Report on Analysis of Used Car Sales in Germany and Czech Republic

Submitted By:

Karthikeyan Suresh Kumar

200489370

BDAT

Time Table C

Table of Contents

Overview

This report provides an in-depth analysis of the used car sales business in Germany and the Czech Republic since 2015. The thorough analysis of the data available from 2015 has provided necessary insights and the current market trends

With the use of this analysis, the management would be able to take key decisions on whether or not to venture into the used car sales business.

Dataset Details

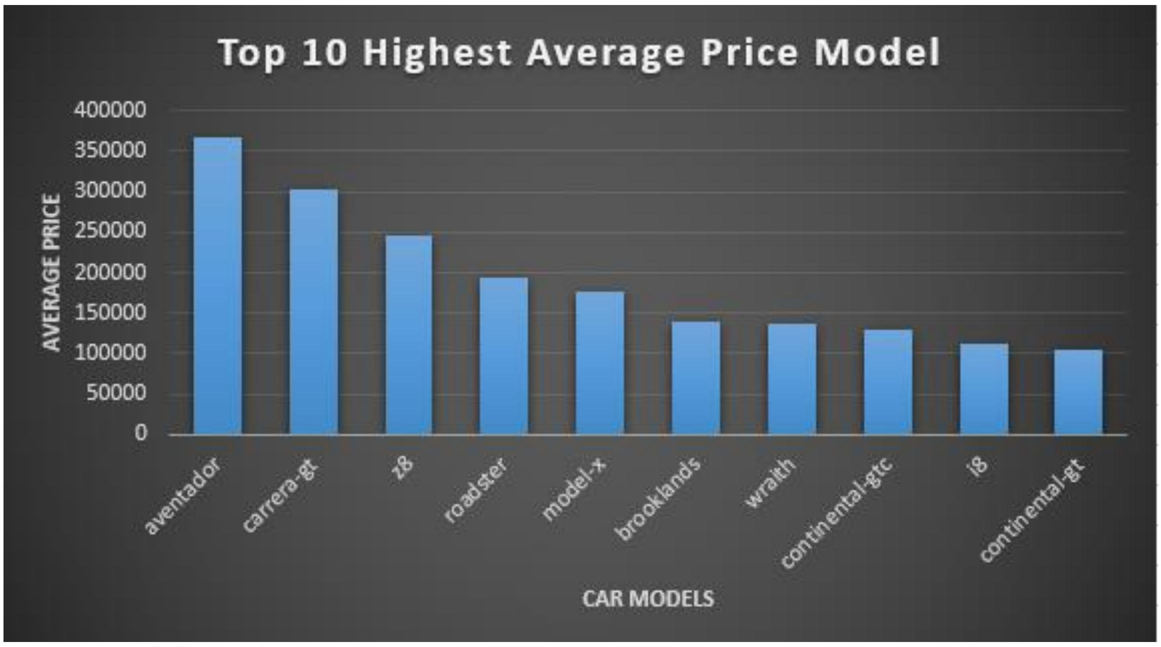
The dataset used for the analysis has been extracted from the Kaggle website. The dataset has a count of 3.5 million rows.

The dataset contains the following columns:

- maker
- model
- mileage
- manufacture_year
- engine_displacement
- engine_power
- body_type
- color_slug
- stk_year
- transmission
- door_count
- seat_count
- fuel_type
- date_created
- date_last_seen
- price_eur

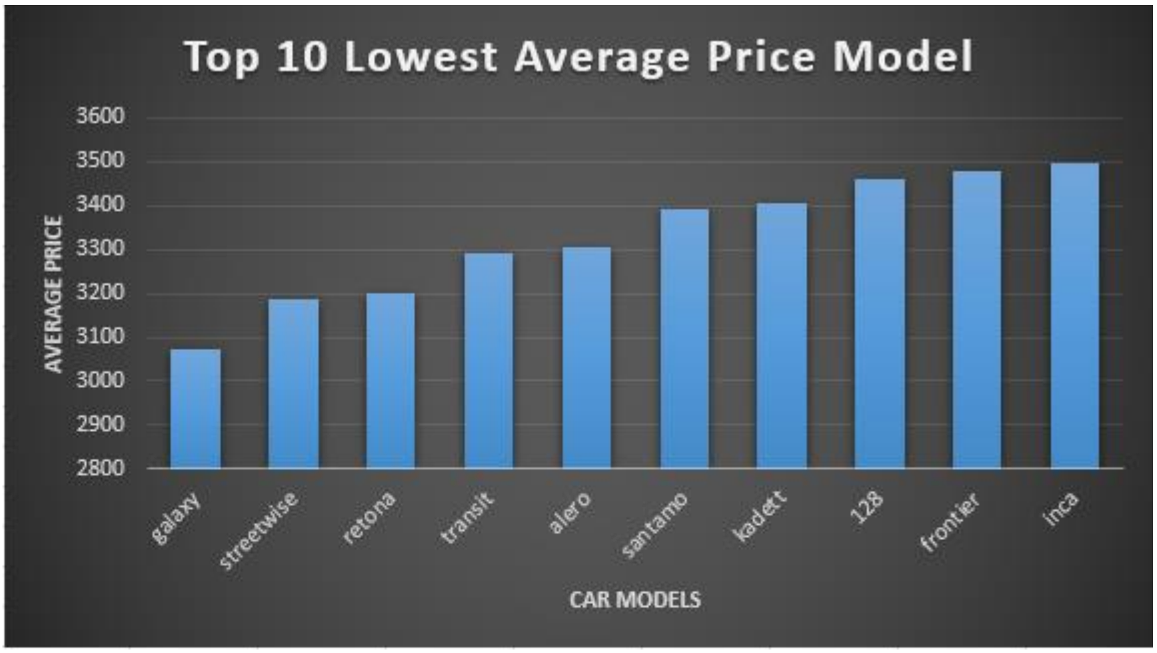
Analysis Questions

1) What are the top 10 cars having the highest average price?



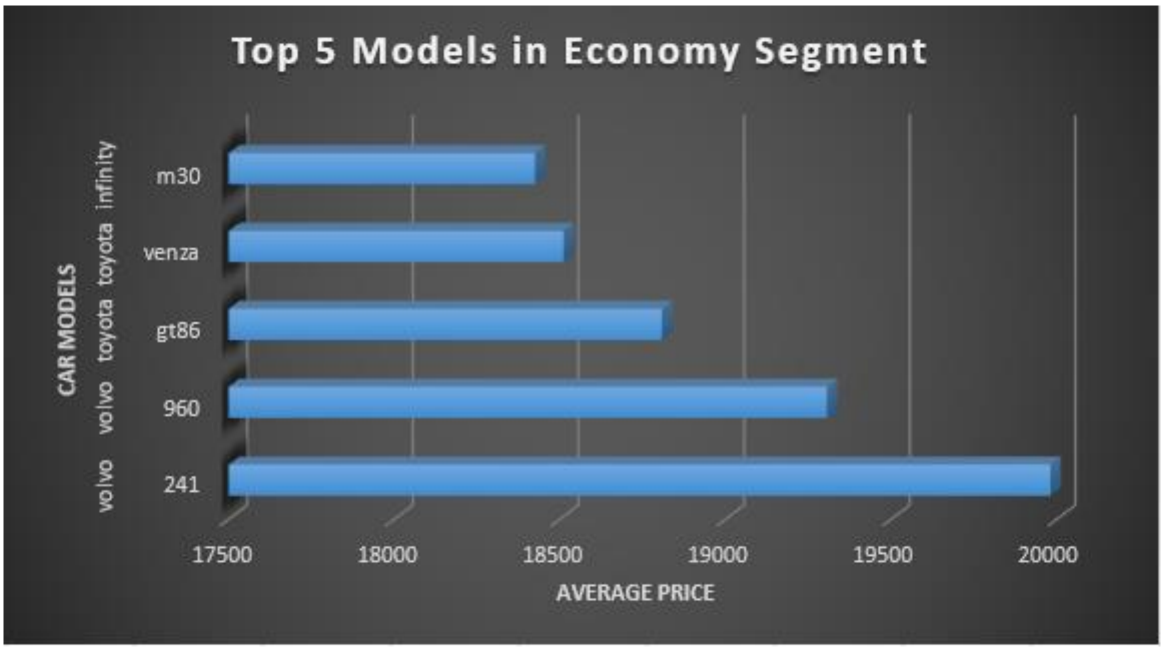
From the above visual, it can be noted that the average price of the top 10 cars with the highest average is above approximately 100000.

2) What are the top 10 cars having the lowest average price?



