Assignment: Analysis with Spark Data Frames

Submitted by: Abhijeet Singh

Student id: 200489554

Course: BDAT 1008 – Data Collection and Curation

Instructor: Saber Amini

Contents

Introduction

The dataset gives the information about used cars for sale in Eastern European countries since 2015. This dataset contains information related to car makers, models,mileage,manufacture_year,engine_displacement (in ccm), engine_power (in kW), body_type, color_slug, stk_year (year of the last emission control), transmission (automatic or manual), door_count, seat_count, fuel_type (gasoline, diesel, cng, lpg, electric), date_created (when the ad was scraped), date_last-seen (when the ad was last seen), price_eur (list price converted to EUR). There are total 16 columns in the dataset and number of rows are around 3.5 million.

The source of this data set is the Kaggle. Here is link

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

Analysis questions

Q1) As per the average price for the economic segment customers which are the top maker and model ?

Q2) Which are the top five maker and model for the intermediate customers as per the average price

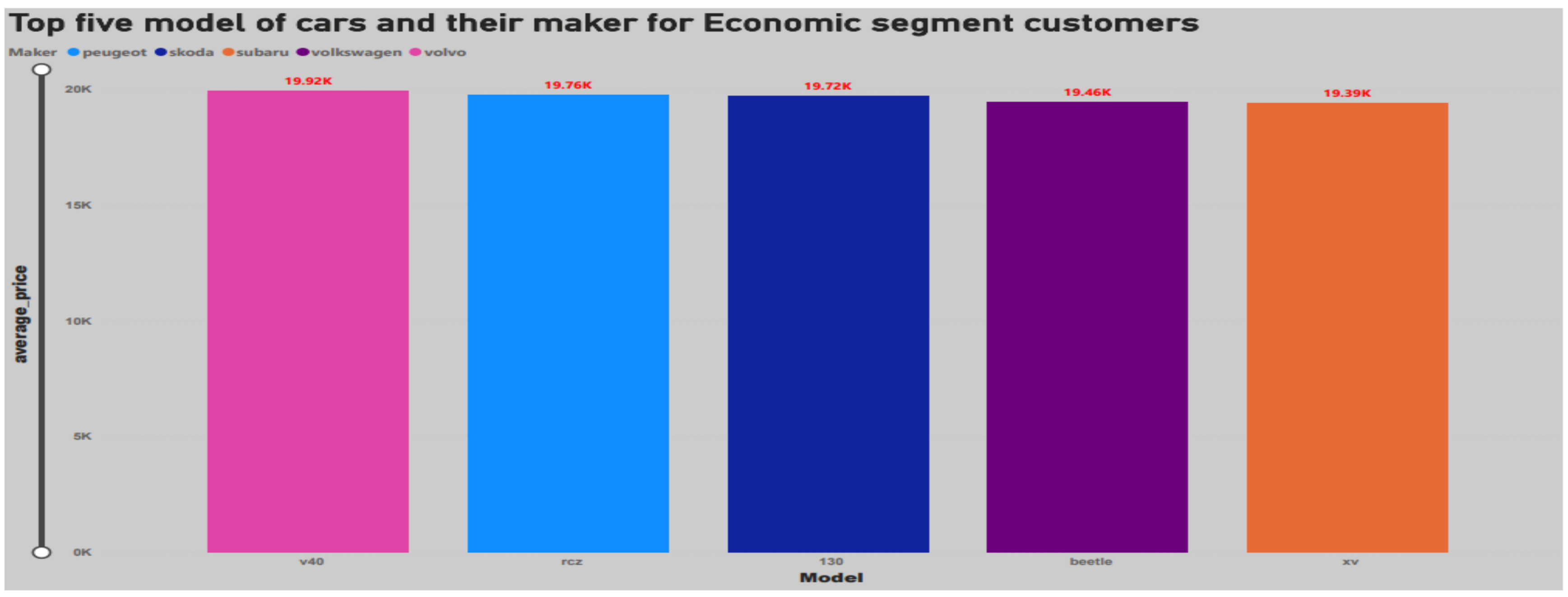
Q3) For the Luxury segment customers what would be the top maker and model based on the average price?

Data Cleaning and Analysis

- The raw data set had multiple columns with null values
- The price euro 8512 appears 956 times so it is removed from the dataset.
- The date time format was appropriately chosen so date column was shown correctly.
- A new table was created called clean used cars on which the further cleaning was done.
- Four columns such as stk year,Body type ,fuel type and colour slug consisted of more than 50% empty values as a result ,these columns were dropped.
- Only those cars were included whose manufacture year range between 2000 and 2017 and the price range of the cars is between 3000 and 2000000 Euro.
- Finally, approximately 1.4M rows were left out.
- Cars with the highest average price are Lamborghini and Porsche
- Cars with the lowest average price are bwn, skoda ,chevrolet, hyundai and rover

