1. Upload hr\_db dataset into HDFS

2. Solve the following use cases using Hive SQL or Spark RDD or Spark DF (Pick any one of these)

a. Retrieve last\_name, first\_name, salary, salary+300 as increment salary from employees table

b. Find out the ID's of departments in which employees are working

c. Retrieve last\_name, job\_id, department\_id of employee having last name as Whalen

d. Retrieve last name and salary of all employees who have salary greater than 5000 working in department 90

e. Retrieve last\_name, salary of all employees working as 'SA\_REP','AD\_PRES' earning above 5000, sort the data in ascending order of last name

f. Retrieve the last names of employees whose last but one character of the last name is e

g. Retrieve the names of employees not having managers

h. Retrieve the employee name and the name of the department in which the employee is working

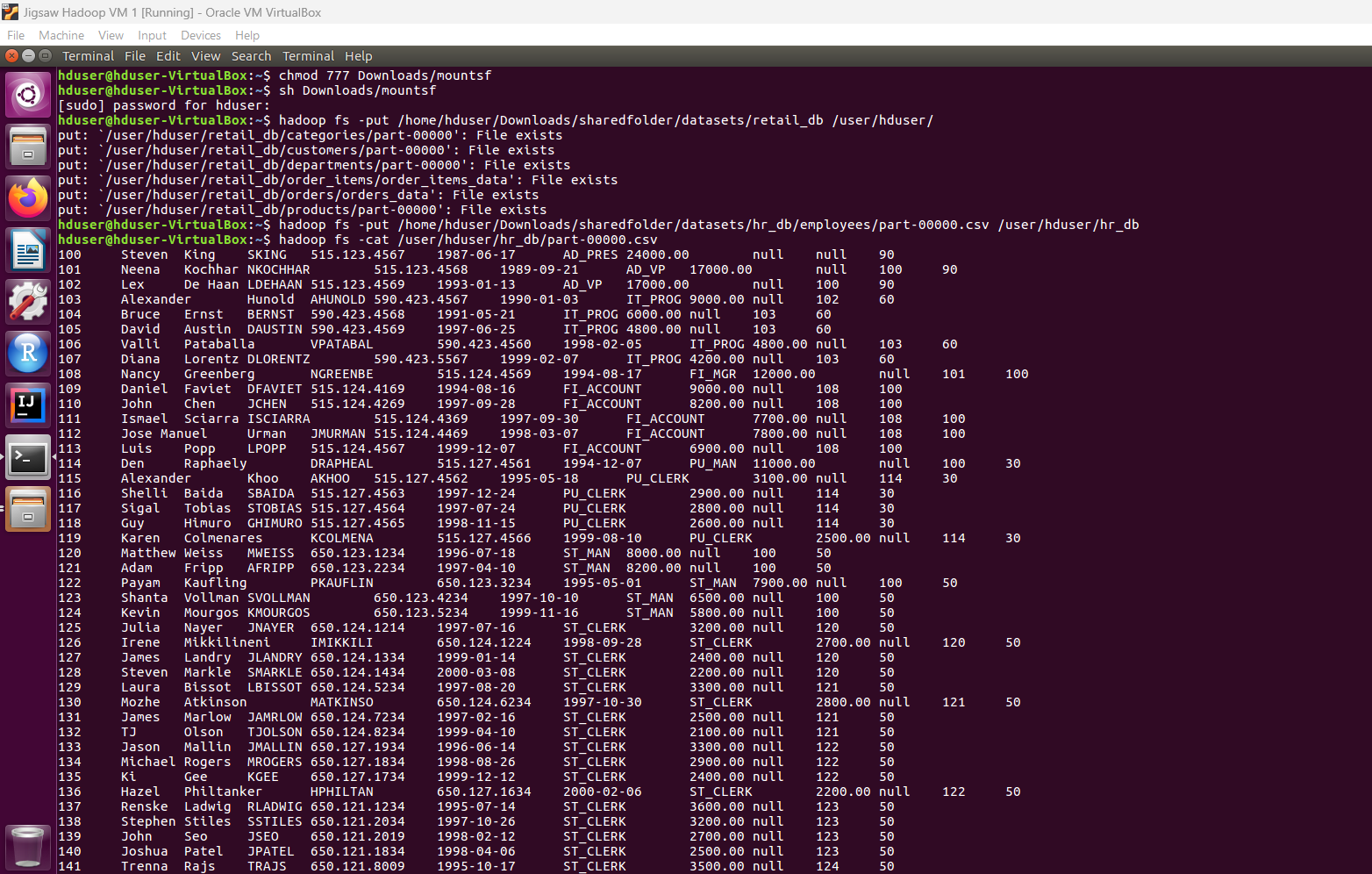
I. List all the department ids having SIX employees

