Employee’s Dataset Analysis

The dataset contains employee’s data. This data can be used to find the employees given their best and are about to leave the organization.

Understanding Data

The data format is comma separated values. There are 15000 observations.

Schema Description

- Employee Id
- Employee satisfaction level
- Last evaluation
- Number of projects
- Average monthly hours
- Time spent at the company
- Whether they have had a work accident
- Whether they have had a promotion in the last 5 years
- Role
- Salary
- Whether the employee has left

Problem Statements

1) Load the employees file from HDFS and make it an RDD & Convert the RDD into dataframe.

```
val sqlContext = new org.apache.spark.sql.SQLContext(sc)

import sqlContext.implicits._

scala> case class EmpDetails(empid: String,satisfaction_level: String, last_evaluation: String,
number_project: String, average_monthly_hours: String,time_spend_company: String,
work_accident: String, left: String, promotion_last_5years: String, role: String, salary: String)

scala> val emp_data =
sc.textFile("file:///home/cloudera/khasimbabu/Spark_Excercise/attachment_Dataset-SparkSQL-
Employee.xls").map(_.split(",")).map(i =>
EmpDetails(i(0),i(1),i(2),i(3),i(4),i(5),i(6),i(7),i(8),i(9),i(10))).toDF()
```

2) Register the temp table for the dataframe.

```
scala> emp_data.registerTempTable("EmpDetailsTable")

scala> val allrecords = sqlContext.sql("select * from EmpDetailsTable")

scala> allrecords.show()
```

4) Find the no. of people in each category and order them in descending order

```
scala> val allrecords = sqlContext.sql("select count(empid) as noofemployees,role from
EmpDetailsTable group by role order by noofemployees desc").show()
```

5) Find the no. of people whose satisfaction level is above 0.80 in each category.

```
scala> val allrecords = sqlContext.sql("select role,count(empid) as noofemployees from
EmpDetailsTable where satisfaction_level>'0.80' group by role order by noofemployees
desc").show()
```

6) How many product managers have less salary and satisfaction level lower than 0.30

```
scala> val allrecords = sqlContext.sql("select empid,satisfaction_level from EmpDetailsTable where
role='product_mng' and salary='low' and satisfaction_level<'0.30' order by satisfaction_level asc" )
```

7) Find the employee who are not fairly compensated? Given Criteria - last_evaluation score is greater than 0.75, handling more than 2 projects, have not been promoted and salary is low. Order the results in descending order by last_evaluation score

```
scala> val allrecords = sqlContext.sql("select empid,role,last_evaluation,number_project,salary from
EmpDetailsTable where last_evaluation>'0.75' and number_project>2 and
promotion_last_5years='0' and salary='low' order by last_evaluation desc").show()
```

8) Find the number of employees not fairly compensated based on role as per the criteria given in above question?

```
scala> val allrecords = sqlContext.sql("select count(empid) as noofemp,role from EmpDetailsTable
where last_evaluation>'0.75' and number_project>2 and promotion_last_5years='0' and salary='low'
group by role order by noofemp desc").show()
```

9) Find the role and number of the employee who left the company given the salary is low for each category. Order the results in descending order.

```
scala> val allrecords = sqlContext.sql("select count(empid) as noofemp,role from EmpDetailsTable
where salary='low' group by role order by noofemp desc").show()
```

10) Find the number of employees with the highest satisfaction score in each category

```
scala> val allrecords = sqlContext.sql("select count(empid) as noofemp,role from EmpDetailsTable
where satisfaction_level='1' group by role order by noofemp desc").show()
```