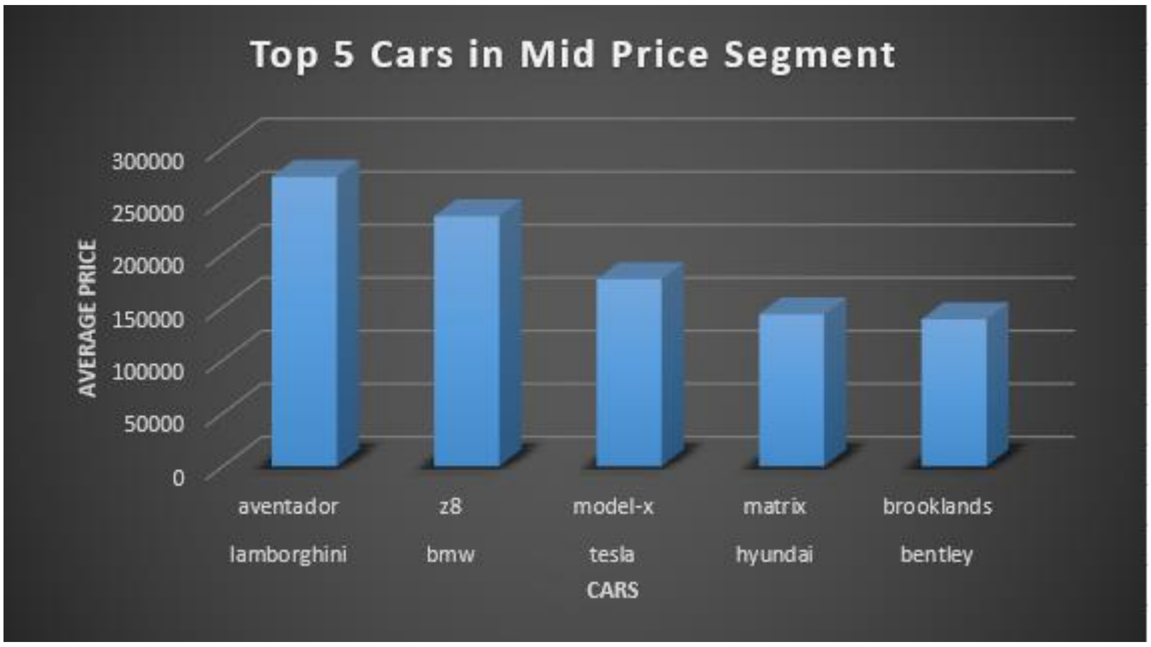
The lowest car model is Skoda Galaxy which has around 3080 Euros as an average price. The lowest 10 models having the average price is well below the 3500 Euros mark.

3) What are the top 5 models in the Economy Segment?



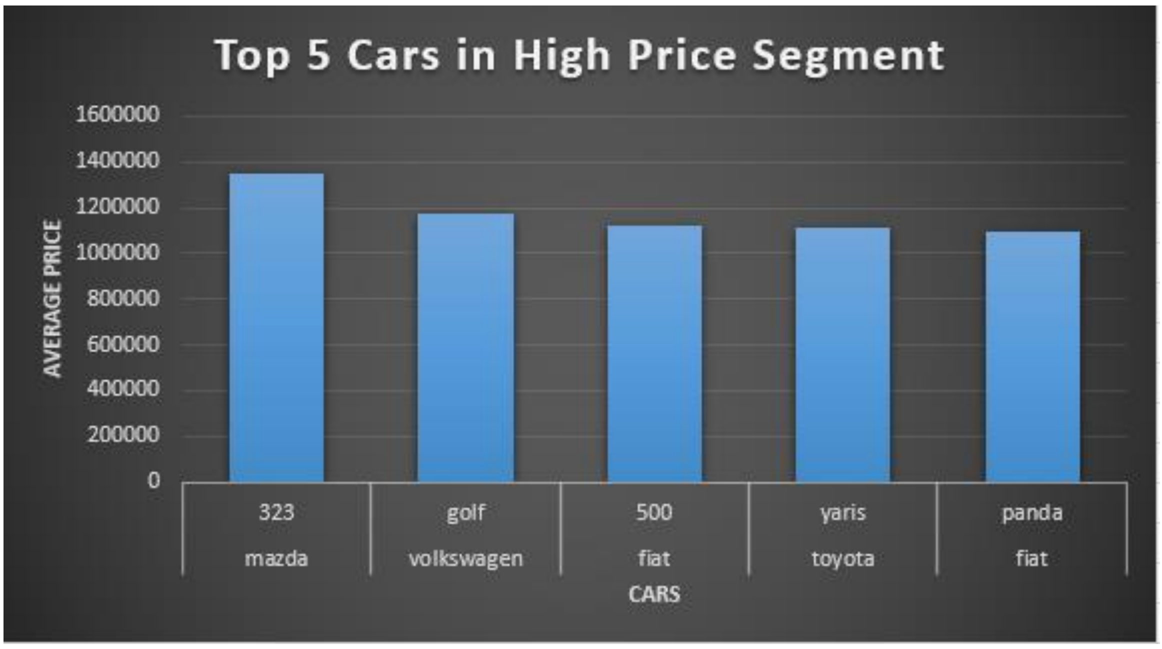
The best suitable cars for the economy model customers are Infinity m30, Toyota Venza, Toyota gt86, Volvo 960, and Volvo 241. These are the best suitable and popular car models which are well within the 3000-20000 Euros range and they can be targeted for sales for the Economy class.

4) What are the top 5 models in the Mid Price Segment?



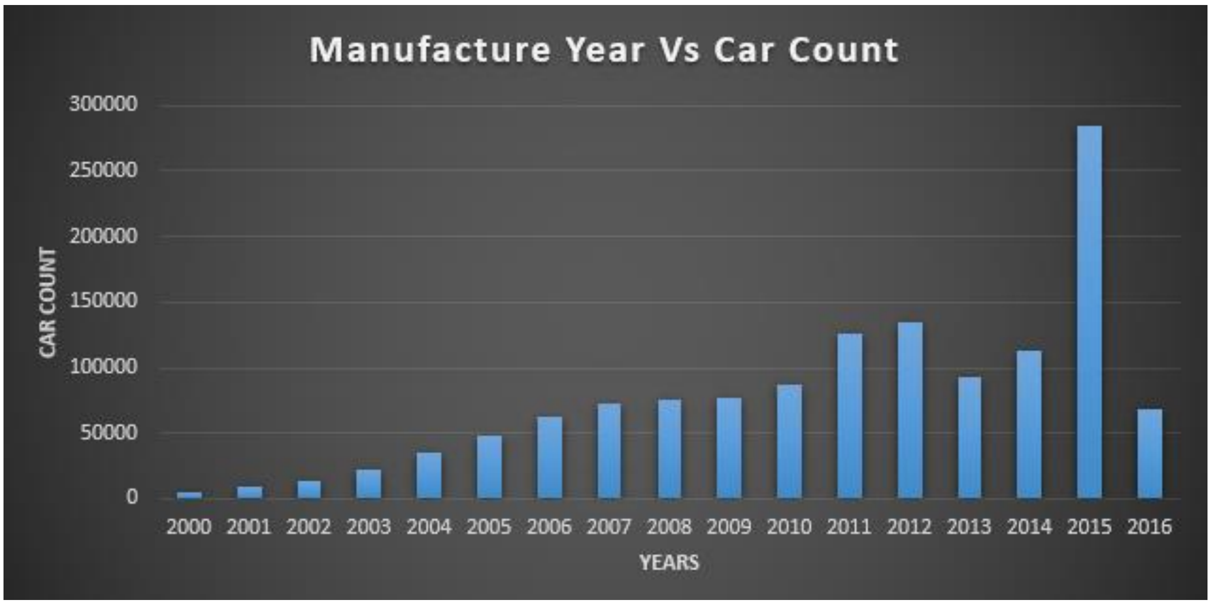
The mid-price segment is the range between 20000 – 300000. The best cars in the segment that can be targeted to the respective users are Lamborghini Aventador, BMW Z8, Tesla model-x, Hyundai matrix, and Bentley Brooklands.

5) **What are the top 5 models in the High-Price Segment?**



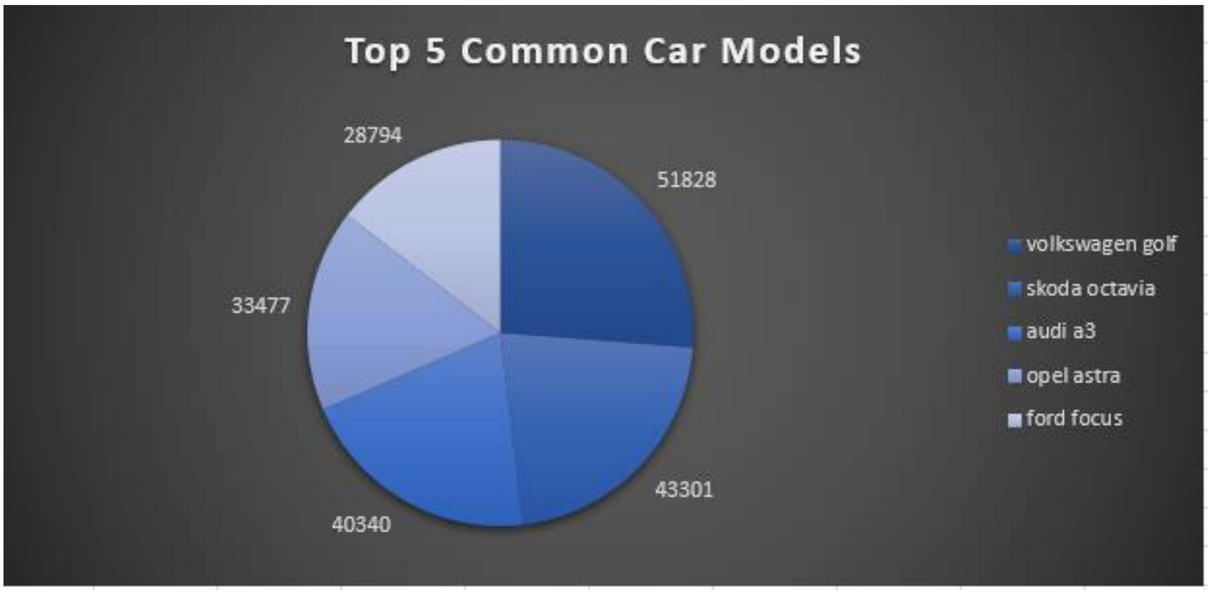
Inferring from the above visual it can be seen that, the cars Mazda 323, Volkswagen Golf, Fiat 500, Toyota Yaris, and Fiat Panda are the high-price segment cars which are in the range of 300000-2000000. These cars can be targeted towards the people with strong finances to make a sale.

