Data Analytics with Spark DataFrames

Kayla Liu

Introduction

The dataset used in this report was extracted from kaggle which contains used car information and sales situation in the Czech Republic and Germany since 2015, including key vehicle information such as car maker, mar model, mileage, manufacture year, fuel type, etc. Based on this accurate and compliant database, the data analysis report aims to comprehensively organize various indicators and data source, provide users with detailed data induction and intelligent analysis, and further provide data-supported advice and recommendations.

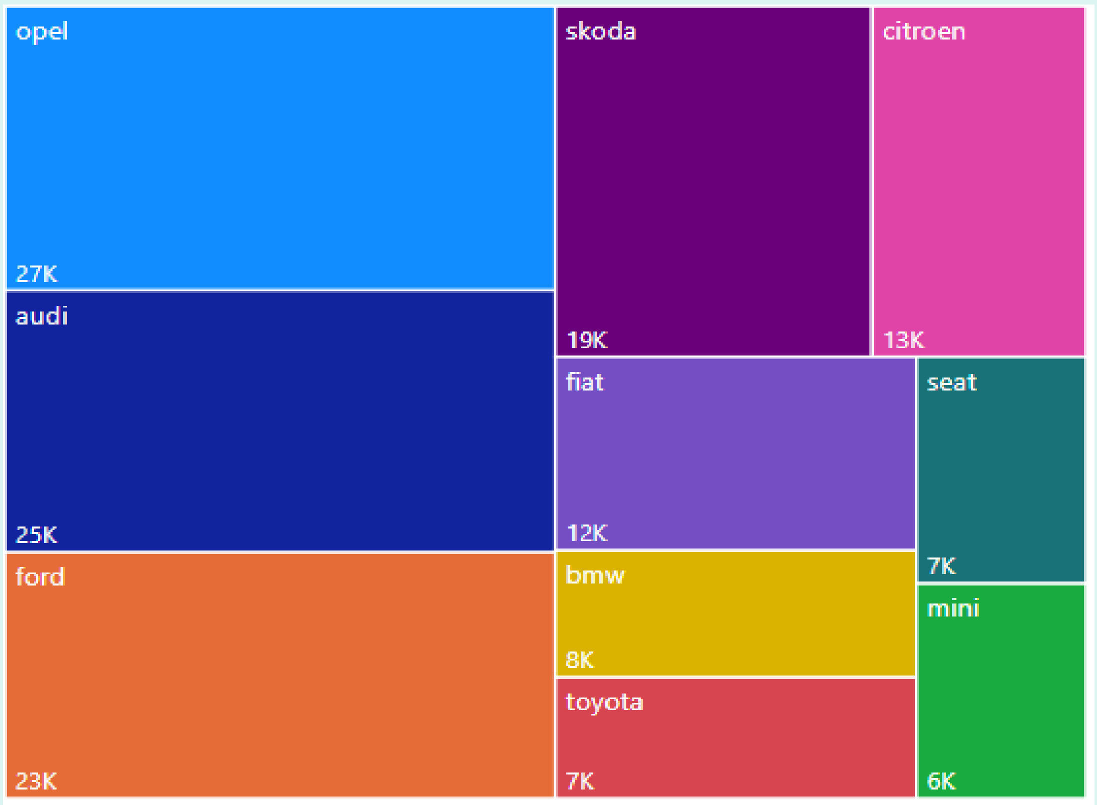
Business Questions

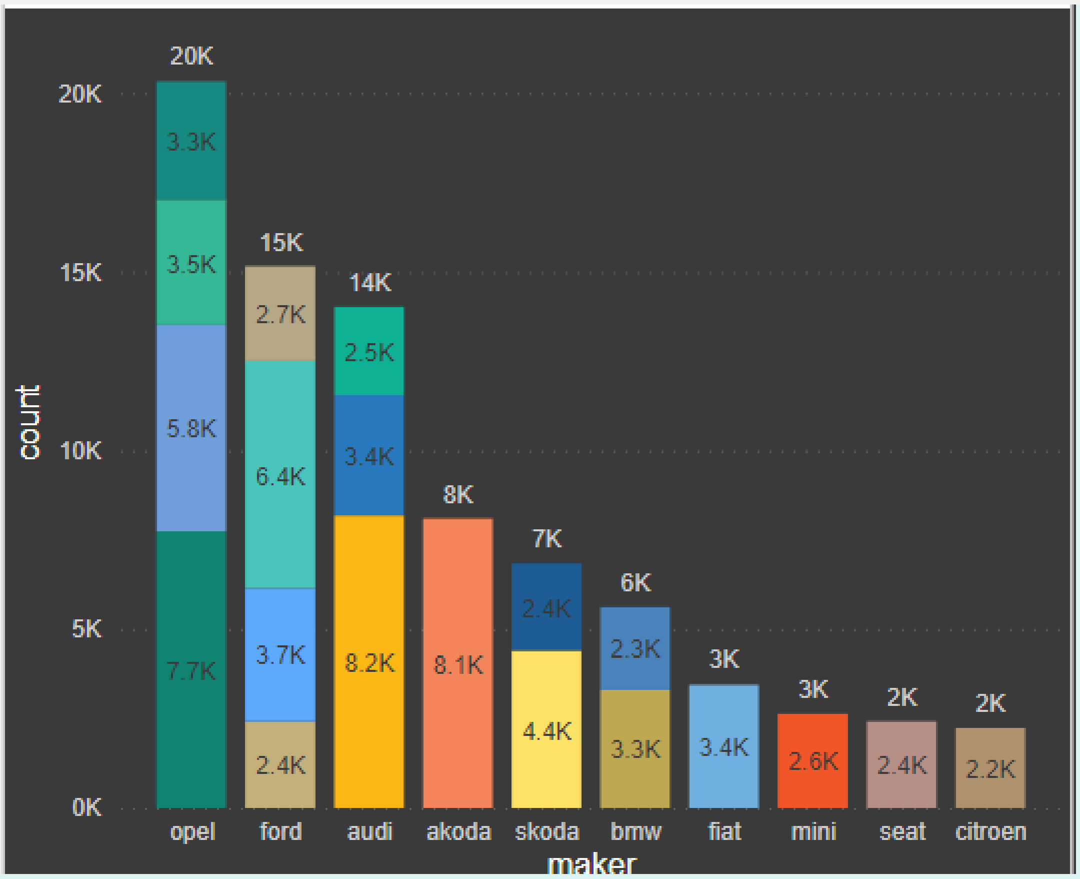
- Top10 car brand and vehicle the company should buy for the used car business
- What is the relationship between car makes, models and manufacture year?
- Does fuel type have any impact on the car price?
- The market share of related used car performance, such as transmission type, door and seat

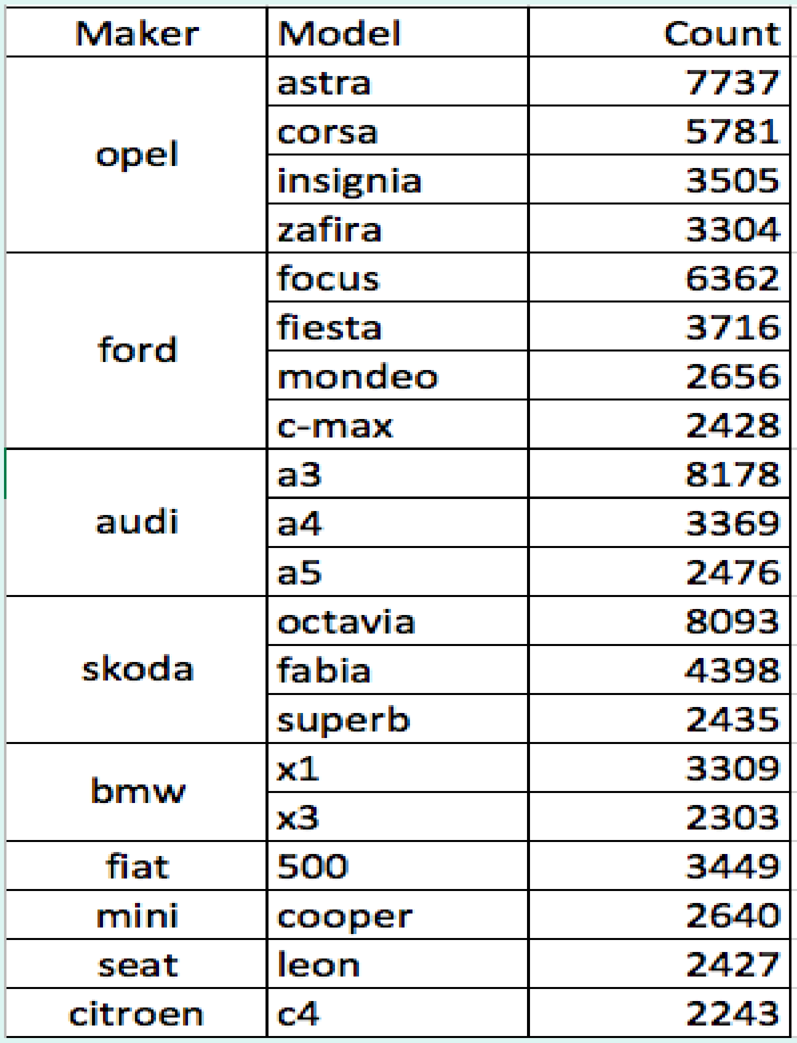
Data Analysis Result

Q1: Top5 car brand and vehicle the company should buy for the used car business

Here are the distribution of the top 10 used car makers and the most popular models during year 2006-2016.


opel
27K
audi
25K
ford
23K
skoda
19K
citroen
13K
fiat
12K
seat
7K
bmw
8K
mini
toyota
7K
6K


count
maker
20K
15K
10K
5K
0K
opel
ford
audi
akoda
skoda
bmw
fiat
mini
seat
citroen
20K
3.3K
3.5K
5.8K
7.7K
15K
2.7K
6.4K
3.7K
2.4K
14K
2.5K
3.4K
8.2K
8K
8.1K
7K
2.4K
4.4K
6K
2.3K
3.3K
3K
3.4K
3K
2.6K
2K
2.4K
2K
2.2K

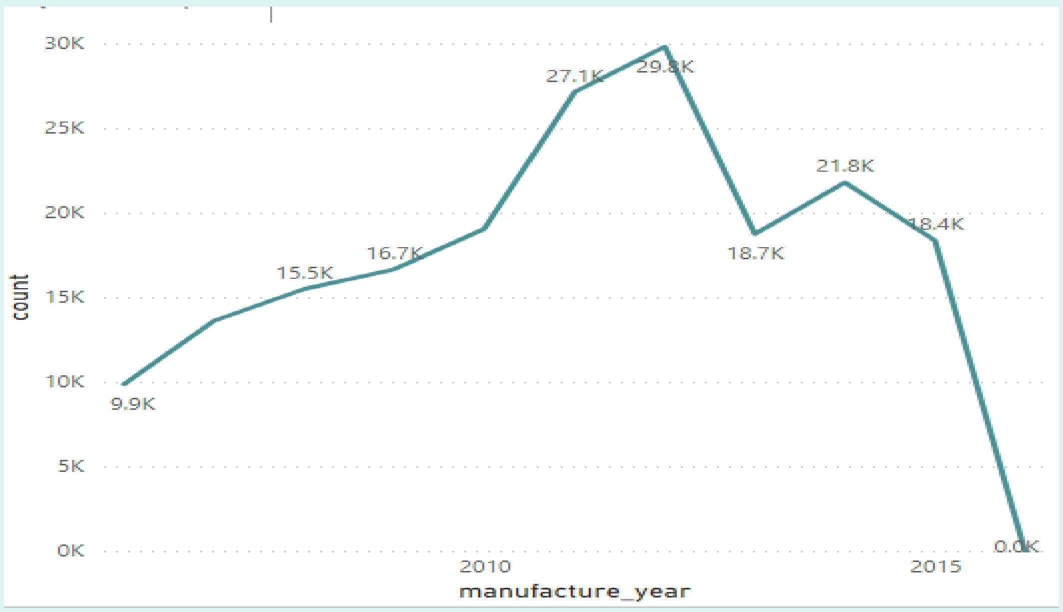


Maker	Model	Count
opel	astra	7737
	corsa	5781
	insignia	3505
	zafira	3304
ford	focus	6362
	fiesta	3716
	mondeo	2656
	c-max	2428
audi	a3	8178
	a4	3369
	a5	2476
skoda	octavia	8093
	fabia	4398
	superb	2435
bmw	x1	3309
	x3	2303
fiat	500	3449
mini	cooper	2640
seat	leon	2427
citroen	c4	2243

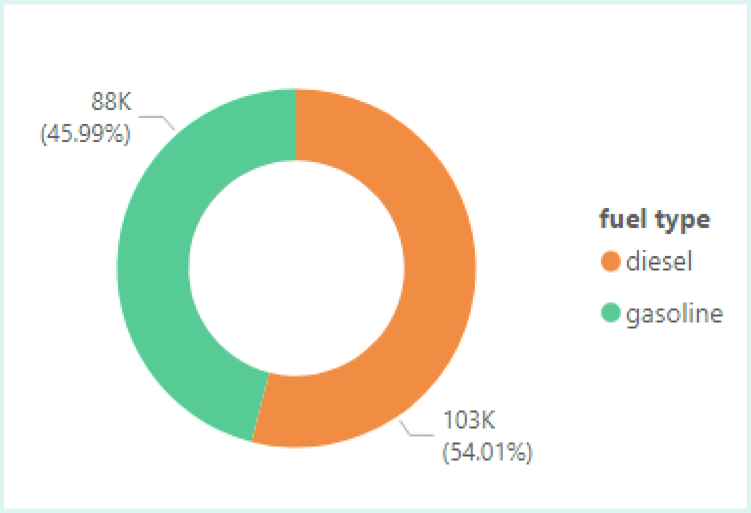
Based on these three graphs and table, the Top 10 used car makers are opel, audi,ford,skoda,citroen, flat,bmw,Toyota, seat,mini respectively. And here also shows the most popular car models from these car brands are: audi a3, skoda cotavia, opel astra, ford focus, opel corsa and so on.