1. **Upload hr\_db dataset into HDFS**

$ hadoop fs -mkdir /user/hduser/hr\_db

$ hadoop fs -put /home/hduser/Downloads/sharedfolder/datasets/hr\_db/employees/part-00000.csv /user/hduser/hr\_db

$ hadoop fs -cat /user/hduser/hr\_db/part-00000.csv



**2. Solve the following use cases using Hive SQL or Spark RDD or Spark DF (Pick any one of these)**

**a. Retrieve last\_name, first\_name, salary, salary+300 as increment salary from employees table**

hive> CREATE DATABASE employees\_riddhi ;

hive> SHOW DATABASES ;

hive> USE employees\_riddhi ;

hive>

CREATE EXTERNAL TABLE employees(

> first\_name STRING,

> last\_name STRING,

> email STRING,

> phone\_number INT,

> hire\_date INT,

> job\_id INT,

> salary INT,

> commission\_pct FLOAT,

> manager\_id INT,

> department\_id INT

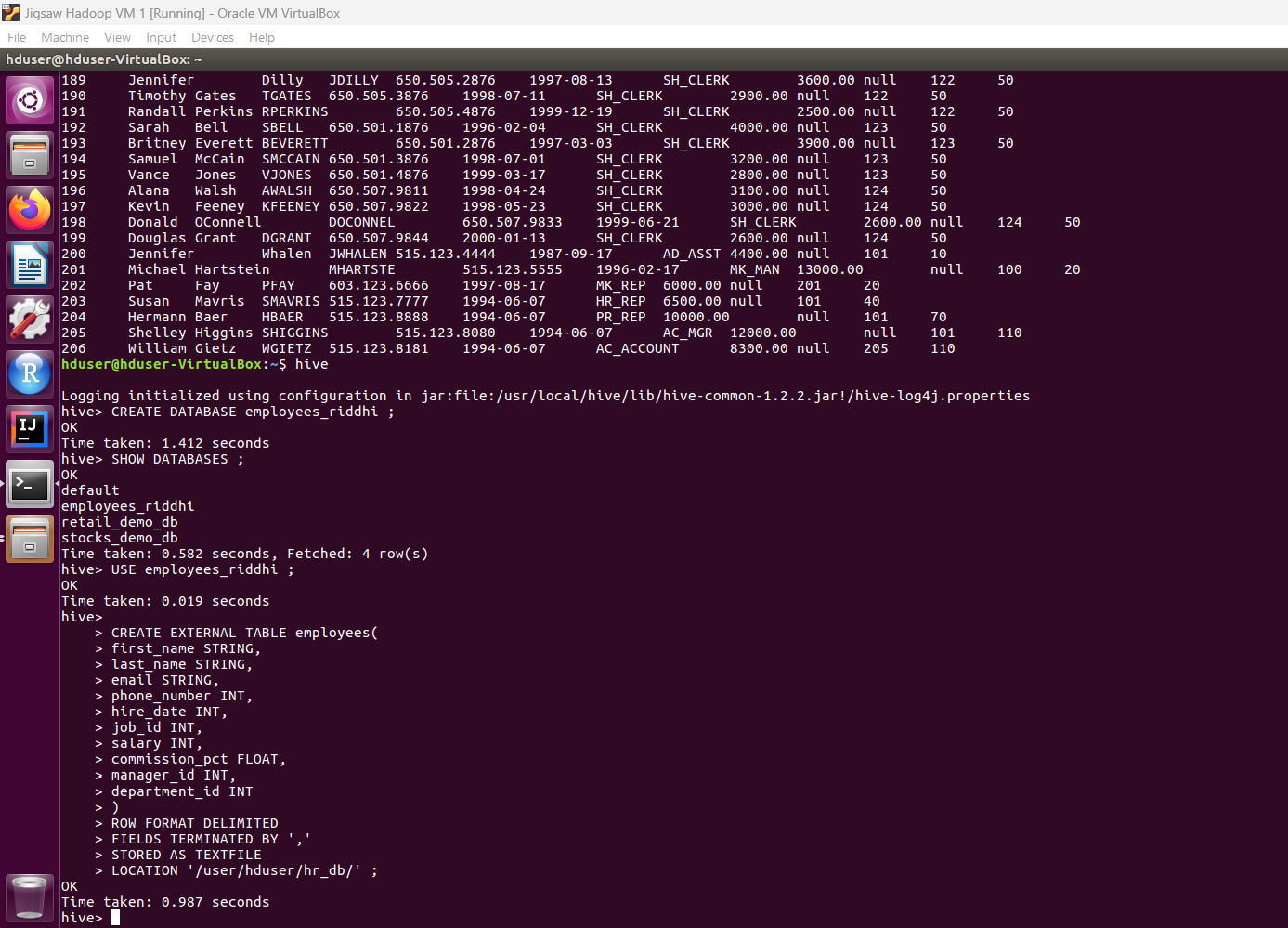
> )

