BUSINESS OBJECTIVE:

To provide data driven decision for the company, which is contemplating to invest in a used car business. And to recommend top 5 car makers with a good resale value which can lead to a sound investment decision.

ABOUT THE DATASET:

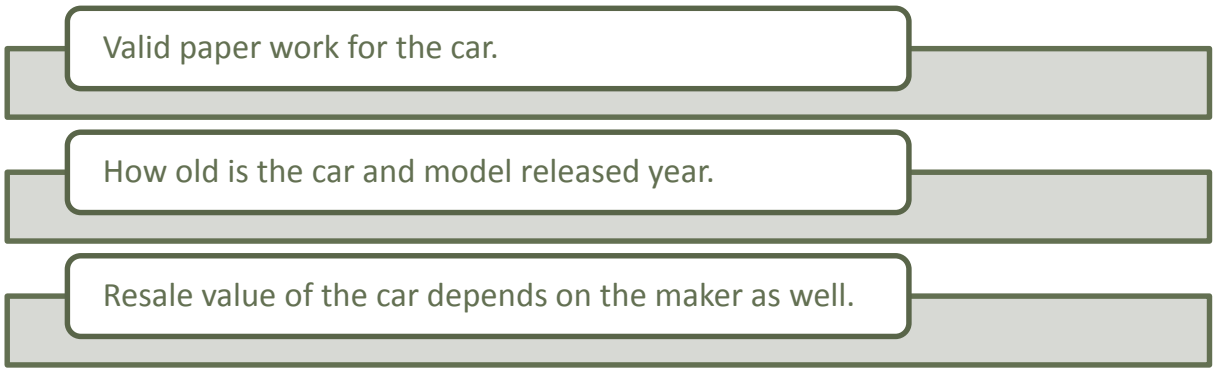
For this analysis we have been asked to use the cars dataset whose data was made up by scraping from several websites in Czech Republic and Germany over a period of more than a year. This dataset was also chosen for its veracity it has as well.

REFRENCE:

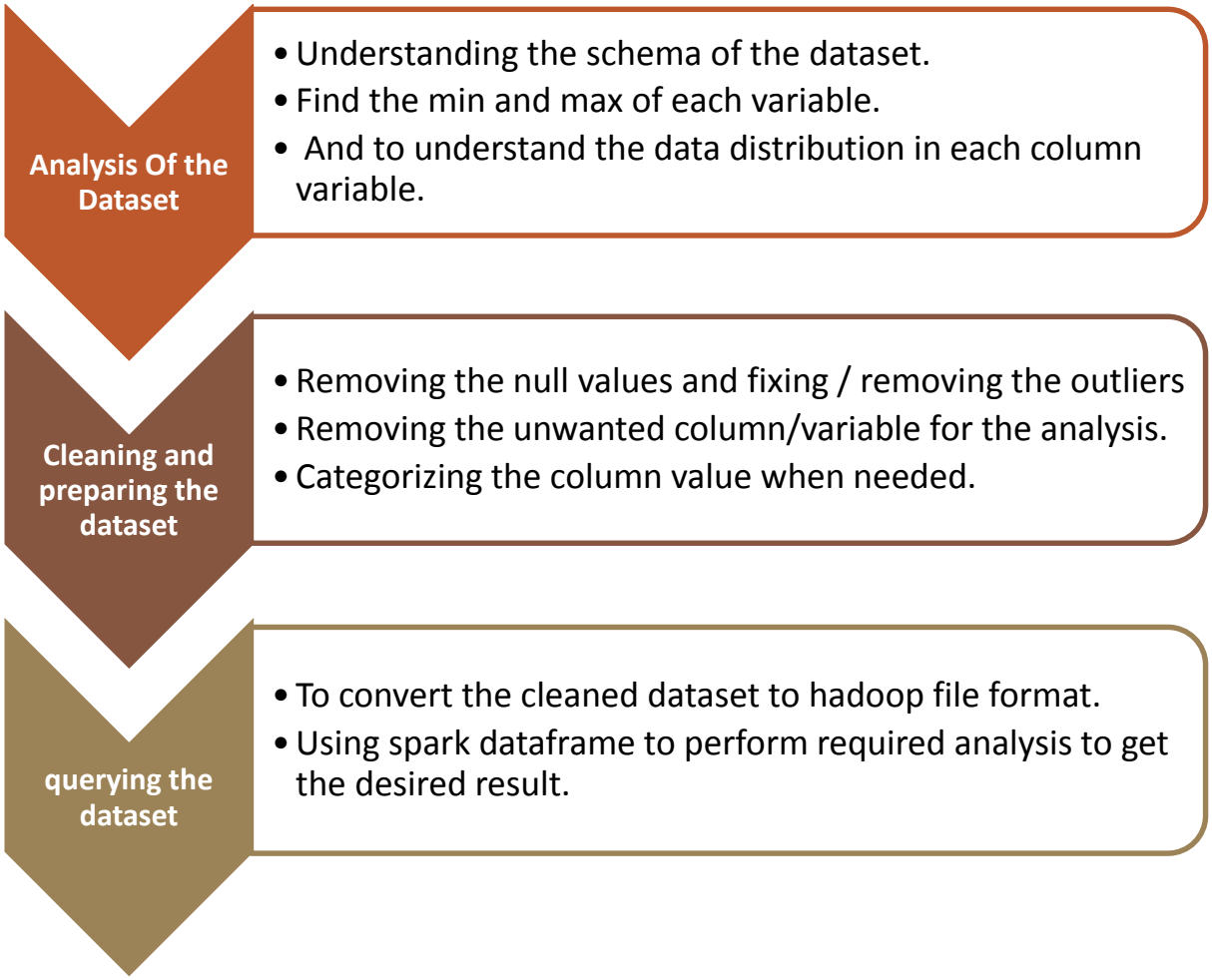
This dataset was taken from kaggle:

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

MARKET RESEARCH RESULT:

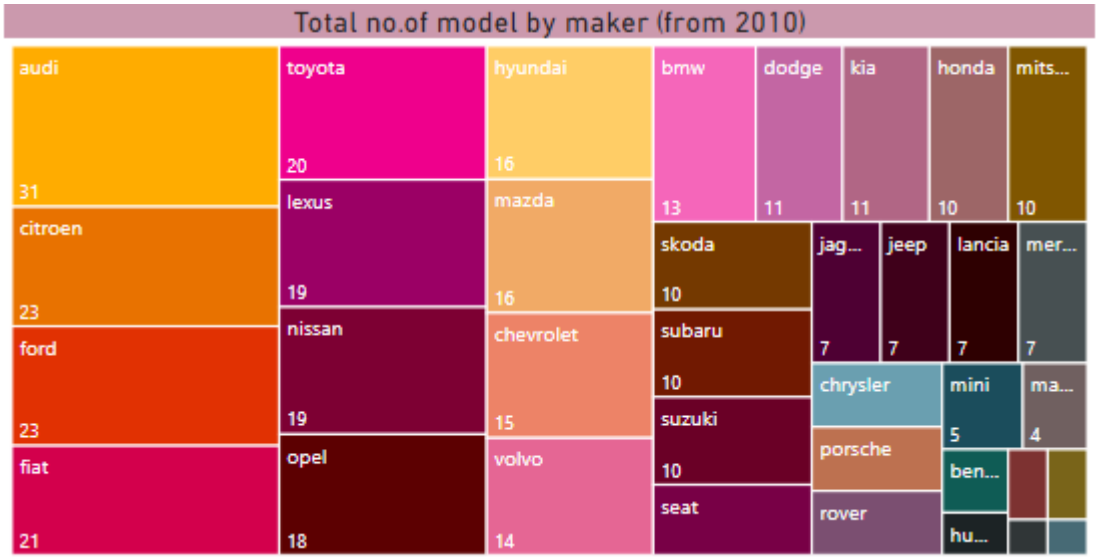


STEPS TO EXTRACT BUSINESS INSIGHT :

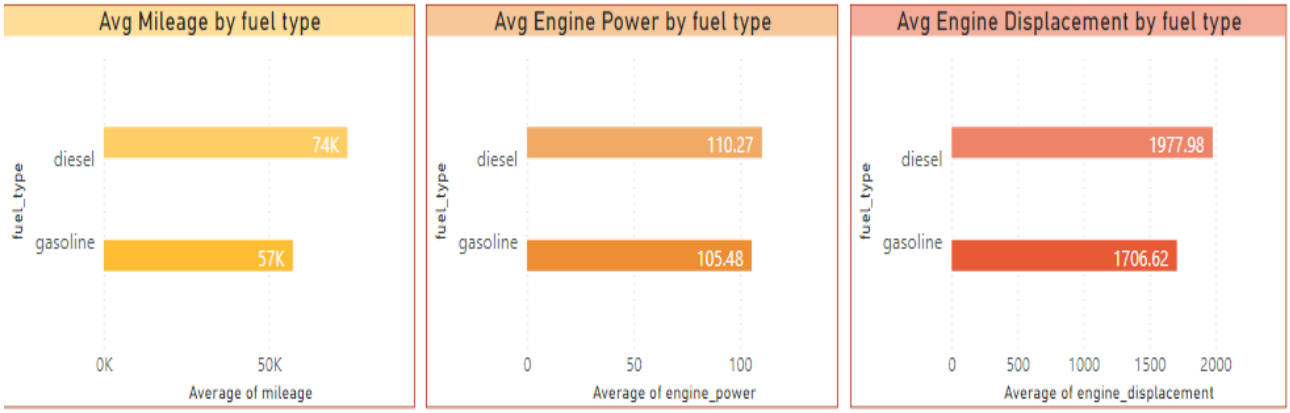


INITIAL ANALYSIS OF THE DATASET IN POWER BI:

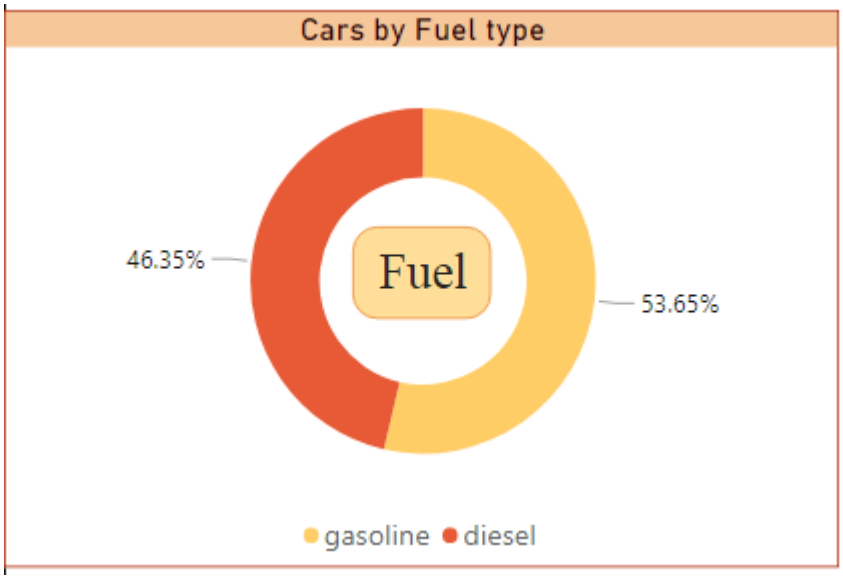
MAKERS AND MODEL COUNT:

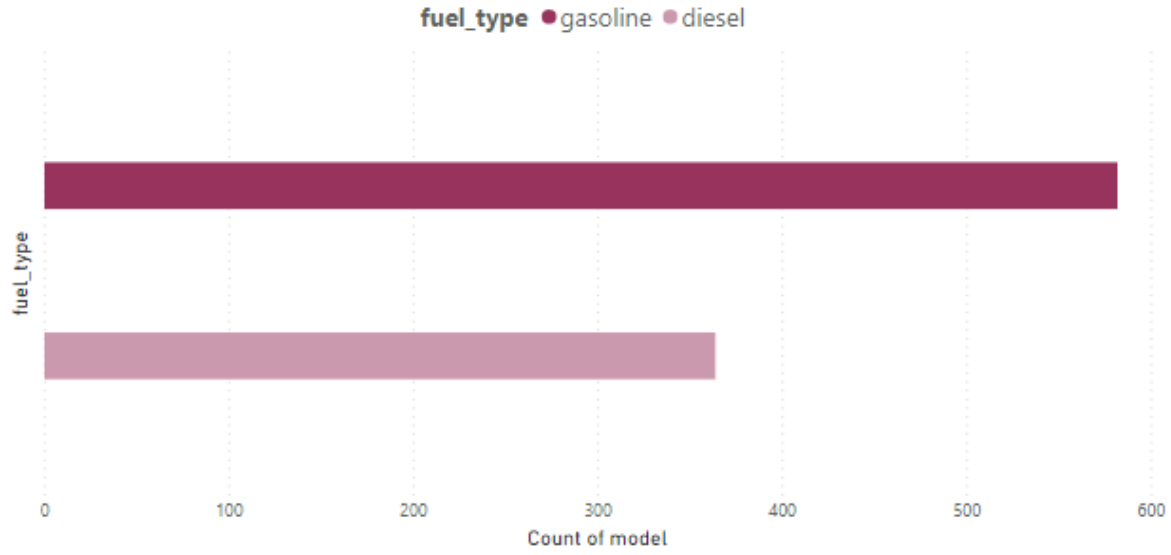


AVERAGE VALUES OF EACH SPECS BY FUEL TYPE:

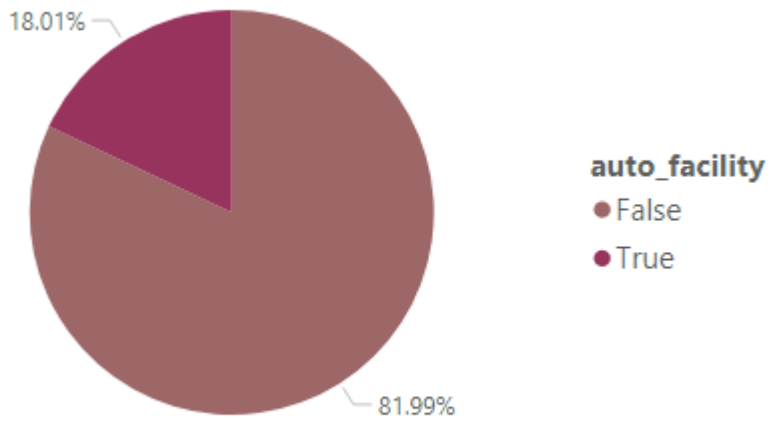


DISTRIBUTION OF CARS BY FUEL TYPE:

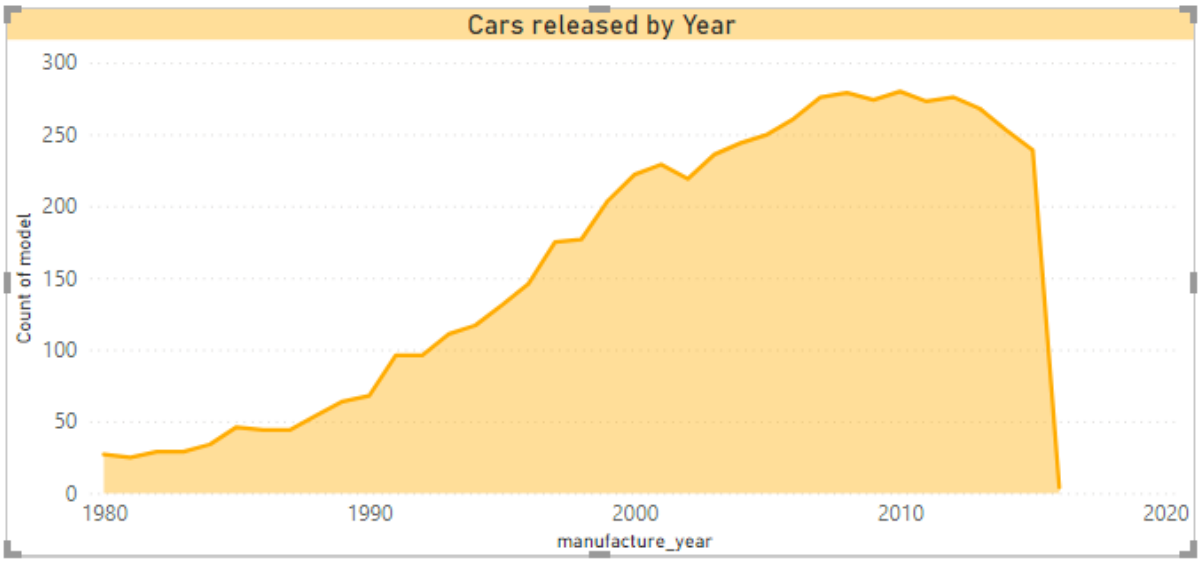




PERCENTAGE OF CARS BY AUTO_FACILITY AND NON-AUTO_FACILITY:



MODELS RELEASED OVER THE YEARS:



DATA CLEANING USING EXCEL:

Contents:

Maker	Model	Mileage (km)	Manufacture year
Engine displacement (cc)	Engine power (kW)	Body type	Color slug
stk year	transmission	Door count	Seat count
Fuel type	Date created	Date *last* seen	Price Eur

UNWANTED ATTRIBUTES:

1. Door count.
2. Seat count.
3. Body type-can't be used as it is so it is transformed into either compact (seat count<7) or SUV (seat count>=7) based on seat count.
4. stk year- year value does not affect the analysis but whether the car has pollution control measure does, hence if given TRUE or FALSE.
5. ad_date created-The data has values which are not logical and so its unreliable.
6. ad_date scraped- The same as ad_date created.

OUTLIER DETECTION AND DATA CLEANING

1. MAKER:

Maker contained lot of blank value and since it is an individual variable, the blank values have been removed.

2. MODEL:

Model also contained lot of blank value and since it is an individual variable, the blank has been removed.

3. MILEAGE:

Mileage column contains values of distance traveled (km) by a car in a year and it had zero, blank and illogical values.

Since it is an independent variable, it was not possible to predict the values in place of error and so data with less than 10,000 has been removed.

4. MANUFACTURE YEAR:

Manufacture year contained lot of illogical values e.g.: 2030,1600 etc. and so for cars with less than 1980 have been replaced with min value of 1980 and values that are greater than 2020 have been replaced with max value of 2020.

5. ENGINE DISPLACEMENT:

Displacement of engine had blank, negative and extreme values which have been removed. And replacement is not performed on this, since it is an important spec to determine the price and so it can't be skewed.

6. ENGINE POWER:

Engine power contained values such as blank, zero and illogical values (e.g.: 2, 5, 10).hence values which are less that 30 and blank are removed.

7. BODY TYPE:

Body type column was incomplete and was not use able hence it was replaced as category data containing COMPACT and SUV. Which is derived from the column seat count.

Also, renamed the column to Car Type.

8. TRANSMISSION:

Renamed this into Auto Facility.

Categorized the value into TRUE OR FALSE.

PERFORMING QUERY ON THE DATASET:

SCHEMA OF THE PREPARED DATASET:

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

val cars_col = spark
  .read.format("csv")
  .option("header", "true")
  .load("hdfs://10.128.0.5:8020/BigData/Cars.csv")

// Exiting paste mode, now interpreting.

cars_col: org.apache.spark.sql.DataFrame = [maker: string, model: string ... 9 more fields]

scala> cars_col.printSchema()
root
 |-- maker: string (nullable = true)
 |-- model: string (nullable = true)
 |-- mileage: string (nullable = true)
 |-- manufacture_year: string (nullable = true)
 |-- engine_displacement: string (nullable = true)
 |-- engine_power: string (nullable = true)
 |-- pollution_test: string (nullable = true)
 |-- auto_facility: string (nullable = true)
 |-- car_type: string (nullable = true)
 |-- fuel_type: string (nullable = true)
 |-- price: string (nullable = true)
```

Popularity of the car maker influences the resale value of the car. Success and popularity of the maker can be determined by number of models released by the maker over the last five years recorded in the dataset.

And to see the popularity of the preferred fuel type over last five years by the number of models released and their average price.

QUERY :

(Filter by DIESEL)

```
val cars_1 =cars_cols
.filter("fuel_type == \"diesel\" ")
.filter("manufacture_year >= 2010")
.groupBy(col("maker"))
.agg(avg(col("price")).alias("average_price"),
countDistinct(col("model")).alias("model_count"))
.orderBy(col("model_count").desc)
.show(10)
```

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

val cars_1 =cars_cols
.filter("fuel_type == \"diesel\" ")
.filter("manufacture_year >= 2010")
.groupBy(col("maker"))
.agg(avg(col("price")).alias("average_price"),
countDistinct(col("model")).alias("model_count"))
.orderBy(col("model_count").desc)
.show(10)

// Exiting paste mode, now interpreting.

+-------+------------------+-----------+
|  maker|     average_price|model_count|
+-------+------------------+-----------+
|   audi|26399.199597408315|         21|
|citroen|15203.072727272727|         21|
|   ford|15491.194953104065|         16|
|   fiat| 14005.63829787234|         16|
| nissan|15742.857683982684|         14|
|  volvo| 19872.21094890511|         14|
|   opel|15464.507004370727|         12|
|hyundai|15474.052052785924|         11|
| toyota|15802.642208245981|         10|
|  skoda| 15210.36880010871|          9|
+-------+------------------+-----------+
only showing top 10 rows
```

QUERY :

(FILTER BY GASOLINE)

```
val cars_1 =cars_cols
.filter("fuel_type == \"gasoline\" ")
.filter("manufacture_year >= 2010")
.groupBy(col("maker"))
.agg(avg(col("price")).alias("average_price"),
countDistinct(col("model")).alias("model_count"))
.orderBy(col("model_count").desc)
.show(10)
```

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

val cars_1 =cars_cols
.filter("fuel_type == \"gasoline\" ")
.filter("manufacture_year >= 2010")
.groupBy(col("maker"))
.agg(avg(col("price")).alias("average_price"),
countDistinct(col("model")).alias("model_count"))
.orderBy(col("model_count").desc)
.show(10)

// Exiting paste mode, now interpreting.

