

**Project 2: Employee data analysis**

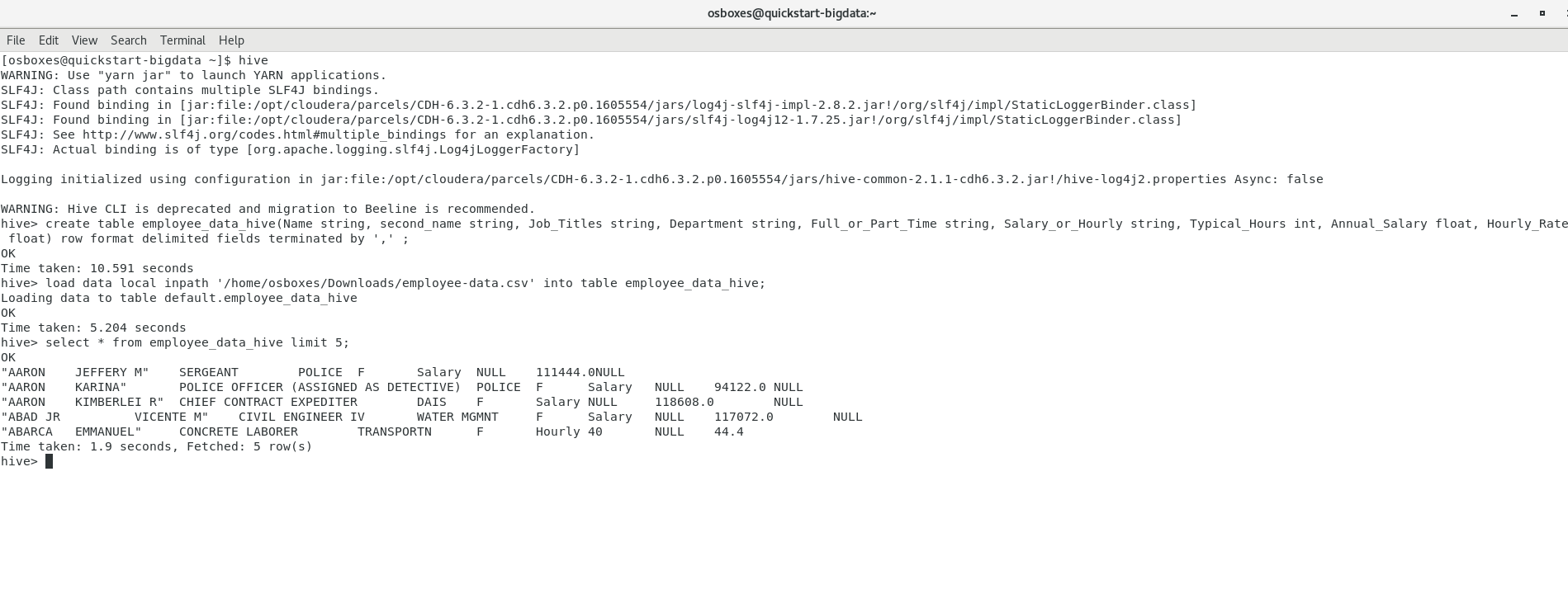
**Group 10**

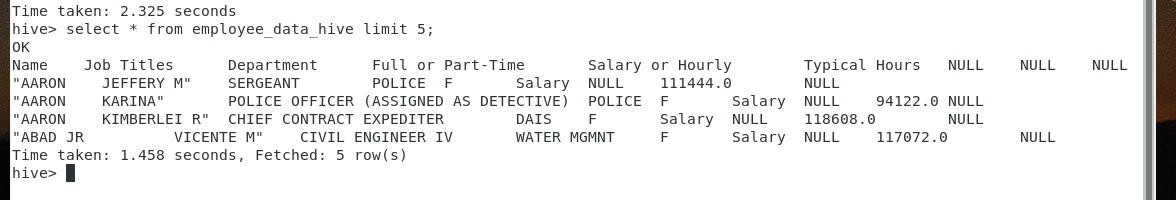
|  |  |
| --- | --- |
| **Name** | **ID** |
| Nourhan Abdelkerim | 21nmma1 |
| Ahmed Salem | 21aaeh |
| Khaled Ahmed | 21kaka |
| Mohamed Adam | 21mrma |

1. Create a Hive table named employee-data-hive based on the given dataset.

Sol:

* We open terminal then hive.
* We create table with the same columns in csv file.
* Import tale from local file on machine to hive system.
* Show first 5 rows.
* Commands:
  + create table employee\_data\_hive(Name string, second\_name string, Job\_Titles string, Department string, Full\_or\_Part\_Time string, Salary\_or\_Hourly string, Typical\_Hours int, Annual\_Salary float, Hourly\_Rate float) row format delimited fields terminated by ',' ;
  + load data local inpath '/home/osboxes/Downloads/employee-data.csv' into table employee\_data\_hive;
  + select \* from employee\_data\_hive limit 5;