> ROW FORMAT DELIMITED

> FIELDS TERMINATED BY ','

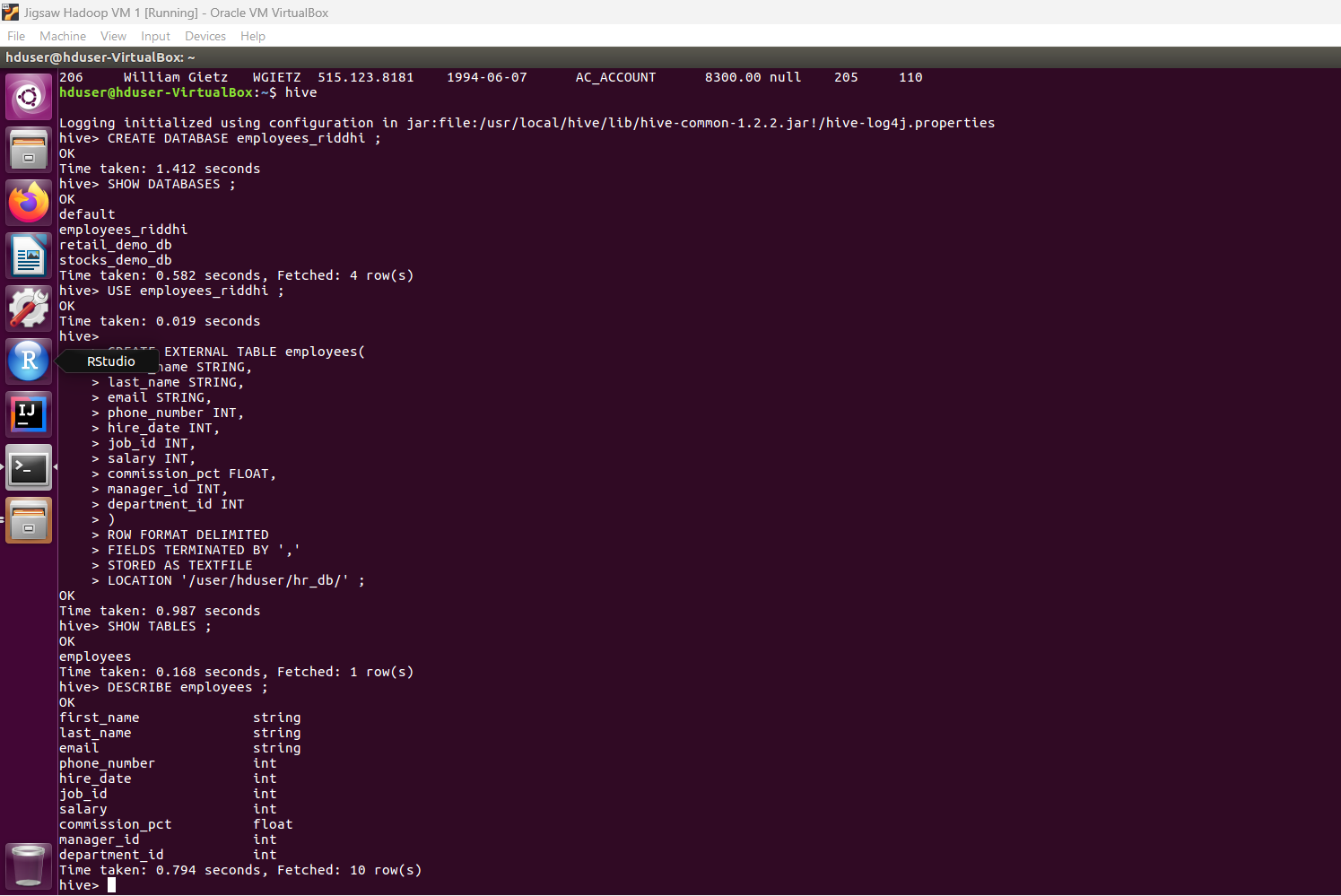
> STORED AS TEXTFILE

> LOCATION '/user/hduser/hr\_db/' ;



hive> SHOW TABLES ;

hive> DESCRIBE employees ;



hive> SELECT \* FROM employees ;

**Solution:**

hive> select last\_name, first\_name, salary, commission\_pct from employees ;

**b. Find out the ID's of departments in which employees are working**

hive> SHOW TABLES ;

hive> select distinct department\_id from employees ;

department\_id

0

10

20

30

40

50

60

70

80

90

100

110

Time taken: 0.908 seconds, Fetched: 12 row(s)

**c. Retrieve last\_name, job\_id, department\_id of employee having last name as Whalen**

hive> SELECT last\_name, job\_id, dapartment\_id FROM employees WHERE last\_name IN('Whalen');

last\_name job\_id department\_id

Whalen AD\_ASST 10

**d. Retrieve last name and salary of all employees who have salary greater than 5000 working in department 90**

hive> SELECT last\_name, salary FROM employees WHERE salary > 5000 AND department\_id = 90 ;

last\_name salary

last\_name salary

King 24000.00

Kochhar 17000.00

De Haan 17000.00

