3/14/2022



Srilekha Sampath Kumar

200499290



**Assignment 1 – Analysis with Spark DataFrames**

BDAT1008- Data Collection and Curation

**Table of Contents**

[Introduction – Spark 2](#_Toc76486220)

[Dataset Background 2](#_Toc76486221)

[Analysis Questions 3](#_Toc76486222)

[Dataset Exploration 3](#_Toc76486223)

[Data Cleaning 4](#_Toc76486224)

[Data Analysis 6](#_Toc76486225)

[Conclusion 1](#_Toc76486226)0

[Appendix 11](#_Toc76486227)

[Scala Codes 17](#_Toc76486221)

[References 20](#_Toc76486222)

**Introduction – Apache Spark DataFrame**

A Spark DataFrame is an integrated data structure with an easy-to-use API for simplifying distributed big data processing. It is a programming abstraction in the Spark SQL module, resembling relational database tables or excel spreadsheets with headers.

**Features of a DataFrame**

* Support for various data formats, such as Hive, CSV, XML, JSON, RDDs, Cassandra, Parquet, etc.
* Support for integration with various Big Data tools.
* Custom memory management to reduce overload and improve performance compared to RDDs.

**Dataset Background**

The Classified ads for cars dataset from Kaggle has data about wide range of used cars from the countries of Germany and Czech Republic since 2015. The whole data is all about Car manufactures, Pricing, Engine capacity, Total door and seat counts, Fuel category and some dates about the car advertisement. The purpose of this assignment is Data Cleaning and Analyzing the dataset to provide valuable information to Stakeholders in order to make business decisions.

There are so many benefits in purchasing a used car. Some of them are, used cars prices and insurance rates are almost 45% lower than new cars with wider variety to choose from our own budget. Also buying a used car reduces the carbon dioxide output into the environment.

Data present in the Dataset ‘all\_anonymized\_2015\_11\_2017\_03.csv’

|  |  |  |  |
| --- | --- | --- | --- |
| **S No** | **Field** | **Data Type** | **Explanation** |
| 1 | maker | String | The name of car manufacturer |
| 2 | model | String | Model of the car |
| 3 | mileage | String | Total distance travelled - Calculated in KM |
| 4 | manufacture\_year | String | The year which the car was manufactured |
| 5 | engine\_displacement | String | volume of cylinders in the engine |
| 6 | engine\_power | String | Power of car's engine |
| 7 | body\_type | String | Description of the car |
| 8 | color\_slug | String | Color of the car |
| 9 | stk\_year | String | Engine sticker year |
| 10 | transmission | String | Type of car - Manual or Automatic |
| 11 | door\_count | String | Number of doors the car have |
| 12 | seat\_count | String | Number of seats available |
| 13 | fuel\_type | String | Type of Fuel - Petrol or Diesel |
| 14 | date\_created | String | Add posted date |
| 15 | date\_last\_seen | String | Last date of the add seen on the website |
| 16 | price\_eur | String | Price of vehicle in Euros |

**Analysis (Questions) to be answered:**

1. List the make and model for the cars with the top 10 highest average price

2. List the make and model for the cars with the top 10 lowest average price

3. List the top five make and model for Economic segment customers price range- 3000 ≤ price < 20,000

4. List the least five make and model for Economic segment customers price range-3000 ≤ price < 20,000

5. List the top five make and model for Intermediate segment customers price range-3000 ≤ price < 20,000

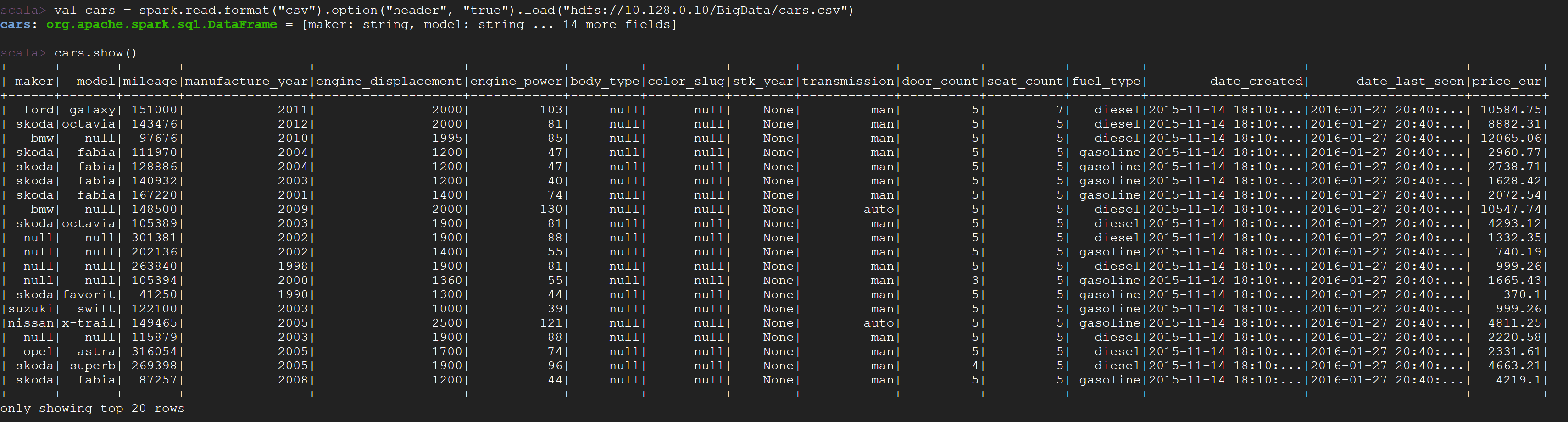
6. List the least make and model for Intermediate segment customers price range-3000 ≤ price < 20,000

7. List the make and model for the Luxury segment customers price range- 300,000 ≤ price < 2000,000

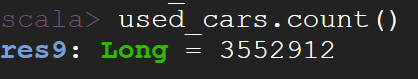
**Dataset Exploration**

All the analysis on the dataset has been performed in **Spark** using **Google Cloud Platform**.

Initial view of the dataset (before cleaning) is shown below,



Total number of records in the dataset

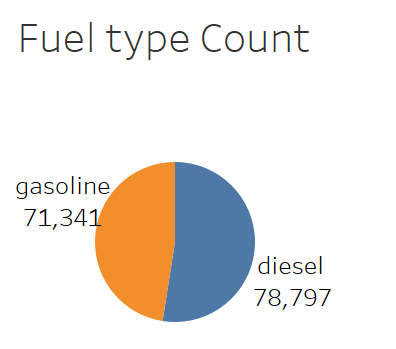
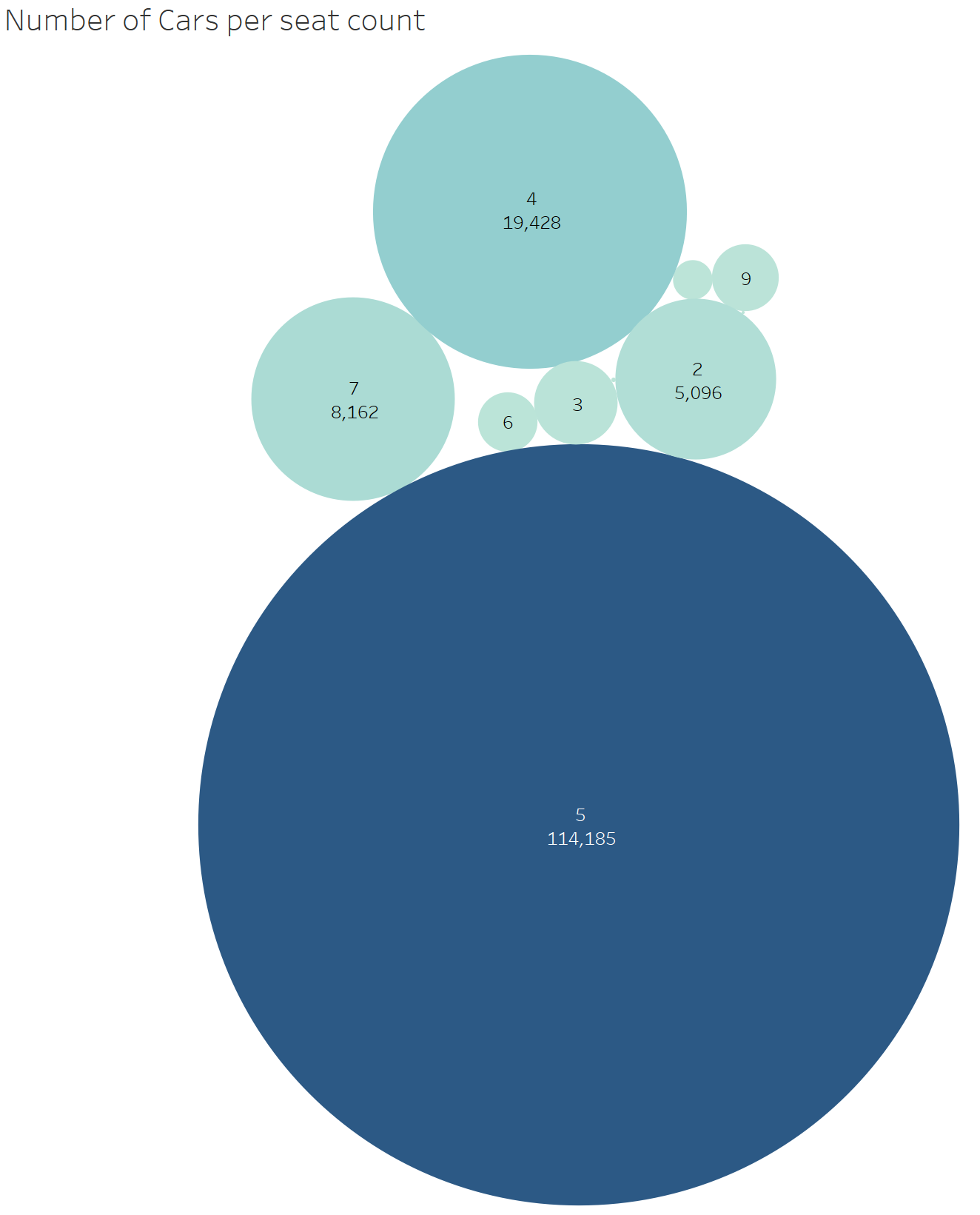


Initial Analysis

Fuel type Count: Compared to Gasoline cars, there are more Diesel cars sold.

Transmission type Count: Manual transmission type cars are good for used car business.

Seat Count: 5-seater cars are mostly preferred.

**Data Cleaning**

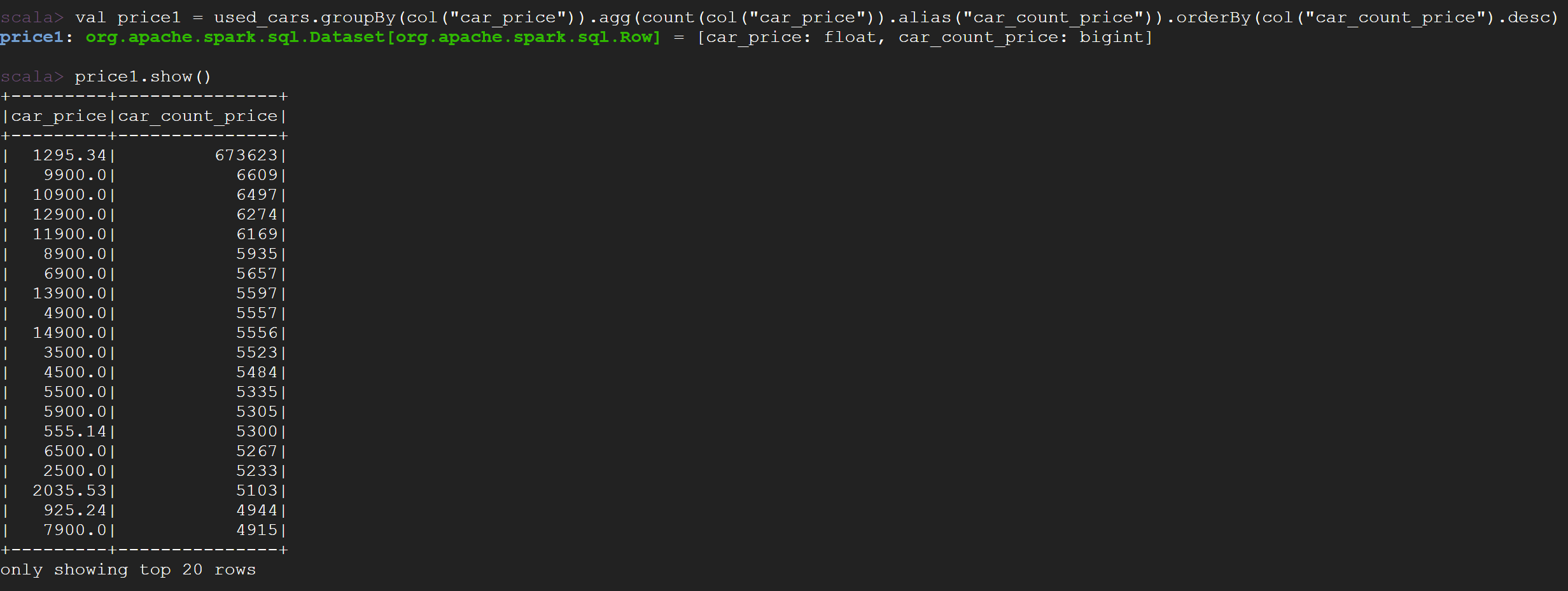
The dataset contains about ~3.5 million rows and 16 columns. As per author, the data is taken about one year ago and also sourced from unstructured sources. Therefore, It is anticipated that there would be fields having missing values or invalid values. Therefore, Data cleaning must be performed on the data before considering it in any kind of decision process.