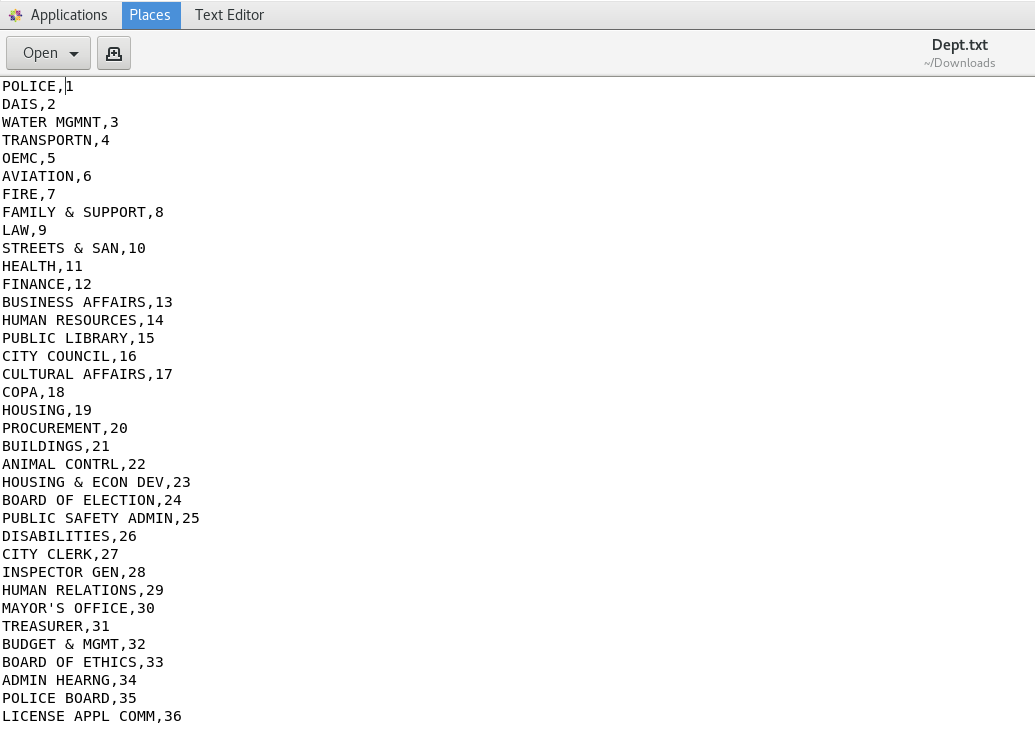


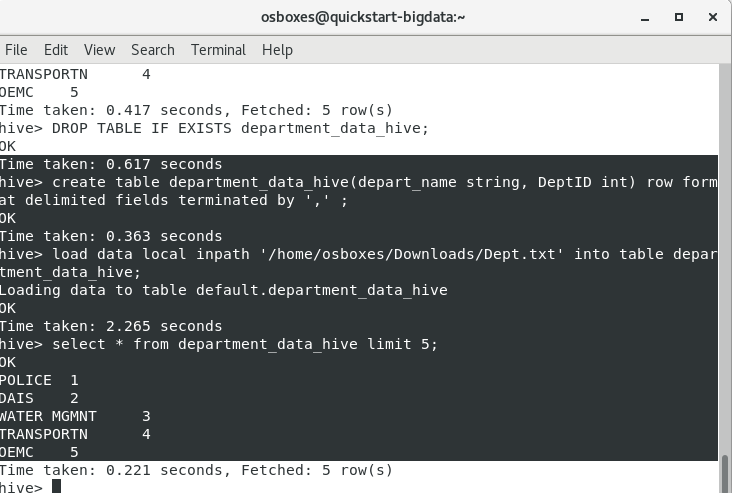
2. Create a department-data-hive table by selecting unique department names from the employee-data-hive and adding a column named deptID in the new department-data-hive table, and put unique values in the deptID column.

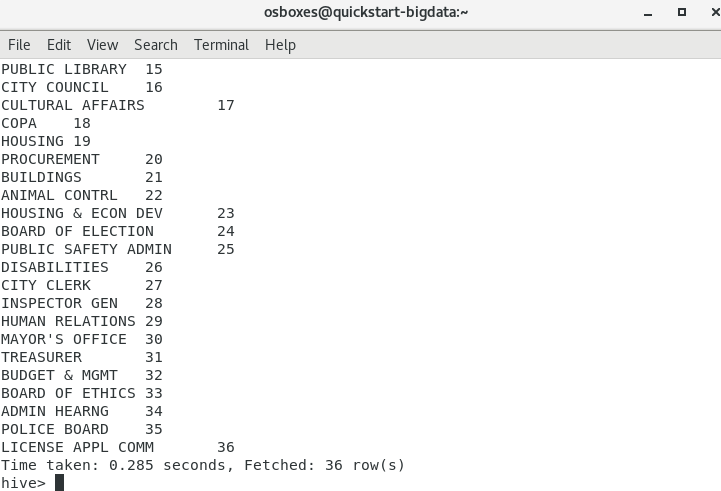
Alternatively, you can pre-process the employee-data and select the unique department names, add DeptID column and assign unique value in the new colum using excel or mySQL database separately, and then consider this structure (depart-name, DeptID) to create the department-data-hive table.