+-------+------------------+-----------+
|  maker|     average_price|model_count|
+-------+------------------+-----------+
|   audi| 27359.06438791733|         31|
|   ford|14939.473399246705|         22|
|citroen|13579.953982300885|         22|
|   fiat| 14124.97710755814|         21|
| toyota|14245.194739787354|         19|
|  lexus| 25490.28510638298|         18|
|   opel| 13670.65974329055|         17|
| nissan|14842.196130753837|         17|
|hyundai|13453.377108983128|         15|
|  mazda| 15535.57710501419|         14|
+-------+------------------+-----------+
only showing top 10 rows
```

From the analysis done, it is can be concluded that makers who have released 10 OR more should be consider for the analysis, since they release constant flow of models in the market over the past years can be considered as being successful and popular.

Now to analyse the SPEC of the car across price and it is listed separately for both the fuel type.

QUERY:
(FILTER PRICE LESS THAN 20,000)

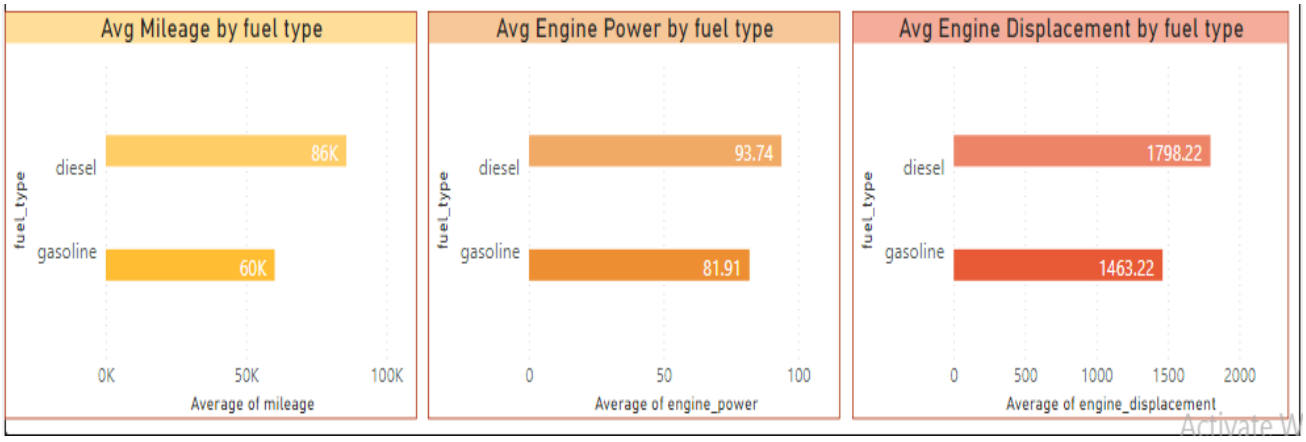
```
val cars_1 =cars_cols
.filter("price<=20000")
.filter("manufacture_year >= 2010")
.groupBy(col("fuel_type"))
.agg(avg(col("mileage")),avg(col("engine_displacement")),avg(col("engine_power")))
.show()
```

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

val cars_1 =cars_cols
.filter("price<=20000")
.filter("manufacture_year >= 2010")
.groupBy(col("fuel_type"))
.agg(avg(col("mileage")),avg(col("engine_displacement")),avg(col("engine_power")))
.show()

// Exiting paste mode, now interpreting.

+---------+----------------+------------------------+-----------------+
|fuel_type|    avg(mileage)|avg(engine_displacement)|avg(engine_power)|
+---------+----------------+------------------------+-----------------+
| gasoline|60168.7319139252|      1463.4775836346626| 81.9360760613953|
|   diesel|85670.9859392336|      1798.3179568148844|93.73625928138603|
+---------+----------------+------------------------+-----------------+
```



QUERY:
(FILTER PRICE BETWEEN 20,000 AND 40,000)

```
val cars_1 =cars_cols
.filter("price>=20000 AND price<=40000")
.filter("manufacture_year >= 2010")
.groupBy(col("fuel_type"))
.agg(avg(col("mileage")),avg(col("engine_displacement")),avg(col("engine_power")))
.show()
```