6) **What are the most common Manufacture year cars for which the ads are posted?**



The above graph gives information about the age of cars which are posted for sale. Cars manufactured in the year 2015 are mostly put up for sale. As the dataset was taken in the year 2017, it can be seen that most of the cars listed for sale are around 5 or 6 years old.

7) What are the top 5 common car models posted in the ads?



The cars which top the most listed cars are Volkswagen Golf, Skoda Octavia, Audi a3, Opel Astra, and Ford Focus. These are the cars which are most likely bought by the people for the first hand or the worst case is that there is a problem related to those models which would have caused a surge in the listing of those cars.

Conclusion

Based on the analysis of the data and the insights obtained, it is seen that the market for the used car business is growing. With the right business tactics and approach, if an investment is made to explore the used car sales business, it can yield high returns.

References

The dataset for the analysis has been taken from the below link.

https://www.kaggle.com/mirosval/personal-cars-classifieds

Appendix

Assignment Questions

2) Look at the date column of the table used_cars. Why does the date column have all NULL values?
The dates columns were initially parsed as String datatype. Once the schema has been used while reading the data and it was converted into Date datatype.

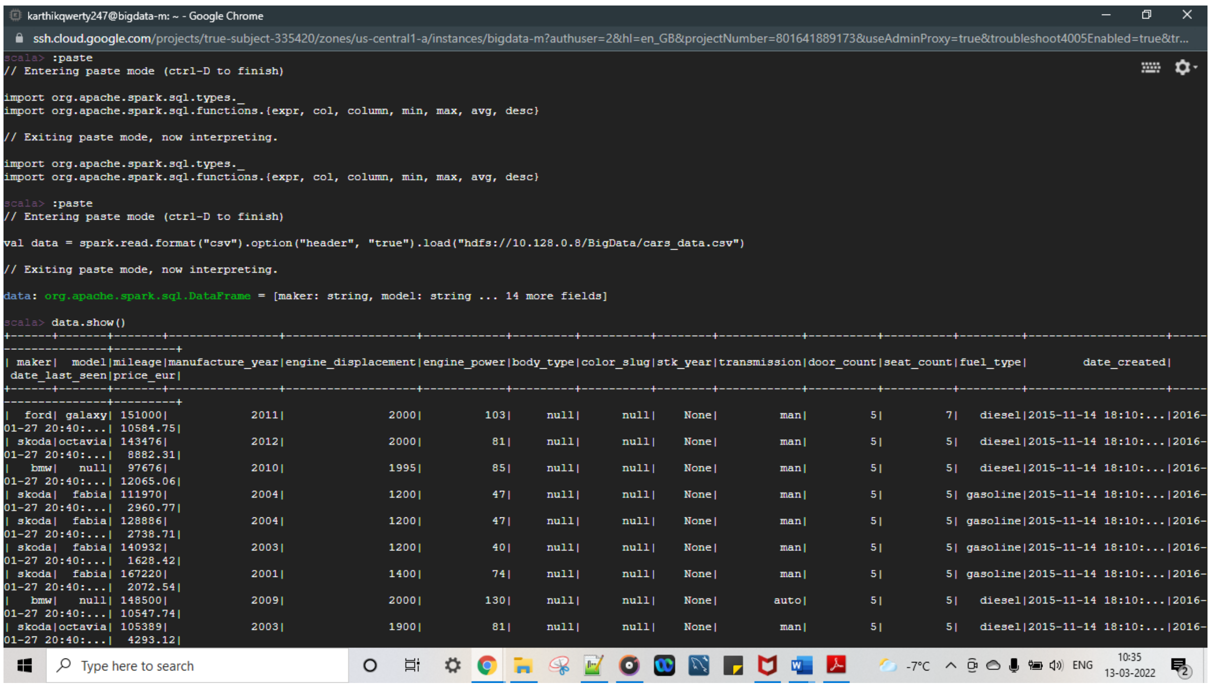
4) Write Spark DataFrames queries to see how many missing values you have in each attribute? Based on the results, document how many missing values in each column we have. Especially, mention those columns with more than 50% missing values.

The columns Colour_Slug and Fuel_Type are having more than 50% of the missing values and those columns have been dropped from the data frame as a part of data cleaning process.

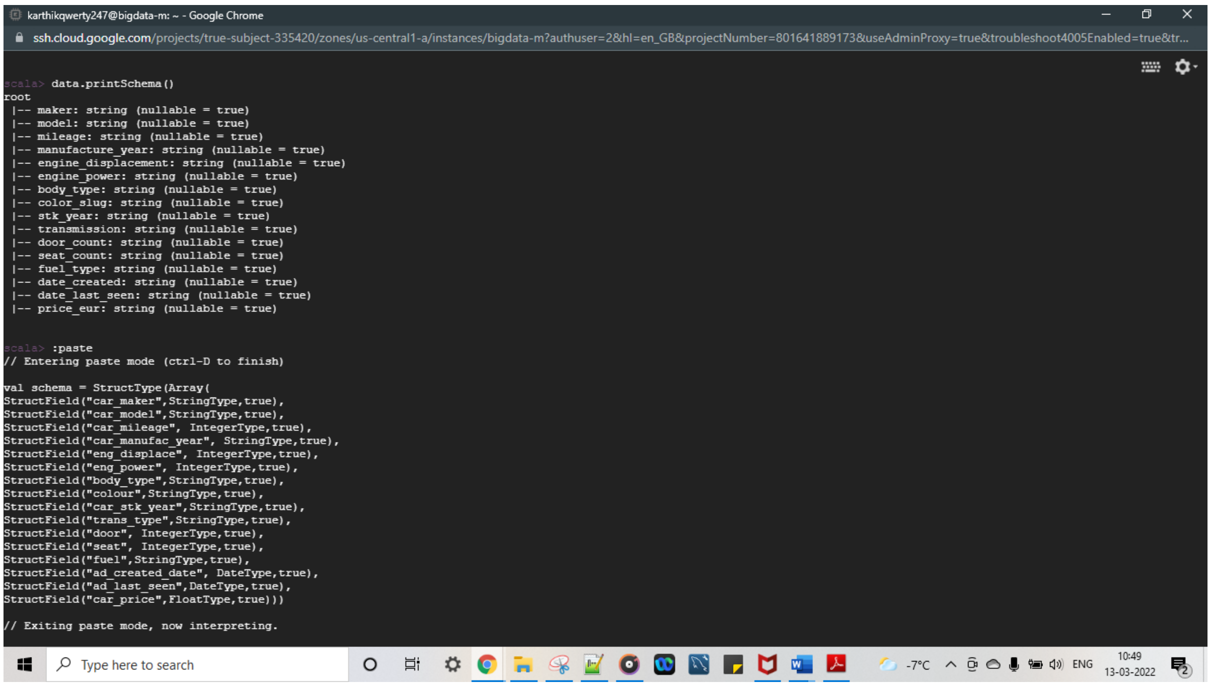
5) Group the price column and count the number of unique prices. Do you notice if there is a single price that is repeating across the ads?

There were many prices which we repeated, as this is a web scraped data, this might be due to the same ad being extracted from multiple sources. The prices which were repeating frequently were removed.

Spark Screenshots


karthikqwerty247@bigdata-m: ~ - Google Chrome
ssh.cloud.google.com/projects/true-subject-335420/zones/us-central1-a/instances/bigdata-m?authuser=2&hl=en_GB&projectNumber=801641889173&useAdminProxy=true&troubleshoot4005Enabled=true&tr...
Type here to search
-7°C
ENG
10:35
13-03-2022

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

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

// Exiting paste mode, now interpreting.

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

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

val data = spark.read.format("csv").option("header", "true").load("hdfs://10.128.0.8/BigData/cars_data.csv")

// Exiting paste mode, now interpreting.

