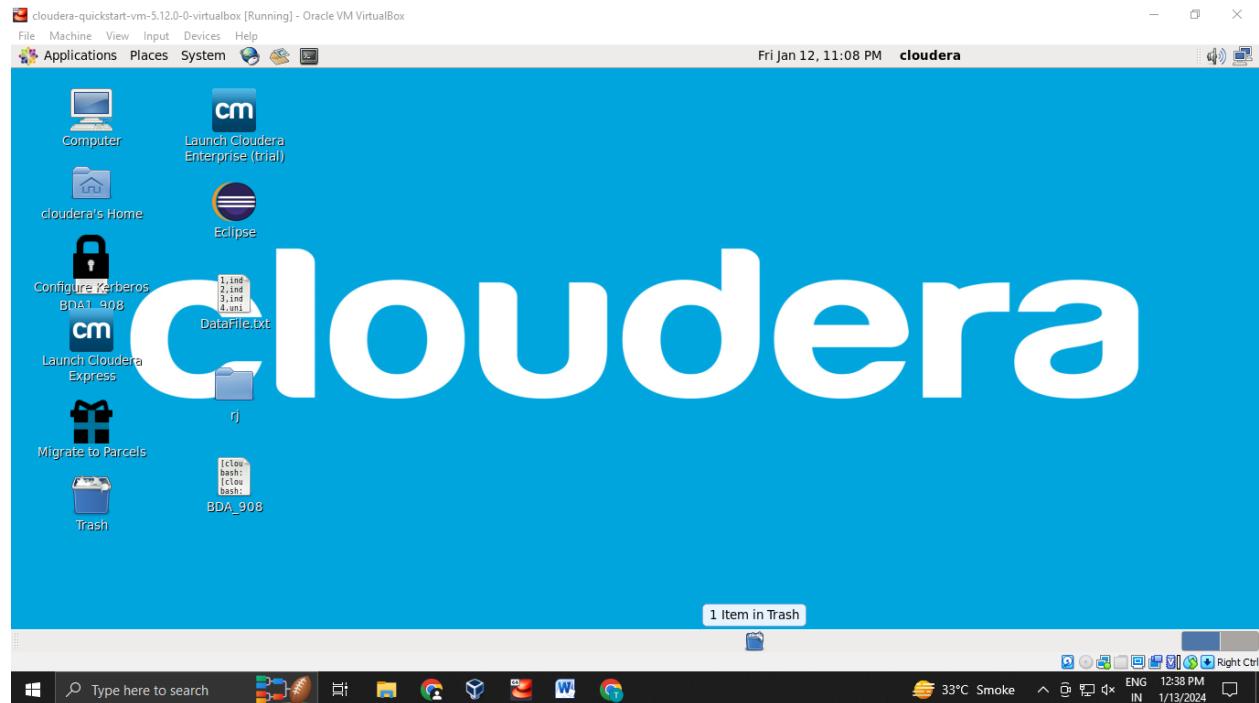
Advantages of Apache Pig

- Less code - The Pig consumes less line of code to perform any operation.
- Reusability - The Pig code is flexible enough to reuse again.
- Nested data types - The Pig provides a useful concept of nested data types like tuple, bag, and map.

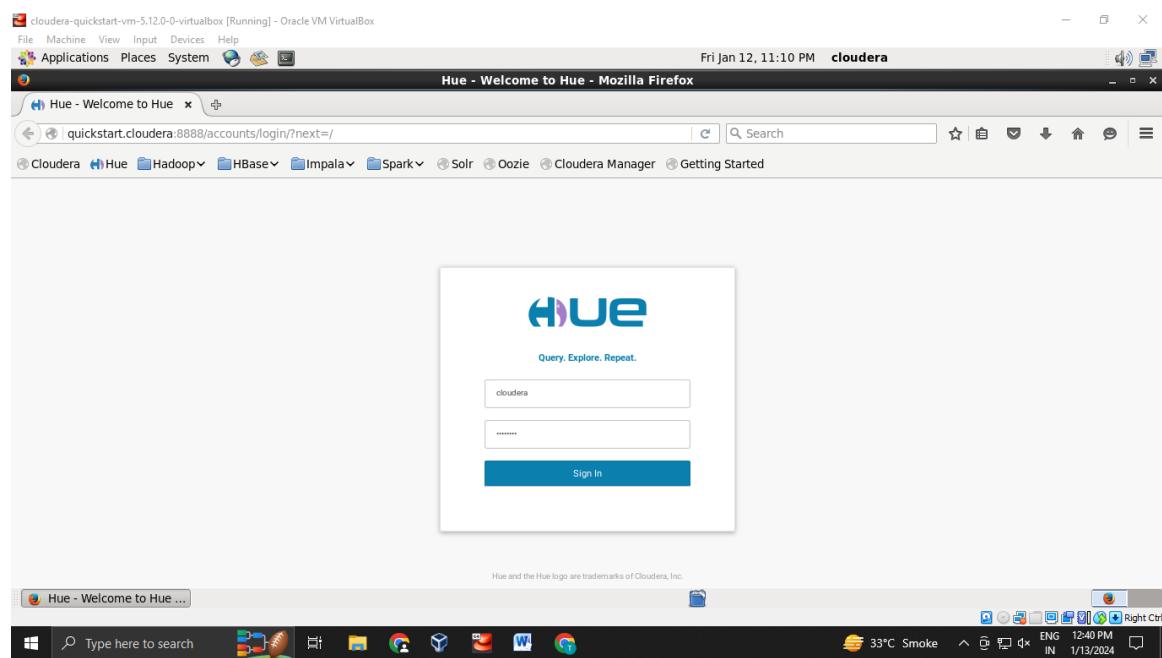
To implement Word Count problem using Pig

Steps:

- 1) Start the cloudera.



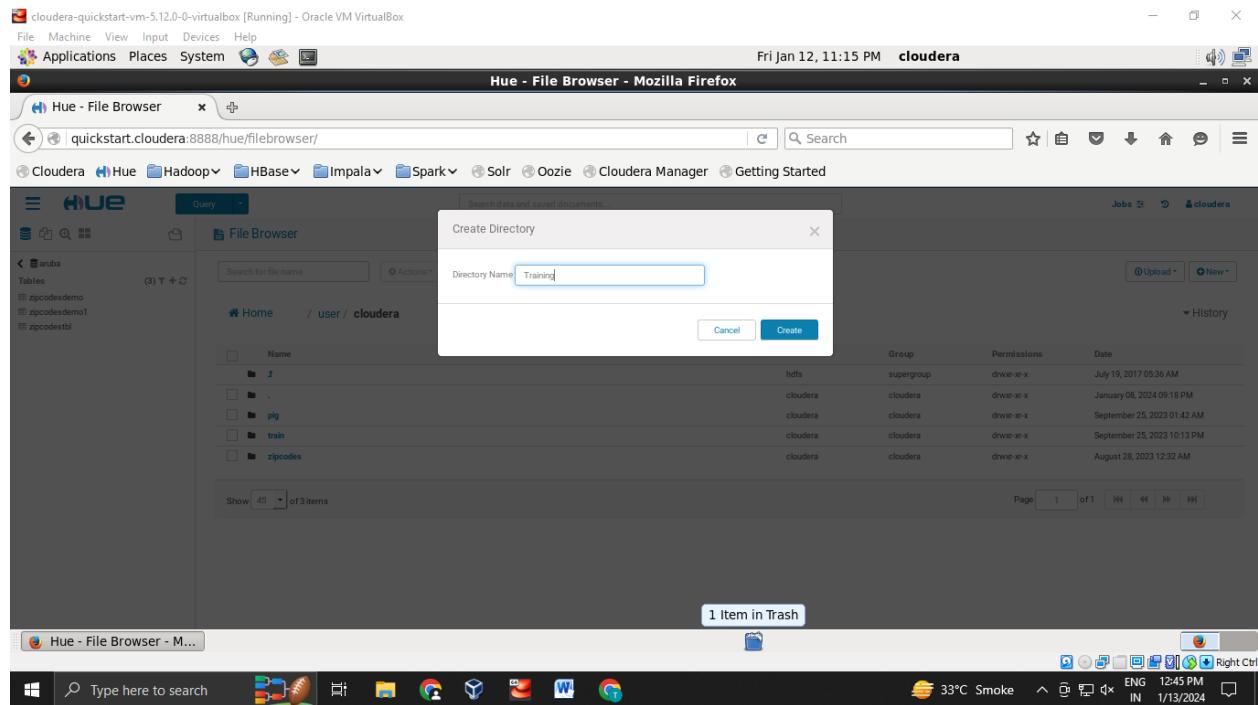
- 2) Open the browser. And then open Hue and login using default credentials.



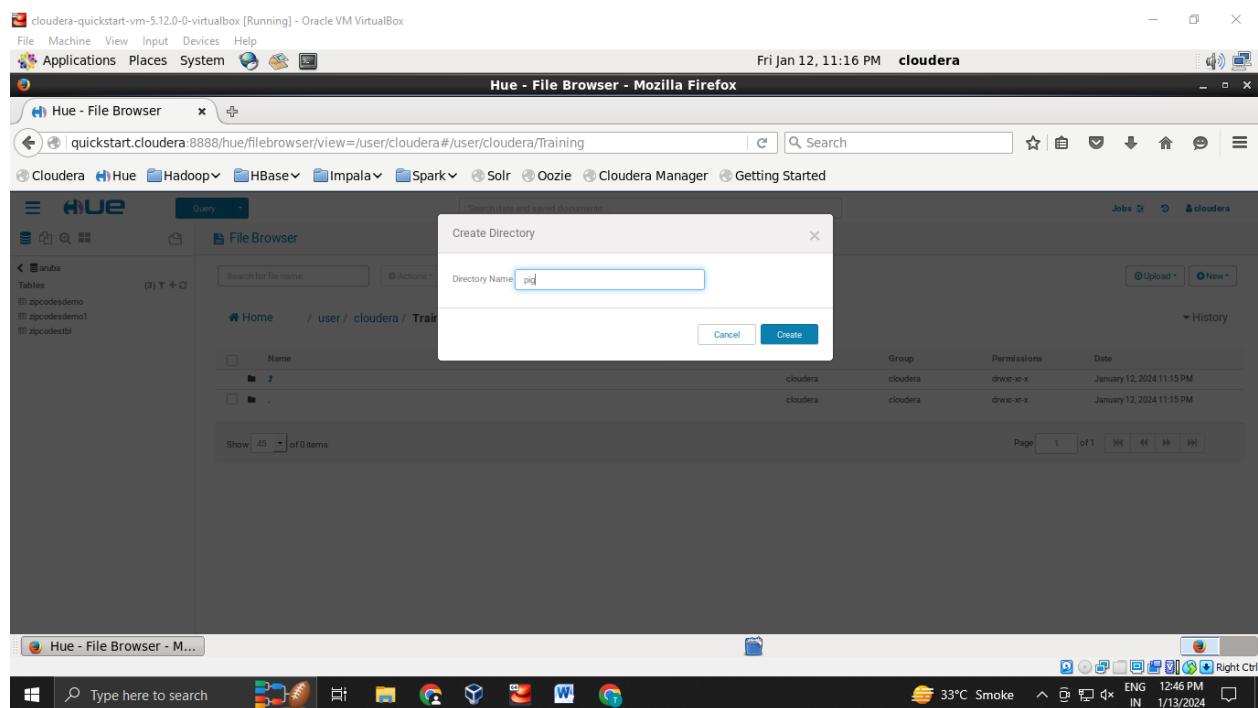
3) Now open the directory /user/cloudera → Click on menu option go to files.

Name	Size	User	Group	Permissions	Date
hdfs		hdfs	supergroup	drwxr-x-x	July 19, 2017 05:36 AM
.		cloudera	cloudera	drwxr-x-x	January 08, 2024 09:18 PM
pig		cloudera	cloudera	drwxr-x-x	September 25, 2023 01:42 AM
train		cloudera	cloudera	drwxr-x-x	September 25, 2023 10:13 PM
zipcodes		cloudera	cloudera	drwxr-x-x	August 28, 2023 12:32 AM

4) Now we are creating the directory as Training inside /user/cloudera → Click on New button in Right side → Click on directory.



5) After creating Training directory now creating the pig directory inside Training.



The screenshot shows the Cloudera Quickstart VM interface. At the top, the title bar reads "cloudera-quickstart-vm-5.12.0-0-virtualbox [Running] - Oracle VM VirtualBox". The main window is titled "Hue - File Browser - Mozilla Firefox". The address bar shows the URL "quickstart.cloudera:8888/hue/filebrowser/view=/user/cloudera#/user/cloudera/Training". The browser displays a file list for the "/user/cloudera/Training" directory, which contains a single item named "pig". The "pig" directory has the following details:

Name	User	Group	Permissions	Date
pig	cloudera	cloudera	drwxr-x-x	January 12, 2024 11:17 PM

The system tray at the bottom right shows the date as 1/13/2024 and the time as 12:47 PM.

6) pig directory has been created inside /user/cloudera/Training

The screenshot shows the Cloudera Quickstart VM interface. At the top, the title bar reads "cloudera-quickstart-vm-5.12.0-0-virtualbox [Running] - Oracle VM VirtualBox". The main window is titled "Hue - File Browser - Mozilla Firefox". The address bar shows the URL "quickstart.cloudera:8888/hue/filebrowser/view=/user/cloudera#/user/cloudera/Training/pig". The browser displays a file list for the "/user/cloudera/Training/pig" directory, which contains a single item named "j". The "j" file has the following details:

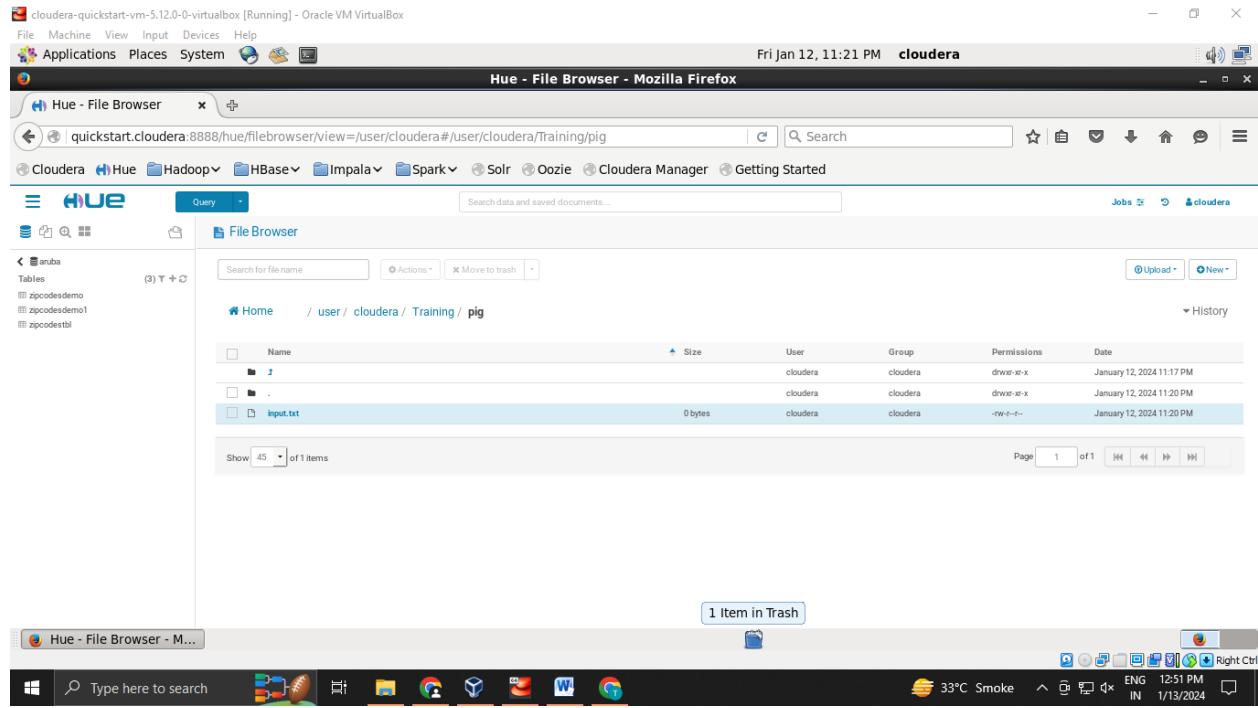
Name	User	Group	Permissions	Date
j	cloudera	cloudera	drwxr-x-x	January 12, 2024 11:17 PM

The system tray at the bottom right shows the date as 1/13/2024 and the time as 12:48 PM.

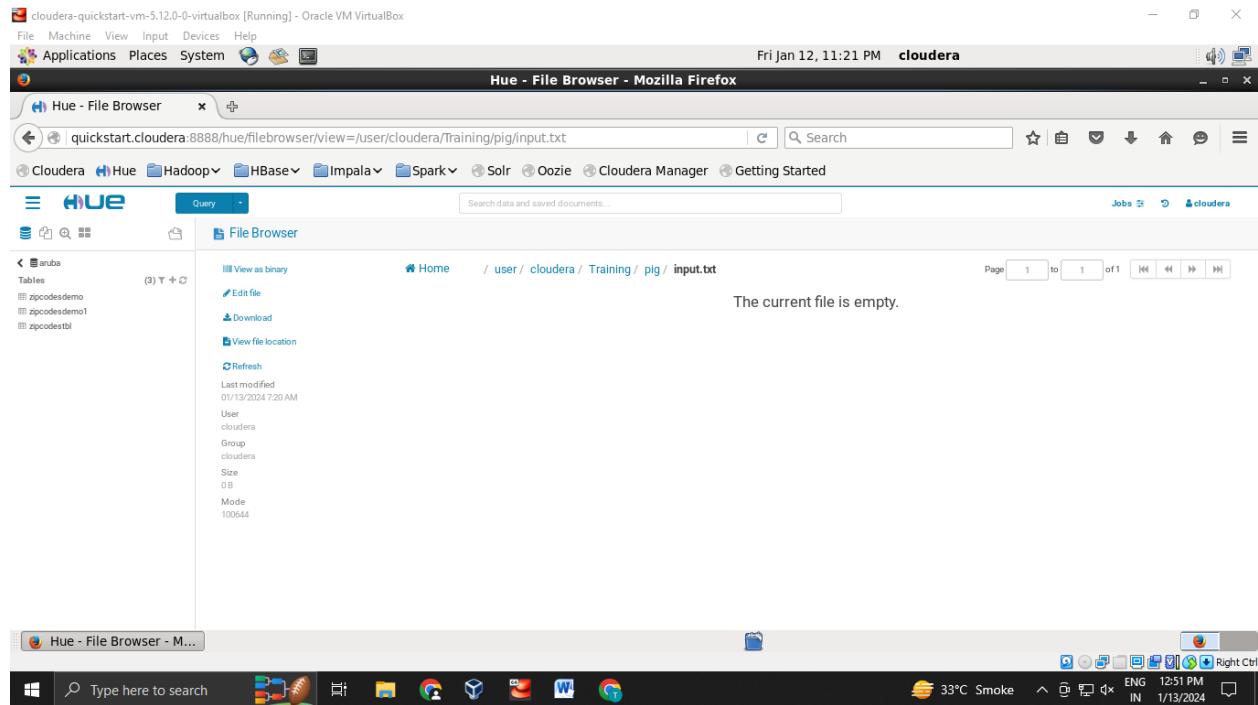
7) Creating input.txt file inside /usr/cloudera/training/pig directory

The screenshot shows the Cloudera Hue File Browser interface. The URL in the address bar is `quickstart.cloudera:8888/hue/filebrowser/view=/user/cloudera#/user/cloudera/Training/pig`. The browser title is "Hue - File Browser - Mozilla Firefox". The left sidebar shows tables like "zippedesdemo", "zippedesdemo1", and "zippedestbl". The main area displays a file list with columns: Name, Size, User, Group, Permissions, and Date. Two items are listed: "j" (size 0, user cloudera, group cloudera, permissions drwxr-x, date Jan 12, 2024 11:17 PM) and "." (size 0, user cloudera, group cloudera, permissions drwxr-x, date Jan 12, 2024 11:17 PM). A context menu is open over the "j" item, showing options: Upload, New, File, and Directory. The bottom status bar shows system information: 33°C, Smoke, ENG IN, 12:50 PM, 1/13/2024.