Sol.

* We used excel to create new file.
* We selected unique values from department column and give each one a specific number.
* We moved file to local machine.
* We created department\_data\_hive table with the required columns.
* Load file into table.
* Show data inside table.
* As we see, we have 36 different department.
* Commands:
  + create table department\_data\_hive(depart\_name string, DeptID int) row format delimited fields terminated by ',' ;
  + load data local inpath '/home/osboxes/Downloads/Dept.txt' into table department\_data\_hive;
  + select \* from department\_data\_hive limit 50;





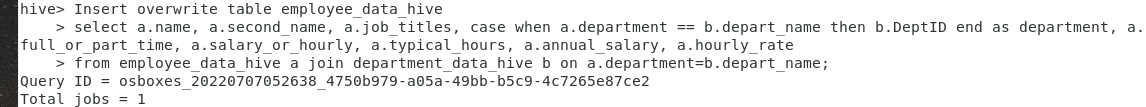


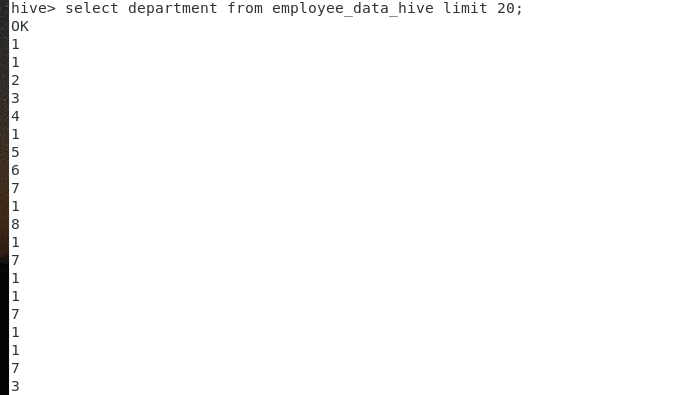
3.

a. Update the employee-data-hive table by replacing the department field data with the deptID values as created in the department-data-hive table.

Sol.

* A full join was carried out between the 2 tables on department column in employees table and depart\_name column in departments table.
* The needed columns were selected.
* Commands:
  + Insert overwrite table employee\_data\_hive select a.name, a.second\_name, a.job\_titles, case when a.department == b.depart\_name then b.DeptID end as department, a.full\_or\_part\_time, a.salary\_or\_hourly, a.typical\_hours, a.annual\_salary, a.hourly\_rate from employee\_data\_hive a join department\_data\_hive b on a.department=b.depart\_name;



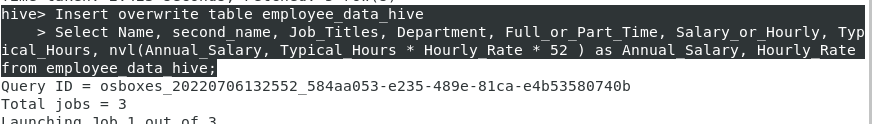


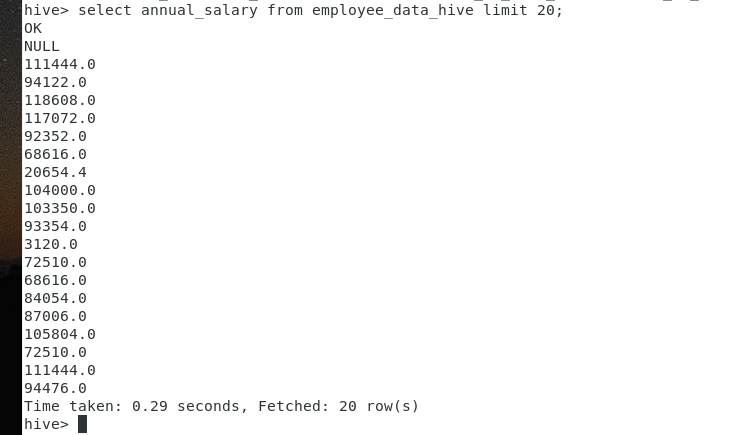
Here we selected the first 20 values from the department column in the employees table to make sure they were replaced with ID’s.

