Group 1 Project 2

```
import org.apache.commons.io.IOUtils
import java.net.URL
import java.nio.charset.Charset
import org.apache.spark.sql.SQLContext
import org.apache.spark.sql.types.{DataType, DateType, TimestampType}
```

FINISHED

```
import org.apache.commons.io.IOUtils
import java.net.URL
import java.nio.charset.Charset
import org.apache.spark.sql.SQLContext
import org.apache.spark.sql.types.{DataType, DateType, TimestampType}
```

Took 1 sec. Last updated by anonymous at September 15 2019, 1:35:34 PM.

```
//We create our directories and load our data below.
```

```
%sh
hadoop fs -mkdir /tmp/data1
wget http://bit.ly/kagglecars -O cars.csv
hdfs dfs -put cars.csv /tmp/data1
```

```
100K .......... .......... .......... .......... ..........  0%  160M 93s
150K .......... .......... .......... .......... ..........  0% 96.7M 70s
200K .......... .......... .......... .......... ..........  0% 1.30M 74s
250K .......... .......... .......... .......... ..........  0%  129M 62s
300K .......... .......... .......... .......... ..........  0%  139M 53s
350K .......... .......... .......... .......... ..........  0%  119M 47s
400K .......... .......... .......... .......... ..........  0% 1.35M 51s
450K .......... .......... .......... .......... ..........  0%  114M 46s
500K .......... .......... .......... .......... ..........  0% 97.4M 42s
550K .......... .......... .......... .......... ..........  0%  135M 39s
600K .......... .......... .......... .......... ..........  0% 94.2M 36s
650K .......... .......... .......... .......... ..........  0%  113M 33s
700K .......... .......... .......... .......... ..........  0%  115M 31s
750K .......... .......... .......... .......... ..........  0%  162M 29s
800K .......... .......... .......... .......... ..........  0%  105M 28s
850K .......... .......... .......... .......... ..........  0% 1.41M 31s
900K .......... .......... .......... .......... ..........  0%  148M 29s
950K .......... .......... .......... .......... ..........  0% 97.4M 28s
```

Output is truncated to 102400 bytes. Learn more about **ZEPPELIN_INTERPRETER_OUTPUT_LIMIT** ×

```
ExitValue: 1
```

```
//Next, we verify the data has been loaded correctly into our specified directory.
```

```
%sh
hdfs dfs -ls /tmp/data1
```

```
Found 1 items
-rw-r--r--   1 imoca_econ hadoop  125979649 2019-09-15 17:33 /tmp/data1/cars.csv
```

Group 1 Project 2

```
//We generate a dataframe from the .csv file we loaded into our directory.
```

```
val cars_df = sqlContext.read.format("com.databricks.spark.csv").option("header", "true").optio
```

```
cars_df: org.apache.spark.sql.DataFrame = [maker: string, model: string ... 14 more fields]
```

```
//We display the schema of our directory
```

```
cars_df.printSchema()
```

```
root
 |-- maker: string (nullable = true)
 |-- model: string (nullable = true)
 |-- mileage: integer (nullable = true)
 |-- manufacture_year: integer (nullable = true)
 |-- engine_displacement: integer (nullable = true)
 |-- engine_power: integer (nullable = true)
 |-- body_type: string (nullable = true)
 |-- color_slug: string (nullable = true)
 |-- stk_year: string (nullable = true)
 |-- transmission: string (nullable = true)
 |-- door_count: string (nullable = true)
 |-- seat_count: string (nullable = true)
 |-- fuel_type: string (nullable = true)
 |-- date_created: string (nullable = true)
 |-- date_last_seen: string (nullable = true)
 |-- price_eur: double (nullable = true)
```

```
cars_df.describe().show
```

```
----------------+---------+----------+------------------+------------+------------------+------
------------+---------+--------------------+--------------------+------------------+
|summary| maker|             model|           mileage|  manufacture_year|engine_displacement|
engine_power|body_type|color_slug|          stk_year|transmission|        door_count|        se
at_count|fuel_type|        date_created|      date_last_seen|         price_eur|
+-------+------+------------------+------------------+------------------+-------------------+--
----------------+---------+----------+------------------+------------+------------------+------
------------+---------+--------------------+--------------------+------------------+
|  count|735331|            525519|            926596|            929072|             771032|
817545|   188176|         0|           1048575|      821242|           1048575|           10485
75|  1048575|             1048575|             1048575|           1048575|
|   mean|  null| 298.4261594202899|116494.24758794556|2000.2785747498579|   2118.75947690887| 1
00.2429603263429|     null|      null|2040.9940750300648|        null|3.6736592590821253| 4.897
718811995331|     null|                null|                null|19750.766370510875|
| stddev|  null|313.22005161866815|389416.89175520703|  83.2682850698299|  2200.832385113793|5
0.875414152070455|     null|      null| 351.7121841555827|        null| 0.849565665730499|0.934
```

```
1340663929358|     null|                null|                null| 640877.1143967404|
```