The screenshot shows the Cloudera Hue File Browser interface with a "Create File" dialog box overlaid. The dialog has a "File Name" input field containing "input.txt". Below the input field are "Cancel" and "Create" buttons. The background shows the same file browser interface as the previous screenshot, with the file list and sidebar visible.



8) Adding some contents to this input.txt file.



Click on Edit file option → Write the below lines and click on save button.

The screenshot shows the Hue File Browser interface. The left sidebar lists tables: zipcodesdemo, zipcodesdemo1, and zipcodestbl. The main content area displays the contents of the input.txt file:

```
We are studying Big Data Technology
We have covered the topic Hadoop ecosystem
We have executed wordcount using mapreduce
We have also implemented CRUD operations using MongoDB
```

At the bottom of the content area, there are 'Save' and 'Save as' buttons. Below the content area, it says '1 item in Trash'.

Save the input.txt file

The screenshot shows the Hue File Browser interface. The left sidebar lists tables: zipcodesdemo, zipcodesdemo1, and zipcodestbl. The main content area displays the contents of the input.txt file, identical to the previous screenshot:

```
We are studying Big Data Technology
We have covered the topic Hadoop ecosystem
We have executed wordcount using mapreduce
We have also implemented CRUD operations using MongoDB
```

At the top of the content area, there are options: 'View as binary', 'Edit file' (which is highlighted in blue), 'Download', and 'View file location'. Below the content area, there is a 'Page' navigation bar with buttons for 1, to, 1, of 1, and arrows. At the bottom of the content area, it says '1 item in Trash'.

9) Now Open the terminal. And start Pig by typing pig on terminal.

```
cloudera@quickstart:~
```

```
File Edit View Search Terminal Help
```

```
[cloudera@quickstart ~]$ pig
```

Now the pig started

```
cloudera@quickstart:~
```

```
File Edit View Search Terminal Help
```

```
[cloudera@quickstart ~]$ pig
```

```
log4j:WARN No appenders could be found for logger (org.apache.hadoop.util.Shell).
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
2024-01-12 23:30:11,605 [main] INFO org.apache.pig.Main - Apache Pig version 0.12.0-cdh5.12.0 (rexported) compiled Jun 29 2017, 04:34:31
2024-01-12 23:30:11,606 [main] INFO org.apache.pig.Main - Logging error messages to: /home/cloudera/pig_1705131011574.log
2024-01-12 23:30:11,640 [main] INFO org.apache.pig.impl.util.Utils - Default bootup file /home/cloudera/.pigbootup not found
2024-01-12 23:30:13,306 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.ad
2024-01-12 23:30:13,306 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2024-01-12 23:30:13,306 [main] INFO org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hadoop file system at: hdfs://quickstart.c
20
2024-01-12 23:30:16,819 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.ad
2024-01-12 23:30:16,819 [main] INFO org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to map-reduce job tracker at: localhost:8021
2024-01-12 23:30:16,821 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2024-01-12 23:30:16,975 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2024-01-12 23:30:16,978 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.ad
2024-01-12 23:30:17,161 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2024-01-12 23:30:17,172 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.ad
2024-01-12 23:30:17,411 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2024-01-12 23:30:17,419 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.ad
2024-01-12 23:30:17,635 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2024-01-12 23:30:17,641 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.ad
2024-01-12 23:30:17,864 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2024-01-12 23:30:17,866 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.ad
2024-01-12 23:30:18,086 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2024-01-12 23:30:18,090 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.ad
2024-01-12 23:30:18,282 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2024-01-12 23:30:18,288 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.ad
2024-01-12 23:30:18,496 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2024-01-12 23:30:18,497 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.ad
grunt> 
```

10) Now we have to load that input file where ever it is stored. By typing the command

```
Input1 = LOAD '/usr/cloudera/Training/pig/input.txt' AS (f1:chararray);
```

```
grunt> Input1 = LOAD '/user/cloudera/Training/pig/input.txt' AS (f1:chararray);
grunt> █
```

11) Now we are dumping the data. It will done the mapreduce task.

```
DUMP input1;
```

```
grunt> DUMP Input1;
2024-01-12 23:36:08,842 [main] INFO org.apache.pig.tools.pigstats.ScriptState - Pig
features used in the script: UNKNOWN
2024-01-12 23:36:08,917 [main] INFO org.apache.pig.newplan.logical.optimizer.Logical
PlanOptimizer - {RULES_ENABLED=[AddForEach, ColumnMapKeyPrune, DuplicateForEachColumn
Rewrite, GroupByConstParallelSetter, ImplicitSplitInserter, LimitOptimizer, LoadTypeC
astInserter, MergeFilter, MergeForEach, NewPartitionFilterOptimizer, PushDownForEachF
latten, PushUpFilter, SplitFilter, StreamTypeCastInserter], RULES_DISABLED=[FilterLog
icExpressionSimplifier, PartitionFilterOptimizer]}
2024-01-12 23:36:09,203 [main] INFO org.apache.pig.backend.hadoop.executionengine.ma
pReduceLayer.MRCompiler - File concatenation threshold: 100 optimistic? false
2024-01-12 23:36:09,246 [main] INFO org.apache.pig.backend.hadoop.executionengine.ma
pReduceLayer.MultiQueryOptimizer - MR plan size before optimization: 1
2024-01-12 23:36:09,246 [main] INFO org.apache.pig.backend.hadoop.executionengine.ma
pReduceLayer.MultiQueryOptimizer - MR plan size after optimization: 1
2024-01-12 23:36:09,636 [main] INFO org.apache.hadoop.yarn.client.RMProxy - Connecti
ng to ResourceManager at /0.0.0:8032
2024-01-12 23:36:10,099 [main] INFO org.apache.pig.tools.pigstats.ScriptState - Pig
script settings are added to the job
2024-01-12 23:36:10,243 [main] INFO org.apache.hadoop.conf.Configuration.deprecation
- mapred.job.reduce.markreset.buffer.percent is deprecated. Instead, use mapreduce.r
```

Job DAG:

```
job_1705129341413_0001
```

```
2024-01-12 23:37:08,216 [main] INFO org.apache.pig.backend.hadoop.executionengine.ma
pReduceLayer.MapReduceLauncher - Success!
2024-01-12 23:37:08,221 [main] INFO org.apache.hadoop.conf.Configuration.deprecation
- fs.default.name is deprecated. Instead, use fs.defaultFS
2024-01-12 23:37:08,221 [main] INFO org.apache.hadoop.conf.Configuration.deprecation
- mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
2024-01-12 23:37:08,222 [main] INFO org.apache.pig.data.SchemaTupleBackend - Key [pi
g.schematuple] was not set... will not generate code.
2024-01-12 23:37:08,254 [main] INFO org.apache.hadoop.mapreduce.lib.input.FileInputF
ormat - Total input paths to process : 1
2024-01-12 23:37:08,254 [main] INFO org.apache.pig.backend.hadoop.executionengine.ut
il.MapRedUtil - Total input paths to process : 1
(We are studying Big Data Technology)
(We have covered the topic Hadoop ecosystem)
(We have executed wordcount using mapreduce)
(We have also implemented CRUD operations using MongoDB)
```

```
grunt> █
```

12) wordsInEachLine = FOREACH input1 GENERATE flatten(TOKENIZE(f1)) as word;

```
grunt> wordsInEachLine = FOREACH Input1 GENERATE flatten(TOKENIZE(f1)) as word;
2024-01-12 23:42:44,316 [main] INFO org.apache.hadoop.conf.Configuration.deprecation
- fs.default.name is deprecated. Instead, use fs.defaultFS
2024-01-12 23:42:44,316 [main] INFO org.apache.hadoop.conf.Configuration.deprecation
- mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
grunt> ■
```

13) dump wordsInEachLine;

```
grunt> dump wordsInEachLine;
2024-01-12 23:44:21,589 [main] INFO org.apache.pig.tools.pigstats.ScriptState - Pig
features used in the script: UNKNOWN
2024-01-12 23:44:21,598 [main] INFO org.apache.pig.newplan.logical.optimizer.Logical
PlanOptimizer - {RULES_ENABLED=[AddForEach, ColumnMapKeyPrune, DuplicateForEachColumn
Rewrite, GroupByConstParallelSetter, ImplicitSplitInserter, LimitOptimizer, LoadTypeC
astInserter, MergeFilter, MergeForEach, NewPartitionFilterOptimizer, PushDownForEachF
latten, PushUpFilter, SplitFilter, StreamTypeCastInserter], RULES_DISABLED=[FilterLog
icExpressionSimplifier, PartitionFilterOptimizer]}
2024-01-12 23:44:21,629 [main] INFO org.apache.pig.backend.hadoop.executionengine.ma
pReduceLayer.MRCompiler - File concatenation threshold: 100 optimistic? false
2024-01-12 23:44:21,632 [main] INFO org.apache.pig.backend.hadoop.executionengine.ma
pReduceLayer.MultiQueryOptimizer - MR plan size before optimization: 1
2024-01-12 23:44:21,632 [main] INFO org.apache.pig.backend.hadoop.executionengine.ma
pReduceLayer.MultiQueryOptimizer - MR plan size after optimization: 1
2024-01-12 23:44:21,689 [main] INFO org.apache.hadoop.yarn.client.RMProxy - Connecti
ng to ResourceManager at /0.0.0.0:8032
2024-01-12 23:44:21,694 [main] INFO org.apache.pig.tools.pigstats.ScriptState - Pig
script settings are added to the job
2024-01-12 23:44:21,711 [main] INFO org.apache.pig.backend.hadoop.executionengine.ma
pReduceLayer.JobControlCompiler - mapred.job.reduce.markreset.buffer.percent is not s
2024-01-12 23:45:17,267 [main] INFO org.apache.hadoop.mapreduce.lib.input.FileInputF
ormat - Total input paths to process : 1
2024-01-12 23:45:17,267 [main] INFO org.apache.pig.backend.hadoop.executionengine.ut
il.MapRedUtil - Total input paths to process : 1
(We)
(are)
(studying)
(Big)
(Data)
(Technology)
(We)
(have)
(covered)
(the)
(topic)
(Hadoop)
(ecosystem)
(We)
(have)
(executed)
(wordcount)
(using)
(mapreduce)
(We)
(have)
(also)
(implemented)
(CRUD)
(operations)
(using)
(MongoDB)
grunt> ■
```

14) Now grouping the words present in each line.

```
groupedWords = group wordsInEachLine by word;
```

```
grunt> groupedWords = group wordsInEachLine by word;
grunt> █
```

```
dump groupedWords;
```

```
grunt> dump groupedWords;
2024-01-12 23:49:55,524 [main] INFO org.apache.pig.tools.pigstats.ScriptState
- Pig features used in the script: GROUP_BY
2024-01-12 23:49:55,526 [main] INFO org.apache.pig.newplan.logical.optimizer.
LogicalPlanOptimizer - {RULES_ENABLED=[AddForEach, ColumnMapKeyPrune, DuplicateForEachColumnRewrite, GroupByConstParallelSetter, ImplicitSplitInserter, LimitOptimizer, LoadTypeCastInserter, MergeFilter, MergeForEach, NewPartitionFilterOptimizer, PushDownForEachFlatten, PushUpFilter, SplitFilter, StreamTypeCastInserter], RULES_DISABLED=[FilterLogicExpressionSimplifier, PartitionFilterOptimizer]}
2024-01-12 23:49:55,538 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MRCompiler - File concatenation threshold: 100 optimistic?
false
2024-01-12 23:49:55,551 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size before optimization: 1
2024-01-12 23:49:55,551 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size after optimization: 1
2024-01-12 23:49:55,613 [main] INFO org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at /0.0.0.0:8032
2024-01-12 23:49:55,620 [main] INFO org.apache.pig.tools.pigstats.ScriptState
- Pig script settings are added to the job
2024-01-12 23:49:55,644 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - mapred.job.reduce.markreset.buffer.percent is not set, set to default 0.3
2024-01-12 23:50:58,710 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(We,{(We),(We),(We)})
(Big,{(Big)})
(are,{(are)})
(the,{(the)})
(CRUD,{(CRUD)})
(Data,{(Data)})
(also,{(also)})
(have,{(have),(have),(have)})
(topic,{(topic)})
(using,{(using),(using)})
(Hadoop,{(Hadoop)})
(MongoDB,{(MongoDB)})
(covered,{(covered)})
(executed,{(executed)})
(studying,{(studying)})
(ecosystem,{(ecosystem)})
(mapreduce,{(mapreduce)})
(wordcount,{(wordcount)})
(Technology,{(Technology)})
(operations,{(operations)})
(implemented,{(implemented)})
grunt> █
```

15) Now we count those words. For each group we count words in each line.

```
countedWords = foreach groupedWords generate group, COUNT(wordsInEachLine);
```

```
|ne)nt> countedWords = foreach groupedWords generate group, COUNT(wordsInEachLine);
grunt> █
```

16) dump countedWords;

Now the Final Output we are getting as word count for every word.

```
grunt> dump countedWords;
2024-01-12 23:55:08,377 [main] INFO org.apache.pig.tools.pigstats.ScriptState - Pig features
used in the script: GROUP_BY
2024-01-12 23:55:08,382 [main] INFO org.apache.pig.newplan.logical.optimizer.LogicalPlanOpti
mizer - {RULES_ENABLED=[AddForEach, ColumnMapKeyPrune, DuplicateForEachColumnRewrite, GroupBy
ConstParallelSetter, ImplicitSplitInserter, LimitOptimizer, LoadTypeCastInserter, MergeFilter
, MergeForEach, NewPartitionFilterOptimizer, PushDownForEachFlatten, PushUpFilter, SplitFilte
r, StreamTypeCastInserter], RULES_DISABLED=[FilterLogicExpressionSimplifier, PartitionFilterO
ptimizer]}
2024-01-12 23:55:08,409 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceL
ayer.MRCompiler - File concatenation threshold: 100 optimistic? false
2024-01-12 23:55:08,417 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceL
ayer.CombinerOptimizer - Choosing to move algebraic foreach to combiner
2024-01-12 23:55:08,433 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceL
ayer.MultiQueryOptimizer - MR plan size before optimization: 1
2024-01-12 23:55:08,433 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceL
ayer.MultiQueryOptimizer - MR plan size after optimization: 1
2024-01-12 23:55:08,476 [main] INFO org.apache.hadoop.yarn.client.RMProxy - Connecting to Re
sourceManager at /0.0.0.0:8032
2024-01-12 23:55:08,480 [main] INFO org.apache.pig.tools.pigstats.ScriptState - Pig script s
ettings are added to the job
2024-01-12 23:55:08,507 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceL
ayer.MapReduceUtil - Total input paths to process : 1
(We,4)
(Big,1)
(are,1)
(the,1)
(CRUD,1)
(Data,1)
(also,1)
(have,3)
(topic,1)
(using,2)
(Hadoop,1)
(MongoDB,1)
(covered,1)
(executed,1)
(studying,1)
(ecosystem,1)
(mapreduce,1)
(wordcount,1)
(Technology,1)
(operations,1)
(implemented,1)
grunt> █
```

As we can see from above image the Word “We” start with capital W occurred twice, word “we” start with small w occurred twice, word “Big” occurred once, and so on.

17) Now Exit from the grunt shell using quit command.

```
grunt> quit
[cloudera@quickstart ~]$ █
```

Practical 6: RDD Actions and Transformations

1. Install Java, PySpark and FindSpark

```
!apt-get install openjdk-8-jdk-headless -qq > /dev/null
```

```
!wget -q http://archive.apache.org/dist/spark/spark-3.1.1/spark-3.1.1-bin-hadoop3.2.tgz !tar xf spark-3.1.1-bin-hadoop3.2.tgz
```

```
!pip install -q findspark
```

```
!apt-get install openjdk-8-jdk-headless -qq > /dev/null
!wget -q http://archive.apache.org/dist/spark/spark-3.1.1/spark-3.1.1-bin-hadoop3.2.tgz
!tar xf spark-3.1.1-bin-hadoop3.2.tgz
!pip install -q findspark
```

2. Set environment variables

```
import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
os.environ["SPARK_HOME"] = "/content/spark-3.1.1-bin-hadoop3.2"
```

```
import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
os.environ["SPARK_HOME"] = "/content/spark-3.1.1-bin-hadoop3.2"
```

```
!ls
```

```
[5] !apt-get install openjdk-8-jdk-headless -qq > /dev/null
!wget -q http://archive.apache.org/dist/spark/spark-3.1.1/spark-3.1.1-bin-hadoop3.2.tgz
!tar xf spark-3.1.1-bin-hadoop3.2.tgz
!pip install -q findspark
```

```
[6] import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
os.environ["SPARK_HOME"] = "/content/spark-3.1.1-bin-hadoop3.2"
```

```
[7] !ls
sample_data          spark-3.1.1-bin-hadoop3.2.tgz  spark-3.1.1-bin-hadoop3.2.tgz.2
spark-3.1.1-bin-hadoop3.2  spark-3.1.1-bin-hadoop3.2.tgz.1
```

```
!pip install pyspark
```

```
!pip install spark
```

3. Initialize findSpark

```
import findspark
findspark.init()
```

```
import findspark
```

findspark.init()

4. Create Spark Session

```
from pyspark.sql import SparkSession  
spark = SparkSession.builder.getOrCreate()
```

```
from pyspark.context import SparkContext  
from pyspark.sql import SparkSession
```

```
spark = SparkSession.builder.master("local[*]").getOrCreate()
spark.conf.set("spark.sql.replEagerEval.enabled",True)
sc = SparkContext.getOrCreate()
```

SC

```
[1] import findspark
findspark.init()

[2] from pyspark.context import SparkContext
from pyspark.sql import SparkSession

[15] spark = SparkSession.builder.master("local[*]").getOrCreate()
spark.conf.set("spark.sql.replEagerEval.enabled", True)
sc = SparkContext.getOrCreate()

▶ sc
▶ Spark UI

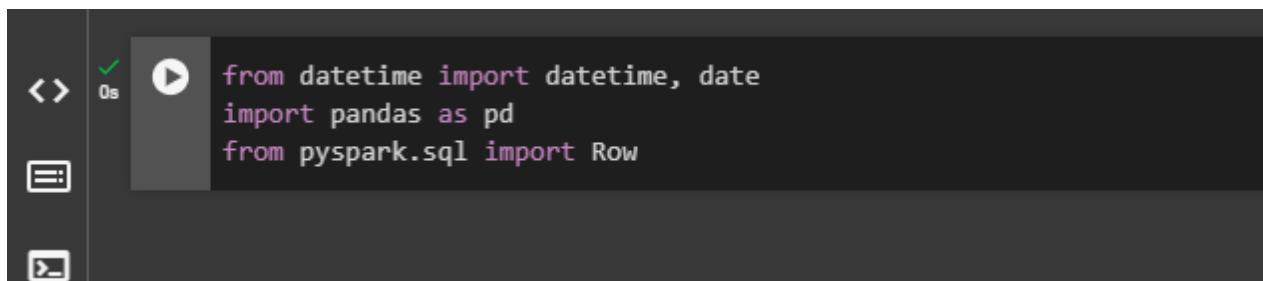
Version
v3.5.0
Master
local[*]
AppName
pyspark-shell
```

5. Create a dataframe by specifying row

```
from datetime import datetime, date  
import pandas as pd  
from pyspark.sql import Row
```