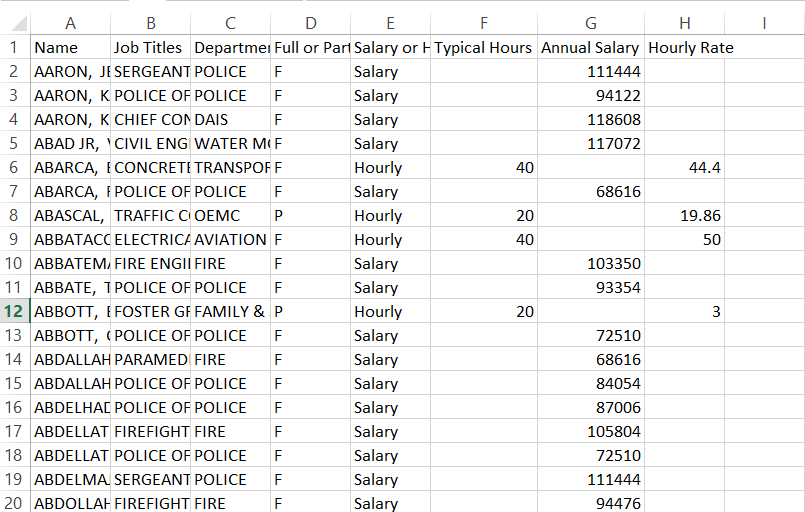
b. Also update the employee-data-hive table ‘annual salary’ field based on the ‘Typical Hours’ \* ‘Hourly Rate’ \* 52 if the annual salary field is empty.

Sol.

* We used insert overwrite to overwrite new data on annual salary column.
* We replaced nulls in this column with the value of (typical\_hour\* hourly\_rate\* 52).
* And leaved the rows that contain values as they are.
* Commands:
  + Insert overwrite table employee\_data\_hive Select Name, second\_name, Job\_Titles, Department, Full\_or\_Part\_Time, Salary\_or\_Hourly, Typical\_Hours, nvl(Annual\_Salary, Typical\_Hours \* Hourly\_Rate \* 52 ) as Annual\_Salary, Hourly\_Rate from employee\_data\_hive;







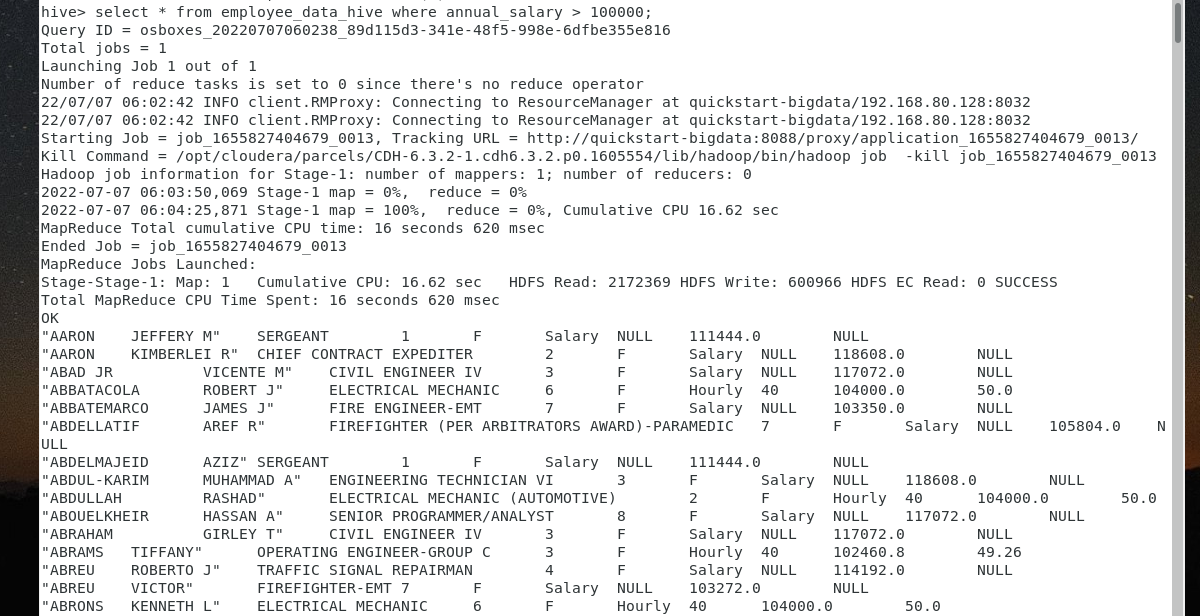
As we can see, the first 20 entries in the “Annual Salary” column had some missing values, but after using the above command and viewing the first 20 values, no null values were found. Note, the first value was null because it has the column name, but the first value is the same as in the excel screenshot.

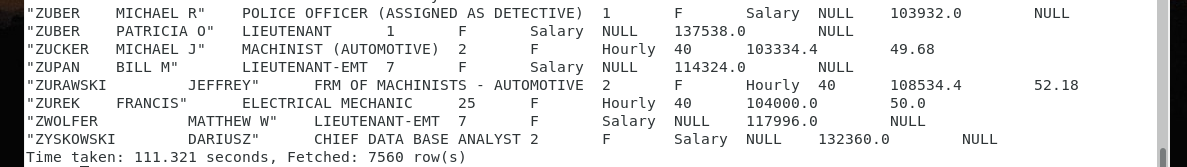
4.

a. Display all employees list with salary more than $100,000 based on employee-data-hive table.

Sol.

* Command:
  + Select \* from employee\_data\_hive limit where annual\_salary > 100000;





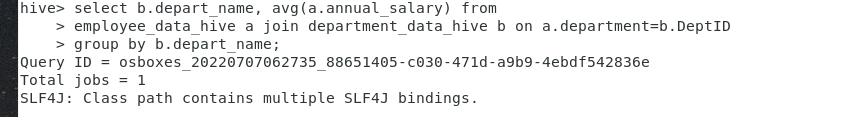
As we can see from the second screenshot, 7560 rows were selected.