**e. Retrieve last\_name, salary of all employees working as 'SA\_REP','AD\_PRES' earning above 5000, sort the data in ascending order of last name**

hive> SELECT last\_name, salary FROM employees WHERE job\_id = ('SA\_REP' OR 'AD\_PRES') AND salary > 5000 ORDER BY salary ASC ;

last\_name salary

Mourgos 5800.00

Ernst 6000.00

Fay 6000.00

Kumar 6100.00

Banda 6200.00

Johnson 6200.00

Ande 6400.00

Mavris 6500.00

Vollman 6500.00

Lee 6800.00

Popp 6900.00

Tuvault 7000.00

Sewall 7000.00

Grant 7000.00

Marvins 7200.00

Bates 7300.00

Smith 7400.00

Doran 7500.00

Cambrault 7500.00

Sciarra 7700.00

Urman 7800.00

Kaufling 7900.00

Weiss 8000.00

Smith 8000.00

Olsen 8000.00

Fripp 8200.00

Chen 8200.00

Gietz 8300.00

Livingston 8400.00

Taylor 8600.00

Hutton 8800.00

Hall 9000.00

McEwen 9000.00

Faviet 9000.00

Hunold 9000.00

Sully 9500.00

Bernstein 9500.00

Greene 9500.00

Fox 9600.00

Baer 10000.00

King 10000.00

Bloom 10000.00

Tucker 10000.00

Vishney 10500.00

Zlotkey 10500.00

Cambrault 11000.00

Raphaely 11000.00

Abel 11000.00

Ozer 11500.00

Errazuriz 12000.00

Greenberg 12000.00

Higgins 12000.00

Hartstein 13000.00

Partners 13500.00

Russell 14000.00

De Haan 17000.00

Kochhar 17000.00

King 24000.00

**f. Retrieve the last names of employees whose last but one character of the last name is e**

hive> SELECT last\_name FROM employees WHERE last\_name LIKE '\_\_e%'