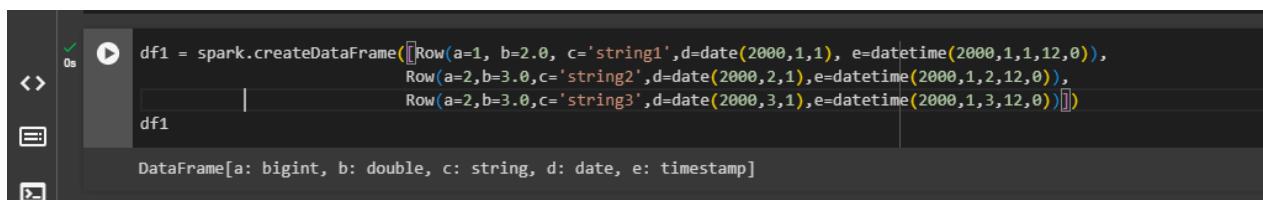
```
df1 = spark.createDataFrame([  
    Row(a=1, b=2., c='string1', d=date(2000, 1, 1), e=datetime(2000, 1, 1, 12, 0)),  
    Row(a=2, b=3., c='string2', d=date(2000, 2, 1), e=datetime(2000, 1, 2, 12, 0)),  
    Row(a=4, b=5., c='string3', d=date(2000, 3, 1), e=datetime(2000, 1, 3, 12, 0))  
])  
df
```

```
from datetime import datetime, date  
import pandas as pd  
from pyspark.sql import Row
```



```
from datetime import datetime, date
import pandas as pd
from pyspark.sql import Row
```

```
df1 = spark.createDataFrame([Row(a=1, b=2.0, c='string1',d=date(2000,1,1),
e=datetime(2000,1,1,12,0)),
                             Row(a=2,b=3.0,c='string2',d=date(2000,2,1),e=datetime(2000,1,2,12,0)),
                             Row(a=2,b=3.0,c='string3',d=date(2000,3,1),e=datetime(2000,1,3,12,0))])
df1
```

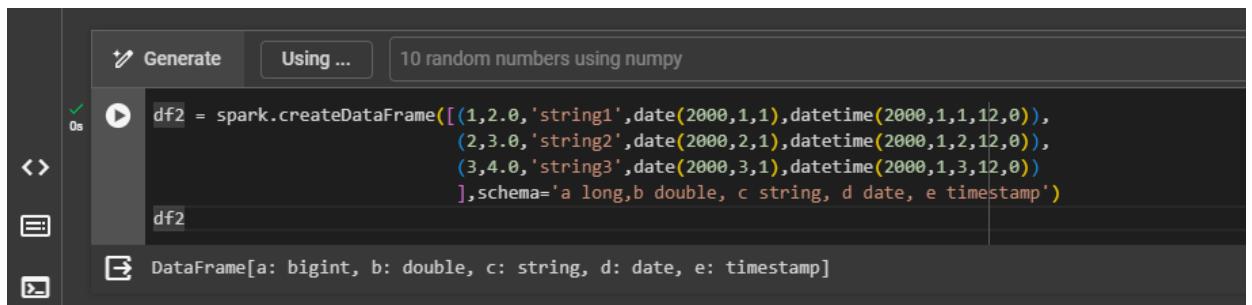


```
df1 = spark.createDataFrame([Row(a=1, b=2.0, c='string1',d=date(2000,1,1),
e=datetime(2000,1,1,12,0)),
                             Row(a=2,b=3.0,c='string2',d=date(2000,2,1),e=datetime(2000,1,2,12,0)),
                             Row(a=2,b=3.0,c='string3',d=date(2000,3,1),e=datetime(2000,1,3,12,0))])
df1
```

DataFrame[a: bigint, b: double, c: string, d: date, e: timestamp]

6. Create a PySpark DataFrame with an explicit schema.

```
df2 = spark.createDataFrame([
    (1, 2., 'string1', date(2000, 1, 1), datetime(2000, 1, 1, 12, 0)),
    (2, 3., 'string2', date(2000, 2, 1), datetime(2000, 1, 2, 12, 0)),
    (3, 4., 'string3', date(2000, 3, 1), datetime(2000, 1, 3, 12, 0))
], schema='a long, b double, c string, d date, e timestamp')
df2 = spark.createDataFrame([(1,2.0,'string1',date(2000,1,1),datetime(2000,1,1,12,0)),
                           (2,3.0,'string2',date(2000,2,1),datetime(2000,1,2,12,0)),
                           (3,4.0,'string3',date(2000,3,1),datetime(2000,1,3,12,0))],
                           schema='a long,b double, c string, d date, e timestamp')
df2
```



Generate Using ... 10 random numbers using numpy

```
df2 = spark.createDataFrame([(1,2.0,'string1',date(2000,1,1),datetime(2000,1,1,12,0)),
                           (2,3.0,'string2',date(2000,2,1),datetime(2000,1,2,12,0)),
                           (3,4.0,'string3',date(2000,3,1),datetime(2000,1,3,12,0))],
                           schema='a long,b double, c string, d date, e timestamp')
df2
```

DataFrame[a: bigint, b: double, c: string, d: date, e: timestamp]

7. Create a PySpark DataFrame from a pandas DataFrame

```
df_pandas = pd.DataFrame({  
    'a': [1, 2, 3],  
    'b': [2., 3., 4.],  
    'c': ['string1', 'string2', 'string3'],  
    'd': [date(2000, 1, 1), date(2000, 2, 1), date(2000, 3, 1)],  
    'e': [datetime(2000, 1, 1, 12, 0), datetime(2000, 1, 2, 12, 0), datetime(2000, 1, 3, 12, 0)]  
})  
df3 = spark.createDataFrame(df_pandas)  
df3
```

```
df_pandas = pd.DataFrame({  
    'a':[1,2,3],  
    'b':[2.,3.,4.],  
    'c':['string1','string2','string3'],  
    'd':[date(2000,1,1),date(2000,2,1),date(2000,3,1)],  
    'e':[datetime(2000,1,1,12,0),datetime(2000,1,2,12,0),datetime(2000,1,3,12,0)]  
})  
df3 = spark.createDataFrame(df_pandas)  
df3
```



The screenshot shows a Jupyter Notebook cell with the following code:

```
df_pandas = pd.DataFrame({  
    'a': [1, 2, 3],  
    'b': [2., 3., 4.],  
    'c': ['string1', 'string2', 'string3'],  
    'd': [date(2000, 1, 1), date(2000, 2, 1), date(2000, 3, 1)],  
    'e': [datetime(2000, 1, 1, 12, 0), datetime(2000, 1, 2, 12, 0), datetime(2000, 1, 3, 12, 0)]  
})  
df3 = spark.createDataFrame(df_pandas)  
df3
```

Below the code, there is a warning message:

```
content/spark-3.1.1-bin-hadoop3.2/python/pyspark/sql/pandas/conversion.py:331: FutureWarning: iteritems is deprecated and will be removed in a future version. Use  
for column, series in pdf.iteritems():  
atDataFrame[a: bigint, b: double, c: string, d: date, e: timestamp]
```

8. Display the dataframe contents.

```
df1.show()  
df2.show()  
df3.show()
```

```
df1.show()  
df2.show()  
df3.show()
```

The screenshot shows a Jupyter Notebook interface with three dataframes (df1, df2, df3) displayed as tabular output. The code cell contains the command `df1.show()`. The resulting output shows a table with columns labeled a, b, c, d, and e. The first row has values 1, 2.0, string1, 2000-01-01, and 2000-01-01 12:00:00. The second row has values 2, 3.0, string2, 2000-02-01, and 2000-01-02 12:00:00. The third row has values 2, 3.0, string3, 2000-03-01, and 2000-01-03 12:00:00. The code cell for df2 and df3 also produces similar tables with identical data. The sidebar on the left includes icons for file operations, search, and other notebook functions.

a	b	c	d	e
1	2.0	string1	2000-01-01	2000-01-01 12:00:00
2	3.0	string2	2000-02-01	2000-01-02 12:00:00
2	3.0	string3	2000-03-01	2000-01-03 12:00:00

a	b	c	d	e
1	2.0	string1	2000-01-01	2000-01-01 12:00:00
2	3.0	string2	2000-02-01	2000-01-02 12:00:00
3	4.0	string3	2000-03-01	2000-01-03 12:00:00

a	b	c	d	e
1	2.0	string1	2000-01-01	2000-01-01 12:00:00
2	3.0	string2	2000-02-01	2000-01-02 12:00:00
3	4.0	string3	2000-03-01	2000-01-03 12:00:00

9. Print the schema of dataframe

```
df1.printSchema()  
df2.printSchema()  
df3.printSchema()
```

```
df1.printSchema()  
df2.printSchema()  
df3.printSchema()
```

The screenshot shows a Jupyter Notebook interface with three code cells and their corresponding schema outputs.

- Code Cell 1:** Contains the code: `df1.printSchema()`, `df2.printSchema()`, and `df3.printSchema()`. The first two lines have been run, indicated by a green checkmark and a play button icon, while the third line is still pending.
- Output 1:** Shows the schema for `df1`:

```
root
|-- a: long (nullable = true)
|-- b: double (nullable = true)
|-- c: string (nullable = true)
|-- d: date (nullable = true)
|-- e: timestamp (nullable = true)
```
- Output 2:** Shows the schema for `df2`:

```
root
|-- a: long (nullable = true)
|-- b: double (nullable = true)
|-- c: string (nullable = true)
|-- d: date (nullable = true)
|-- e: timestamp (nullable = true)
```
- Output 3:** Shows the schema for `df3`:

```
root
|-- a: long (nullable = true)
|-- b: double (nullable = true)
|-- c: string (nullable = true)
|-- d: date (nullable = true)
|-- e: timestamp (nullable = true)
```

10. Show only the first row of the dataframe.

```
df1.show(1)
```

```
df1.show(1)

+---+---+-----+-----+
| a| b|      c|      d|      e|
+---+---+-----+-----+
| 1|2.0|string1|2000-01-01|2000-01-01 12:00:00|
+---+---+-----+-----+
only showing top 1 row
```

11. Enable spark.sql.repl.eagerEval.enabled configuration for the eager evaluation of PySpark DataFrame in notebooks such as Jupyter. The number of rows to show can be controlled via spark.sql.repl.eagerEval.maxNumRows configuration.

```
spark.conf.set('spark.sql.repl.eagerEval.enabled', True)
df1
```

```
spark.conf.set('spark.sql.repl.eagerEval.enabled',True)
df1
```

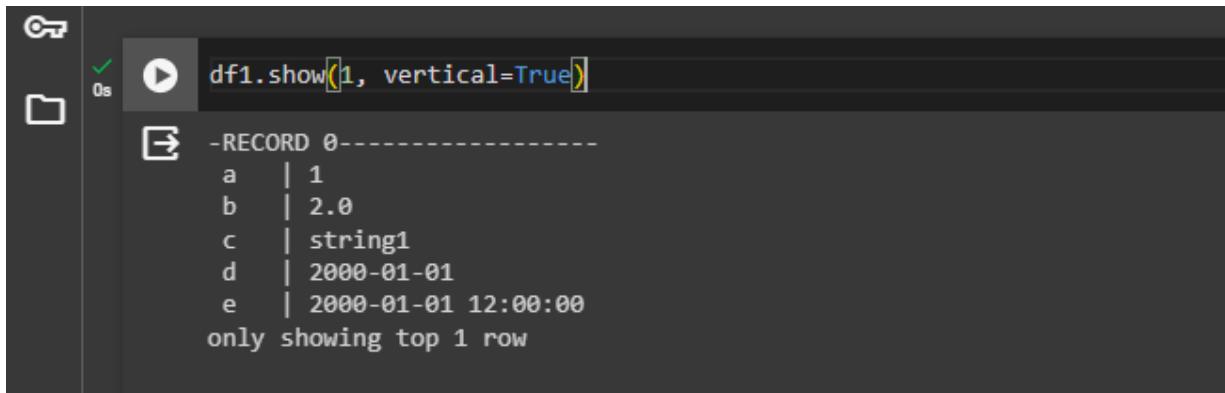
```
spark.conf.set('spark.sql.repl.eagerEval.enabled',True)
df1

+---+---+-----+-----+
| a| b|      c|      d|      e|
+---+---+-----+-----+
| 1|2.0|string1|2000-01-01|2000-01-01 12:00:00|
| 2|3.0|string2|2000-02-01|2000-01-02 12:00:00|
| 2|3.0|string3|2000-03-01|2000-01-03 12:00:00|
+---+---+-----+-----+
```

12. Display the row in vertical fashion.

```
df1.show(1, vertical=True)
```

```
df1.show(1, vertical=True)
```



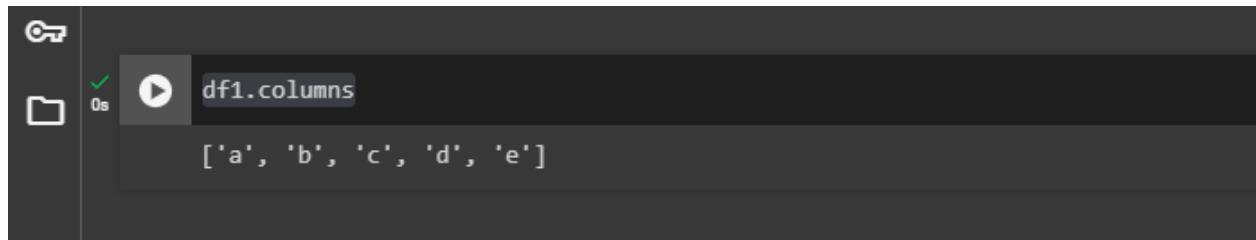
```
df1.show(1, vertical=True)
```

```
-RECORD 0-
a | 1
b | 2.0
c | string1
d | 2000-01-01
e | 2000-01-01 12:00:00
only showing top 1 row
```

13. Display list of columns

```
df1.columns
```

```
df1.columns
```



```
df1.columns
```

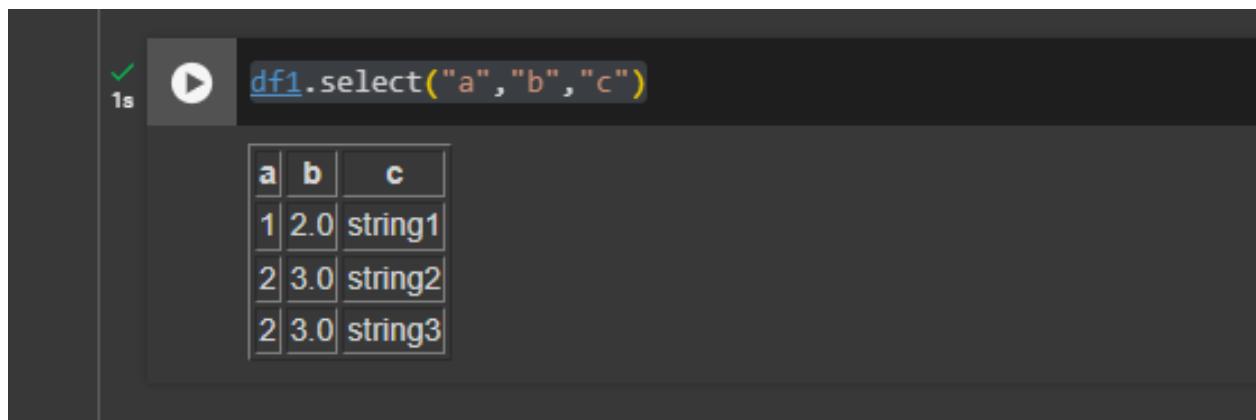
```
['a', 'b', 'c', 'd', 'e']
```

14. Show the summary of the DataFrame

```
df1.select("a", "b", "c")
```

```
df1.select("a", "b", "c").describe().show()
```

```
df1.select("a", "b", "c")
```



```
df1.select("a", "b", "c")
```

a	b	c
1	2.0	string1
2	3.0	string2
2	3.0	string3

```
df1.select("a", "b", "c").describe().show()
```

```
2s  df1.select("a", "b", "c").describe().show()
```

summary	a	b	c
count	3	3	3
mean	1.6666666666666667	2.6666666666666665	null
stddev	0.5773502691896257	0.5773502691896257	null
min	1	2.0	string1
max	2	3.0	string3

15. collects the distributed data to the driver side as the local data in Python.

```
df1.collect
```

```
0s  df1.collect
```

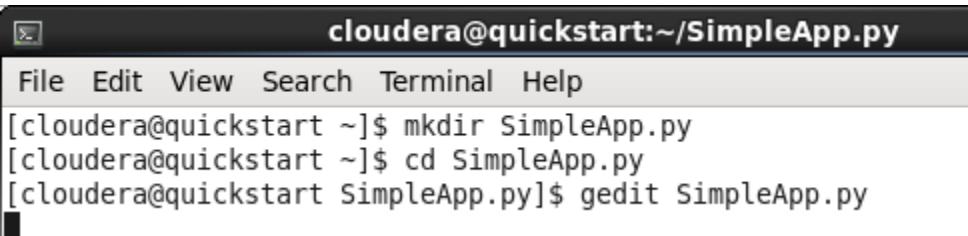
```
→ <bound method DataFrame.collect of DataFrame[a: bigint, b: double, c: string, d: date, e: timestamp]>
```

Practical No. 7: Creating PySpark Application

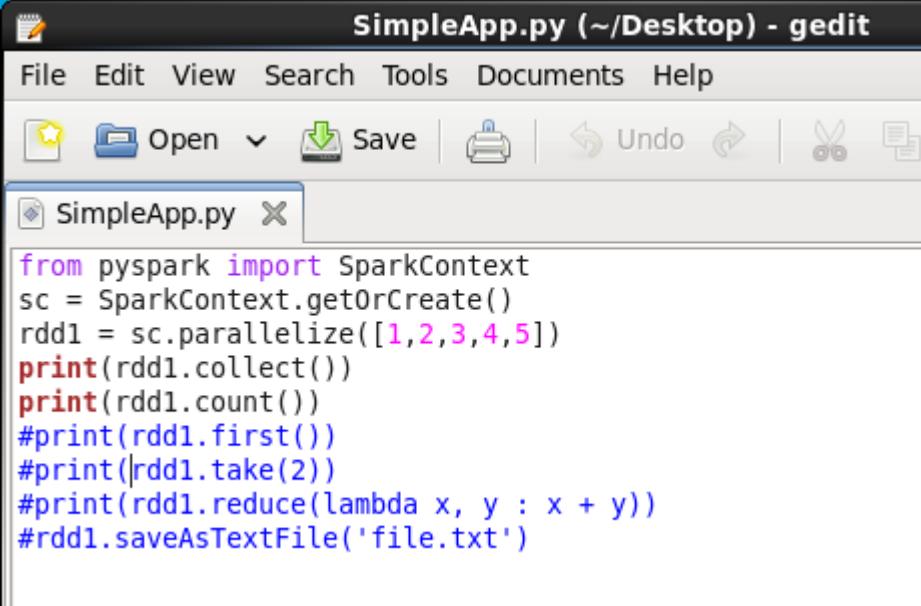
Example 1: RDD Actions



Open another Terminal



```
cloudera@quickstart:~/SimpleApp.py
File Edit View Search Terminal Help
[cloudera@quickstart ~]$ mkdir SimpleApp.py
[cloudera@quickstart ~]$ cd SimpleApp.py
[cloudera@quickstart SimpleApp.py]$ gedit SimpleApp.py
```