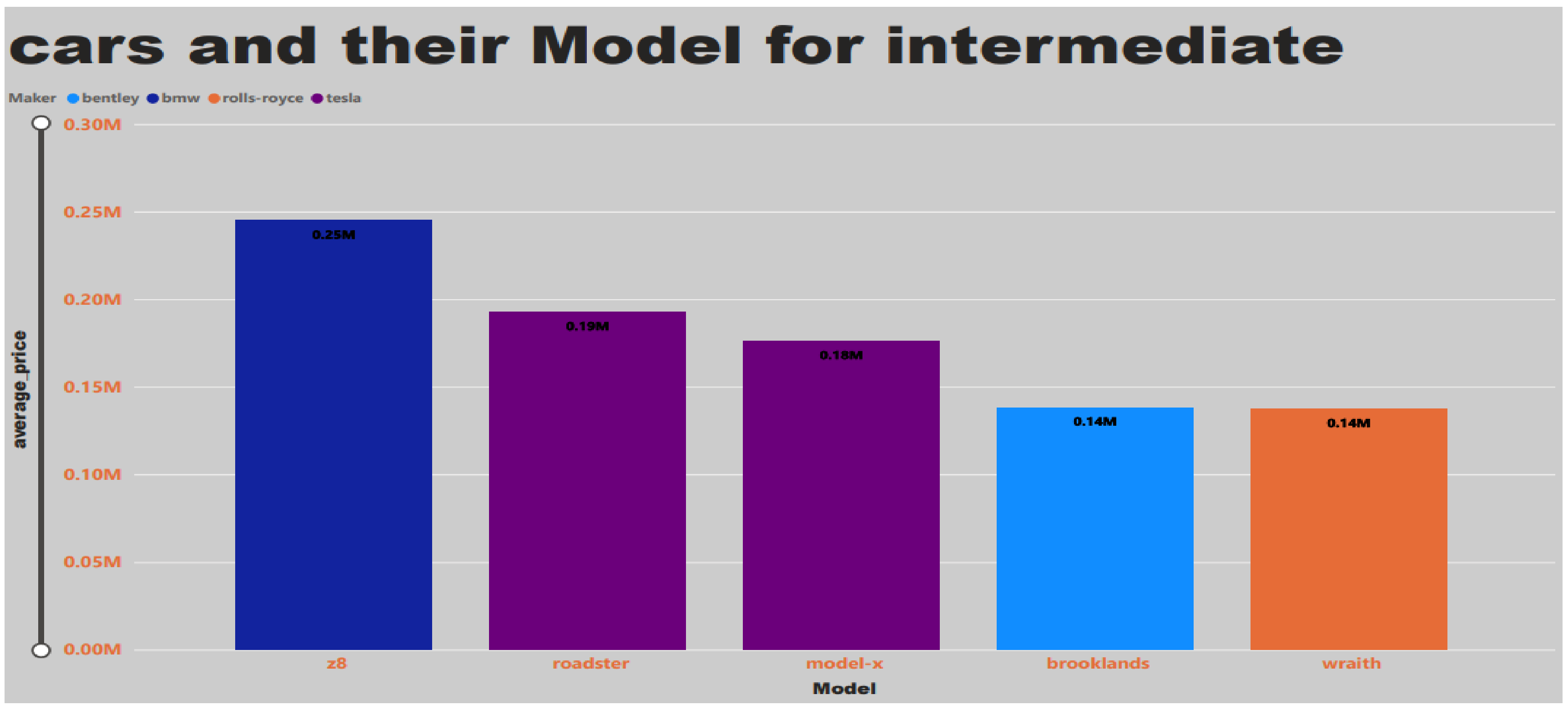
Data Analysis

✓ 1



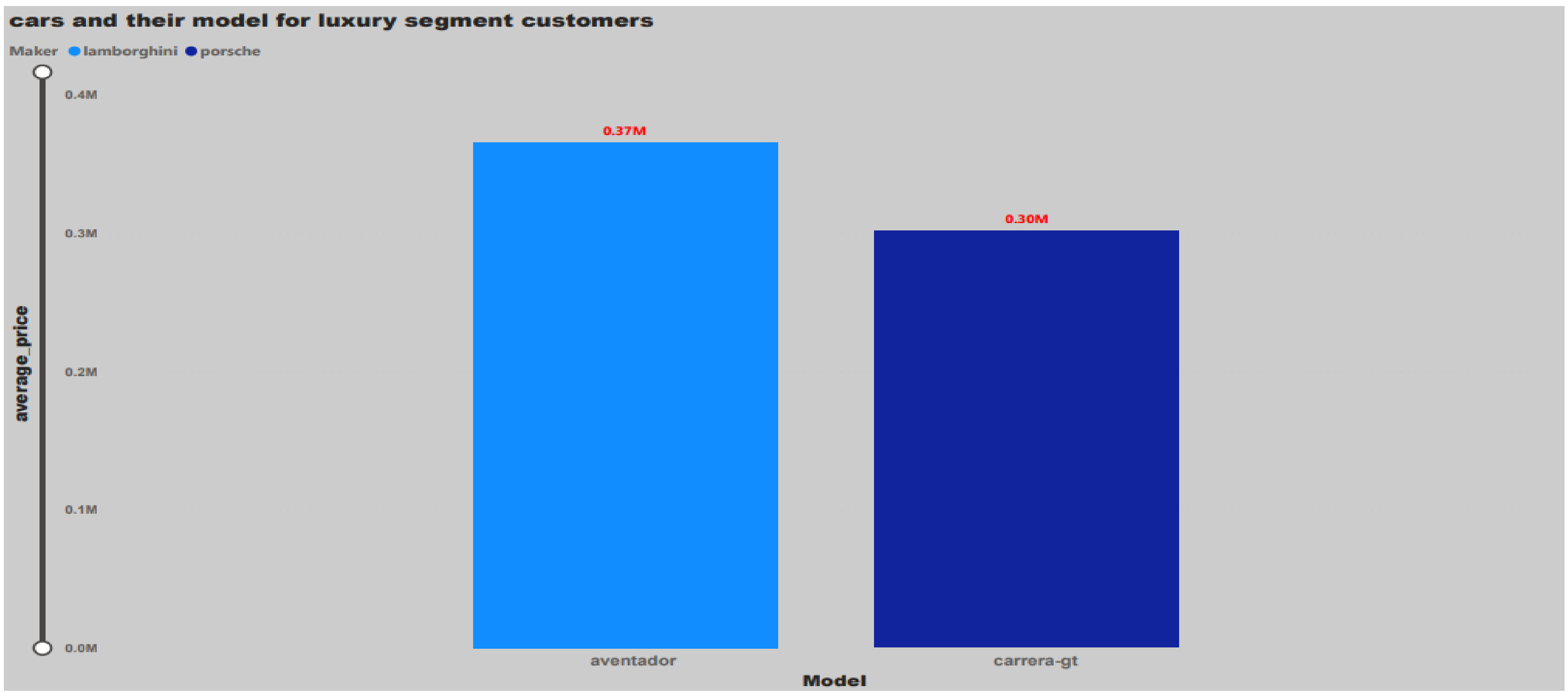
For the economic segment customers from the above graph, consideration can be made for these top five Models and Makers based on the average price. While comparing these top five models, it can be seen that there is slight difference between the top five models as per the average price. Investors can consider investing in volvo cars as it seems must popular among the economic segment customers. Apart from that, the average price of volvo car v40 lies within the range of economic segment customers so Volvo model v40 can be considered for the investment.

✓ 2



According to the above graph, these are the top five Models and Maker based on the average price that lies between 3000 to 20,000. Investment should be made on rolls-royce model wraith as this model has the least price range as compared to the other models which would be range friendly for the intermediate segment customers.

✓ 3

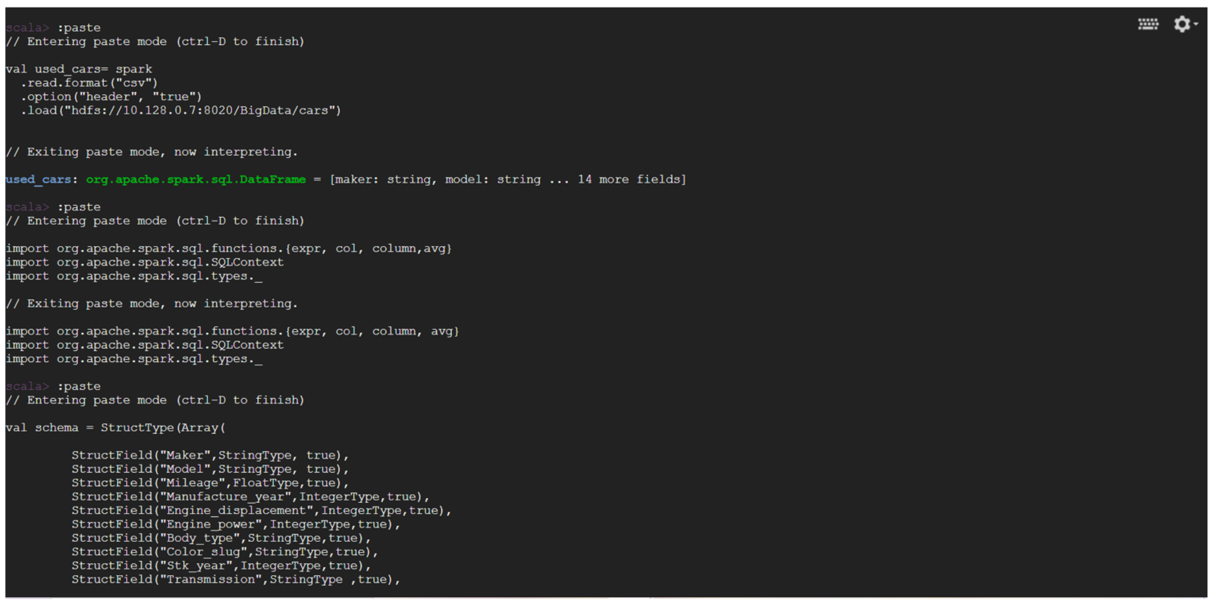


For the customers who belong to the luxury segment have two options with the average price range between 300000 to 2000000 which is Lamborghini and Porsche . In the luxury model customers can prefer to buy Porsche's Carrera-gt because average price of the car seems to be less than Lamborghini's aventador .

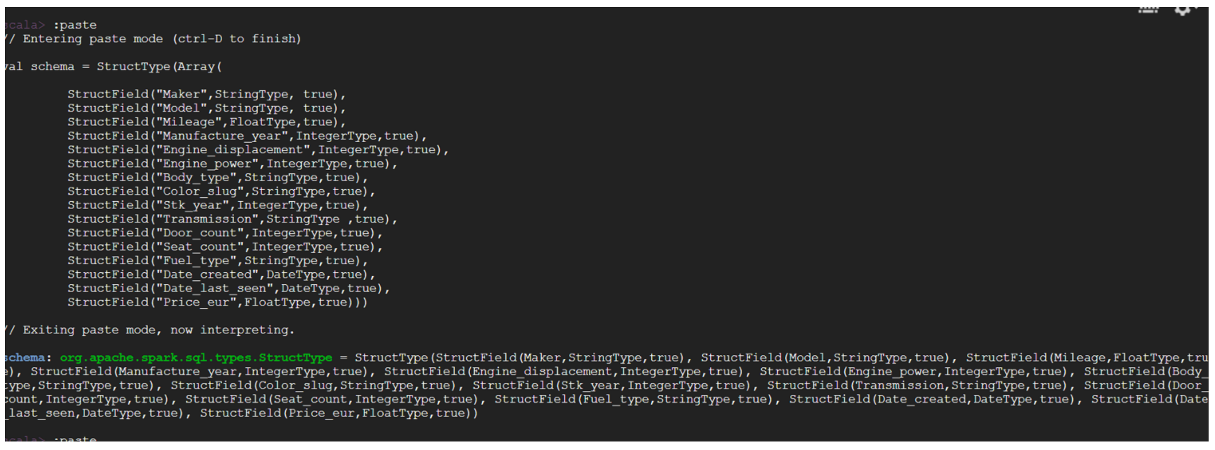
Conclusion

To conclude, based on the above analysis customers are being divided into three parts such as Economic, Intermediate and luxury segments . Investment should be done based on the Economic class as it has the large portion of the population which will lead to more sales of the cars which will result in the must profit earned from the economic class. In the economic segment, volvo car v40 can be considered for the investment because of its average price range which lies within the range because it will target the most customers.

