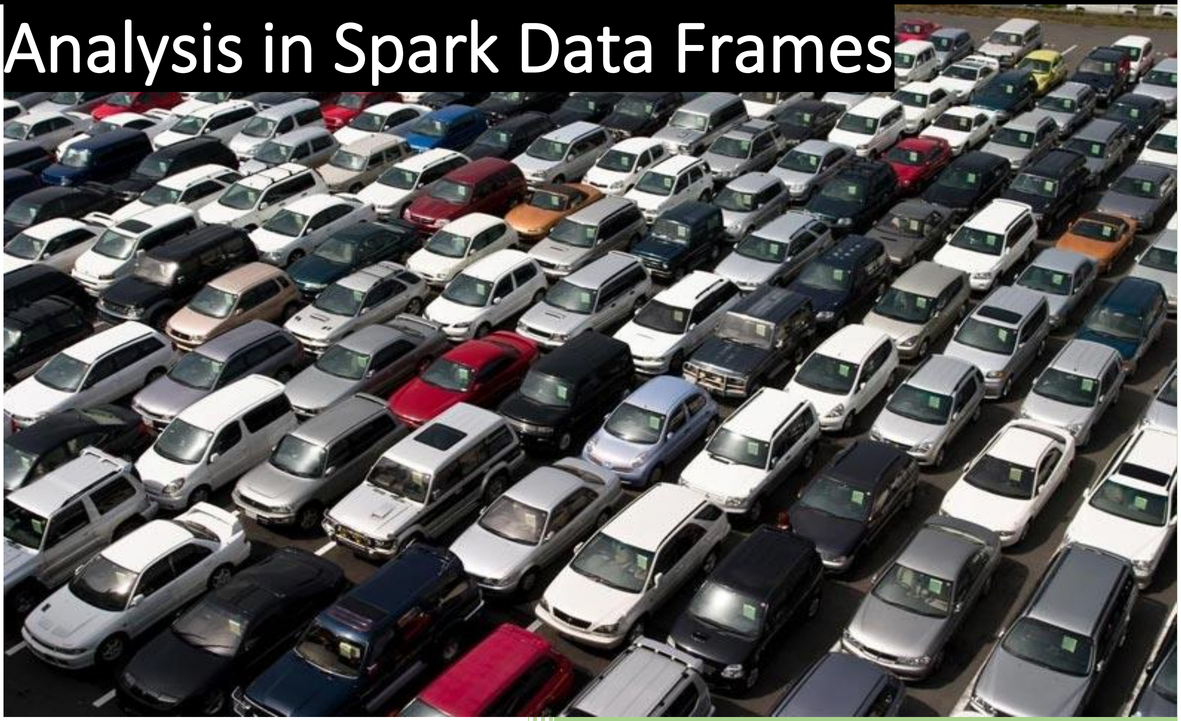
2022

Business Plan: Used Cars Dealership

Analysis in Spark Data Frames



VISHAL KAATAL

3/14/2022

TABLE OF CONTENTS

TABLE OF FIGURES 2

INTRODUCTION 3

DATA CLEANING 4

ANALYSIS RESULT 5

CONCLUSION 10

BIBLIOGRAPHY 11

APPENDIX 12

TABLE OF FIGURES

Figure 1: Inventory of used cars...5

Figure 2: Top 10 highest average prices of a car 6

Figure 3: Top 10 lowest average prices of car 6

Figure 4: Top 10 most popular car makers 7

Figure 5: Top 5 Economic segment cars 7

Figure 6: Top 5 Intermediate segment cars..8

Figure 7: Luxury Segment cars 8

Figure 8: Top 10 car makers transmission type 9

Figure 9: Number of seats preferred in a car...9

INTRODUCTION

The used car business is booming worldwide because people invest their money in properties, stocks, businesses instead of brand-new cars. (DriveNation, 2019) The brand-new vehicle loses over 10% of its value during the first month of purchase and up to 20% within the first year; therefore, people are leaning towards used cars to save these losses. A market survey has been conducted by Miroslav Zoricak and published the dataset called cars.csv on the Kaggle website under the name as classified ads for cars (Zoricak, 2017) which is the basis of this analysis report for setting up a used car dealership in the Czech Republic and Germany.

This report also covers cleaning the datasets based on the Business Plan requirement. The visualization graphs have been provided to understand the data to make an investment decision. Some of the analysis questions that have been discussed in the report are as follows:

1. How many cars are available in each vehicle manufacturing year between 2000 – 2017?
2. What is the top 10 highest average price of a vehicle based on make and model?
3. What is the top 10 lowest average price of a car based on make and model?
4. Describe and identify the top 5 make and models based on the average price of a car for three different segments as Economic, Intermediate and Luxury?
5. What kind of transmission do people prefer in the Czech Republic and Germany?
6. What are the top 5 vehicle manufacturers recommended to invest in? Why?

DATA CLEANING

(Zoricak, 2017) The collected data has been scraped over one year, and sources are completely unstructured, so as a result, the information is missing values, and some values are wrong (E.g., phone numbers scraped as mileage or price). Therefore, the dataset must be cleaned to answer the analysis questions described earlier. The dataset consists of roughly 3.5 million rows and 16 columns. The 16 columns consist of entities such as maker, model, mileage, manufacture year, engine displacement, engine power, body type, colour type, last emission check year, transmission, door count, seat count, fuel type, data collection date, car last seen on the website, and price in euro. Out of 16 columns, only 13 answered the business plan in the cleaning and analysis because 50% or more values were missing in those removed columns. The cleaning steps have been discussed in the Appendix with details.

To start a used car business, selecting the famous car manufacturers (makers) is essential such that the customers are attracted to your dealership and ultimately make their purchase. Therefore, the manufacturing year for the car has to be between 2000 and 2017 included with a price range from €3000 to €2,000,000. Furthermore, it is divided into three segments. First is Economic segment customers, who would like to spend between €3000 - €20,000. Second is Intermediate segment customers, who have a budget for a car between €20,000 – €300,000. Finally, the Luxury segment customers can spend between €300,000 – €2,000,000 for a luxury vehicle. Hence, some people might prefer specific transmission or number of doors in a car. Therefore, Using these criteria, customers will be happy to select their dream car based on their vehicle's budget, requirement, and purpose, like a daily commute or high-end luxury cars for recreational purposes, etc.

ANALYSIS RESULT

This part of the report answers the earlier analysis questions to make an investment decision.

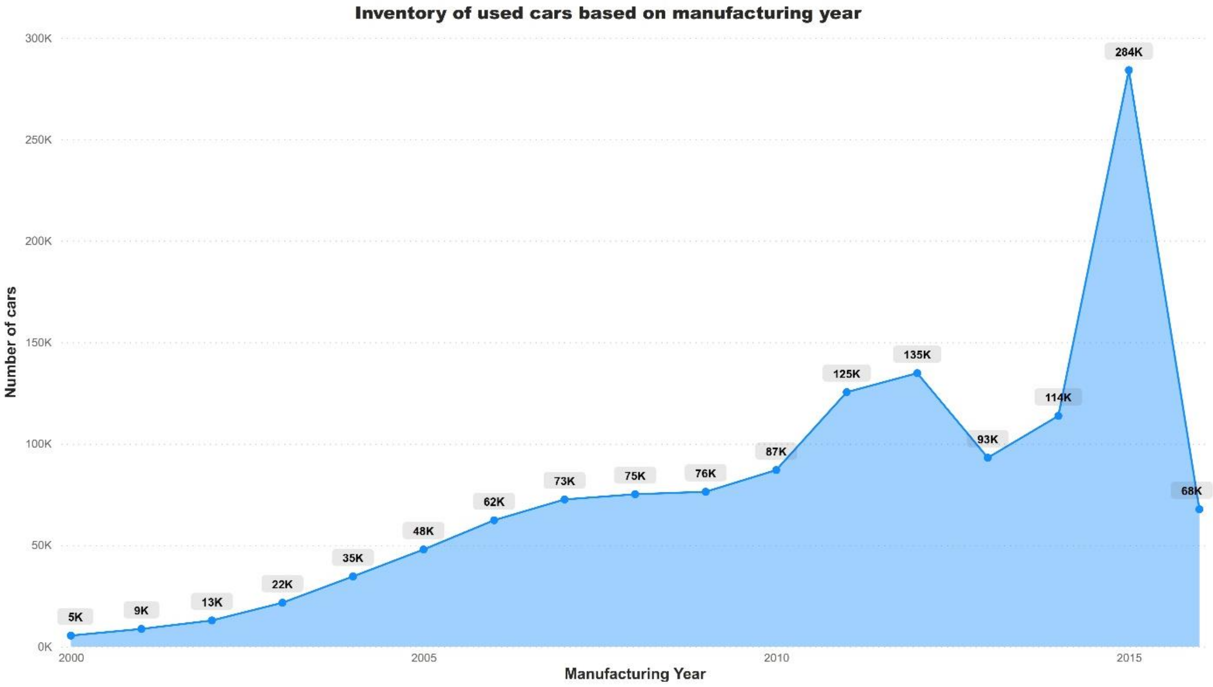


Figure 1: Inventory of used cars

Based on Figure 1, it can be stated that most used cars are available for the manufacturing year 2015. If anyone is looking for a newer used car, they should look for 2010 – 2016 maker cars with low mileage, better transmission quality, and low price.

Figure 2 represents the Top 10 average prices of a car based on maker and model, out of which Lamborgini Aventador has the highest average price of € 365,961. Lamborgini, Porche, BMW, Tesla, Bentley, and Rolls-Royce are the most expensive makers. On the other hand, figure 3 represents the Top 10 lowest average car prices. The least expensive car is the Skoda Galaxy, based on the analysis with an average value of €3071.80, whereas the most miniature makers are Fiat, Opel, Skoda. However, the least expensive ones are the most popular makers such as Volkswagen, Audi, Opel, Ford, Skoda and have the highest inventory count, as shown in Figure 4.

