Data Analytics with Hive

11.07.2021

Ronak Kumar
8740799

Introduction

Apache Hive is a data warehouse management tool for processing structured data stored in Hadoop. Hive provides a query language HiveQL (similar to SQL) that enables querying HDFS data easily for beginners and prevents the user's need to write MapReduce programs. It is usually used by analysts and non-programmers to efficiently extract Big Data stored in the HDFS where normal processing tools like Excel fail.

The purpose of this assignment is to provide data driven advice to the stakeholders, that will enable them to make a sound investment decision using the cars.csv dataset on Kaggle. The dataset has the classified records for several Eastern European countries over several years. This has been chosen since the veracity of the dataset was established before.

Data Cleaning

The dataset roughly involves 3.5 million rows and contains 16 columns which are mentioned below :-

- 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

For analysis purposes, most of the columns weren't relevant as they either contain a lot of NULL values or wouldn't help in providing data driven advice to the stakeholders.

Columns like door_count, seat_cound, fuel_type, transmission, color_slug, stk_year, body_type, engine_power and engine_displacement shouldn't affect the sales of the car and so weren't taken into consideration. About 94% of the records in the color_slug field were NULL's.

After completely researching the dataset, the columns chosen were maker, model, mileage, manufacture_year, date_created, date_last_seen, price_eur. The dataset was converted into multiple tables for efficiently answering our business question. The below diagram showcases the additional tables being created along with the fields in each table :-

Table	Fields
cars_clean	maker, model, mileage, manufacture_year, date_created, date_last_seen, price_eur
cars_with_days_listed	maker, model, date_created, date_last_seen, days_listed
cars_with_age	maker, model, manufacture_year, date_last_seen, age
cars_with_mileage	maker, model, mileage, range
cars_sold	maker, model
cars_sold_price	maker, model, price_eur, range, days_listed

Firstly, the dataset was cleaned and converted to cars_clean table by allowing only those records which had maker, model and a manufacture_year. This reduced the dataset size to about 2.4 million records.

The field days_listed was calculated by taking the difference of column date_last_seen and date_created indicating how many days the car was listed in the agency.

The field age was computed by taking the YEAR(date_last_seen) and subtracting manufacture_year from it. This resulted in cars_with_age table showing the age of the car when it was sold.

Based on initial analysis and sample queries mentioned, a mileage of about 250000 was considered as a sweet spot. About 2 million records out of 2.4 million had mileage less than 250000, hence the table cars_with_mileage was constructed. The range was computed by performing int(mileage/25000) which was helpful in the analysis process later and in grouping cars within a particular mileage range.

From the analysis process mentioned in the later sections, it was found that cars usually get sold max in 60 days of listing. So, the cars_sold table was created by taking the value of days_listed column <= 60.

Finally, cars_sold_price was created to identify for which price, the majority of cars sold. This was helpful later in the analysis process. The range represents int(price_eur/1000.00) to allow grouping of cars within a particular price range.

Queries Used

CREATE TABLE IF NOT EXISTS cars_clean as select maker, model, mileage, manufacture_year, date_created, date_last_seen, price_eur from cars where maker != '' and model != '' and manufacture_year != '';

CREATE TABLE IF NOT EXISTS cars_with_days_listed as select maker, model, date_created, date_last_seen, datediff(to_date(date_last_seen), to_date(date_created)) as days_listed from cars_clean;

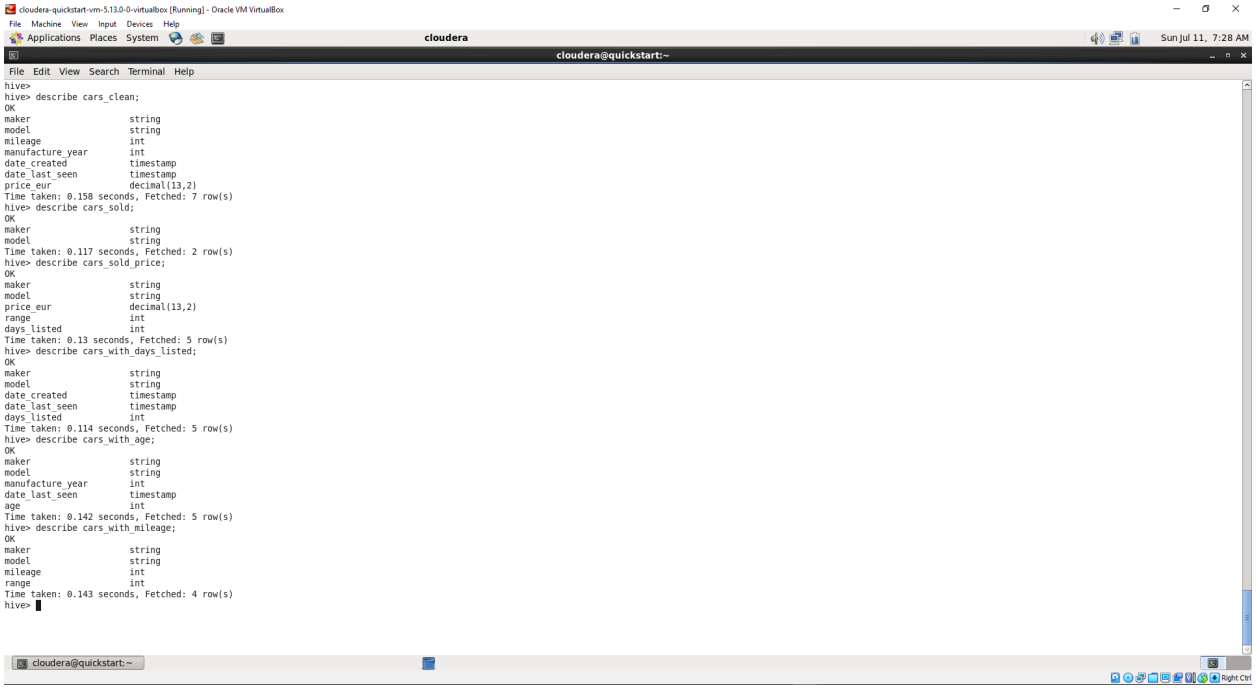
CREATE TABLE IF NOT EXISTS cars_with_age as select maker, model, manufacture_year, date_last_seen, year(date_last_seen) - manufacture_year as age from cars_clean where manufacture_year is not null;