Missing Value Analysis:



Removal of Null values and Variables:

* Variables with 50% of missing values are removed. The removed variables are, color\_slug, stk\_year and fuel\_type.
* Null values in the fields maker and model are removed as major analysis depends on these two variables.
* Cars with the price value of ‘1295.34’ is removed as it is too frequent. The same is shown below,



Changing data types of the variables:

The datatypes of the following variables has been changed from String to its respective datatypes as per the data present in each variable.



Creating a new DataFrame with following conditions and the analysis has been done based on this selection,

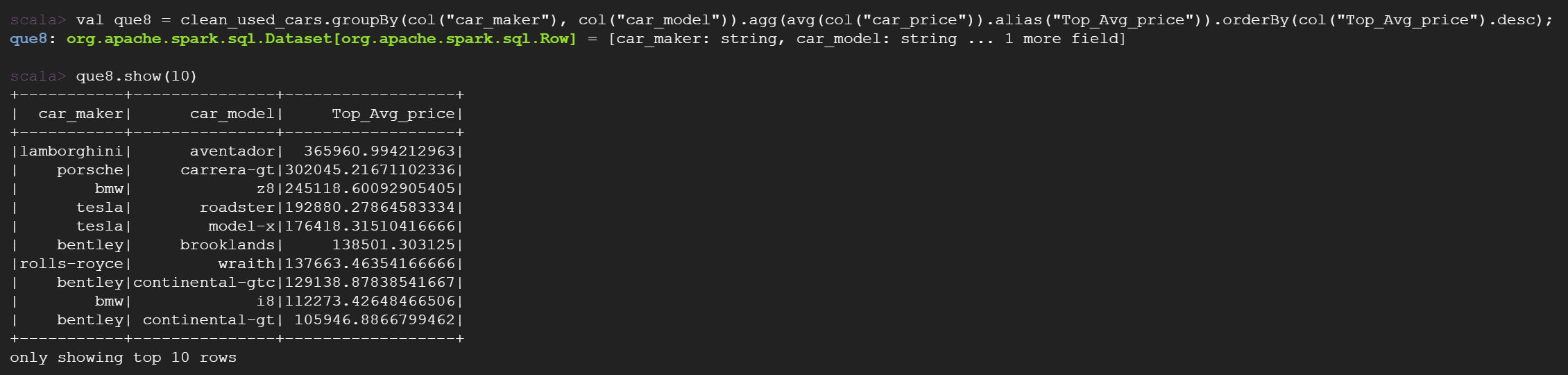


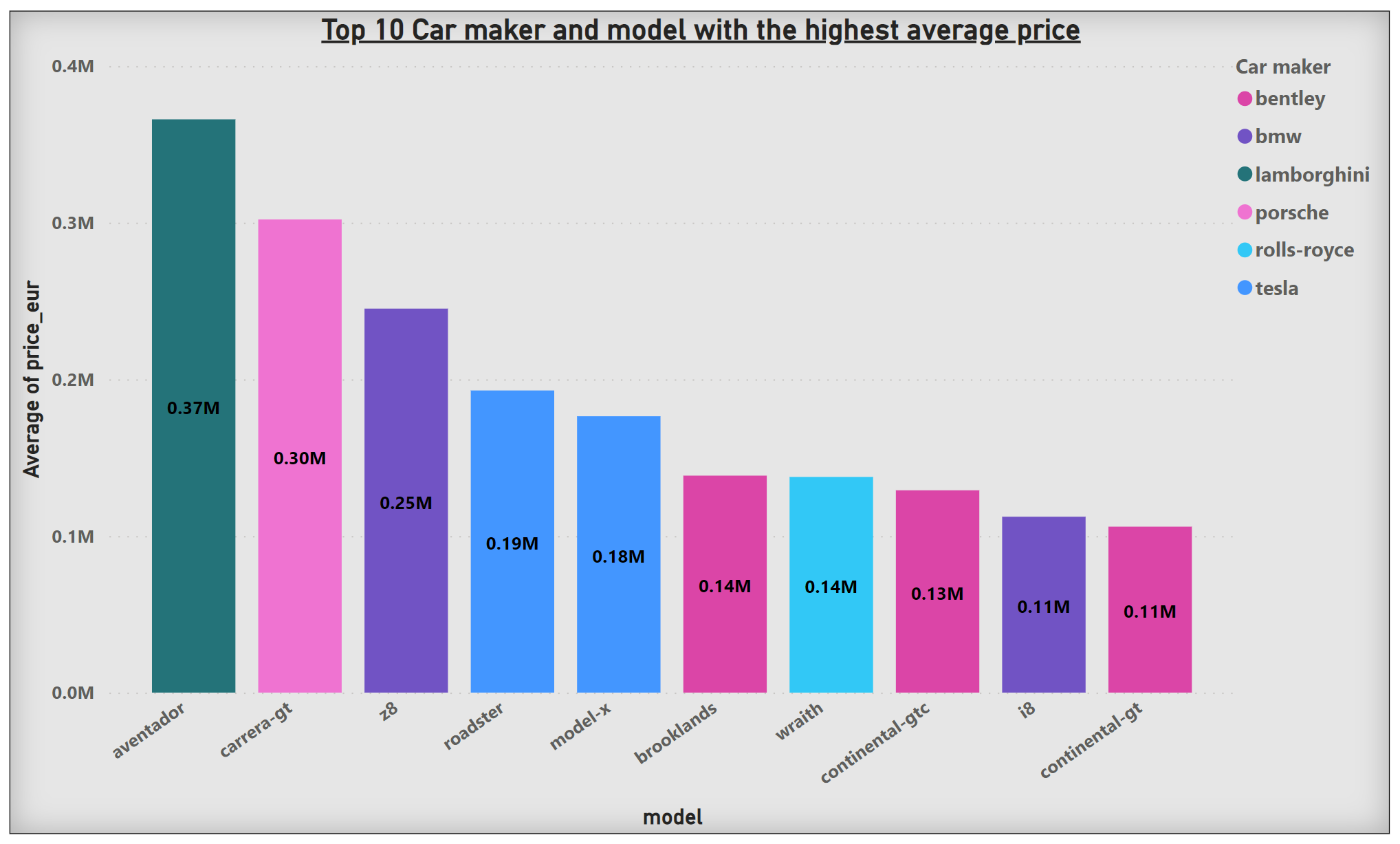
**Data Analysis:**

Answers for the above-mentioned research questions are presented below using Spark and Power BI visualization,

**Question 1**:-

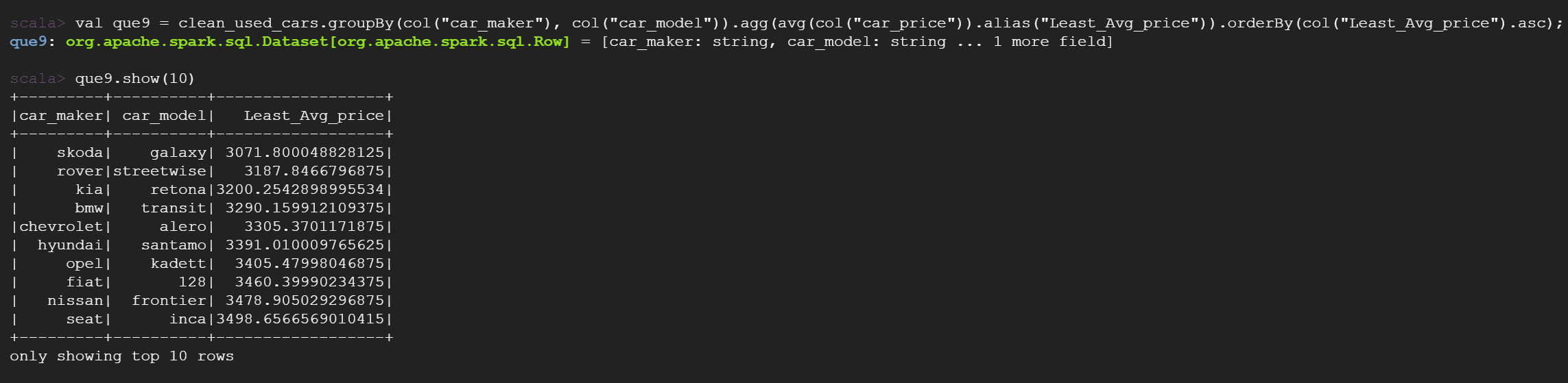
List the make and model for the cars with the top 10 highest average price