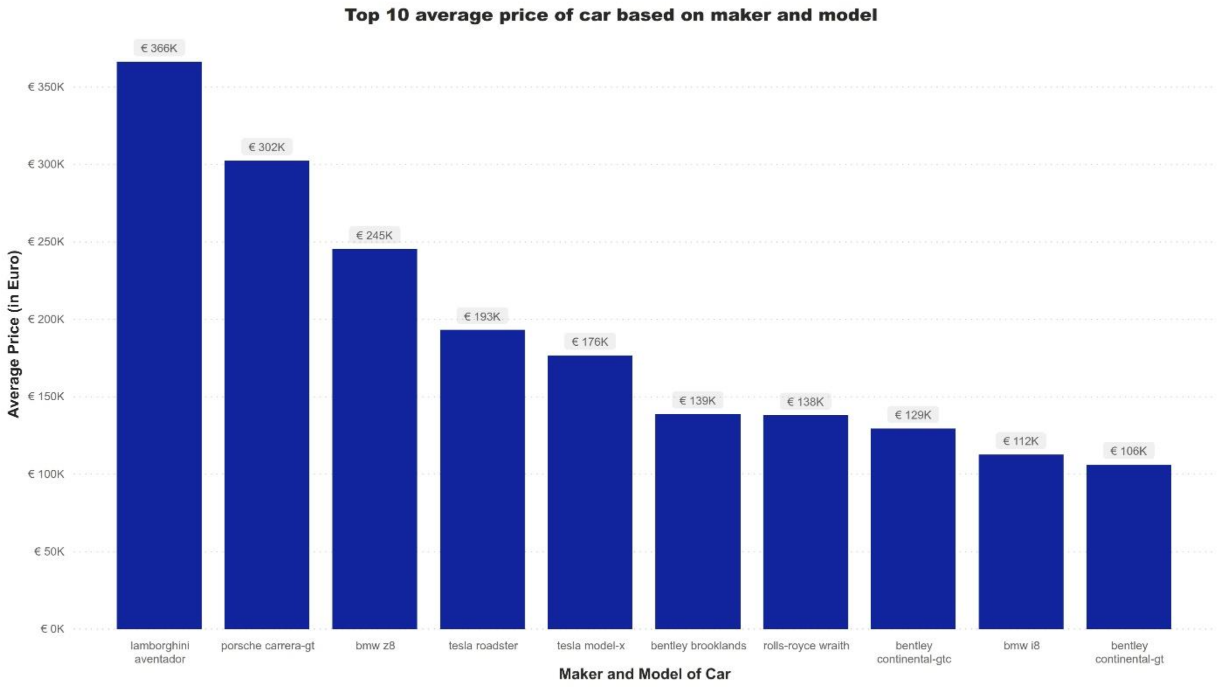


Figure 2: Top 10 highest average prices of a car

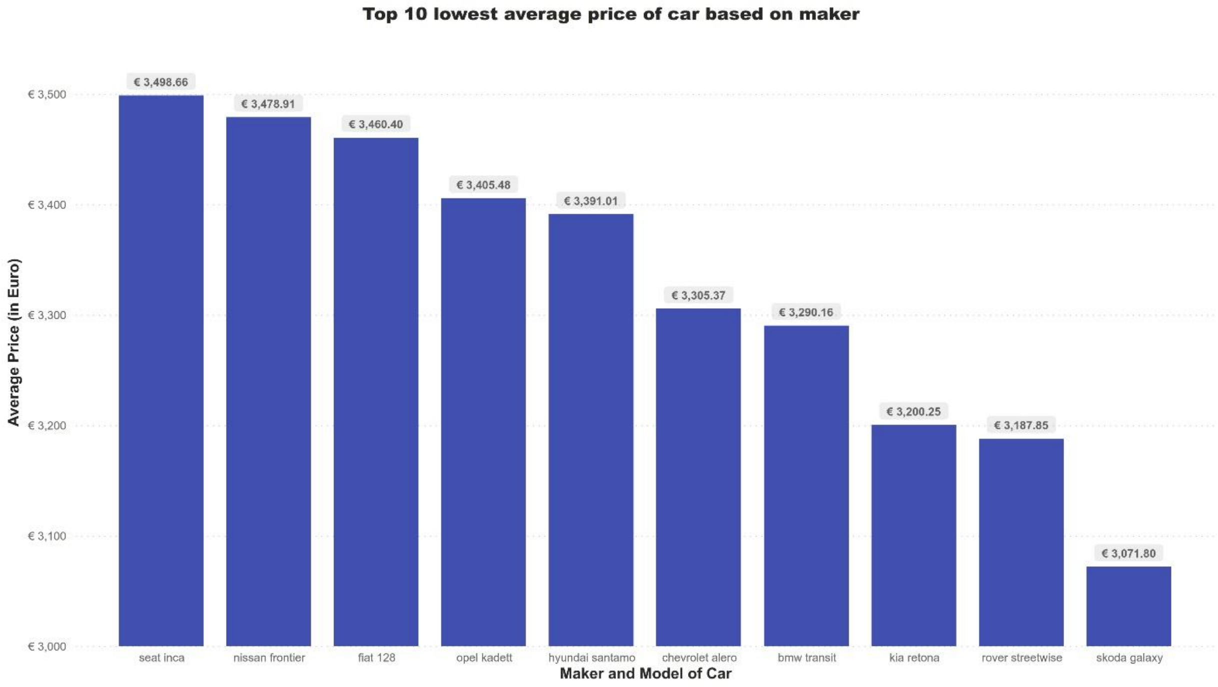


Figure 3: Top 10 lowest average prices of car

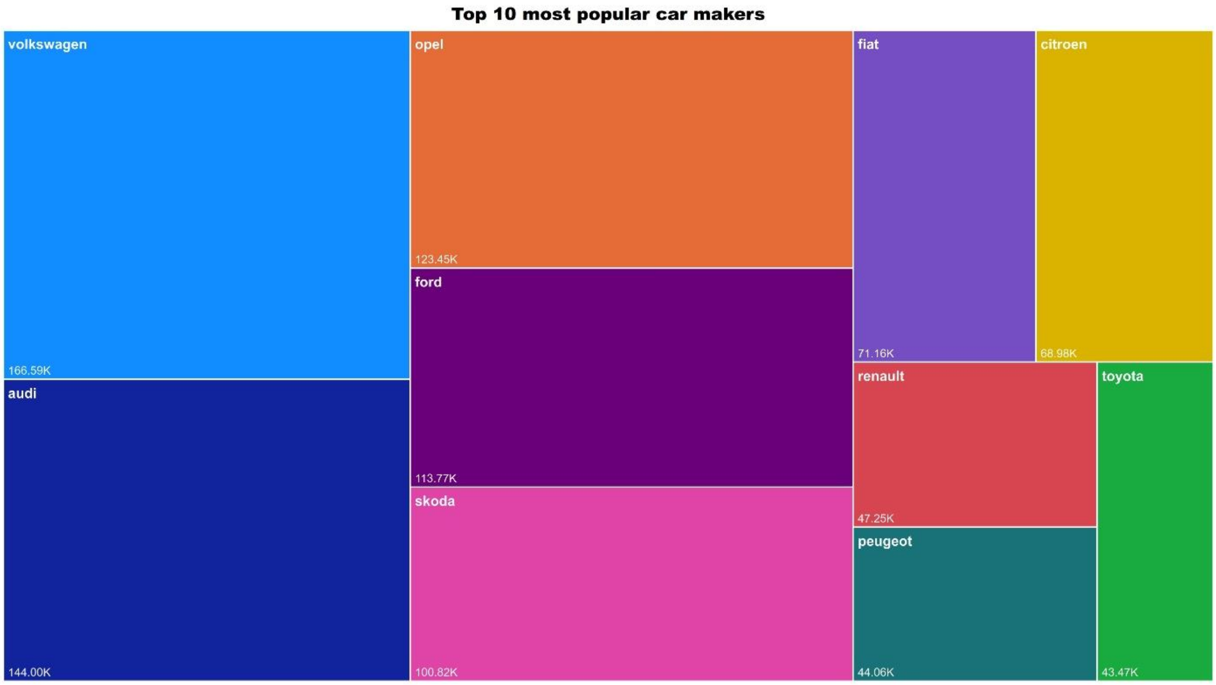


Figure 4: Top 10 most popular car makers

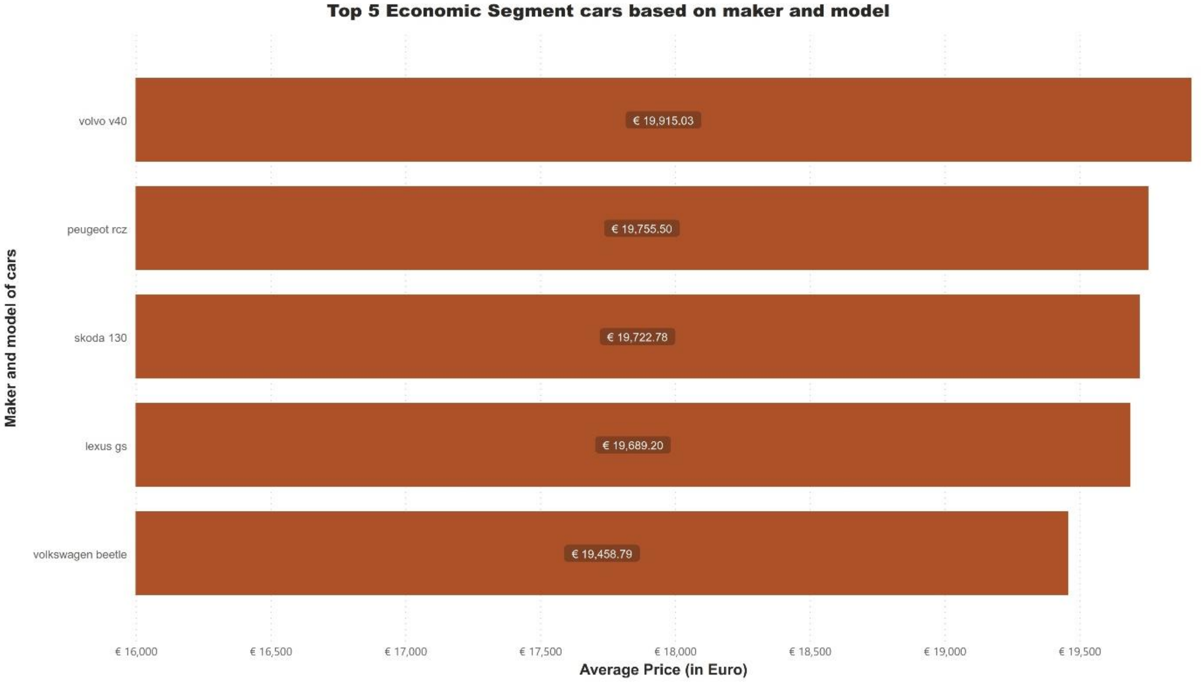


Figure 5: Top 5 Economic segment cars

The average price can be from € 3000 - € 2,000,000 for the customers based on their needs and budget. It has been divided into three segments of this analysis: Economic (€ 3000 - € 20,000), Intermediate (€ 20,000 - € 300,000) and Luxury (€ 300,000 - € 2,000,000). Figure 5 represents the Top 10 Economic segments cars with their maker and model: Volvo V40, Peugeot RCZ, Skoda 130, Lexus GS, and Volkswagen Beetle. Figure 6 shows the Top 5 Intermediate segment cars with their maker and model: BMW Z8, Tesla Roadster, Tesla Model-X, Bentley Brooklands and Rolls-Royce Wraith. In the last, the Luxury segment has only two cars: Lamborgini Aventor and Porsche Carrera-GT, as shown in figure 7.

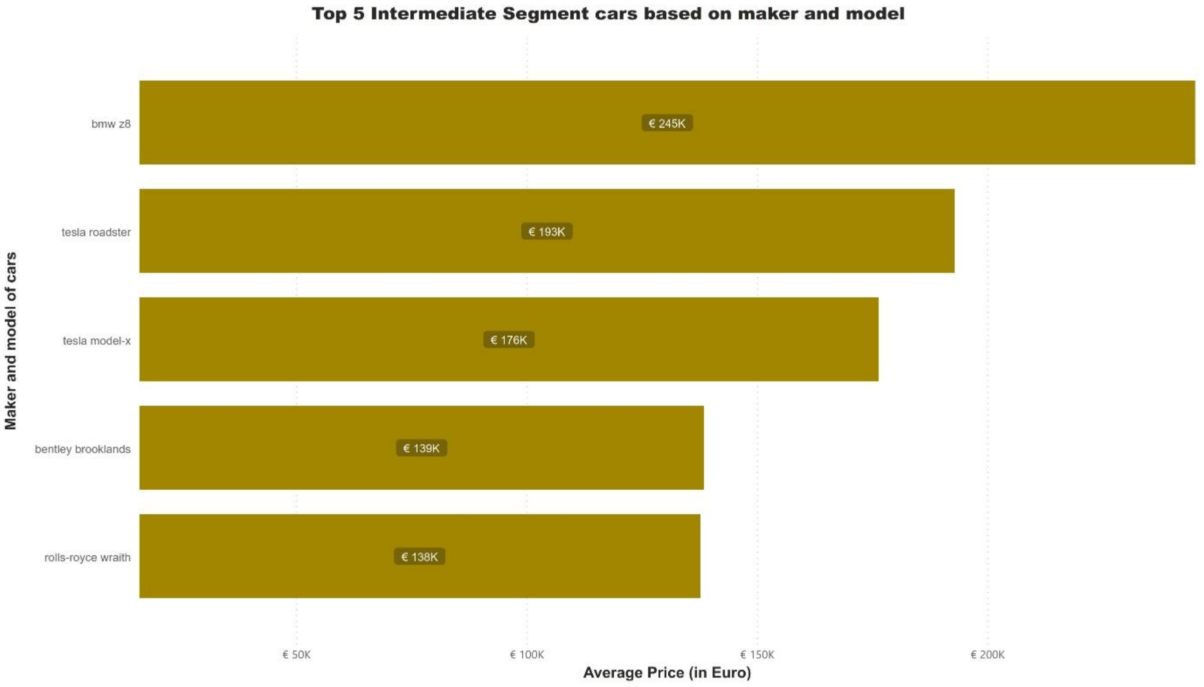


Figure 6: Top 5 Intermediate segment cars

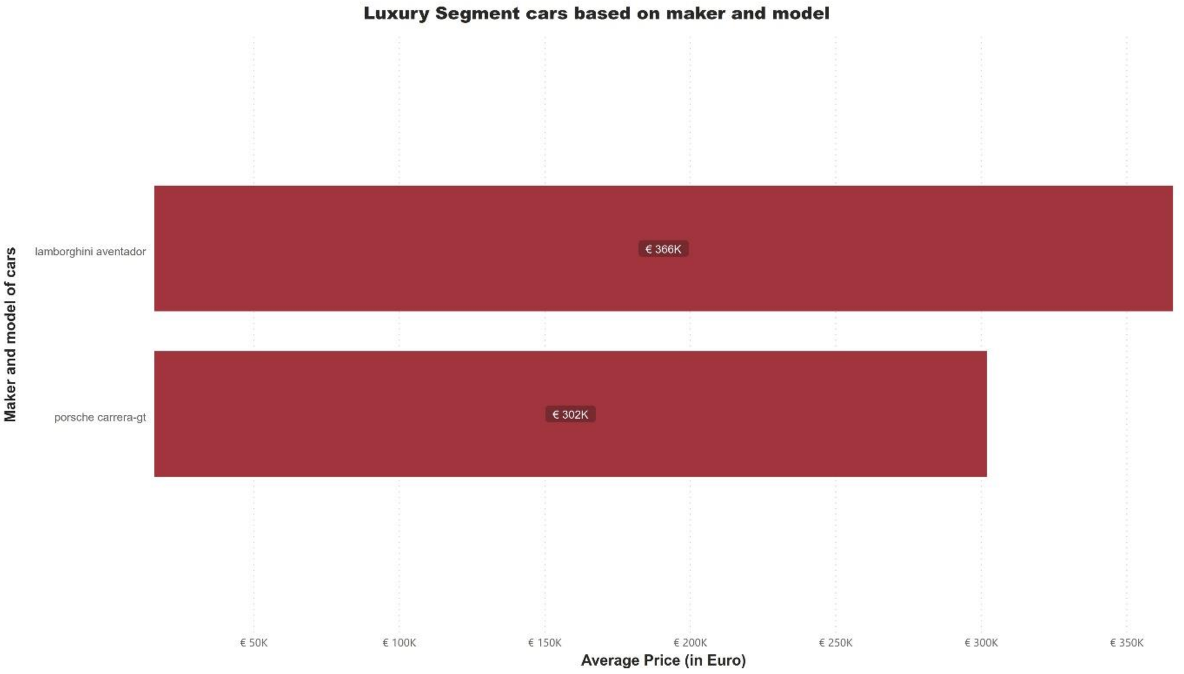


Figure 7: Luxury Segment cars

From Figure 8, it can be derived that many cars in the Czech Republic and Germany are manual cars instead of automatic. Also, only two significant carmakers, Audi and Volkswagen, produce most cars with automatic transmission, whereas others tend to make manual cars. The reason behind manual vs automatic is the cost (price); as for all carmakers, cars with automatic transmission are more expensive than those with manual transmission. Also, the maximum number of cars produced in Manual transmission are Volkswagen, Opel, Ford and Skoda from these manufacturers.