CREATE TABLE IF NOT EXISTS cars_with_mileage as select maker, model, mileage, int(mileage/25000) as range from cars_clean where mileage is not null and mileage <= 250000;

CREATE TABLE IF NOT EXISTS cars_sold as select maker, model from cars_with_days_listed where days_listed <= 60;

CREATE TABLE IF NOT EXISTS cars_sold_price as select maker, model, price_eur, int(price_eur/1000.00) as range, datediff(to_date(date_last_seen), to_date(date_created) as days_listed from cars_clean where datediff(to_date(date_last_seen), to_date(date_created)) <= 60;

Output



```
hive>
hive> describe cars_clean;
OK
maker                   string
model                   string
mileage                 int
manufacture_year        int
date_created            timestamp
date_last_seen          timestamp
price_eur               decimal(13,2)
Time taken: 0.158 seconds, Fetched: 7 row(s)
hive> describe cars_sold;
OK
maker                   string
model                   string
Time taken: 0.117 seconds, Fetched: 2 row(s)
hive> describe cars_sold_price;
OK
maker                   string
model                   string
price_eur               decimal(13,2)
range                   int
days_listed             int
Time taken: 0.13 seconds, Fetched: 5 row(s)
hive> describe cars_with_days_listed;
OK
maker                   string
model                   string
date_created            timestamp
date_last_seen          timestamp
days_listed             int
Time taken: 0.114 seconds, Fetched: 5 row(s)
hive> describe cars_with_age;
OK
maker                   string
model                   string
manufacture_year        int
date_last_seen          timestamp
age                     int
Time taken: 0.142 seconds, Fetched: 5 row(s)
hive> describe cars_with_mileage;
OK
maker                   string
model                   string
mileage                 int
range                   int
Time taken: 0.143 seconds, Fetched: 4 row(s)
hive>
```

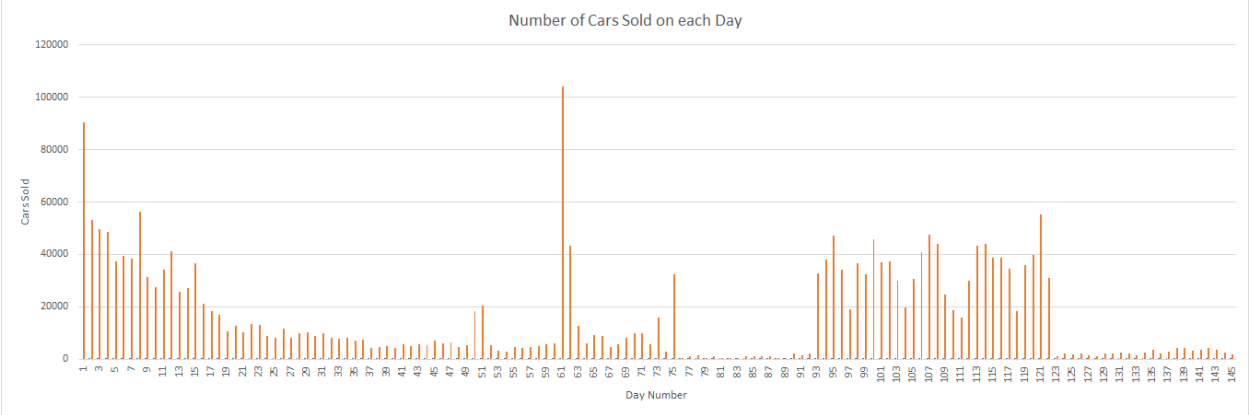
Data Analysis

Based upon the tables created above, various DML commands like SELECT were tested to develop inferences on the dataset.

Queries

select days_listed, count(*) as cnt from cars_with_days_listed group by days_listed order by days_listed;