**g. Retrieve the names of employees not having managers**

hive> SELECT first\_name, last\_name FROM employees WHERE manager\_id IS NOT NULL;

first\_name last\_name

Steven King

Neena Kochhar

Lex De Haan

Alexander Hunold

Bruce Ernst

David Austin

Valli Pataballa

Diana Lorentz

Nancy Greenberg

Daniel Faviet

John Chen

Ismael Sciarra

Jose Manuel Urman

Luis Popp

Den Raphaely

Alexander Khoo

Shelli Baida

Sigal Tobias

Guy Himuro

Karen Colmenares

Matthew Weiss

Adam Fripp

Payam Kaufling

Shanta Vollman

Kevin Mourgos

Julia Nayer

Irene Mikkilineni

James Landry

Steven Markle

Laura Bissot

Mozhe Atkinson

James Marlow

TJ Olson

Jason Mallin

Michael Rogers

Ki Gee

Hazel Philtanker

Renske Ladwig

Stephen Stiles

John Seo

Joshua Patel

Trenna Rajs

Curtis Davies

Randall Matos

Peter Vargas

John Russell

Karen Partners

Alberto Errazuriz

Gerald Cambrault

Eleni Zlotkey

Peter Tucker

David Bernstein

Peter Hall

Christopher Olsen

Nanette Cambrault

Oliver Tuvault

Janette King

Patrick Sully

Allan McEwen

Lindsey Smith

Louise Doran

Sarath Sewall

Clara Vishney

Danielle Greene

Mattea Marvins

David Lee

Sundar Ande

Amit Banda

Lisa Ozer

Harrison Bloom

Tayler Fox

William Smith

Elizabeth Bates

Sundita Kumar

Ellen Abel

Alyssa Hutton

Jonathon Taylor

Jack Livingston

Kimberely Grant

Charles Johnson

Winston Taylor

Jean Fleaur

Martha Sullivan

Girard Geoni

Nandita Sarchand

Alexis Bull

Julia Dellinger

Anthony Cabrio

Kelly Chung

Jennifer Dilly

Timothy Gates

Randall Perkins

Sarah Bell

Britney Everett

Samuel McCain

Vance Jones

Alana Walsh

Kevin Feeney

Donald OConnell

Douglas Grant

Jennifer Whalen

Michael Hartstein

Pat Fay

Susan Mavris

Hermann Baer

Shelley Higgins

William Gietz

**h. Retrieve the employee name and the name of the department in which the employee is working**

hive> SELECT fisrt\_name, last\_name FROM employees WHERE department\_id IS NOT NULL;

first\_name last\_name

Steven King

Neena Kochhar

Lex De Haan

Alexander Hunold

Bruce Ernst

David Austin

Valli Pataballa

Diana Lorentz

Nancy Greenberg

Daniel Faviet

John Chen

Ismael Sciarra

Jose Manuel Urman

Luis Popp

Den Raphaely

Alexander Khoo

Shelli Baida

Sigal Tobias

Guy Himuro

Karen Colmenares

Matthew Weiss

Adam Fripp

Payam Kaufling

Shanta Vollman

Kevin Mourgos

Julia Nayer

Irene Mikkilineni

James Landry

Steven Markle

Laura Bissot

Mozhe Atkinson

James Marlow

TJ Olson

Jason Mallin

Michael Rogers

Ki Gee

Hazel Philtanker

Renske Ladwig

Stephen Stiles

John Seo

Joshua Patel

Trenna Rajs

Curtis Davies

Randall Matos

Peter Vargas

John Russell

Karen Partners

Alberto Errazuriz

Gerald Cambrault

Eleni Zlotkey

Peter Tucker

David Bernstein

Peter Hall

Christopher Olsen

Nanette Cambrault

Oliver Tuvault

Janette King

Patrick Sully

Allan McEwen

Lindsey Smith

Louise Doran

Sarath Sewall

Clara Vishney

Danielle Greene

Mattea Marvins

David Lee

Sundar Ande

Amit Banda

Lisa Ozer

Harrison Bloom

Tayler Fox

William Smith

Elizabeth Bates

Sundita Kumar

Ellen Abel

Alyssa Hutton

Jonathon Taylor

Jack Livingston

Kimberely Grant

Charles Johnson

Winston Taylor

Jean Fleaur

Martha Sullivan

Girard Geoni

Nandita Sarchand

Alexis Bull

Julia Dellinger

Anthony Cabrio

Kelly Chung

Jennifer Dilly

Timothy Gates

Randall Perkins

Sarah Bell

Britney Everett

Samuel McCain

Vance Jones

Alana Walsh

Kevin Feeney

Donald OConnell

Douglas Grant

Jennifer Whalen

Michael Hartstein

Pat Fay

Susan Mavris

Hermann Baer

Shelley Higgins

William Gietz

**i. List all the department ids having SIX employees**

hive> Select DepartmentName, Count(\*) as EmployeeCount

from employees

JOIN Departments

on Employees.DepartmentID = Departments.DepartmentID

Group By DepartmentName

**Using Pyspark**

Pre-requisites:

a. Download the hr\_db dataset folder into your local.