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

scala> data.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|
 date_last_seen|price_eur|
+------+-------+-------+----------------+-------------------+------------+---------+----------+--------+------------+----------+----------+---------+--------------------+-----
---------------+---------+
|  ford| galaxy| 151000|            2011|               2000|         103|     null|      null|    None|         man|         5|         7|   diesel|2015-11-14 18:10:...|2016-
01-27 20:40:...| 10584.75|
| skoda|octavia| 143476|            2012|               2000|          81|     null|      null|    None|         man|         5|         5|   diesel|2015-11-14 18:10:...|2016-
01-27 20:40:...|  8882.31|
|   bmw|   null|  97676|            2010|               1995|          85|     null|      null|    None|         man|         5|         5|   diesel|2015-11-14 18:10:...|2016-
01-27 20:40:...| 12065.06|
| skoda|  fabia| 111970|            2004|               1200|          47|     null|      null|    None|         man|         5|         5| gasoline|2015-11-14 18:10:...|2016-
01-27 20:40:...|  2960.77|
| skoda|  fabia| 128886|            2004|               1200|          47|     null|      null|    None|         man|         5|         5| gasoline|2015-11-14 18:10:...|2016-
01-27 20:40:...|  2738.71|
| skoda|  fabia| 140932|            2003|               1200|          40|     null|      null|    None|         man|         5|         5| gasoline|2015-11-14 18:10:...|2016-
01-27 20:40:...|  1628.42|
| skoda|  fabia| 167220|            2001|               1400|          74|     null|      null|    None|         man|         5|         5| gasoline|2015-11-14 18:10:...|2016-
01-27 20:40:...|  2072.54|
|   bmw|   null| 148500|            2009|               2000|         130|     null|      null|    None|        auto|         5|         5|   diesel|2015-11-14 18:10:...|2016-
01-27 20:40:...| 10547.74|
| skoda|octavia| 105389|            2003|               1900|          81|     null|      null|    None|         man|         5|         5|   diesel|2015-11-14 18:10:...|2016-
01-27 20:40:...|  4293.12|
```



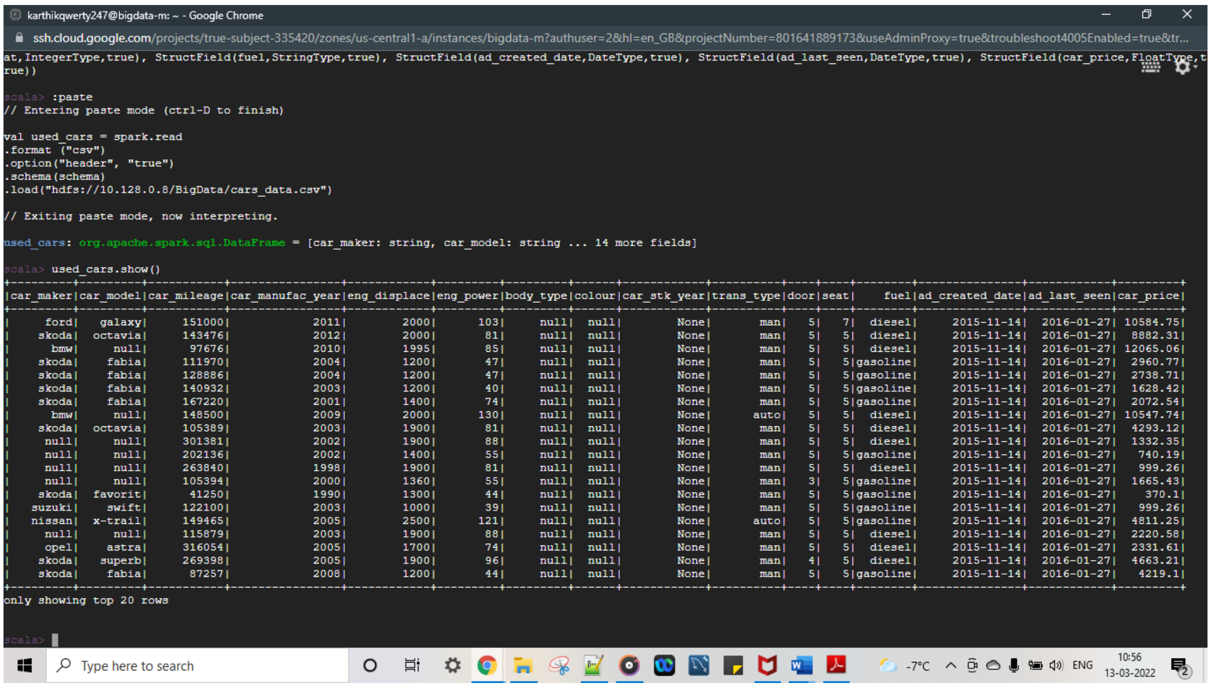
karthikqwerty247@bigdata-m: ~ - Google Chrome

ssh.cloud.google.com/projects/true-subject-335420/zones/us-central1-a/instances/bigdata-m?authuser=2&hl=en_GB&projectNumber=801641889173&useAdminProxy=true&troubleshoot4005Enabled=true&tr...

```
scala> data.printSchema()
root
 |-- maker: string (nullable = true)
 |-- model: string (nullable = true)
 |-- mileage: string (nullable = true)
 |-- manufacture_year: string (nullable = true)
 |-- engine_displacement: string (nullable = true)
 |-- engine_power: string (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: string (nullable = true)


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

val schema = StructType(Array(
StructField("car_maker",StringType,true),
StructField("car_model",StringType,true),
StructField("car_mileage", IntegerType,true),
StructField("car_manufac_year", StringType,true),
StructField("eng_displace", IntegerType,true),
StructField("eng_power", IntegerType,true),
StructField("body_type",StringType,true),
StructField("colour",StringType,true),
StructField("car_stk_year",StringType,true),
StructField("trans_type",StringType,true),
StructField("door", IntegerType,true),
StructField("seat", IntegerType,true),
StructField("fuel",StringType,true),
StructField("ad_created_date", DateType,true),
StructField("ad_last_seen",DateType,true),
StructField("car_price",FloatType,true)))

// Exiting paste mode, now interpreting.
```

Type here to search

-7°C ENG 10:49 13-03-2022



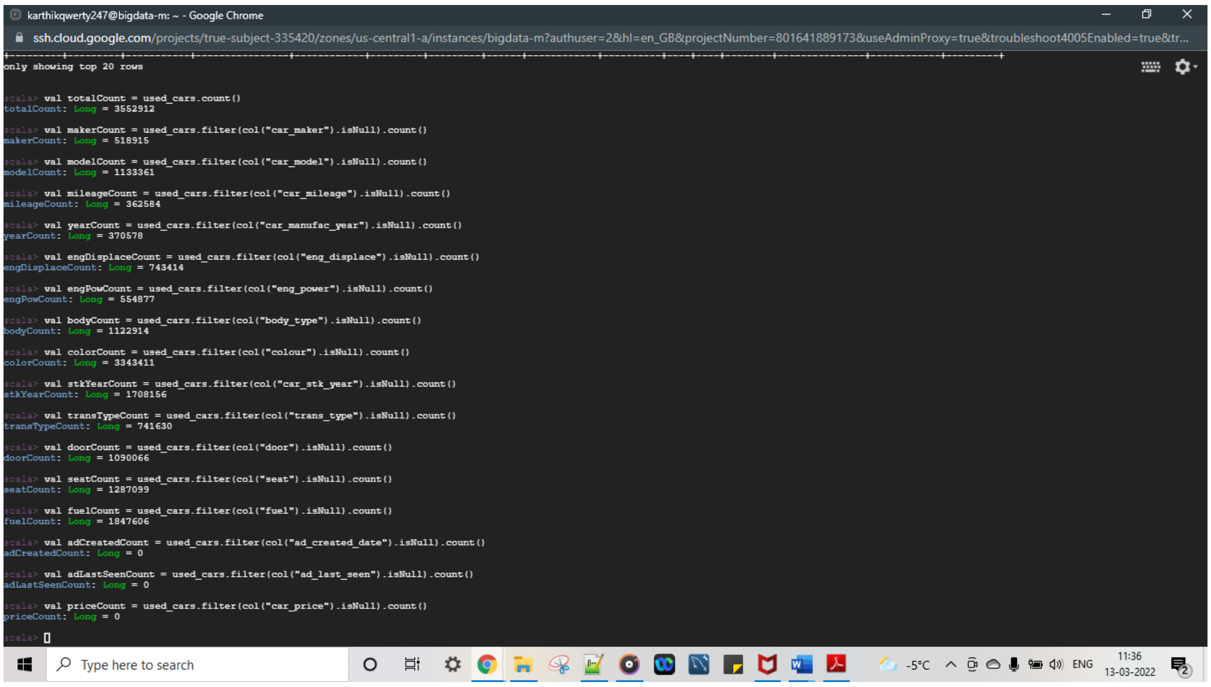
karthikqwerty247@bigdata-m: ~ - Google Chrome

ssh.cloud.google.com/projects/true-subject-335420/zones/us-central1-a/instances/bigdata-m?authuser=2&hl=en_GB&projectNumber=801641889173&useAdminProxy=true&troubleshoot4005Enabled=true&tr...