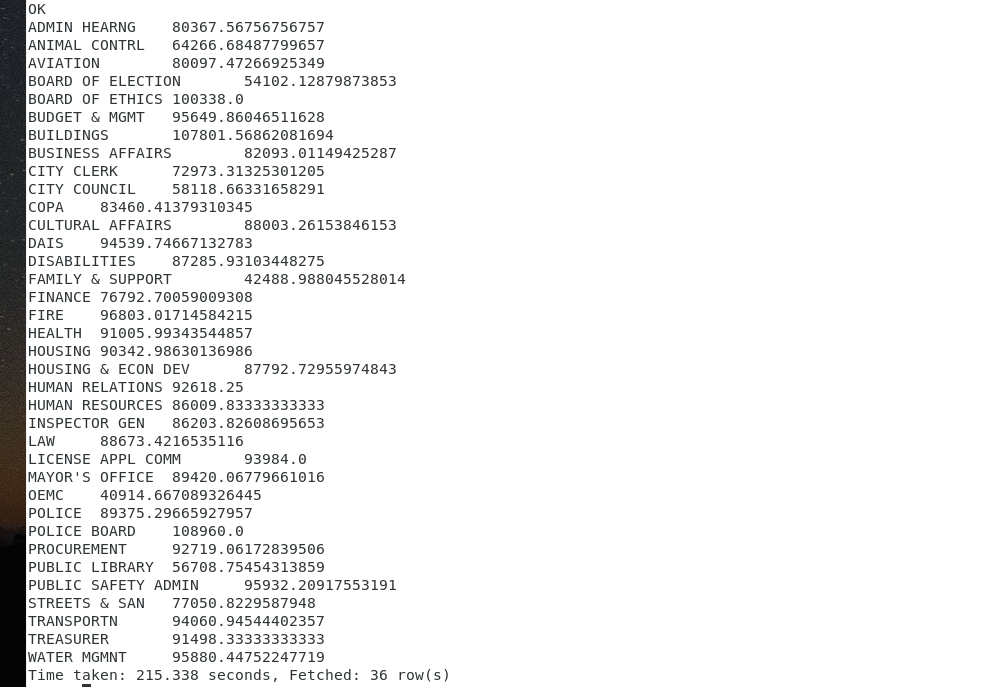
b. join the employee-data-hive and department-data-hive table to show the average salary of employees by department name

Sol.

* Commands:
  + select b.depart\_name, avg(a.annual\_salary) from employee\_data\_hive a join department\_data\_hive b on a.department=b.DeptID

group by b.depart\_name;





As shown in the previous screenshot, we have the average annual salary for each department name.

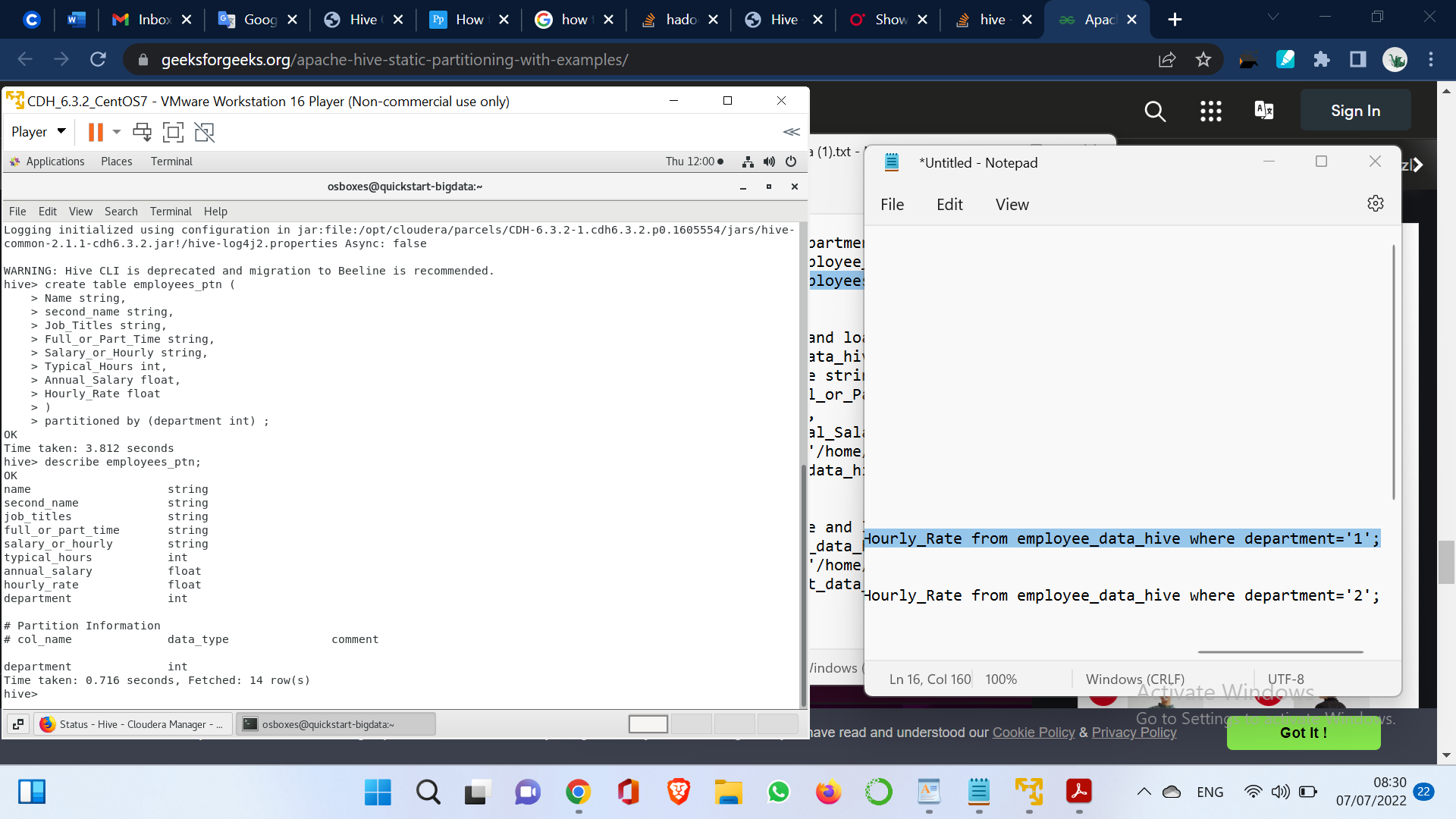
5.