Group 1 Project 2

```
//We create a second dataframe to filter out the null values for maker and model, limit cars to
```

```
val cars1_df = cars_df.select("maker", "model", "mileage", "engine_displacement", "engine_power
    manufacture_year between 2000 and 2016 and engine_displacement between 800 and 10000 and en
```

```
cars1_df: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [maker: string, model: strin
g ... 6 more fields]
```

```
//We query the head of our new filtered dataframe, and the first ten rows.
```

```
cars1_df.head()
```

```
res113: org.apache.spark.sql.Row = [ford,galaxy,151000,2000,103,man,diesel,10584.75]
```

```
cars1_df.select("*")take(10)
```

```
res115: Array[org.apache.spark.sql.Row] = Array([ford,galaxy,151000,2000,103,man,diesel,10584.7
5], [skoda,octavia,143476,2000,81,man,diesel,8882.31], [skoda,fabia,167220,1400,74,man,gasolin
e,2072.54], [skoda,octavia,105389,1900,81,man,diesel,4293.12], [nissan,x-trail,149465,2500,121,
auto,gasoline,4811.25], [opel,astra,316054,1700,74,man,diesel,2331.61], [skoda,superb,269398,19
00,96,man,diesel,4663.21], [skoda,fabia,130340,1400,50,man,gasoline,2442.64], [ford,focus,22741
5,1800,85,man,diesel,2146.56], [citroen,c4-picasso,112313,1700,92,man,gasoline,7105.85])
```

```
//Below we show the statistical summary of our new filtered dataframe.
```

```
cars1_df.describe().show
```

```
+-------+------+-----------------+----------------+------------------+-----------------+---
---------+---------+-----------------+
|summary| maker|            model|         mileage|engine_displacement|     engine_power|tra
nsmission|fuel_type|        price_eur|
+-------+------+-----------------+----------------+------------------+-----------------+---
---------+---------+-----------------+
|  count|285241|           285241|          285241|            285241|           285241|
271837|   285241|           285241|
|   mean|  null|290.17969474660595|84062.59335789736|  1836.760938995446|101.19573623707672|
null|     null|13851.446262214395|
| stddev|  null| 315.7904420841931|77303.47199588586|  702.6314871087066| 50.78600854341952|
null|     null|21348.111623601177|
|    min|  audi|              100|               0|               810|              50|
auto|   diesel|             0.04|
|    max| volvo|           zafira|          999999|             10000|              999|
man| gasoline|       8404605.03|
+-------+------+-----------------+----------------+------------------+-----------------+---
```

```
//We save a temporary view of our dataframe to be able to run more operations and query it
```

Group 1 Project 2

```
cars1_df.cache()
cars1_df.createOrReplaceTempView("cars1df")
```

```
res119: cars1_df.type = [maker: string, model: string ... 6 more fields]
```

```
%sql
describe extended cars1df
```

settings

col_name	data_type
maker	string
model	string
mileage	int
engine_displacement	int
engine_power	int
transmission	string
fuel_type	string
price_eur	double

```
//We query our new dataframe to show the average price by maker. We see that lamborghinis have
```

```
spark.sql("select maker, avg(price_eur) as avgprice from cars1df group by maker order by avgpri
```

```
+------------+------------------+
|       maker|          avgprice|
+------------+------------------+
| lamborghini|145496.79582278483|
|     bentley| 99072.16486666667|
|     porsche| 63282.58178232651|
|    maserati|53957.159525222545|
|      hummer|30550.732788461537|
|       lotus|28917.460666666673|
|      jaguar|28240.747665338644|
|         bmw|27809.797418659386|
|        jeep|23107.045196989835|
|        audi|22887.456536755784|
|       lexus| 20214.97817531306|
```

```
|        isuzu|    19047.83859375|
|        dodge| 18353.48535836178|
|        volvo| 15555.19799844481|
```

Group 1 Project 2

```
//We create a table to house the information shown above.
```

```
%spark
sqlContext.sql("create table price as select maker, model, avg(price_eur) as avgprice from cars
```

```
++
||
++
++
```

```
//We query our dataframe for the average engine displacement by maker. We see that bentleys off
```

```
%spark
spark.sql("select maker, avg(engine_displacement) as avgenginedisp from cars1df group by maker
```

```
+-------------+------------------+
|        maker|     avgenginedisp|
+-------------+------------------+
|      bentley|5580.7733333333335|
|  lamborghini| 5327.151898734177|
|       hummer| 5280.826923076923|
|     maserati| 3959.379821958457|
|        dodge| 3846.377133105802|
|      porsche|3666.3042935596604|
|       jaguar| 3072.586454183267|
|        lexus|  3061.87656529517|
|         jeep|2768.9588313413014|
|     chrysler|2638.7752007136487|
|        isuzu|           2631.75|
|          bmw| 2620.702469619757|
|        rover|2222.1674008810573|
|         audi|2187.9285944352223|
|mercedes-benz|2172.4946682464456|
```

```
//We create a table to house the information above.
```

```
sqlContext.sql("create table enginetable as select maker, avg(engine_displacement) as avgengine
```

```
++
||
++
++
```

```
//We also create tables for average mileage, using the same  method as for the average engine d

sqlContext.sql("create table miletable as select maker, avg(mileage) as avgmileage from cars1df
```