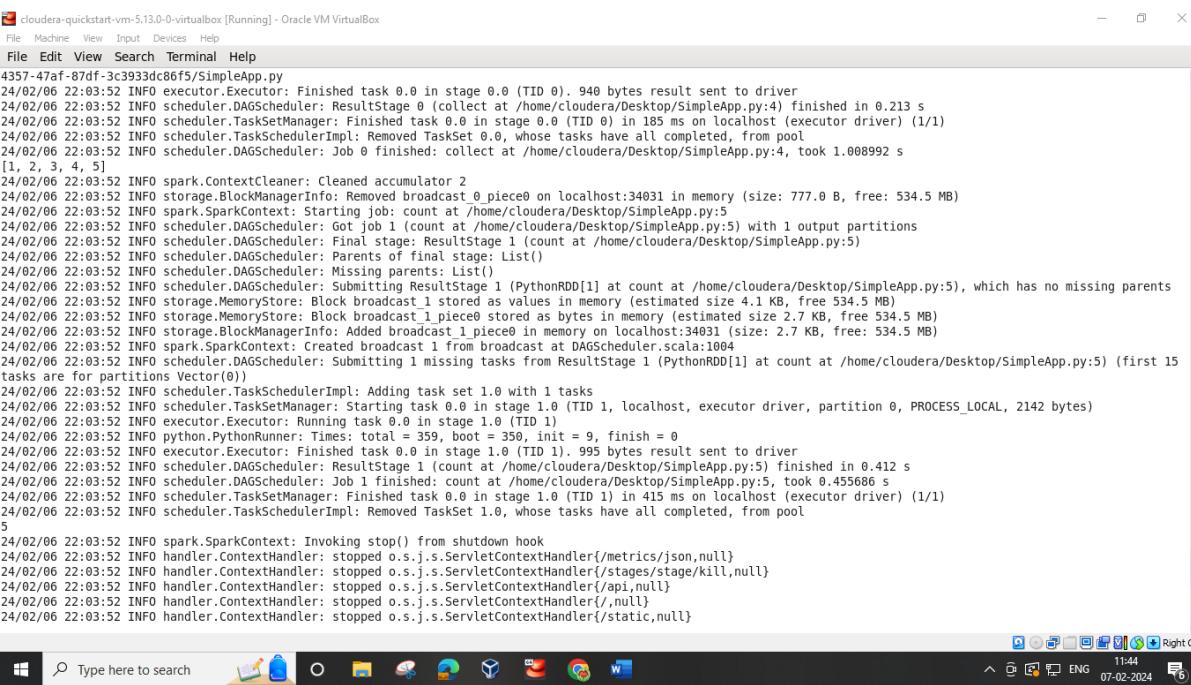



```
SimpleApp.py (~/Desktop) - gedit
File Edit View Search Tools Documents Help
File Open Save Undo | | | | | | | |
SimpleApp.py X
from pyspark import SparkContext
sc = SparkContext.getOrCreate()
rdd1 = sc.parallelize([1,2,3,4,5])
print(rdd1.collect())
print(rdd1.count())
#print(rdd1.first())
#print(rdd1.take(2))
#print(rdd1.reduce(lambda x, y : x + y))
#rdd1.saveAsTextFile('file.txt')
```



```
[cloudera@quickstart SimpleApp.py]$ /usr/lib/spark/bin/spark-submit /home/cloudera/Desktop/SimpleApp.py
[cloudera@quickstart SimpleApp.py]$
```

OUTPUT



```
cloudera-quickstart-vm-5.13.0-0-virtualbox [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
File Edit View Search Terminal Help
4357-47af-87df-3c393dc86f5/SimpleApp.py
24/02/06 22:03:52 INFO executor.Executor: Finished task 0.0 in stage 0.0 (TID 0). 940 bytes result sent to driver
24/02/06 22:03:52 INFO scheduler.DAGScheduler: ResultStage 0 (collect at /home/cloudera/Desktop/SimpleApp.py:4) finished in 0.213 s
24/02/06 22:03:52 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 0.0 (TID 0) in 185 ms on localhost (executor driver) (1/1)
24/02/06 22:03:52 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 0.0, whose tasks have all completed, from pool
24/02/06 22:03:52 INFO DAGScheduler: Job 0 finished: collect at /home/cloudera/Desktop/SimpleApp.py:4, took 1.008992 s
[1, 2, 3, 4, 5]
24/02/06 22:03:52 INFO spark.ContextCleaner: Cleaned accumulator 2
24/02/06 22:03:52 INFO storage.BlockManagerInfo: Removed broadcast_0_piece0 on localhost:34031 in memory (size: 777.0 B, free: 534.5 MB)
24/02/06 22:03:52 INFO spark.SparkContext: Starting job: count at /home/cloudera/Desktop/SimpleApp.py:5
24/02/06 22:03:52 INFO scheduler.DAGScheduler: Got job 1 (count at /home/cloudera/Desktop/SimpleApp.py:5) with 1 output partitions
24/02/06 22:03:52 INFO scheduler.DAGScheduler: Final stage: ResultStage 1 (count at /home/cloudera/Desktop/SimpleApp.py:5)
24/02/06 22:03:52 INFO scheduler.DAGScheduler: Parents of final stage: List()
24/02/06 22:03:52 INFO scheduler.DAGScheduler: Submitting parents: List()
24/02/06 22:03:52 INFO scheduler.DAGScheduler: Submitting ResultStage 1 (PythonRDD[1] at count at /home/cloudera/Desktop/SimpleApp.py:5), which has no missing parents
24/02/06 22:03:52 INFO storage.MemoryStore: Block broadcast_1 stored as values in memory (estimated size 4.1 KB, free 534.5 MB)
24/02/06 22:03:52 INFO storage.MemoryStore: Block broadcast_1_piece0 stored as bytes in memory (estimated size 2.7 KB, free 534.5 MB)
24/02/06 22:03:52 INFO storage.BlockManagerInfo: Added broadcast_1_piece0 in memory on localhost:34031 (size: 2.7 KB, free: 534.5 MB)
24/02/06 22:03:52 INFO spark.SparkContext: Created broadcast 1 from broadcast at DAGScheduler.scala:1004
24/02/06 22:03:52 INFO scheduler.DAGScheduler: Submitting 1 missing tasks from ResultStage 1 (PythonRDD[1] at count at /home/cloudera/Desktop/SimpleApp.py:5) (first 15 tasks are for partitions Vector(0))
24/02/06 22:03:52 INFO scheduler.TaskschedulerImpl: Adding task set 1.0 with 1 tasks
24/02/06 22:03:52 INFO scheduler.TaskSetManager: Starting task 0.0 in stage 1.0 (TID 1, localhost, executor driver, partition 0, PROCESS_LOCAL, 2142 bytes)
24/02/06 22:03:52 INFO executor.Executor: Running task 0.0 in stage 1.0 (TID 1)
24/02/06 22:03:52 INFO python.PythonRunner: Times: total = 359, boot = 350, init = 9, finish = 0
24/02/06 22:03:52 INFO executor.Executor: Finished task 0.0 in stage 1.0 (TID 1). 995 bytes result sent to driver
24/02/06 22:03:52 INFO scheduler.DAGScheduler: ResultStage 1 (count at /home/cloudera/Desktop/SimpleApp.py:5) finished in 0.412 s
24/02/06 22:03:52 INFO scheduler.DAGScheduler: Job 1 finished: count at /home/cloudera/Desktop/SimpleApp.py:5, took 0.455686 s
24/02/06 22:03:52 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 1.0 (TID 1) in 415 ms on localhost (executor driver) (1/1)
24/02/06 22:03:52 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 1.0, whose tasks have all completed, from pool
5
24/02/06 22:03:52 INFO spark.SparkContext: Invoking stop() from shutdown hook
24/02/06 22:03:52 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/metrics/json,null}
24/02/06 22:03:52 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/stages/stage/kill,null}
24/02/06 22:03:52 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/api,null}
24/02/06 22:03:52 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/null,null}
24/02/06 22:03:52 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/static,null}
```

```

SimpleApp.py (~/Desktop) - gedit
File Edit View Search Tools Documents Help
Open Save Undo Redo Cut Copy Paste Find Replace Select All
SimpleApp.py X
from pyspark import SparkContext
sc = SparkContext.getOrCreate()
rdd1 = sc.parallelize([1,2,3,4,5])
print(rdd1.collect())
print(rdd1.count())
print(rdd1.first())
print(rdd1.take(2))
print(rdd1.reduce(lambda x, y : x + y))
rdd1.saveAsTextFile('file.txt')

```

OUTPUT

```

cloudera@quickstart-vm-5.13.0-0-virtualbox [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System Terminal Help
Tue Feb 6, 10:17 PM cloudera
cloudera@quickstart:~/SimpleApp.py
File Edit View Search Terminal Help
24/02/06 22:17:11 INFO scheduler.DAGScheduler: Job 0 finished: collect at /home/cloudera/Desktop/SimpleApp.py:4, took 1.049866 s
[1, 2, 3, 4, 5]
24/02/06 22:17:12 INFO spark.ContextCleaner: Cleaned accumulator 2
24/02/06 22:17:12 INFO storage.BlockManagerInfo: Removed broadcast_0_piece0 on localhost:36502 in memory (size: 777.0 B, free: 534.5 MB)
24/02/06 22:17:12 INFO spark.SparkContext: Starting job: count at /home/cloudera/Desktop/SimpleApp.py:5
24/02/06 22:17:12 INFO scheduler.DAGScheduler: Got job 1 (count at /home/cloudera/Desktop/SimpleApp.py:5) with 1 output partitions
24/02/06 22:17:12 INFO scheduler.DAGScheduler: Final stage: ResultStage 1 (count at /home/cloudera/Desktop/SimpleApp.py:5)
24/02/06 22:17:12 INFO scheduler.DAGScheduler: Parents of final stage: List()
24/02/06 22:17:12 INFO scheduler.DAGScheduler: Missing parents: List()
24/02/06 22:17:12 INFO scheduler.DAGScheduler: Submitting ResultStage 1 (PythonRDD[1] at count at /home/cloudera/Desktop/SimpleApp.py:5), which has no missing parents
24/02/06 22:17:12 INFO storage.MemoryStore: Block broadcast_1 stored as values in memory (estimated size 4.1 KB, free 534.5 MB)
24/02/06 22:17:12 INFO storage.MemoryStore: Block broadcast_1_piece0 stored as bytes in memory (estimated size 2.7 KB, free 534.5 MB)
24/02/06 22:17:12 INFO storage.BlockManagerInfo: Added broadcast_1_piece0 in memory on localhost:36502 (size: 2.7 KB, free: 534.5 MB)
24/02/06 22:17:12 INFO spark.SparkContext: Created broadcast 1 from broadcast at DAGScheduler.scala:1004
24/02/06 22:17:12 INFO scheduler.DAGScheduler: Submitting 1 missing tasks from ResultStage 1 (PythonRDD[1] at count at /home/cloudera/Desktop/SimpleApp.py:5) (first 15 tasks are for partitions Vector(0))
24/02/06 22:17:12 INFO scheduler.TaskSchedulerImpl: Adding task set 1.0 with 1 tasks
24/02/06 22:17:12 INFO scheduler.TaskSetManager: Starting task 0.0 in stage 1.0 (TID 1, localhost, executor driver, partition 0, PROCESS_LOCAL, 2142 bytes)
24/02/06 22:17:12 INFO executor.Executor: Running task 0.0 in stage 1.0 (TID 1)
24/02/06 22:17:12 INFO python.PythonRunner: Times: total = 349, boot = 337, init = 12, finish = 0
24/02/06 22:17:12 INFO executor.Executor: Finished task 0.0 in stage 1.0 (TID 1). 995 bytes result sent to driver
24/02/06 22:17:12 INFO scheduler.DAGScheduler: ResultStage 1 (count at /home/cloudera/Desktop/SimpleApp.py:5) finished in 0.433 s
24/02/06 22:17:12 INFO scheduler.DAGScheduler: Job 1 finished: count at /home/cloudera/Desktop/SimpleApp.py:5, took 0.480339 s
24/02/06 22:17:12 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 1.0 (TID 1) in 436 ms on localhost (executor driver) (1/1)
24/02/06 22:17:12 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 1.0, whose tasks have all completed, from pool
5
24/02/06 22:17:12 INFO spark.SparkContext: Starting job: runJob at PythonRDD.scala:393
24/02/06 22:17:12 INFO scheduler.DAGScheduler: Got job 2 (runJob at PythonRDD.scala:393) with 1 output partitions
24/02/06 22:17:12 INFO scheduler.DAGScheduler: Final stage: ResultStage 2 (runJob at PythonRDD.scala:393)
24/02/06 22:17:12 INFO scheduler.DAGScheduler: Parents of final stage: List()
24/02/06 22:17:12 INFO scheduler.DAGScheduler: Missing parents: List()
24/02/06 22:17:12 INFO scheduler.DAGScheduler: Submitting ResultStage 2 (PythonRDD[2] at RDD at PythonRDD.scala:43), which has no missing parents
24/02/06 22:17:12 INFO storage.MemoryStore: Block broadcast_2 stored as values in memory (estimated size 3.2 KB, free 534.5 MB)

```

Example 2: RDD Transformations

1. map()

```
my_rdd = sc.parallelize([1,2,3,4])
print(my_rdd.map(lambda x: x+ 10).collect())
```

SimpleApp.py (~/Desktop) - gedit

```
[cloudera@quickstart SimpleApp.py]$ /usr/lib/spark/bin/spark-submit /home/cloudera/Desktop/SimpleApp.py
[cloudera@quickstart SimpleApp.py]$
```

Output:

```
cloudera@quickstart-vm-5.13.0-0-virtualbox [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System
Tue Feb 6, 10:33 PM cloudera
cloudera@quickstart:~/SimpleApp.py
File Edit View Terminal Help
24/02/06 22:29:33 INFO scheduler.TaskSetManager: Starting task 0.0 in stage 0.0 (TID 0, localhost, executor driver, partition 0, PROCESS_LOCAL, 2140 bytes)
24/02/06 22:29:33 INFO executor.Executor: Running task 0.0 in stage 0.0 (TID 0)
24/02/06 22:29:33 INFO executor.Executor: Fetching file:/home/cloudera/Desktop/SimpleApp.py with timestamp 1707287379963
24/02/06 22:29:33 INFO util.Utils: /home/cloudera/Desktop/SimpleApp.py has been previously copied to /tmp/spark-e9246f02-0cad-4621-ad30-b3649a67819d/userFiles-0af43bbd-876f-4b6a-9d01-11fec6c525d/SimpleApp.py
24/02/06 22:29:33 INFO python.PythonRunner: Times: total = 372, boot = 361, init = 10, finish = 1
24/02/06 22:29:33 INFO executor.Executor: Finished task 0.0 in stage 0.0 (TID 0). 1036 bytes result sent to driver
24/02/06 22:29:33 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 0.0 (TID 0) in 616 ms on localhost (executor driver) (1/1)
24/02/06 22:29:33 INFO scheduler.DAGScheduler: ResultStage 0 (collect at /home/cloudera/Desktop/SimpleApp.py:4) finished in 0.658 s
24/02/06 22:29:33 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 0.0, whose tasks have all completed, from pool
24/02/06 22:29:34 INFO scheduler.DAGScheduler: Job 0 finished: collect: at /home/cloudera/Desktop/SimpleApp.py:4, took 1.592898 s
[11, 12, 13, 14]
24/02/06 22:29:34 INFO spark.SparkContext: Invoking stop() from shutdown hook
24/02/06 22:29:34 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/metrics/json,null}
24/02/06 22:29:34 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/stages/kill,null}
24/02/06 22:29:34 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/api,null}
24/02/06 22:29:34 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/_,null}
24/02/06 22:29:34 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/static,null}
24/02/06 22:29:34 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/executors/threadDump/json,null}
24/02/06 22:29:34 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/executors/threadDump,null}
24/02/06 22:29:34 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/executors/json,null}
24/02/06 22:29:34 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/executors,null}
24/02/06 22:29:34 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/environment/json,null}
24/02/06 22:29:34 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/environment,null}
24/02/06 22:29:34 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/storage/rdd/json,null}
24/02/06 22:29:34 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/storage/rdd,null}
24/02/06 22:29:34 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/storage/json,null}
24/02/06 22:29:34 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/storage,null}
24/02/06 22:29:34 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/stages/json,null}
24/02/06 22:29:34 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/stages/null}
24/02/06 22:29:34 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/stages/stage/json,null}
24/02/06 22:29:34 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/stages/stage,null}
24/02/06 22:29:34 INFO cloudera@quickstart:~/SimpleApp.py o.s.j.s.ServletContextHandler{/stages/json,null}
```

2. filter()

```
filter_rdd = sc.parallelize([2, 3, 4, 5, 6, 7])
print(filter_rdd.filter(lambda x: x%2 == 0).collect())
```

filter.py (~/Desktop) - gedit

```
[cloudera@quickstart-vm-5.13.0-0-virtualbox [Running] - Oracle VM VirtualBox
File Edit View Search Tools Documents Help
Applications Places System
Tue Feb 6, 10:33 PM cloudera
cloudera@quickstart:~/filter.py
File Edit View Terminal Help
24/02/06 22:29:33 INFO scheduler.TaskSetManager: Starting task 0.0 in stage 0.0 (TID 0, localhost, executor driver, partition 0, PROCESS_LOCAL, 2140 bytes)
24/02/06 22:29:33 INFO executor.Executor: Running task 0.0 in stage 0.0 (TID 0)
24/02/06 22:29:33 INFO executor.Executor: Fetching file:/home/cloudera/Desktop/filter.py with timestamp 1707287379963
24/02/06 22:29:33 INFO util.Utils: /home/cloudera/Desktop/filter.py has been previously copied to /tmp/spark-e9246f02-0cad-4621-ad30-b3649a67819d/userFiles-0af43bbd-876f-4b6a-9d01-11fec6c525d/filter.py
24/02/06 22:29:33 INFO python.PythonRunner: Times: total = 372, boot = 361, init = 10, finish = 1
24/02/06 22:29:33 INFO executor.Executor: Finished task 0.0 in stage 0.0 (TID 0). 1036 bytes result sent to driver
24/02/06 22:29:33 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 0.0 (TID 0) in 616 ms on localhost (executor driver) (1/1)
24/02/06 22:29:33 INFO scheduler.DAGScheduler: ResultStage 0 (filter at /home/cloudera/Desktop/filter.py:4) finished in 0.658 s
24/02/06 22:29:33 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 0.0, whose tasks have all completed, from pool
24/02/06 22:29:34 INFO scheduler.DAGScheduler: Job 0 finished: filter: at /home/cloudera/Desktop/filter.py:4, took 1.592898 s
[2, 4, 6]
```

```
[cloudera@quickstart SimpleApp.py]$ /usr/lib/spark/bin/spark-submit /home/cloudera/Desktop/filter.py
```

OUTPUT:

```

cloudera@quickstart-vm-5.13.0-0-virtualbox [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System
Tue Feb 6, 10:43 PM cloudera
cloudera@quickstart:~/SimpleApp.py
File Edit View Search Terminal Help
24/02/06 22:37:11 INFO spark.SparkContext: Created broadcast 0 from broadcast at DAGScheduler.scala:1004
24/02/06 22:37:11 INFO scheduler.DAGScheduler: Submitting 1 missing tasks from ResultStage 0 (PythonRDD[1] at collect at /home/cloudera/Desktop/filter.py:4) (first 15 tasks are for partitions Vector(0))
24/02/06 22:37:11 INFO scheduler.TaskSchedulerImpl: Adding task set 0.0 with 1 tasks
24/02/06 22:37:11 INFO scheduler.TaskSetManager: Starting task 0.0 in stage 0.0 (TID 0, localhost, executor driver, partition 0, PROCESS_LOCAL, 2141 bytes)
24/02/06 22:37:11 INFO executor.Executor: Running task 0.0 in stage 0.0 (TID 0)
24/02/06 22:37:11 INFO executor.Executor: Fetching file:/home/cloudera/Desktop/filter.py with timestamp 1707287828982
24/02/06 22:37:11 INFO util.Utils: /home/cloudera/Desktop/filter.py has been previously copied to /tmp/spark-67c4d608-5b30-a3a3-0fefc573a3b9/userFiles-a113c121-8317-47eb-ba67-75713abc971/filter.py
24/02/06 22:37:12 INFO python.PythonRunner: Times: total = 405, boot = 390, init = 14, finish = 1
24/02/06 22:37:12 INFO executor.Executor: Finished task 0.0 in stage 0.0 (TID 0). 1017 bytes result sent to driver
24/02/06 22:37:12 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 0.0 (TID 0) in 656 ms on localhost (executor driver) (1/1)
24/02/06 22:37:12 INFO scheduler.DAGScheduler: ResultStage 0 (collect at /home/cloudera/Desktop/filter.py:4) finished in 0.689 s
24/02/06 22:37:12 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 0.0, whose tasks have all completed, from pool
24/02/06 22:37:12 INFO scheduler.DAGScheduler: Job 0 finished: collect at /home/cloudera/Desktop/filter.py:4, took 1.675568 s
[2, 4, 6]
24/02/06 22:37:12 INFO spark.SparkContext: Invoking stop() from shutdown hook
24/02/06 22:37:12 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/metrics/json,null}
24/02/06 22:37:12 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/stages/stage/kill,null}
24/02/06 22:37:12 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/api,null}
24/02/06 22:37:12 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/,null}
24/02/06 22:37:12 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/static,null}
24/02/06 22:37:12 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/executors/threadDump/json,null}
24/02/06 22:37:12 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/executors/threadDump,null}
24/02/06 22:37:12 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/executors/json,null}
24/02/06 22:37:12 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/environment/json,null}
24/02/06 22:37:12 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/environment,null}
24/02/06 22:37:12 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/storage/rdd/json,null}
24/02/06 22:37:12 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/storage/rdd,null}
24/02/06 22:37:12 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/storage/json,null}
24/02/06 22:37:12 INFO handler.ContextHandler: stopped o.s.j.s.SampleContextHandler{/storage,null}
24/02/06 22:37:12 INFO handler.ContextHandler: stopped o.s.j.s.Handler{/stages/pool/json,null}
24/02/06 22:37:12 INFO handler.ContextHandler: stopped o.s.j.s.Handler{/stages/pool,null}
[cloudera@quickstart:~] cloudera@quickstart:~ [filter.py (~/Desktop) ...]
```