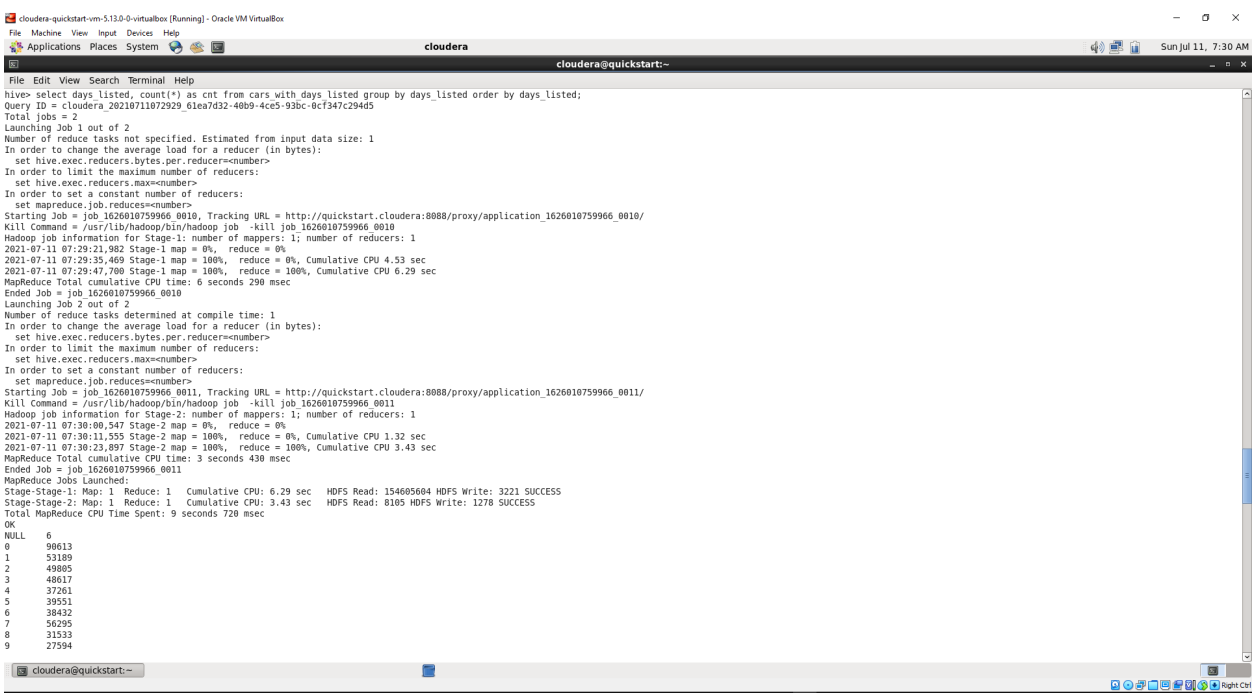
The query returns days_listed and count of records for particular days_listed value and sorts the results in ascending order. So, the output of this query has days_listed from 0 -> 180 and the corresponding count. A line chart was created in Excel based upon the results got



From the chart, it can be seen that from Day 0 to Day 60, the number of cars being sold decreased which is obvious since more recent the car ad is, higher is the probability of getting it sold. Although, from Day 61, the number of cars rose rapidly which can be interpreted that unsold cars were again listed on the agency. Also, Day 40 had the least number of cars being sold.

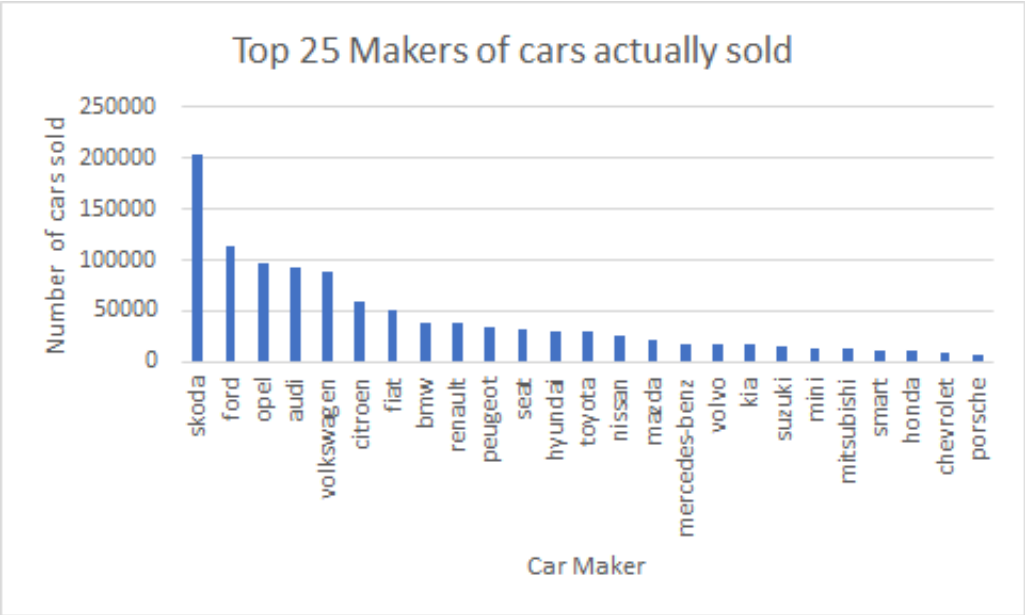
Therefore from the above analysis, we consider that 60 days was a good time to sell a car.

Output



select maker, count(*) as csg_count from cars_sold group by maker order by csg_count desc limit 25;