Group 1 Project 2

```
++
||
++
++
```

```
//We bin our mileage in the following categories: if average mileage is less than 1,000, we con
    considered used. If average mileage is between 100,000 and 200,000, these cars are consider

sqlContext.sql("create table mileage1 as select *, if(avgmileage < 1000, 'new', if(avgmileage >
    as mileagecategory from miletable")
```

```
res134: org.apache.spark.sql.DataFrame = []
```

```
//Below we can see the number of makers who, on average, offer cars on the used market in the f

spark.sql("select count(maker), mileagecategory from mileage1 group by mileagecategory").show
```

```
+------------+---------------+
|count(maker)|mileagecategory|
+------------+---------------+
|          29|           used|
|           7|            old|
+------------+---------------+
```

```
//Next we bin our average engine displacement. Where cars have an engine displacement below 1,7
    they are considered high. Higher engine displacement is associated with engine power, but a

sqlContext.sql("create table enginedisp1 as select *, if(avgenginedisp < 1700, 'low', if(avgeng
```

```
res138: org.apache.spark.sql.DataFrame = []
```

```
//Like before, we can see how many makers, on average offer cars in each category of engine dis

spark.sql("select count(maker), enginedispcategory from enginedisp1 group by enginedispcategory
```

```
+------------+------------------+
|count(maker)|enginedispcategory|
+------------+------------------+
|          11|               low|
|           8|          moderate|
|          17|              high|
```

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

Group 1 Project 2

```
%spark
sqlContext.sql("create table pricetable as select maker, avg(price_eur) as avgprice from cars1d
```

```
res142: org.apache.spark.sql.DataFrame = []
```

```
//We consider the use case where a powerful engine is important, and it is okay to sacrifice fu
    that Chrylser and Rover offer the lowest average prices for powerful cars.
```

```
spark.sql("select distinct enginedisp1.maker, pricetable.avgprice, mileage1.mileagecategory fro
    'high') join mileage1 on (mileage1.maker = pricetable.maker) order by pricetable.avgprice a
```

```
+-------------+------------------+---------------+
|        maker|          avgprice|mileagecategory|
+-------------+------------------+---------------+
|     chrysler|   5962.1179750223|            old|
|        rover| 8473.823612334803|            old|
|       subaru|12100.513452950554|            old|
|mercedes-benz|14503.765068127968|            old|
|        volvo| 15555.19799844481|            old|
|        dodge| 18353.48535836178|           used|
|        isuzu|    19047.83859375|           used|
|        lexus| 20214.97817531306|            old|
|         audi|22887.456536755784|           used|
|         jeep|23107.045196989835|           used|
|          bmw|27809.797418659386|           used|
|       jaguar|28240.747665338644|           used|
|       hummer|30550.732788461537|           used|
|     maserati|53957.159525222545|           used|
|      porsche| 63282.58178232651|           used|
```

```
//Next we consider the moderate engine displacement category and repeat the analysis above. Her
```

```
spark.sql("select distinct enginedisp1.maker, pricetable.avgprice, mileage1.mileagecategory fro
    'moderate') join mileage1 on (mileage1.maker = pricetable.maker) order by pricetable.avgpri
```

```
+----------+------------------+---------------+
|     maker|          avgprice|mileagecategory|
+----------+------------------+---------------+
|mitsubishi| 9547.702741005232|           used|
| chevrolet| 9708.739462827676|           used|
|      ford| 9787.903873268231|           used|
|     honda|10143.739616817455|           used|
|     mazda|10985.300389291086|           used|
|       kia| 12198.09159401309|           used|
|    nissan|12491.097930442207|           used|
|     lotus|28917.460666666673|           used|
+----------+------------------+---------------+
```

```
//Lastly, we consider the use case for low engine displacement. This group of cars will not off
    Lancia in the used car category. As a result, our top five manufacturers to recommend are C
```

Group 1 Project 2

```
spark.sql("select distinct enginedisp1.maker, pricetable.avgprice, mileage1.mileagecategory fro
    'low') join mileage1 on (mileage1.maker = pricetable.maker) order by pricetable.avgprice as
```

```
+-------+-----------------+---------------+
|  maker|         avgprice|mileagecategory|
+-------+-----------------+---------------+
| lancia| 8115.453021015773|           used|
|  smart| 8205.928553592476|           used|
|   fiat|  8245.20742990228|           used|
| suzuki| 8311.505395170154|           used|
|  skoda| 9626.388986586233|            old|
|   opel| 9926.983863652575|           used|
|citroen| 10219.25581365317|           used|
| toyota|10273.669870485051|           used|
|hyundai|10525.306437118094|           used|
|   seat|11208.510258813016|           used|
|   mini|13424.097270660823|           used|
+-------+-----------------+---------------+
```

```
//Lastly we query average price by fuel consumption for each maker. On average, cars that take
```

```
spark.sql("select fuel_type, avg(price_eur) as avgprice from cars1df group by fuel_type").show
```

```
+---------+-----------------+
|fuel_type|         avgprice|
+---------+-----------------+
| gasoline|12088.256722116112|
|   diesel|15829.990701277055|
+---------+-----------------+
```