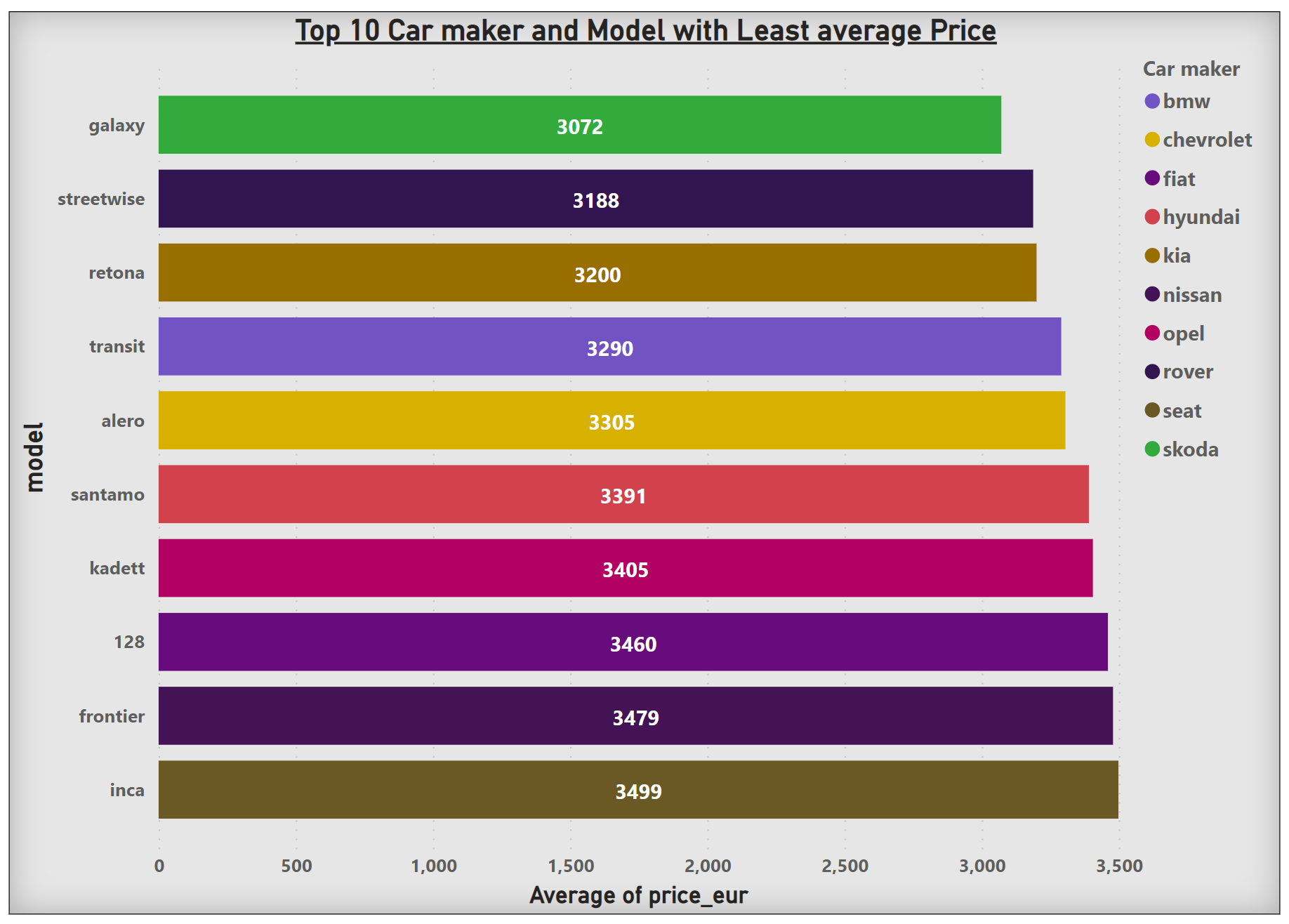




**Question 2**:-

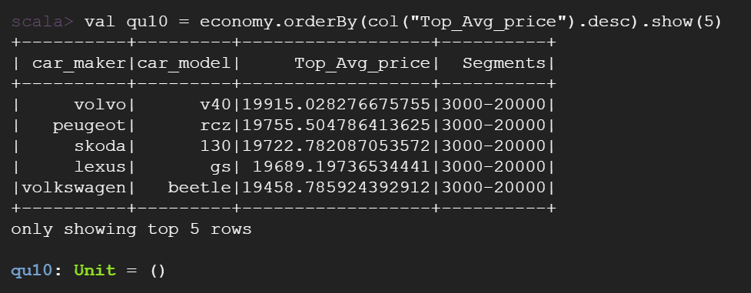
List the make and model for the cars with the top 10 lowest average price

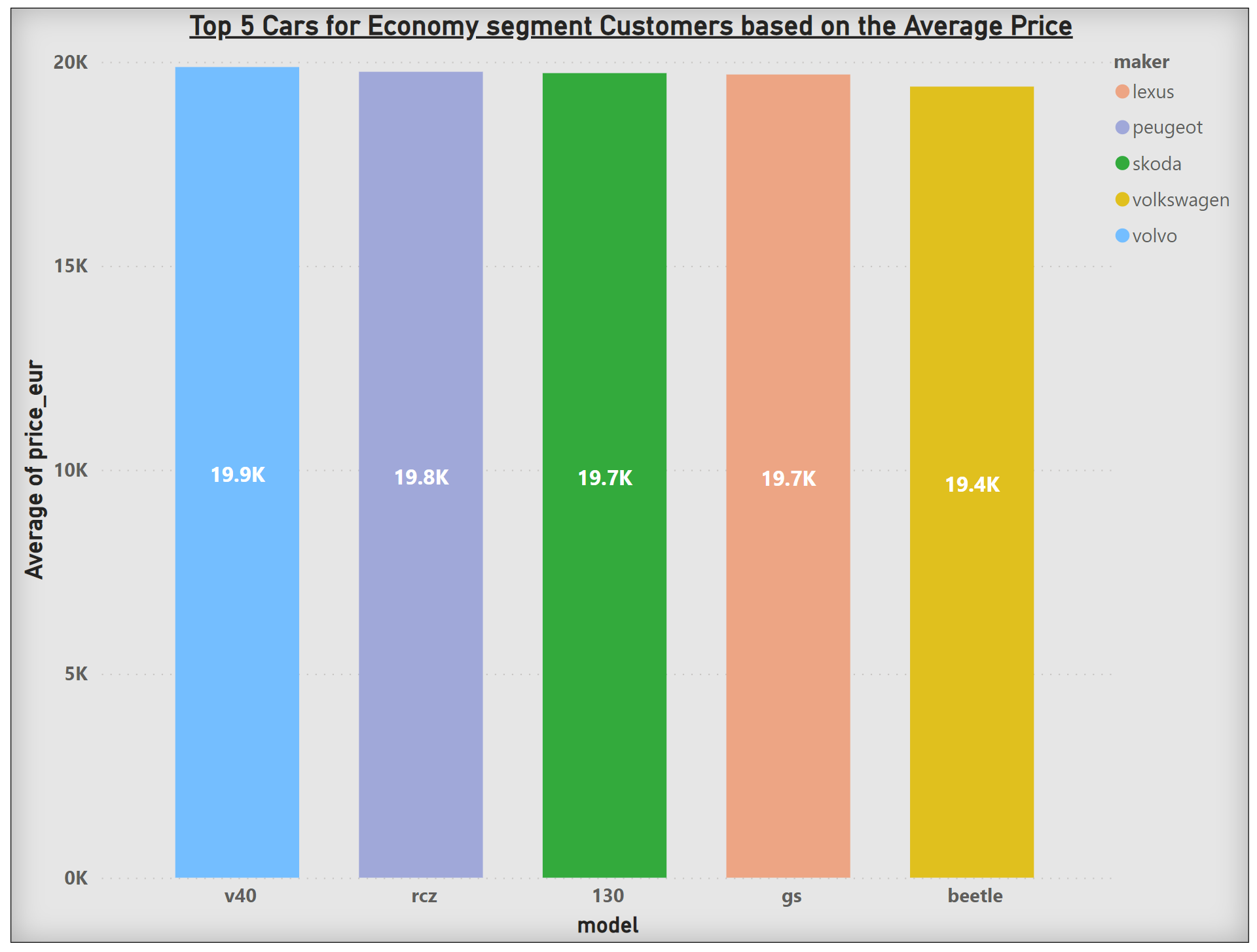




**Question 3**:-

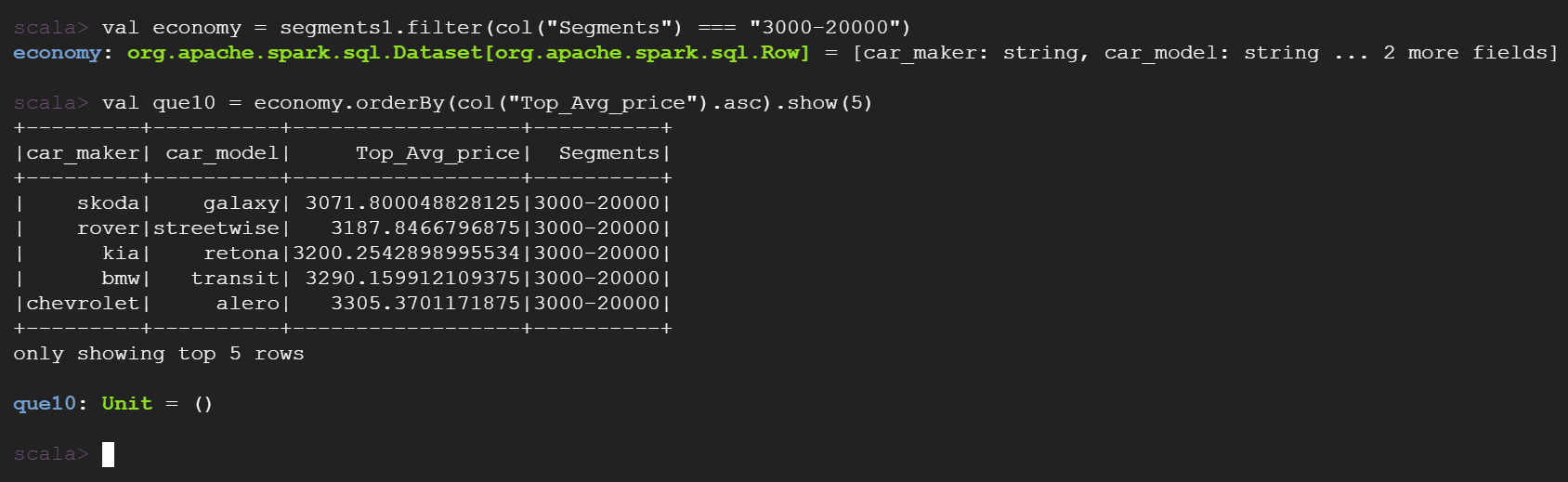
List the top five make and model for Economic segment customers price range- 3000 ≤ price < 20,000

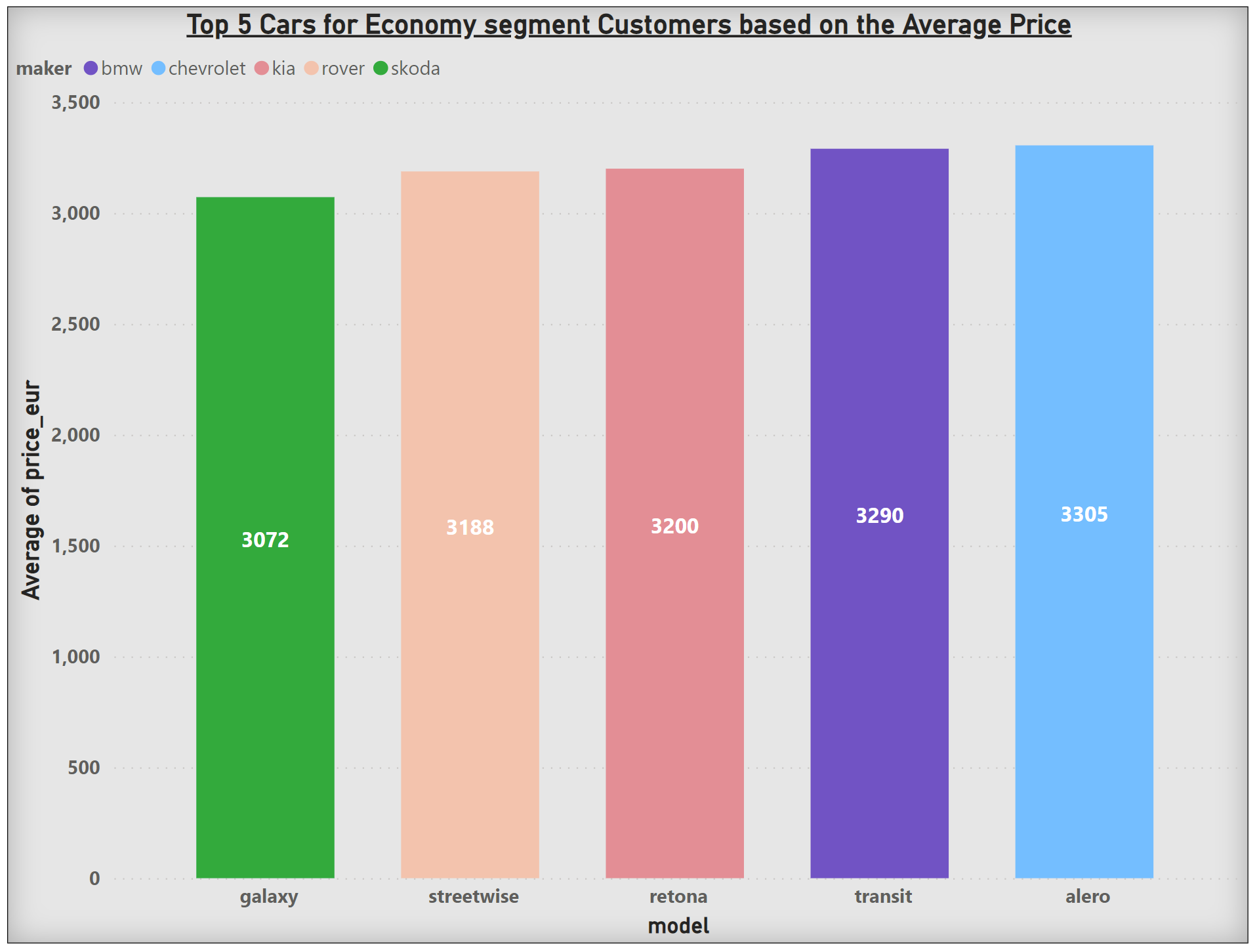




**Question 4**:-

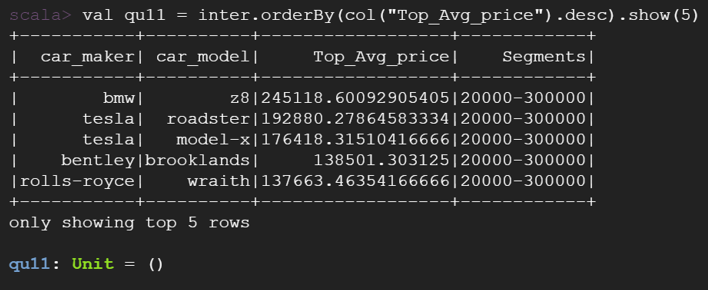
List the least five make and model for Economic segment customers price range-3000 ≤ price < 20,000