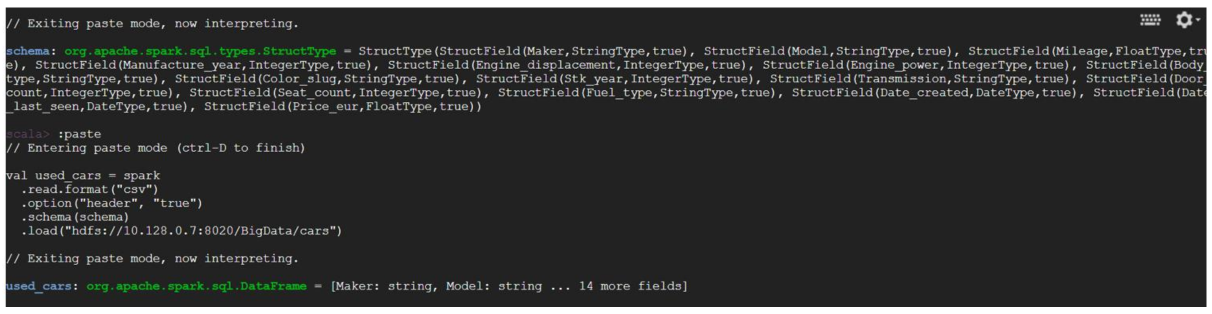
Appendix



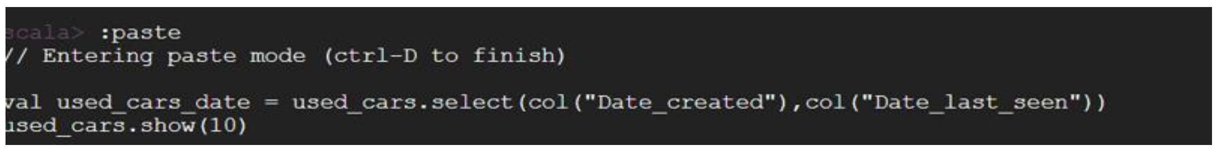
```
scala> :paste
// Entering paste mode (ctrl-D to finish)

val used_cars= spark
  .read.format("csv")
  .option("header", "true")
  .load("hdfs://10.128.0.7:8020/BigData/cars")


// Exiting paste mode, now interpreting.

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

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

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

// Exiting paste mode, now interpreting.

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

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

val schema = StructType(Array(

         StructField("Maker",StringType, true),
         StructField("Model",StringType, true),
         StructField("Mileage",FloatType,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),
```



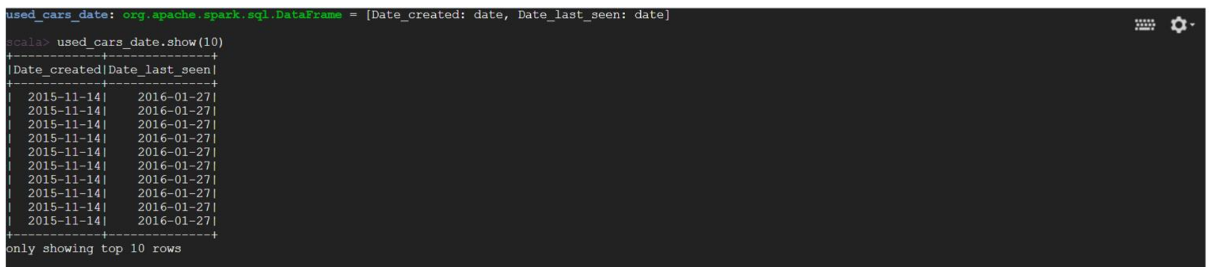
```
cala> :paste
/ Entering paste mode (ctrl-D to finish)

al schema = StructType(Array(

         StructField("Maker",StringType, true),
         StructField("Model",StringType, true),
         StructField("Mileage",FloatType,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("Date_last_seen",DateType,true),
         StructField("Price_eur",FloatType,true)))

/ Exiting paste mode, now interpreting.

chema: org.apache.spark.sql.types.StructType = StructType(StructField(Maker,StringType,true), StructField(Model,StringType,true), StructField(Mileage,FloatType,tru
e), StructField(Manufacture_year,IntegerType,true), StructField(Engine_displacement,IntegerType,true), StructField(Engine_power,IntegerType,true), StructField(Body_
ype,StringType,true), StructField(Color_slug,StringType,true), StructField(Stk_year,IntegerType,true), StructField(Transmission,StringType,true), StructField(Door_
ount,IntegerType,true), StructField(Seat_count,IntegerType,true), StructField(Fuel_type,StringType,true), StructField(Date_created,DateType,true), StructField(Date
_last_seen,DateType,true), StructField(Price_eur,FloatType,true))

cala> :paste
```



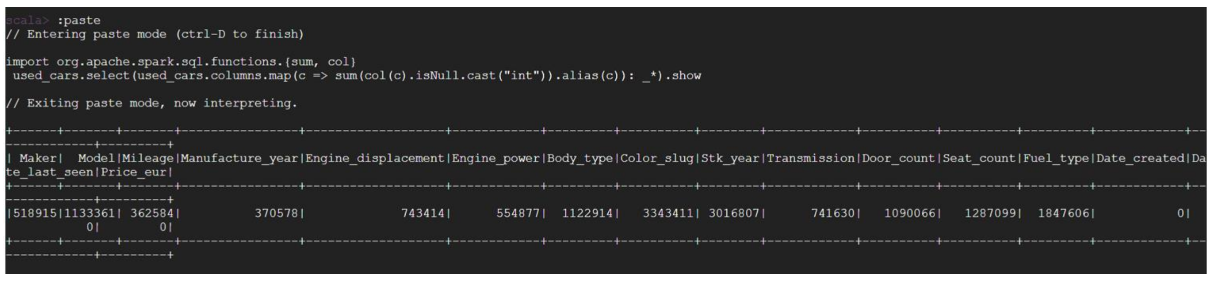
```
// Exiting paste mode, now interpreting.

schema: org.apache.spark.sql.types.StructType = StructType(StructField(Maker,StringType,true), StructField(Model,StringType,true), StructField(Mileage,FloatType,tru
e), 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(Date
_last_seen,DateType,true), StructField(Price_eur,FloatType,true))

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

val used_cars = spark
  .read.format("csv")
  .option("header", "true")
  .schema(schema)
  .load("hdfs://10.128.0.7:8020/BigData/cars")

// Exiting paste mode, now interpreting.

used_cars: org.apache.spark.sql.DataFrame = [Maker: string, Model: string ... 14 more fields]
```