Q2: What is the relationship between car makes, models and manufacture year?

From the perspective of automobile trading volume, during 2006-2016, it was basically concentrated in 2010-2015, among which, the market circulation rate of automobiles produced in 2012 was the highest, followed by 2011, 2014, 2010 and 2015.

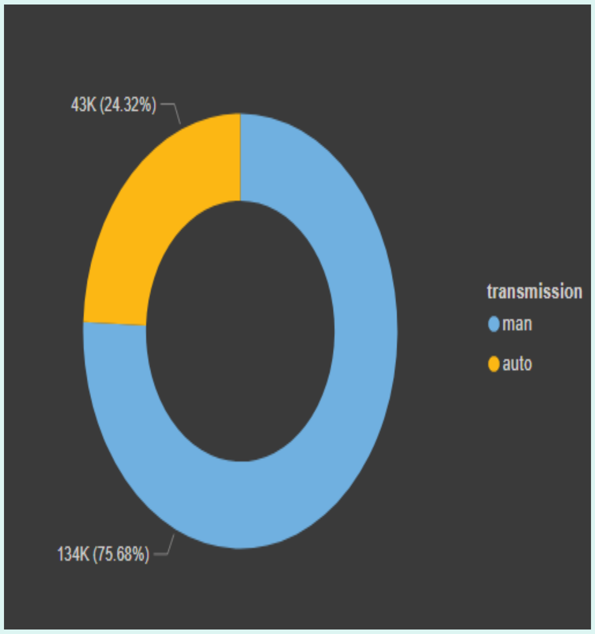


Q3: Does fuel type have any impact on the car price?

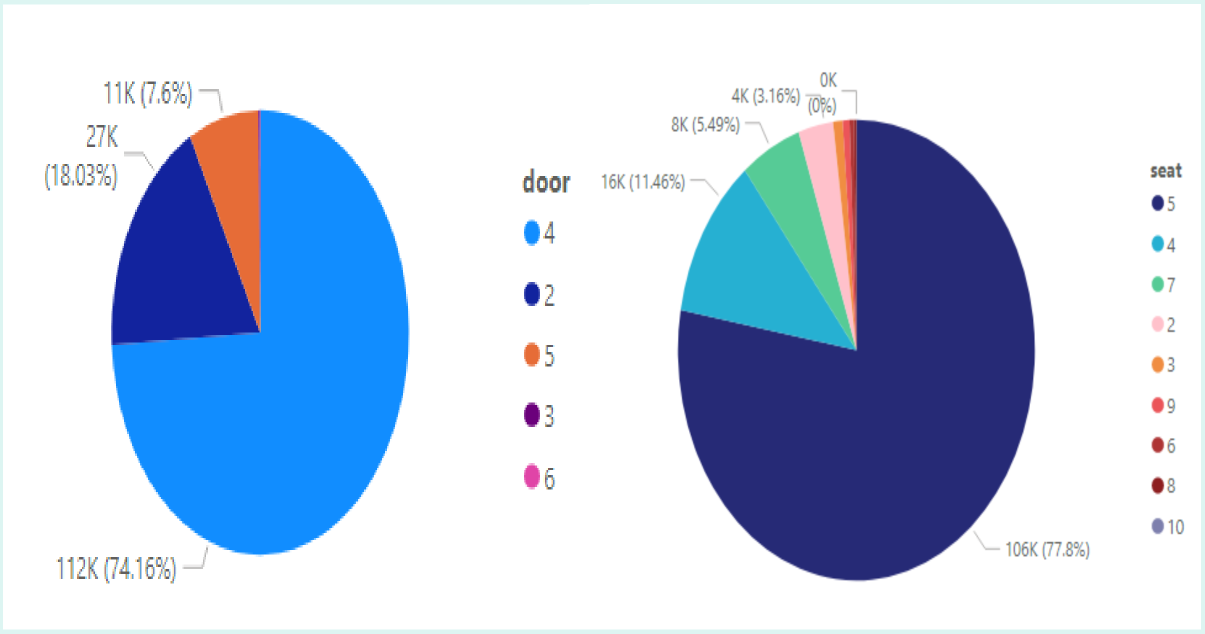


54% used car's fuel type is diesel, 46% is gasoline, so from this dimension, the consumer acceptance of the two fuels is similar, both type of cars can circulate or turn round in the market.

Q4: The market share of related used car performance, such as transmission type, door and seat



In the second-hand car market, there are three obvious tendencies in transmission, door count and seat count. First of all, three quarters are manual transmission, and only one fourth are automatic transmission, which is also related to the year of data source from 2006 to 2016. In addition, 4-door cars are still dominant, then, 2-door or 5-door cars account for about 25% in total. Moreover, 5-seat cars account for 78% of the market, followed by a small distribution of 4-seat, 7-seat etc.



Conclusion & Recommendation

According to the above data analytics, this used car investment should give priority to following   key factors:

1. Makers: the dealership should focus more on the top5 makers,which are **opel, audi, ford, skoda, citroda**. These brands may attract more potential consumers and stable car sources to develop and expand market share
2. Models: more popular models are also one of the most essential aspects to consider, and the top 5 popular car models: **audi a3, skoda cotavia, opel astra, ford focus, opel corsa**. Moreover, it is worth mentioning that, after further analysis and research, among the five most popular models, Audi a3 has the most stable price in the market, with a concentration of 22,000-26,000 euros
3. Mileage: 12,000 miles is the most common yardstick per year (most leases allow 12,000 miles per year) and the maximum is 200,000 miles to perform reliably, so the company’s mileage criterion may not higher than 200,000 miles
4. Other factors: 4-door, 5-seat, and man transmission cars has the highest turnover on the market

Appendix

Here are the data analytics process:

Load the dataset and the schema

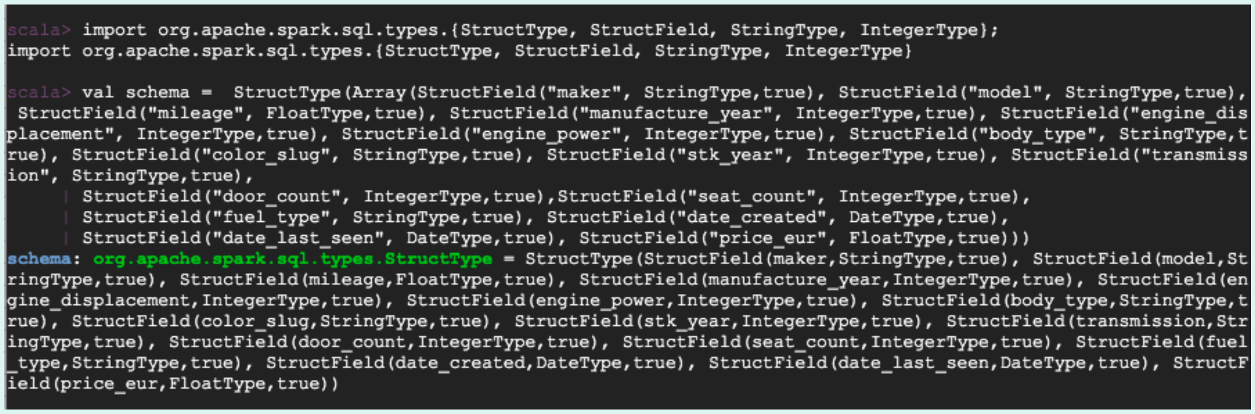
Steps:

#Create the schema

```
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),

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)))
```

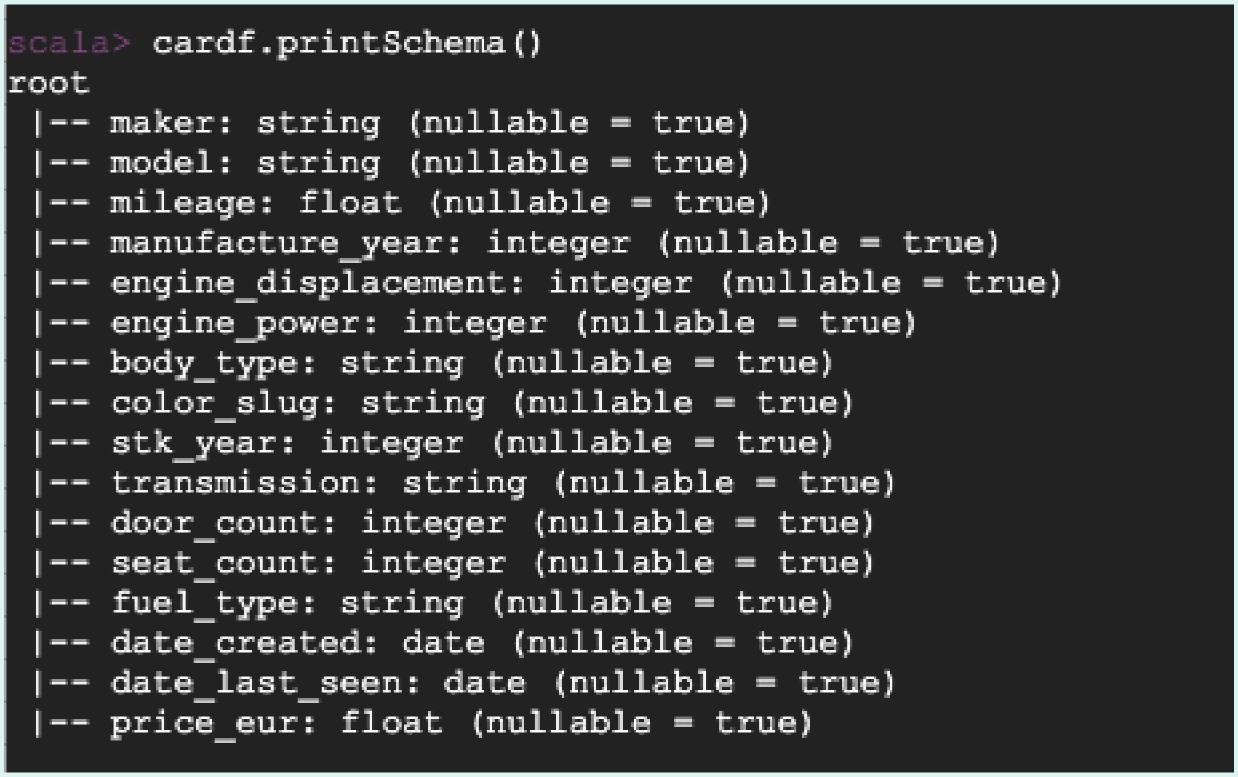
#Load the dataset

```
val cardf = spark.read.option("header", "true").schema(schema).csv("/BigData/car.csv")
```

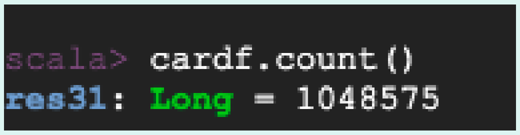


```
scala> import org.apache.spark.sql.types.{StructType, StructField, StringType, IntegerType};
import org.apache.spark.sql.types.{StructType, StructField, StringType, IntegerType}

scala> val schema =  StructType(Array(StructField("maker", StringType,true), StructField("model", StringType,true),
 StructField("mileage", FloatType,true), StructField("manufacture_year", IntegerType,true), StructField("engine_dis
placement", IntegerType,true), StructField("engine_power", IntegerType,true), StructField("body_type", StringType,t
rue), StructField("color_slug", StringType,true), StructField("stk_year", IntegerType,true), StructField("transmiss
ion", 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)))
schema: org.apache.spark.sql.types.StructType = StructType(StructField(maker,StringType,true), StructField(model,St
ringType,true), StructField(mileage,FloatType,true), StructField(manufacture_year,IntegerType,true), StructField(en
gine_displacement,IntegerType,true), StructField(engine_power,IntegerType,true), StructField(body_type,StringType,t
rue), StructField(color_slug,StringType,true), StructField(stk_year,IntegerType,true), StructField(transmission,Str
ingType,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), StructF
ield(price_eur,FloatType,true))
```

```
cardf.printSchema()
```


```
scala> cardf.printSchema()
root
 |-- maker: string (nullable = true)
 |-- model: string (nullable = true)
 |-- mileage: float (nullable = true)
 |-- manufacture_year: integer (nullable = true)
 |-- engine_displacement: integer (nullable = true)
 |-- engine_power: integer (nullable = true)
 |-- body_type: string (nullable = true)
 |-- color_slug: string (nullable = true)
 |-- stk_year: integer (nullable = true)
 |-- transmission: string (nullable = true)
 |-- door_count: integer (nullable = true)
 |-- seat_count: integer (nullable = true)
 |-- fuel_type: string (nullable = true)
 |-- date_created: date (nullable = true)
 |-- date_last_seen: date (nullable = true)
 |-- price_eur: float (nullable = true)
```