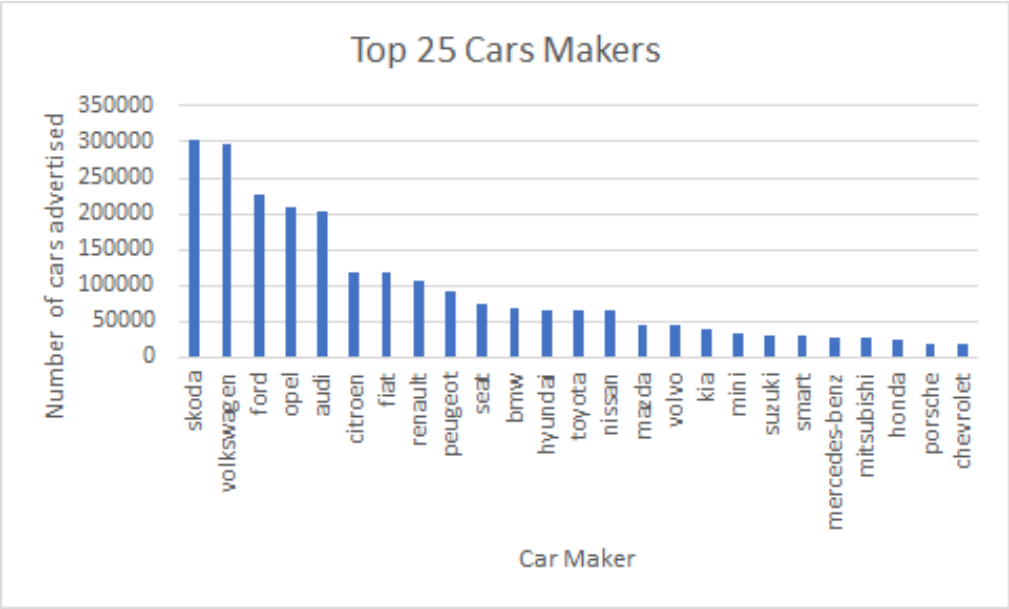
The query returns the top 25 car makers and number of cars they sold (i.e in less than 60 days). A column chart was created in Excel based upon the results got



From the chart, it can be seen Skoda significantly tops the list with the biggest market share in the cars industry. Porsche stands last which can be understood since it manufactures luxurious cars only. These results can be compared with another query mentioned below

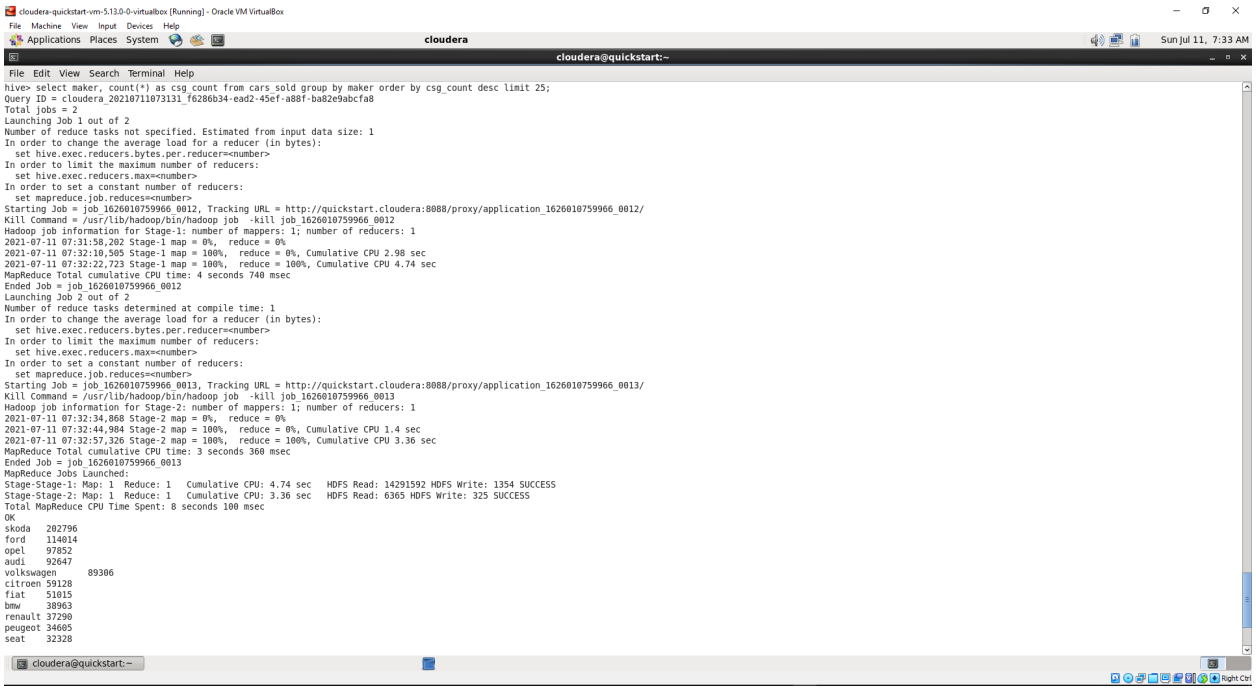
select maker, count(*) as cwdl_count from cars_with_days_listed group by maker order by cwdl_count desc limit 25;

The query returns the top 25 car makers and the number of cars they advertised. A column chart was created in Excel based upon the results got