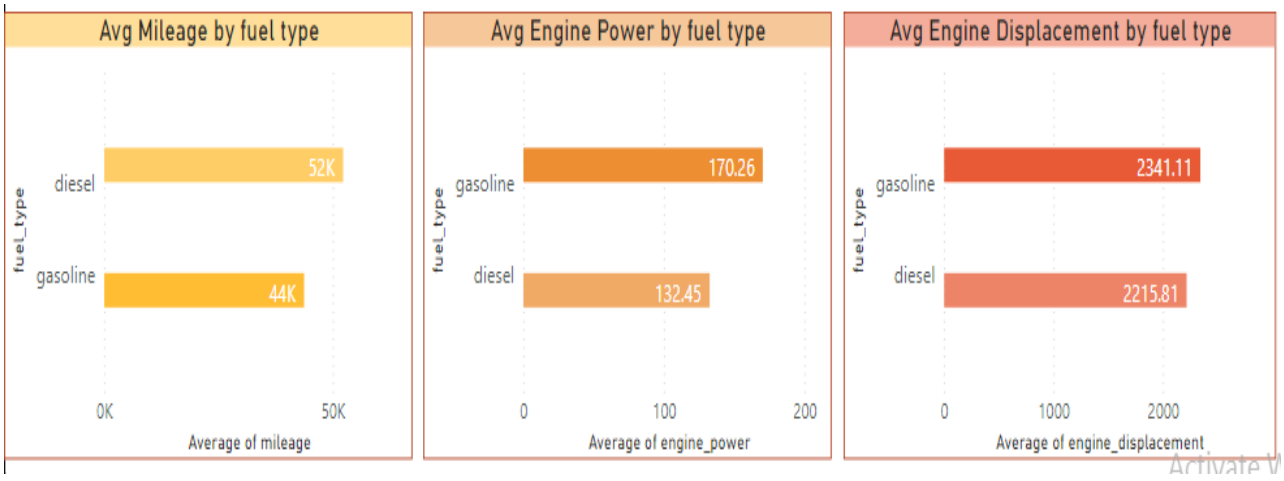
```
scala> :paste
// Entering paste mode (ctrl-D to finish)

val cars_1 =cars_cols
.filter("price>=20000 AND price<=40000")
.filter("manufacture_year >= 2010")
.groupBy(col("fuel_type"))
.agg(avg(col("mileage")),avg(col("engine_displacement")),avg(col("engine_power")))
.show()

// Exiting paste mode, now interpreting.

+---------+------------------+------------------------+------------------+
|fuel_type|      avg(mileage)|avg(engine_displacement)| avg(engine_power)|
+---------+------------------+------------------------+------------------+
| gasoline| 43707.06821106821|       2341.113807685236| 170.2592388306674|
|   diesel|52240.885168158726|      2215.8523590612144|132.45526252117105|
+---------+------------------+------------------------+------------------+

cars_1: Unit = ()
```



QUERY:
(FILTER PRICE BETWEEN 40,000 AND 60,000)

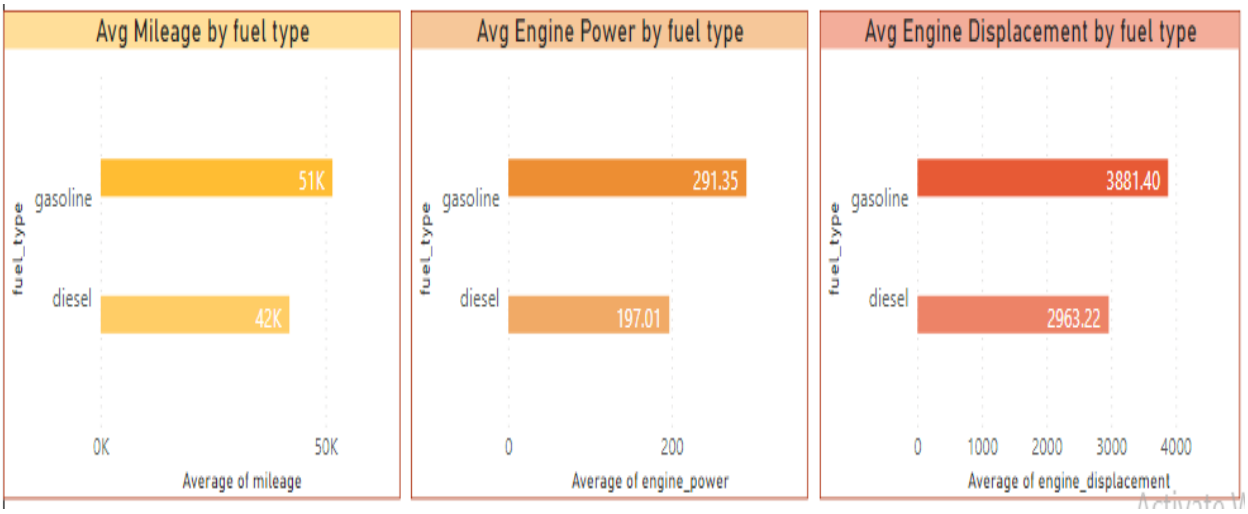
```
val cars_1 =cars_cols
.filter("price>=40000 AND price<=60000")
.filter("manufacture_year >= 2010")
.groupBy(col("fuel_type"))
.agg(avg(col("mileage")),avg(col("engine_displacement")),avg(col("engine_power")))
.show()
```

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

val cars_1 =cars_cols
.filter("price>=40000 AND price<=60000")
.filter("manufacture_year >= 2010")
.groupBy(col("fuel_type"))
.agg(avg(col("mileage")),avg(col("engine_displacement")),avg(col("engine_power")))
.show()

// Exiting paste mode, now interpreting.