cardf.count()


```
scala> cardf.count()
res31: Long = 1048575
```

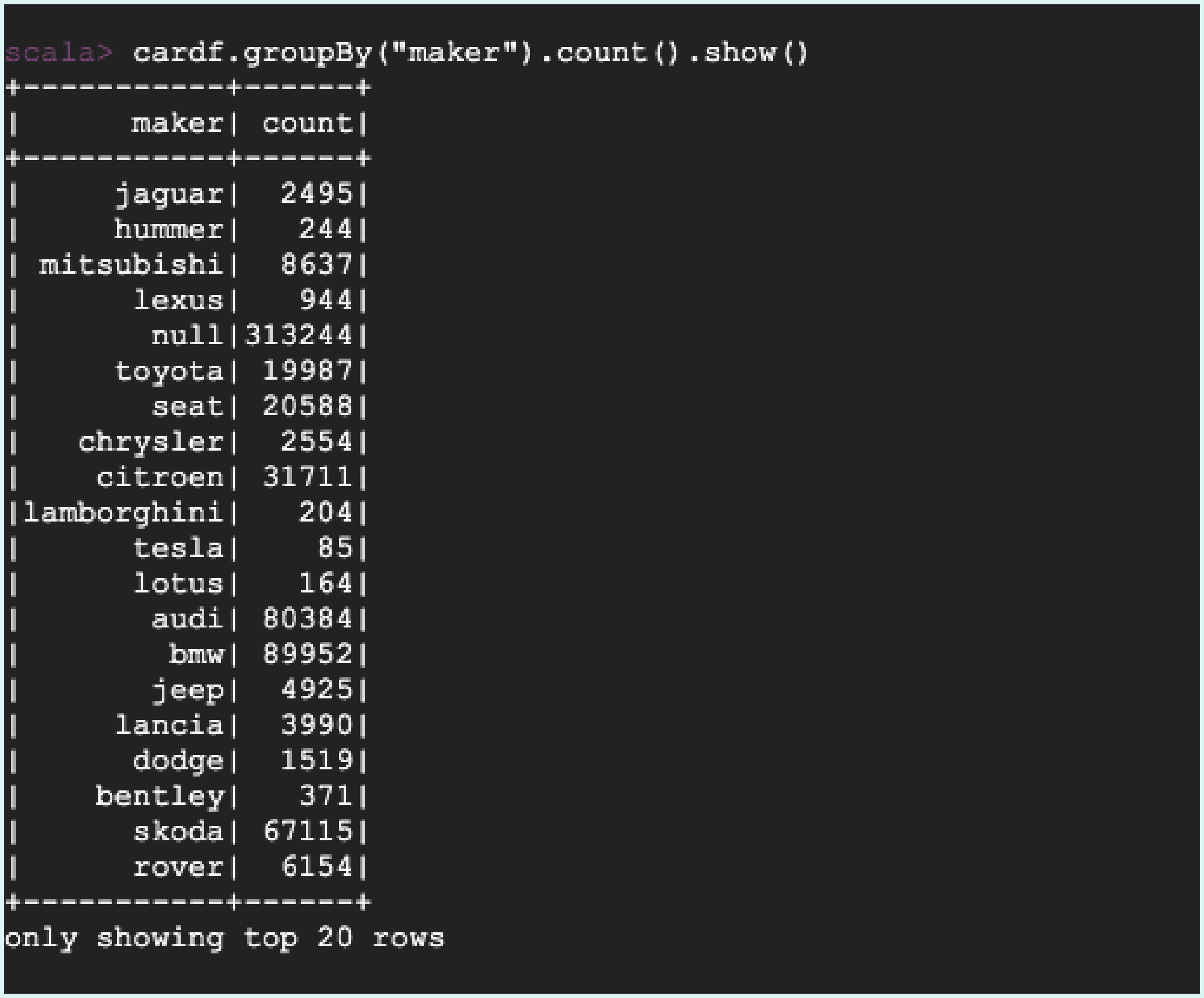
Data Cleaning

Steps:

- Check the overall situation of 16 variables
- Remove null values under maker & model fields
- Clarify the rationality of the remaining data fields
- Identify key factors related to the specific business question

# Check null values of maker & model

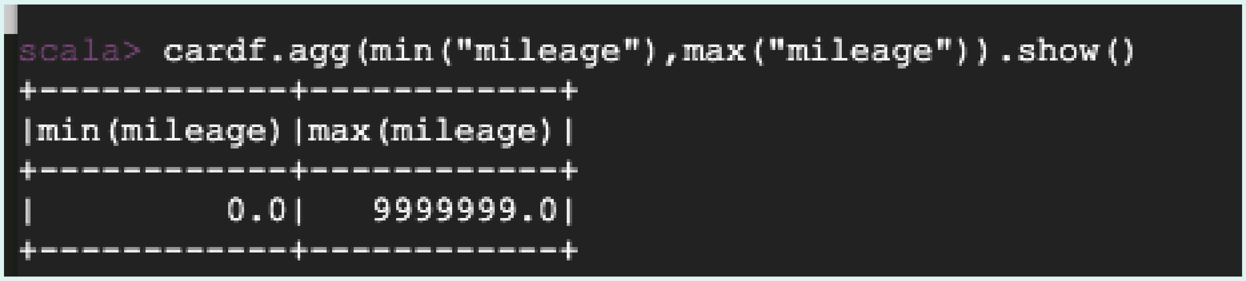
cardf.groupBy("maker").count().show()



```
scala> cardf.groupBy("maker").count().show()
+-----------+------+
|      maker| count|
+-----------+------+
|     jaguar|  2495|
|     hummer|   244|
| mitsubishi|  8637|
|      lexus|   944|
|       null|313244|
|     toyota| 19987|
|       seat| 20588|
|   chrysler|  2554|
|    citroen| 31711|
|lamborghini|   204|
|      tesla|    85|
|      lotus|   164|
|       audi| 80384|
|        bmw| 89952|
|       jeep|  4925|
|     lancia|  3990|
|      dodge|  1519|
|    bentley|   371|
|      skoda| 67115|
|      rover|  6154|
+-----------+------+
only showing top 20 rows
```

# Check the range of mileage

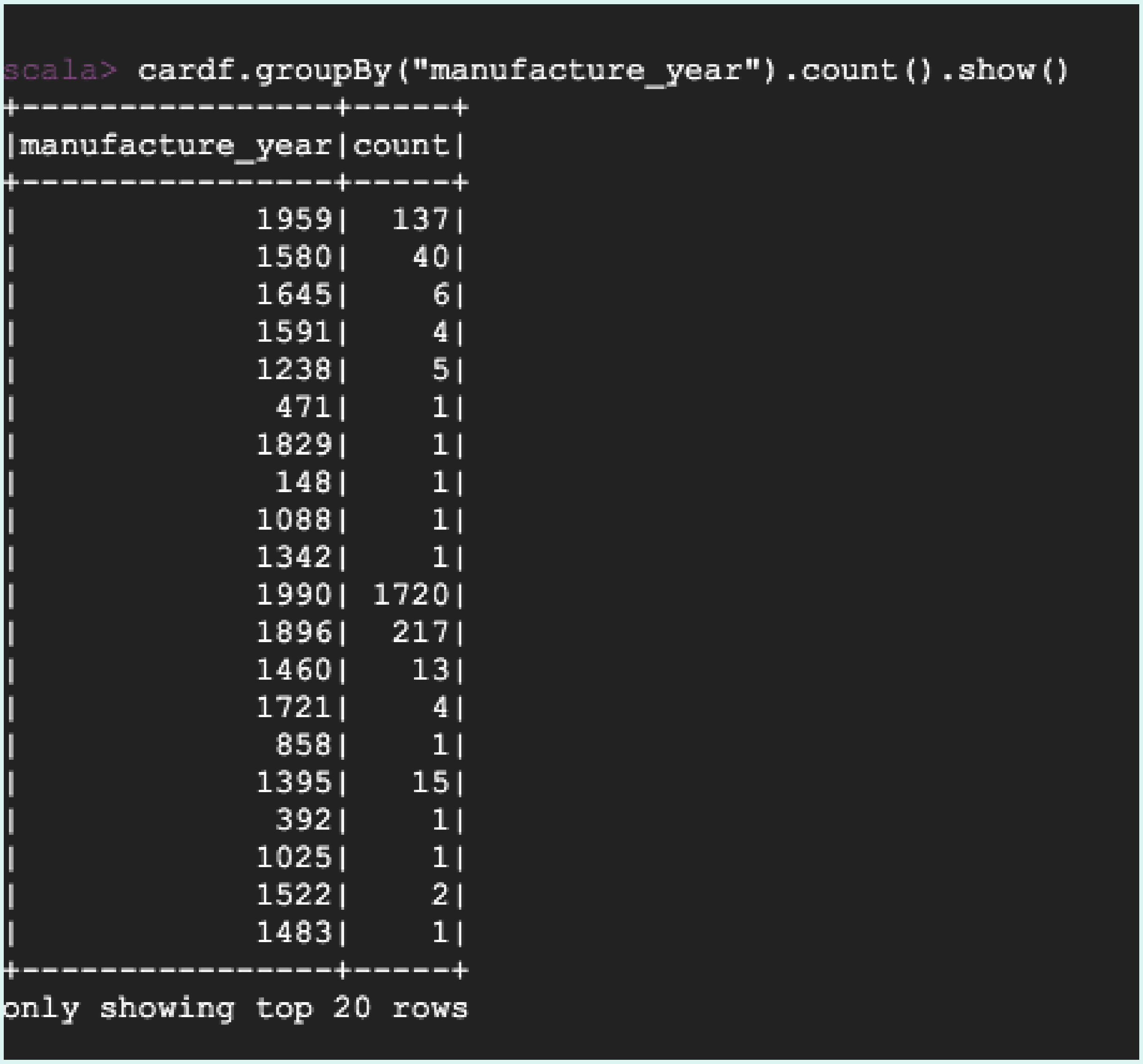
cardf.agg(min("mileage"),max("mileage")).show()



```
scala> cardf.agg(min("mileage"),max("mileage")).show()
+------------+------------+
|min(mileage)|max(mileage)|
+------------+------------+
|         0.0|   9999999.0|
+------------+------------+
```

# Check the range of manufacture year

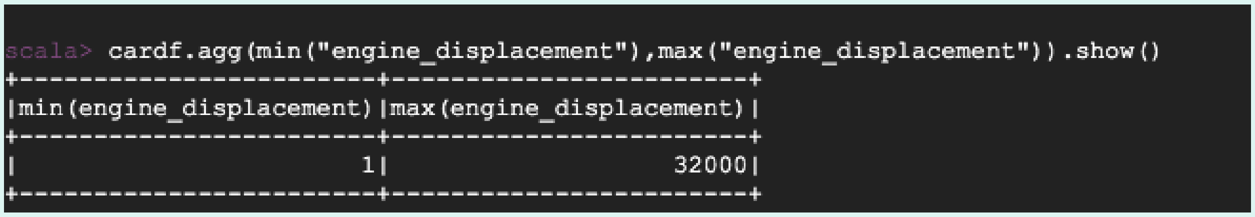
cardf.groupBy("manufacture_year").count().show()



```
scala> cardf.groupBy("manufacture_year").count().show()
+----------------+-----+
|manufacture_year|count|
+----------------+-----+
|            1959|  137|
|            1580|   40|
|            1645|    6|
|            1591|    4|
|            1238|    5|
|             471|    1|
|            1829|    1|
|             148|    1|
|            1088|    1|
|            1342|    1|
|            1990| 1720|
|            1896|  217|
|            1460|   13|
|            1721|    4|
|             858|    1|
|            1395|   15|
|             392|    1|
|            1025|    1|
|            1522|    2|
|            1483|    1|
+----------------+-----+
only showing top 20 rows
```

# Check the range of engine displacement

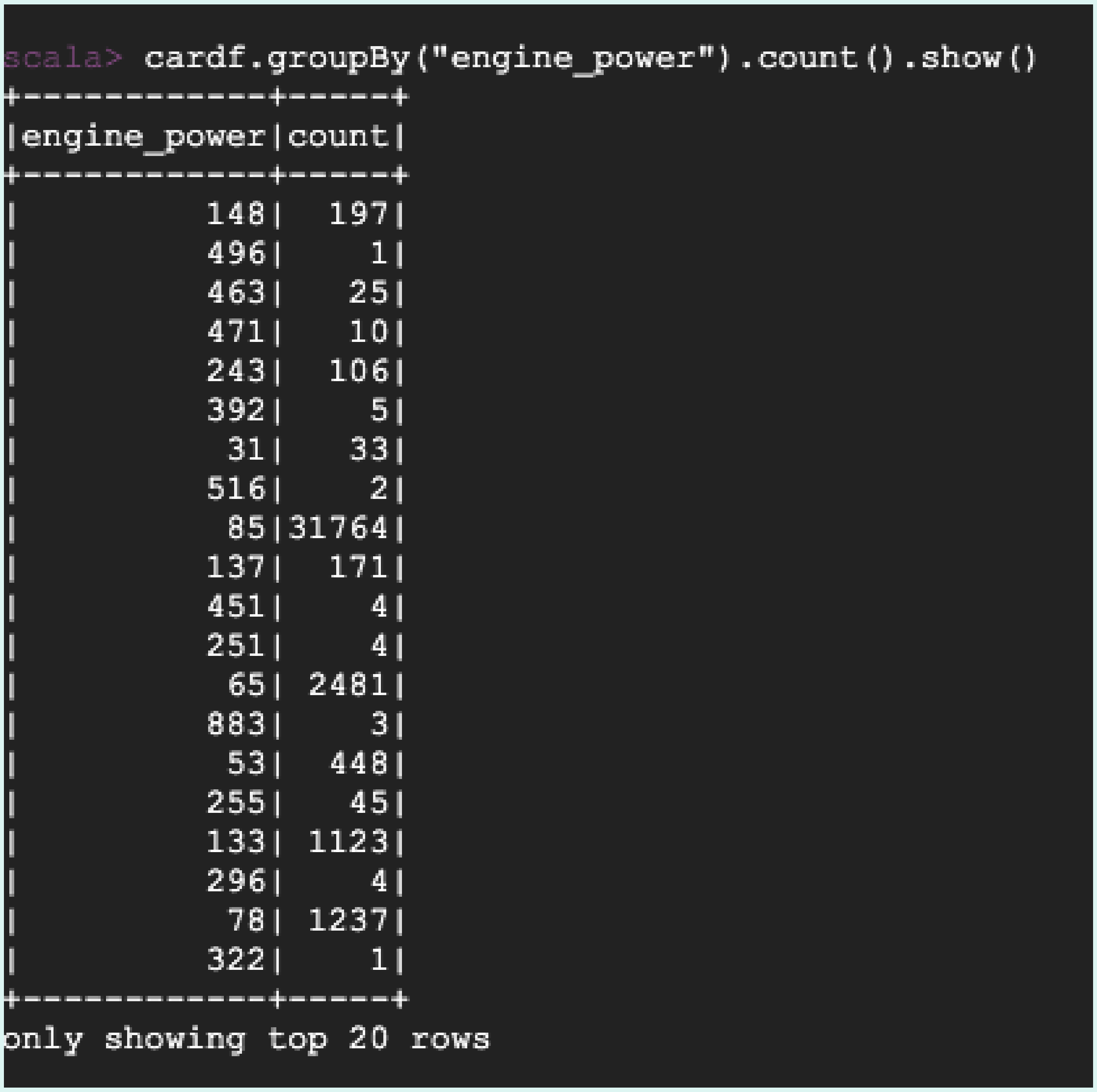
cardf.agg(min("engine_displacement"),max("engine_displacement")).show()



```
scala> cardf.agg(min("engine_displacement"),max("engine_displacement")).show()
+------------------------+------------------------+
|min(engine_displacement)|max(engine_displacement)|
+------------------------+------------------------+
|                       1|                   32000|
+------------------------+------------------------+
```