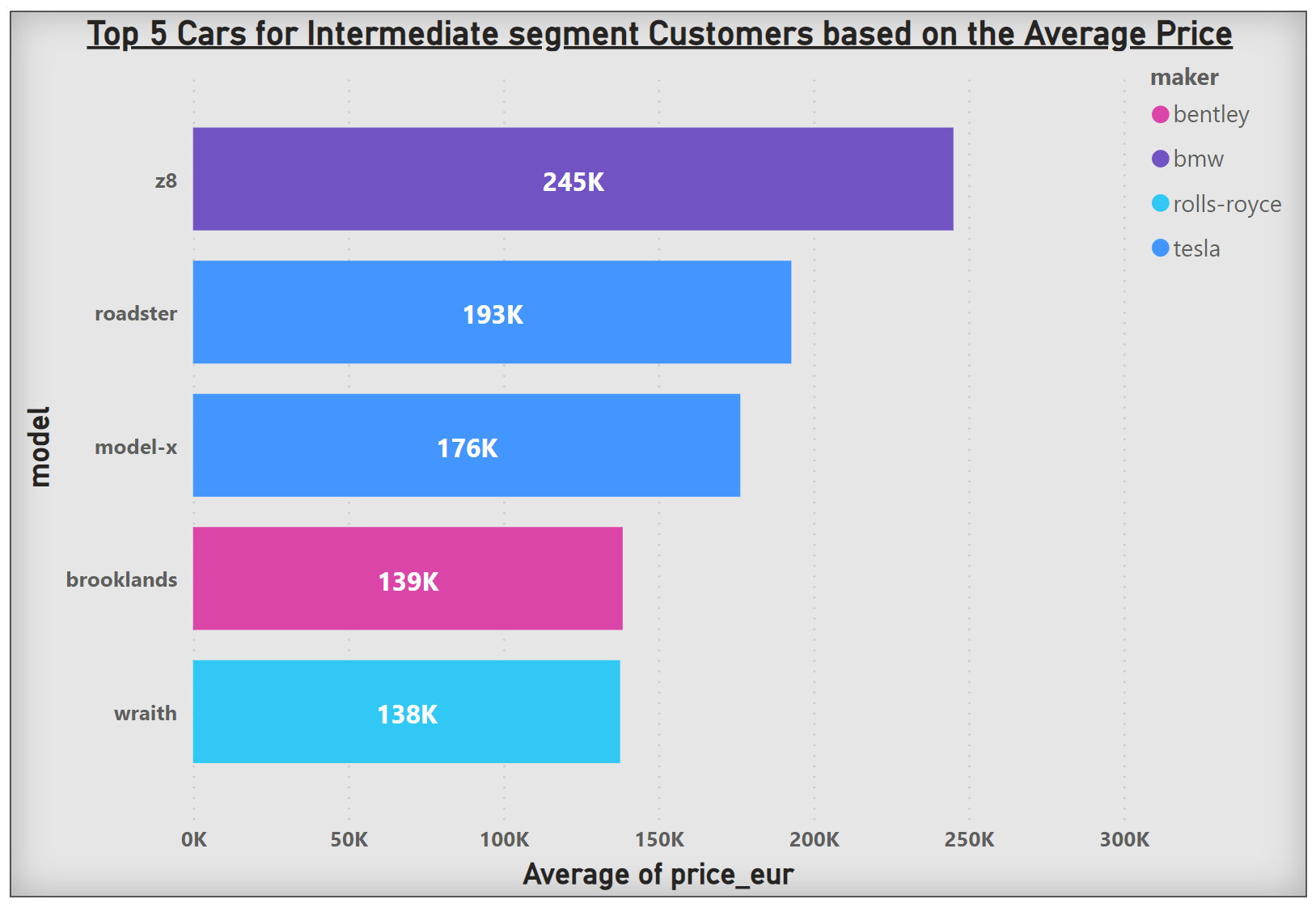




**Question 5**:-

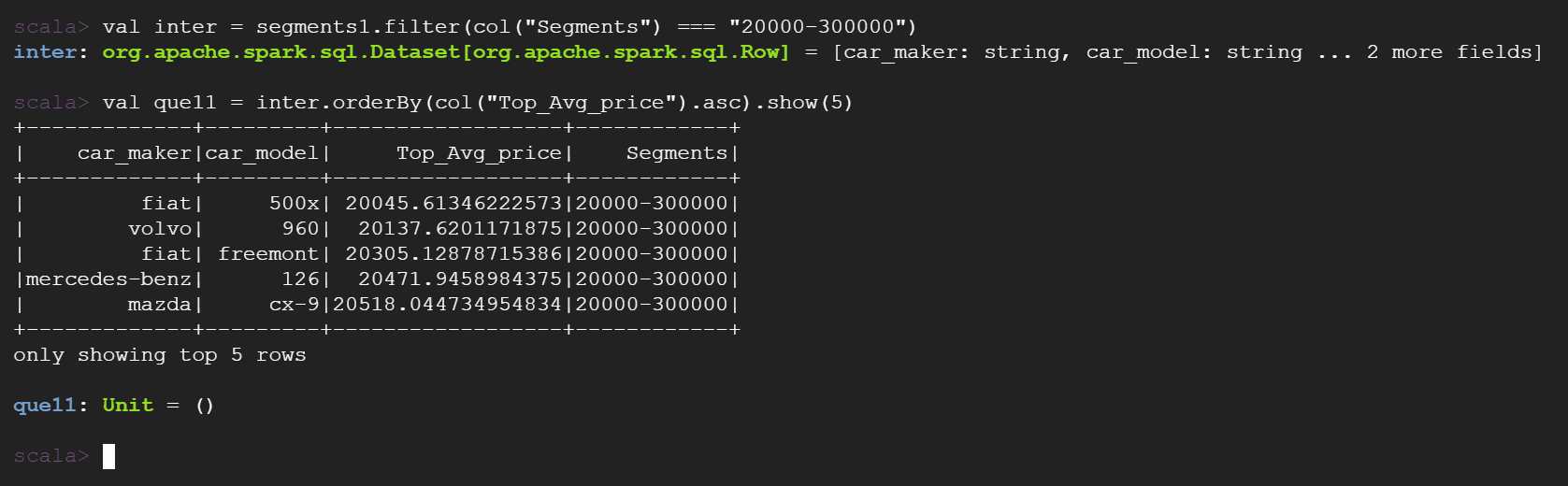
List the top five make and model for Intermediate segment customers price range-3000 ≤ price < 20,000

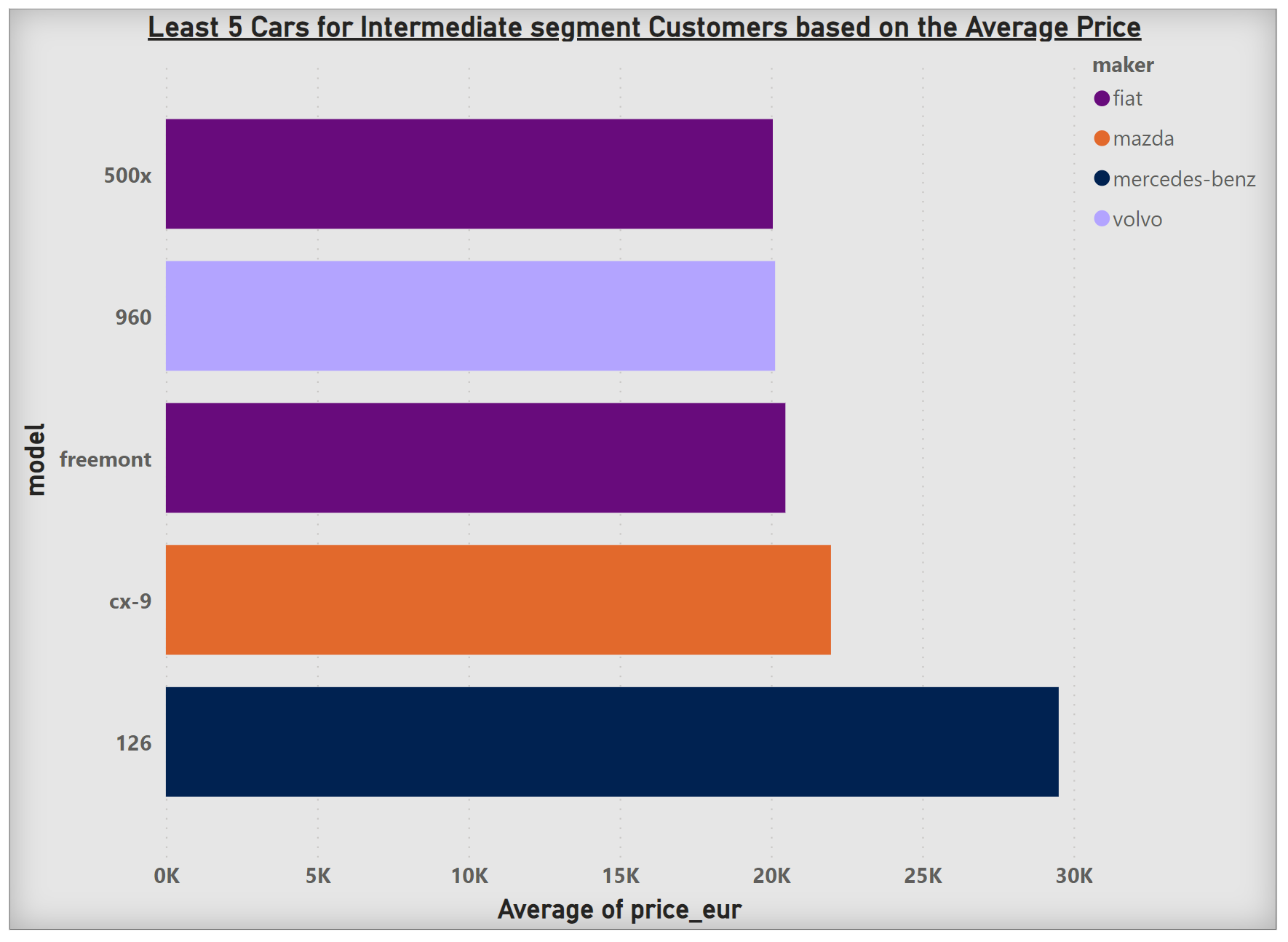




**Question 6**:-

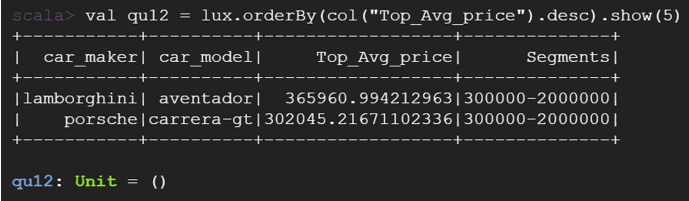
List the least make and model for Intermediate segment customers price range-3000 ≤ price < 20,000