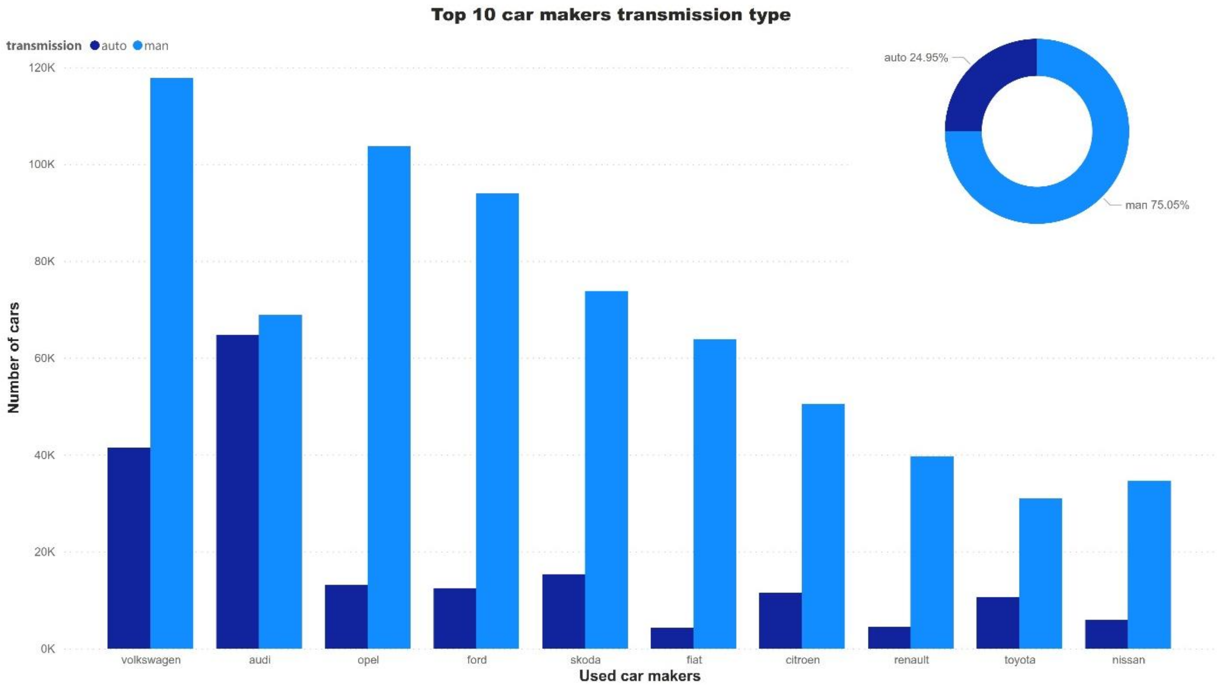


Figure 8: Top 10 car makers transmission type

Lastly, in figure 9, we notice that most manufacturers produce cars with a maximum of 5 seats, equivalent to more than 75 % of the market.

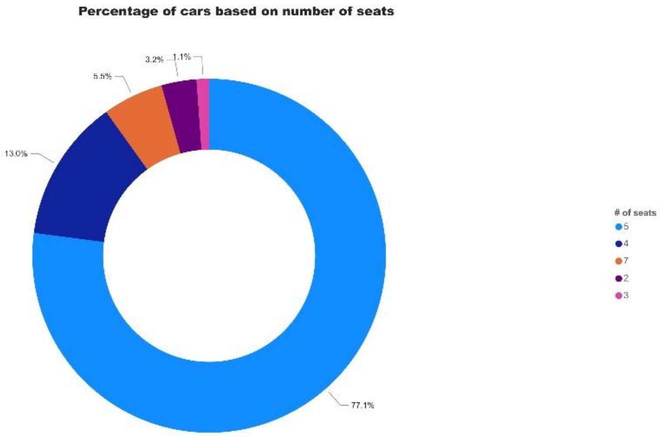


Figure 9: Number of seats preferred in a car

CONCLUSION

In general, any business plan has risks, whether its business model will be successful or not. During the analysis, it is clear that if the desired selection of used car maker, model, and manufacturing year is selected, the business shall not fail. All the parameters have been discussed in previous chapters, and appropriate graphs have been generated such that visualization should give us a better understanding. The most crucial part of data cleaning has been done using Spark such that the decisions can be data-driven for the investment in the operated car dealership business.

It can be concluded that if the investment firm is investing in this business plan, it can generate a considerable profit quickly. Therefore, to make it successful, the car manufacturers selected are Audi, Skoda, Ford, Seat, BMW, and Opel manufactured between 2000 – 2016 with a number of car seats above 4 with any transmission type. However, to offer some high-budget buyers, the dealership can include manufacturers such as Lamborghini, Porsche, Tesla, Bently, and Rolls-Royce to attract customers into the dealership. Based on the sales pitch, deals can be secured to earn more profit from Intermediate and luxury cars. The best practice is to include variety and price range so customers come to the dealership. The salesperson can secure them a deal that will make them happy and comfortable to buy again in the future.

BIBLIOGRAPHY

DriveNation. (2019, 05 03). Retrieved from drivenation.ca/blog: https://www.drivenation.ca/blog/depreciation-vehicle-much-money-will-lose-buying-new-car/

Zoricak, M. (2017, 03 16). *Classified Ads for Cars*. Retrieved from Kaggle: https://www.kaggle.com/mirosval/personal-cars-classifieds

APPENDIX

////Loading of Data in GCP///

```
wget https://www.dropbox.com/s/rsrxro7r1c5a4i2/cars.csv
```

////Moving Data into HDFS////

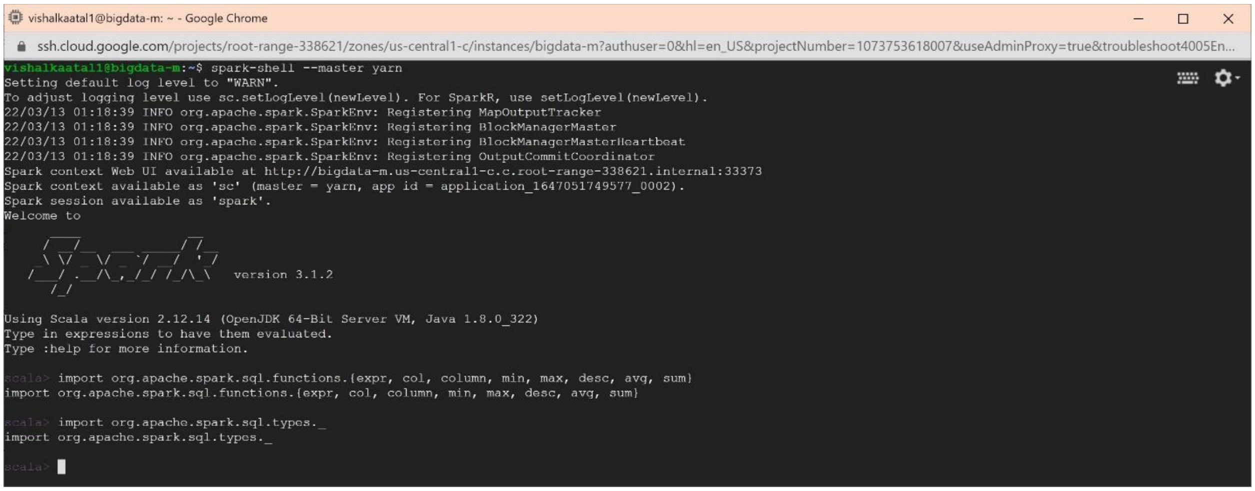
```
hadoop fs -mkdir /BigData

hadoop fs -copyFromLocal cars.csv /BigData/.
```

////Analysis in Spark DataFrame////

```
spark-shell --master yarn
```

/* Run import command for sql functions */

```
import org.apache.spark.sql.functions.{expr, col, column, min, max, desc, avg, sum}

import org.apache.spark.sql.types._
```


vishalkaatal1@bigdata-m: ~ - Google Chrome
ssh.cloud.google.com/projects/root-range-338621/zones/us-central1-c/instances/bigdata-m?authuser=0&hl=en_US&projectNumber=1073753618007&useAdminProxy=true&troubleshoot4005En...

```
vishalkaatal1@bigdata-m:~$ spark-shell --master yarn
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
22/03/13 01:18:39 INFO org.apache.spark.SparkEnv: Registering MapOutputTracker
22/03/13 01:18:39 INFO org.apache.spark.SparkEnv: Registering BlockManagerMaster
22/03/13 01:18:39 INFO org.apache.spark.SparkEnv: Registering BlockManagerMasterHeartbeat
22/03/13 01:18:39 INFO org.apache.spark.SparkEnv: Registering OutputCommitCoordinator
Spark context Web UI available at http://bigdata-m.us-central1-c.c.root-range-338621.internal:33373
Spark context available as 'sc' (master = yarn, app id = application_1647051749577_0002).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.1.2
      /_/

Using Scala version 2.12.14 (OpenJDK 64-Bit Server VM, Java 1.8.0_322)
Type in expressions to have them evaluated.
Type :help for more information.

scala> import org.apache.spark.sql.functions.{expr, col, column, min, max, desc, avg, sum}
import org.apache.spark.sql.functions.{expr, col, column, min, max, desc, avg, sum}

scala> import org.apache.spark.sql.types._
import org.apache.spark.sql.types._

scala> 
```

Task 1: Write a Spark DataFrames query to create a table called ***used_cars*** *from data. Use a schema that is appropriate for the column headings*

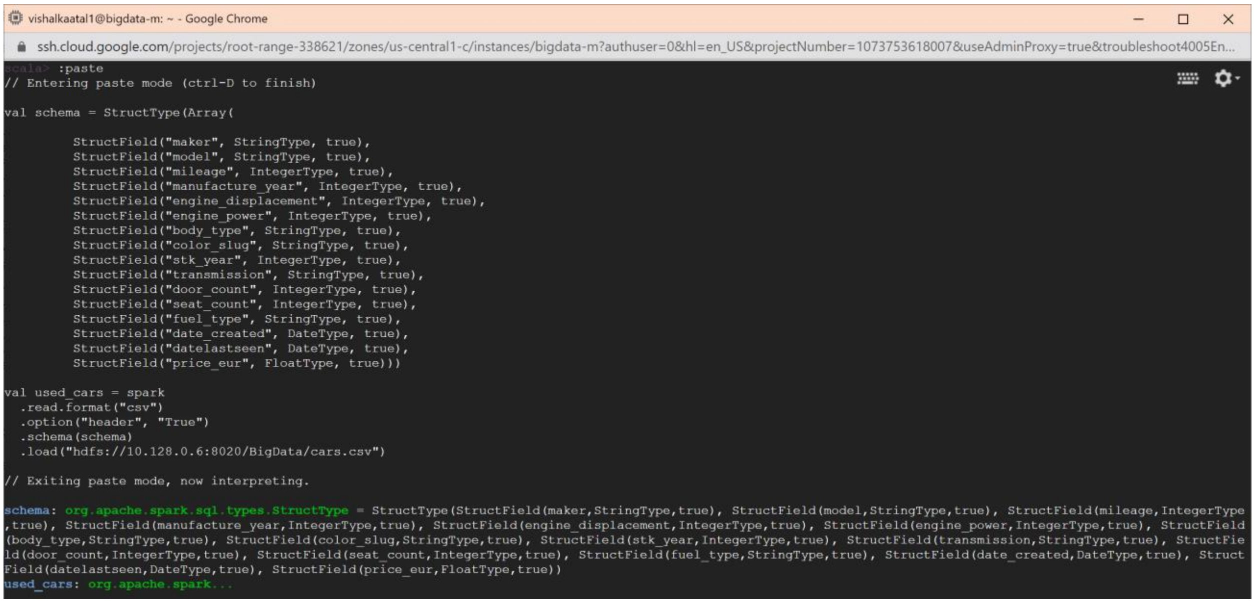
/* Loading of a cars dataset using schema in Spark as */

-- use :paste for the following set of instructions

```
val schema = StructType(Array(
           StructField("maker", StringType, true),
           StructField("model", StringType, true),
           StructField("mileage", IntegerType, true),
           StructField("manufacture_year", IntegerType, true),
           StructField("engine_displacement", IntegerType, true),
           StructField("engine_power", IntegerType, true),
           StructField("body_type", StringType, true),
           StructField("color_slug", StringType, true),
           StructField("stk_year", IntegerType, true),
           StructField("transmission", StringType, true),
           StructField("door_count", IntegerType, true),
           StructField("seat_count", IntegerType, true),
           StructField("fuel_type", StringType, true),
           StructField("date_created", DateType, true),
           StructField("datelastseen", DateType, true),
           StructField("price_eur", FloatType, true)))

val used_cars = spark
  .read.format("csv")
  .option("header", "True")
  .schema(schema)
  .load("hdfs://10.128.0.6:8020/BigData/cars.csv")
```

-- end paste (Cntrl-D)

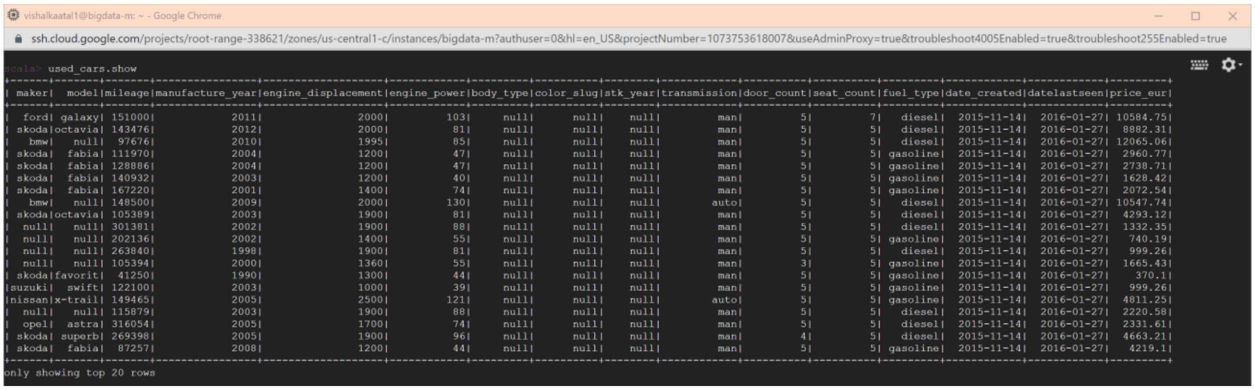
```
used_cars.show
```



vishalkaatal1@bigdata-m: ~ - Google Chrome

ssh.cloud.google.com/projects/root-range-338621/zones/us-central1-c/instances/bigdata-m?authuser=0&hl=en_US&projectNumber=1073753618007&useAdminProxy=true&troubleshoot4005En...

```
scala> :paste
// Entering paste mode (ctrl-D to finish)

val schema = StructType(Array(

         StructField("maker", StringType, true),
         StructField("model", StringType, true),
         StructField("mileage", IntegerType, true),
         StructField("manufacture_year", IntegerType, true),
         StructField("engine_displacement", IntegerType, true),
         StructField("engine_power", IntegerType, true),
         StructField("body_type", StringType, true),
         StructField("color_slug", StringType, true),
         StructField("stk_year", IntegerType, true),
         StructField("transmission", StringType, true),
         StructField("door_count", IntegerType, true),
         StructField("seat_count", IntegerType, true),
         StructField("fuel_type", StringType, true),
         StructField("date_created", DateType, true),
         StructField("datelastseen", DateType, true),
         StructField("price_eur", FloatType, true)))

val used_cars = spark
  .read.format("csv")
  .option("header", "True")
  .schema(schema)
  .load("hdfs://10.128.0.6:8020/BigData/cars.csv")

// Exiting paste mode, now interpreting.

schema: org.apache.spark.sql.types.StructType = StructType(StructField(maker,StringType,true), StructField(model,StringType,true), StructField(mileage,IntegerType
,true), StructField(manufacture_year,IntegerType,true), StructField(engine_displacement,IntegerType,true), StructField(engine_power,IntegerType,true), StructField
(body_type,StringType,true), StructField(color_slug,StringType,true), StructField(stk_year,IntegerType,true), StructField(transmission,StringType,true), StructFie
ld(door_count,IntegerType,true), StructField(seat_count,IntegerType,true), StructField(fuel_type,StringType,true), StructField(date_created,DateType,true), Struct
Field(datelastseen,DateType,true), StructField(price_eur,FloatType,true))
used_cars: org.apache.spark...
```