# Count the range of engine power

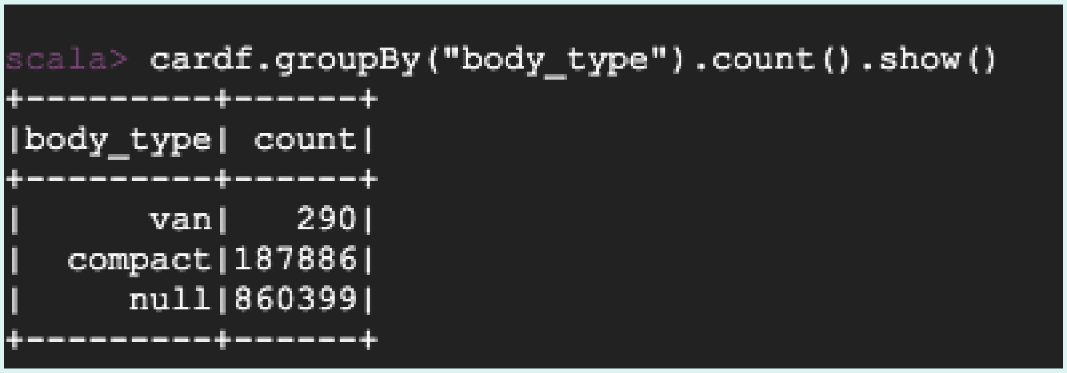
cardf.groupBy("engine_power").count().show()



```
scala> cardf.groupBy("engine_power").count().show()
+------------+-----+
|engine_power|count|
+------------+-----+
|         148|  197|
|         496|    1|
|         463|   25|
|         471|   10|
|         243|  106|
|         392|    5|
|          31|   33|
|         516|    2|
|          85|31764|
|         137|  171|
|         451|    4|
|         251|    4|
|          65| 2481|
|         883|    3|
|          53|  448|
|         255|   45|
|         133| 1123|
|         296|    4|
|          78| 1237|
|         322|    1|
+------------+-----+
only showing top 20 rows
```

# Count the value of body type

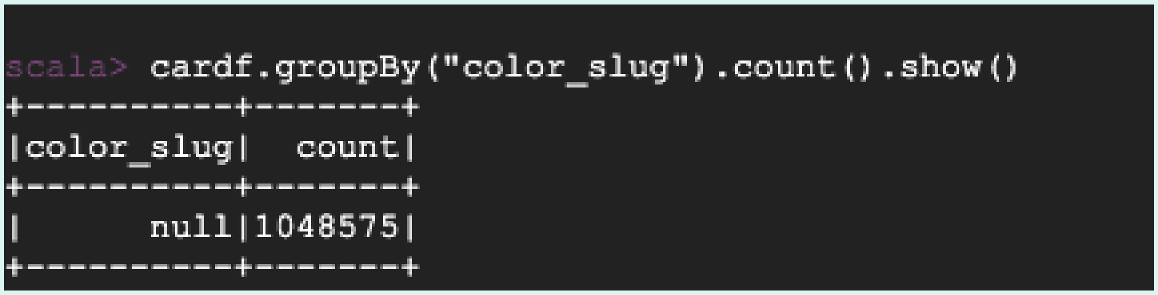
cardf.groupBy("body_type").count().show()



```
scala> cardf.groupBy("body_type").count().show()
+---------+------+
|body_type| count|
+---------+------+
|      van|   290|
|  compact|187886|
|     null|860399|
+---------+------+
```

# Check the color_slug

cardf.groupBy("color_slug").count().show()

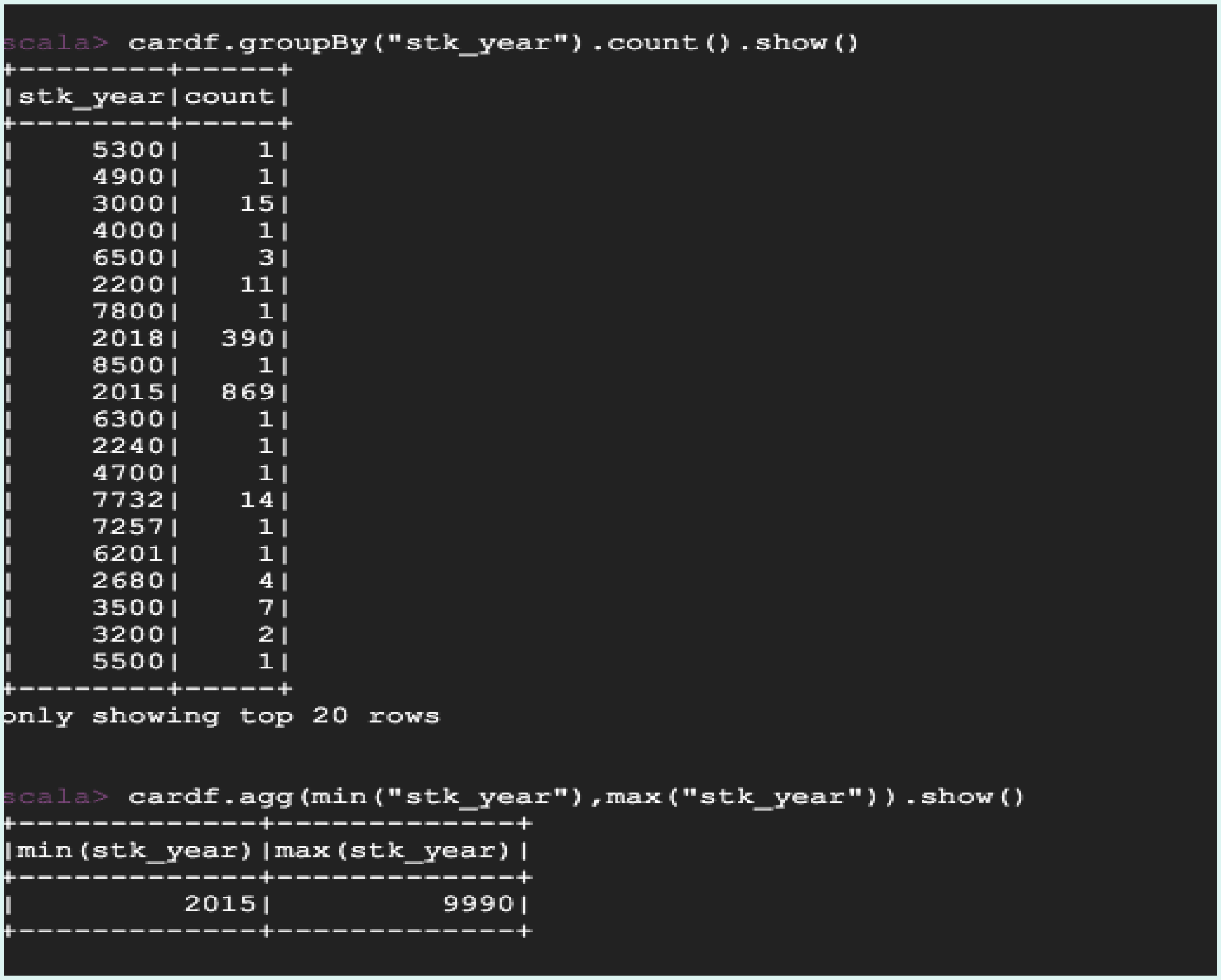


```
scala> cardf.groupBy("color_slug").count().show()
+----------+-------+
|color_slug|  count|
+----------+-------+
|      null|1048575|
+----------+-------+
```

# Check the range of stk_year

cardf.groupBy("stk_year").count().show()

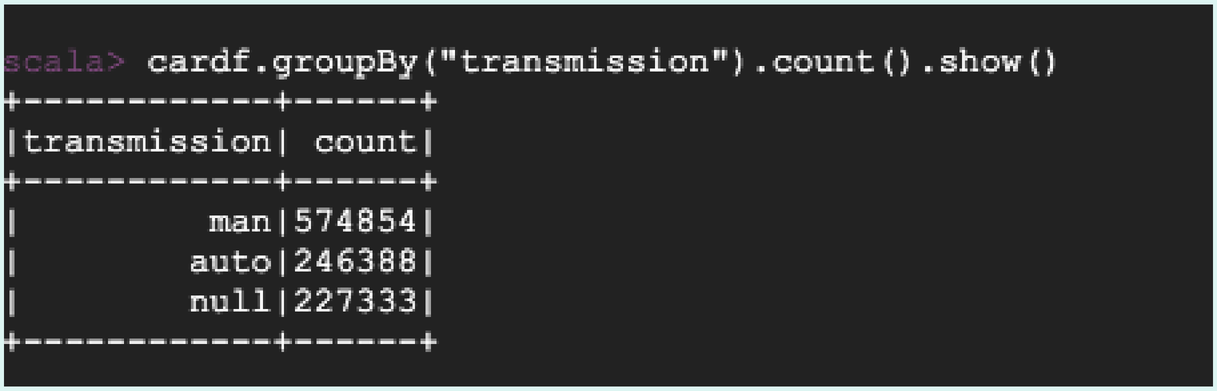
cardf.agg(min("stk_year"),max("stk_year")).show()



```
scala> cardf.groupBy("stk_year").count().show()
+--------+-----+
|stk_year|count|
+--------+-----+
|    5300|    1|
|    4900|    1|
|    3000|   15|
|    4000|    1|
|    6500|    3|
|    2200|   11|
|    7800|    1|
|    2018|  390|
|    8500|    1|
|    2015|  869|
|    6300|    1|
|    2240|    1|
|    4700|    1|
|    7732|   14|
|    7257|    1|
|    6201|    1|
|    2680|    4|
|    3500|    7|
|    3200|    2|
|    5500|    1|
+--------+-----+
only showing top 20 rows

scala> cardf.agg(min("stk_year"),max("stk_year")).show()
+-------------+-------------+
|min(stk_year)|max(stk_year)|
+-------------+-------------+
|         2015|         9990|
+-------------+-------------+
```

# Count the transmission type

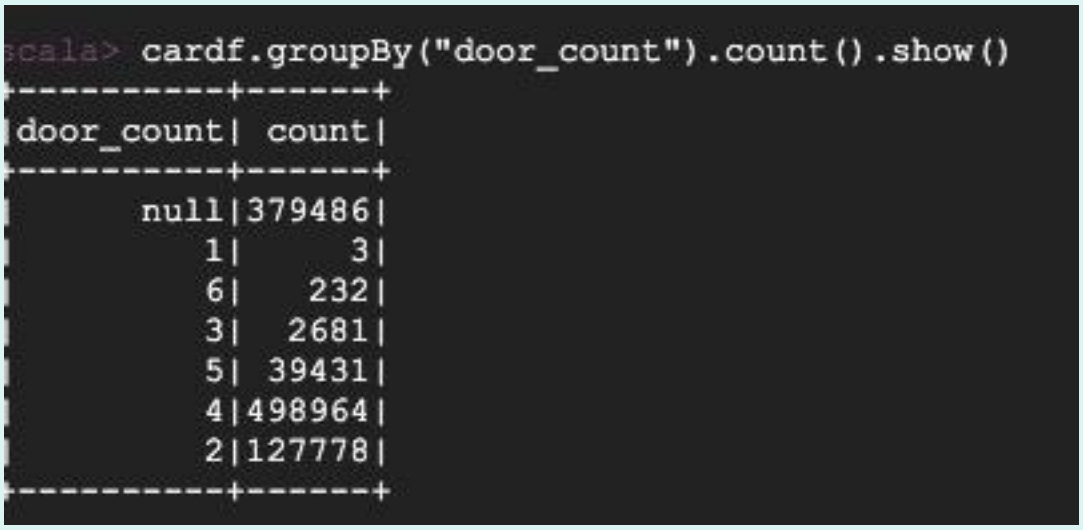
cardf.groupBy("transmission").count().show()



```
scala> cardf.groupBy("transmission").count().show()
+------------+------+
|transmission| count|
+------------+------+
|         man|574854|
|        auto|246388|
|        null|227333|
+------------+------+
```