3. union()

```

union_inp = sc.parallelize([2,4,5,6,7,8,9])
union_rdd_1 = union_inp.filter(lambda x: x % 2 == 0)
union_rdd_2 = union_inp.filter(lambda x: x % 3 == 0)
print(union_rdd_1.union(union_rdd_2).collect())

```

```

union.py (~/Desktop) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
filter.py union.py
from pyspark import SparkContext
sc = SparkContext.getOrCreate()
union_inp = sc.parallelize([2,4,5,6,7,8,9])
union_rdd_1 = union_inp.filter(lambda x: x % 2 == 0)
union_rdd_2 = union_inp.filter(lambda x: x % 3 == 0)
print(union_rdd_1.union(union_rdd_2).collect())

```

```
[cloudera@quickstart SimpleApp.py]$ /usr/lib/spark/bin/spark-submit /home/cloudera/Desktop/union.py
```

OUTPUT:

```

24/02/06 22:45:44 INFO storage.MemoryStore: Block broadcast_0_piece0 stored as bytes in memory (estimated size 2.7 KB, free 534.5 MB)
24/02/06 22:45:44 INFO storage.BlockManagerInfo: Added broadcast_0_piece0 in memory on localhost:46687 (size: 2.7 KB, free: 534.5 MB)
24/02/06 22:45:44 INFO spark.SparkContext: Created broadcast 0 from broadcast at DAGScheduler.scala:1004
24/02/06 22:45:44 INFO scheduler.DAGScheduler: Submitting 2 missing tasks from ResultStage 0 (UnionRDD[3] at union at NativeMethodAccessorImpl.java:-2) (first 15 tasks are for partitions Vector(0, 1))
24/02/06 22:45:44 INFO scheduler.TaskSchedulerImpl: Adding task set 0.0 with 2 tasks
24/02/06 22:45:44 INFO scheduler.TaskSetManager: Starting task 0.0 in stage 0.0 (TID 0, localhost, executor driver, partition 0, PROCESS_LOCAL, 2251 bytes)
24/02/06 22:45:45 INFO executor.Executor: Running task 0.0 in stage 0.0 (TID 0)
24/02/06 22:45:45 INFO util.Utils: Fetching file:/home/cloudera/Desktop/union.py with timestamp 1707288342593
24/02/06 22:45:45 INFO util.Utils: /home/cloudera/Desktop/union.py has been previously copied to /tmp/spark-fc79ace6-356d-4099-aecf-8fb28bc2ecaa/userFiles-af58c9ae-1e3d-4e94-bf34-982cdbe5e11e/union.py
24/02/06 22:45:45 INFO python.PythonRunner: Times: total = 353, boot = 339, init = 14, finish = 0
24/02/06 22:45:45 INFO executor.Executor: Finished task 0.0 in stage 0.0 (TID 0). 1036 bytes result sent to driver
24/02/06 22:45:45 INFO scheduler.TaskSetManager: Starting task 1.0 in stage 0.0 (TID 1, localhost, executor driver, partition 1, PROCESS_LOCAL, 2251 bytes)
24/02/06 22:45:45 INFO executor.Executor: Running task 1.0 in stage 0.0 (TID 1)
24/02/06 22:45:45 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 0.0 (TID 0) in 618 ms on localhost (executor driver) (1/2)
24/02/06 22:45:45 INFO python.PythonRunner: Times: total = 185, boot = -64, init = 249, finish = 0
24/02/06 22:45:45 INFO executor.Executor: Finished task 1.0 in stage 0.0 (TID 1). 1014 bytes result sent to driver
24/02/06 22:45:45 INFO scheduler.DAGScheduler: ResultStage 0 (collect at /home/cloudera/Desktop/union.py:6) finished in 0.915 s
24/02/06 22:45:45 INFO scheduler.TaskSetManager: Finished task 1.0 in stage 0.0 (TID 1) in 283 ms on localhost (executor driver) (2/2)
24/02/06 22:45:45 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 0.0, whose tasks have all completed, from pool
24/02/06 22:45:45 INFO scheduler.DAGScheduler: Job 0 finished: collect at /home/cloudera/Desktop/union.py:6, took 1.738610 s
[2, 4, 6, 8, 6, 9]
24/02/06 22:45:45 INFO spark.SparkContext: Invoking stop() from shutdown hook
24/02/06 22:45:45 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/metrics/json,null}
24/02/06 22:45:45 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/stages/stage/kill,null}
24/02/06 22:45:45 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/api,null}
24/02/06 22:45:45 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/null}
24/02/06 22:45:45 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/static,null}
24/02/06 22:45:45 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/executors/threadDump/json,null}
24/02/06 22:45:45 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/executors/threadDump,null}
24/02/06 22:45:45 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/executors/json,null}
24/02/06 22:45:45 INFO cloudera@quickstart:~/SimpleApp.py o.s.j.s.ServletContextHandler{/executors,null}

```

4. flatMap()

```

flatmap_rdd = sc.parallelize(["Hey there", "This is PySpark RDD Transformations"])

(flatmap_rdd.flatMap(lambda x: x.split(" ")).collect())

```

```

flatmap.py (~/Desktop) - gedit
File Edit View Search Tools Documents Help
Open Save Undo | 
flatmap.py X
from pyspark import SparkContext
sc = SparkContext.getOrCreate()
flatmap_rdd = sc.parallelize(["Hey there", "This is PySpark RDD Transformations"])
(flatmap_rdd.flatMap(lambda x: x.split(" ")).collect())

[cloudera@quickstart SimpleApp.py]$ /usr/lib/spark/bin/spark-submit /home/cloudera/Desktop/flatmap.py

```

OUTPUT:

5. reduceByKey()

```
marks_rdd = sc.parallelize([('Rahul', 25), ('Swati', 26), ('Shreya', 22), ('Abhay', 29),
('Rohan', 22), ('Rahul', 23), ('Swati', 19), ('Shreya', 28), ('Abhay', 26), ('Rohan', 22)])

print(marks_rdd.reduceByKey(lambda x, y: x + y).collect())
```

The screenshot shows a window titled "reducebykey.py (~/Desktop) - gedit". The menu bar includes File, Edit, View, Search, Tools, Documents, and Help. The toolbar contains icons for Open, Save, Undo, and Redo. The code in the editor is:

```
from pyspark import SparkContext
sc = SparkContext.getOrCreate()
marks_rdd = sc.parallelize([('Rahul', 25), ('Swati', 26), ('Shreya', 22), ('Abhay', 29),
('Rohan', 22), ('Rahul', 23), ('Swati', 19), ('Shreya', 28), ('Abhay', 26), ('Rohan', 22)])
print(marks_rdd.reduceByKey(lambda x, y: x + y).collect())
```

```
[cloudera@quickstart SimpleApp.py]$ /usr/lib/spark/bin/spark-submit /home/cloudera/Desktop/reducebykey.py
```

OUTPUT:

The terminal window title is "cloudera@quickstart:~/SimpleApp.py". The log output is:

```
Tue Feb 6, 11:01 PM cloudera
File Machine View Input Devices Help
File Applications Places System
cloudera@quickstart:~/SimpleApp.py
File Edit View Search Terminal Help
24/02/06 22:58:54 INFO storage.BlockManagerInfo: Added broadcast_1_piece0 in memory on localhost:58012 (size: 3.2 KB, free: 534.5 MB)
24/02/06 22:58:54 INFO spark.SparkContext: Created broadcast 1 from broadcast at DAGScheduler.scala:1004
24/02/06 22:58:54 INFO scheduler.DAGScheduler: Submitting 1 missing tasks from ResultStage 1 (PythonRDD[5] at collect at /home/cloudera/Desktop/reducebykey.py:4) (first 15 tasks are for partitions Vector(0))
24/02/06 22:58:54 INFO scheduler.TaskSchedulerImpl: Adding task set 1.0 with 1 tasks
24/02/06 22:58:54 INFO scheduler.TaskSetManager: Starting task 0.0 in stage 1.0 (TID 1, localhost, executor driver, partition 0, NODE_LOCAL, 1946 bytes)
24/02/06 22:58:54 INFO executor.Executor: Running task 0.0 in stage 1.0 (TID 1)
24/02/06 22:58:54 INFO storage.ShuffleBlockFetcherIterator: Getting 1 non-empty blocks out of 1 blocks
24/02/06 22:58:54 INFO storage.ShuffleBlockFetcherIterator: Started 0 remote fetches in 13 ms
24/02/06 22:58:54 INFO python.PythonRunner: Times: total = 44, boot = -400, init = 443, finish = 1
24/02/06 22:58:54 INFO executor.Executor: Finished task 0.0 in stage 1.0 (TID 1). 1340 bytes result sent to driver
24/02/06 22:58:54 INFO scheduler.DAGScheduler: ResultStage 1 (collect at /home/cloudera/Desktop/reducebykey.py:4) finished in 0.180 s
24/02/06 22:58:54 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 1.0 (TID 1) in 180 ms on localhost (executor driver) (1/1)
24/02/06 22:58:54 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 1.0, whose tasks have all completed, from pool
24/02/06 22:58:54 INFO scheduler.DAGScheduler: Job 0 finished: collect at /home/cloudera/Desktop/reducebykey.py:4, took 1.968394 s
[('Abhay', 55), ('Rohan', 44), ('Swati', 45), ('Shreya', 50), ('Rahul', 48)]
24/02/06 22:58:55 INFO spark.SparkContext: Invoking stop() From shutdown hook
24/02/06 22:58:55 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/metrics/json,null}
24/02/06 22:58:55 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/stages/stage/kill,null}
24/02/06 22:58:55 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/api,null}
24/02/06 22:58:55 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/null}
24/02/06 22:58:55 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/static,null}
24/02/06 22:58:55 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/executors/threadDump/json,null}
24/02/06 22:58:55 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/executors/threadDump,null}
24/02/06 22:58:55 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/executors/json,null}
24/02/06 22:58:55 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/environment/json,null}
24/02/06 22:58:55 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/environment,null}
24/02/06 22:58:55 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/storage/rdd/json,null}
24/02/06 22:58:55 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/storage/rdd,null}
24/02/06 22:58:55 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/storage/json,null}
24/02/06 22:58:55 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/storage,null}
24/02/06 22:58:55 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/stages/pool/json,null}
```

6. sortByKey()

```
marks_rdd = sc.parallelize([('Rahul', 25), ('Swati', 26), ('Shreya', 22), ('Abhay', 29),
('Rohan', 22), ('Rahul', 23), ('Swati', 19), ('Shreya', 28), ('Abhay', 26), ('Rohan', 22)])
print(marks_rdd.sortByKey('ascending').collect())
```

```
sortbykey.py (~/Desktop) - gedit
File Edit View Search Tools Documents Help
Open Save Undo Redo Cut Copy Paste Find Select All
sortbykey.py X
from pyspark import SparkContext
sc = SparkContext.getOrCreate()
marks_rdd = sc.parallelize([('Rahul', 25), ('Swati', 26), ('Shreya', 22), ('Abhay', 29),
('Rohan', 22), ('Rahul', 23), ('Swati', 19), ('Shreya', 28), ('Abhay', 26), ('Rohan', 22)])
print(marks_rdd.sortByKey('ascending').collect())
[cloudera@quickstart SimpleApp.py]$ /usr/lib/spark/bin/spark-submit /home/cloudera/Desktop/sortbykey.py
```

OUTPUT:

```
cloudera-quickstart-vm-5.13.0-0-virtualbox [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System Terminal cloudera@quickstart:~/SimpleApp.py
Tue Feb 6, 11:06 PM cloudera
File Edit View Search Terminal Help
24/02/06 23:05:15 INFO storage.MemoryStore: Block broadcast_0 stored as values in memory (estimated size 3.7 KB, free 534.5 MB)
24/02/06 23:05:15 INFO storage.MemoryStore: Block broadcast_0_piece0 stored as bytes in memory (estimated size 2.4 KB, free 534.5 MB)
24/02/06 23:05:15 INFO storage.BlockManagerInfo: Added broadcast_0_piece0 in memory on localhost:38789 (size: 2.4 KB, free: 534.5 MB)
24/02/06 23:05:15 INFO spark.SparkContext: Created broadcast 0 from broadcast at DAGScheduler.scala:1004
24/02/06 23:05:15 INFO scheduler.DAGScheduler: Submitting 1 missing tasks from ResultStage 0 (PythonRDD[1] at collect at /home/cloudera/Desktop/sortbykey.py:4) (first 1 tasks are for partitions Vector(0))
24/02/06 23:05:15 INFO scheduler.TaskSchedulerImpl: Adding task set 0.0 with 1 tasks
24/02/06 23:05:15 INFO scheduler.TaskSetManager: Starting task 0.0 in stage 0.0 (TID 0, localhost, executor driver, partition 0, PROCESS_LOCAL, 2238 bytes)
24/02/06 23:05:15 INFO executor.Executor: Running task 0.0 in stage 0.0 (TID 0)
24/02/06 23:05:15 INFO executor.Executor: Fetching file:/home/cloudera/Desktop/sortbykey.py with timestamp 1707289512723
24/02/06 23:05:15 INFO util.Utils: /home/cloudera/Desktop/sortbykey.py has been previously copied to /tmp/spark-cc4610e-18fb-4e17-9a74-2c75b22e7a69/userFiles-3e36d5aa-1cb1-465b-aae-99b8be1127cf/sortbykey.py
24/02/06 23:05:15 INFO python.PythonRunner: Times: total = 358, boot = 327, init = 10, finish = 21
24/02/06 23:05:15 INFO executor.Executor: Finished task 0.0 in stage 0.0 (TID 0). 1173 bytes sent to driver
24/02/06 23:05:15 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 0.0 (TID 0) in 618 ms on localhost (executor driver) (1/1)
24/02/06 23:05:15 INFO scheduler.DAGScheduler: ResultStage 0 (collect at /home/cloudera/Desktop/sortbykey.py:4) finished in 0.645 s
24/02/06 23:05:15 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 0.0, whose tasks have all completed, from pool
24/02/06 23:05:15 INFO scheduler.DAGScheduler: Job 0 finished: collect at /home/cloudera/Desktop/sortbykey.py:4, took 1.641803 s
[('Abhay', 29), ('Abhay', 26), ('Rahul', 25), ('Rahul', 23), ('Rohan', 22), ('Shreya', 22), ('Shreya', 28), ('Swati', 26), ('Swati', 19)]
24/02/06 23:05:16 INFO spark.SparkContext: Invoking stop() from shutdown hook
24/02/06 23:05:16 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/metrics/json,null}
24/02/06 23:05:16 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/stages/stage/kill,null}
24/02/06 23:05:16 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/api,null}
24/02/06 23:05:16 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/null}
24/02/06 23:05:16 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/static,null}
24/02/06 23:05:16 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/executors/threadDump/json,null}
24/02/06 23:05:16 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/executors/threadDump,null}
24/02/06 23:05:16 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/executors/json,null}
24/02/06 23:05:16 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/executors,null}
24/02/06 23:05:16 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/environment/json,null}
24/02/06 23:05:16 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/environment,null}
24/02/06 23:05:16 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/storage/rdd/json,null}
24/02/06 23:05:16 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/storage/rdd,null}
[cloudera@quickstart:~] cloudera@quickstart:~ [sortbykey.py (~/Desktop)]
```

7. groupByKey()

```
marks_rdd = sc.parallelize([('Rahul', 25), ('Swati', 26), ('Shreya', 22), ('Abhay', 29),
                           ('Rohan', 22), ('Rahul', 23), ('Swati', 19), ('Shreya', 28), ('Abhay', 26), ('Rohan', 22)])
```

```
dict_rdd = marks_rdd.groupByKey().collect()
```

```
for key, value in dict_rdd:
```

```
    print(key, list(value))
```

```
group.py (~/Desktop) - gedit
File Edit View Search Tools Documents Help
Open Save Undo Redo Cut Copy Paste Find Replace
group.py X
from pyspark import SparkContext
sc = SparkContext.getOrCreate()
marks_rdd = sc.parallelize([('Rahul', 25), ('Swati', 26), ('Shreya', 22), ('Abhay', 29),
                           ('Rohan', 22), ('Rahul', 23), ('Swati', 19), ('Shreya', 28), ('Abhay', 26), ('Rohan', 22)])
dict_rdd = marks_rdd.groupByKey().collect()
for key, value in dict_rdd:
    print(key, list(value))
```

```
[cloudera@quickstart SimpleApp.py]$ /usr/lib/spark/bin/spark-submit /home/cloudera/Desktop/group.py
[cloudera@quickstart ~]$
```

Output:

```
cloudera-quickstart-vm-5.13.0-0-virtualbox [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System
cloudera@quickstart:~/SimpleApp.py
File Edit View Terminal Help
Tue Feb 6, 11:10:10 PM cloudera
cloudera@quickstart:~/SimpleApp.py
24/02/06 23:10:12 INFO scheduler.TaskSchedulerImpl: Adding task set 1.0 with 1 tasks
24/02/06 23:10:12 INFO scheduler.TaskSetManager: Starting task 0.0 in stage 1.0 (TID 1, localhost, executor driver, partition 0, NODE_LOCAL, 1940 bytes)
24/02/06 23:10:12 INFO executor.Executor: Running task 0.0 in stage 1.0 (TID 1)
24/02/06 23:10:12 INFO storage.ShuffleBlockFetcherIterator: Getting 1 non-empty blocks out of 1 blocks
24/02/06 23:10:12 INFO storage.ShuffleBlockFetcherIterator: Started 0 remote fetches in 12 ms
24/02/06 23:10:12 INFO python.PythonRDD: Times: total = 43, begin = 489, in = 532, finish = 0
24/02/06 23:10:12 INFO scheduler.ResultStage: ResultStage 1 (collect at /home/cloudera/Desktop/group.py:4) finished in 0.194 s
24/02/06 23:10:12 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 1.0 (TID 1) in 194 ms on localhost (executor driver) (1/1)
24/02/06 23:10:12 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 1.0, whose tasks have all completed, from pool
24/02/06 23:10:12 INFO scheduler.DAGScheduler: Job 0 finished: collect at /home/cloudera/Desktop/group.py:4, took 2.191156 s
(['Abhay', [29, 26])
('Rohan', [22, 22])
('Swati', [26, 19])
('Shreya', [22, 28])
('Rahul', [25, 23])
24/02/06 23:10:12 INFO spark.SparkContext: Invoking stop() from shutdown hook
24/02/06 23:10:12 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/metrics/json,null}
24/02/06 23:10:12 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/stages/stage/kill,null}
24/02/06 23:10:12 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/api,null}
24/02/06 23:10:12 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/null}
24/02/06 23:10:12 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/static,null}
24/02/06 23:10:12 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/executors/threadDump/json,null}
24/02/06 23:10:12 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/executors/threadDump,null}
24/02/06 23:10:12 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/executors/json,null}
24/02/06 23:10:12 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/executors,null}
24/02/06 23:10:12 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/environment/json,null}
24/02/06 23:10:12 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/environment,null}
24/02/06 23:10:12 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/storage/rdd/json,null}
24/02/06 23:10:12 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/storage/rdd,null}
24/02/06 23:10:12 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/storage/json,null}
24/02/06 23:10:12 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/storage,null}
24/02/06 23:10:12 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/stages/pool/json,null}
```