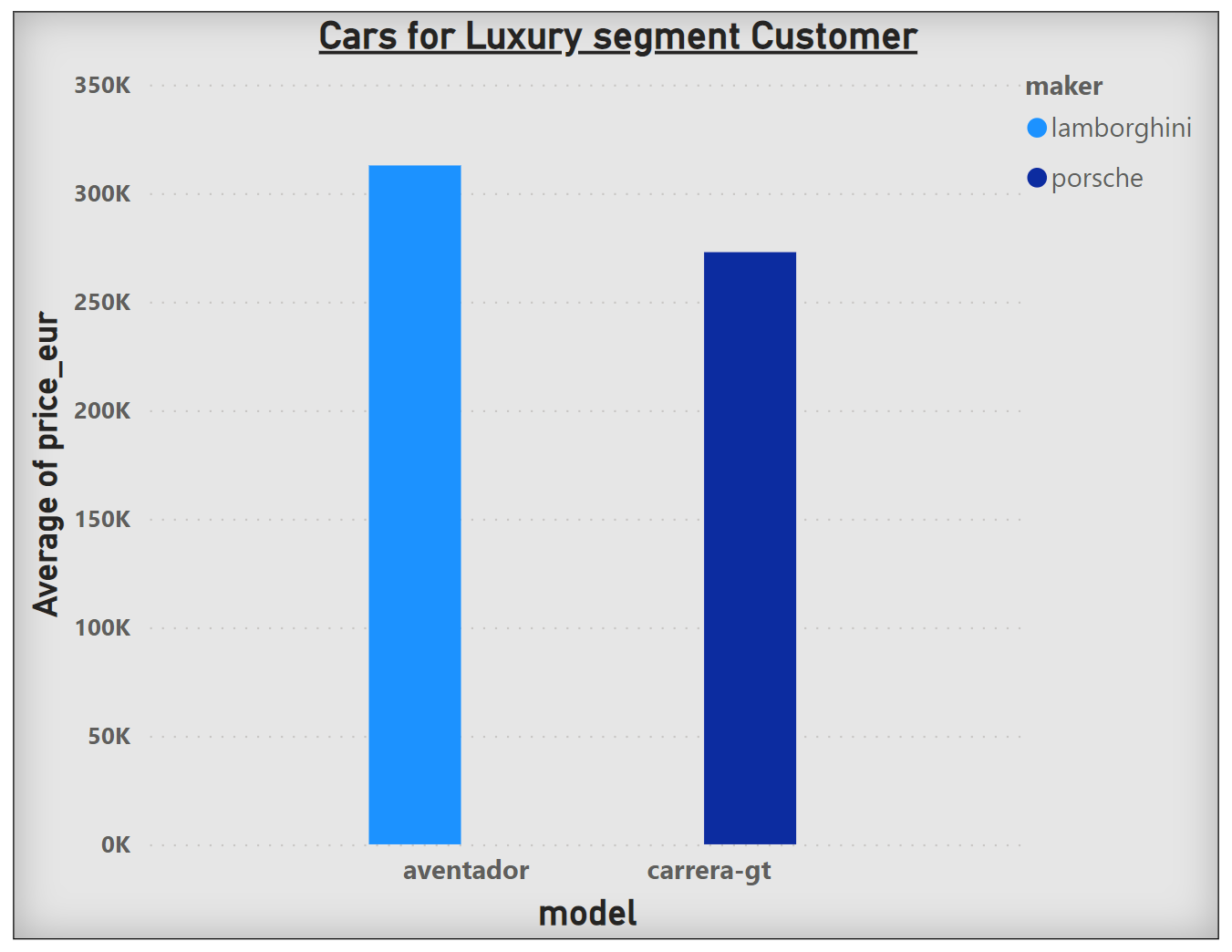




**Question 7**:-

List the make and model for the Luxury segment customers price range- 300,000 ≤ price < 2000,000





**Conclusion**

Recommendation based on the above analysis are stated below,

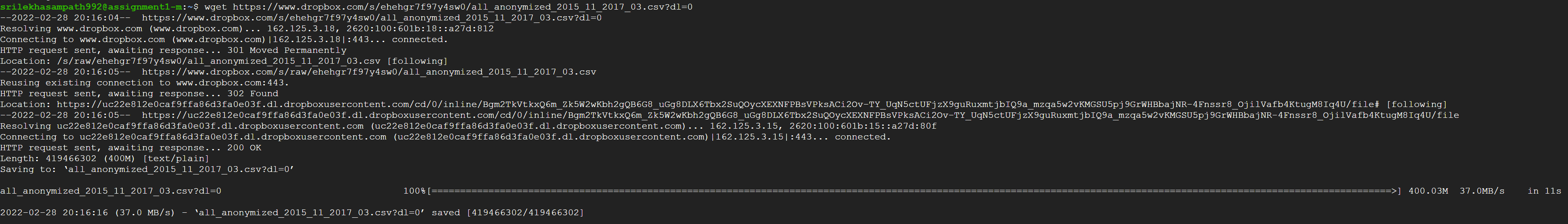
* Based on the average car pricing, best cars for Economy customers are Skoda-galaxy, Rover-streetwise, Kia-retona, BMW-transit and Chevrolet-alero.
* Based on the average car pricing, best cars for Intermediate customers are Fiat-500x, Volvo-960, Fiat-freemont , Mercedes Benz-126 and Mazda-cx9.
* Based on the average car pricing, best cars for Luxury customers are Lamborghini-aventador and Porsche-carrera gt.

**Appendix**:

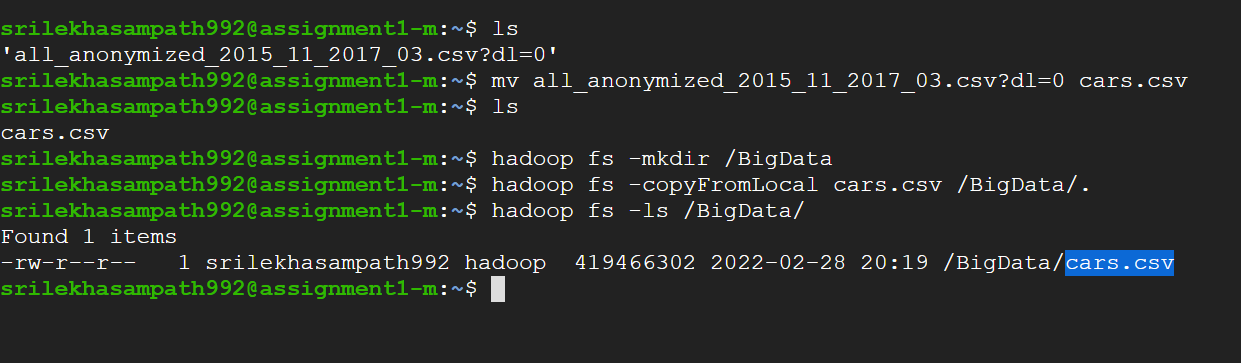
Answers to Assignment questions using Spark

The dataset is loaded using wget command,

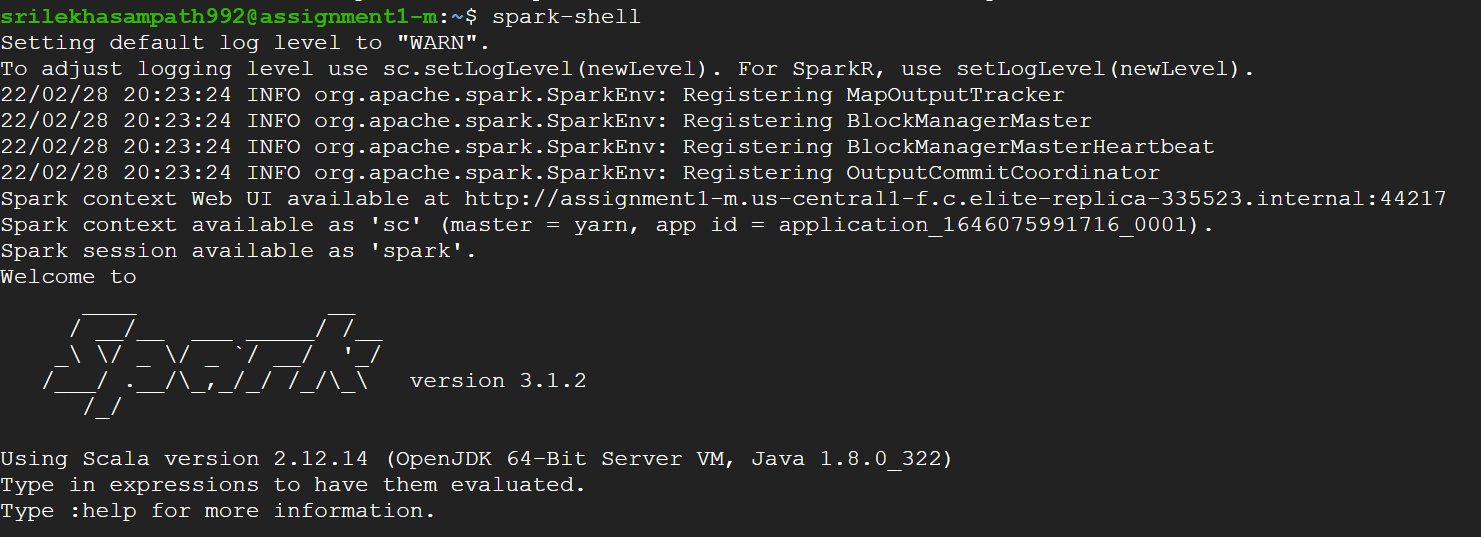
wget <https://www.dropbox.com/s/ehehgr7f97y4sw0/all_anonymized_2015_11_2017_03.csv?dl=0>



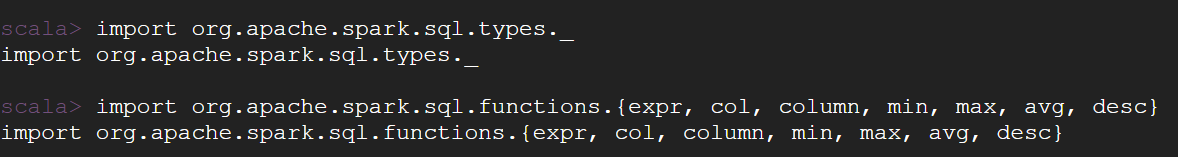
Creating a Directory and Copying the dataset in it,