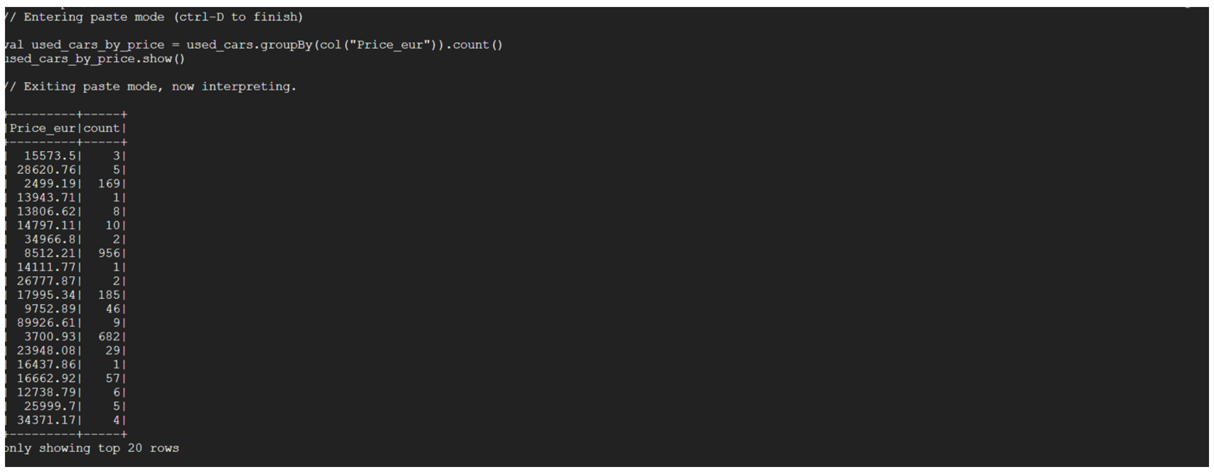
```
scala> :paste
// Entering paste mode (ctrl-D to finish)

val used_cars_date = used_cars.select(col("Date_created"),col("Date_last_seen"))
used_cars.show(10)
```



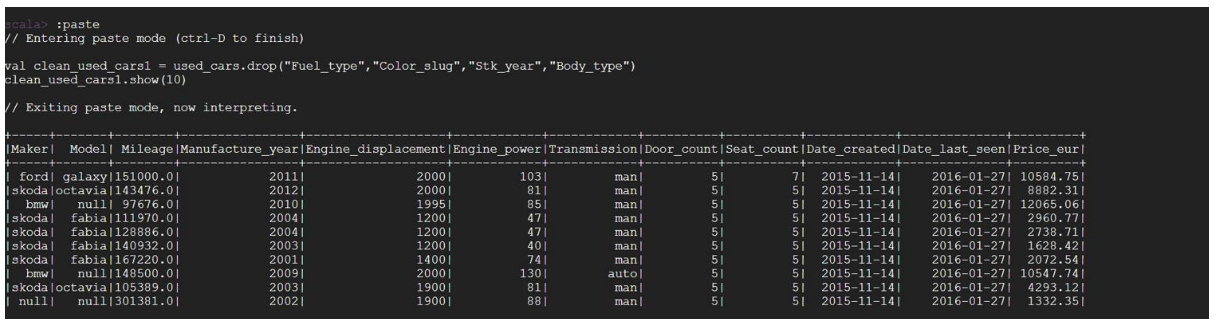
```
used_cars_date: org.apache.spark.sql.DataFrame = [Date_created: date, Date_last_seen: date]

scala> used_cars_date.show(10)
+------------+--------------+
|Date_created|Date_last_seen|
+------------+--------------+
|  2015-11-14|    2016-01-27|
|  2015-11-14|    2016-01-27|
|  2015-11-14|    2016-01-27|
|  2015-11-14|    2016-01-27|
|  2015-11-14|    2016-01-27|
|  2015-11-14|    2016-01-27|
|  2015-11-14|    2016-01-27|
|  2015-11-14|    2016-01-27|
|  2015-11-14|    2016-01-27|
|  2015-11-14|    2016-01-27|
+------------+--------------+
only showing top 10 rows
```



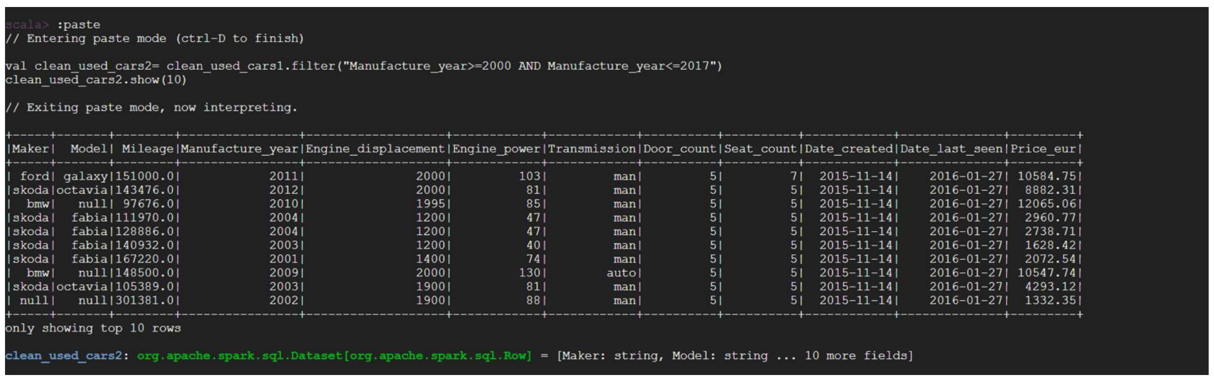
```
scala> :paste
// Entering paste mode (ctrl-D to finish)

import org.apache.spark.sql.functions.{sum, col}
 used_cars.select(used_cars.columns.map(c => sum(col(c).isNull.cast("int")).alias(c)): _*).show

// Exiting paste mode, now interpreting.

+------+-------+-------+----------------+-------------------+------------+---------+----------+--------+------------+----------+----------+---------+------------+--
------------+---------+
| Maker|  Model|Mileage|Manufacture_year|Engine_displacement|Engine_power|Body_type|Color_slug|Stk_year|Transmission|Door_count|Seat_count|Fuel_type|Date_created|Da
te_last_seen|Price_eur|
+------+-------+-------+----------------+-------------------+------------+---------+----------+--------+------------+----------+----------+---------+------------+--
------------+---------+
|518915|1133361| 362584|          370578|             743414|      554877|  1122914|   3343411| 3016807|      741630|   1090066|   1287099|  1847606|           0|
             0|        0|
+------+-------+-------+----------------+-------------------+------------+---------+----------+--------+------------+----------+----------+---------+------------+--
------------+---------+
```



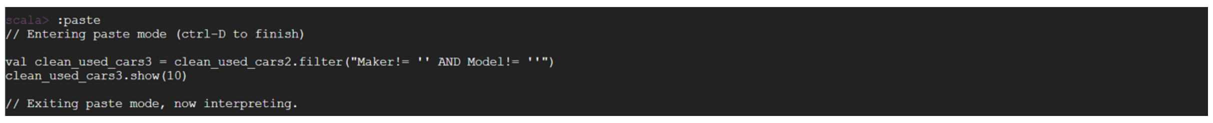
```
// Entering paste mode (ctrl-D to finish)

val used_cars_by_price = used_cars.groupBy(col("Price_eur")).count()
used_cars_by_price.show()

// Exiting paste mode, now interpreting.

+---------+-----+
|Price_eur|count|
+---------+-----+
|  15573.5|    3|
| 28620.76|    5|
|  2499.19|  169|
| 13943.71|    1|
| 13806.62|    8|
| 14797.11|   10|
|  34966.8|    2|
|  8512.21|  956|
| 14111.77|    1|
| 26777.87|    2|
| 17995.34|  185|
|  9752.89|   46|
| 89926.61|    9|
|  3700.93|  682|
| 23948.08|   29|
| 16437.86|    1|
| 16662.92|   57|
| 12738.79|    6|
|  25999.7|    5|
| 34371.17|    4|
+---------+-----+
only showing top 20 rows
```