vishalkaatal1@bigdata-m: ~ - Google Chrome

ssh.cloud.google.com/projects/root-range-338621/zones/us-central1-c/instances/bigdata-m?authuser=0&hl=en_US&projectNumber=1073753618007&useAdminProxy=true&troubleshoot4005Enabled=true&troubleshoot255Enabled=true

```
scala> used_cars.show
+------+-------+-------+----------------+-------------------+------------+---------+----------+--------+------------+----------+----------+---------+------------+------------+---------+
| maker|  model|mileage|manufacture_year|engine_displacement|engine_power|body_type|color_slug|stk_year|transmission|door_count|seat_count|fuel_type|date_created|datelastseen|price_eur|
+------+-------+-------+----------------+-------------------+------------+---------+----------+--------+------------+----------+----------+---------+------------+------------+---------+
|  ford| galaxy| 151000|            2011|               2000|         103|     null|      null|    null|         man|         5|         7|   diesel|  2015-11-14|  2016-01-27| 10584.75|
| skoda|octavia| 143476|            2012|               2000|          81|     null|      null|    null|         man|         5|         5|   diesel|  2015-11-14|  2016-01-27|  8882.31|
|   bmw|   null|  97676|            2010|               1995|          85|     null|      null|    null|         man|         5|         5|   diesel|  2015-11-14|  2016-01-27| 12065.06|
| skoda|  fabia| 111970|            2004|               1200|          47|     null|      null|    null|         man|         5|         5| gasoline|  2015-11-14|  2016-01-27|  2960.77|
| skoda|  fabia| 128886|            2004|               1200|          47|     null|      null|    null|         man|         5|         5| gasoline|  2015-11-14|  2016-01-27|  2738.71|
| skoda|  fabia| 140932|            2003|               1200|          40|     null|      null|    null|         man|         5|         5| gasoline|  2015-11-14|  2016-01-27|  1628.42|
| skoda|  fabia| 167220|            2001|               1400|          74|     null|      null|    null|         man|         5|         5| gasoline|  2015-11-14|  2016-01-27|  2072.54|
|   bmw|   null| 148500|            2009|               2000|         130|     null|      null|    null|        auto|         5|         5|   diesel|  2015-11-14|  2016-01-27| 10547.74|
| skoda|octavia| 105389|            2003|               1900|          81|     null|      null|    null|         man|         5|         5|   diesel|  2015-11-14|  2016-01-27|  4293.12|
|  null|   null| 301381|            2002|               1900|          88|     null|      null|    null|         man|         5|         5|   diesel|  2015-11-14|  2016-01-27|  1332.35|
|  null|   null| 202136|            2002|               1400|          55|     null|      null|    null|         man|         5|         5| gasoline|  2015-11-14|  2016-01-27|   740.19|
|  null|   null| 263840|            1998|               1900|          81|     null|      null|    null|         man|         5|         5|   diesel|  2015-11-14|  2016-01-27|   999.26|
|  null|   null| 105394|            2000|               1360|          55|     null|      null|    null|         man|         3|         5| gasoline|  2015-11-14|  2016-01-27|  1665.43|
| skoda|favorit|  41250|            1990|               1300|          44|     null|      null|    null|         man|         5|         5| gasoline|  2015-11-14|  2016-01-27|    370.1|
|suzuki|  swift| 122100|            2003|               1000|          39|     null|      null|    null|         man|         5|         5| gasoline|  2015-11-14|  2016-01-27|   999.26|
|nissan|x-trail| 149465|            2005|               2500|         121|     null|      null|    null|        auto|         5|         5| gasoline|  2015-11-14|  2016-01-27|  4811.25|
|  null|   null| 115879|            2003|               1900|          88|     null|      null|    null|         man|         5|         5|   diesel|  2015-11-14|  2016-01-27|  2220.58|
|  opel|  astra| 316054|            2005|               1700|          74|     null|      null|    null|         man|         5|         5|   diesel|  2015-11-14|  2016-01-27|  2331.61|
| skoda| superb| 269398|            2005|               1900|          96|     null|      null|    null|         man|         4|         5|   diesel|  2015-11-14|  2016-01-27|  4663.21|
| skoda|  fabia|  87257|            2008|               1200|          44|     null|      null|    null|         man|         5|         5| gasoline|  2015-11-14|  2016-01-27|   4219.1|
+------+-------+-------+----------------+-------------------+------------+---------+----------+--------+------------+----------+----------+---------+------------+------------+---------+
only showing top 20 rows
```

Task 2: Look at the date column of the table used_cars. Why does the date column have all NULL or incomplete values?

The data type selected for a date is DateType, due to which in-completed values of date have been shown as output. Even though it still shows the proper dates instead of nulls, it can be fixed in the next task.

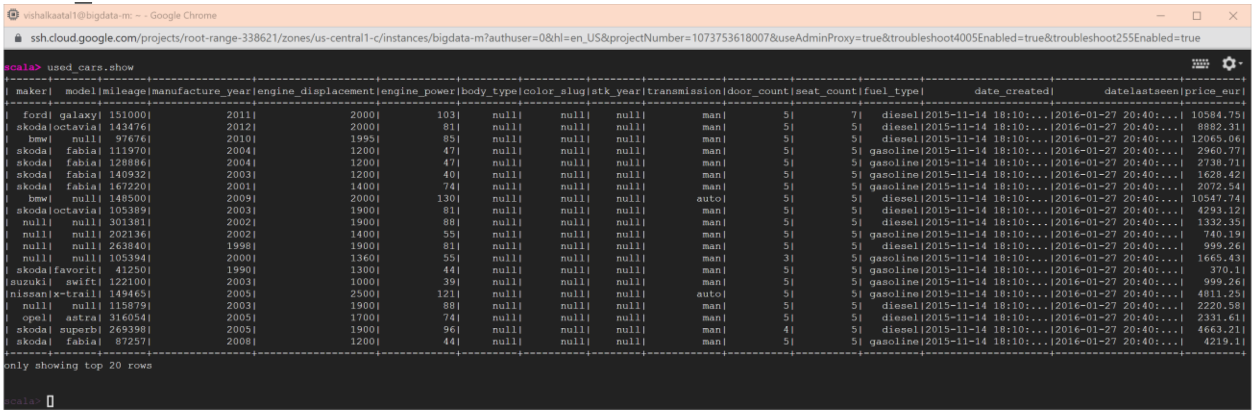
Task 3: Create a table such that the date column is read correctly based on the format in the dataset.

/* Fixing the DataType for dates and creating new schema. Loading the dataset again into spark */
-- use :paste

```
val schema_new = StructType(Array(
           StructField("maker", StringType, true),
           StructField("model", StringType, true),
           StructField("mileage", IntegerType, true),
           StructField("manufacture_year", IntegerType, true),
           StructField("engine_displacement", IntegerType, true),
           StructField("engine_power", IntegerType, true),
           StructField("body_type", StringType, true),
           StructField("color_slug", StringType, true),
           StructField("stk_year", IntegerType, true),
           StructField("transmission", StringType, true),
           StructField("door_count", IntegerType, true),
           StructField("seat_count", IntegerType, true),
           StructField("fuel_type", StringType, true),
           StructField("date_created", TimestampType, true),
           StructField("datelastseen", TimestampType, true),
           StructField("price_eur", FloatType, true)))

val used_cars = spark
  .read.format("csv")
  .option("header", "True")
  .schema(schema_new)
  .load("hdfs://10.128.0.6:8020/BigData/cars.csv")
```

-- end paste (Cntrl-D)

```
used_cars.show
```



vishalkaatal1@bigdata-m: ~ - Google Chrome
ssh.cloud.google.com/projects/root-range-338621/zones/us-central1-c/instances/bigdata-m?authuser=0&hl=en_US&projectNumber=1073753618007&useAdminProxy=true&troubleshoot4005Enabled=true&troubleshoot255Enabled=true

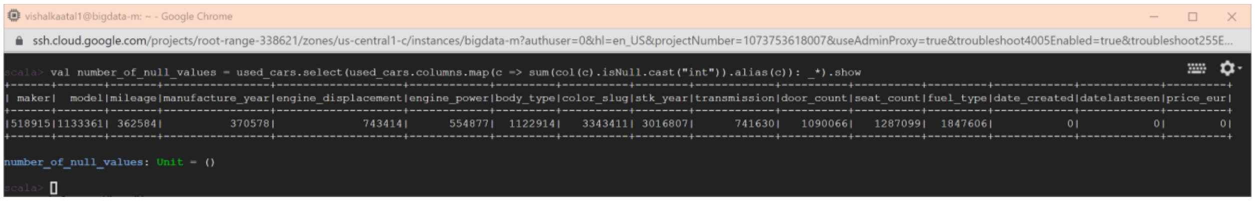
```
scala> used_cars.show
+------+-------+-------+----------------+-------------------+------------+---------+----------+--------+------------+----------+----------+---------+--------------------+--------------------+---------+
| maker|  model|mileage|manufacture_year|engine_displacement|engine_power|body_type|color_slug|stk_year|transmission|door_count|seat_count|fuel_type|        date_created|        datelastseen|price_eur|
+------+-------+-------+----------------+-------------------+------------+---------+----------+--------+------------+----------+----------+---------+--------------------+--------------------+---------+
|  ford| galaxy| 151000|            2011|               2000|         103|     null|      null|    null|         man|         5|         7|   diesel|2015-11-14 18:10:...|2016-01-27 20:40:...| 10584.75|
| skoda|octavia| 143476|            2012|               2000|          81|     null|      null|    null|         man|         5|         5|   diesel|2015-11-14 18:10:...|2016-01-27 20:40:...|  8882.31|
|   bmw|   null|  97676|            2010|               1995|          85|     null|      null|    null|         man|         5|         5|   diesel|2015-11-14 18:10:...|2016-01-27 20:40:...| 12065.06|
| skoda|  fabia| 111970|            2004|               1200|          47|     null|      null|    null|         man|         5|         5| gasoline|2015-11-14 18:10:...|2016-01-27 20:40:...|  2960.77|
| skoda|  fabia| 128886|            2004|               1200|          47|     null|      null|    null|         man|         5|         5| gasoline|2015-11-14 18:10:...|2016-01-27 20:40:...|  2738.71|
| skoda|  fabia| 140932|            2003|               1200|          40|     null|      null|    null|         man|         5|         5| gasoline|2015-11-14 18:10:...|2016-01-27 20:40:...|  1628.42|
| skoda|  fabia| 167220|            2001|               1400|          74|     null|      null|    null|         man|         5|         5| gasoline|2015-11-14 18:10:...|2016-01-27 20:40:...|  2072.54|
|   bmw|   null| 148500|            2009|               2000|         130|     null|      null|    null|        auto|         5|         5|   diesel|2015-11-14 18:10:...|2016-01-27 20:40:...| 10547.74|
| skoda|octavia| 105389|            2003|               1900|          81|     null|      null|    null|         man|         5|         5|   diesel|2015-11-14 18:10:...|2016-01-27 20:40:...|  4293.12|
|  null|   null| 301381|            2002|               1900|          88|     null|      null|    null|         man|         5|         5|   diesel|2015-11-14 18:10:...|2016-01-27 20:40:...|  1332.35|
|  null|   null| 202136|            2002|               1400|          55|     null|      null|    null|         man|         5|         5| gasoline|2015-11-14 18:10:...|2016-01-27 20:40:...|   740.19|
|  null|   null| 263840|            1998|               1900|          81|     null|      null|    null|         man|         5|         5|   diesel|2015-11-14 18:10:...|2016-01-27 20:40:...|   999.26|
|  null|   null| 105394|            2000|               1360|          55|     null|      null|    null|         man|         3|         5| gasoline|2015-11-14 18:10:...|2016-01-27 20:40:...|  1665.43|
| skoda|favorit|  41250|            1990|               1300|          44|     null|      null|    null|         man|         5|         5| gasoline|2015-11-14 18:10:...|2016-01-27 20:40:...|    370.1|
|suzuki|  swift| 122100|            2003|               1000|          39|     null|      null|    null|         man|         5|         5| gasoline|2015-11-14 18:10:...|2016-01-27 20:40:...|   999.26|
|nissan|x-trail| 149465|            2005|               2500|         121|     null|      null|    null|        auto|         5|         5| gasoline|2015-11-14 18:10:...|2016-01-27 20:40:...|  4811.25|
|  null|   null| 115879|            2003|               1900|          88|     null|      null|    null|         man|         5|         5|   diesel|2015-11-14 18:10:...|2016-01-27 20:40:...|  2220.58|
|  opel|  astra| 316054|            2005|               1700|          74|     null|      null|    null|         man|         5|         5|   diesel|2015-11-14 18:10:...|2016-01-27 20:40:...|  2331.61|
| skoda| superb| 269398|            2005|               1900|          96|     null|      null|    null|         man|         4|         5|   diesel|2015-11-14 18:10:...|2016-01-27 20:40:...|  4663.21|
| skoda|  fabia|  87257|            2008|               1200|          44|     null|      null|    null|         man|         5|         5| gasoline|2015-11-14 18:10:...|2016-01-27 20:40:...|   4219.1|
+------+-------+-------+----------------+-------------------+------------+---------+----------+--------+------------+----------+----------+---------+--------------------+--------------------+---------+
only showing top 20 rows

scala> 
```

In the above figure, the datatype for a date has been solved, which displays the date and time.