```
at,IntegerType,true), StructField(fuel,StringType,true), StructField(ad_created_date,DateType,true), StructField(ad_last_seen,DateType,true), StructField(car_price,FloatType,t
rue))

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

val used_cars = spark.read
.format ("csv")
.option("header", "true")
.schema(schema)
.load("hdfs://10.128.0.8/BigData/cars_data.csv")

// Exiting paste mode, now interpreting.

used_cars: org.apache.spark.sql.DataFrame = [car_maker: string, car_model: string ... 14 more fields]

scala> used_cars.show()
+---------+---------+-----------+----------------+------------+---------+---------+------+------------+----------+----+----+--------+---------------+------------+---------+
|car_maker|car_model|car_mileage|car_manufac_year|eng_displace|eng_power|body_type|colour|car_stk_year|trans_type|door|seat|    fuel|ad_created_date|ad_last_seen|car_price|
+---------+---------+-----------+----------------+------------+---------+---------+------+------------+----------+----+----+--------+---------------+------------+---------+
|     ford|   galaxy|     151000|            2011|        2000|      103|     null|  null|        None|       man|   5|   7|  diesel|     2015-11-14|  2016-01-27| 10584.75|
|    skoda|  octavia|     143476|            2012|        2000|       81|     null|  null|        None|       man|   5|   5|  diesel|     2015-11-14|  2016-01-27|  8882.31|
|      bmw|     null|      97676|            2010|        1995|       85|     null|  null|        None|       man|   5|   5|  diesel|     2015-11-14|  2016-01-27| 12065.06|
|    skoda|    fabia|     111970|            2004|        1200|       47|     null|  null|        None|       man|   5|   5|gasoline|     2015-11-14|  2016-01-27|  2960.77|
|    skoda|    fabia|     128886|            2004|        1200|       47|     null|  null|        None|       man|   5|   5|gasoline|     2015-11-14|  2016-01-27|  2738.71|
|    skoda|    fabia|     140932|            2003|        1200|       40|     null|  null|        None|       man|   5|   5|gasoline|     2015-11-14|  2016-01-27|  1628.42|
|    skoda|    fabia|     167220|            2001|        1400|       74|     null|  null|        None|       man|   5|   5|gasoline|     2015-11-14|  2016-01-27|  2072.54|
|      bmw|     null|     148500|            2009|        2000|      130|     null|  null|        None|      auto|   5|   5|  diesel|     2015-11-14|  2016-01-27| 10547.74|
|    skoda|  octavia|     105389|            2003|        1900|       81|     null|  null|        None|       man|   5|   5|  diesel|     2015-11-14|  2016-01-27|  4293.12|
|     null|     null|     301381|            2002|        1900|       88|     null|  null|        None|       man|   5|   5|  diesel|     2015-11-14|  2016-01-27|  1332.35|
|     null|     null|     202136|            2002|        1400|       55|     null|  null|        None|       man|   5|   5|gasoline|     2015-11-14|  2016-01-27|   740.19|
|     null|     null|     263840|            1998|        1900|       81|     null|  null|        None|       man|   5|   5|  diesel|     2015-11-14|  2016-01-27|   999.26|
|     null|     null|     105394|            2000|        1360|       55|     null|  null|        None|       man|   3|   5|gasoline|     2015-11-14|  2016-01-27|  1665.43|
|    skoda|  favorit|      41250|            1990|        1300|       44|     null|  null|        None|       man|   5|   5|gasoline|     2015-11-14|  2016-01-27|    370.1|
|   suzuki|    swift|     122100|            2003|        1000|       39|     null|  null|        None|       man|   5|   5|gasoline|     2015-11-14|  2016-01-27|   999.26|
|   nissan|  x-trail|     149465|            2005|        2500|      121|     null|  null|        None|      auto|   5|   5|gasoline|     2015-11-14|  2016-01-27|  4811.25|
|     null|     null|     115879|            2003|        1900|       88|     null|  null|        None|       man|   5|   5|  diesel|     2015-11-14|  2016-01-27|  2220.58|
|     opel|    astra|     316054|            2005|        1700|       74|     null|  null|        None|       man|   5|   5|  diesel|     2015-11-14|  2016-01-27|  2331.61|
|    skoda|   superb|     269398|            2005|        1900|       96|     null|  null|        None|       man|   4|   5|  diesel|     2015-11-14|  2016-01-27|  4663.21|
|    skoda|    fabia|      87257|            2008|        1200|       44|     null|  null|        None|       man|   5|   5|gasoline|     2015-11-14|  2016-01-27|   4219.1|
+---------+---------+-----------+----------------+------------+---------+---------+------+------------+----------+----+----+--------+---------------+------------+---------+
only showing top 20 rows


scala> 
```

Type here to search

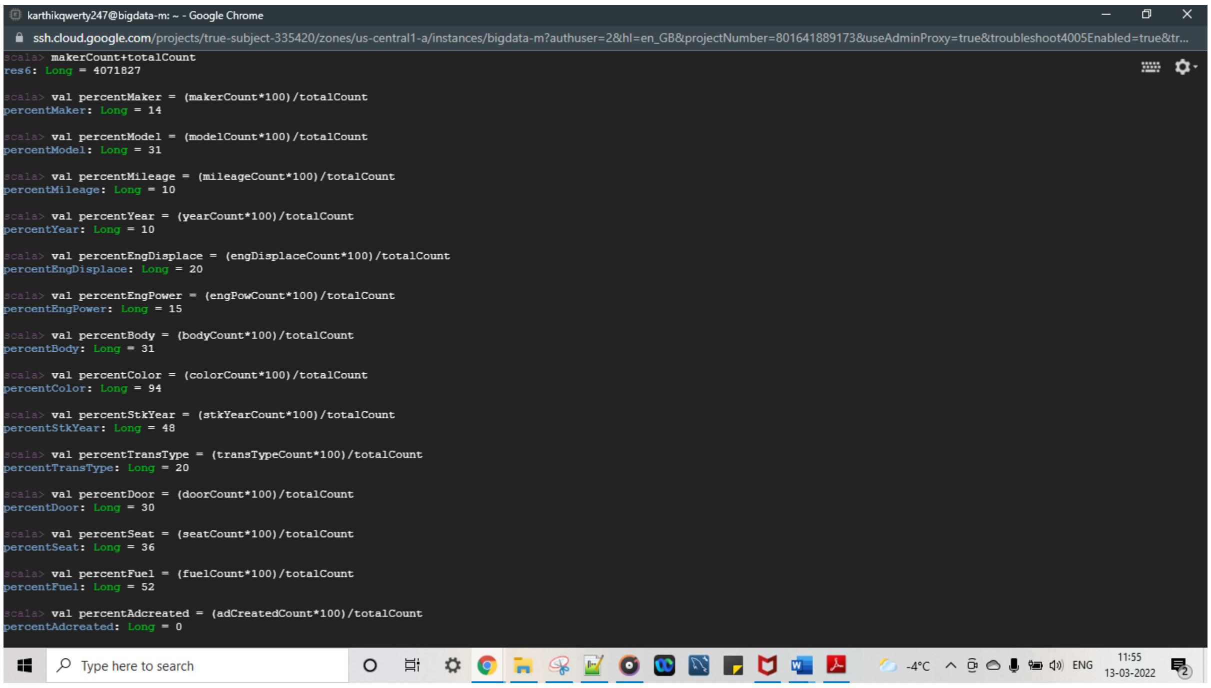
-7°C ENG 10:56 13-03-2022



karthikqwerty247@bigdata-m: ~ - Google Chrome

ssh.cloud.google.com/projects/true-subject-335420/zones/us-central1-a/instances/bigdata-m?authuser=2&hl=en_GB&projectNumber=801641889173&useAdminProxy=true&troubleshoot4005Enabled=true&tr...

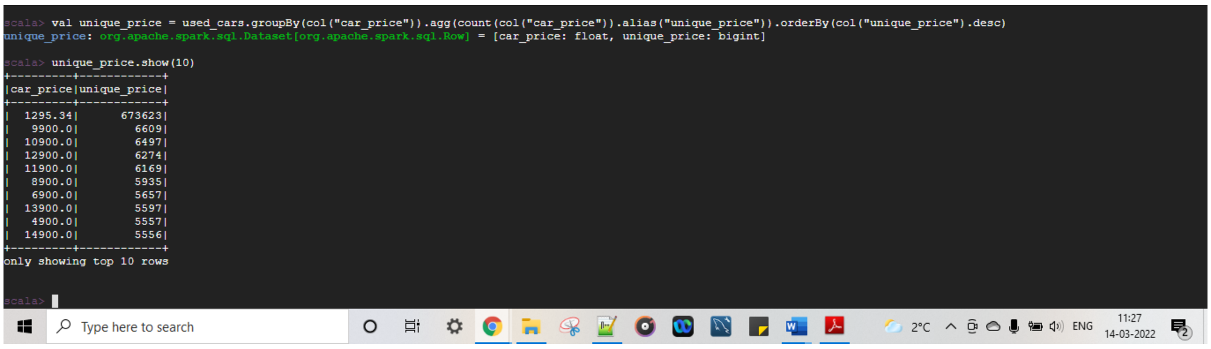
```
only showing top 20 rows

scala> val totalCount = used_cars.count()
totalCount: Long = 3552912

scala> val makerCount = used_cars.filter(col("car_maker").isNull).count()
makerCount: Long = 518915

scala> val modelCount = used_cars.filter(col("car_model").isNull).count()
modelCount: Long = 1133361

scala> val mileageCount = used_cars.filter(col("car_mileage").isNull).count()
mileageCount: Long = 362584

scala> val yearCount = used_cars.filter(col("car_manufac_year").isNull).count()
yearCount: Long = 370578

scala> val engDisplaceCount = used_cars.filter(col("eng_displace").isNull).count()
engDisplaceCount: Long = 743414

scala> val engPowCount = used_cars.filter(col("eng_power").isNull).count()
engPowCount: Long = 554877

scala> val bodyCount = used_cars.filter(col("body_type").isNull).count()
bodyCount: Long = 1122914

scala> val colorCount = used_cars.filter(col("colour").isNull).count()
colorCount: Long = 3343411

scala> val stkYearCount = used_cars.filter(col("car_stk_year").isNull).count()
stkYearCount: Long = 1708156

scala> val transTypeCount = used_cars.filter(col("trans_type").isNull).count()
transTypeCount: Long = 741630

scala> val doorCount = used_cars.filter(col("door").isNull).count()
doorCount: Long = 1090066

scala> val seatCount = used_cars.filter(col("seat").isNull).count()
seatCount: Long = 1287099