+---------+----------------+-----------------------+------------------+
|fuel_type|     avg(mileage)|avg(engine_displacement)| avg(engine_power)|
+---------+----------------+-----------------------+------------------+
| gasoline|51498.22519685039|      3881.4007874015747| 291.3472440944882|
|   diesel|41935.59116607774|       2963.216254416961|197.01413427561837|
+---------+----------------+-----------------------+------------------+
```



QUERY:
(FILTER PRICE ABOVE 60,000)

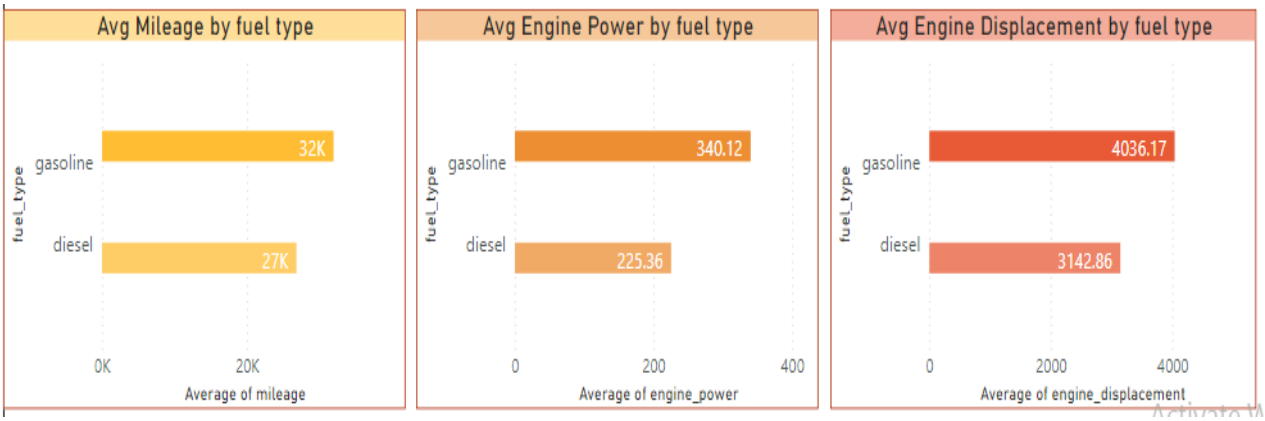
```
val cars_1 =cars_cols
.filter("price>=60000")
.filter("manufacture_year >= 2010")
.groupBy(col("fuel_type"))
.agg(avg(col("mileage")),avg(col("engine_displacement")),avg(col("engine_power")),
avg(col("price")))
.show()
```

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

val cars_1 =cars_cols
.filter("price>=60000")
.filter("manufacture_year >= 2010")
.groupBy(col("fuel_type"))
.agg(avg(col("mileage")),avg(col("engine_displacement")),avg(col("engine_power")),
avg(col("price")))
.show()

// Exiting paste mode, now interpreting.

+---------+------------------+----------------------+------------------+-----------------+
|fuel_type|      avg(mileage)|avg(engine_displacement)| avg(engine_power)|       avg(price)|
+---------+------------------+----------------------+------------------+-----------------+
| gasoline|31925.502049180326|      4036.1734972677596|340.12158469945354|85802.37978142076|
|   diesel| 26808.49512987013|      3142.8636363636365| 225.3555194805195|72390.60551948052|
+---------+------------------+----------------------+------------------+-----------------+
```



From the analysis done it can be observed that diesel car models have an advantage over the gasoline models below the price range of 40,000. After 40,000 we can see the change of trend between both fuel types.

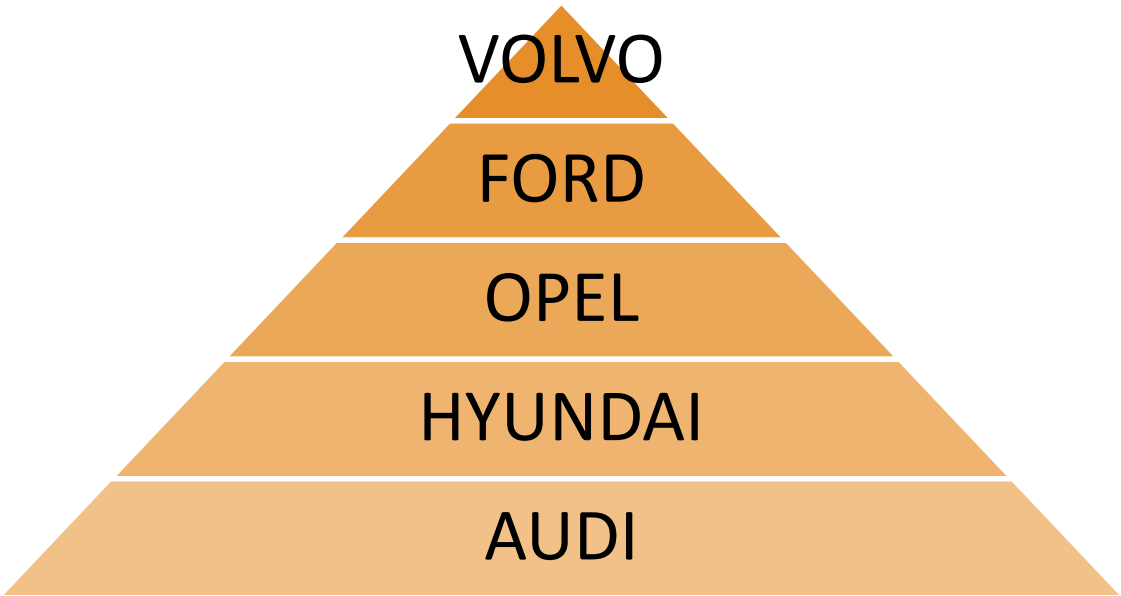
It can be concluded that for commercial second hand car sale (PRICE<40,000), diesel car model has an advantage over its alternative and for luxury second hand car sale (PRICE>60,000), gasoline car model has an advantage over its alternative.

So top 5 makers for commercial second hand car sale filtered by diesel.