1. Create 5 partitions in a employees\_ptn table to store 5 departments in the appropriate partition.

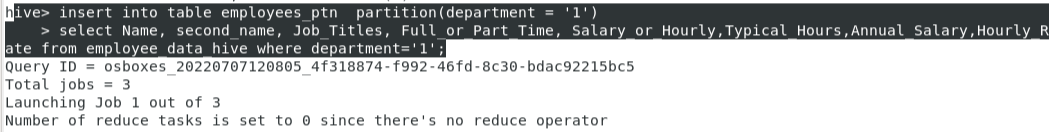
Sol.

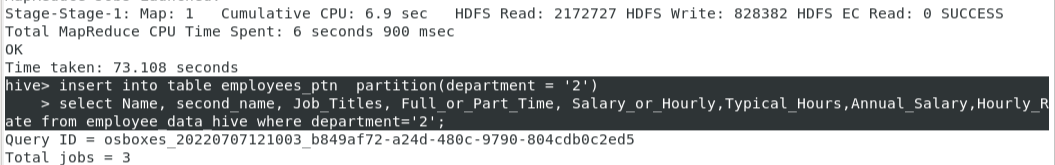
* Create new table and partition by department.
* Commands:
  + create table employees\_ptn ( Name string,second\_name string, Job\_Titles string, Full\_or\_Part\_Time string, Salary\_or\_Hourly string, Typical\_Hours int,Annual\_Salary float,Hourly\_Rate float)

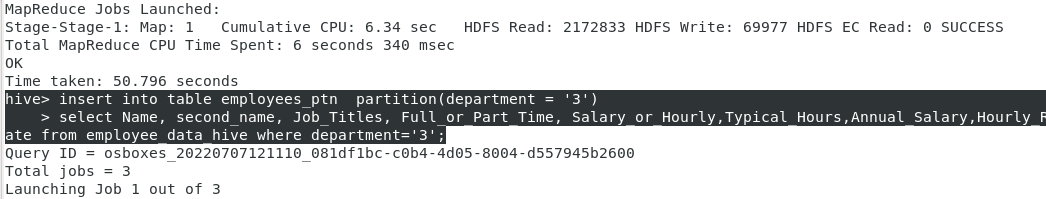
partitioned by (department int) ;

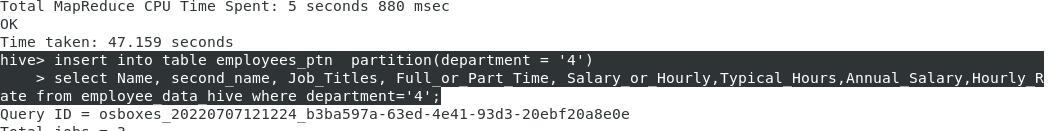


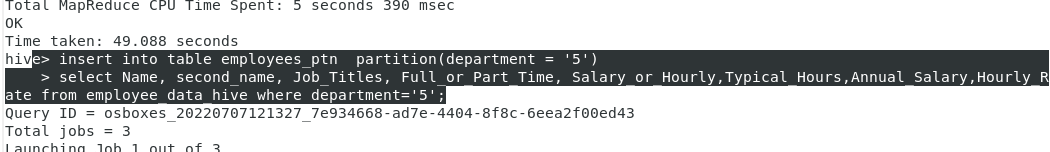
* Create partitions for first 5 department from 1 --> 5 as we converted department names to numbers