scala> val fuelCount = used_cars.filter(col("fuel").isNull).count()
fuelCount: Long = 1847606

scala> val adCreatedCount = used_cars.filter(col("ad_created_date").isNull).count()
adCreatedCount: Long = 0

scala> val adLastSeenCount = used_cars.filter(col("ad_last_seen").isNull).count()
adLastSeenCount: Long = 0

scala> val priceCount = used_cars.filter(col("car_price").isNull).count()
priceCount: Long = 0

scala> 
```



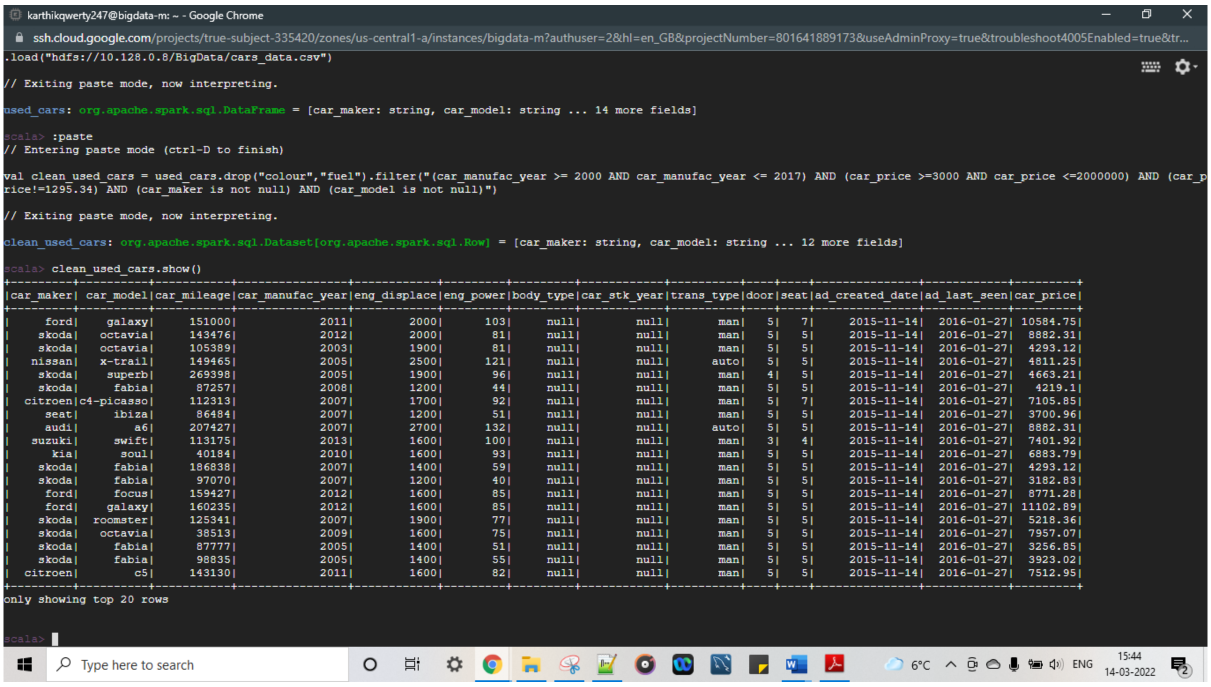
karthikqwerty247@bigdata-m: ~ - Google Chrome

ssh.cloud.google.com/projects/true-subject-335420/zones/us-central1-a/instances/bigdata-m?authuser=2&hl=en_GB&projectNumber=801641889173&useAdminProxy=true&troubleshoot4005Enabled=true&tr...

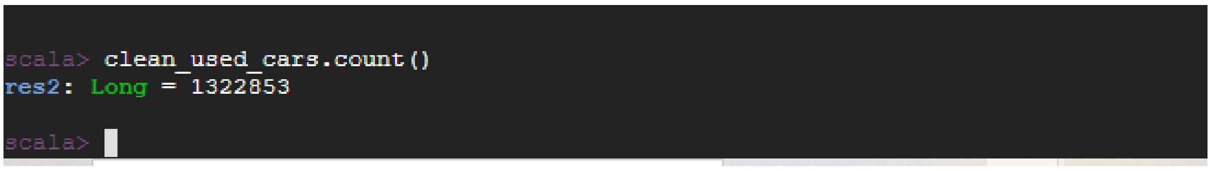
```
scala> makerCount+totalCount
res6: Long = 4071827

scala> val percentMaker = (makerCount*100)/totalCount
percentMaker: Long = 14

scala> val percentModel = (modelCount*100)/totalCount
percentModel: Long = 31

scala> val percentMileage = (mileageCount*100)/totalCount
percentMileage: Long = 10

scala> val percentYear = (yearCount*100)/totalCount
percentYear: Long = 10

scala> val percentEngDisplace = (engDisplaceCount*100)/totalCount
percentEngDisplace: Long = 20

scala> val percentEngPower = (engPowCount*100)/totalCount
percentEngPower: Long = 15

scala> val percentBody = (bodyCount*100)/totalCount
percentBody: Long = 31

scala> val percentColor = (colorCount*100)/totalCount
percentColor: Long = 94

scala> val percentStkYear = (stkYearCount*100)/totalCount
percentStkYear: Long = 48

scala> val percentTransType = (transTypeCount*100)/totalCount
percentTransType: Long = 20

scala> val percentDoor = (doorCount*100)/totalCount
percentDoor: Long = 30

scala> val percentSeat = (seatCount*100)/totalCount
percentSeat: Long = 36

scala> val percentFuel = (fuelCount*100)/totalCount
percentFuel: Long = 52

scala> val percentAdcreated = (adCreatedCount*100)/totalCount
percentAdcreated: Long = 0
```



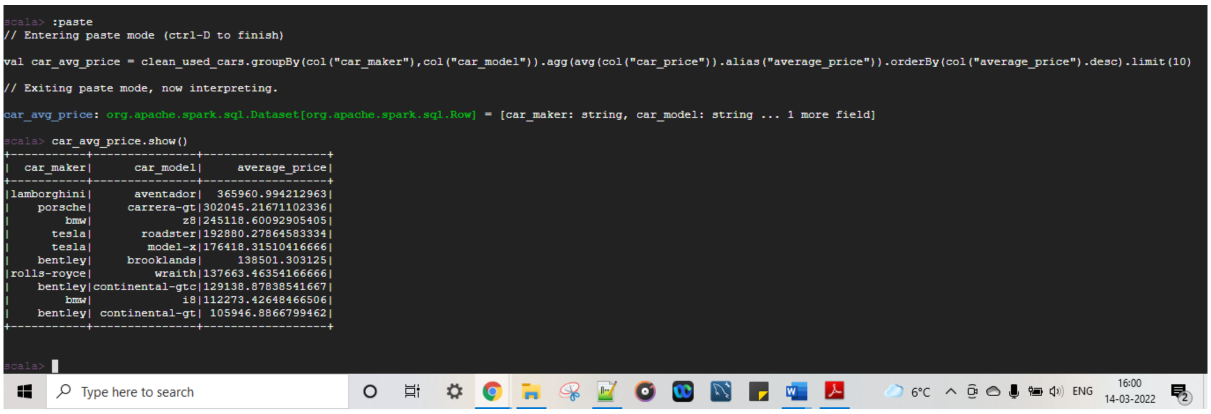
```
scala> val unique_price = used_cars.groupBy(col("car_price")).agg(count(col("car_price")).alias("unique_price")).orderBy(col("unique_price").desc)
unique_price: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [car_price: float, unique_price: bigint]

scala> unique_price.show(10)
+---------+------------+
|car_price|unique_price|
+---------+------------+
|  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|
+---------+------------+
only showing top 10 rows


scala> 
```


karthikqwerty247@bigdata-m: ~ - Google Chrome
ssh.cloud.google.com/projects/true-subject-335420/zones/us-central1-a/instances/bigdata-m?authuser=2&hl=en_GB&projectNumber=801641889173&useAdminProxy=true&troubleshoot4005Enabled=true&tr...
Type here to search
6°C
ENG
15:44
14-03-2022

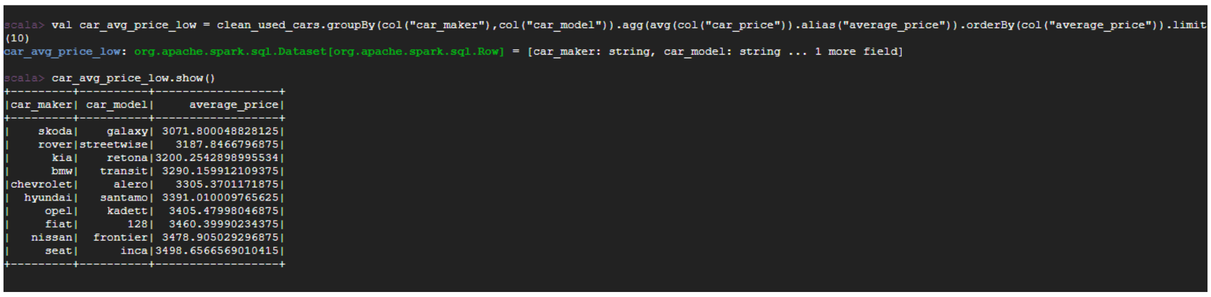
```
.load("hdfs://10.128.0.8/BigData/cars_data.csv")

// Exiting paste mode, now interpreting.