Task 4: Write Spark DataFrames queries to see how many missing values you have in each attribute? Based on the results, document how many missing values in each column we have. Especially, mention those columns with more than 50% missing values.

/* Finding number of null values in dataset */

```
val number_of_null_values = used_cars.select(used_cars.columns.map(c =>
sum(col(c).isNull.cast("int")).alias(c)): _*).show
```


vishalkaatal1@bigdata-m: ~ - Google Chrome
ssh.cloud.google.com/projects/root-range-338621/zones/us-central1-c/instances/bigdata-m?authuser=0&hl=en_US&projectNumber=1073753618007&useAdminProxy=true&troubleshoot4005Enabled=true&troubleshoot255E...

```
scala> val number_of_null_values = used_cars.select(used_cars.columns.map(c => sum(col(c).isNull.cast("int")).alias(c)): _*).show
+------+-------+-------+----------------+-------------------+------------+---------+----------+--------+------------+----------+----------+---------+------------+------------+---------+
| maker|  model|mileage|manufacture_year|engine_displacement|engine_power|body_type|color_slug|stk_year|transmission|door_count|seat_count|fuel_type|date_created|datelastseen|price_eur|
+------+-------+-------+----------------+-------------------+------------+---------+----------+--------+------------+----------+----------+---------+------------+------------+---------+
|518915|1133361| 362584|          370578|             743414|      554877|  1122914|   3343411| 3016807|      741630|   1090066|   1287099|  1847606|           0|           0|        0|
+------+-------+-------+----------------+-------------------+------------+---------+----------+--------+------------+----------+----------+---------+------------+------------+---------+

number_of_null_values: Unit = ()

scala> 
```

The current dataset contains 3.5 million records in each column; therefore, anything over 1.7 million null values columns represents more than 50% missing values that are colour_slug (3.34 million), stk_year (3.01 million) and fuel_type (1.84 million).

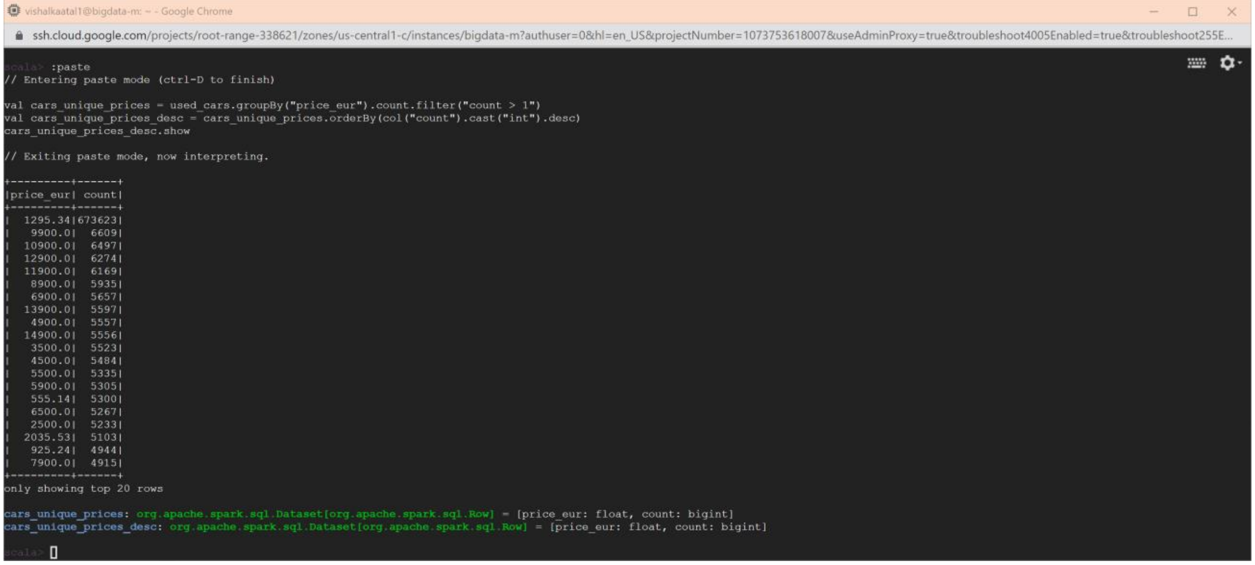
Task 5: Group the price column and count the number of unique prices. Do you notice if there is a single price repeating across the ads?

/* Group the price column and count the number of unique prices */

-- use :paste for the following set of instructions

```
val cars_unique_prices =
used_cars.groupBy("price_eur").count.filter("count > 1")
val cars_unique_prices_desc =
cars_unique_prices.orderBy(col("count").cast("int").desc)
cars_unique_prices_desc.show
```

-- end paste (Cntrl-D)


vishalkaatal1@bigdata-m: ~ - Google Chrome
ssh.cloud.google.com/projects/root-range-338621/zones/us-central1-c/instances/bigdata-m?authuser=0&hl=en_US&projectNumber=1073753618007&useAdminProxy=true&troubleshoot4005Enabled=true&troubleshoot255E...

```
scala> :paste
// Entering paste mode (ctrl-D to finish)

val cars_unique_prices = used_cars.groupBy("price_eur").count.filter("count > 1")
val cars_unique_prices_desc = cars_unique_prices.orderBy(col("count").cast("int").desc)
cars_unique_prices_desc.show

// Exiting paste mode, now interpreting.

+---------+------+
|price_eur| count|
+---------+------+
|  1295.34|673623|
|   9900.0|  6609|
|  10900.0|  6497|
|  12900.0|  6274|
|  11900.0|  6169|
|   8900.0|  5935|
|   6900.0|  5657|
|  13900.0|  5597|
|   4900.0|  5557|
|  14900.0|  5556|
|   3500.0|  5523|
|   4500.0|  5484|
|   5500.0|  5335|
|   5900.0|  5305|
|   555.14|  5300|
|   6500.0|  5267|
|   2500.0|  5233|
|  2035.53|  5103|
|   925.24|  4944|
|   7900.0|  4915|
+---------+------+
only showing top 20 rows

cars_unique_prices: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [price_eur: float, count: bigint]
cars_unique_prices_desc: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [price_eur: float, count: bigint]

scala> 
```

The price of 1295.34 has a maximum count of 673623.

*Task 6: Write a Spark DataFrames query to create a new table called **clean_used_cars** from **used_cars** with the following conditions:*

- *Drop the columns with more than 50% missing values*
- *The manufacture year between 2000 and 2017, including 2000 and 2017*
- *Both maker and model exist in the row*
- *The price range is from 3000 to 2000,000 (3000 ≤ price ≤ 2000,000)*
- *Remove any price you singled out in Task 5 (i.e. a price that repeats too frequently for a random set of ads).*

/* Removing the col with more than 50 % null values i.e. color_slug, stk_year, fuel_type */

```
val clean_used_cars1 =
used_cars.drop("color_slug","stk_year","fuel_type")
```

/* Manufacture Year between 2000 and 2017 (inclusive) */

```
val clean_used_cars2 = clean_used_cars1.filter("manufacture_year >= 2000
AND manufacture_year <=2017")
```

/* Both maker and model exist in the row */

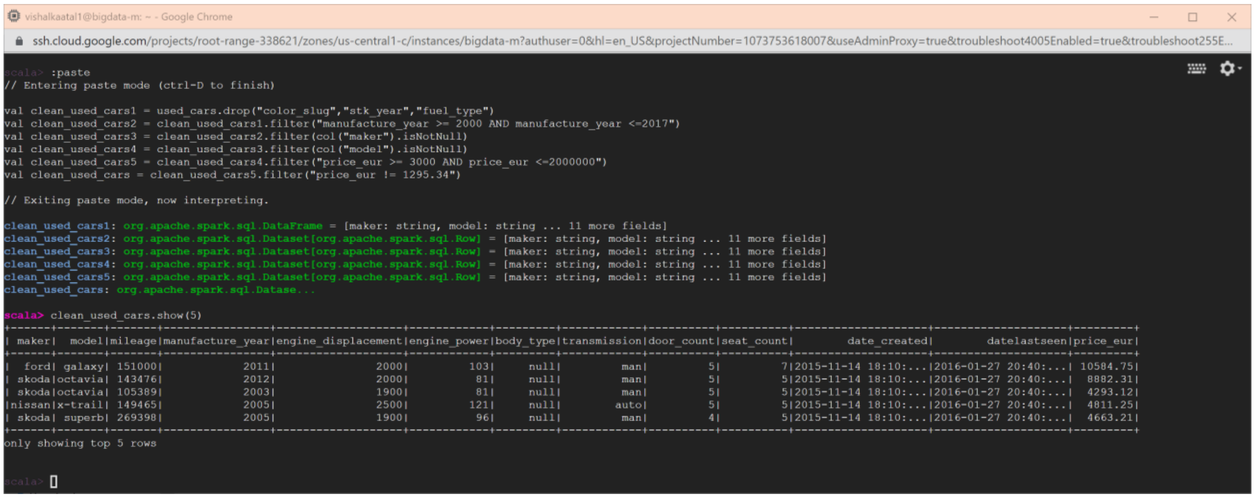
```
val clean_used_cars3 = clean_used_cars2.filter(col("maker").isNotNull)
val clean_used_cars4 = clean_used_cars3.filter(col("model").isNotNull)
```

/* Price range for car is from 3000 to 2000,000 */

```
val clean_used_cars5 = clean_used_cars4.filter("price_eur >= 3000 AND
price_eur <=2000000")
```

/* Removing price_eur = 1295.34 in the Task #5 and removing this price from the dataset. If you notice the highest one 1295.34 got removed during previous filtering process. Just for filteration, perform the following query */

```
val clean_used_cars = clean_used_cars5.filter("price_eur != 1295.34")

clean_used_cars.show(5)
```



vishalkaatal1@bigdata-m: ~ - Google Chrome

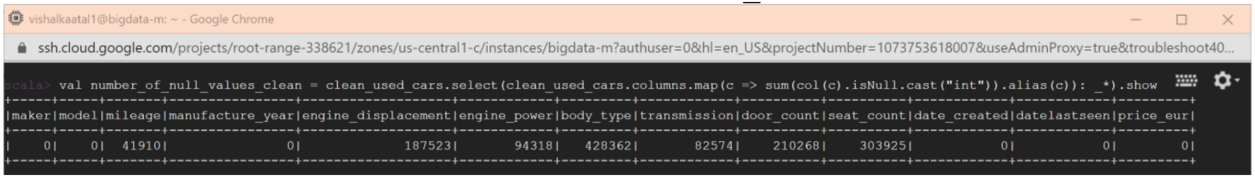
ssh.cloud.google.com/projects/root-range-338621/zones/us-central1-c/instances/bigdata-m?authuser=0&hl=en_US&projectNumber=1073753618007&useAdminProxy=true&troubleshoot4005Enabled=true&troubleshoot255E...

```
scala> :paste
// Entering paste mode (ctrl-D to finish)

val clean_used_cars1 = used_cars.drop("color_slug","stk_year","fuel_type")
val clean_used_cars2 = clean_used_cars1.filter("manufacture_year >= 2000 AND manufacture_year <=2017")
val clean_used_cars3 = clean_used_cars2.filter(col("maker").isNotNull)
val clean_used_cars4 = clean_used_cars3.filter(col("model").isNotNull)
val clean_used_cars5 = clean_used_cars4.filter("price_eur >= 3000 AND price_eur <=2000000")
val clean_used_cars = clean_used_cars5.filter("price_eur != 1295.34")

// Exiting paste mode, now interpreting.

clean_used_cars1: org.apache.spark.sql.DataFrame = [maker: string, model: string ... 11 more fields]
clean_used_cars2: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [maker: string, model: string ... 11 more fields]
clean_used_cars3: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [maker: string, model: string ... 11 more fields]
clean_used_cars4: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [maker: string, model: string ... 11 more fields]
clean_used_cars5: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [maker: string, model: string ... 11 more fields]
clean_used_cars: org.apache.spark.sql.Datase...

scala> clean_used_cars.show(5)
+------+-------+-------+----------------+-------------------+------------+---------+------------+----------+----------+--------------------+--------------------+---------+
| maker|  model|mileage|manufacture_year|engine_displacement|engine_power|body_type|transmission|door_count|seat_count|        date_created|        datelastseen|price_eur|
+------+-------+-------+----------------+-------------------+------------+---------+------------+----------+----------+--------------------+--------------------+---------+
|  ford| galaxy| 151000|            2011|               2000|         103|     null|         man|         5|         7|2015-11-14 18:10:...|2016-01-27 20:40:...| 10584.75|
| skoda|octavia| 143476|            2012|               2000|          81|     null|         man|         5|         5|2015-11-14 18:10:...|2016-01-27 20:40:...|  8882.31|
| skoda|octavia| 105389|            2003|               1900|          81|     null|         man|         5|         5|2015-11-14 18:10:...|2016-01-27 20:40:...|  4293.12|
|nissan|x-trail| 149465|            2005|               2500|         121|     null|        auto|         5|         5|2015-11-14 18:10:...|2016-01-27 20:40:...|  4811.25|
| skoda| superb| 269398|            2005|               1900|          96|     null|         man|         4|         5|2015-11-14 18:10:...|2016-01-27 20:40:...|  4663.21|
+------+-------+-------+----------------+-------------------+------------+---------+------------+----------+----------+--------------------+--------------------+---------+
only showing top 5 rows

scala> 
```

Task 7: Write a Spark DataFrames query to find how many records remained ***clean_used_cars***