b. In Databricks community edition, create a cluster hr\_db and confirm once it is up and running

c. In Data tab. create Table and import employees.csv using drag and drop.

d. Import rest of the datasets -all folders in hr\_db using drag and drop

Steps:

**#1. Upload hr\_db dataset into HDFS**

import pyspark.sql.types as typ

#Upload employees dataset

empCols = [('employee\_id', typ.IntegerType(), True), ('first\_name', typ.StringType(), True),('last\_name', typ.StringType(), True),('email', typ.StringType(), True),('phone\_number', typ.StringType(), True),('hire\_date', typ.StringType(), True),('job\_id', typ.StringType(), True),('salary', typ.DoubleType(), True),('commission\_pct', typ.StringType(), True),('manager\_id', typ.IntegerType(), True),('department\_id', typ.IntegerType(), True)]

empSchema = typ.StructType([typ.StructField(e[0], e[1], e[2]) for e in empCols ])

# File location and type

file\_location = "/FileStore/tables/employees.csv"

file\_type = "csv"

# CSV options

first\_row\_is\_header = "false"

delimiter = "\t"

# The applied options are for CSV files. For other file types, these will be ignored.

df = spark.read.format(file\_type).option("header", first\_row\_is\_header).option("sep", delimiter).schema(empSchema).load(file\_location)

display(df)

# Create a view or table

temp\_table\_name = "EMPLOYEES"

df.createOrReplaceTempView(temp\_table\_name)

#Upload departments dataset

deptCols = [('department\_id', typ.IntegerType(), True), ('department\_name', typ.StringType(), True),('manager\_id', typ.IntegerType(), True),('location\_id', typ.IntegerType(), True)]

deptSchema = typ.StructType([typ.StructField(e[0], e[1], e[2]) for e in deptCols ])

# File location and type

dept\_file\_location1 = "/FileStore/tables/hr\_db/departments/"

# CSV options

delimiter = " "

# The applied options are for CSV files. For other file types, these will be ignored.

dept\_df = spark.read.format(file\_type).option("header", first\_row\_is\_header).option("sep", delimiter).schema(deptSchema).load(dept\_file\_location1)

display(dept\_df)

temp\_table\_name = "DEPARTMENTS"

dept\_df.createOrReplaceTempView(temp\_table\_name)

#Upload rest of the datasets

import pyspark.sql.types as typ

countriesCols = [('country\_id', typ.StringType(), True), ('country\_name', typ.StringType(), True),('region\_id', typ.IntegerType(), True)]

countriesSchema = typ.StructType([typ.StructField(e[0], e[1], e[2]) for e in countriesCols ])

# File location and type

countries\_file\_location1 = "/FileStore/tables/hr\_db/countries/"

file\_type = "csv"

# CSV options

#infer\_schema = "false"

first\_row\_is\_header = "false"

delimiter = " "

# The applied options are for CSV files. For other file types, these will be ignored.

countries\_df = spark.read.format(file\_type).option("header", first\_row\_is\_header).option("sep", delimiter).schema(countriesSchema).load(countries\_file\_location1)

temp\_table\_name = "COUNTRIES"

countries\_df.createOrReplaceTempView(temp\_table\_name)

regionsCols = [('region\_id', typ.StringType(), True), ('region\_name', typ.StringType(), True)]

regionsSchema = typ.StructType([typ.StructField(e[0], e[1], e[2]) for e in regionsCols ])

# File location and type

regions\_file\_location1 = "/FileStore/tables/hr\_db/regions/"

regions\_df = spark.read.format(file\_type).option("header", first\_row\_is\_header).option("sep", delimiter).schema(regionsSchema).load(regions\_file\_location1)

temp\_table\_name = "REGIONS"

regions\_df.createOrReplaceTempView(temp\_table\_name)

locationsCols = [('location\_id', typ.IntegerType(), True), ('street\_address', typ.StringType(), True),('postal\_code', typ.IntegerType(), True), ('city', typ.StringType(), True),('state\_province', typ.StringType(), True), ('country\_id', typ.StringType(), True)]

locationsSchema = typ.StructType([typ.StructField(e[0], e[1], e[2]) for e in locationsCols ])