# Count the number of doors for each car

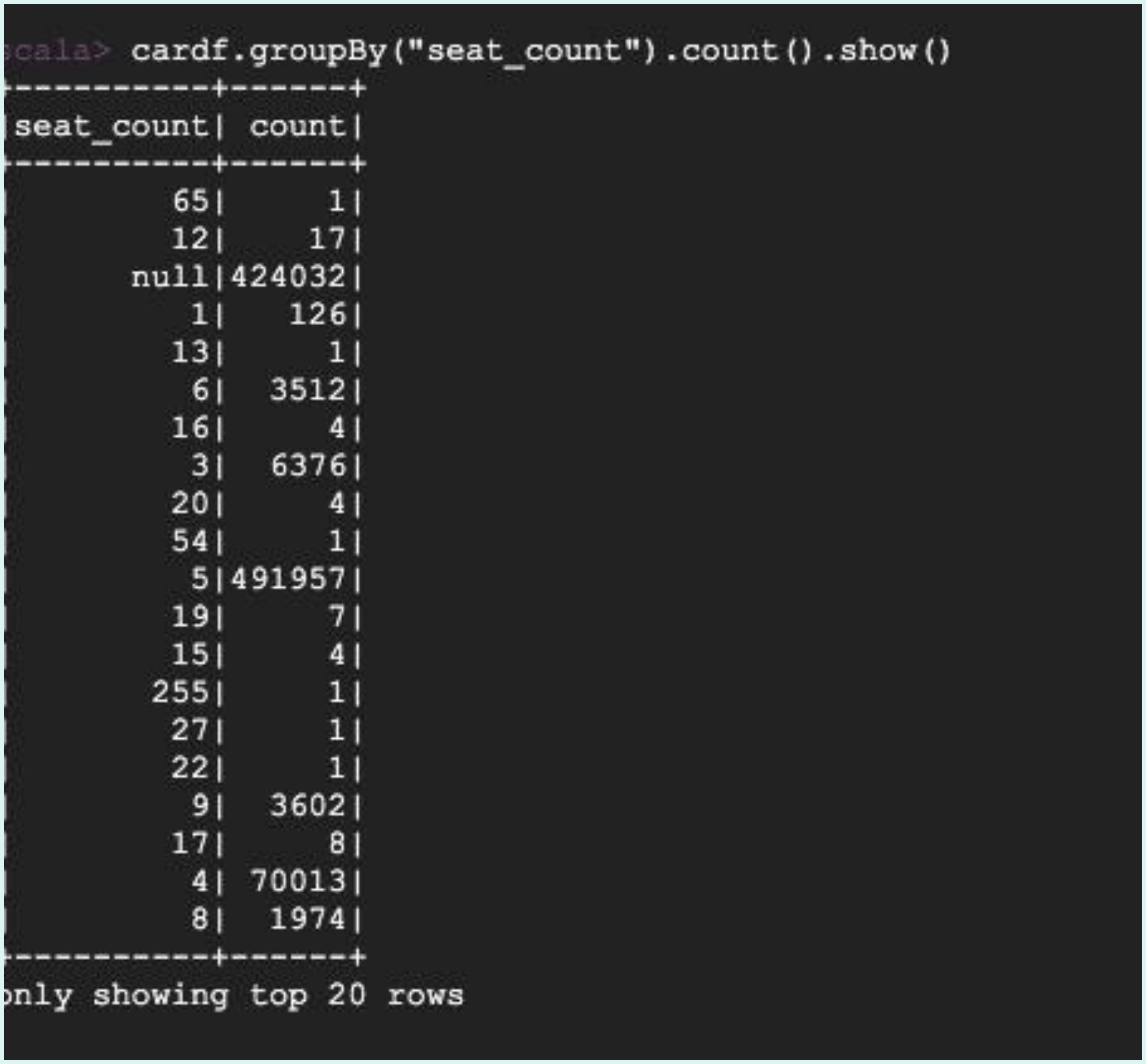
cardf.groupBy("door_count").count().show()



```
scala> cardf.groupBy("door_count").count().show()
+----------+------+
|door_count| count|
+----------+------+
|      null|379486|
|         1|     3|
|         6|   232|
|         3|  2681|
|         5| 39431|
|         4|498964|
|         2|127778|
+----------+------+
```

# Count the number of seats for each car

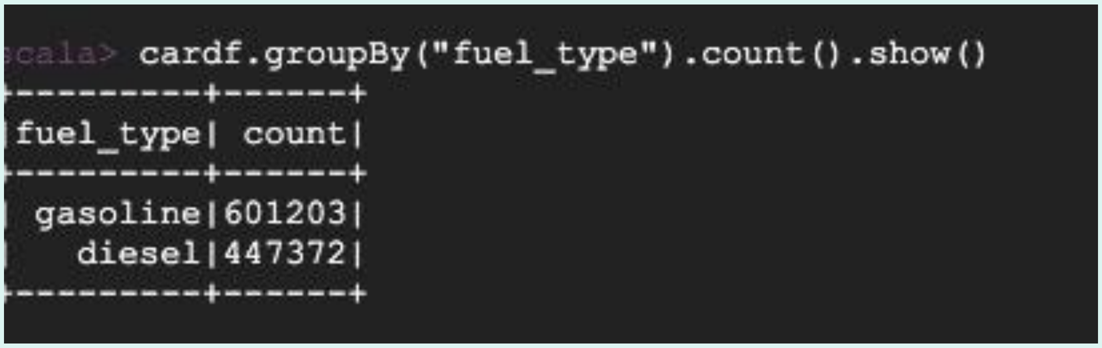
cardf.groupBy("seat_count").count().show()



```
cala> cardf.groupBy("seat_count").count().show()
+----------+------+
|seat_count| count|
+----------+------+
|        65|     1|
|        12|    17|
|      null|424032|
|         1|   126|
|        13|     1|
|         6|  3512|
|        16|     4|
|         3|  6376|
|        20|     4|
|        54|     1|
|         5|491957|
|        19|     7|
|        15|     4|
|       255|     1|
|        27|     1|
|        22|     1|
|         9|  3602|
|        17|     8|
|         4| 70013|
|         8|  1974|
+----------+------+
nly showing top 20 rows
```

# Check the fuel type

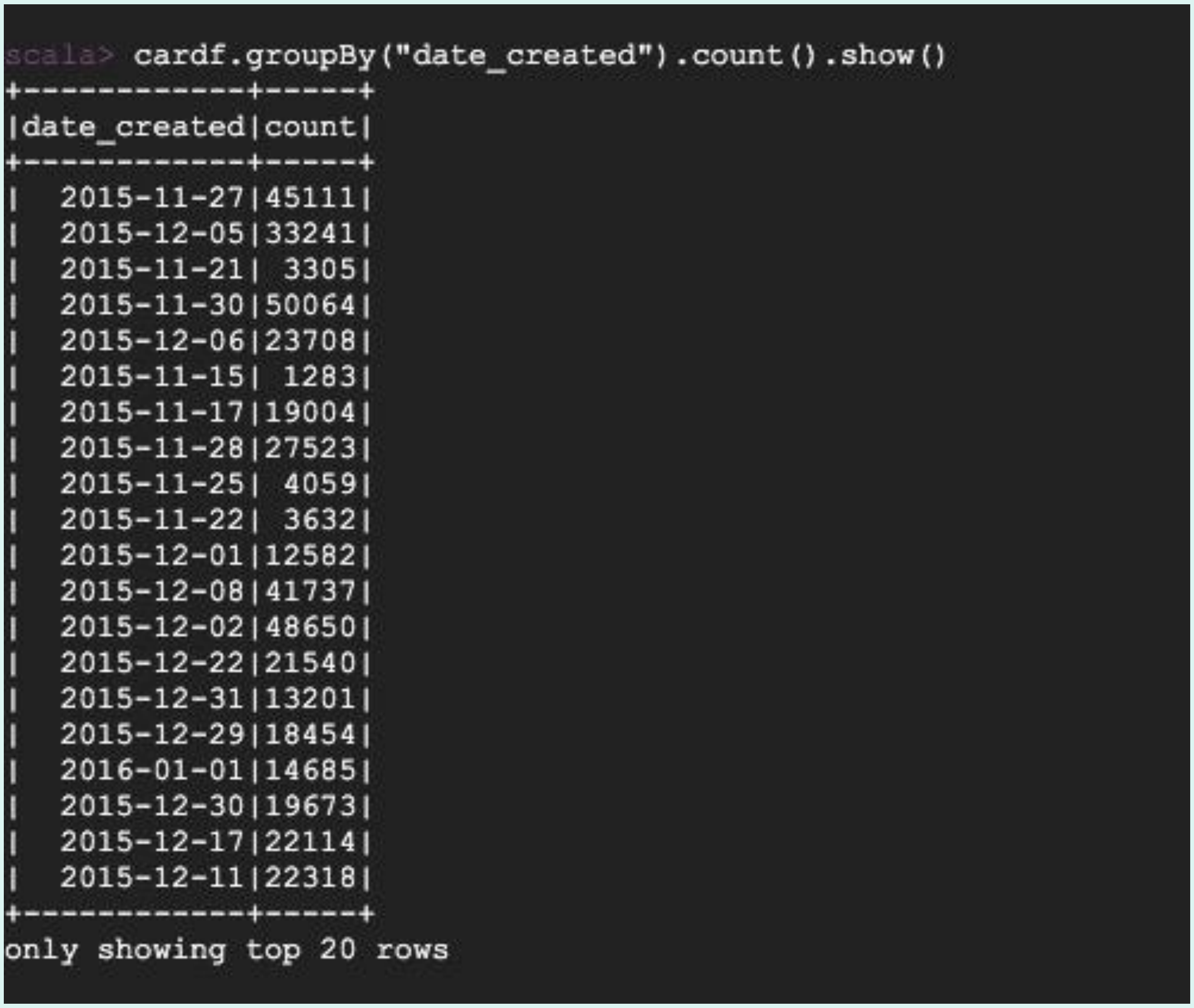
cardf.groupBy("fuel_type").count().show()



```
cala> cardf.groupBy("fuel_type").count().show()
+---------+------+
|fuel_type| count|
+---------+------+
| gasoline|601203|
|   diesel|447372|
+---------+------+
```

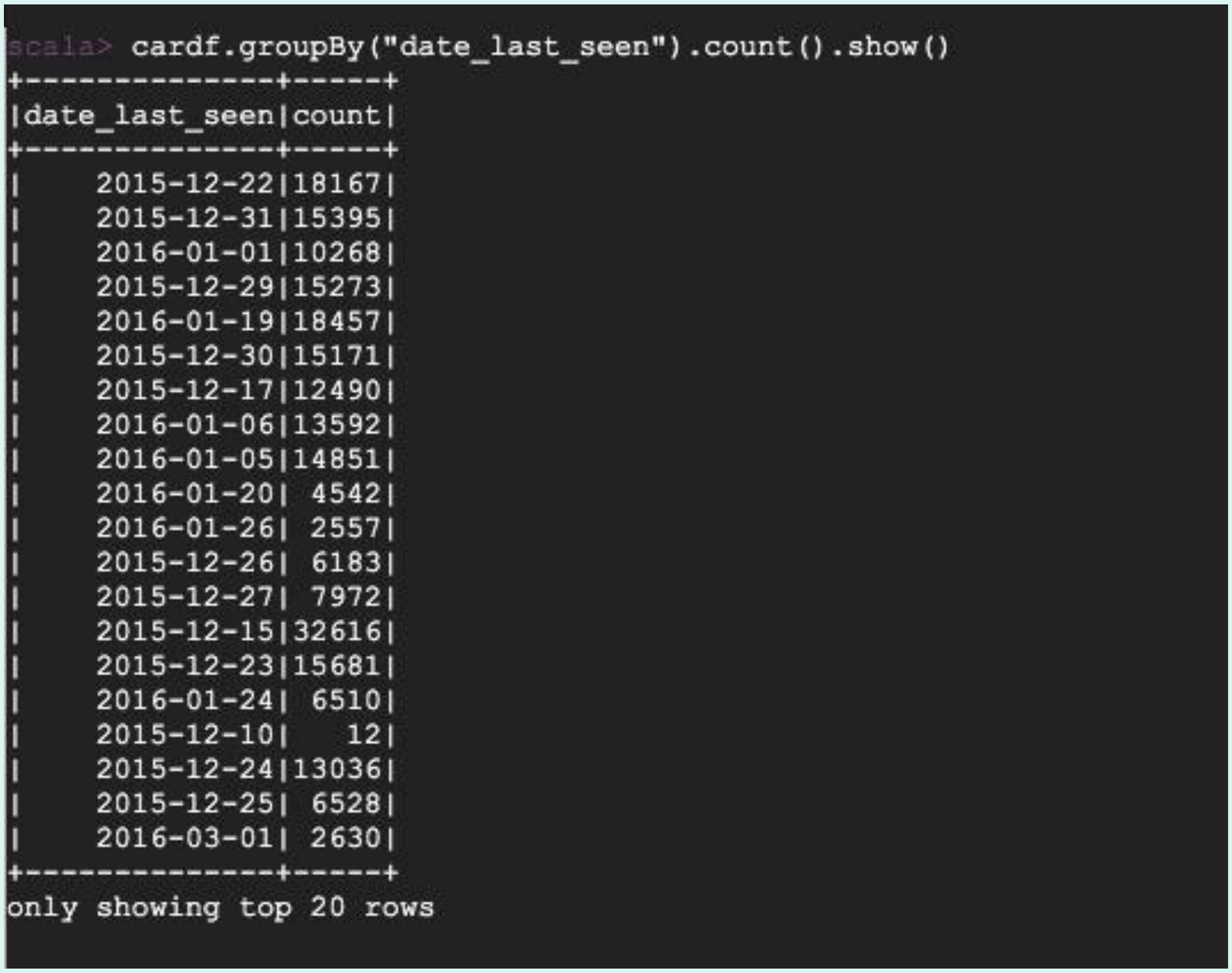
# Review the created date and last-seen date

cardf.groupBy("date_created").count().show()



```
scala> cardf.groupBy("date_created").count().show()
+------------+-----+
|date_created|count|
+------------+-----+
|  2015-11-27|45111|
|  2015-12-05|33241|
|  2015-11-21| 3305|
|  2015-11-30|50064|
|  2015-12-06|23708|
|  2015-11-15| 1283|
|  2015-11-17|19004|
|  2015-11-28|27523|
|  2015-11-25| 4059|
|  2015-11-22| 3632|
|  2015-12-01|12582|
|  2015-12-08|41737|
|  2015-12-02|48650|
|  2015-12-22|21540|
|  2015-12-31|13201|
|  2015-12-29|18454|
|  2016-01-01|14685|
|  2015-12-30|19673|
|  2015-12-17|22114|
|  2015-12-11|22318|
+------------+-----+
only showing top 20 rows
```

cardf.groupBy("date_last_seen").count().show()