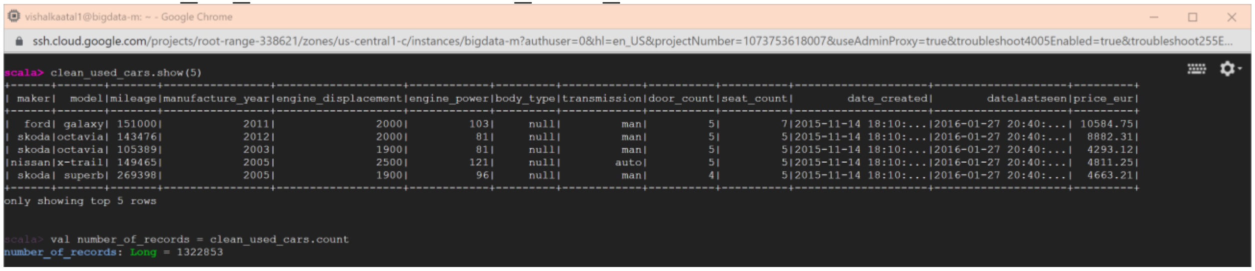
/* How many records remaining in clean_used_cars dataframe */

```
val number_of_null_values_clean =
clean_used_cars.select(clean_used_cars.columns.map(c =>
sum(col(c).isNull.cast("int")).alias(c)): _*).show
```


vishalkaatal1@bigdata-m: ~ - Google Chrome
ssh.cloud.google.com/projects/root-range-338621/zones/us-central1-c/instances/bigdata-m?authuser=0&hl=en_US&projectNumber=1073753618007&useAdminProxy=true&troubleshoot40...

```
scala> val number_of_null_values_clean = clean_used_cars.select(clean_used_cars.columns.map(c => sum(col(c).isNull.cast("int")).alias(c)): _*).show
+-----+-----+-------+----------------+-------------------+------------+---------+------------+----------+----------+------------+------------+---------+
|maker|model|mileage|manufacture_year|engine_displacement|engine_power|body_type|transmission|door_count|seat_count|date_created|datelastseen|price_eur|
+-----+-----+-------+----------------+-------------------+------------+---------+------------+----------+----------+------------+------------+---------+
|    0|    0|  41910|               0|             187523|       94318|   428362|       82574|    210268|    303925|           0|           0|        0|
+-----+-----+-------+----------------+-------------------+------------+---------+------------+----------+----------+------------+------------+---------+
```

```
val number_of_records = clean_used_cars.count
```


vishalkaatal1@bigdata-m: ~ - Google Chrome
ssh.cloud.google.com/projects/root-range-338621/zones/us-central1-c/instances/bigdata-m?authuser=0&hl=en_US&projectNumber=1073753618007&useAdminProxy=true&troubleshoot4005Enabled=true&troubleshoot255E...

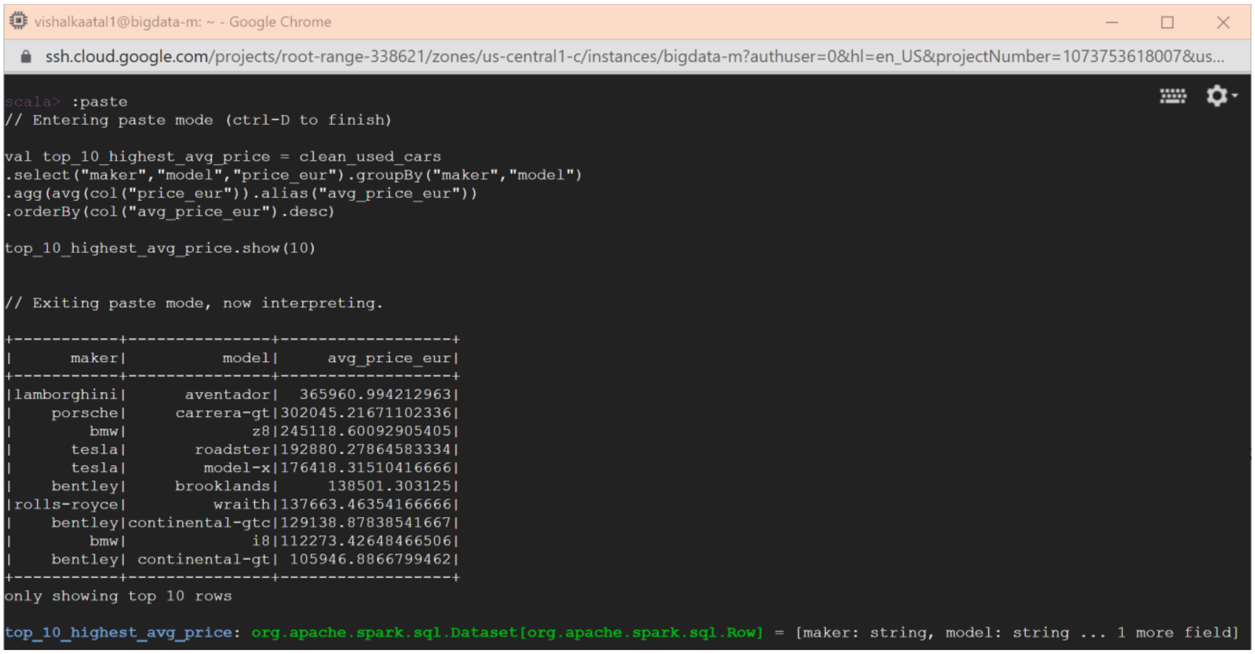
```
scala> clean_used_cars.show(5)
+------+-------+-------+----------------+-------------------+------------+---------+------------+----------+----------+--------------------+--------------------+---------+
| maker|  model|mileage|manufacture_year|engine_displacement|engine_power|body_type|transmission|door_count|seat_count|        date_created|        datelastseen|price_eur|
+------+-------+-------+----------------+-------------------+------------+---------+------------+----------+----------+--------------------+--------------------+---------+
|  ford| galaxy| 151000|            2011|               2000|         103|     null|         man|         5|         7|2015-11-14 18:10:...|2016-01-27 20:40:...| 10584.75|
| skoda|octavia| 143476|            2012|               2000|          81|     null|         man|         5|         5|2015-11-14 18:10:...|2016-01-27 20:40:...|  8882.31|
| skoda|octavia| 105389|            2003|               1900|          81|     null|         man|         5|         5|2015-11-14 18:10:...|2016-01-27 20:40:...|  4293.12|
|nissan|x-trail| 149465|            2005|               2500|         121|     null|        auto|         5|         5|2015-11-14 18:10:...|2016-01-27 20:40:...|  4811.25|
| skoda| superb| 269398|            2005|               1900|          96|     null|         man|         4|         5|2015-11-14 18:10:...|2016-01-27 20:40:...|  4663.21|
+------+-------+-------+----------------+-------------------+------------+---------+------------+----------+----------+--------------------+--------------------+---------+
only showing top 5 rows


scala> val number_of_records = clean_used_cars.count
number_of_records: Long = 1322853
```

The number of records remaining in the clean_used_cars is **1322853**.

Task 8: Write a Spark DataFrames query to find the make and model for the cars with the top 10 ***highest average price***

/* Spark DataFrames query, find the make & model for the cars with the top 10 highest average price */

```
val top_10_highest_avg_price = clean_used_cars
.select("maker","model","price_eur").groupBy("maker","model")
.agg(avg(col("price_eur")).alias("avg_price_eur"))
.orderBy(col("avg_price_eur").desc)
top_10_highest_avg_price.show(10)
```

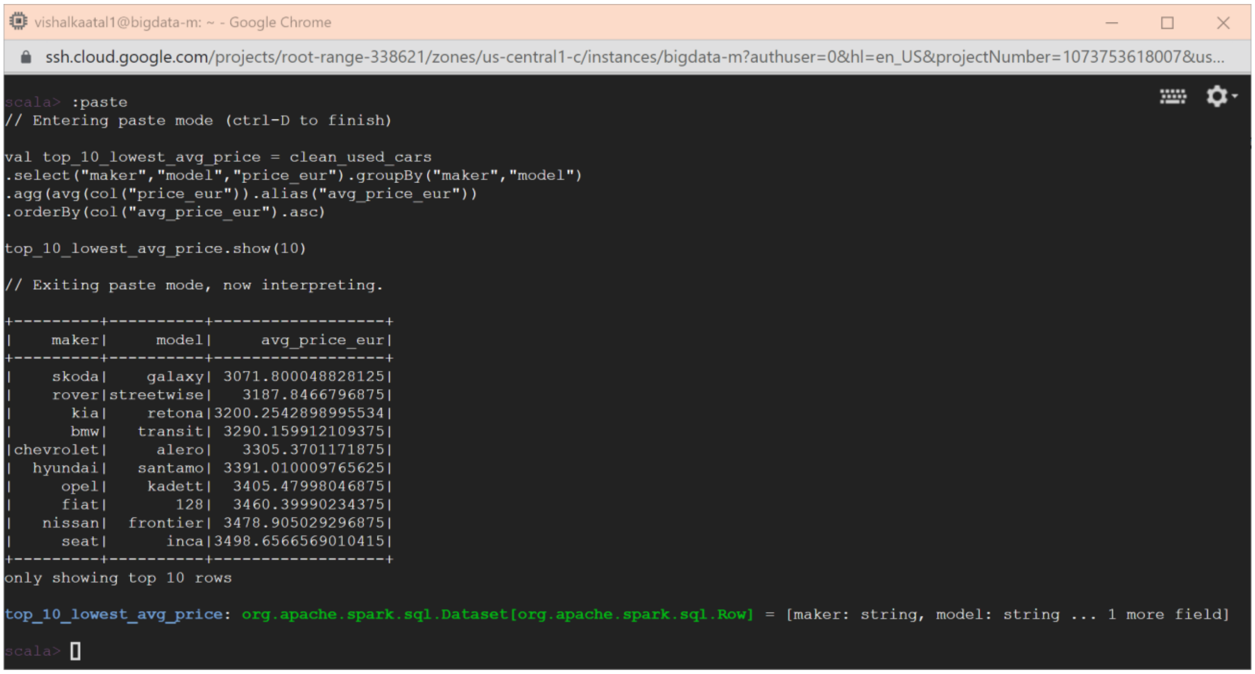

vishalkaatal1@bigdata-m: ~ - Google Chrome
ssh.cloud.google.com/projects/root-range-338621/zones/us-central1-c/instances/bigdata-m?authuser=0&hl=en_US&projectNumber=1073753618007&us...

```
scala> :paste
// Entering paste mode (ctrl-D to finish)

val top_10_highest_avg_price = clean_used_cars
.select("maker","model","price_eur").groupBy("maker","model")
.agg(avg(col("price_eur")).alias("avg_price_eur"))
.orderBy(col("avg_price_eur").desc)

top_10_highest_avg_price.show(10)


// Exiting paste mode, now interpreting.

+-----------+---------------+------------------+
|      maker|          model|     avg_price_eur|
+-----------+---------------+------------------+
|lamborghini|      aventador|  365960.994212963|
|    porsche|    carrera-gt|302045.21671102336|
|        bmw|             z8|245118.60092905405|
|      tesla|       roadster|192880.27864583334|
|      tesla|        model-x|176418.31510416666|
|    bentley|      brooklands|         138501.303125|
|rolls-royce|          wraith|137663.46354166666|
|    bentley|continental-gtc|129138.87838541667|
|        bmw|             i8|112273.42648466506|
|    bentley| continental-gt| 105946.8866799462|
+-----------+---------------+------------------+
only showing top 10 rows

top_10_highest_avg_price: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [maker: string, model: string ... 1 more field]
```

Task 9: Write a Spark DataFrames query to find the make and model for the cars with the top 10 ***lowest average price***

/* Spark DataFrames query, find the make & model for the cars with the top 10 lowest average price */

```
val top_10_lowest_avg_price = clean_used_cars
.select("maker","model","price_eur").groupBy("maker","model")
.agg(avg(col("price_eur")).alias("avg_price_eur"))
.orderBy(col("avg_price_eur").asc)

top_10_lowest_avg_price.show(10)
```


vishalkaatal1@bigdata-m: ~ - Google Chrome
ssh.cloud.google.com/projects/root-range-338621/zones/us-central1-c/instances/bigdata-m?authuser=0&hl=en_US&projectNumber=1073753618007&us...

```
scala> :paste
// Entering paste mode (ctrl-D to finish)

val top_10_lowest_avg_price = clean_used_cars
.select("maker","model","price_eur").groupBy("maker","model")
.agg(avg(col("price_eur")).alias("avg_price_eur"))
.orderBy(col("avg_price_eur").asc)

top_10_lowest_avg_price.show(10)

// Exiting paste mode, now interpreting.

+---------+----------+-----------------+
|    maker|     model|    avg_price_eur|
+---------+----------+-----------------+
|    skoda|    galaxy| 3071.800048828125|
|    rover|streetwise|   3187.8466796875|
|      kia|    retona|3200.2542898995534|
|      bmw|   transit| 3290.159912109375|
|chevrolet|     alero|   3305.3701171875|
|  hyundai|   santamo| 3391.010009765625|
|     opel|    kadett|  3405.47998046875|
|     fiat|       128|  3460.39990234375|
|   nissan|  frontier| 3478.905029296875|
|     seat|      inca|3498.6566569010415|
+---------+----------+-----------------+
only showing top 10 rows

top_10_lowest_avg_price: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [maker: string, model: string ... 1 more field]

scala> ▯
```

Task 10 – 12: Write a Spark DataFrames query to recommend the top five make and models for different segments for customers

- ***Economic*** *segment customers (3000≤price<20,000)* - *based on the top* ***average price***
- ***Intermediate*** *segment customers (20,000≤price<300,000)* - *based on the top* ***average price***
- ***Luxury*** *segment customers (300,000≤price<2000,000)* - *based on the top* ***average price***

/* Economic Segment DataFrame*/

```
val used_car_segment = clean_used_cars
.groupBy("maker","model")
.agg(avg(col("price_eur")).alias("avg_price_eur"))
.orderBy(col("avg_price_eur").desc)
```