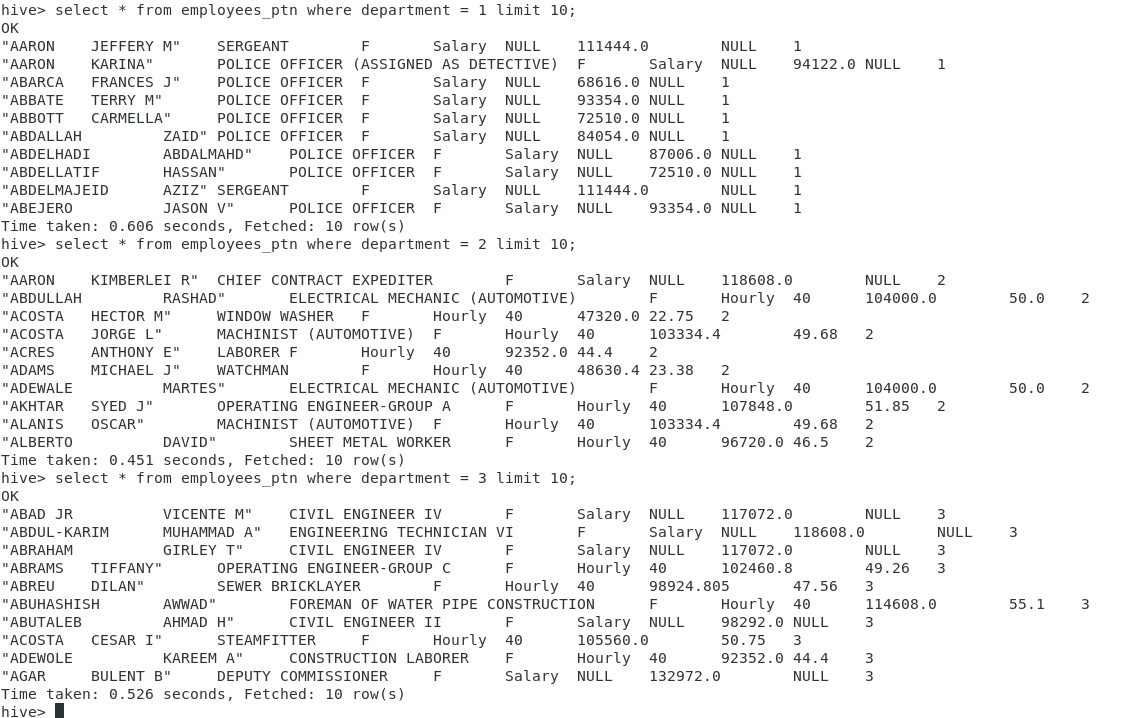


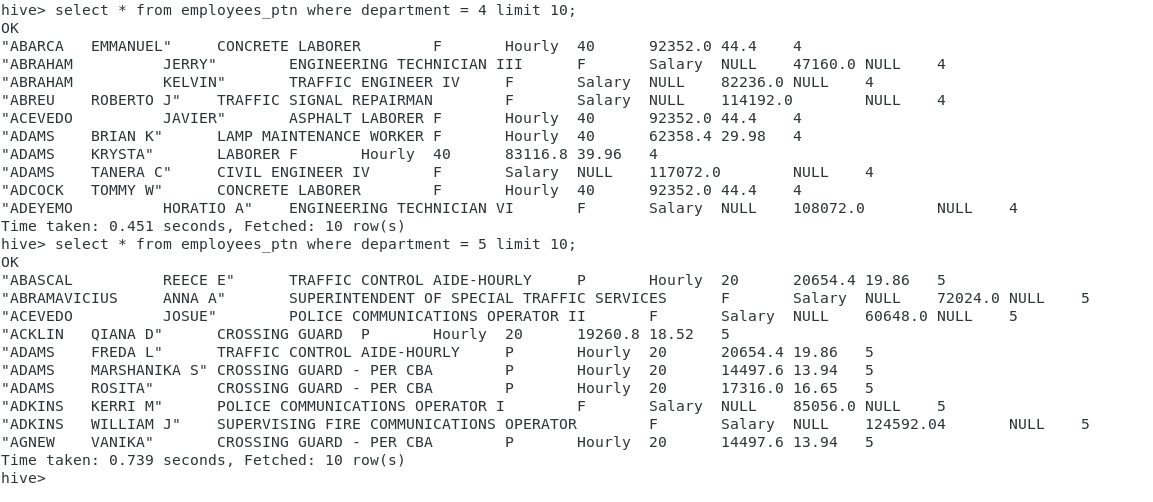


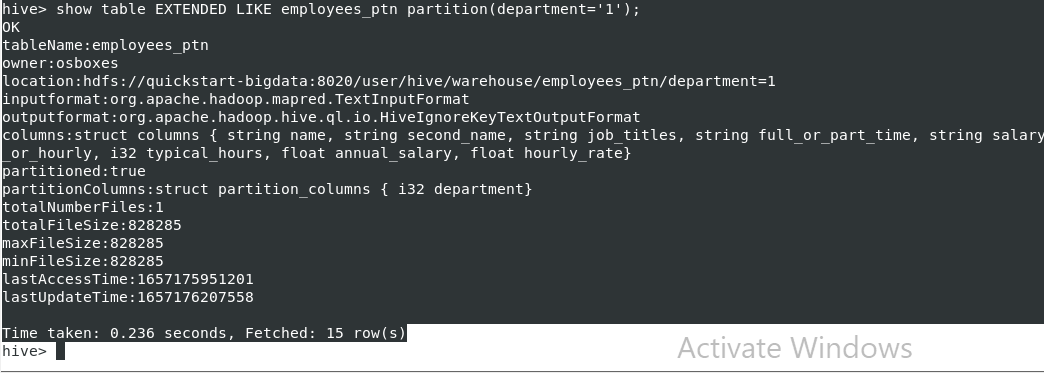


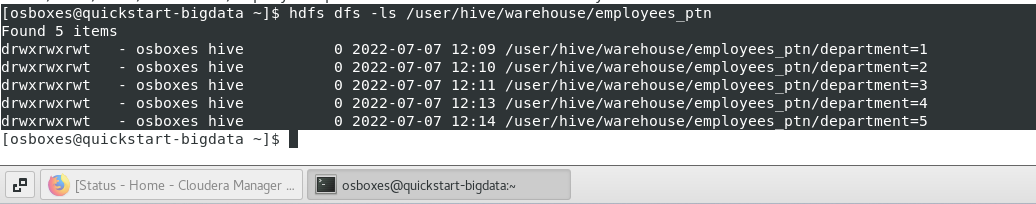


1. Display the partition structure.

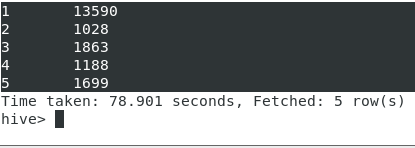










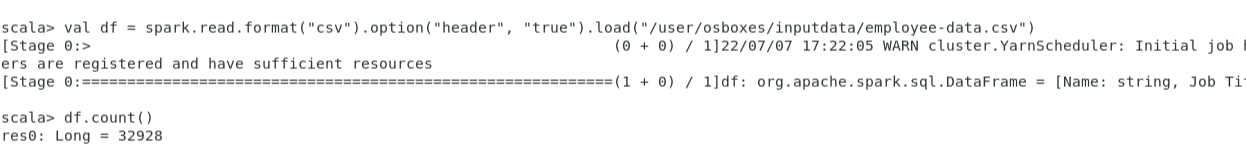


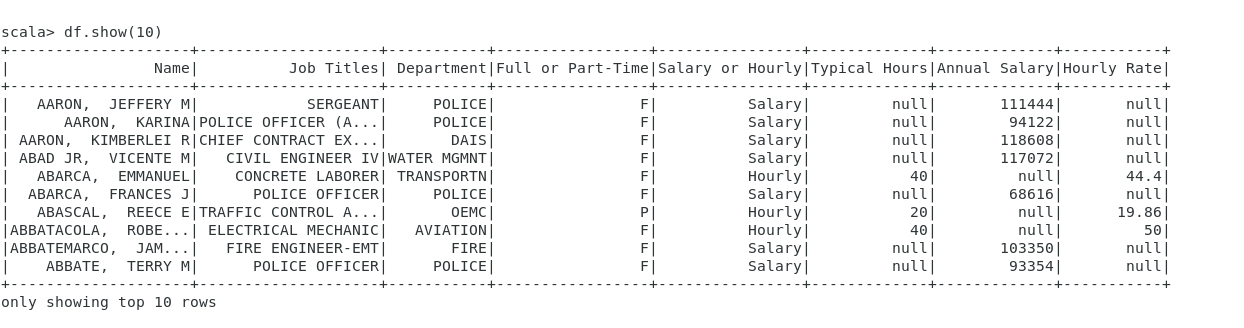
As we see, we have 5 partitions contain 5 departments.

6. Create spark DataFrame based on the given dataset. Identify # of records in the DataFrame and show top 10 records.

Sol.

* At the first we copied data from local to hdfs.
* Opened spark-shell.
* We used sc.textfile to read data from hdfs.
* We converted text data to dataframe.
* As we see, we have 32929 records.
* At the end we displayed the first 10 records of dataframe.
* Commands:
  + hdfs dfs -copyFromLocal /home/osboxes/Downloads/employee-data.csv /user/osboxes/inputdata
  + spark-shell
  + val df = spark.read.format("csv").option("header", "true").load("/user/osboxes/inputdata/employee-data.csv")
  + df.count()
  + df.show(10)





* **Workload:**

|  |  |
| --- | --- |
| Member | Steps |
| **Nourhan Abdelkerim** | 1, 2, 3.b |
| **Khaled Ahmed** | 3.a, 4 |
| **Ahmed Salem** |  |
| **Mohamed Adam** |  |