used_cars: org.apache.spark.sql.DataFrame = [car_maker: string, car_model: string ... 14 more fields]

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

val clean_used_cars = used_cars.drop("colour","fuel").filter("(car_manufac_year >= 2000 AND car_manufac_year <= 2017) AND (car_price >=3000 AND car_price <=2000000) AND (car_p
rice!=1295.34) AND (car_maker is not null) AND (car_model is not null)")

// Exiting paste mode, now interpreting.

clean_used_cars: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [car_maker: string, car_model: string ... 12 more fields]

scala> clean_used_cars.show()
+---------+----------+-----------+----------------+------------+---------+---------+------------+----------+----+----+---------------+------------+---------+
|car_maker| car_model|car_mileage|car_manufac_year|eng_displace|eng_power|body_type|car_stk_year|trans_type|door|seat|ad_created_date|ad_last_seen|car_price|
+---------+----------+-----------+----------------+------------+---------+---------+------------+----------+----+----+---------------+------------+---------+
|     ford|    galaxy|     151000|            2011|        2000|      103|     null|        null|       man|   5|   7|     2015-11-14|  2016-01-27| 10584.75|
|    skoda|   octavia|     143476|            2012|        2000|       81|     null|        null|       man|   5|   5|     2015-11-14|  2016-01-27|  8882.31|
|    skoda|   octavia|     105389|            2003|        1900|       81|     null|        null|       man|   5|   5|     2015-11-14|  2016-01-27|  4293.12|
|   nissan|   x-trail|     149465|            2005|        2500|      121|     null|        null|      auto|   5|   5|     2015-11-14|  2016-01-27|  4811.25|
|    skoda|    superb|     269398|            2005|        1900|       96|     null|        null|       man|   4|   5|     2015-11-14|  2016-01-27|  4663.21|
|    skoda|     fabia|      87257|            2008|        1200|       44|     null|        null|       man|   5|   5|     2015-11-14|  2016-01-27|   4219.1|
|  citroen|c4-picasso|     112313|            2007|        1700|       92|     null|        null|       man|   5|   7|     2015-11-14|  2016-01-27|  7105.85|
|     seat|     ibiza|      86484|            2007|        1200|       51|     null|        null|       man|   5|   5|     2015-11-14|  2016-01-27|  3700.96|
|     audi|        a6|     207427|            2007|        2700|      132|     null|        null|      auto|   5|   5|     2015-11-14|  2016-01-27|  8882.31|
|   suzuki|     swift|     113175|            2013|        1600|      100|     null|        null|       man|   3|   4|     2015-11-14|  2016-01-27|  7401.92|
|      kia|      soul|      40184|            2010|        1600|       93|     null|        null|       man|   5|   5|     2015-11-14|  2016-01-27|  6883.79|
|    skoda|     fabia|     186838|            2007|        1400|       59|     null|        null|       man|   5|   5|     2015-11-14|  2016-01-27|  4293.12|
|    skoda|     fabia|      97070|            2007|        1200|       40|     null|        null|       man|   5|   5|     2015-11-14|  2016-01-27|  3182.83|
|     ford|     focus|     159427|            2012|        1600|       85|     null|        null|       man|   5|   5|     2015-11-14|  2016-01-27|  8771.28|
|     ford|    galaxy|     160235|            2012|        1600|       85|     null|        null|       man|   5|   5|     2015-11-14|  2016-01-27| 11102.89|
|    skoda|  roomster|     125341|            2007|        1900|       77|     null|        null|       man|   5|   5|     2015-11-14|  2016-01-27|  5218.36|
|    skoda|   octavia|      38513|            2009|        1600|       75|     null|        null|       man|   5|   5|     2015-11-14|  2016-01-27|  7957.07|
|    skoda|     fabia|      87777|            2005|        1400|       51|     null|        null|       man|   5|   5|     2015-11-14|  2016-01-27|  3256.85|
|    skoda|     fabia|      98835|            2005|        1400|       55|     null|        null|       man|   5|   5|     2015-11-14|  2016-01-27|  3923.02|
|  citroen|        c5|     143130|            2011|        1600|       82|     null|        null|       man|   5|   5|     2015-11-14|  2016-01-27|  7512.95|
+---------+----------+-----------+----------------+------------+---------+---------+------------+----------+----+----+---------------+------------+---------+
only showing top 20 rows


scala> 
```



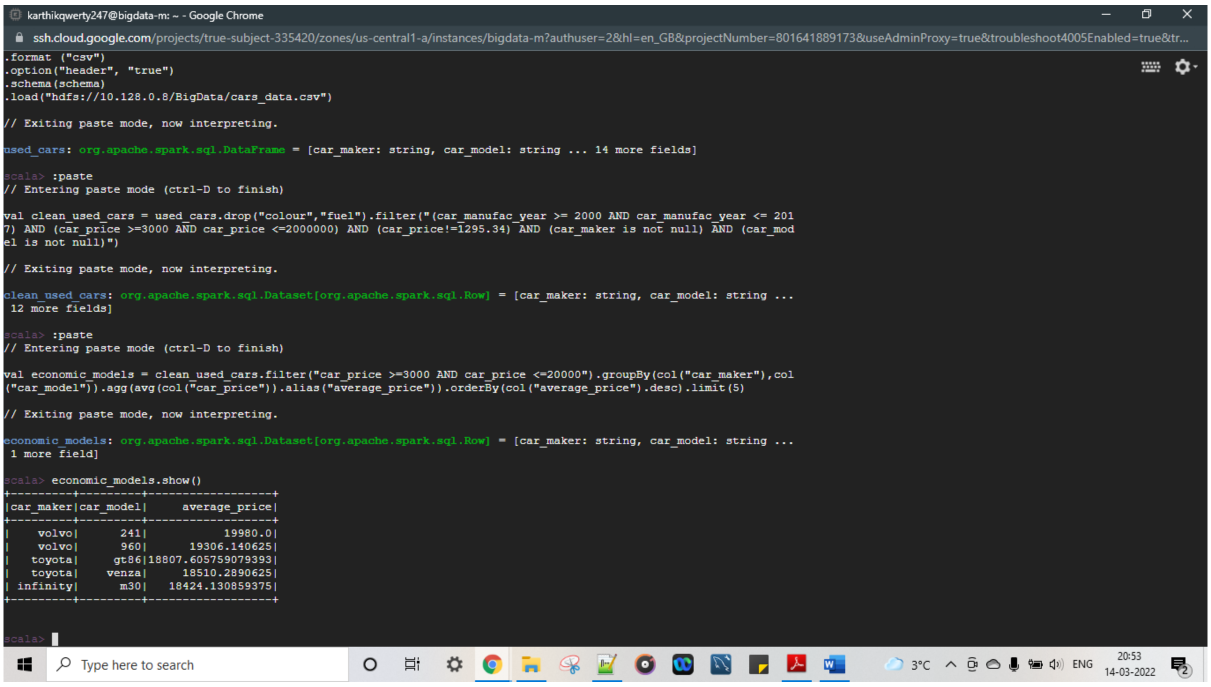
```
scala> clean_used_cars.count()
res2: Long = 1322853

scala> 
```



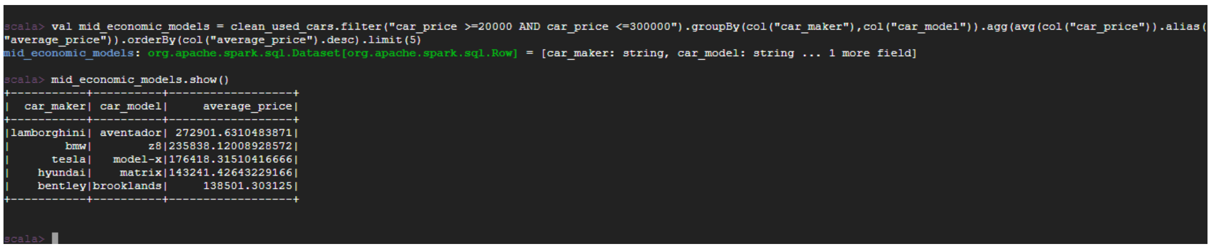
```
scala> :paste
// Entering paste mode (ctrl-D to finish)

val car_avg_price = clean_used_cars.groupBy(col("car_maker"),col("car_model")).agg(avg(col("car_price")).alias("average_price")).orderBy(col("average_price").desc).limit(10)

// Exiting paste mode, now interpreting.

car_avg_price: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [car_maker: string, car_model: string ... 1 more field]

scala> car_avg_price.show()
+-----------+---------------+------------------+
|  car_maker|      car_model|     average_price|
+-----------+---------------+------------------+
|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|
+-----------+---------------+------------------+


scala> 
```



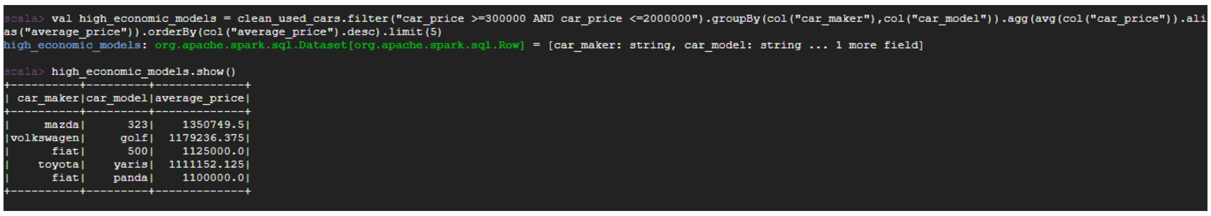
```
scala> val car_avg_price_low = clean_used_cars.groupBy(col("car_maker"),col("car_model")).agg(avg(col("car_price")).alias("average_price")).orderBy(col("average_price")).limit
(10)
car_avg_price_low: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [car_maker: string, car_model: string ... 1 more field]

scala> car_avg_price_low.show()
+---------+----------+------------------+
|car_maker| car_model|     average_price|
+---------+----------+------------------+
|    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|
+---------+----------+------------------+
```


karthikqwerty247@bigdata-m: ~ - Google Chrome
ssh.cloud.google.com/projects/true-subject-335420/zones/us-central1-a/instances/bigdata-m?authuser=2&hl=en_GB&projectNumber=801641889173&useAdminProxy=true&troubleshoot4005Enabled=true&tr...
Type here to search
3°C
ENG
20:53
14-03-2022