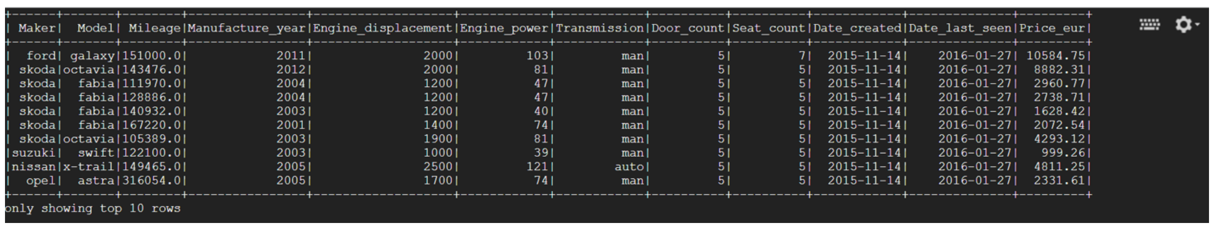
```
scala> :paste
// Entering paste mode (ctrl-D to finish)

val clean_used_cars1 = used_cars.drop("Fuel_type","Color_slug","Stk_year","Body_type")
clean_used_cars1.show(10)

// Exiting paste mode, now interpreting.

+-----+-------+--------+----------------+-------------------+------------+------------+----------+----------+------------+--------------+---------+
|Maker|  Model| Mileage|Manufacture_year|Engine_displacement|Engine_power|Transmission|Door_count|Seat_count|Date_created|Date_last_seen|Price_eur|
+-----+-------+--------+----------------+-------------------+------------+------------+----------+----------+------------+--------------+---------+
| ford| galaxy|151000.0|            2011|               2000|         103|         man|         5|         7|  2015-11-14|    2016-01-27| 10584.75|
|skoda|octavia|143476.0|            2012|               2000|          81|         man|         5|         5|  2015-11-14|    2016-01-27|  8882.31|
|  bmw|   null| 97676.0|            2010|               1995|          85|         man|         5|         5|  2015-11-14|    2016-01-27| 12065.06|
|skoda|  fabia|111970.0|            2004|               1200|          47|         man|         5|         5|  2015-11-14|    2016-01-27|  2960.77|
|skoda|  fabia|128886.0|            2004|               1200|          47|         man|         5|         5|  2015-11-14|    2016-01-27|  2738.71|
|skoda|  fabia|140932.0|            2003|               1200|          40|         man|         5|         5|  2015-11-14|    2016-01-27|  1628.42|
|skoda|  fabia|167220.0|            2001|               1400|          74|         man|         5|         5|  2015-11-14|    2016-01-27|  2072.54|
|  bmw|   null|148500.0|            2009|               2000|         130|        auto|         5|         5|  2015-11-14|    2016-01-27| 10547.74|
|skoda|octavia|105389.0|            2003|               1900|          81|         man|         5|         5|  2015-11-14|    2016-01-27|  4293.12|
| null|   null|301381.0|            2002|               1900|          88|         man|         5|         5|  2015-11-14|    2016-01-27|  1332.35|
```



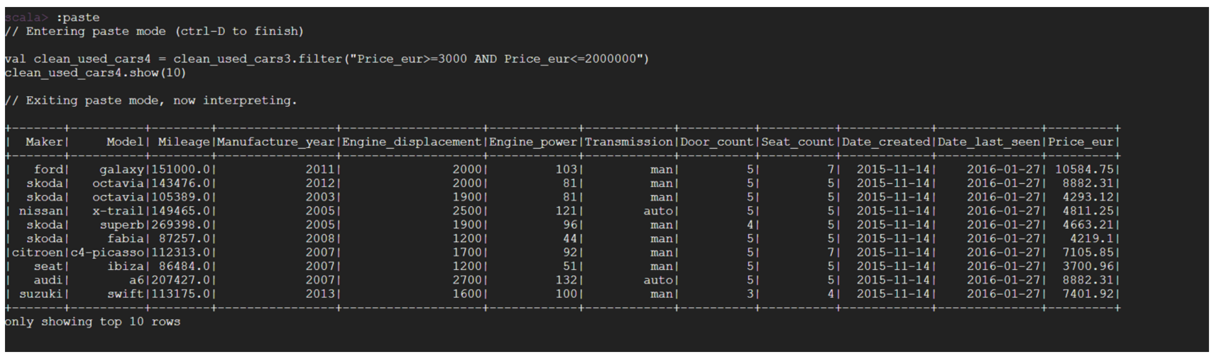
```
scala> :paste
// Entering paste mode (ctrl-D to finish)

val clean_used_cars2= clean_used_cars1.filter("Manufacture_year>=2000 AND Manufacture_year<=2017")
clean_used_cars2.show(10)

// Exiting paste mode, now interpreting.

+-----+-------+--------+----------------+-------------------+------------+------------+----------+----------+------------+--------------+---------+
|Maker|  Model| Mileage|Manufacture_year|Engine_displacement|Engine_power|Transmission|Door_count|Seat_count|Date_created|Date_last_seen|Price_eur|
+-----+-------+--------+----------------+-------------------+------------+------------+----------+----------+------------+--------------+---------+
| ford| galaxy|151000.0|            2011|               2000|         103|         man|         5|         7|  2015-11-14|    2016-01-27| 10584.75|
|skoda|octavia|143476.0|            2012|               2000|          81|         man|         5|         5|  2015-11-14|    2016-01-27|  8882.31|
|  bmw|   null| 97676.0|            2010|               1995|          85|         man|         5|         5|  2015-11-14|    2016-01-27| 12065.06|
|skoda|  fabia|111970.0|            2004|               1200|          47|         man|         5|         5|  2015-11-14|    2016-01-27|  2960.77|
|skoda|  fabia|128886.0|            2004|               1200|          47|         man|         5|         5|  2015-11-14|    2016-01-27|  2738.71|
|skoda|  fabia|140932.0|            2003|               1200|          40|         man|         5|         5|  2015-11-14|    2016-01-27|  1628.42|
|skoda|  fabia|167220.0|            2001|               1400|          74|         man|         5|         5|  2015-11-14|    2016-01-27|  2072.54|
|  bmw|   null|148500.0|            2009|               2000|         130|        auto|         5|         5|  2015-11-14|    2016-01-27| 10547.74|
|skoda|octavia|105389.0|            2003|               1900|          81|         man|         5|         5|  2015-11-14|    2016-01-27|  4293.12|
| null|   null|301381.0|            2002|               1900|          88|         man|         5|         5|  2015-11-14|    2016-01-27|  1332.35|
+-----+-------+--------+----------------+-------------------+------------+------------+----------+----------+------------+--------------+---------+
only showing top 10 rows

clean_used_cars2: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [Maker: string, Model: string ... 10 more fields]
```