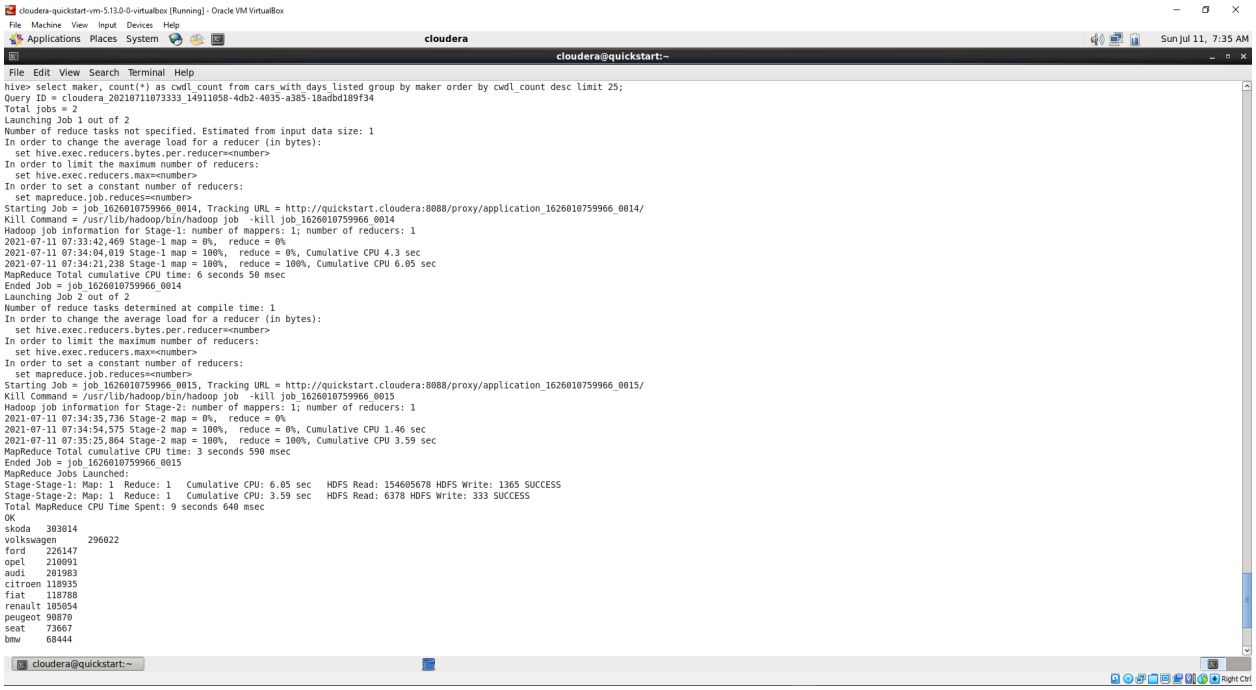


From the previous chart and this, Skoda was still the most advertised car and was also actually sold well. However, Volkswagen was also advertised a lot, however the car brand wasn't able to sell as many cars. On the contrary, for Mercedes-Benz, even though the cars were being advertised less but its sales were good compared with other brands.

Outputs

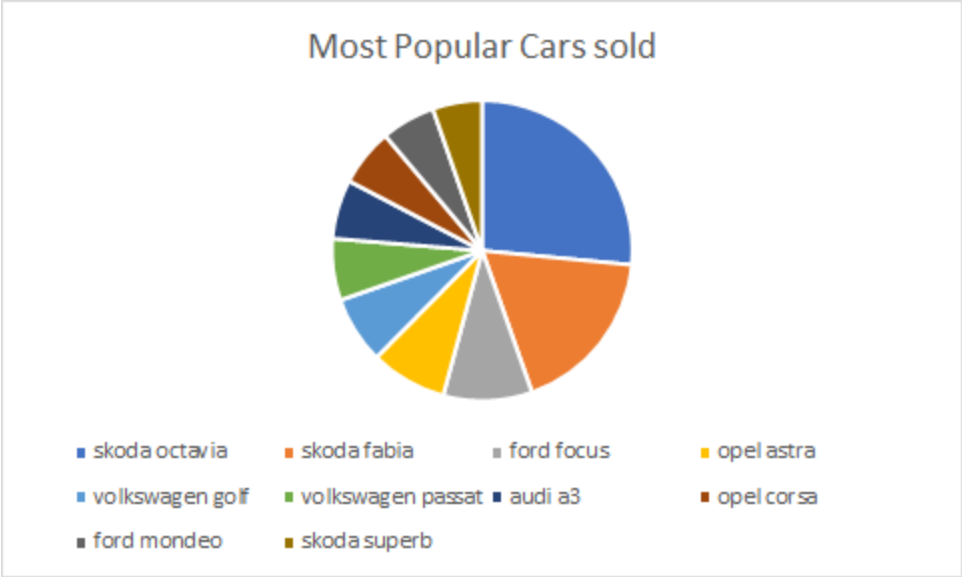


```
hive> select maker, count(*) as csg_count from cars_sold group by maker order by csg_count desc limit 25;
Query ID = cloudera_20210711073131_f6286b34-ead2-45ef-a88f-ba82e9abcfa8
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1626010759966_0012, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1626010759966_0012/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1626010759966_0012
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2021-07-11 07:31:58,202 Stage-1 map = 0%,  reduce = 0%
2021-07-11 07:32:10,505 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.98 sec
2021-07-11 07:32:22,723 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 4.74 sec
MapReduce Total cumulative CPU time: 4 seconds 740 msec
Ended Job = job_1626010759966_0012
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1626010759966_0013, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1626010759966_0013/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1626010759966_0013
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2021-07-11 07:32:34,868 Stage-2 map = 0%,  reduce = 0%
2021-07-11 07:32:44,984 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 1.4 sec
2021-07-11 07:32:57,326 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 3.36 sec
MapReduce Total cumulative CPU time: 3 seconds 360 msec
Ended Job = job_1626010759966_0013
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 4.74 sec   HDFS Read: 14291592 HDFS Write: 1354 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 3.36 sec   HDFS Read: 6365 HDFS Write: 325 SUCCESS
Total MapReduce CPU Time Spent: 8 seconds 100 msec
OK
skoda   202796
ford    114014
opel    97852
audi    92647
volkswagen      89306
citroen 59128
fiat    51015
bmw     38963
renault 37290
peugeot 34605
seat    32328
```



select maker, model, count(*) as csg_count from cars_sold group by maker, model order by csg_count desc limit 10;