```
.format ("csv")
.option("header", "true")
.schema(schema)
.load("hdfs://10.128.0.8/BigData/cars_data.csv")

// Exiting paste mode, now interpreting.

used_cars: org.apache.spark.sql.DataFrame = [car_maker: string, car_model: string ... 14 more fields]

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

val clean_used_cars = used_cars.drop("colour","fuel").filter("(car_manufac_year >= 2000 AND car_manufac_year <= 201
7) AND (car_price >=3000 AND car_price <=2000000) AND (car_price!=1295.34) AND (car_maker is not null) AND (car_mod
el is not null)")

// Exiting paste mode, now interpreting.

clean_used_cars: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [car_maker: string, car_model: string ...
 12 more fields]

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

val economic_models = clean_used_cars.filter("car_price >=3000 AND car_price <=20000").groupBy(col("car_maker"),col
("car_model")).agg(avg(col("car_price")).alias("average_price")).orderBy(col("average_price").desc).limit(5)

// Exiting paste mode, now interpreting.

economic_models: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [car_maker: string, car_model: string ...
 1 more field]

scala> economic_models.show()
+---------+---------+-----------------+
|car_maker|car_model|    average_price|
+---------+---------+-----------------+
|    volvo|      241|          19980.0|
|    volvo|      960|     19306.140625|
|   toyota|     gt86|18807.605759079393|
|   toyota|    venza|    18510.2890625|
| infinity|      m30|  18424.130859375|
+---------+---------+-----------------+


scala> 
```



```
scala> val mid_economic_models = clean_used_cars.filter("car_price >=20000 AND car_price <=300000").groupBy(col("car_maker"),col("car_model")).agg(avg(col("car_price")).alias(
"average_price")).orderBy(col("average_price").desc).limit(5)
mid_economic_models: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [car_maker: string, car_model: string ... 1 more field]

scala> mid_economic_models.show()
+-----------+----------+-----------------+
|  car_maker| car_model|    average_price|
+-----------+----------+-----------------+
|lamborghini| aventador| 272901.6310483871|
|        bmw|        z8|235838.12008928572|
|      tesla|   model-x|176418.31510416666|
|    hyundai|    matrix|143241.42643229166|
|    bentley|brooklands|     138501.303125|
+-----------+----------+-----------------+


scala> 
```



```
scala> val high_economic_models = clean_used_cars.filter("car_price >=300000 AND car_price <=2000000").groupBy(col("car_maker"),col("car_model")).agg(avg(col("car_price")).ali
as("average_price")).orderBy(col("average_price").desc).limit(5)
high_economic_models: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [car_maker: string, car_model: string ... 1 more field]

scala> high_economic_models.show()
+----------+---------+-------------+
| car_maker|car_model|average_price|
+----------+---------+-------------+
|     mazda|      323|    1350749.5|
|volkswagen|     golf|  1179236.375|
|      fiat|      500|    1125000.0|
|    toyota|    yaris|  1111152.125|
|      fiat|    panda|    1100000.0|
+----------+---------+-------------+
```

Codes:

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

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

val data = spark.read.format("csv").option("header", "true").load("hdfs://10.128.0.8/BigData/cars_data.csv")

data.show()

```
data.printSchema()
```

(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 2.

```
val schema = StructType(Array(
StructField("car_maker",StringType,true),
StructField("car_model",StringType,true),
StructField("car_mileage", IntegerType,true),
StructField("car_manufac_year", IntegerType,true),
StructField("eng_displace", IntegerType,true),
StructField("eng_power", IntegerType,true),
StructField("body_type",StringType,true),
StructField("colour",StringType,true),
StructField("car_stk_year",IntegerType,true),
StructField("trans_type",StringType,true),
StructField("door", IntegerType,true),
StructField("seat", IntegerType,true),
StructField("fuel",StringType,true),
StructField("ad_created_date", DateType,true),
StructField("ad_last_seen",DateType,true),
StructField("car_price",FloatType,true)))
```

```
val used_cars = spark.read
.format ("csv")
.option("header", "true")
.schema(schema)
.load("hdfs://10.128.0.8/BigData/cars_data.csv")
```

(2) Look at the date column of the table used_cars. Why does the date column have all NULL values?

(4)

```
val totalCount = used_cars.count()
val makerCount = used_cars.filter(col("car_maker").isNull).count()
val modelCount = used_cars.filter(col("car_model").isNull).count()
val mileageCount = used_cars.filter(col("car_mileage").isNull).count()
val yearCount = used_cars.filter(col("car_manufac_year").isNull).count()
val engDisplaceCount = used_cars.filter(col("eng_displace").isNull).count()
val engPowCount = used_cars.filter(col("eng_power").isNull).count()
val bodyCount = used_cars.filter(col("body_type").isNull).count()
val colorCount = used_cars.filter(col("colour").isNull).count()
val stkYearCount = used_cars.filter(col("car_stk_year").isNull).count()
val transTypeCount = used_cars.filter(col("trans_type").isNull).count()
val doorCount = used_cars.filter(col("door").isNull).count()
val seatCount = used_cars.filter(col("seat").isNull).count()
val fuelCount = used_cars.filter(col("fuel").isNull).count()
val adCreatedCount = used_cars.filter(col("ad_created_date").isNull).count()
val adLastSeenCount = used_cars.filter(col("ad_last_seen").isNull).count()
val priceCount = used_cars.filter(col("car_price").isNull).count()

val percentMaker = (makerCount*100)/totalCount
val percentModel = (modelCount*100)/totalCount
val percentMileage = (mileageCount*100)/totalCount
val percentYear = (yearCount*100)/totalCount
val percentEngDisplace = (engDisplaceCount*100)/totalCount
val percentEngPower = (engPowCount*100)/totalCount
val percentBody = (bodyCount*100)/totalCount
val percentColor = (colorCount*100)/totalCount

val percentStkYear = (stkYearCount*100)/totalCount

val percentTransType = (transTypeCount*100)/totalCount

val percentDoor = (doorCount*100)/totalCount

val percentSeat = (seatCount*100)/totalCount

val percentFuel = (fuelCount*100)/totalCount

val percentAdcreated = (adCreatedCount*100)/totalCount

val percentAdLastSeen = (adLastSeenCount*100)/totalCount

val percentPrice = (priceCount*100)/totalCount
```

(5)

```
val unique_price =
used_cars.groupBy(col("car_price")).agg(count(col("car_price")).alias("unique_price")).orderBy(col("
unique_price").desc)
```

(6) val clean_used_cars = used_cars.drop("colour","fuel").filter("(car_manufac_year >= 2000 AND car_manufac_year <= 2017) AND (car_price >=3000 AND car_price <=2000000) AND (car_price!=1295.34) AND (car_maker is not null) AND (car_model is not null)")

(7) clean_used_cars.count()

(8) val car_avg_price = clean_used_cars.groupBy(col("car_maker"),col("car_model")).agg(avg(col("car_price")).alias("average_price")).orderBy(col("average_price").desc).limit(10)

(9) val car_avg_price_low = clean_used_cars.groupBy(col("car_maker"),col("car_model")).agg(avg(col("car_price")).alias("average_price")).orderBy(col("average_price")).limit(10)

(10) val economic_models = clean_used_cars.filter("car_price >=3000 AND car_price <=20000").groupBy(col("car_maker"),col("car_model")).agg(avg(col("car_price")).alias("average_price")).orderBy(col("average_price").desc).limit(5)

(11) val mid_economic_models = clean_used_cars.filter("car_price >=20000 AND car_price <=300000").groupBy(col("car_maker"),col("car_model")).agg(avg(col("car_price")).alias("average_price")).orderBy(col("average_price").desc).limit(5)

(12) val high_economic_models = clean_used_cars.filter("car_price >=300000 AND car_price <=2000000").groupBy(col("car_maker"),col("car_model")).agg(avg(col("car_price")).alias("average_price")).orderBy(col("average_price").desc).limit(5)

Extra_queries

1)val car_manufac_year_val= clean_used_cars.groupBy(col("car_manufac_year")).count()

2) val most_common_car= clean_used_cars.groupBy(col("car_maker"),col("car_model")).count().orderBy(col("count").desc).limit(5)

```
high_economic_models
.coalesce(1)
.write.option("header", true)
.csv("hdfs://10.128.0.8/BigData/high_economic_models.csv")
```

```
hadoop fs -copyToLocal /BigData/high_economic_models.csv /home/karthikqwerty247/BigData/Assign1
```