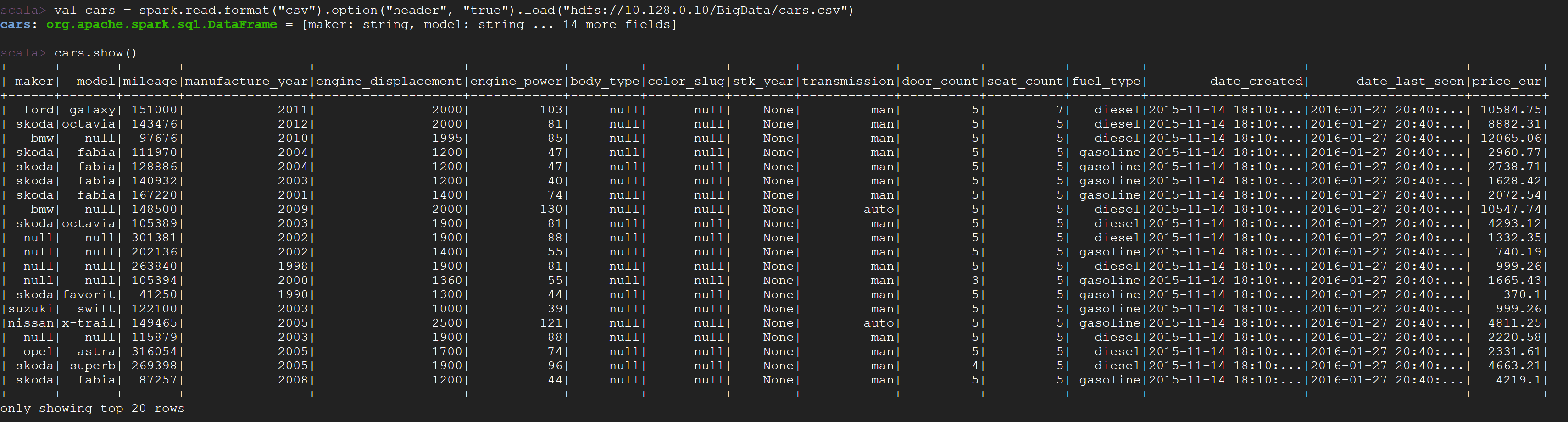
Initiating spark-shell



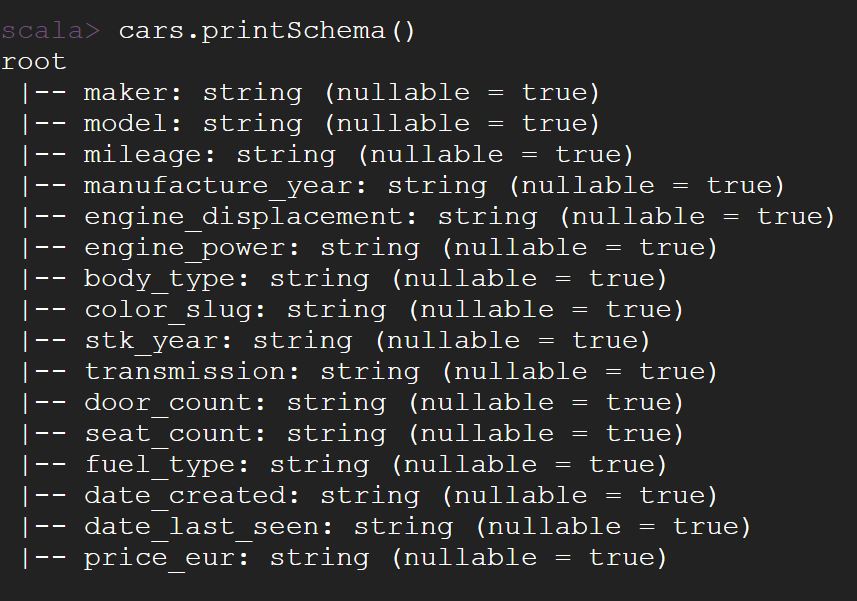
Importing necessary libraries,



Creating a new DataFrame to load the data into spark from local,

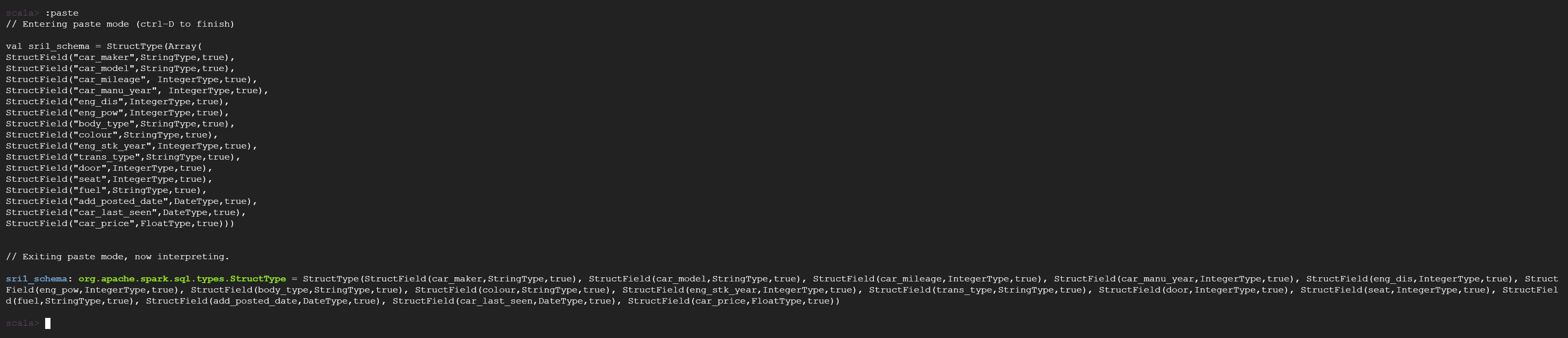


Initial Datatypes of the variables in the dataset,



**Question 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

Creating a new schema called ‘sri1\_schema’ to change the data types of the variables,

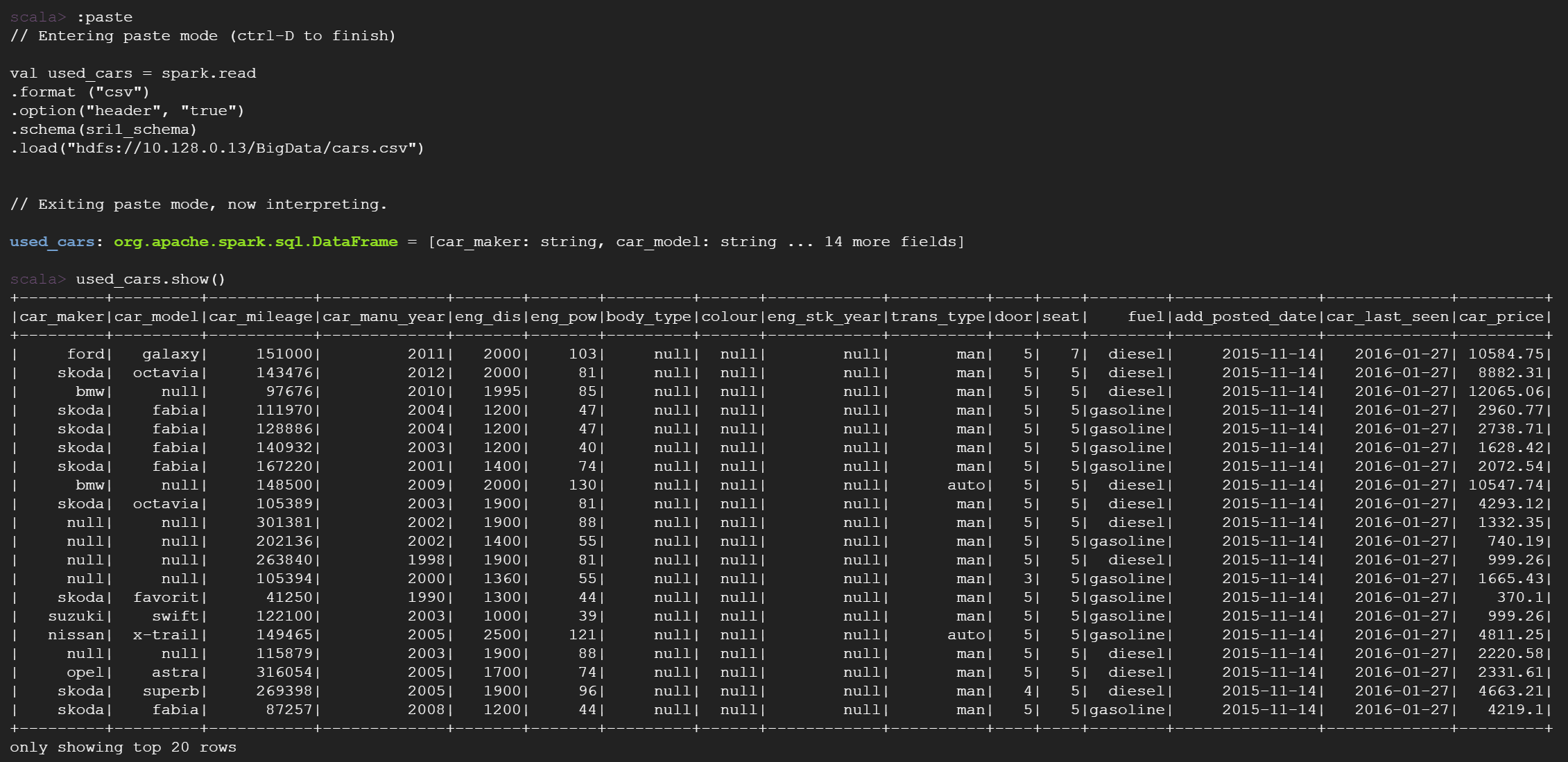


**Question 2:** Look at the date column of the table used\_cars. Why does the date column have all NULL values?

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

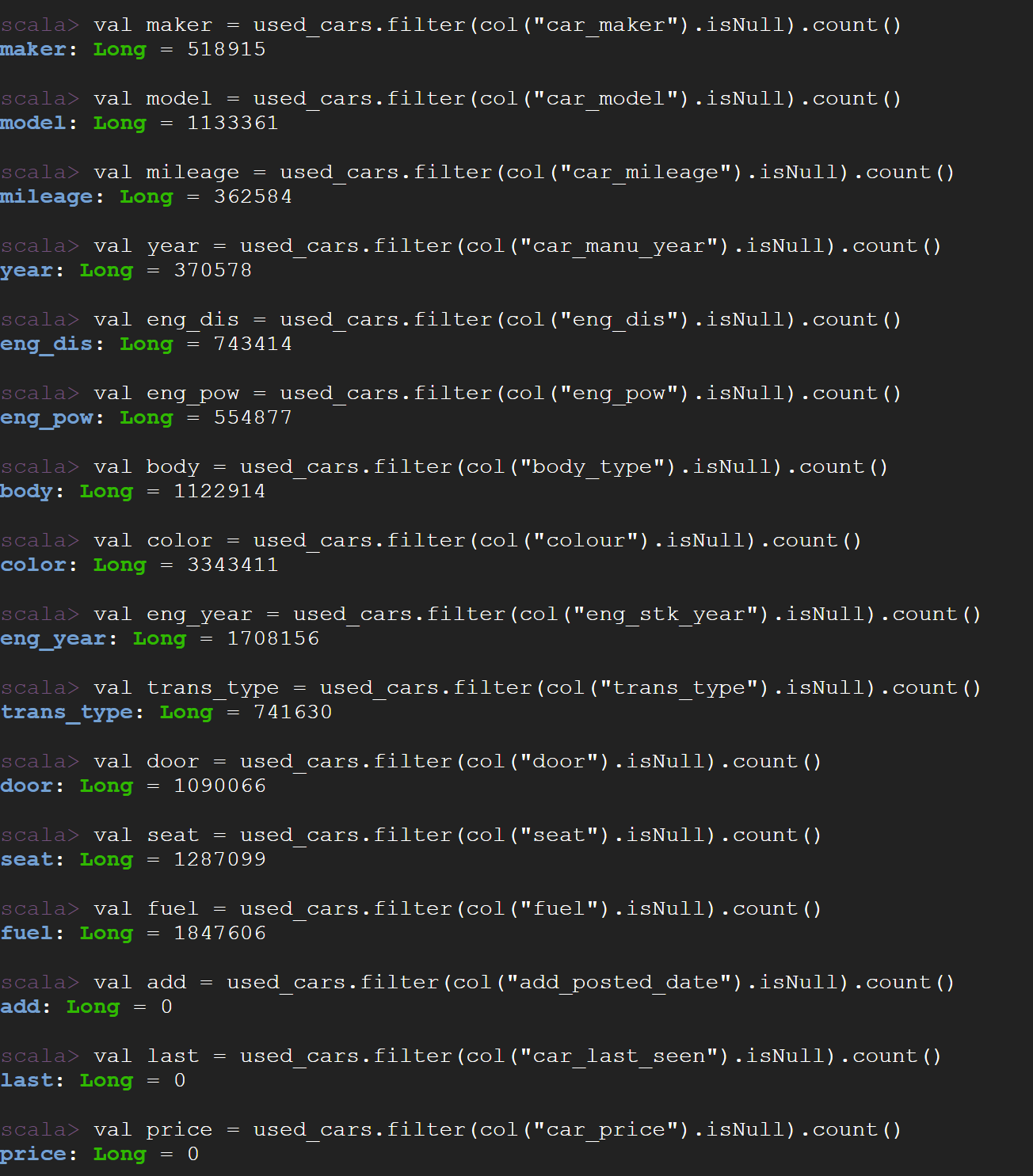
Data types for both the date column has been changed from ‘String’ to ‘Date’

A new Data Frame called ‘used\_cars’ has been created with the above-mentioned schema,