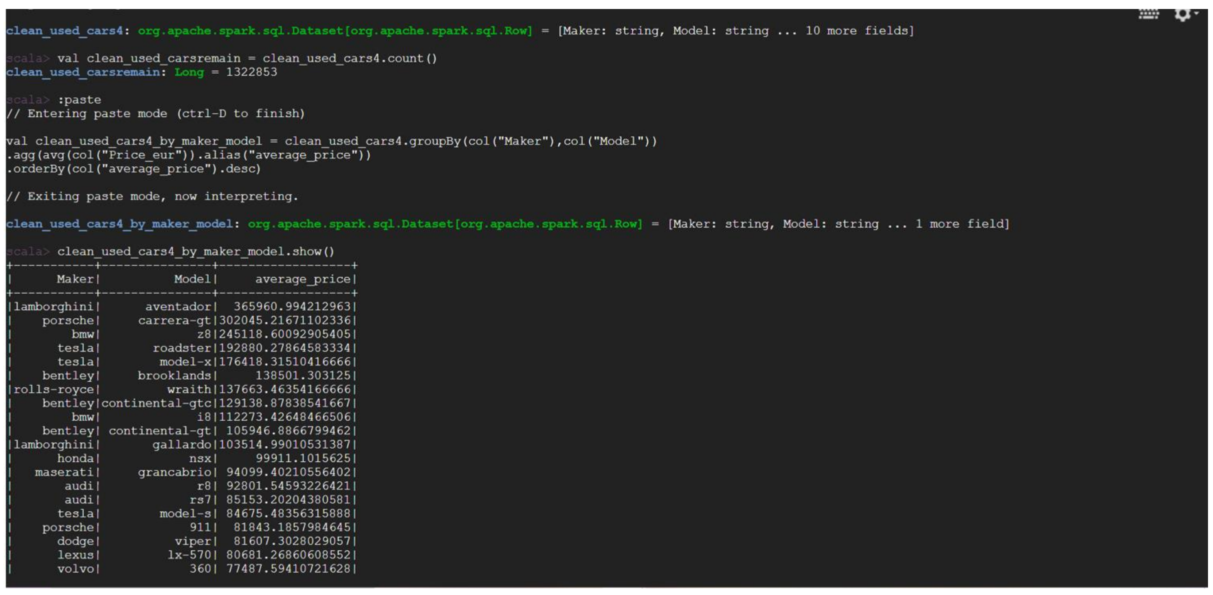
```
scala> :paste
// Entering paste mode (ctrl-D to finish)

val clean_used_cars3 = clean_used_cars2.filter("Maker!= '' AND Model!= ''")
clean_used_cars3.show(10)

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



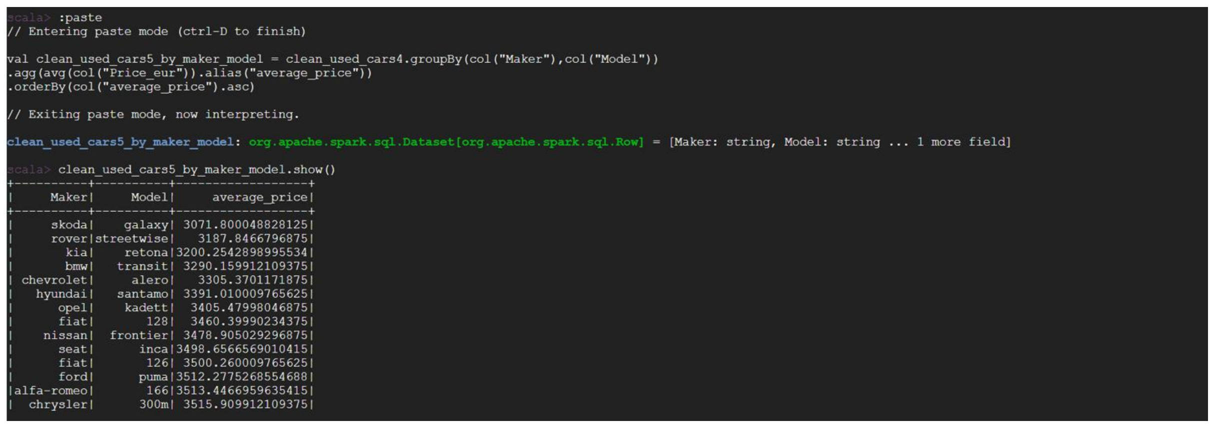
```
+------+-------+--------+----------------+-------------------+------------+------------+----------+----------+------------+--------------+---------+
| Maker|  Model| Mileage|Manufacture_year|Engine_displacement|Engine_power|Transmission|Door_count|Seat_count|Date_created|Date_last_seen|Price_eur|
+------+-------+--------+----------------+-------------------+------------+------------+----------+----------+------------+--------------+---------+
|  ford| galaxy|151000.0|            2011|               2000|         103|         man|         5|         7|  2015-11-14|    2016-01-27| 10584.75|
| skoda|octavia|143476.0|            2012|               2000|          81|         man|         5|         5|  2015-11-14|    2016-01-27|  8882.31|
| skoda|  fabia|111970.0|            2004|               1200|          47|         man|         5|         5|  2015-11-14|    2016-01-27|  2960.77|
| skoda|  fabia|128886.0|            2004|               1200|          47|         man|         5|         5|  2015-11-14|    2016-01-27|  2738.71|
| skoda|  fabia|140932.0|            2003|               1200|          40|         man|         5|         5|  2015-11-14|    2016-01-27|  1628.42|
| skoda|  fabia|167220.0|            2001|               1400|          74|         man|         5|         5|  2015-11-14|    2016-01-27|  2072.54|
| skoda|octavia|105389.0|            2003|               1900|          81|         man|         5|         5|  2015-11-14|    2016-01-27|  4293.12|
|suzuki|  swift|122100.0|            2003|               1000|          39|         man|         5|         5|  2015-11-14|    2016-01-27|   999.26|
|nissan|x-trail|149465.0|            2005|               2500|         121|        auto|         5|         5|  2015-11-14|    2016-01-27|  4811.25|
|  opel|  astra|316054.0|            2005|               1700|          74|         man|         5|         5|  2015-11-14|    2016-01-27|  2331.61|
+------+-------+--------+----------------+-------------------+------------+------------+----------+----------+------------+--------------+---------+
only showing top 10 rows
```



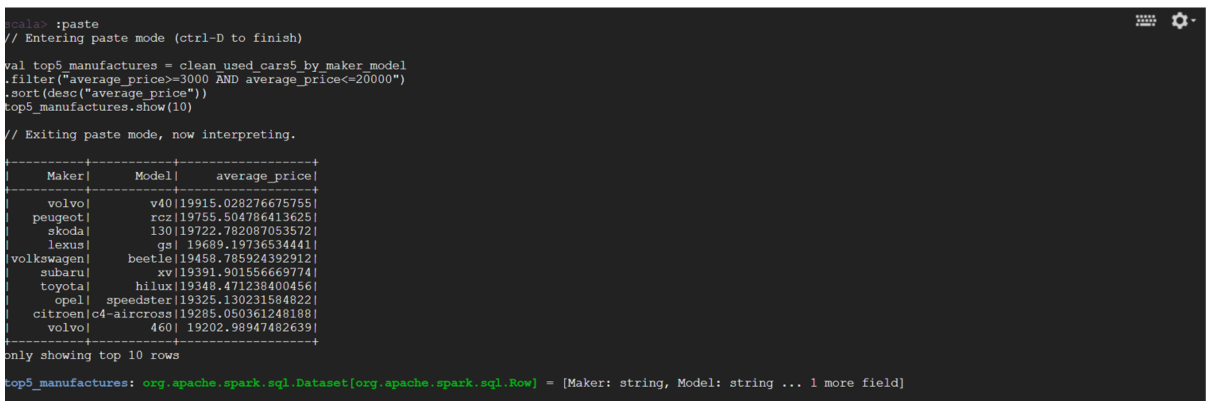
```
scala> :paste
// Entering paste mode (ctrl-D to finish)

val clean_used_cars4 = clean_used_cars3.filter("Price_eur>=3000 AND Price_eur<=2000000")
clean_used_cars4.show(10)

// Exiting paste mode, now interpreting.

+-------+----------+--------+----------------+-------------------+------------+------------+----------+----------+------------+--------------+---------+
|  Maker|     Model| Mileage|Manufacture_year|Engine_displacement|Engine_power|Transmission|Door_count|Seat_count|Date_created|Date_last_seen|Price_eur|
+-------+----------+--------+----------------+-------------------+------------+------------+----------+----------+------------+--------------+---------+
|   ford|    galaxy|151000.0|            2011|               2000|         103|         man|         5|         7|  2015-11-14|    2016-01-27| 10584.75|
|  skoda|   octavia|143476.0|            2012|               2000|          81|         man|         5|         5|  2015-11-14|    2016-01-27|  8882.31|
|  skoda|   octavia|105389.0|            2003|               1900|          81|         man|         5|         5|  2015-11-14|    2016-01-27|  4293.12|
| nissan|   x-trail|149465.0|            2005|               2500|         121|        auto|         5|         5|  2015-11-14|    2016-01-27|  4811.25|
|  skoda|    superb|269398.0|            2005|               1900|          96|         man|         4|         5|  2015-11-14|    2016-01-27|  4663.21|
|  skoda|     fabia| 87257.0|            2008|               1200|          44|         man|         5|         5|  2015-11-14|    2016-01-27|   4219.1|
|citroen|c4-picasso|112313.0|            2007|               1700|          92|         man|         5|         7|  2015-11-14|    2016-01-27|  7105.85|
|   seat|     ibiza| 86484.0|            2007|               1200|          51|         man|         5|         5|  2015-11-14|    2016-01-27|  3700.96|
|   audi|        a6|207427.0|            2007|               2700|         132|        auto|         5|         5|  2015-11-14|    2016-01-27|  8882.31|
| suzuki|     swift|113175.0|            2013|               1600|         100|         man|         3|         4|  2015-11-14|    2016-01-27|  7401.92|
+-------+----------+--------+----------------+-------------------+------------+------------+----------+----------+------------+--------------+---------+
only showing top 10 rows
```



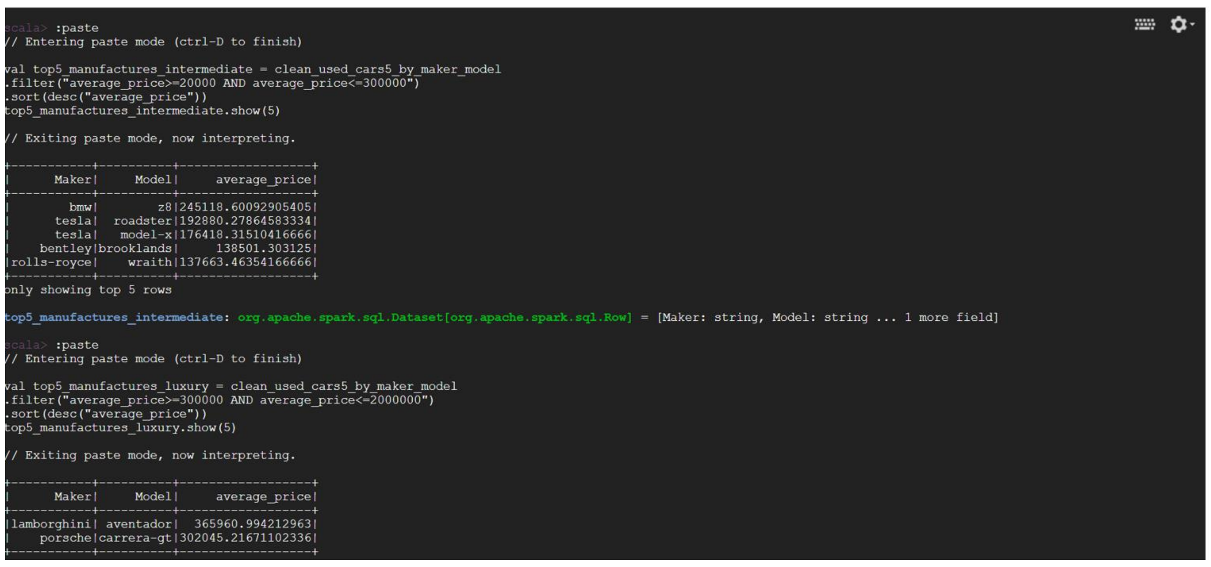
```
clean_used_cars4: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [Maker: string, Model: string ... 10 more fields]