The query returns the top 10 popular cars being sold. A pie chart was created in Excel based upon the results got



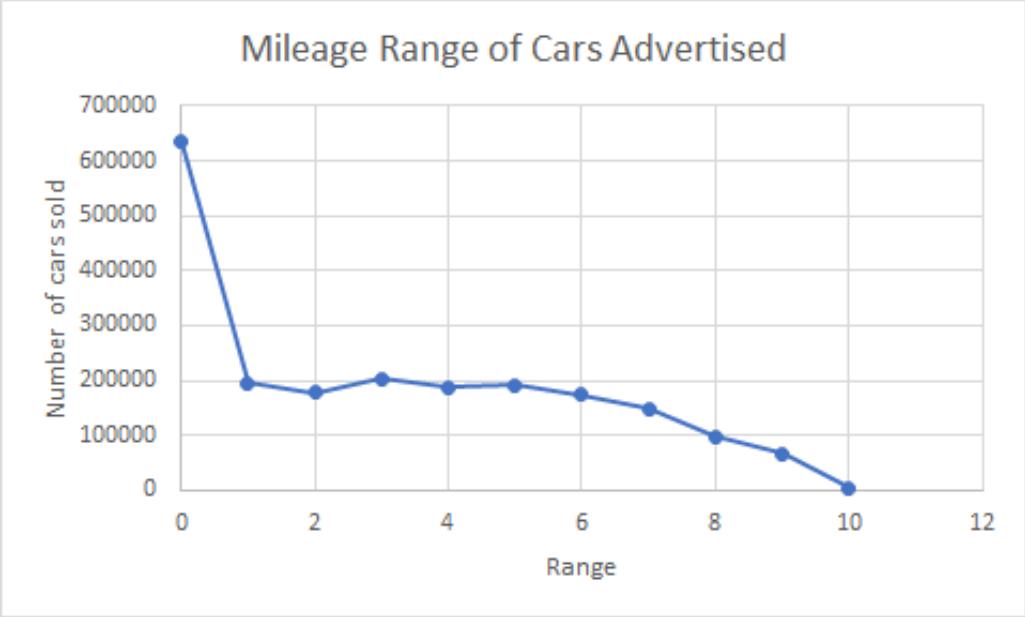
From the chart, it can be seen that cars from Skoda like Octavia, Fabia were sold the most followed by Ford's Focus and so on.

Output

```
cloudera@quickstart:~
hive> select maker, model, count(*) as csg_count from cars_sold group by maker, model order by csg_count desc limit 10;
Query ID = cloudera_20210711073636_adec568d-de64-49aa-9554-f287eecac938
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1626010759966_0017, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1626010759966_0017/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1626010759966_0017
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2021-07-11 07:36:34,639 Stage-1 map = 0%,  reduce = 0%
2021-07-11 07:36:46,831 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 3.53 sec
2021-07-11 07:36:59,053 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 5.7 sec
MapReduce Total cumulative CPU time: 5 seconds 700 msec
Ended Job = job_1626010759966_0017
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1626010759966_0018, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1626010759966_0018/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1626010759966_0018
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2021-07-11 07:37:13,240 Stage-2 map = 0%,  reduce = 0%
2021-07-11 07:37:24,432 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 2.19 sec
2021-07-11 07:37:36,664 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 4.24 sec
MapReduce Total cumulative CPU time: 4 seconds 240 msec
Ended Job = job_1626010759966_0018
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 5.7 sec   HDFS Read: 14292002 HDFS Write: 37820 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 4.24 sec   HDFS Read: 43177 HDFS Write: 186 SUCCESS
Total MapReduce CPU Time Spent: 9 seconds 940 msec
OK
skoda   octavia 89532
skoda   fabia   61153
ford    focus   32283
opel    astra   27801
volkswagen      golf    24425
volkswagen      passat  22391
audi    a3      21528
opel    corsa   20862
ford    mondeo  19665
skoda   superb  17998
Time taken: 76.192 seconds, Fetched: 10 row(s)
```

select range, count(range) from cars_with_mileage group by range;

The query returns the range and the number of cars that fall into it. Range was calculated previously by calculating int(mileage/25000). A line chart was created in Excel based upon the results obtained. Ex:- If the mileage of the car is 70000, then range would be 2.