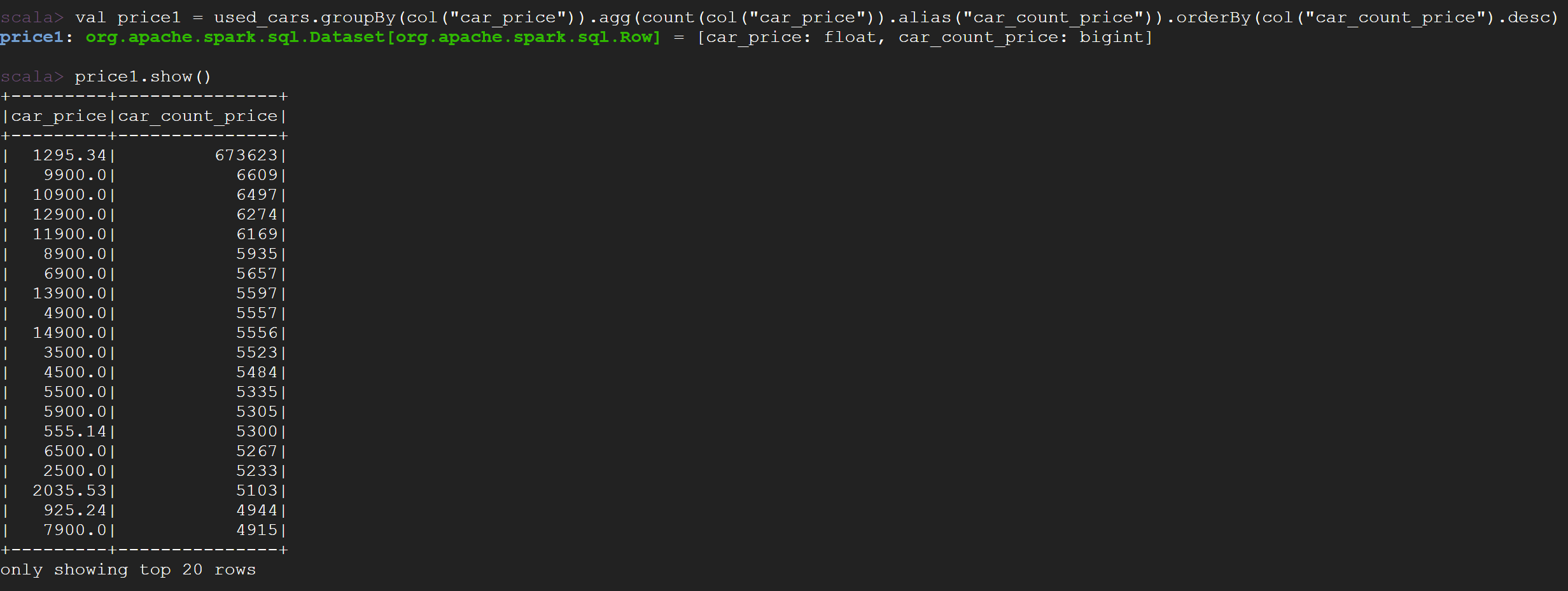
**Question 4:** Write Spark DataFrames queries to see how many missing values you have in each attribute? Based on the results, document how many missing values in each column we have. Especially, mention those columns with more than 50% missing values.

Variables with above 50% of missing values are - color\_slug (94%) and fuel\_type (52%).



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

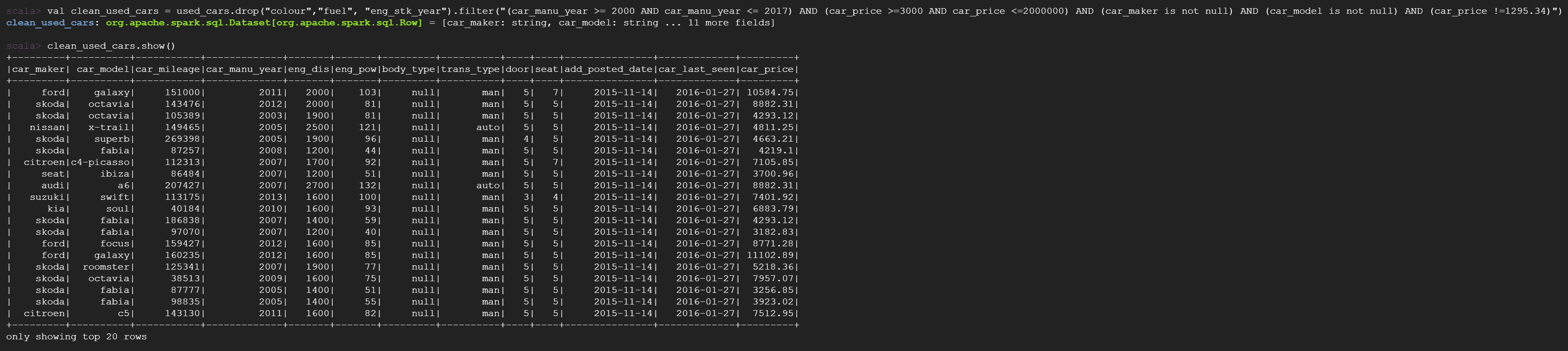
Car Price ‘1295.34’ is getting repeated ’673623’ times.



**Question 6**: Write a Spark DataFrames query to create a new table called clean\_used\_cars from used\_cars with the following conditions:

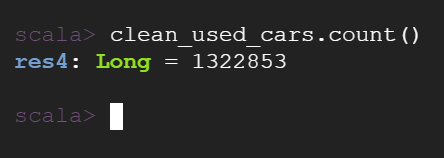
* Drop the columns with more than 50% missing values
* The manufacture year between 2000 and 2017 including 2000 and 2017
* Both maker and model exist in the row
* The price range is from 3000 to 2000,000 (3000 ≤ price ≤ 2000,000)
* Remove any price you singled out in Step 3 (i.e., a price that repeats too frequently for a random set of ads).

A new DataFrame ‘clean\_used\_cars’ is created with the above-mentioned conditions.

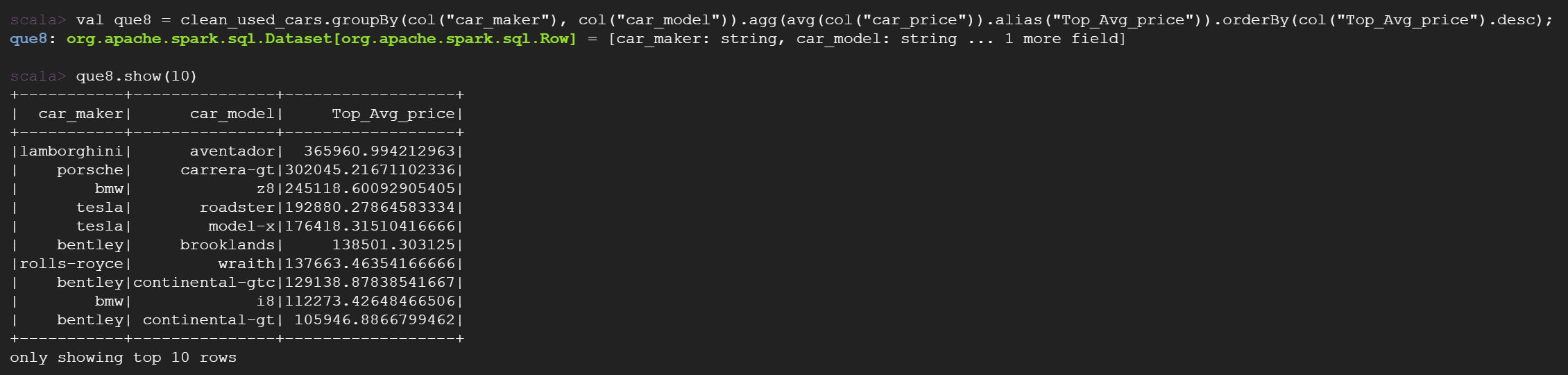


**Question 7**: Write a Spark DataFrames query to find how many records remained clean\_used\_cars

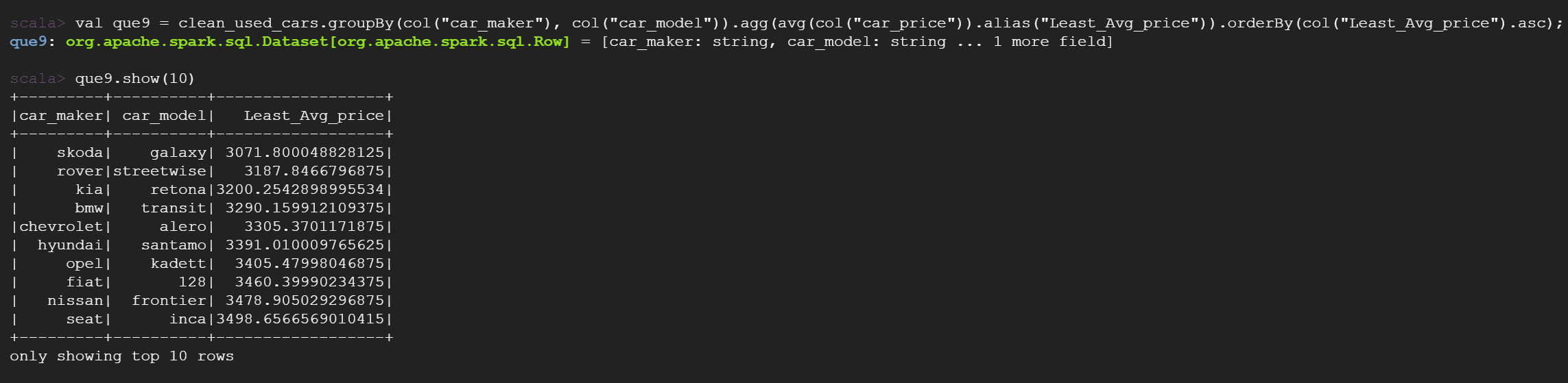
Total count of the records: 1322853



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



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

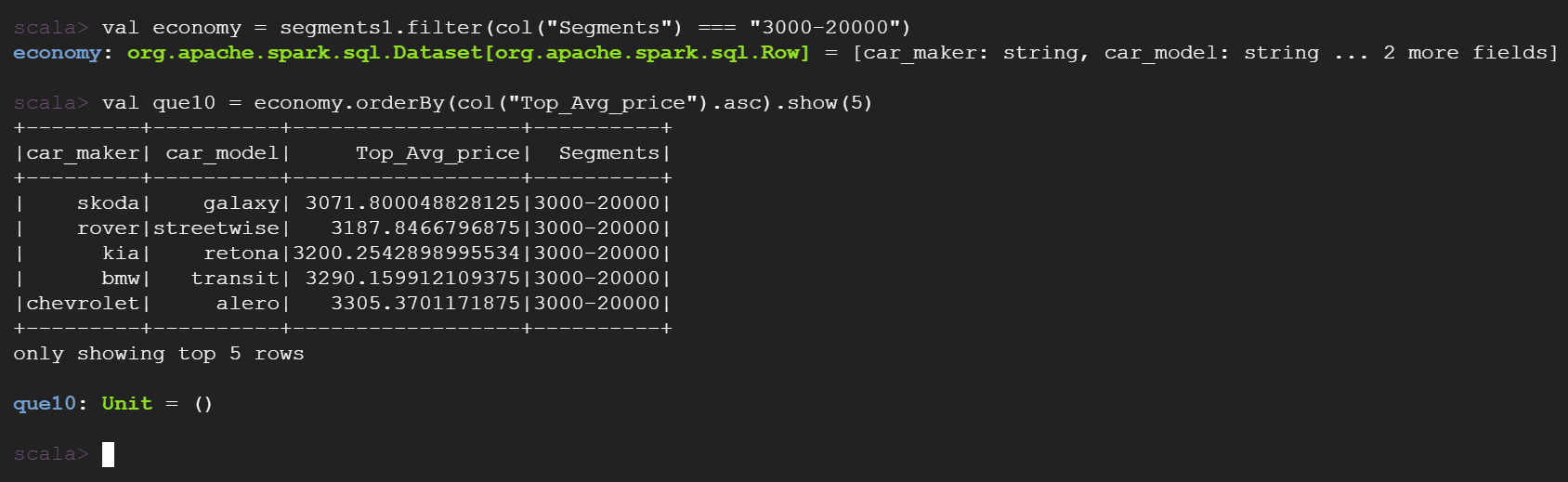


To answer Question 10 – 12, a new DataFrame called ‘segments1’ has been created with a new variable ‘Segments’ to differentiate Economy, Intermediate and Luxury customers.