scala> val clean_used_carsremain = clean_used_cars4.count()
clean_used_carsremain: Long = 1322853

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

val clean_used_cars4_by_maker_model = clean_used_cars4.groupBy(col("Maker"),col("Model"))
.agg(avg(col("Price_eur")).alias("average_price"))
.orderBy(col("average_price").desc)

// Exiting paste mode, now interpreting.

clean_used_cars4_by_maker_model: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [Maker: string, Model: string ... 1 more field]

scala> clean_used_cars4_by_maker_model.show()
+-----------+---------------+------------------+
|      Maker|          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|
|lamborghini|       gallardo|103514.99010531387|
|      honda|            nsx|     99911.1015625|
|   maserati|     grancabrio| 94099.40210556402|
|       audi|             r8| 92801.54593226421|
|       audi|            rs7| 85153.20204380581|
|      tesla|        model-s| 84675.48356315888|
|    porsche|            911|  81843.1857984645|
|      dodge|          viper|  81607.3028029057|
|      lexus|         lx-570| 80681.26860608552|
|      volvo|            360| 77487.59410721628|
```



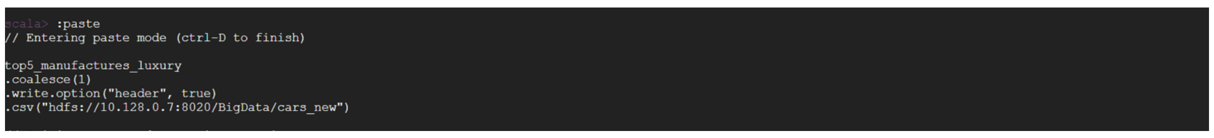
```
scala> :paste
// Entering paste mode (ctrl-D to finish)

val clean_used_cars5_by_maker_model = clean_used_cars4.groupBy(col("Maker"),col("Model"))
.agg(avg(col("Price_eur")).alias("average_price"))
.orderBy(col("average_price").asc)

// Exiting paste mode, now interpreting.

clean_used_cars5_by_maker_model: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [Maker: string, Model: string ... 1 more field]

scala> clean_used_cars5_by_maker_model.show()
+----------+----------+-----------------+
|     Maker|     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|
|      fiat|       126|3500.260009765625|
|      ford|      puma|3512.2775268554688|
|alfa-romeo|       166|3513.4466959635415|
|  chrysler|      300m|3515.909912109375|
```



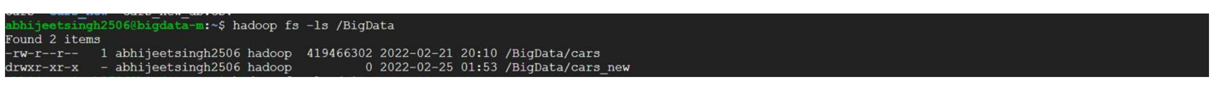
```
scala> :paste
// Entering paste mode (ctrl-D to finish)

val top5_manufactures = clean_used_cars5_by_maker_model
.filter("average_price>=3000 AND average_price<=20000")
.sort(desc("average_price"))
top5_manufactures.show(10)

// Exiting paste mode, now interpreting.

+----------+----------+------------------+
|     Maker|     Model|     average_price|
+----------+----------+------------------+
|     volvo|       v40|19915.028276675755|
|   peugeot|       rcz|19755.504786413625|
|     skoda|       130|19722.782087053572|
|     lexus|        gs| 19689.19736534441|
|volkswagen|    beetle|19458.785924392912|
|    subaru|        xv|19391.901556669774|
|    toyota|     hilux|19348.471238400456|
|      opel| speedster|19325.130231584822|
|   citroen|c4-aircross|19285.050361248188|
|     volvo|       460| 19202.98947482639|
+----------+----------+------------------+
only showing top 10 rows

top5_manufactures: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [Maker: string, Model: string ... 1 more field]
```



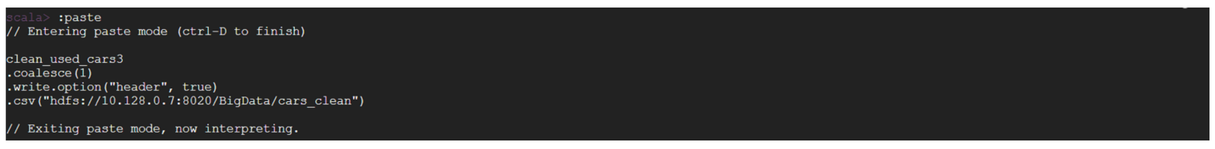
```
scala> :paste
// Entering paste mode (ctrl-D to finish)

val top5_manufactures_intermediate = clean_used_cars5_by_maker_model
.filter("average_price>=20000 AND average_price<=300000")
.sort(desc("average_price"))
top5_manufactures_intermediate.show(5)

// Exiting paste mode, now interpreting.

+-----------+----------+------------------+
|      Maker|     Model|     average_price|
+-----------+----------+------------------+
|        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

top5_manufactures_intermediate: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [Maker: string, Model: string ... 1 more field]

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

val top5_manufactures_luxury = clean_used_cars5_by_maker_model
.filter("average_price>=300000 AND average_price<=2000000")
.sort(desc("average_price"))
top5_manufactures_luxury.show(5)

// Exiting paste mode, now interpreting.

+-----------+----------+------------------+
|      Maker|     Model|     average_price|
+-----------+----------+------------------+
|lamborghini| aventador|  365960.994212963|
|    porsche|carrera-gt|302045.21671102336|
+-----------+----------+------------------+
```



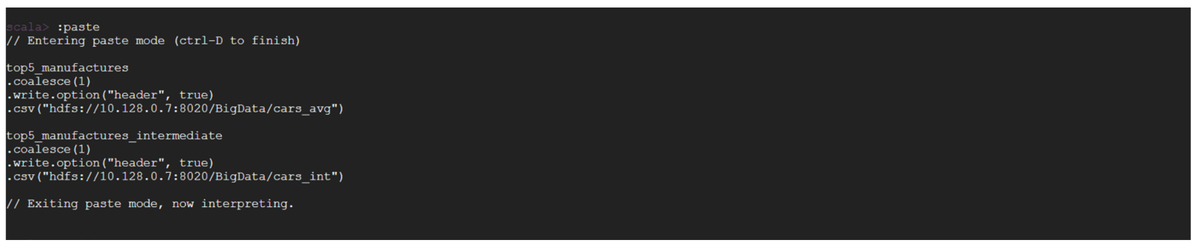
```
scala> :paste
// Entering paste mode (ctrl-D to finish)