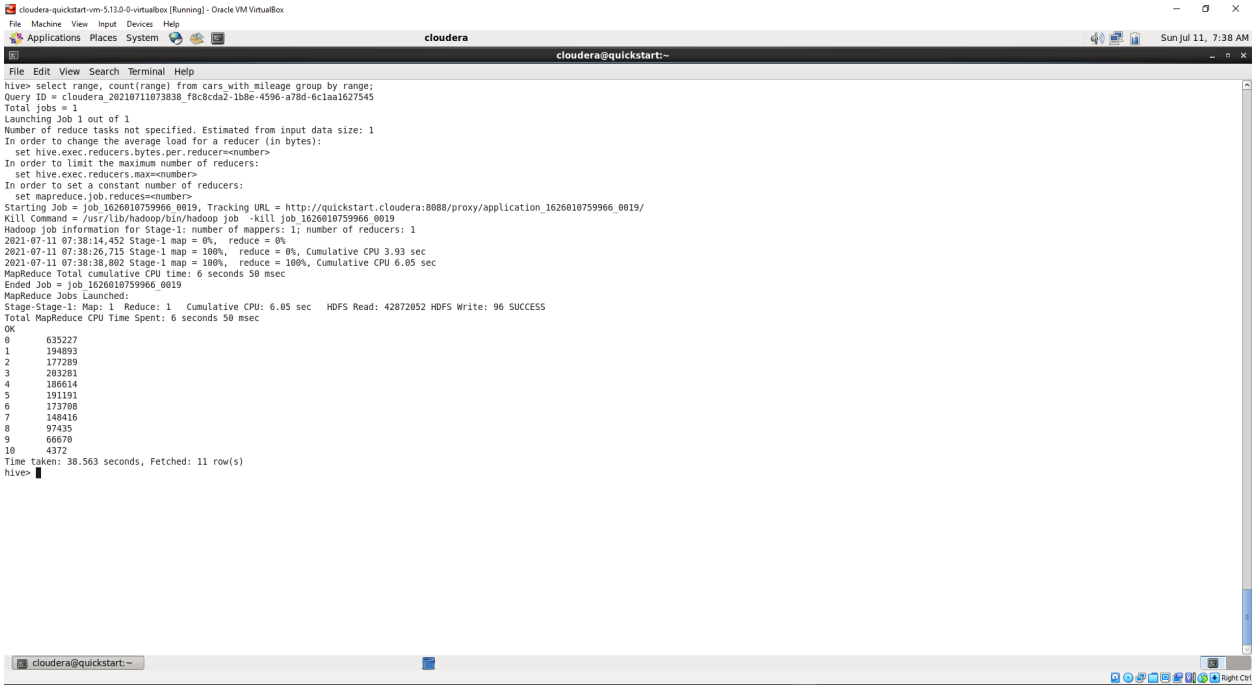
From the chart, it can be seen that cars with mileage less than 25000 or range 0 significantly had higher chances of selling with about 6.35 lakh selling in this case. After crossing the 25000 mark, the number of cars being sold reduced to just 2 lakh which remains almost even until range = 6. Then, it dropped until the mileage reached the threshold value of 250000.

select range, count(*) from cars_sold_price group by range;

The query returns the range of price and the number of cars being sold. Range here is calculated by calculating int(price_eur/1000.00).

However, since the price_eur column had few values which were impossible, another query was run to find the price values of most cars being sold. The threshold value for car price was set as 5000 * 1000.00 = 5 million dollars.

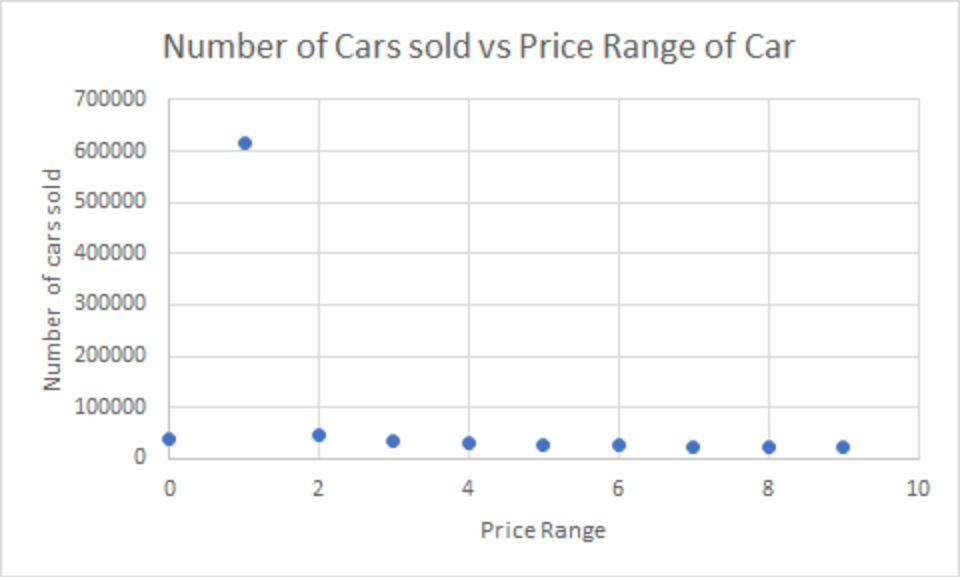
Output



```
hive> select range, count(range) from cars_with_mileage group by range;
Query ID = cloudera_20210711073838_f8c8cda2-1b8e-4596-a78d-6c1aa1627545
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1626010759966_0019, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1626010759966_0019/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1626010759966_0019
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2021-07-11 07:38:14,452 Stage-1 map = 0%,  reduce = 0%
2021-07-11 07:38:26,715 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 3.93 sec
2021-07-11 07:38:38,802 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 6.05 sec
MapReduce Total cumulative CPU time: 6 seconds 50 msec
Ended Job = job_1626010759966_0019
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 6.05 sec   HDFS Read: 42872052 HDFS Write: 96 SUCCESS
Total MapReduce CPU Time Spent: 6 seconds 50 msec
OK
0       635227
1       194893
2       177289
3       203281
4       186614
5       191191
6       173708
7       148416
8       97435
9       66670
10      4372
Time taken: 38.563 seconds, Fetched: 11 row(s)
hive>
```