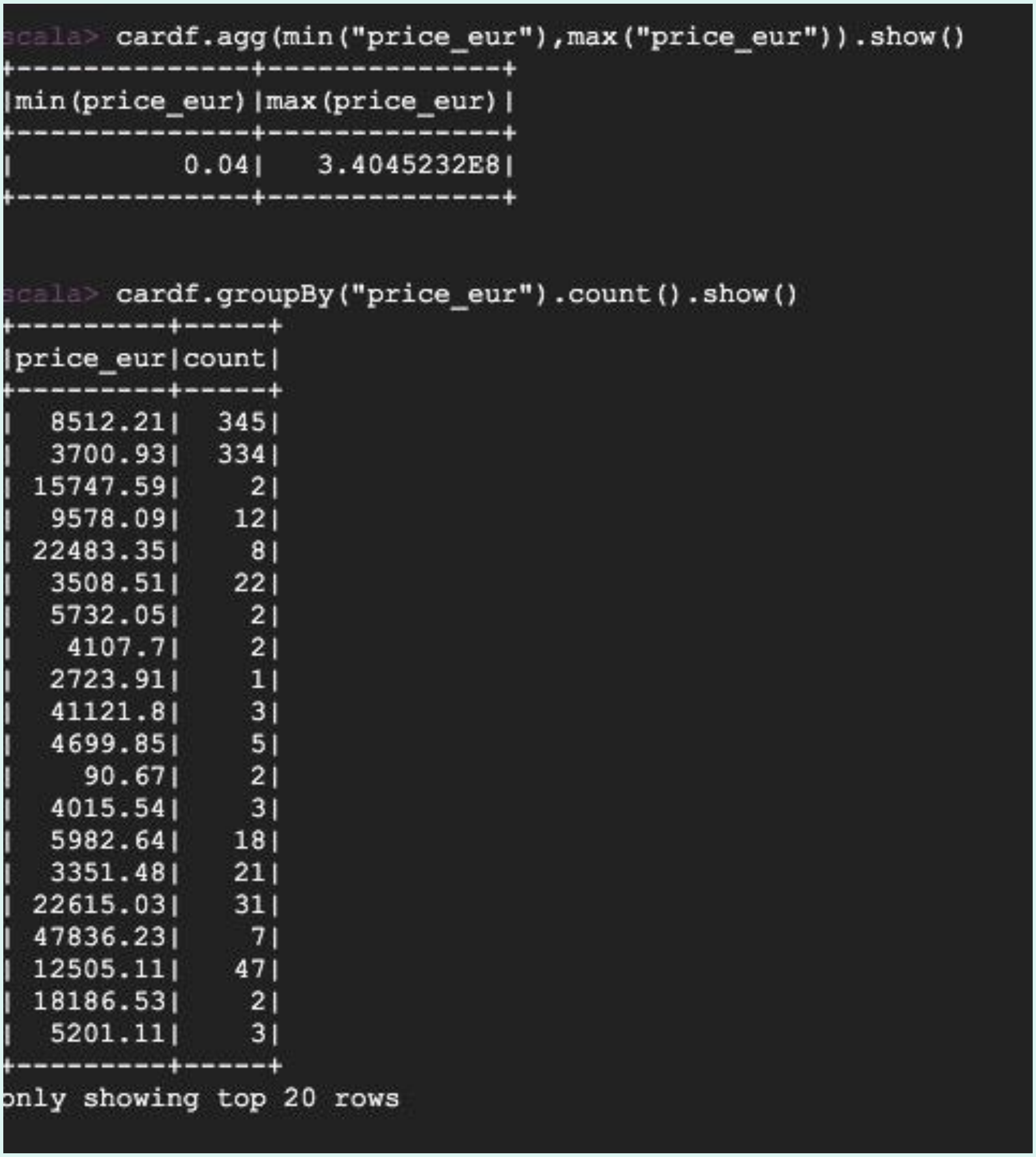


```
scala> cardf.groupBy("date_last_seen").count().show()
+--------------+-----+
|date_last_seen|count|
+--------------+-----+
|    2015-12-22|18167|
|    2015-12-31|15395|
|    2016-01-01|10268|
|    2015-12-29|15273|
|    2016-01-19|18457|
|    2015-12-30|15171|
|    2015-12-17|12490|
|    2016-01-06|13592|
|    2016-01-05|14851|
|    2016-01-20| 4542|
|    2016-01-26| 2557|
|    2015-12-26| 6183|
|    2015-12-27| 7972|
|    2015-12-15|32616|
|    2015-12-23|15681|
|    2016-01-24| 6510|
|    2015-12-10|   12|
|    2015-12-24|13036|
|    2015-12-25| 6528|
|    2016-03-01| 2630|
+--------------+-----+
only showing top 20 rows
```

# Check the range of price

cardf.agg(min("price_eur"),max("price_eur")).show()

cardf.groupBy("price_eur").count().show()



```
scala> cardf.agg(min("price_eur"),max("price_eur")).show()
+--------------+--------------+
|min(price_eur)|max(price_eur)|
+--------------+--------------+
|          0.04|   3.4045232E8|
+--------------+--------------+


scala> cardf.groupBy("price_eur").count().show()
+---------+-----+
|price_eur|count|
+---------+-----+
|  8512.21|  345|
|  3700.93|  334|
| 15747.59|    2|
|  9578.09|   12|
| 22483.35|    8|
|  3508.51|   22|
|  5732.05|    2|
|   4107.7|    2|
|  2723.91|    1|
|  41121.8|    3|
|  4699.85|    5|
|    90.67|    2|
|  4015.54|    3|
|  5982.64|   18|
|  3351.48|   21|
| 22615.03|   31|
| 47836.23|    7|
| 12505.11|   47|
| 18186.53|    2|
|  5201.11|    3|
+---------+-----+
only showing top 20 rows
```

Preliminary Analytics based on above data cleaning:

- More than 300,000 data that have no maker or model which need to be removed
- Several figures, such as mileage, manufacture year, seat count, contain unreasonable data based on the research of the car industry

- There are huge amount of null value or unnecessary value in body type, color slug, stk_year, which are not the key factors for this business case
- There are some inaccurate corresponding relationship, such are the luxury car maker/model related to the very low price

Create the reasonable dataset after data cleaning:

- Remove maker & model columns with null value
- Select mileage range from 12,000 to 200,000 miles because with around 12,000 the most common yardstick per year (most leases allow 12,000 miles per year) and the maximum is 200,000 miles to perform reliably (cars.com, June 2020)
- Vehicle years：15 years backwards from 2021, that is, greater than or equal to 2006, were selected to exclude obvious typing errors or obsolescence cars
- Delete color_slug variable because all are null value
- Delete stk_year, body type since these figures is not closely related to this business analysis
- The prices of the car are selected in the reasonable range of 4000 euro to 30000 euro
- Other variables remain the same
- There are 12 columns and 190,508 rows after cleaning

Exploratory Data Analysis

Create car2 table and data frame:

Count	Variable Name	Type	Description
1	maker	String	car brand
2	model	String	specific car product name
3	mileage	Float	overall distance - in KM
4	manufacture_year	Integer	year of production
5	engine_power	Integer	engine_power - in kW
6	transmission	String	automatic or manual

7	door_count	Integer	NO. of doors
8	seat_count	Integer	NO. of seats
9	fuel_type	String	gasoline, diesel, cng, lpg, electric
10	date_created	Date	when the ad was scraped
11	date_last_seen	Date	when the ad was last seen. The policy was to remove all ads older than 60 days
12	price_eur	Float	list price converted to EUR

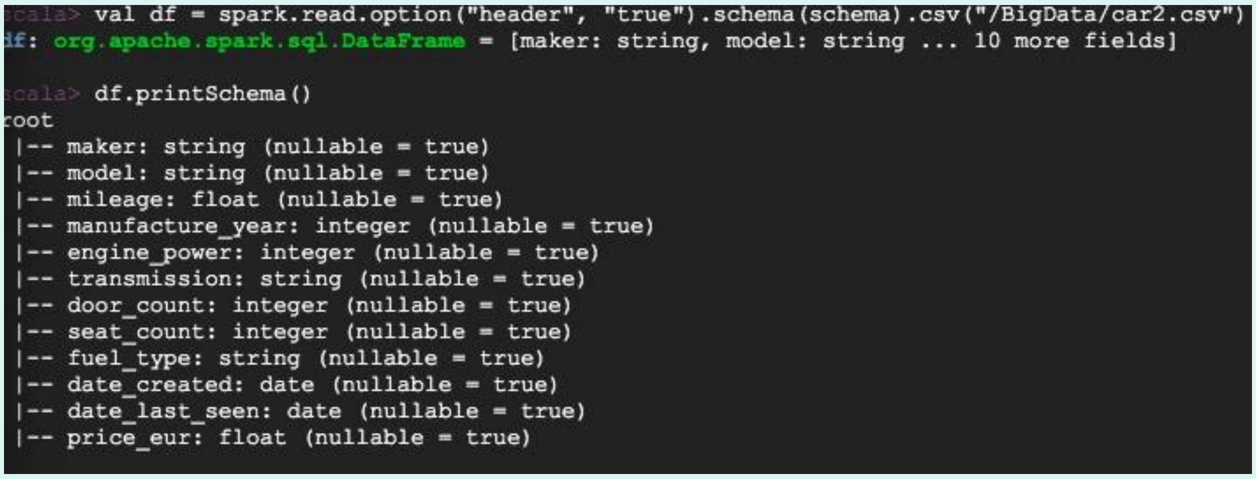
#Create the schema

```
val schema =  StructType(Array(StructField("maker", StringType,true),
StructField("model", StringType,true), StructField("mileage", FloatType,true),
StructField("manufacture_year", IntegerType,true), StructField("engine_power",
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)))
```

#Load the dataset

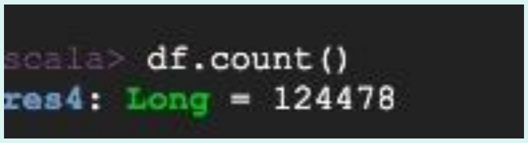
```
val df = spark.read.option("header", "true").schema(schema).csv("/BigData/car2.csv")
```

```
df.printSchema()
```



```
cala> val df = spark.read.option("header", "true").schema(schema).csv("/BigData/car2.csv")
df: org.apache.spark.sql.DataFrame = [maker: string, model: string ... 10 more fields]

cala> df.printSchema()
root
 |-- maker: string (nullable = true)
 |-- model: string (nullable = true)
 |-- mileage: float (nullable = true)
 |-- manufacture_year: integer (nullable = true)
 |-- engine_power: integer (nullable = true)
 |-- transmission: string (nullable = true)
 |-- door_count: integer (nullable = true)
 |-- seat_count: integer (nullable = true)
 |-- fuel_type: string (nullable = true)
 |-- date_created: date (nullable = true)
 |-- date_last_seen: date (nullable = true)
 |-- price_eur: float (nullable = true)
```

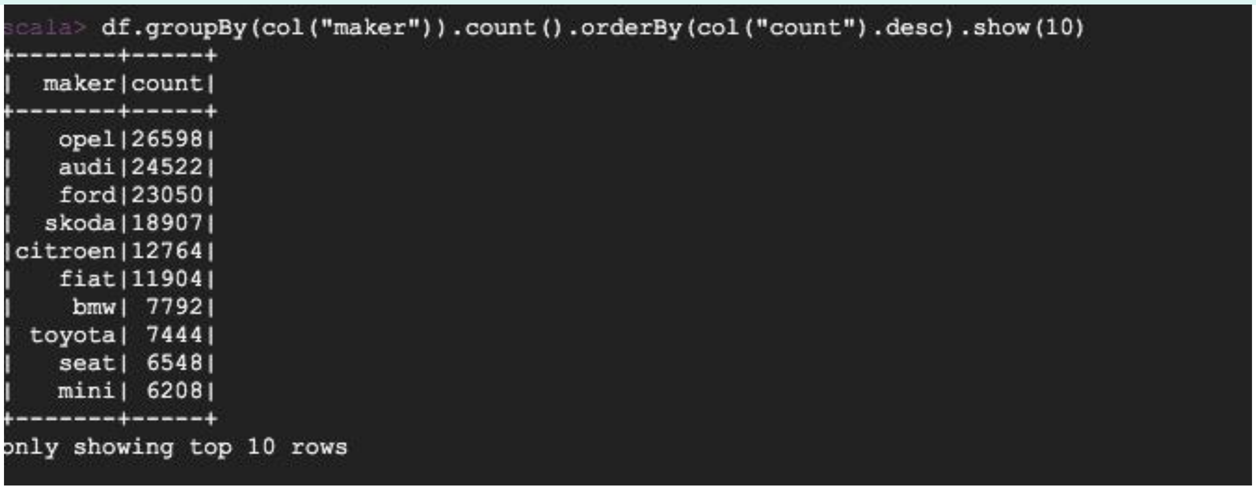
df.count()



```
scala> df.count()
res4: Long = 124478
```

# Check top10 popular makers

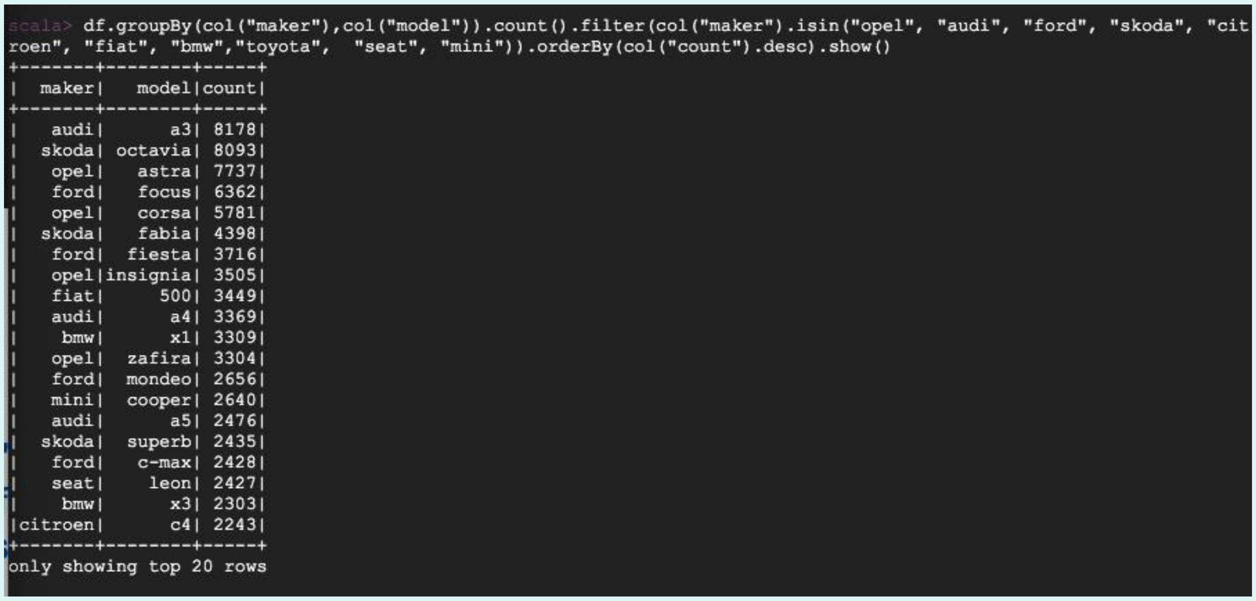
df.groupBy(col("maker")).count().orderBy(col("count").desc).show(10)



```
cala> df.groupBy(col("maker")).count().orderBy(col("count").desc).show(10)
+-------+-----+
|  maker|count|
+-------+-----+
|   opel|26598|
|   audi|24522|
|   ford|23050|
|  skoda|18907|
|citroen|12764|
|   fiat|11904|
|    bmw| 7792|
| toyota| 7444|
|   seat| 6548|
|   mini| 6208|
+-------+-----+
only showing top 10 rows
```