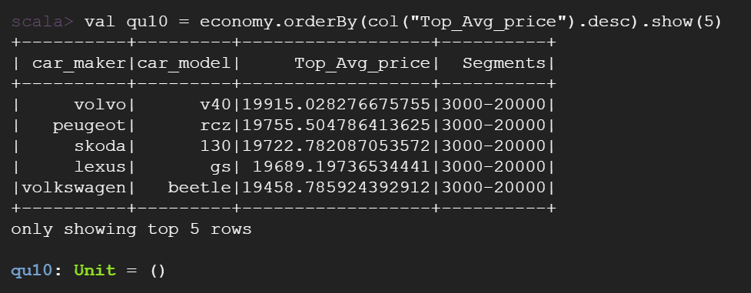


**Question 10**: Write a Spark DataFrames query to recommend top five make and model for Economic segment customers (Top five manufacturers in the 3000 to 20,000 price range;3000≤price<20,000) – based on the top average price.

Least 5 cars for Economy segment customers are,

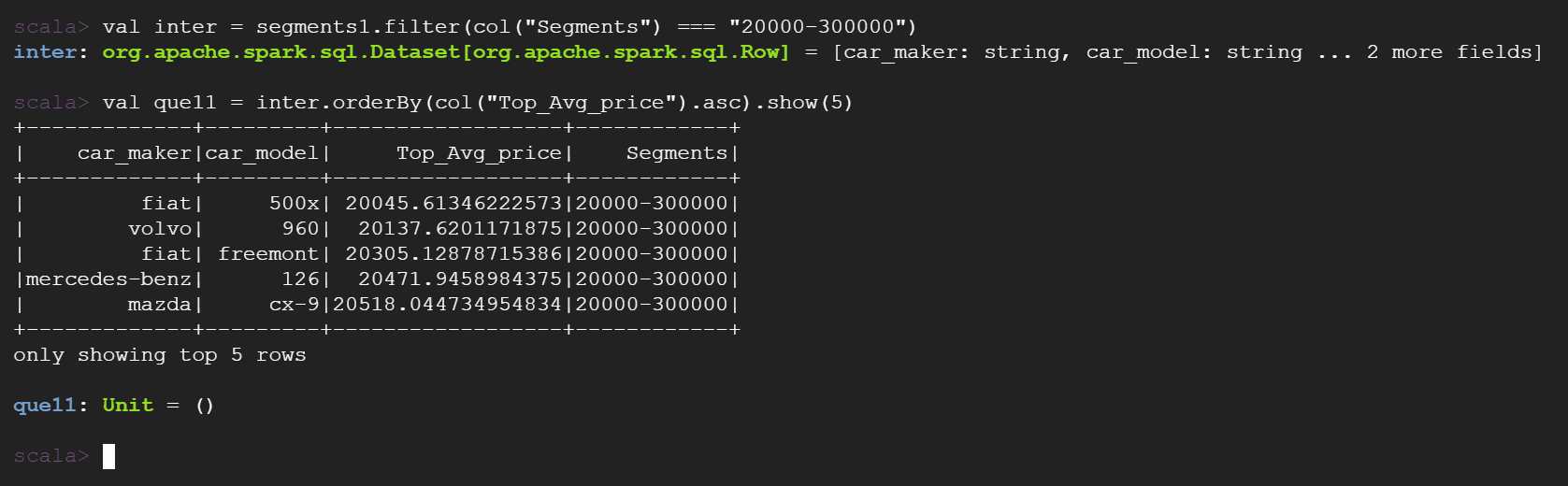


Top 5 cars for Economy segment customers are,

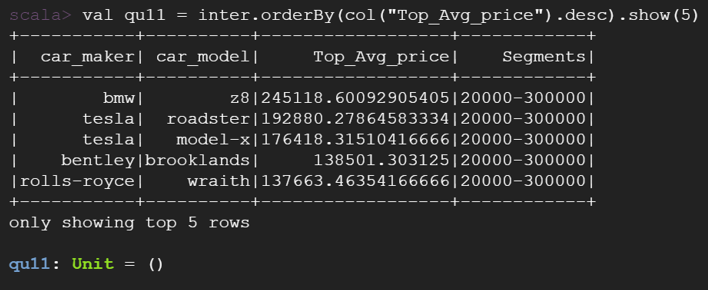


**Question 11**: Write a Spark DataFrames query to recommend top five make and model for Intermediate segment customers (Top five manufacturers in the 20,000 to 300,000 price range; 3000≤price<20,000) - based on the top average price.

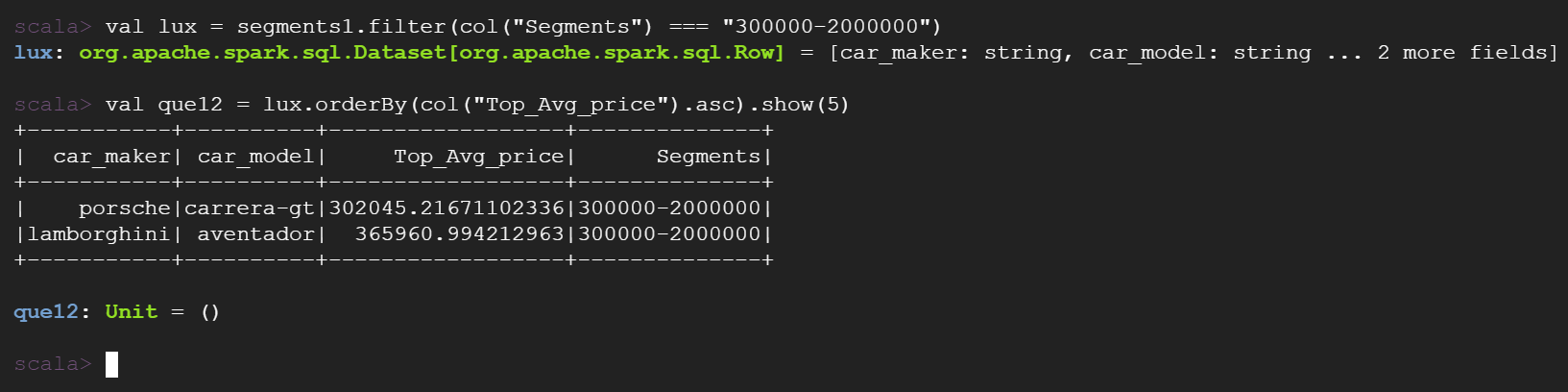
Least 5 cars for Intermediate segment customers are,



Top 5 cars for Intermediate segment customers are,



**Question 12**: Write a Spark DataFrames query to recommend the top five make and model for the Luxury segment customers (Top five manufacturers in the 300,000 to 2000,000 price range; 300,000≤price<2000,000) - based on the top average price.



**Scala Codes:**

The dataset is loaded using wget command,

wget <https://www.dropbox.com/s/ehehgr7f97y4sw0/all_anonymized_2015_11_2017_03.csv?dl=0>

Creating a Directory and Copying the dataset in it,

> mv all\_anonymized\_2015\_11\_2017\_03.csv?dl=0 cars.csv

> hadoop fs -mkdir /BigData

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

> hadoop fs -ls /BigData/

Importing necessary libraries,

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

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

Creating a new DataFrame to load the data into spark from local,

> val cars = spark.read.format("csv").option("header", "true").load ("hdfs://10.128.0.13/BigData/cars.csv")

> cars.show()

Printing the schema to view the Datatypes of the variables in the dataset,

cars.printSchema()

**Question 1, 2 and 3:**

Creating a new schema called ‘sri\_schema’ to change the data types of the variables,

Data types for both the date column has been changed from ‘String’ to ‘Date’

val sri\_schema = StructType(Array(StructField("car\_maker",StringType,true),StructField("car\_model",StringType,true),StructField("car\_mileage",IntegerType,true),StructField("car\_manu\_year",StringType,true),StructField("eng\_dis",IntegerType,true),StructField("eng\_pow",IntegerType,true),StructField("body\_type",StringType,true),StructField("colour",StringType,true),StructField("eng\_stk\_year",IntegerType,true),StructField("trans\_type",StringType,true),StructField("door",IntegerType,true),StructField("seat",IntegerType,true),StructField("fuel",StringType,true),StructField("add\_posted\_date",DateType,true),StructField("car\_last\_seen",DateType,true),StructField("car\_price",FloatType,true)))

A new Data Frame called ‘used\_cars’ has been created with the above-mentioned schema,

val used\_cars = spark.read.format ("csv").option("header", "true")

.schema(sri\_schema)

.load("hdfs://10.128.0.13/BigData/cars.csv")

**Question 4:** To get the count of null values in each field

> val maker = used\_cars.filter(col("car\_maker").isNull).count()

> val model = used\_cars.filter(col("car\_model").isNull).count()

> val mileage = used\_cars.filter(col("car\_mileage").isNull).count()

> val year = used\_cars.filter(col("car\_manu\_year").isNull).count()

> val eng\_dis = used\_cars.filter(col("eng\_dis").isNull).count()

> val eng\_pow = used\_cars.filter(col("eng\_pow").isNull).count()

> val body = used\_cars.filter(col("body\_type").isNull).count()

> val color = used\_cars.filter(col("colour").isNull).count()

> val eng\_year = used\_cars.filter(col("eng\_stk\_year").isNull).count()

> val trans\_type = used\_cars.filter(col("trans\_type").isNull).count()

> val door = used\_cars.filter(col("door").isNull).count()

> val seat = used\_cars.filter(col("seat").isNull).count()

> val fuel = used\_cars.filter(col("fuel").isNull).count()

> val add = used\_cars.filter(col("add\_posted\_date").isNull).count()

> val last = used\_cars.filter(col("car\_last\_seen").isNull).count()

> val price = used\_cars.filter(col("car\_price").isNull).count()

**Question 5**:

> val price1 = used\_cars.groupBy(col("car\_price")).agg(count(col("car\_price")).alias("car\_count\_price")).orderBy(col("car\_count\_price").desc)

> price1.show()

**Question 6**:

val clean\_used\_cars = used\_cars.drop("colour","fuel", "eng\_stk\_year").filter("(car\_manu\_year >= 2000 AND car\_manu\_year <= 2017) AND (car\_price >=3000 AND car\_price <=2000000) AND (car\_maker is not null) AND (car\_model is not null) AND (car\_price !=1295.34)")

**Question 7**:

clean\_used\_cars.count()

**Question 8**:

val que8 = clean\_used\_cars.groupBy(col("car\_maker"), col("car\_model")).agg(avg(col("car\_price")).alias("Top\_Avg\_price")).orderBy(col("Top\_Avg\_price").desc);

**Question 9**:

val que9 = clean\_used\_cars.groupBy(col("car\_maker"), col("car\_model")).agg(avg(col("car\_price")).alias("Least\_Avg\_price")).orderBy(col("Least\_Avg\_price").asc);

To answer Question 10 – 12, a new DataFrame called ‘segments1’ has been created with a new variable ‘Segments’ to differentiate Economy, Intermediate and Luxury customers.

val segments1 = que8.withColumn("Segments", when(col("Top\_Avg\_price") >= "3000" && col("Top\_Avg\_price") < "20000" , "3000-20000").when(col("Top\_Avg\_price") >= "20000" && col("Top\_Avg\_price") < "300000" , "20000-300000") .when(col("Top\_Avg\_price") >= "300000" && col("Top\_Avg\_price") < "2000000" , "300000-2000000"))

**Question 10**:

Least 5 cars for Economy segment customers are,

> val economy = segments1.filter(col("Segments") === "3000-20000")

> val que10 = economy.orderBy(col("Top\_Avg\_price").asc).show(5)

Top 5 cars for Economy segment customers are,

> val qu10 = economy.orderBy(col("Top\_Avg\_price").desc).show(5)

**Question 11**:

Least 5 cars for Intermediate segment customers are,

> val inter = segments1.filter(col("Segments") === "20000-300000")

> val que11 = inter.orderBy(col("Top\_Avg\_price").asc).show(5)

Top 5 cars for Intermediate segment customers are,

> val qu11 = inter.orderBy(col("Top\_Avg\_price").desc).show(5)

**Question 12**:

> val lux = segments1.filter(col("Segments") === "300000-2000000")

> val que12 = lux.orderBy(col("Top\_Avg\_price").asc).show(5)

**References**

College, C. (2022, 03 10). *blackboard* . Retrieved from blackboard:

<https://gc.blackboard.com/ultra/courses/_328785_1/cl/outline>