Example 3: Create a Python application to create a Dataframe in Python Pandas.

```
import pandas as pd
data = [['Scott', 50], ['Jeff', 45], ['Thomas', 54], ['Ann', 34]]
```

Cmd 1

```
1 #Example 3: Create a Python application to create a Dataframe in Python Pandas.
2 import pandas as pd
3 data = [['Scott', 50], ['Jeff', 45], ['Thomas', 54], ['Ann', 34]]
```

Command took 5.27 seconds -- by ridadange2408@gmail.com at 19/02/2024, 19:18:47 on My Cluster

Create the pandas DataFrame

```
pandasDF = pd.DataFrame(data, columns = ['Name', 'Age'])
# print dataframe.
print(pandasDF)
```

Cmd 2

```
1 # Create the pandas DataFrame
2 pandasDF = pd.DataFrame(data, columns = ['Name', 'Age'])
3 # print dataframe.
4 print(pandasDF)
```

Name	Age
0 Scott	50
1 Jeff	45
2 Thomas	54
3 Ann	34

Command took 0.16 seconds -- by ridadange2408@gmail.com at 19/02/2024, 19:27:19 on My Cluster

Example 4: Creating a dataframe from Spark RDD

1. Create a directory list

```
data = [{"Category": 'A', "ID": 1, "Value": 121.44, "Truth": True},
        {"Category": 'B', "ID": 2, "Value": 300.01, "Truth": False},
        {"Category": 'C', "ID": 3, "Value": 10.99, "Truth": None},
        {"Category": 'E', "ID": 4, "Value": 33.87, "Truth": True}
    ]
```

2. Import and create a SparkContext:

```
from pyspark import SparkContext, SparkConf
conf = SparkConf().setAppName("projectName").setMaster("local[*]")
sc = SparkContext.getOrCreate(conf)
```

3. Generate an RDD from the created data. Check the type to confirm the object is an RDD:

```
rdd = sc.parallelize(data)
type(rdd)
```

4. Call the toDF() method on the RDD to create the DataFrame. Test the object type to confirm:

```
df = rdd.toDF()
type(df)
```

Cmd 3

```

1  data = [{"Category": "A", "ID": 1, "Value": 121.44, "Truth": True},
2    {"Category": "B", "ID": 2, "Value": 300.01, "Truth": False},
3    {"Category": "C", "ID": 3, "Value": 10.99, "Truth": None},
4    {"Category": "E", "ID": 4, "Value": 33.87, "Truth": True}
5  ]
6
7  #Import and create a SparkContext:
8  from pyspark import SparkContext, SparkConf
9  conf = SparkConf().setAppName("projectName").setMaster("local[*]")
10 sc = SparkContext.getOrCreate(conf)
11
12 #Generate an RDD from the created data. Check the type to confirm the object is an RDD:
13 rdd = sc.parallelize(data)
14 type(rdd)
15

```

Out[5]: pyspark.rdd.RDD

Command took 0.72 seconds -- by ridadange2408@gmail.com at 19/02/2024, 19:42:46 on My Cluster

Cmd 4

```

1  #Call the toDF() method on the RDD to create the DataFrame. Test the object type to confirm:
2  df = rdd.toDF()
3  type(df)
4

```

▶ (2) Spark Jobs

- ▼ df: pyspark.sql.dataframe.DataFrame
 - Category: string
 - ID: long
 - Truth: boolean
 - Value: double

Out[6]: pyspark.sql.dataframe.DataFrame

Command took 3.64 seconds -- by ridadange2408@gmail.com at 19/02/2024, 19:43:18 on My Cluster

Practical 8: Spark Dataframes and SQL

Example 1: Create a Dataframe in Python Pandas.

```
import pandas as pd
data = [['Scott', 50], ['Jeff', 45], ['Thomas', 54], ['Ann', 34]]
```

The screenshot shows a Jupyter Notebook cell. At the top, there is a button labeled "Generate" with the sub-label "10 random numbers using numpy". Below the button, the code is displayed:

```
[1] ⏪ import pandas as pd
    data = [['Scott', 50], ['Jeff', 45], ['Thomas', 54], ['Ann', 34]] |
```

At the bottom of the cell, there is a placeholder text: "[] Start coding or generate with AI.".

```
# Create the pandas DataFrame
```

```
pandasDF = pd.DataFrame(data, columns = ['Name', 'Age'])
```

The screenshot shows a Jupyter Notebook cell. At the top, there is a button labeled "Generate" with the sub-label "10 random numbers using numpy". Below the button, the code is displayed:

```
[1] ✓ [1] ⏪ import pandas as pd
    data = [['Scott', 50], ['Jeff', 45], ['Thomas', 54], ['Ann', 34]] |
```

Below the first code block, another code block is shown:

```
[2] ✓ [2] ⏪ # Create the pandas DataFrame
    pandasDF = pd.DataFrame(data, columns = ['Name', 'Age']) |
```

```
# print dataframe.
```

```
print(pandasDF)
```

The screenshot shows a Jupyter Notebook cell. At the top, there is a button labeled "Generate" with the sub-label "10 random numbers using numpy". Below the button, the code is displayed:

```
[1] [1] ⏪ import pandas as pd
    data = [['Scott', 50], ['Jeff', 45], ['Thomas', 54], ['Ann', 34]] |
```

Below the first code block, another code block is shown:

```
[2] [2] ⏪ # Create the pandas DataFrame
    pandasDF = pd.DataFrame(data, columns = ['Name', 'Age']) |
```

Below the second code block, the output of the "print(pandasDF)" command is shown:

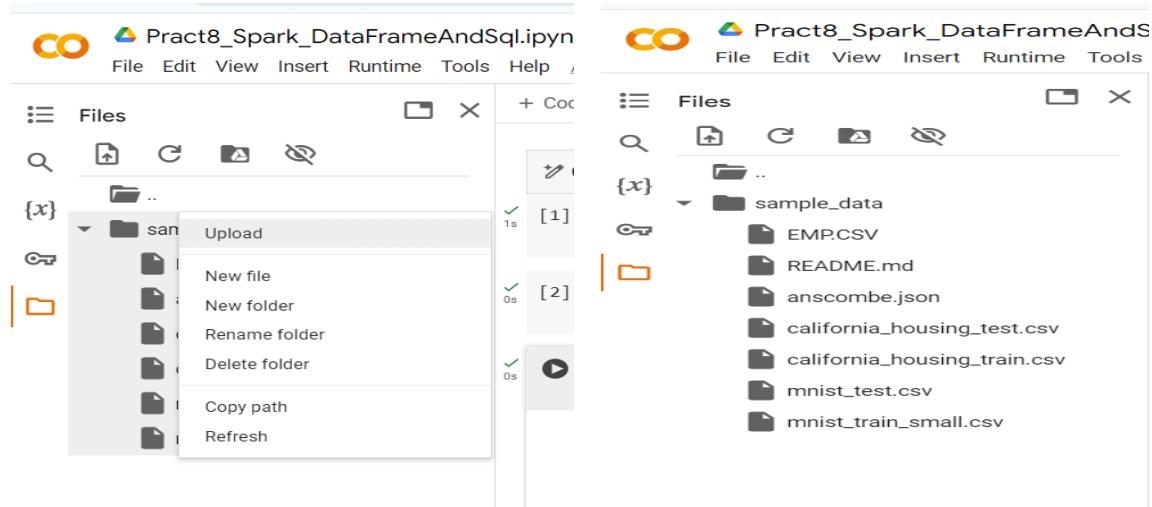
```
▶ [3] # print dataframe.
    print(pandasDF) |
```

	Name	Age
0	Scott	50
1	Jeff	45
2	Thomas	54
3	Ann	34

Example 2:

1. Load emp.csv in the dataframe.

Ref: <https://drive.google.com/file/d/1Tc2SFSMq-Na7rnLgH2kBUD81XReDqsHv6/view?usp=sharing>



```
# spark is an existing SparkSession
df =
spark.read.option("header",True).csv("/content/sample_data/EMP.CSV")
```

```
[8] ! pip install pyspark
Collecting pyspark
  Downloading pyspark-3.5.1.tar.gz (317.0 MB)
    317.0/317.0 MB 5.1 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: py4j==0.10.9.7 in /usr/local/lib/python3.10/dist-packages (from pyspark) (0.10.9.7)
Building wheels for collected packages: pyspark
  Building wheel for pyspark (setup.py) ... done
    Created wheel for pyspark: filename=pyspark-3.5.1-py2.py3-none-any.whl size=317488491 sha256=957a6a4c40c82a781e8f6befd018a56bbf05ef9d4ab479a9ac1f6c5d8c324ad0
    Stored in directory: /root/.cache/pip/wheels/80/1d/60/2c256ed38ddce2fdd93be545214a63e02fb8d74fb0b7f3a6
Successfully built pyspark
Installing collected packages: pyspark
Successfully installed pyspark-3.5.1

[9] import pyspark
      from pyspark.sql import SparkSession

      spark = SparkSession.builder.getOrCreate()

# spark is an existing SparkSession
df = spark.read.option("header",True).csv("/content/sample_data/EMP.CSV")
```

1. Displays the content of the DataFrame to stdout

df.show()

```
✓ [10] # spark is an existing SparkSession
6s df = spark.read.option("header",True).csv("/content/sample_data/EMP.CSV")

✓ 1s df.show()

+-----+-----+-----+-----+-----+-----+-----+-----+
|EMPLOYEE_ID|FIRST_NAME|LAST_NAME|EMAIL|PHONE_NUMBER|HIRE_DATE|JOB_ID|SALARY|COMMISSION_PCT|MANAGER_ID|DEPARTMENT_ID|
+-----+-----+-----+-----+-----+-----+-----+-----+
| 198| Donald| OConnell| DOCONNEL| 650.507.9833| 21-JUN-07| SH_CLERK| 2600| - | 124| 50|
| 199| Douglas| Grant| DGRANT| 650.507.9844| 13-JAN-08| SH_CLERK| 2600| - | 124| 50|
| 200| Jennifer| Whalen| JWAHLEN| 515.123.4444| 17-SEP-03| AD_ASST| 4400| - | 101| 10|
| 201| Michael| Hartstein| MHARTSTE| 515.123.5555| 17-FEB-04| MK_MAN| 13000| - | 100| 20|
| 202| Pat| Fay| PFAY| 603.123.6666| 17-AUG-05| MK_REP| 6000| - | 201| 20|
| 203| Susan| Mavris| SMAVRIS| 515.123.7777| 07-JUN-02| HR REP| 6500| - | 101| 40|
| 204| Hermann| Baer| HBAER| 515.123.8888| 07-JUN-02| PR REP| 10000| - | 101| 70|
| 205| Shelley| Higgins| SHIGGINS| 515.123.8080| 07-JUN-02| AC_MGR| 12008| - | 101| 110|
| 206| William| Gietz| WGIETZ| 515.123.8181| 07-JUN-02| AC_ACCOUNT| 8300| - | 205| 110|
| 100| Steven| King| SKING| 515.123.4567| 17-JUN-03| AD PRES| 24000| - | - | 90|
| 101| Neena| Kochhar| NKOCHHAR| 515.123.4568| 21-SEP-05| AD_VP| 17000| - | 100| 90|
| 102| Lex| De Haan| LDEHAAN| 515.123.4569| 13-JAN-01| AD_VP| 17000| - | 100| 90|
| 103| Alexander| Hunold| AHUNOLD| 590.423.4567| 03-JAN-06| IT_PROG| 9000| - | 102| 60|
| 104| Bruce| Ernst| BERNST| 590.423.4568| 21-MAY-07| IT_PROG| 6000| - | 103| 60|
| 105| David| Austin| DAUSTIN| 590.423.4569| 25-JUN-05| IT_PROG| 4800| - | 103| 60|
| 106| Valli| Pataballa| VPATABAL| 590.423.4560| 05-FEB-06| IT_PROG| 4800| - | 103| 60|
| 107| Diana| Lorentz| DLORENTZ| 590.423.5567| 07-FEB-07| IT_PROG| 4200| - | 103| 60|
| 108| Nancy| Greenberg| NGREENBE| 515.124.4569| 17-AUG-02| FI_MGR| 12008| - | 101| 100|
| 109| Daniel| Faviet| DFAVIET| 515.124.4169| 16-AUG-02| FI_ACCOUNT| 9000| - | 108| 100|
| 110| John| Chen| JCHEN| 515.124.4269| 28-SEP-05| FI_ACCOUNT| 8200| - | 108| 100|
+-----+-----+-----+-----+-----+-----+-----+-----+
only showing top 20 rows
```

2. Display the dataframe schema.

df.printSchema()

```
✓ 0s df.printSchema()

root
|-- EMPLOYEE_ID: string (nullable = true)
|-- FIRST_NAME: string (nullable = true)
|-- LAST_NAME: string (nullable = true)
|-- EMAIL: string (nullable = true)
|-- PHONE_NUMBER: string (nullable = true)
|-- HIRE_DATE: string (nullable = true)
|-- JOB_ID: string (nullable = true)
|-- SALARY: string (nullable = true)
|-- COMMISSION_PCT: string (nullable = true)
|-- MANAGER_ID: string (nullable = true)
|-- DEPARTMENT_ID: string (nullable = true)
```

3. Check the first five or last few records of the dataframe.

df.head()

```
✓ 0s [14] df.head()

Row(EMPLOYEE_ID='198', FIRST_NAME='Donald', LAST_NAME='OConnell', EMAIL='DOCONNEL', PHONE_NUMBER='650.507.9833', HIRE_DATE='21-JUN-07', JOB_ID='SH_CLERK', SALARY='2600',
COMMISSION_PCT=' - ', MANAGER_ID='124', DEPARTMENT_ID='50')
```

df.tail(5)

```
[✓] [15] df.tail(5)

[Row(EMPLOYEE_ID='136', FIRST_NAME='Hazel', LAST_NAME='Phil tanker', EMAIL='HPHILITAN', PHONE_NUMBER='650.127.1634', HIRE_DATE='06-FEB-08', JOB_ID='ST_CLERK', SALARY='2200', COMMISSION_PCT=' - ', MANAGER_ID='122', DEPARTMENT_ID='50'),
 Row(EMPLOYEE_ID='137', FIRST_NAME='Renske', LAST_NAME='Ladwig', EMAIL='RLADWIG', PHONE_NUMBER='650.121.1234', HIRE_DATE='14-JUL-03', JOB_ID='ST_CLERK', SALARY='3600', COMMISSION_PCT=' - ', MANAGER_ID='123', DEPARTMENT_ID='50'),
 Row(EMPLOYEE_ID='138', FIRST_NAME='Stephen', LAST_NAME='Stiles', EMAIL='SSTILES', PHONE_NUMBER='650.121.2034', HIRE_DATE='26-OCT-05', JOB_ID='ST_CLERK', SALARY='3200', COMMISSION_PCT=' - ', MANAGER_ID='123', DEPARTMENT_ID='50'),
 Row(EMPLOYEE_ID='139', FIRST_NAME='John', LAST_NAME='Seo', EMAIL='JSEO', PHONE_NUMBER='650.121.2019', HIRE_DATE='12-FEB-06', JOB_ID='ST_CLERK', SALARY='2700', COMMISSION_PCT=' - ', MANAGER_ID='123', DEPARTMENT_ID='50'),
 Row(EMPLOYEE_ID='140', FIRST_NAME='Joshua', LAST_NAME='Patel', EMAIL='JPATEL', PHONE_NUMBER='650.121.1834', HIRE_DATE='06-APR-06', JOB_ID='ST_CLERK', SALARY='2500', COMMISSION_PCT=' - ', MANAGER_ID='123', DEPARTMENT_ID='50')]
```

df.head(2)

```
[✓] [16] df.head(2)

[Row(EMPLOYEE_ID='198', FIRST_NAME='Donald', LAST_NAME='OConnell', EMAIL='DOCONNEL', PHONE_NUMBER='650.507.9833', HIRE_DATE='21-JUN-07', JOB_ID='SH_CLERK', SALARY='2600', COMMISSION_PCT=' - ', MANAGER_ID='124', DEPARTMENT_ID='50'),
 Row(EMPLOYEE_ID='199', FIRST_NAME='Douglas', LAST_NAME='Grant', EMAIL='DGRANT', PHONE_NUMBER='650.507.9844', HIRE_DATE='13-JAN-08', JOB_ID='SH_CLERK', SALARY='2600', COMMISSION_PCT=' - ', MANAGER_ID='124', DEPARTMENT_ID='50')]
```

4. Display employees list

df. select("FIRST_NAME", "LAST_NAME")

```
[✓] [18] df.select("FIRST_NAME", "LAST_NAME")
```

DataFrame[FIRST_NAME: string, LAST_NAME: string]

df. select("FIRST_NAME","LAST_NAME").show(2)

```
[✓] [19] df.select("FIRST_NAME", "LAST_NAME").show(2)
```

FIRST_NAME	LAST_NAME
Donald	OConnell
Douglas	Grant

only showing top 2 rows

5. Increment the salary of each employee by 100 rs.

```
df.select(df['FIRST_NAME'], df['SALARY'] + 100).show()
```

```
df.select(df['FIRST_NAME'], df['SALARY'] + 100).show()

+-----+-----+
|FIRST_NAME|SALARY + 100|
+-----+-----+
|    Donald| 2700.0|
|   Douglas| 2700.0|
| Jennifer| 4500.0|
|   Michael|13100.0|
|     Pat| 6100.0|
|    Susan| 6600.0|
| Hermann|10100.0|
| Shelley|12108.0|
|  William| 8400.0|
|   Steven|24100.0|
|    Neena| 17100.0|
|      Lex| 17100.0|
| Alexander| 9100.0|
|    Bruce| 6100.0|
|    David| 4900.0|
|    Valli| 4900.0|
|   Diana| 4300.0|
|   Nancy| 12108.0|
|   Daniel| 9100.0|
|     John| 8300.0|
+-----+-----+
only showing top 20 rows
```

6. Display the employees list having a salary more than 10000.

```
df.filter(df['SALARY'] > 21).show()
```

Page 85 | 107

7. Count the employees having the same designation.

```
df.groupBy("JOB_ID").count().show()
```

1s

df.groupBy("JOB_ID").count().show()

JOB_ID	count
FI_ACCOUNT	5
MK_MAN	1
IT_PROG	5
FI_MGR	1
AC_ACCOUNT	1
HR_REP	1
PU_CLERK	5
AC_MGR	1
PR_REP	1
ST_MAN	5
MK_REP	1
PU_MAN	1
SH_CLERK	2
AD_PRES	1
AD_ASST	1
ST_CLERK	16
AD_VP	2

8. Display the list of employees who have joined on the date 12-Dec-07 or 07-Jun-02

```
df.select("FIRST_NAME", "LAST_NAME").where(df["HIRE_DATE"] == "16-AUG-02").show()
```

0s

df.select("FIRST_NAME", "LAST_NAME").where(df["HIRE_DATE"] == "16-AUG-02").show()