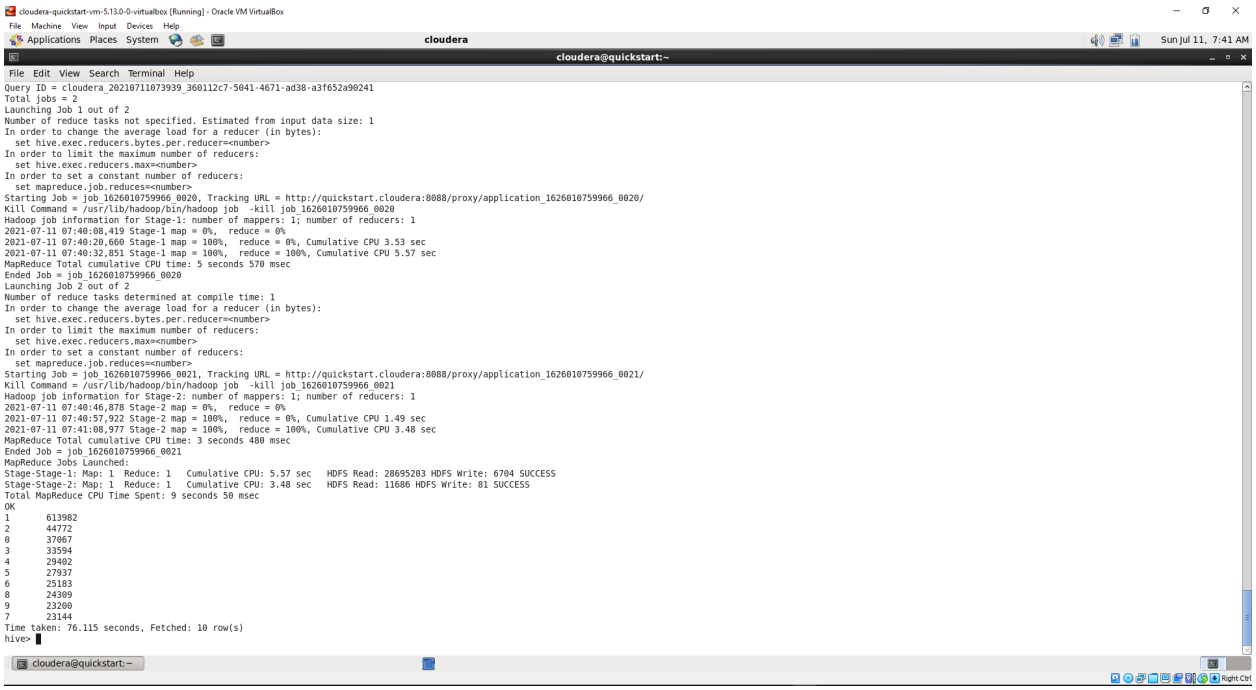
select range, count(*) as cnt from cars_sold_price group by range having range <= 5000 order by cnt desc limit 10;

A scatter plot was created in Excel based upon the results obtained.



It was clear that the majority of the cars that were sold were in the price range of $[1000, 2000). So, it can be considered that cars within this price range should be preferred.

Output



```
Query ID = cloudera_20210711073939_360112c7-5041-4671-ad38-a3f652a90241
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1626010759966_0020, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1626010759966_0020/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1626010759966_0020
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2021-07-11 07:40:08,419 Stage-1 map = 0%,  reduce = 0%
2021-07-11 07:40:20,660 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 3.53 sec
2021-07-11 07:40:32,851 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 5.57 sec
MapReduce Total cumulative CPU time: 5 seconds 570 msec
Ended Job = job_1626010759966_0020
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1626010759966_0021, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1626010759966_0021/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1626010759966_0021
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2021-07-11 07:40:46,878 Stage-2 map = 0%,  reduce = 0%
2021-07-11 07:40:57,922 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 1.49 sec
2021-07-11 07:41:08,977 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 3.48 sec
MapReduce Total cumulative CPU time: 3 seconds 480 msec
Ended Job = job_1626010759966_0021
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 5.57 sec   HDFS Read: 28695203 HDFS Write: 6704 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 3.48 sec   HDFS Read: 11686 HDFS Write: 81 SUCCESS
Total MapReduce CPU Time Spent: 9 seconds 50 msec
OK
1       613982
2       44772
0       37067
3       33594
4       29402
5       27937
6       25183
8       24309
9       23200
7       23144
Time taken: 76.115 seconds, Fetched: 10 row(s)
hive>
```

The column manufacture_year has values like years 327, 486, 1500 etc which was impossible as a car can't be that much old. Therefore, the below query was run to identify how many cars had age <= 25.

select count(*) from cars_with_age where age <= 25;

The query resulted in a value of 2.13 million which means the majority of the cars in the dataset were less than 25 years old. Hence, we can consider that to be on the agency, the car should not be more than 25 years old.

Output

Suggestions for the stakeholders

- Most popular cars having good selling/advertisement ratio comes from makers like Skoda, Ford, Audi, Mercedes Benz, Volkswagen
- Car age matters and shouldn't be more than 25 years old
- Preferably, newer cars with mileage less than 25000 typically sell way faster than older cars
- Normally, cars should be sold within 60 days of its listing period
- Top cars that sell are Skoda Octavia/Fabia, Ford Focus, Opel Astra etc.
- Car price should be greater than equal to 1000$ and less than 2000$ for its better demand in the market.

Conclusion

To recapitulate, the dataset of 16 columns and 4 million records was filtered down to about 7 columns and 2.4 million records after taking into account the important columns like maker, model and manufacture_year. Some attributes like days_listed, age, mileage, price_eur played a significant role and after analyzing the additional tables, the above mentioned inferences were suggested which would help answer the business question.