# File location and type

locations\_file\_location1 = "/FileStore/tables/hr\_db/locations/"

locations\_df = spark.read.format(file\_type).option("header", first\_row\_is\_header).option("sep", delimiter).schema(locationsSchema).load(locations\_file\_location1)

temp\_table\_name = "LOCATIONS"

locations\_df.createOrReplaceTempView(temp\_table\_name)

jobsCols = [('job\_id', typ.StringType(), True), ('job\_title', typ.StringType(), True),('min\_salary', typ.DoubleType(), True), ('max\_salary', typ.DoubleType(), True)]

jobsSchema = typ.StructType([typ.StructField(e[0], e[1], e[2]) for e in jobsCols ])

# File location and type

jobs\_file\_location1 = "/FileStore/tables/hr\_db/jobs/"

jobs\_df = spark.read.format(file\_type).option("header", first\_row\_is\_header).option("sep", delimiter).schema(jobsSchema).load(jobs\_file\_location1)

temp\_table\_name = "JOBS"

jobs\_df.createOrReplaceTempView(temp\_table\_name)

job\_historyCols = [('employee\_id', typ.IntegerType(), True), ('start\_date', typ.StringType(), True),('end\_date', typ.StringType(), True), ('job\_id', typ.StringType(), True),('department\_id', typ.IntegerType(), True)]

job\_historySchema = typ.StructType([typ.StructField(e[0], e[1], e[2]) for e in job\_historyCols ])

# File location and type

job\_history\_file\_location1 = "/FileStore/tables/hr\_db/job\_history/"

job\_history\_df = spark.read.format(file\_type).option("header", first\_row\_is\_header).option("sep", delimiter).schema(job\_historySchema).load(job\_history\_file\_location1)

temp\_table\_name = "JOB\_HISTORY"

job\_history\_df.createOrReplaceTempView(temp\_table\_name)

**#2. Solve the following use cases using Hive SQL or Spark RDD or Spark DF (Pick any one of these)**

**#a. Retrieve last\_name, first\_name, salary, salary+300 as increment salary from employees table**

query=("select last\_name,first\_name,salary, salary+300 as incremental\_salary from EMPLOYEES")

spark.sql(query).show()

**#b. Find out the ID's of departments in which employees are working**

query=("select distinct department\_id from EMPLOYEES")

spark.sql(query).show()

**#c. Retrieve last\_name, job\_id, department\_id of employee having last name as Whalen**

query=("select LAST\_NAME,job\_id,department\_id from EMPLOYEES where last\_name='Whalen'")

spark.sql(query).show()

**#d. Retrieve last name and salary of all employees who have salary greater than 5000 working in department 90**

query=("select LAST\_NAME,salary from EMPLOYEES where salary>5000 and department\_id=90")

spark.sql(query).show()

**#e. Retrieve last\_name, salary of all employees working as 'SA\_REP','AD\_PRES' earning above 5000, sort the data in ascending order of last name**

query=("select LAST\_NAME,salary from EMPLOYEES where salary>5000 and job\_id in ('SA\_REP','AD\_PRES') order by last\_name")

spark.sql(query).show()

**#f. Retrieve the last names of employees whose last but one character of the last name is e**

query=("select LAST\_NAME from EMPLOYEES where last\_name like '%e\_'")

spark.sql(query).show()

**#g. Retrieve the names of employees not having managers**

query=("select first\_name ||' ' || last\_name as EmployeeName from EMPLOYEES where manager\_id is null")

spark.sql(query).show()

**#h. Retrieve the employee name and the name of the department in which the employee is working**

query="select distinct first\_name ||' ' || last\_name as EmployeeName,department\_name from EMPLOYEES emp join DEPARTMENTS dept on emp.department\_id=dept.department\_id "

spark.sql(query).show()

**#i. List all the department ids having SIX employees**

query="select department\_id, count(distinct first\_name ||' ' || last\_name) as EmployeeCount from EMPLOYEES group by department\_id having EmployeeCount =6"

spark.sql(query).show()