FIRST_NAME	LAST_NAME
Daniel	Faviet

9. Display the ascending order list of employees who are working under the manager with manager id 124.

```
df.select("FIRST_NAME", "LAST_NAME").where(df["MANAGER_ID"] == "124").orderBy("FIRST_NAME", "LAST_NAME").show()
```

FIRST_NAME	LAST_NAME
Donald	O'Connell
Douglas	Grant

10. Display list of employees and their email ids in lowercase for the department 50.

```
df.select(F.lower("FIRST_NAME"), F.lower("LAST_NAME"), F.lower("EMAIL")).where(df["DEPARTMENT_ID"] == "50").show()
```

FIRST_NAME	LAST_NAME	EMAIL
donald	oconnell	doconnel
douglas	grant	dgrant
matthew	weiss	mweiss
adam	fripp	afripp
payam	kaufling	pkauflin
shanta	vollman	svollman
kevin	mourgos	kmourgos
julia	nayer	jnayer
irene	mikkilineni	imikkili
james	landry	jlandry
steven	markle	smarkle
laura	bissot	lbissot
mozhe	atkinson	matkinso
james	marlow	jammflow
tj	olson	tjolson
jason	mallin	jmallin
michael	rogers	mrrogers
ki	gee	kgee
hazel	philtanker	hphiltan
renske	ladwig	rladwig

only showing top 20 rows

11. Display the last 4 characters of PHONE_NUMBER of employees of manager with manager id108.

substring(str, pos, len)

```
df.filter(df["MANAGER_ID"] == 108).select(df["FIRST_NAME"], df["PHONE_NUMBER"]).withColumn("substring", F.substring("PHONE_NUMBER", 9, 4)).show()
```

FIRST_NAME	PHONE_NUMBER	substring
Daniel	515.124.4169	4169
John	515.124.4269	4269
Ismael	515.124.4369	4369
Jose Manuel	515.124.4469	4469
Luis	515.124.4567	4567

12. Get the joining day, month and year of all employees having a job as a FI_ACCOUNT.

dayofmonth()

month(col)

year(col)

```
df.select(F.dayofmonth(F.to_date("HIRE_DATE", "dd-MMM-yy"))
           .alias("JOINING_DAY"),
           F.month(F.to_date("HIRE_DATE", "dd-MMM-yy"))
           .alias("JOINING_MONTH"),
           F.year(F.to_date("HIRE_DATE", "dd-MMM-yy"))
           .alias("JOINING_YEAR"))
    .where(df["JOB_ID"] == "FI_ACCOUNT")
    .show()
```

JOINING_DAY	JOINING_MONTH	JOINING_YEAR
16	8	2002
28	9	2005
30	9	2005
7	3	2006
7	12	2007

13. Get the list of names of managers.

```
df.filter(df["MANAGER_ID"].isNotNull())
        .select("FIRST_NAME", "LAST_NAME")
        .show()
```

FIRST_NAME	LAST_NAME
Donald	OConnell
Douglas	Grant
Jennifer	Whalen
Michael	Hartstein
Pat	Fay
Susan	Mavris
Hermann	Baer
Shelley	Higgins
William	Gietz
Steven	King
Neena	Kochhar
Lex	De Haan
Alexander	Hunold
Bruce	Ernst
David	Austin
Valli	Patahalla

14. Who is manager of an employee having employee_id 107.

```
df.select("FIRST_NAME", "LAST_NAME").where(df["EMPLOYEE_ID"] ==
df.select("MANAGER_ID").where(df["EMPLOYEE_ID"] ==
"107").first()[0]).show()

✓ [95] df.select("FIRST_NAME", "LAST_NAME").where(df["EMPLOYEE_ID"] == df.select("MANAGER_ID").where(df["EMPLOYEE_ID"] == "107").first()[0]).show()

+-----+-----+
|FIRST_NAME|LAST_NAME|
+-----+-----+
| Alexander| Hunold|
+-----+-----+
```

15. Get the employee with the maximum salary.

```
df.select("FIRST_NAME", "LAST_NAME", "SALARY").where(df["SALARY"] ==
df.agg(F.max("SALARY"))).first()[0]).show()

✓ [86] df.select("FIRST_NAME", "LAST_NAME", "SALARY").where(df["SALARY"] == df.agg(F.max("SALARY")).first()[0]).show()

+-----+-----+-----+
|FIRST_NAME|LAST_NAME|SALARY|
+-----+-----+-----+
| Alexander| Hunold| 9000|
| Daniel| Faviet| 9000|
+-----+-----+-----+
```

16. Display the descending ordered list of employees who are working under the manager with manager id 101 and having salary more than 10000.

```
df.filter((df["MANAGER_ID"] == 101) & (df["SALARY"] >
10000)).sort(df["SALARY"].desc()).show()

✓ ⏪ df.filter((df["MANAGER_ID"] == 101) & (df["SALARY"] > 10000)).sort(df["SALARY"].desc()).show()

[1]: +-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|EMPLOYEE_ID|FIRST_NAME|LAST_NAME|EMAIL|PHONE_NUMBER|HIRE_DATE|JOB_ID|SALARY|COMMISSION_PCT|MANAGER_ID|DEPARTMENT_ID|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|     205| Shelley| Higgins|SHIGGINS|515.123.8080|07-JUN-02|AC_MGR| 12008|      -|    101|     110|
|     108| Nancy| Greenberg|NGREENBE|515.124.4569|17-AUG-02|FI_MGR| 12008|      -|    101|     100|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
```

17. Using spark.sql instead of df.select :

Display the list of employees who have joined on the date 12-Dec-07 or 07-Jun-02

```
spark.sql("""select "FIRST_NAME","LAST_NAME" where("df[HIRE_DATE]=='12-Dec-07")""").show()
df.createOrReplaceTempView("employees")
#Running the sql queries
result = spark.sql("""
SELECT * FROM employees
WHERE TO_DATE(HIRE_DATE, 'dd-MM-yy') = '2007-12-12' OR
TO_DATE(HIRE_DATE, 'dd-MM-yy') = '2002-06-07'
""")
#Display the result
result.show()
```

```
1 df.createOrReplaceTempView("employees")
2 # Run SQL query to select employees who have joined on the specified dates
3 result = spark.sql("""
4     SELECT * FROM employees
5     WHERE TO_DATE(HIRE_DATE, 'dd-MM-yy') = '2007-12-12' OR TO_DATE(HIRE_DATE, 'dd-MM-yy') = '2002-06-07'
6 """
7 # Display the result
8 result.show()
```

```
▶ (1) Spark Jobs
▶ [ ] result: pyspark.sql.dataframe.DataFrame = [EMPLOYEE_ID: string, FIRST_NAME: string ... 9 more fields]
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|EMPLOYEE_ID|FIRST_NAME|LAST_NAME|EMAIL|PHONE_NUMBER|HIRE_DATE|JOB_ID|SALARY|COMMISSION_PCT|MANAGER_ID|DEPARTMENT_ID|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 203| Susan| Mavris| SMAVRIS|515.123.7777|07-JUN-02| HR_REP| 6500|      - | 101|    40|
| 204| Hermann| Baer| HBAER|515.123.8888|07-JUN-02| PR_REP| 10000|      - | 101|    70|
| 205| Shelley| Higgins|SHIGGINS|515.123.8080|07-JUN-02| AC_MGR| 12008|      - | 101|   110|
| 206| William| Gietz| WGIETZ|515.123.8181|07-JUN-02| AC_ACCOUNT| 8300|      - | 205|   110|
| 135| Ki| Gee| KGEE|650.127.1734|12-DEC-07| ST_CLERK| 2400|      - | 122|    50|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
```

Example 2: Dataframes Joining

1. Create the data list and dataframe for employee and department data.

```
import pyspark
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("sparkbyexamples.com").getOrCreate()
```

```
✓ [116] import pyspark
      from pyspark.sql import SparkSession
      spark = SparkSession.builder.appName("sparkbyexamples.com").getOrCreate()
```

```
emp = [(1,"Smith",-1,"2018","10","M",3000),
        (2,"Rose",1,"2010","20","M",4000),
        (3,"Williams",1,"2010","10","M",1000),
        (4,"Jones",2,"2005","10","F",2000),
        (5,"Brown",2,"2010","40","", -1),
        (6,"Brown",2,"2010","50","", -1)
    ]
empColumns = ["emp_id", "name", "superior_emp_id", "year_joined",
              "emp_dept_id", "gender", "salary"]
```

```
empDF = spark.createDataFrame(data=emp, schema = empColumns)
```

```
✓ [117] emp = [(1,"Smith",-1,"2018","10","M",3000),
      (2,"Rose",1,"2010","20","M",4000),
      (3,"Williams",1,"2010","10","M",1000),
      (4,"Jones",2,"2005","10","F",2000),
      (5,"Brown",2,"2010","40","", -1),
      (6,"Brown",2,"2010","50","", -1)
    ]
empColumns = ["emp_id", "name", "superior_emp_id", "year_joined",
              "emp_dept_id", "gender", "salary"]

empDF = spark.createDataFrame(data=emp, schema = empColumns)
```

```
empDF.printSchema()
```

```
✓ [118] empDF.printSchema()

root
|-- emp_id: long (nullable = true)
|-- name: string (nullable = true)
|-- superior_emp_id: long (nullable = true)
|-- year_joined: string (nullable = true)
|-- emp_dept_id: string (nullable = true)
|-- gender: string (nullable = true)
|-- salary: long (nullable = true)
```

```
empDF.show(truncate=False)
```

0s

empDF.show(truncate=False)

emp_id	name	superior_emp_id	year_joined	emp_dept_id	gender	salary
1	Smith	-1	2018	10	M	3000
2	Rose	1	2010	20	M	4000
3	Williams	1	2010	10	M	1000
4	Jones	2	2005	10	F	2000
5	Brown	2	2010	40		-1
6	Brown	2	2010	50		-1

```
dept = [("Finance",10), ("Marketing",20), ("Sales",30), ("IT",40) ]
```

```
deptColumns = ["dept_name","dept_id"]
```

```
deptDF = spark.createDataFrame(data=dept, schema = deptColumns)
```

```
deptDF.printSchema()
```

0s

dept = [("Finance",10), ("Marketing",20), ("Sales",30), ("IT",40)]
deptColumns = ["dept_name","dept_id"]
deptDF = spark.createDataFrame(data=dept, schema = deptColumns)
deptDF.printSchema()

root
|-- dept_name: string (nullable = true)
|-- dept_id: long (nullable = true)

```
deptDF.show(truncate=False)
```

0s

deptDF.show(truncate=False)

dept_name	dept_id
Finance	10
Marketing	20
Sales	30
IT	40

2. Join the employee DataFrames with itself.

```
from pyspark.sql.functions import col
```

```
empDF.alias("emp1").join(empDF.alias("emp2"),
    col("emp1.superior_emp_id") ==
    col("emp2.emp_id"), "inner").select(col("emp1.emp_id"), col("emp1.name"),
    col("emp2.emp_id").alias("superior_emp_id"),
    col("emp2.name").alias("superior_emp_name")).show(truncate=False)
```

```
✓ 1s  from pyspark.sql.functions import col

empDF.alias("emp1").join(empDF.alias("emp2"),
    col("emp1.superior_emp_id") == col("emp2.emp_id"), "inner").select(col("emp1.emp_id"), col("emp1.name"),
    col("emp2.emp_id").alias("superior_emp_id"),
    col("emp2.name").alias("superior_emp_name")).show(truncate=False)

+-----+-----+-----+
|emp_id|name |superior_emp_id|superior_emp_name|
+-----+-----+-----+
|2     |Rose  |1             |Smith
|3     |Williams|1           |Smith
|4     |Jones  |2             |Rose
|5     |Brown  |2             |Rose
|6     |Brown  |2             |Rose
+-----+-----+-----+
```

3. Create a view and join the employee and department dataframes.

```
empDF.createOrReplaceTempView("EMP")
```

```
deptDF.createOrReplaceTempView("DEPT")
```

```
joinDF2 = spark.sql("SELECT e.* FROM EMP e LEFT OUTER JOIN DEPT d ON
e.emp_dept_id == d.dept_id").show(truncate=False)
```

```
✓ 1s  empDF.createOrReplaceTempView("EMP")
deptDF.createOrReplaceTempView("DEPT")

joinDF2 = spark.sql("SELECT e.* FROM EMP e LEFT OUTER JOIN DEPT d ON e.emp_dept_id == d.dept_id").show(truncate=False)

[+] +-----+-----+-----+-----+-----+
|emp_id|name |superior_emp_id|year_joined|emp_dept_id|gender|salary|
+-----+-----+-----+-----+-----+
|1     |Smith | -1          |2018        |10          |M    |3000 |
|3     |Williams|1           |2010        |10          |M    |1000 |
|2     |Rose   |1             |2010        |20          |M    |4000 |
|6     |Brown  |2             |2010        |50          |-1   |
|4     |Jones  |2             |2005        |10          |F    |2000 |
|5     |Brown  |2             |2010        |40          |-1   |
+-----+-----+-----+-----+-----+
```

4. Try inner, left and right joining for the employee and department table.

```
empDF.join(deptDF, empDF["emp_dept_id"] ==  
deptDF["dept_id"], "inner").show(truncate=False)
```

1s

empDF.join(deptDF, empDF["emp_dept_id"] == deptDF["dept_id"], "inner").show(truncate=False)

emp_id	name	superior_emp_id	year_joined	emp_dept_id	gender	salary	dept_name	dept_id
1	Smith	-1	2018	10	M	3000	Finance	10
3	Williams	1	2010	10	M	1000	Finance	10
4	Jones	2	2005	10	F	2000	Finance	10
2	Rose	1	2010	20	M	4000	Marketing	20
5	Brown	2	2010	40	-1	IT	40	

Example 3: Creating a Data Table in PySpark.

1. Check the list of existing databases

```
df=spark.sql("show databases")
```

```
df.show()
```

0s

df=spark.sql("show databases")
df.show()

namespace
default

2. Read the csv file and load it into the data table.

```
datafile=spark.read.csv("/FileStore/tables/Emp.csv",header=True)
```

```
datafile.show(5)
```

```
datafile.write.saveAsTable("EmpTable")
```

```
[128] datafile=spark.read.csv("/content/sample_data/EMP.CSV",header=True)  
datafile.show(5)  
datafile.write.saveAsTable("EmpTable")
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
198	Donald	OConnell	DOCONNEL	650.507.9833	21-JUN-07	SH_CLERK	2600	-	124	50
199	Douglas	Grant	DGRANT	650.507.9844	13-JAN-08	SH_CLERK	2600	-	124	50
200	Jennifer	Whalen	JWHALEN	515.123.4444	17-SEP-03	AD_ASST	4400	-	101	10
201	Michael	Hartstein	MHARTSTE	515.123.5555	17-FEB-04	MK_MAN	13000	-	100	20
202	Pat	Fay	PFAY	603.123.6666	17-AUG-05	MK_REP	6000	-	201	20

only showing top 5 rows

3. Fetch the data tables rows into the data frame.

```
df1=spark.sql("select * from EmpTable limit 5")
```

```
df1.show()
```

```
✓ [129] df1=spark.sql("select * from EmpTable limit 5")
df1.show()

+-----+-----+-----+-----+-----+-----+-----+
|EMPLOYEE_ID|FIRST_NAME|LAST_NAME|EMAIL|PHONE_NUMBER|HIRE_DATE|JOB_ID|SALARY|COMMISSION_PCT|MANAGER_ID|DEPARTMENT_ID|
+-----+-----+-----+-----+-----+-----+-----+
|    198|   Donald| OConnell|DOCONNEL|650.507.9833|21-JUN-07|SH_CLERK|  2600|      -|     124|      50|
|    199|  Douglas|   Grant| DGRANT|650.507.9844|13-JAN-08|SH_CLERK|  2600|      -|     124|      50|
|    200| Jennifer| Whalen|JWHALEN|515.123.4444|17-SEP-03|AD_ASST|  4400|      -|     101|      10|
|    201| Michael|Hartstein|MHARTSTE|515.123.5555|17-FEB-04| MK_MAN| 13000|      -|     100|      20|
|    202|      Pat|    Fay| PFAY|603.123.6666|17-AUG-05| MK_REP|  6000|      -|     201|      20|
+-----+-----+-----+-----+-----+-----+-----+
```

4. Display all records of EmpTable.

```
empDF.show(truncate=False)
```

```
✓ [130] empDF.show(truncate=False)
```

```
+-----+-----+-----+-----+-----+-----+
|emp_id|name      |superior_emp_id|year_joined|emp_dept_id|gender|salary|
+-----+-----+-----+-----+-----+-----+
| 1    |Smith      |-1            |2018        |10          |M      |3000  |
| 2    |Rose       |1             |2010        |20          |M      |4000  |
| 3    |Williams   |1             |2010        |10          |M      |1000  |
| 4    |Jones      |2             |2005        |10          |F      |2000  |
| 5    |Brown      |2             |2010        |40          |        |-1    |
| 6    |Brown      |2             |2010        |50          |        |-1    |
+-----+-----+-----+-----+-----+-----+
```

PRACTICAL 9: Spark MLlib: Regression, Classification and Clustering

Regression is a type of supervised learning where the goal is to predict a continuous value output based on input features. In this, the algorithm learns a mapping from input variables to a continuous target variable. Common regression algorithms include linear regression, polynomial regression, support vector regression, and neural networks.

Classification is another type of supervised learning where the goal is to predict the categorical class label of a new instance based on its input features. In this, the algorithm learns to classify data into predefined categories or classes. Common classification algorithms include logistic regression, decision trees, random forests, support vector machines, naive Bayes, and neural networks.

Clustering is an unsupervised learning technique where the goal is to group similar instances together in a dataset. Unlike regression and classification, clustering does not require labeled data; it automatically discovers the natural grouping or structure in the data. Common clustering algorithms include K-means clustering, hierarchical clustering, DBSCAN (Density-Based Spatial Clustering of Applications with Noise), and Gaussian Mixture Models (GMM).

1. Linear Regression

Example: Demonstrates training an elastic net regularized linear regression model and extracting model summary statistics.