# Identify the popular models based on makers

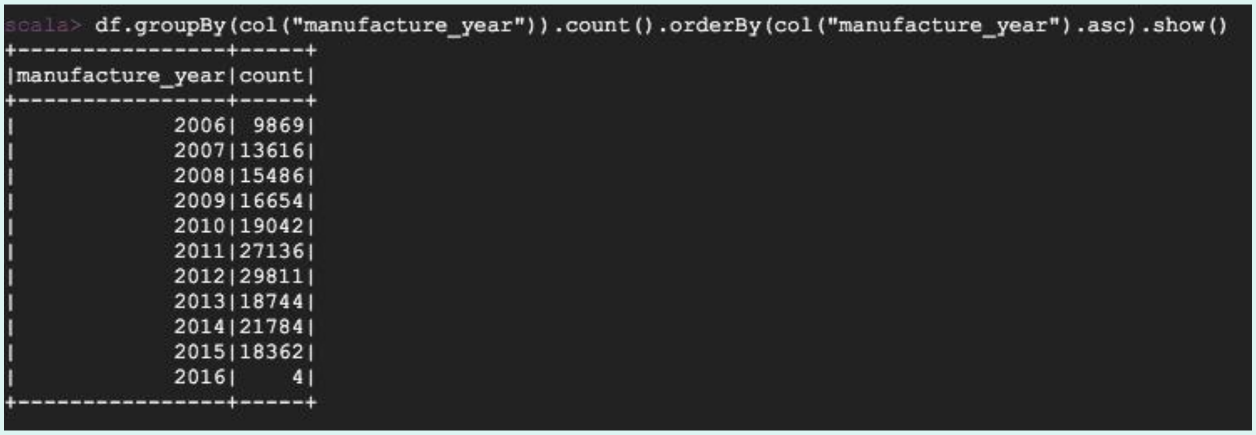
df.groupBy(col("maker"),col("model")).count().filter(col("maker").isin("opel", "audi", "ford", "skoda", "citroen", "fiat", "bmw","toyota", "seat", "mini")).orderBy(col("count").desc).show()



```
scala> df.groupBy(col("maker"),col("model")).count().filter(col("maker").isin("opel", "audi", "ford", "skoda", "cit
roen", "fiat", "bmw","toyota",  "seat", "mini")).orderBy(col("count").desc).show()
+-------+--------+-----+
|  maker|   model|count|
+-------+--------+-----+
|   audi|      a3| 8178|
|  skoda| octavia| 8093|
|   opel|   astra| 7737|
|   ford|   focus| 6362|
|   opel|   corsa| 5781|
|  skoda|   fabia| 4398|
|   ford|  fiesta| 3716|
|   opel|insignia| 3505|
|   fiat|     500| 3449|
|   audi|      a4| 3369|
|    bmw|      x1| 3309|
|   opel|  zafira| 3304|
|   ford|  mondeo| 2656|
|   mini|  cooper| 2640|
|   audi|      a5| 2476|
|  skoda|  superb| 2435|
|   ford|   c-max| 2428|
|   seat|    leon| 2427|
|    bmw|      x3| 2303|
|citroen|      c4| 2243|
+-------+--------+-----+
only showing top 20 rows
```

# Count the distribution of each manufacture year

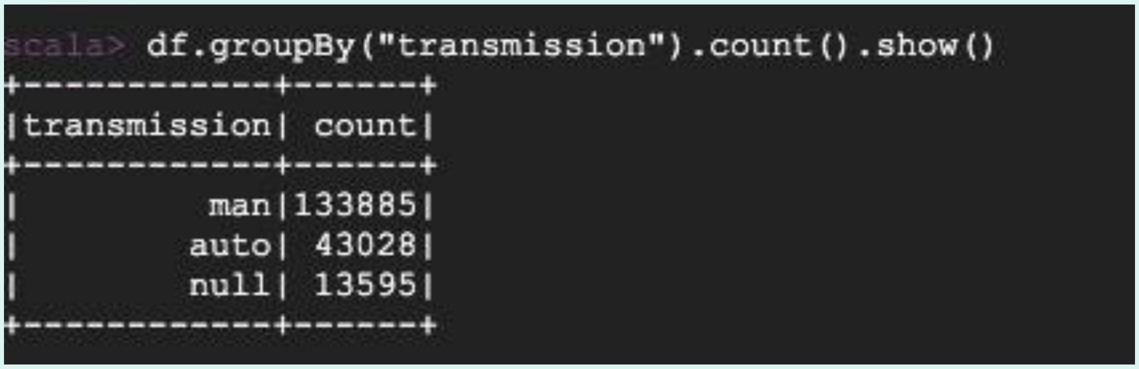
df.groupBy(col("manufacture_year")).count().orderBy(col("manufacture_year").asc).show()



```
scala> df.groupBy(col("manufacture_year")).count().orderBy(col("manufacture_year").asc).show()
+----------------+-----+
|manufacture_year|count|
+----------------+-----+
|            2006| 9869|
|            2007|13616|
|            2008|15486|
|            2009|16654|
|            2010|19042|
|            2011|27136|
|            2012|29811|
|            2013|18744|
|            2014|21784|
|            2015|18362|
|            2016|    4|
+----------------+-----+
```

# Count the transmission type

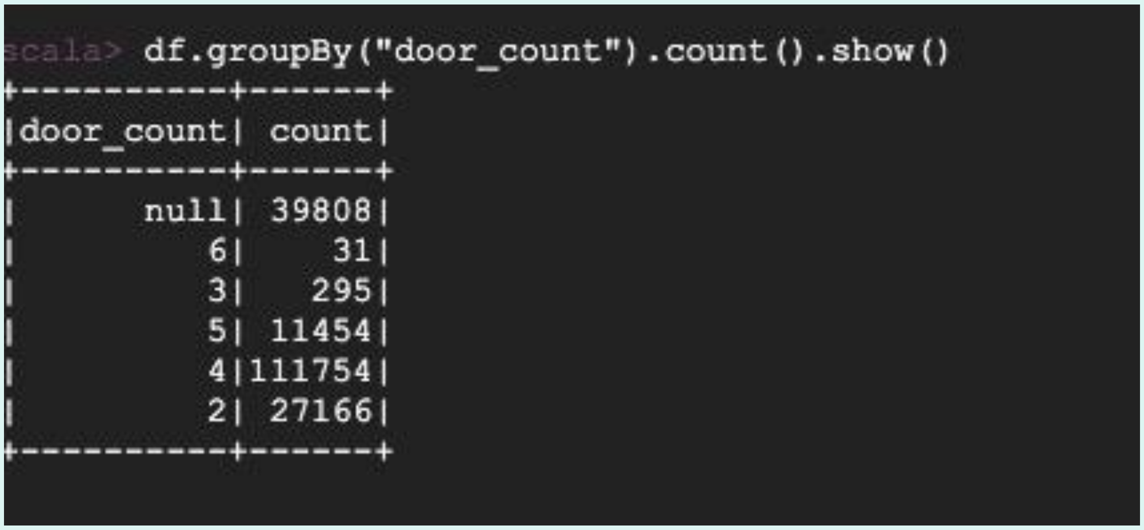
df.groupBy("transmission").count().show()



```
scala> df.groupBy("transmission").count().show()
+------------+------+
|transmission| count|
+------------+------+
|         man|133885|
|        auto| 43028|
|        null| 13595|
+------------+------+
```

# Count the number of doors for each car

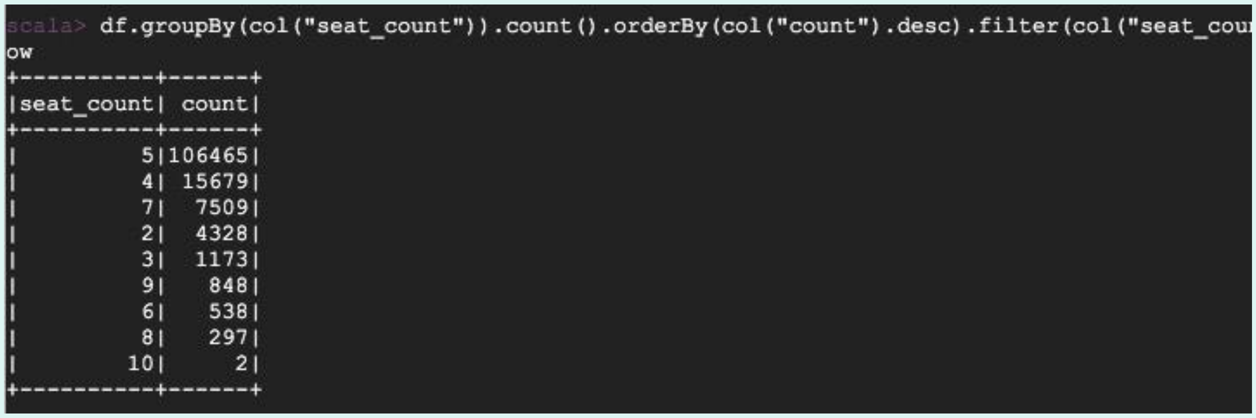
df.groupBy("door_count").count().show()



```
scala> df.groupBy("door_count").count().show()
+----------+------+
|door_count| count|
+----------+------+
|      null| 39808|
|         6|    31|
|         3|   295|
|         5| 11454|
|         4|111754|
|         2| 27166|
+----------+------+
```

# Count the number of seats for each car, min=2. max=10

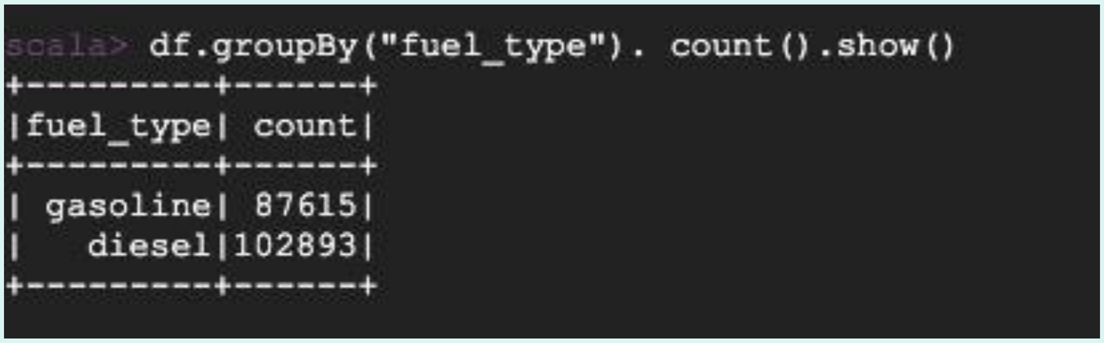
df.groupBy(col("seat_count")).count().orderBy(col("count").desc).filter(col("seat_count") . between(2,10)).show



```
scala> df.groupBy(col("seat_count")).count().orderBy(col("count").desc).filter(col("seat_cou
ow
+----------+------+
|seat_count| count|
+----------+------+
|         5|106465|
|         4| 15679|
|         7|  7509|
|         2|  4328|
|         3|  1173|
|         9|   848|
|         6|   538|
|         8|   297|
|        10|     2|
+----------+------+
```

# Check the fuel type

df.groupBy("fuel_type"). count().show()

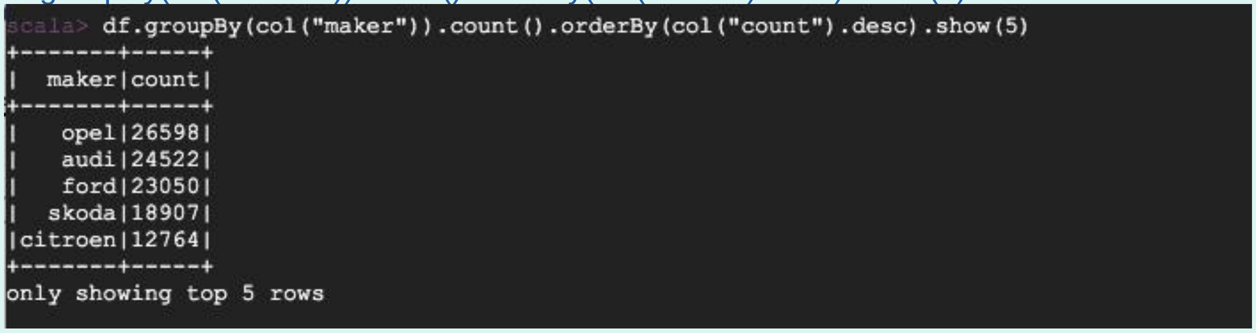


```
scala> df.groupBy("fuel_type"). count().show()
+---------+------+
|fuel_type| count|
+---------+------+
| gasoline| 87615|
|   diesel|102893|
+---------+------+
```

Based on the key business questions, narrow down the dataset to Top 5:

# Check top5 popular makers

df.groupBy(col("maker")).count().orderBy(col("count").desc).show(5)