/* Economic Segment Customers | Price: 3000 - 20000*/

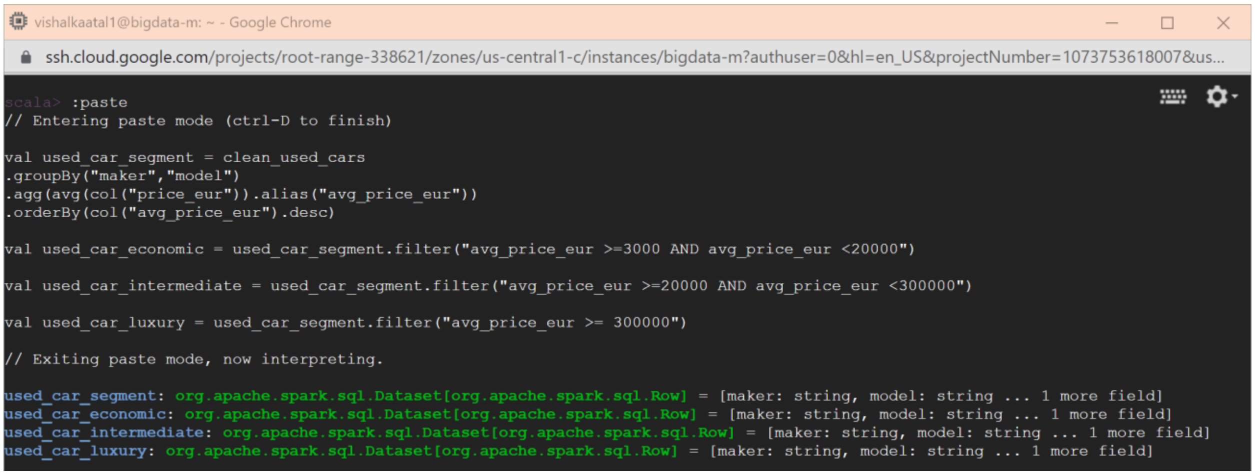
```
val used_car_economic = used_car_segment.filter("avg_price_eur >=3000
AND avg_price_eur <20000")
used_car_economic.show(5)
```

/* Intermediate Segment Customers | Price: 20000 - 300000*/

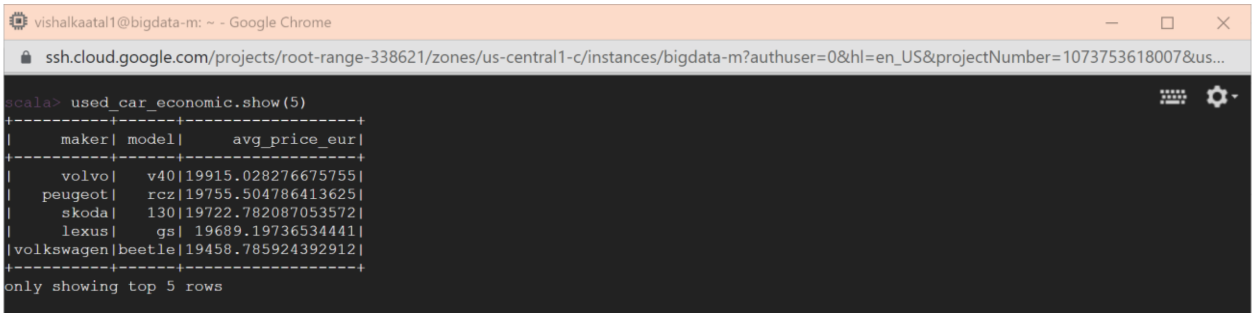
```
val used_car_intermediate = used_car_segment.filter("avg_price_eur
>=20000 AND avg_price_eur <300000")
used_car_intermediate.show(5)
```

/* Luxury Segment Customers | Price: 300000 - 2000000*/

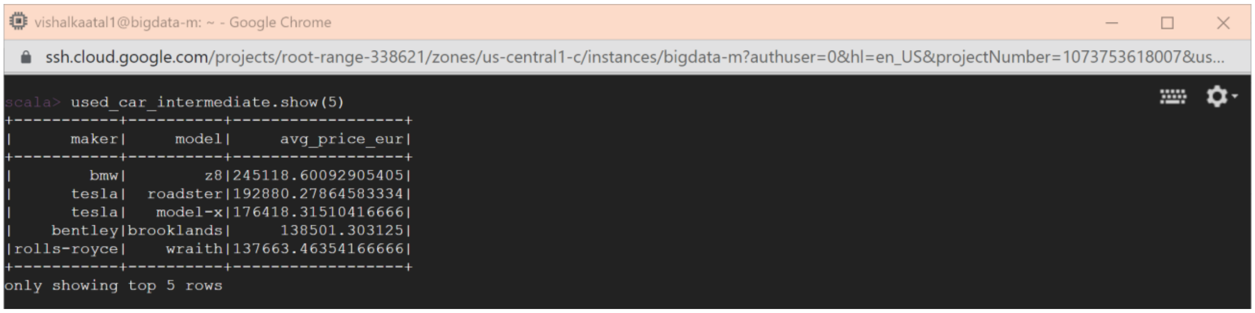
```
val used_car_luxury = used_car_segment.filter("avg_price_eur >=
300000")
used_car_luxury.show(5)
```


vishalkaatal1@bigdata-m: ~ - Google Chrome
ssh.cloud.google.com/projects/root-range-338621/zones/us-central1-c/instances/bigdata-m?authuser=0&hl=en_US&projectNumber=1073753618007&us...

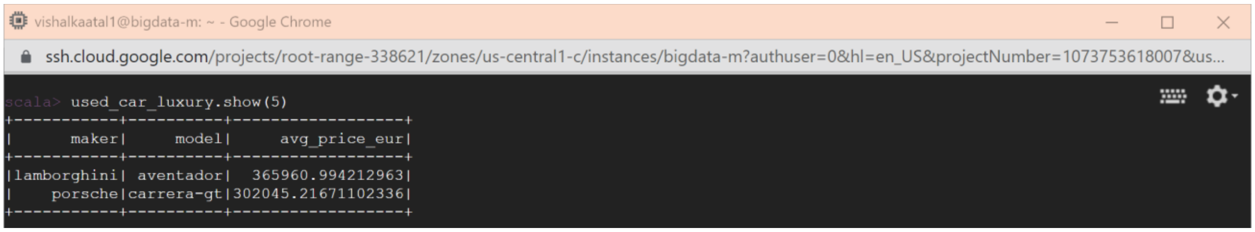
```
scala> :paste
// Entering paste mode (ctrl-D to finish)

val used_car_segment = clean_used_cars
.groupBy("maker","model")
.agg(avg(col("price_eur")).alias("avg_price_eur"))
.orderBy(col("avg_price_eur").desc)

val used_car_economic = used_car_segment.filter("avg_price_eur >=3000 AND avg_price_eur <20000")

val used_car_intermediate = used_car_segment.filter("avg_price_eur >=20000 AND avg_price_eur <300000")

val used_car_luxury = used_car_segment.filter("avg_price_eur >= 300000")

// Exiting paste mode, now interpreting.

used_car_segment: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [maker: string, model: string ... 1 more field]
used_car_economic: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [maker: string, model: string ... 1 more field]
used_car_intermediate: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [maker: string, model: string ... 1 more field]
used_car_luxury: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [maker: string, model: string ... 1 more field]
```


vishalkaatal1@bigdata-m: ~ - Google Chrome
ssh.cloud.google.com/projects/root-range-338621/zones/us-central1-c/instances/bigdata-m?authuser=0&hl=en_US&projectNumber=1073753618007&us...

```
scala> used_car_economic.show(5)
+----------+------+------------------+
|     maker| model|     avg_price_eur|
+----------+------+------------------+
|     volvo|   v40|19915.028276675755|
|   peugeot|   rcz|19755.504786413625|
|     skoda|   130|19722.782087053572|
|     lexus|    gs| 19689.19736534441|
|volkswagen|beetle|19458.785924392912|
+----------+------+------------------+
only showing top 5 rows
```


vishalkaatal1@bigdata-m: ~ - Google Chrome
ssh.cloud.google.com/projects/root-range-338621/zones/us-central1-c/instances/bigdata-m?authuser=0&hl=en_US&projectNumber=1073753618007&us...

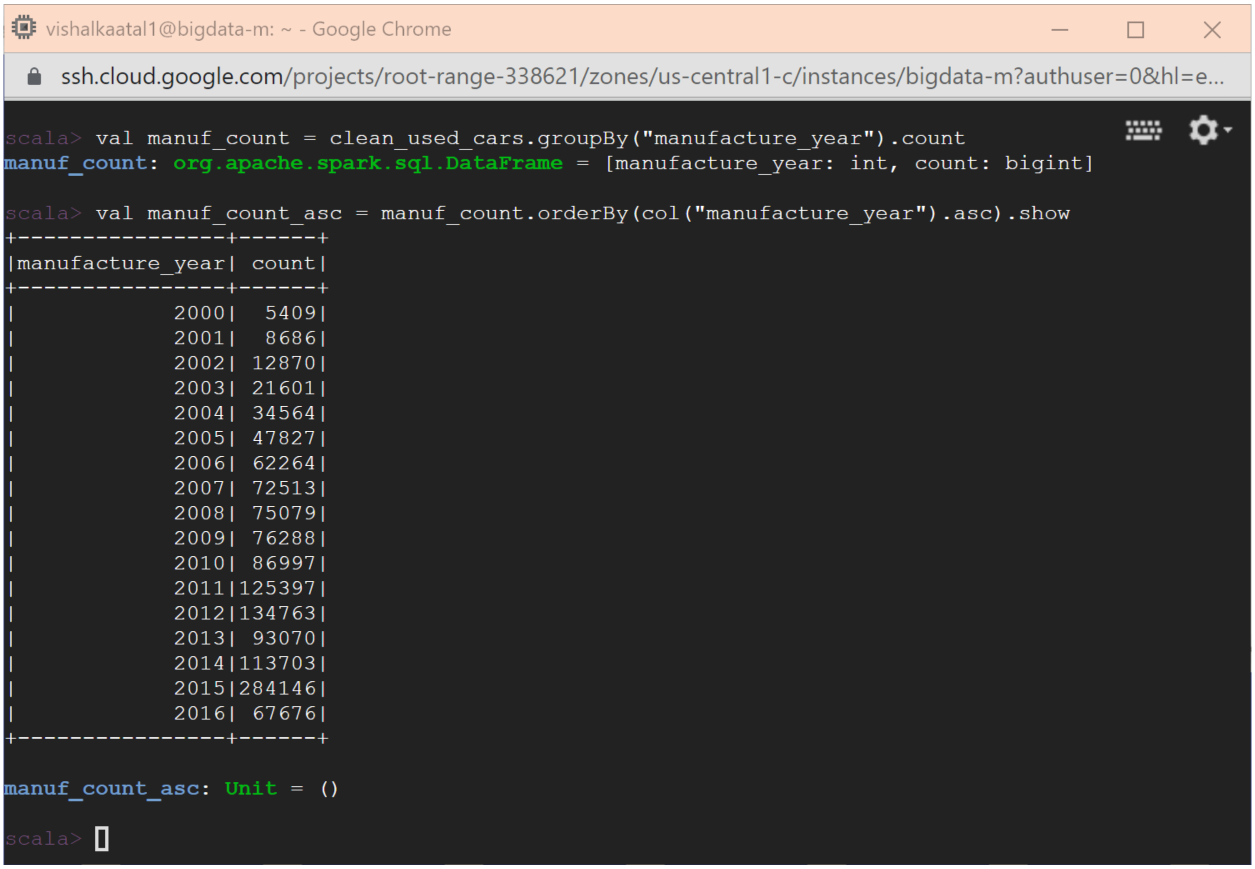
```
scala> used_car_intermediate.show(5)
+-----------+----------+------------------+
|      maker|     model|     avg_price_eur|
+-----------+----------+------------------+
|        bmw|        z8|245118.60092905405|
|      tesla|  roadster|192880.27864583334|
|      tesla|   model-x|176418.31510416666|
|    bentley|brooklands|     138501.303125|
|rolls-royce|    wraith|137663.46354166666|
+-----------+----------+------------------+
only showing top 5 rows
```


vishalkaatal1@bigdata-m: ~ - Google Chrome
ssh.cloud.google.com/projects/root-range-338621/zones/us-central1-c/instances/bigdata-m?authuser=0&hl=en_US&projectNumber=1073753618007&us...

```
scala> used_car_luxury.show(5)
+-----------+----------+------------------+
|      maker|     model|     avg_price_eur|
+-----------+----------+------------------+
|lamborghini| aventador|  365960.994212963|
|    porsche|carrera-gt|302045.21671102336|
+-----------+----------+------------------+
```

/* Inventory of used cars */

```
val manuf_count = clean_used_cars.groupBy("manufacture_year").count
val manuf_count_asc =
manuf_count.orderBy(col("manufacture_year").asc).show
```