```
from pyspark.ml.regression import LinearRegression

# Load training data
training = spark.read.format("libsvm")\
    .load("data/mllib/sample_linear_regression_data.txt")
training.display()

from pyspark.ml.regression import LinearRegression

# Load training data
training = spark.read.format("libsvm")\
    .load("/FileStore/tables/sample_linear_regression_data-1.
txt")

training.display()
```

▼ (2) Spark Jobs

- Job 14 View (Stages: 1/1)
- Job 15 View (Stages: 1/1)

► training: pyspark.sql.dataframe.DataFrame = [label: double, features: udt]

Table +

	label	features
1	-9.490009878824548	► [{"vectorType": "sparse", "length": 10, "indices": [0, 1, 2, 3, 4, 5, 6, 7, 8, 9], "values": [0.4551273600657362, 0.36644694351969087, -0.38255109933468047, -0.4458430198517267, 0.33109790358914726, 0.0067445293443565, -0.2624341731773807, -0.44850386111659524, -0.07266284838169332, 0.5658035575800715]}]
2	0.2577820163584905	► [{"vectorType": "sparse", "length": 10, "indices": [0, 1, 2, 3, 4, 5, 6, 7, 8, 9], "values": [0.0386555657374337, -0.1270180511534269, 0.499812362510895, -0.12686625128130267, -0.6452430441812433, 0.18869982177936828, -0.5804648622873358, 0.651931743775642, -0.6555641246242951, 0.17485476357259122]}]
3	-4.438869807456516	► [{"vectorType": "sparse", "length": 10, "indices": [0, 1, 2, 3, 4, 5, 6, 7, 8, 9], "values": [0.5025608135349202, 0.14208069682973434, 0.16004976900412138, 0.505019897181302, -0.9371035223468384, -0.2841601610457427, 0.6355938616712786, -0.1646249064941625, 0.9480713629917628, 0.42681251564645817]}]
4	-19.782762789614537	► [{"vectorType": "sparse", "length": 10, "indices": [0, 1, 2, 3, 4, 5, 6, 7, 8, 9], "values": [-0.0368509668871313, -0.4166870051763918, 0.8997202693109332, 0.6409836467726933, 0.273209095712564, -0.26175701211620517, -0.2794902492677298, -0.1306778297187794, -0.08536581111046115, -0.05462315824820923]}]
5	-7.966593841555266	► [{"vectorType": "sparse", "length": 10, "indices": [0, 1, 2, 3, 4, 5, 6, 7, 8, 9], "values": [-0.06195495076086281, 0.654644040299902, -0.6979368909424035, 0.6677324700880314, -0.07938725467767771, -0.43885601665437957, -0.608071585153888, -0.6414531182501653, 0.7313735926547045, -0.026818676347611925]}]
6	-7.806274316726144	► [{"vectorType": "sparse", "length": 10, "indices": [0, 1, 2, 3, 4, 5, 6, 7, 8, 9], "values": [-0.15805638673794265, 0.26573958270655806, 0.3097172001343442, -0.3693430098845541, 0.14324061105095334, -0.25797542063247825, 0.7436291919296774, 0.6114618853239959, 0.2324273700709574, -0.25128128782199144]}]
	-8.464803554195287	► [{"vectorType": "sparse", "length": 10, "indices": [0, 1, 2, 3, 4, 5, 6, 7, 8, 9], "values": [0.39449745853945895, 0.817229160415142, ...]}]

↓ 501 rows | 1.54 seconds runtime

```
lr = LinearRegression(maxIter=10, regParam=0.3, elasticNetParam=0.8)
```

Fit the model

```
lrModel = lr.fit(training)
```

Print the coefficients and intercept for linear regression

```
print("Coefficients: %s" % str(lrModel.coefficients))
```

```
print("Intercept: %s" % str(lrModel.intercept))
```

```
lr = LinearRegression(maxIter=10, regParam=0.3, elasticNetParam=0.8)

# Fit the model
lrModel = lr.fit(training)

# Print the coefficients and intercept for linear regression
print("Coefficients: %s" % str(lrModel.coefficients))
print("Intercept: %s" % str(lrModel.intercept))
```

OUTPUT:

► (3) Spark Jobs

► training: pyspark.sql.dataframe.DataFrame

- label: double
- features: udt

```
Coefficients: [0.0, 0.3229251667740594, -0.3438548034562219, 1.915601702345841, 0.05288058680386255, 0.765962720459771, 0.0, -0.15105392669186676, -0.21587930360904645, 0.2202536918881343]
Intercept: 0.15989368442397356
```

```
# Summarize the model over the training set and print out some metrics
trainingSummary = lrModel.summary
print("numIterations: %d" % trainingSummary.totalIterations)
print("objectiveHistory: %s" % str(trainingSummary.objectiveHistory))
trainingSummary.residuals.show()
print("RMSE: %f" % trainingSummary.rootMeanSquaredError)
print("r2: %f" % trainingSummary.r2)

# Summarize the model over the training set and print out some
metrics
trainingSummary = lrModel.summary
print("numIterations: %d" % trainingSummary.totalIterations)
print("objectiveHistory: %s" % str(trainingSummary.
objectiveHistory))
trainingSummary.residuals.show()
print("RMSE: %f" % trainingSummary.rootMeanSquaredError)
print("r2: %f" % trainingSummary.r2)
```

▶ (1) Spark Jobs

```
numIterations: 6
objectiveHistory: [0.4999999999999994, 0.4967620357443381, 0.493
63616643404634, 0.4936351537897608, 0.4936351214177871, 0.4936351
2062528014, 0.4936351206216114]
+-----+
|      residuals|
+-----+
| -9.889232683103197|
|  0.5533794340053553|
| -5.204019455758822|
| -20.566686715507508|
| -9.4497405180564|
| -6.909112502719487|
| -10.00431602969873|
|  2.0623978070504845|
|  3.1117508432954772|
| -15.89360822941938|
| -5.036284254673026|
|  6.4832158769943335|
|  12.429497299109002|
| -20.32003219007654|
```

```
RMSE: 10.189077
r2: 0.022861
```

```
ypredict=lrModel.transform(training)
ypredict.show(5)
```

```
ypredict=lrModel.transform(training)
ypredict.show(5)

▶ (1) Spark Jobs
  ▶  ypredict: pyspark.sql.dataframe.DataFrame = [label: double, features: udt ... 1 more field]

+-----+-----+-----+
|       label|      features|      prediction|
+-----+-----+-----+
| -9.490009878824548|(10,[0,1,2,3,4,5,...| 0.3992228042786488|
| 0.2577820163584905|(10,[0,1,2,3,4,5,...|-0.2955974176468648|
| -4.438869807456516|(10,[0,1,2,3,4,5,...| 0.7651496483023065|
| -19.782762789614537|(10,[0,1,2,3,4,5,...| 0.783923925892972|
| -7.966593841555266|(10,[0,1,2,3,4,5,...| 1.4831466765011345|
+-----+-----+-----+
only showing top 5 rows
```

Example 2:

#Load the data

```
dataset = spark.read.csv("/databricks-datasets/adult/adult.data", schema=schema)
```

```
dataset=spark.read.csv("/FileStore/tables/adult1.txt",schema=schema)
dataset.show()
```

```
▶ (1) Spark Jobs
  ▶  dataset: pyspark.sql.dataframe.DataFrame = [age: double, workclass: string ... 13 more fields]

35.0| United-States| <=50K|
|48.0|          Private|279724.0|      HS-grad|         9.0|Married-civ-spouse
|Machine-op-inspct|      Husband|White| Male|     3103.0|        0.0|
48.0| United-States| >50K|
|43.0|          Private|346189.0|      Masters|        14.0|Married-civ-spouse
| Exec-managerial|      Husband|White| Male|        0.0|        0.0|
50.0| United-States| >50K|
|20.0|      State-gov|444554.0|Some-college|       10.0|    Never-married
| Other-service|      Own-child|White| Male|        0.0|        0.0|
25.0| United-States| <=50K|
|43.0|          Private|128354.0|      HS-grad|         9.0|Married-civ-spouse
| Adm-clerical|      Wife|White|Female|        0.0|        0.0|
30.0| United-States| <=50K|
|37.0|          Private| 60548.0|      HS-grad|         9.0|        Widowed
|Machine-op-inspct|      Unmarried|White|Female|        0.0|        0.0|
20.0| United-States| <=50K|
+-----+-----+-----+
+-----+-----+-----+
-----+-----+
only showing top 20 rows
```

```
#Define the column names
schema = """`age` DOUBLE,
`workclass` STRING,
`fnlwgt` DOUBLE,
`education` STRING,
`education_num` DOUBLE,
`marital_status` STRING,
`occupation` STRING,
`relationship` STRING,
`race` STRING,
`sex` STRING,
`capital_gain` DOUBLE,
`capital_loss` DOUBLE,
`hours_per_week` DOUBLE,
`native_country` STRING,
`income` STRING"""

schema = """ age` DOUBLE,
`workclass` STRING,
`fnlwgt` DOUBLE,
`education` STRING,
`education_num` DOUBLE,
`marital_status` STRING,
`occupation` STRING,
`relationship` STRING,
`race` STRING,
`sex` STRING,
`capital_gain` DOUBLE,
`capital_loss` DOUBLE,
`hours_per_week` DOUBLE,
`native_country` STRING,
`income` STRING"""

```

```
#Divide data in training and testing dataframe
trainDF, testDF = dataset.randomSplit([0.8, 0.2], seed=42)
print(trainDF.cache().count()) # Cache because accessing training data multiple times
print(testDF.count())
```

```
trainDF,testDF=dataset.randomSplit([0.8,0.2],seed=42)
print(trainDF.cache().count())
print(testDF.count())

▶ (4) Spark Jobs
▶   trainDF: pyspark.sql.dataframe.DataFrame = [age: double, workclass: string ... 13 more fields]
▶   testDF: pyspark.sql.dataframe.DataFrame = [age: double, workclass: string ... 13 more fields]
39137
9706
```

```
#Explore data  
display(trainDF)
```

```
display(trainDF)
```

▶ (1) Spark Jobs

Table ▼ +

	age	workclass	fnlwgt	education	education_nu
1	null	workclass	null	education	null
2	17	?	27251	11th	7
3	17	?	34505	11th	7
4	17	?	35603	11th	7
5	17	?	40299	10th	6
6	17	?	45037	10th	6
7	17	?	54978	7th-8th	4

▼ ▼ 10,000 rows | Truncated data | 0.52 seconds runtime | Refreshed 14 minutes ago

#What's the distribution of the number of hours_per_week?

```
display(trainDF.select("hours_per_week").summary())
```

```
#What's the distribution of the number of hours_per_week?  
display(trainDF.select("hours_per_week").summary())
```

► (2) Spark Jobs

Table +

	summary	hours_per_week
1	count	39136
2	mean	40.40561120196239
3	stddev	12.40954323086354
4	min	1.0
5	25%	40.0
6	50%	40.0
7	75%	45.0

8 rows | 0.87 seconds runtime Refreshed 15 minutes ago

```
#What is education status?
display(trainDF
    .groupBy("education")
    .count()
    .sort("count", ascending=False))
```

#What is education status?

```
display(trainDF.groupBy("education").count().sort("count", ascending=False))
```

▶ (2) Spark Jobs

Table +

	education	count
1	HS-grad	12647
2	Some-college	8693
3	Bachelors	6429
4	Masters	2117
5	Assoc-voc	1663
6	11th	1478
7	Assoc-acdm	1282

↓ 17 rows | 1.99 seconds runtime Refreshed 14 minutes ago

```
# Feature Engineering
# Convert the categorical data into numeric data
from pyspark.ml.feature import StringIndexer, OneHotEncoder
categoricalCols = ["workclass", "education", "marital_status", "occupation", "relationship",
"race", "sex"]
# The following two lines are estimators. They return functions that we will later apply to
transform the dataset.
stringIndexer = StringIndexer(inputCols=categoricalCols, outputCols=[x + "Index" for x in
categoricalCols])
encoder = OneHotEncoder(inputCols=stringIndexer.getOutputCols(), outputCols=[x +
"OhE" for x in categoricalCols])
# The label column ("income") is also a string value - it has two possible values, "<=50K"
and ">50K".
# Convert it to a numeric value using StringIndexer.
labelToIndex = StringIndexer(inputCol="income", outputCol="label")
```

```

from pyspark.ml.feature import StringIndexer, OneHotEncoder
categoricalCols=["workclass","education","marital_status","occupation",
"relationship","race","sex"]
stringIndexer = StringIndexer(inputCols=categoricalCols, outputCols=[x +
"x_Index" for x in categoricalCols])
encoder = OneHotEncoder(inputCols=stringIndexer.getOutputCols(), outputCols=
[x + "OHE" for x in categoricalCols])
labelToIndex = StringIndexer(inputCol="income", outputCol="label")

stringIndexerModel = stringIndexer.fit(trainDF)
display(stringIndexerModel.transform(trainDF))

stringIndexerModel=stringIndexer.fit(trainDF)
display(stringIndexerModel.transform(trainDF))

▶ (3) Spark Jobs

```

Table +

	age	workclass	fnlwgt	education	education_nu
1	null	workclass	null	education	null
2	17	?	27251	11th	7
3	17	?	34505	11th	7
4	17	?	35603	11th	7
5	17	?	40299	10th	6
6	17	?	45037	10th	6
7	17	?	54978	7th-8th	4

↓ ▾ 10,000 rows | Truncated data | 2.55 seconds runtime Refreshed now

```

# Convert all features into a single feature
from pyspark.ml.feature import VectorAssembler
# This includes both the numeric columns and the one-hot encoded binary vector columns in
our dataset.
numericCols = ["age", "fnlwgt", "education_num", "capital_gain", "capital_loss",
"hours_per_week"]
assemblerInputs = [c + "OHE" for c in categoricalCols] + numericCols
vecAssembler = VectorAssembler(inputCols=assemblerInputs, outputCol="features")

```

```

#Convert all features into single feature
from pyspark.ml.feature import VectorAssembler
numericCols = ["age", "fnlwgt", "education_num", "capital_gain",
"capital_loss", "hours_per_week"]
assemblerInputs = [c + "OHE" for c in categoricalCols] + numericCols
vecAssembler = VectorAssembler(inputCols=assemblerInputs,
outputCol="features")

```

```
# Define model
```

```
from pyspark.ml.classification import LogisticRegression
lr = LogisticRegression(featuresCol="features", labelCol="label", regParam=1.0)
```

```
#Define model
```

```
from pyspark.ml.classification import LogisticRegression
lr = LogisticRegression(featuresCol="features", labelCol="label", regParam=1.
0)
```

```
# Build the pipeline
```

```
from pyspark.ml import Pipeline
```

```
# Define the pipeline based on the stages created in previous steps.
```

```
pipeline = Pipeline(stages=[stringIndexer, encoder, labelToIndex, vecAssembler, lr])
```

```
# Define the pipeline model.
```

```
pipelineModel = pipeline.fit(trainDF)
```

```
# Apply the pipeline model to the test dataset.
```

```
predDF = pipelineModel.transform(testDF)
```

```
# Build the pipeline
from pyspark.ml import Pipeline
# Define the pipeline based on the stages created in previous steps.
pipeline = Pipeline(stages=[stringIndexer, encoder, labelToIndex,
vecAssembler, lr])
# Define the pipeline model.
pipelineModel = pipeline.fit(trainDF)
# Apply the pipeline model to the test dataset.
predDF = pipelineModel.transform(testDF)
```

▶ (17) Spark Jobs

▶ predDF: pyspark.sql.dataframe.DataFrame = [age: double, workclass: string ... 32]

```
# Display the predictions
```

```
display(predDF.select("features", "label", "prediction", "probability"))
```

```
# Display the predictions
display(predDF.select("features", "label", "prediction", "probability"))
```

▶ (1) Spark Jobs

Table ▾ +

	features
1	▶ {"vectorType": "sparse", "length": 59, "indices": [3, 13, 24, 36, 45, 48, 53, 54, 55, 58], "values": [1, 1, 1, 1, 1, 1, 1, 1, 1]}
2	▶ {"vectorType": "sparse", "length": 59, "indices": [3, 15, 24, 36, 45, 48, 52, 53, 54, 55, 58], "values": [1, 1, 1, 1, 1, 1, 1, 1, 1]}
3	▶ {"vectorType": "sparse", "length": 59, "indices": [3, 13, 24, 36, 45, 48, 53, 54, 55, 58], "values": [1, 1, 1, 1, 1, 1, 1, 1, 1]}
4	▶ {"vectorType": "sparse", "length": 59, "indices": [3, 13, 24, 36, 45, 48, 52, 53, 54, 55, 58], "values": [1, 1, 1, 1, 1, 1, 1, 1, 1]}
5	▶ {"vectorType": "sparse", "length": 59, "indices": [3, 15, 24, 36, 45, 48, 53, 54, 55, 58], "values": [1, 1, 1, 1, 1, 1, 1, 1, 1]}
...	...

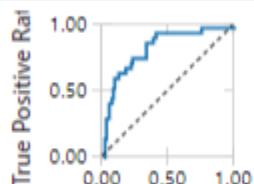
↓ 6,485 rows | 2.26 seconds runtime

Refreshed now

```
# Evaluate model  
display(pipelineModel.stages[-1], predDF.drop("prediction", "rawPrediction",  
"probability"), "ROC")
```

```
# Evaluate model  
display(pipelineModel.stages[-1], predDF.drop("prediction", "rawPrediction",  
"probability"), "ROC")
```

▶ (8) Spark Jobs



False Positive Rate

↓ 123 rows | 2.76 seconds runtime

Refreshed now

Practical 10: Data Ingestion: Importing Relational Data with Sqoop, Ingesting Streaming Data with Flume.

1. hostname -f //Checks the hostname

```
[cloudera@quickstart ~]$ hostname -f
quickstart.cloudera
[cloudera@quickstart ~]$ █
```

2. Lists the databases using Sqoop

```
sqoop list-databases --connect jdbc:mysql://localhost/ --password cloudera --username root;
```

```
[cloudera@quickstart ~]$ sqoop list-databases --connect jdbc:mysql://localhost/ --password cloudera --username root;
Warning: /usr/lib/sqoop/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
24/03/11 11:49:15 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.13.0
24/03/11 11:49:15 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.
24/03/11 11:49:15 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
information_schema
cm
firehose
hue
metastore
mysql
nav
navms
oozie
retail_db
rman
sentry
[cloudera@quickstart ~]$ █
```

3. Now to list the tables present in retail_db using Sqoop, type the following command:

```
sqoop list-tables --connect jdbc:mysql://quickstart:3306/retail_db --password cloudera --username root;
```

```
[cloudera@quickstart ~]$ sqoop list-tables --connect jdbc:mysql://quickstart:3306/retail_db --password cloudera --username root;
Warning: /usr/lib/sqoop/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
24/03/11 11:52:37 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.13.0
24/03/11 11:52:37 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.
24/03/11 11:52:37 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
categories
customers
departments
order_items
orders
products
[cloudera@quickstart ~]$ █
```

4. Executes Map Tasks at the back end

```
sqoop import --connect jdbc:mysql://quickstart:3306/retail_db --password cloudera --username root --table departments;
```

```
[clouder@quickstart ~]$ sqoop import --connect jdbc:mysql://quickstart:3306/retail_db --password cloudera --username root --table departments;
Warning: /usr/lib/sqoop/.Accumulo does not exist! Accumulo imports will fail.
Please set $ACUMULO_HOME to the root of your Accumulo installation.
24/03/11 11:55:23 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.13.0
24/03/11 11:55:23 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.
24/03/11 11:55:24 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
24/03/11 11:55:24 INFO tool.CompressionTool: Beginning code generation
24/03/11 11:55:24 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `departments` AS t LIMIT 1
24/03/11 11:55:24 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `departments` AS t LIMIT 1
Note: /tmp/sqoop-cloudera/compile/7b3758d458a555c8a08d634c65d73c5/departments.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
24/03/11 11:55:26 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-cloudera/compile/7b3758d458a555c8a08d634c65d73c5/departments.jar
24/03/11 11:55:26 INFO Configuration.deprecation: mapred.job.tracker looks like you have deprecated configuration, did you mean mapred.job.tracker?
24/03/11 11:55:26 WARN manager.MySQLManager: This transfer can be faster! Use the -direct
24/03/11 11:55:26 INFO manager.MySQLManager: option to exercise a MySQL-specific fast path.
24/03/11 11:55:26 INFO manager.MySQLManager: Setting zero DATETIME behavior to convertToNull (mysql)
24/03/11 11:55:26 INFO mapreduce.ImportJobBase: Beginning import of departments
24/03/11 11:55:26 INFO Configuration.deprecation: mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
24/03/11 11:55:26 INFO Configuration.deprecation: mapred.jar is deprecated. Instead, use mapreduce.job.jar
24/03/11 11:55:26 INFO Configuration.deprecation: mapred.map.tasks is deprecated. Instead, use mapreduce.job.maps
24/03/11 11:55:27 INFO org.apache.hadoop.net.NetUtils: Connecting to ResourceManager at /0.0.0.0:8032
24/03/11 11:55:28 WARN hdfs.DFSClient: Caught exception
java.lang.InterruptedException
    at java.lang.Object.wait(Native Method)
    at java.lang.Thread.join(Thread.java:1281)
    at java.lang.Thread.join(Thread.java:1351)
    at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.closeResponder(DFSOutputStream.java:967)
    at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.endBlock(DFSOutputStream.java:705)
    at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.run(DFSOutputStream.java:894)
24/03/11 11:55:28 WARN hdfs.DFSClient: Caught exception
java.lang.InterruptedException
    at java.lang.Object.wait(Native Method)
    at java.lang.Thread.join(Thread.java:1281)
    at java.lang.Thread.join(Thread.java:1351)
    at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.closeResponder(DFSOutputStream.java:967)
    at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.endBlock(DFSOutputStream.java:705)
```