top5_manufactures_luxury
.coalesce(1)
.write.option("header", true)
.csv("hdfs://10.128.0.7:8020/BigData/cars_new")
```



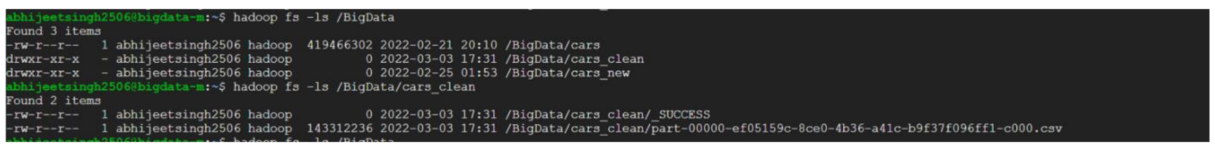
```
abhijeetsingh2506@bigdata-m:~$ hadoop fs -ls /BigData
Found 2 items
-rw-r--r--   1 abhijeetsingh2506 hadoop  419466302 2022-02-21 20:10 /BigData/cars
drwxr-xr-x   - abhijeetsingh2506 hadoop          0 2022-02-25 01:53 /BigData/cars_new
```



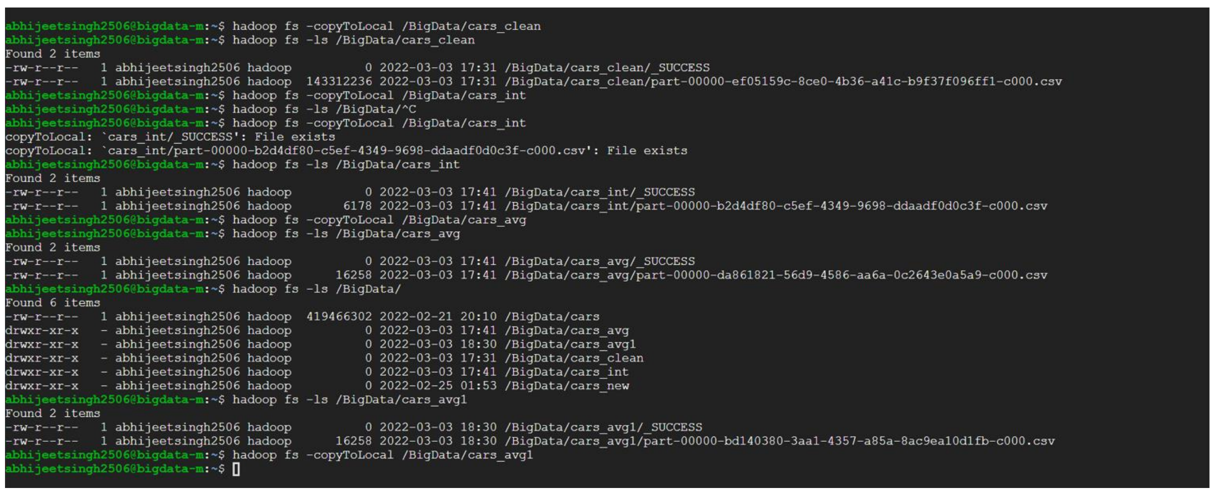
```
scala> :paste
// Entering paste mode (ctrl-D to finish)

clean_used_cars3
.coalesce(1)
.write.option("header", true)
.csv("hdfs://10.128.0.7:8020/BigData/cars_clean")

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



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

top5_manufactures
.coalesce(1)
.write.option("header", true)
.csv("hdfs://10.128.0.7:8020/BigData/cars_avg")

top5_manufactures_intermediate
.coalesce(1)
.write.option("header", true)
.csv("hdfs://10.128.0.7:8020/BigData/cars_int")

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



```
abhijeetsingh2506@bigdata-m:~$ hadoop fs -ls /BigData
Found 3 items
-rw-r--r--   1 abhijeetsingh2506 hadoop  419466302 2022-02-21 20:10 /BigData/cars
drwxr-xr-x   - abhijeetsingh2506 hadoop          0 2022-03-03 17:31 /BigData/cars_clean
drwxr-xr-x   - abhijeetsingh2506 hadoop          0 2022-02-25 01:53 /BigData/cars_new
abhijeetsingh2506@bigdata-m:~$ hadoop fs -ls /BigData/cars_clean
Found 2 items
-rw-r--r--   1 abhijeetsingh2506 hadoop          0 2022-03-03 17:31 /BigData/cars_clean/_SUCCESS
-rw-r--r--   1 abhijeetsingh2506 hadoop  143312236 2022-03-03 17:31 /BigData/cars_clean/part-00000-ef05159c-8ce0-4b36-a41c-b9f37f096ff1-c000.csv
```



```
abhijeetsingh2506@bigdata-m:~$ hadoop fs -copyToLocal /BigData/cars_clean
abhijeetsingh2506@bigdata-m:~$ hadoop fs -ls /BigData/cars_clean
Found 2 items
-rw-r--r--   1 abhijeetsingh2506 hadoop          0 2022-03-03 17:31 /BigData/cars_clean/_SUCCESS
-rw-r--r--   1 abhijeetsingh2506 hadoop  143312236 2022-03-03 17:31 /BigData/cars_clean/part-00000-ef05159c-8ce0-4b36-a41c-b9f37f096ff1-c000.csv
abhijeetsingh2506@bigdata-m:~$ hadoop fs -copyToLocal /BigData/cars_int
abhijeetsingh2506@bigdata-m:~$ hadoop fs -ls /BigData/^C
abhijeetsingh2506@bigdata-m:~$ hadoop fs -copyToLocal /BigData/cars_int
copyToLocal: `cars_int/_SUCCESS': File exists
copyToLocal: `cars_int/part-00000-b2d4df80-c5ef-4349-9698-ddaadf0d0c3f-c000.csv': File exists
abhijeetsingh2506@bigdata-m:~$ hadoop fs -ls /BigData/cars_int
Found 2 items
-rw-r--r--   1 abhijeetsingh2506 hadoop          0 2022-03-03 17:41 /BigData/cars_int/_SUCCESS
-rw-r--r--   1 abhijeetsingh2506 hadoop       6178 2022-03-03 17:41 /BigData/cars_int/part-00000-b2d4df80-c5ef-4349-9698-ddaadf0d0c3f-c000.csv
abhijeetsingh2506@bigdata-m:~$ hadoop fs -copyToLocal /BigData/cars_avg
abhijeetsingh2506@bigdata-m:~$ hadoop fs -ls /BigData/cars_avg
Found 2 items
-rw-r--r--   1 abhijeetsingh2506 hadoop          0 2022-03-03 17:41 /BigData/cars_avg/_SUCCESS
-rw-r--r--   1 abhijeetsingh2506 hadoop      16258 2022-03-03 17:41 /BigData/cars_avg/part-00000-da861821-56d9-4586-aa6a-0c2643e0a5a9-c000.csv
abhijeetsingh2506@bigdata-m:~$ hadoop fs -ls /BigData/
Found 6 items
-rw-r--r--   1 abhijeetsingh2506 hadoop  419466302 2022-02-21 20:10 /BigData/cars
drwxr-xr-x   - abhijeetsingh2506 hadoop          0 2022-03-03 17:41 /BigData/cars_avg
drwxr-xr-x   - abhijeetsingh2506 hadoop          0 2022-03-03 18:30 /BigData/cars_avg1
drwxr-xr-x   - abhijeetsingh2506 hadoop          0 2022-03-03 17:31 /BigData/cars_clean
drwxr-xr-x   - abhijeetsingh2506 hadoop          0 2022-03-03 17:41 /BigData/cars_int
drwxr-xr-x   - abhijeetsingh2506 hadoop          0 2022-02-25 01:53 /BigData/cars_new
abhijeetsingh2506@bigdata-m:~$ hadoop fs -ls /BigData/cars_avg1
Found 2 items
-rw-r--r--   1 abhijeetsingh2506 hadoop          0 2022-03-03 18:30 /BigData/cars_avg1/_SUCCESS
-rw-r--r--   1 abhijeetsingh2506 hadoop      16258 2022-03-03 18:30 /BigData/cars_avg1/part-00000-bd140380-3aa1-4357-a85a-8ac9ea10d1fb-c000.csv
abhijeetsingh2506@bigdata-m:~$ hadoop fs -copyToLocal /BigData/cars_avg1
abhijeetsingh2506@bigdata-m:~$ 
```

Reference

https://stackoverflow.com/questions/44413132/count-the-number-of-missing-values-in-a-dataframe-spark