QUERY:

```
val cars_1 =cars_cols
.filter("manufacture_year >= 2010")
.filter("price<40000")
.filter("fuel_type == \"diesel\" ")
.groupBy(col("maker"))
.agg(avg(col("mileage")).alias("average_mileage"),
avg(col("price")).alias("average_price"),
countDistinct(col("model")).alias("model_count")).filter("model_count >=10")
.orderBy(col("average_mileage").desc).limit(5).show()
```

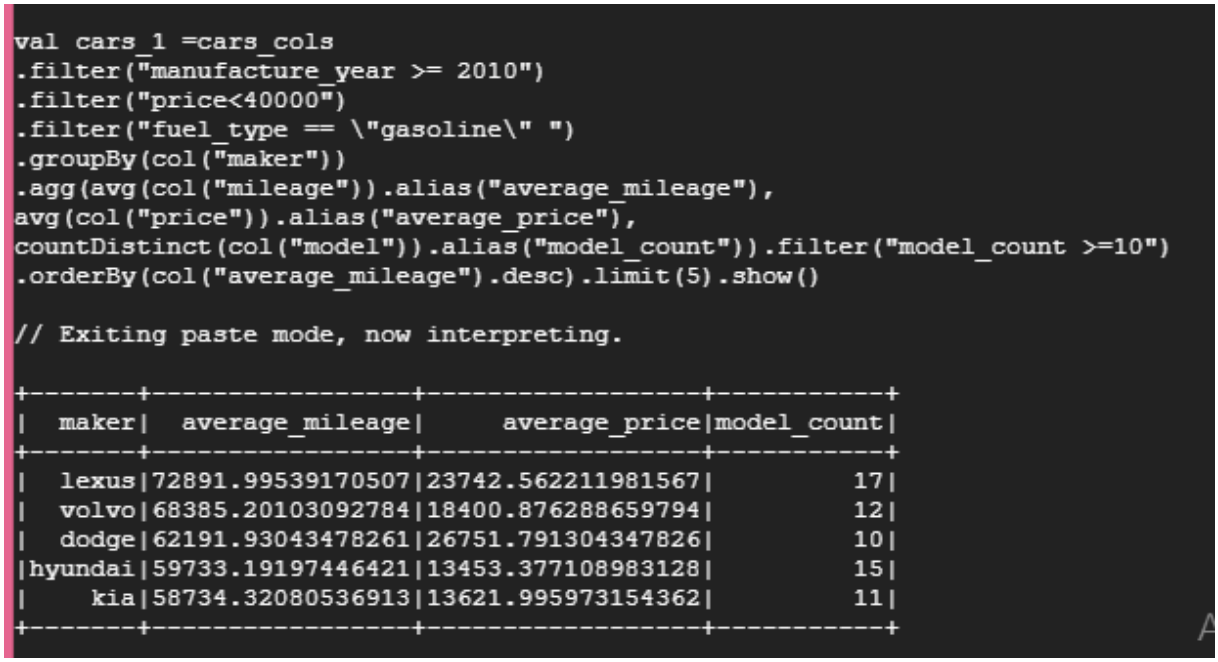
```
+-------+-----------------+------------------+-----------+
|  maker|  average_mileage|     average_price|model_count|
+-------+-----------------+------------------+-----------+
|  volvo|85634.76659292035|19241.383849557522|         14|
|   ford|80942.37056064329| 15474.35246817065|         16|
|   opel|72214.32242152466|  15330.6067264574|         12|
|hyundai|72180.77859237537|15474.052052785924|         11|
|   audi| 70359.9519060754|23436.072848150507|         19|
+-------+-----------------+------------------+-----------+
```



Even though it is preferred to go with diesel models for commercial second hand sale, if wanted some of the top 5 makers by gasoline can be queried using:

QUERY:
(FILTER BY GASOLINE)

```
val cars_1 =cars_cols
.filter("manufacture_year >= 2010")
.filter("price<40000")
.filter("fuel_type == \"gasoline\" ")
.groupBy(col("maker"))
.agg(avg(col("mileage")).alias("average_mileage"),
avg(col("price")).alias("average_price"),
countDistinct(col("model")).alias("model_count")).filter("model_count >=10")
.orderBy(col("average_mileage").desc).limit(5).show()
```



```
val cars_1 =cars_cols
.filter("manufacture_year >= 2010")
.filter("price<40000")
.filter("fuel_type == \"gasoline\" ")
.groupBy(col("maker"))
.agg(avg(col("mileage")).alias("average_mileage"),
avg(col("price")).alias("average_price"),
countDistinct(col("model")).alias("model_count")).filter("model_count >=10")
.orderBy(col("average_mileage").desc).limit(5).show()

// Exiting paste mode, now interpreting.

+-------+-----------------+------------------+-----------+
|  maker|  average_mileage|     average_price|model_count|
+-------+-----------------+------------------+-----------+
|  lexus|72891.99539170507|23742.562211981567|         17|
|  volvo|68385.20103092784|18400.876288659794|         12|
|  dodge|62191.93043478261|26751.791304347826|         10|
|hyundai|59733.19197446421|13453.377108983128|         15|
|    kia|58734.32080536913|13621.995973154362|         11|
+-------+-----------------+------------------+-----------+
```

Top 5 best makers for luxury second hand sale filtered by gasoline.

QUERY:

```
val cars_1 =cars_cols
.filter("manufacture_year >= 2010")
.filter("price>=40000")
.filter("fuel_type == \"gasoline\" ")
.filter("engine_power>=150")
.groupBy(col("maker"))
.agg(avg(col("mileage")).alias("average_mileage"),
avg(col("price")).alias("average_price"))
.orderBy(col("average_mileage").desc).limit(5).show()
```

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

+-------+------------------+------------------+
|  maker|   average_mileage|     average_price|
+-------+------------------+------------------+
| hummer|           60000.0|           59519.0|
|porsche|52491.237768240346| 72656.44978540772|
|  lexus|          42916.25|          47127.25|
|    bmw| 39686.90476190476|59592.409523809525|
|bentley|          36014.45|        121498.475|
+-------+------------------+------------------+
```