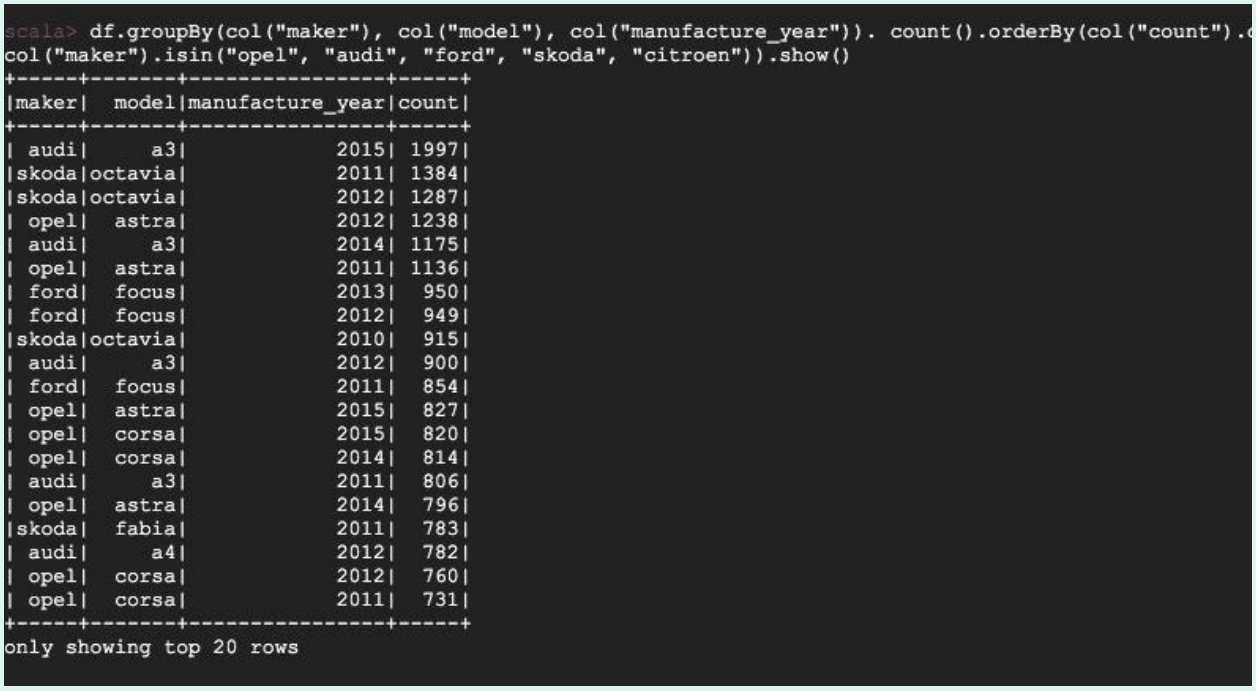


```
scala> df.groupBy(col("maker")).count().orderBy(col("count").desc).show(5)
+-------+-----+
|  maker|count|
+-------+-----+
|   opel|26598|
|   audi|24522|
|   ford|23050|
|  skoda|18907|
|citroen|12764|
+-------+-----+
only showing top 5 rows
```

# Check the most common manufacture year for top5 brands

df.groupBy(col("maker"), col("model"), col("manufacture_year")).

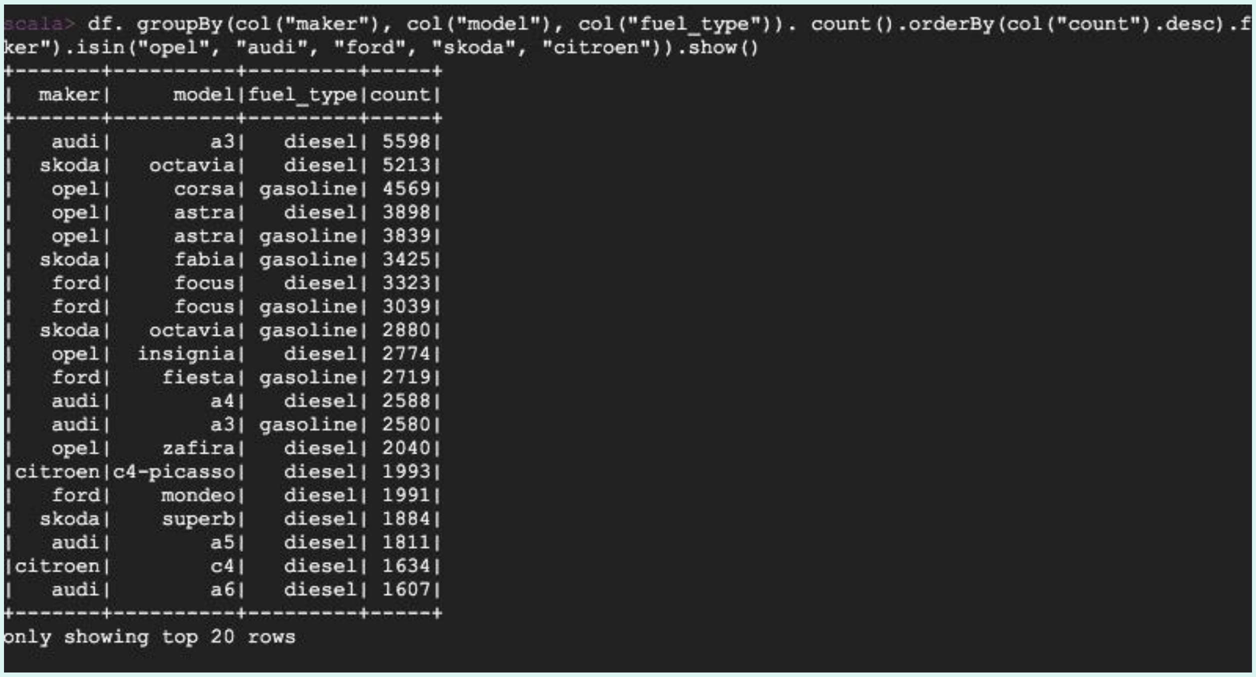
count().orderBy(col("count").desc). filter(col("maker").isin("opel", "audi", "ford", "skoda", "citroen")).show()



```
scala> df.groupBy(col("maker"), col("model"), col("manufacture_year")). count().orderBy(col("count").d
col("maker").isin("opel", "audi", "ford", "skoda", "citroen")).show()
+-----+-------+----------------+-----+
|maker|  model|manufacture_year|count|
+-----+-------+----------------+-----+
| audi|     a3|            2015| 1997|
|skoda|octavia|            2011| 1384|
|skoda|octavia|            2012| 1287|
| opel|  astra|            2012| 1238|
| audi|     a3|            2014| 1175|
| opel|  astra|            2011| 1136|
| ford|  focus|            2013|  950|
| ford|  focus|            2012|  949|
|skoda|octavia|            2010|  915|
| audi|     a3|            2012|  900|
| ford|  focus|            2011|  854|
| opel|  astra|            2015|  827|
| opel|  corsa|            2015|  820|
| opel|  corsa|            2014|  814|
| audi|     a3|            2011|  806|
| opel|  astra|            2014|  796|
|skoda|  fabia|            2011|  783|
| audi|     a4|            2012|  782|
| opel|  corsa|            2012|  760|
| opel|  corsa|            2011|  731|
+-----+-------+----------------+-----+
only showing top 20 rows
```

# Check the different fuel types for top5 brands

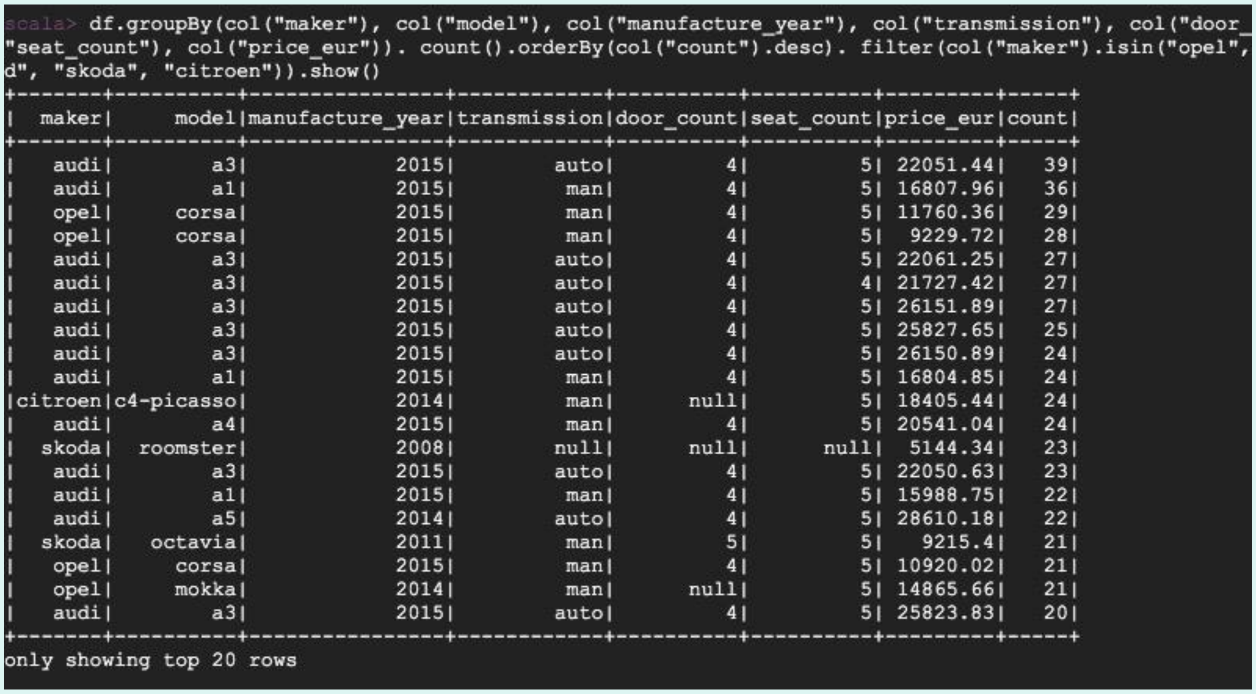
df. groupBy(col("maker"), col("model"), col("fuel_type")). count().orderBy(col("count").desc).filter(col("maker").isin("opel", "audi", "ford", "skoda", "citroen")).show()



```
scala> df. groupBy(col("maker"), col("model"), col("fuel_type")). count().orderBy(col("count").desc).f
ker").isin("opel", "audi", "ford", "skoda", "citroen")).show()
+-------+----------+---------+-----+
|  maker|     model|fuel_type|count|
+-------+----------+---------+-----+
|   audi|        a3|   diesel| 5598|
|  skoda|   octavia|   diesel| 5213|
|   opel|     corsa| gasoline| 4569|
|   opel|     astra|   diesel| 3898|
|   opel|     astra| gasoline| 3839|
|  skoda|     fabia| gasoline| 3425|
|   ford|     focus|   diesel| 3323|
|   ford|     focus| gasoline| 3039|
|  skoda|   octavia| gasoline| 2880|
|   opel|  insignia|   diesel| 2774|
|   ford|    fiesta| gasoline| 2719|
|   audi|        a4|   diesel| 2588|
|   audi|        a3| gasoline| 2580|
|   opel|    zafira|   diesel| 2040|
|citroen|c4-picasso|   diesel| 1993|
|   ford|    mondeo|   diesel| 1991|
|  skoda|    superb|   diesel| 1884|
|   audi|        a5|   diesel| 1811|
|citroen|        c4|   diesel| 1634|
|   audi|        a6|   diesel| 1607|
+-------+----------+---------+-----+
only showing top 20 rows
```

# Figure out the relationship among makes, models and price (top 5 brands)

```
df.groupBy(col("maker"), col("model"), col("manufacture_year"), col("transmission"),
col("door_count"), col("seat_count"), col("price_eur")).
count().orderBy(col("count").desc). filter(col("maker").isin("opel", "audi", "ford", "skoda",
"citroen")).show()
```



```
scala> df.groupBy(col("maker"), col("model"), col("manufacture_year"), col("transmission"), col("door_
"seat_count"), col("price_eur")). count().orderBy(col("count").desc). filter(col("maker").isin("opel",
d", "skoda", "citroen")).show()
+-------+----------+----------------+------------+----------+----------+---------+-----+
|  maker|     model|manufacture_year|transmission|door_count|seat_count|price_eur|count|
+-------+----------+----------------+------------+----------+----------+---------+-----+
|   audi|        a3|            2015|        auto|         4|         5| 22051.44|   39|
|   audi|        a1|            2015|         man|         4|         5| 16807.96|   36|
|   opel|     corsa|            2015|         man|         4|         5| 11760.36|   29|
|   opel|     corsa|            2015|         man|         4|         5|  9229.72|   28|
|   audi|        a3|            2015|        auto|         4|         5| 22061.25|   27|
|   audi|        a3|            2015|        auto|         4|         4| 21727.42|   27|
|   audi|        a3|            2015|        auto|         4|         5| 26151.89|   27|
|   audi|        a3|            2015|        auto|         4|         5| 25827.65|   25|
|   audi|        a3|            2015|        auto|         4|         5| 26150.89|   24|
|   audi|        a1|            2015|         man|         4|         5| 16804.85|   24|
|citroen|c4-picasso|            2014|         man|      null|         5| 18405.44|   24|
|   audi|        a4|            2015|         man|         4|         5| 20541.04|   24|
|  skoda|  roomster|            2008|        null|      null|      null|  5144.34|   23|
|   audi|        a3|            2015|        auto|         4|         5| 22050.63|   23|
|   audi|        a1|            2015|         man|         4|         5| 15988.75|   22|
|   audi|        a5|            2014|        auto|         4|         5| 28610.18|   22|
|  skoda|   octavia|            2011|         man|         5|         5|   9215.4|   21|
|   opel|     corsa|            2015|         man|         4|         5| 10920.02|   21|
|   opel|     mokka|            2014|         man|      null|         5| 14865.66|   21|
|   audi|        a3|            2015|        auto|         4|         5| 25823.83|   20|
+-------+----------+----------------+------------+----------+----------+---------+-----+
only showing top 20 rows
```

- The code and screenshots shown above indicate the data source and analyze process, which provide the basis and authenticity for the charts and text descriptions in the “Data Analysis Result” and “Conclusion & Recommendation” parts
- The visualization charts and graphs in main body are generated by Power BI

Citation

Dataset: Classified Ads for Cars- Used cars for sale in Germany and Czech Republic since 2015 https://www.kaggle.com/mirosval/personal-cars-classifieds

Car.com (June 2020) How Many Miles Is Too Many for a Used Car?

https://www.cars.com/articles/how-many-miles-is-too-many-for-a-used-car-2-422606/