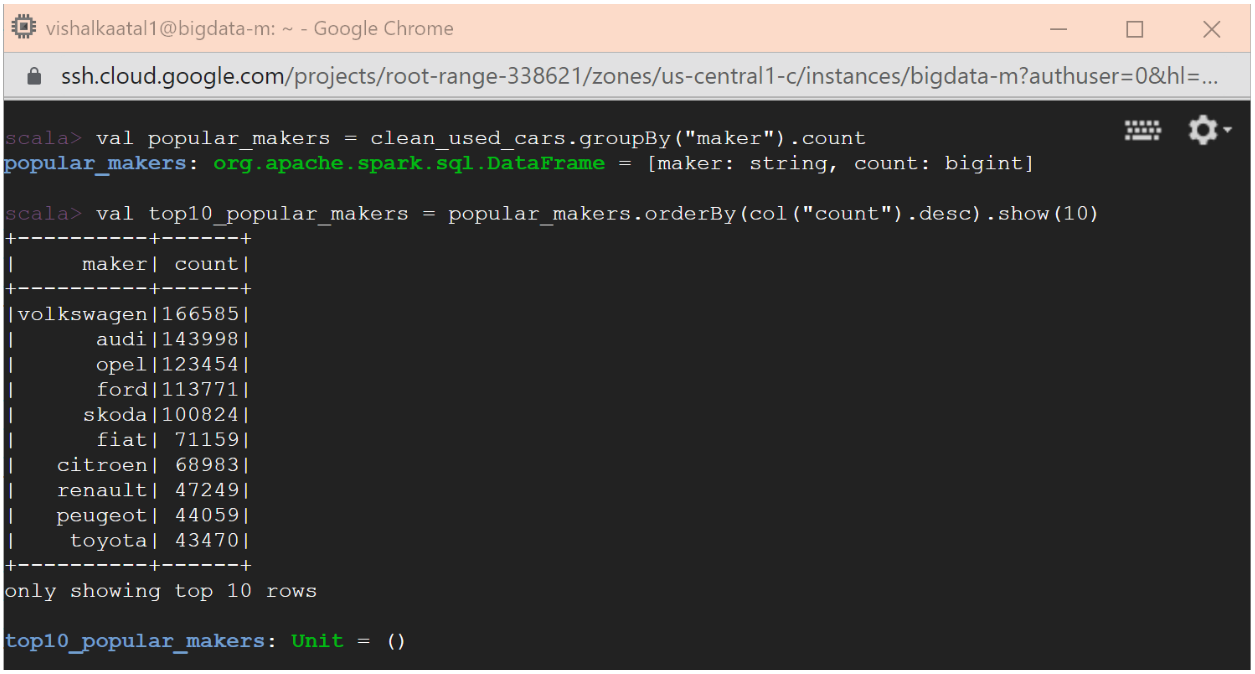


vishalkaatal1@bigdata-m: ~ - Google Chrome
ssh.cloud.google.com/projects/root-range-338621/zones/us-central1-c/instances/bigdata-m?authuser=0&hl=e...

```
scala> val manuf_count = clean_used_cars.groupBy("manufacture_year").count
manuf_count: org.apache.spark.sql.DataFrame = [manufacture_year: int, count: bigint]

scala> val manuf_count_asc = manuf_count.orderBy(col("manufacture_year").asc).show
+----------------+------+
|manufacture_year| count|
+----------------+------+
|            2000|  5409|
|            2001|  8686|
|            2002| 12870|
|            2003| 21601|
|            2004| 34564|
|            2005| 47827|
|            2006| 62264|
|            2007| 72513|
|            2008| 75079|
|            2009| 76288|
|            2010| 86997|
|            2011|125397|
|            2012|134763|
|            2013| 93070|
|            2014|113703|
|            2015|284146|
|            2016| 67676|
+----------------+------+

manuf_count_asc: Unit = ()

scala> 
```

/* Top 10 most popular car makers */

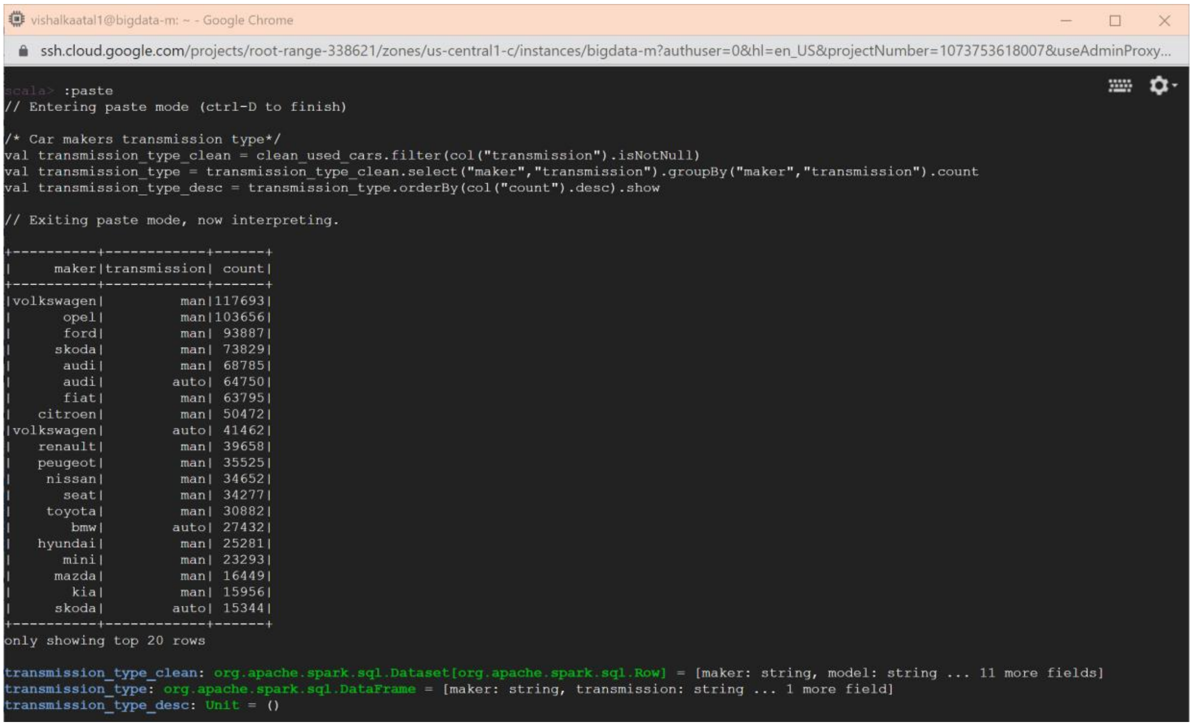
```
val popular_makers = clean_used_cars.groupBy("maker").count
val top10_popular_makers =
popular_makers.orderBy(col("count").desc).show(10)
```


vishalkaatal1@bigdata-m: ~ - Google Chrome
ssh.cloud.google.com/projects/root-range-338621/zones/us-central1-c/instances/bigdata-m?authuser=0&hl=...

```
scala> val popular_makers = clean_used_cars.groupBy("maker").count
popular_makers: org.apache.spark.sql.DataFrame = [maker: string, count: bigint]

scala> val top10_popular_makers = popular_makers.orderBy(col("count").desc).show(10)
+----------+------+
|     maker| count|
+----------+------+
|volkswagen|166585|
|      audi|143998|
|      opel|123454|
|      ford|113771|
|     skoda|100824|
|      fiat| 71159|
|   citroen| 68983|
|   renault| 47249|
|   peugeot| 44059|
|    toyota| 43470|
+----------+------+
only showing top 10 rows

top10_popular_makers: Unit = ()
```

/* Car makers transmission type*/

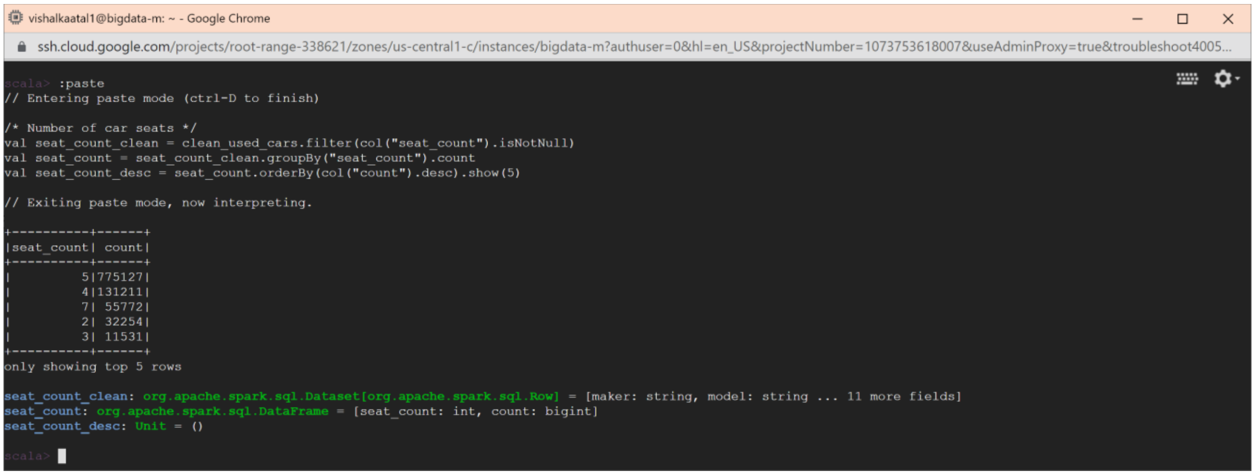
```
val transmission_type_clean =
clean_used_cars.filter(col("transmission").isNotNull)
val transmission_type =
transmission_type_clean.select("maker","transmission").groupBy("maker"
,"transmission").count
val transmission_type_desc =
transmission_type.orderBy(col("count").desc).show
```


vishalkaatal1@bigdata-m: ~ - Google Chrome
ssh.cloud.google.com/projects/root-range-338621/zones/us-central1-c/instances/bigdata-m?authuser=0&hl=en_US&projectNumber=1073753618007&useAdminProxy...

```
scala> :paste
// Entering paste mode (ctrl-D to finish)

/* Car makers transmission type*/
val transmission_type_clean = clean_used_cars.filter(col("transmission").isNotNull)
val transmission_type = transmission_type_clean.select("maker","transmission").groupBy("maker","transmission").count
val transmission_type_desc = transmission_type.orderBy(col("count").desc).show

// Exiting paste mode, now interpreting.

+----------+------------+------+
|     maker|transmission| count|
+----------+------------+------+
|volkswagen|         man|117693|
|      opel|         man|103656|
|      ford|         man| 93887|
|     skoda|         man| 73829|
|      audi|         man| 68785|
|      audi|        auto| 64750|
|      fiat|         man| 63795|
|   citroen|         man| 50472|
|volkswagen|        auto| 41462|
|   renault|         man| 39658|
|   peugeot|         man| 35525|
|    nissan|         man| 34652|
|      seat|         man| 34277|
|    toyota|         man| 30882|
|       bmw|        auto| 27432|
|   hyundai|         man| 25281|
|      mini|         man| 23293|
|     mazda|         man| 16449|
|       kia|         man| 15956|
|     skoda|        auto| 15344|
+----------+------------+------+
only showing top 20 rows

transmission_type_clean: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [maker: string, model: string ... 11 more fields]
transmission_type: org.apache.spark.sql.DataFrame = [maker: string, transmission: string ... 1 more field]
transmission_type_desc: Unit = ()
```

/* Number of car seats */

```
val seat_count_clean =
clean_used_cars.filter(col("seat_count").isNotNull)
val seat_count = seat_count_clean.groupBy("seat_count").count
val seat_count_desc = seat_count.orderBy(col("count").desc).show(5)
```

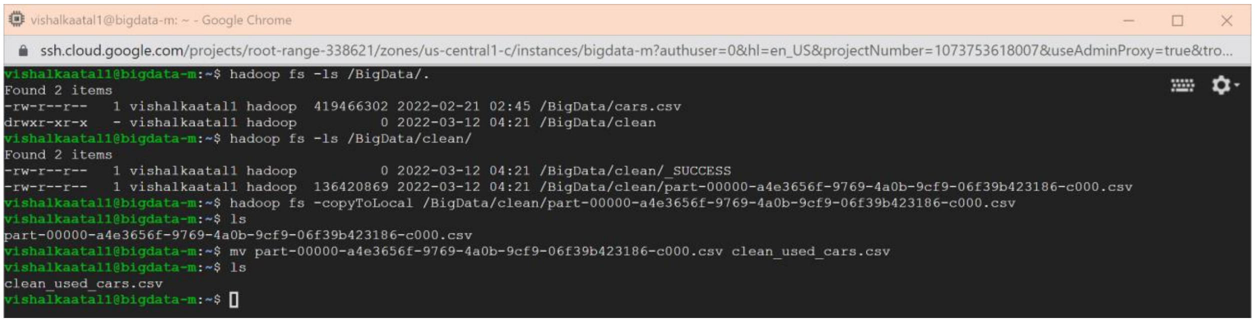


vishalkaatal1@bigdata-m: ~ - Google Chrome
ssh.cloud.google.com/projects/root-range-338621/zones/us-central1-c/instances/bigdata-m?authuser=0&hl=en_US&projectNumber=1073753618007&useAdminProxy=true&troubleshoot4005...

```
scala> :paste
// Entering paste mode (ctrl-D to finish)

/* Number of car seats */
val seat_count_clean = clean_used_cars.filter(col("seat_count").isNotNull)
val seat_count = seat_count_clean.groupBy("seat_count").count
val seat_count_desc = seat_count.orderBy(col("count").desc).show(5)

// Exiting paste mode, now interpreting.

+----------+------+
|seat_count| count|
+----------+------+
|         5|775127|
|         4|131211|
|         7| 55772|
|         2| 32254|
|         3| 11531|
+----------+------+
only showing top 5 rows

seat_count_clean: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [maker: string, model: string ... 11 more fields]
seat_count: org.apache.spark.sql.DataFrame = [seat_count: int, count: bigint]
seat_count_desc: Unit = ()

scala> 
```

/* Download the dataset for visualization using the following commands */

```
clean_used_cars.coalesce(1).write
.option("header", true).csv("hdfs://10.128.0.6:8020/BigData/clean")
```



vishalkaatal1@bigdata-m: ~ - Google Chrome
ssh.cloud.google.com/projects/root-range-338621/zones/us-central1-c/instances/bigdata-m?authuser=0&hl=en_US&projectNumber=1073753618007&useAdminProxy=true&tro...

```
vishalkaatal1@bigdata-m:~$ hadoop fs -ls /BigData/.
Found 2 items
-rw-r--r--   1 vishalkaatal1 hadoop  419466302 2022-02-21 02:45 /BigData/cars.csv
drwxr-xr-x   - vishalkaatal1 hadoop          0 2022-03-12 04:21 /BigData/clean
vishalkaatal1@bigdata-m:~$ hadoop fs -ls /BigData/clean/
Found 2 items
-rw-r--r--   1 vishalkaatal1 hadoop          0 2022-03-12 04:21 /BigData/clean/_SUCCESS
-rw-r--r--   1 vishalkaatal1 hadoop  136420869 2022-03-12 04:21 /BigData/clean/part-00000-a4e3656f-9769-4a0b-9cf9-06f39b423186-c000.csv
vishalkaatal1@bigdata-m:~$ hadoop fs -copyToLocal /BigData/clean/part-00000-a4e3656f-9769-4a0b-9cf9-06f39b423186-c000.csv
vishalkaatal1@bigdata-m:~$ ls
part-00000-a4e3656f-9769-4a0b-9cf9-06f39b423186-c000.csv
vishalkaatal1@bigdata-m:~$ mv part-00000-a4e3656f-9769-4a0b-9cf9-06f39b423186-c000.csv clean_used_cars.csv
vishalkaatal1@bigdata-m:~$ ls
clean_used_cars.csv
vishalkaatal1@bigdata-m:~$ 
```

After moving the clean_used_cars dataset into GCP using the above commands, download the file directly to create all the visualizations in the Analysis section using Power BI.