3. Data Profiling

3.1 Understanding the Dataset

```
CarSales_Data.shape                                        # This will print the number of rows and comlumns of the Data Fram
```

```
(9576, 10)
```

CarSales_Data has **9576 rows** and **10 columns.**

```
CarSales_Data.columns
# This will print the names of all columns.
```

```
Index(['car', 'price', 'body', 'mileage', 'engV', 'engType', 'registration',
       'year', 'model', 'drive'],
      dtype='object')
```

```
CarSales_Data
```

	car	price	body	mileage	engV	engType	registration	year	mode
0	Ford	15500.0	crossover	68	2.5	Gas	yes	2010	Kug
1	Mercedes-Benz	20500.0	sedan	173	1.8	Gas	yes	2011	E Clas
			other	135	5.5	Petrol	yes	2008	CL 55
3	Mercedes-Benz	17800.0	van	162	1.8	Diesel	yes	2012	B 18
4	Mercedes-Benz	33000.0	vagon	91	NaN	Other	yes	2013	E Clas
...	...	...	...	...	...	...	...	...	.
9571	Hyundai	14500.0	crossover	140	2.0	Gas	yes	2011	Tucso
9572	Volkswagen	2200.0	vagon	150	1.6	Petrol	yes	1986	Passa B
9573	Mercedes-	18500.0	crossover	180	3.5	Petrol	yes	2008	M

Creating a copy...

```
CarSales_Data.describe()
```

	price	mileage	engV	year
count	9576.000000	9576.000000	9142.000000	9576.000000
mean	15633.317316	138.862364	2.646344	2006.605994
std	24106.523436	98.629754	5.927699	7.067924
min	0.000000	0.000000	0.100000	1953.000000
25%	4999.000000	70.000000	1.600000	2004.000000
50%	9200.000000	128.000000	2.000000	2008.000000
75%	16700.000000	194.000000	2.500000	2012.000000
max	547800.000000	999.000000	99.990000	2016.000000

```
CarSales_Data.describe(include="all")
```

	car	price	body	mileage	engV	engType	registration	year	model	drive
count	9576	9576.000000	9576	9576.000000	9142.000000	9576	9576	9576.000000	9576	9065
unique	87	NaN	6	NaN	NaN	4	2	NaN	863	3
top	Volkswagen	NaN	sedan	NaN	NaN	Petrol	yes	NaN	E-Class	front
freq	936	NaN	3646	NaN	NaN	4379	9015	NaN	199	5188
mean	NaN	15633.317316	NaN	138.862364	2.646344	NaN	NaN	2006.605994	NaN	NaN

```
CarSales_Data.sort_values(by=['price'],ascending= False).head(10)
```

	car	price	body	mileage	engV	engType	registration	year	model	drive
7621	Bentley	547800.0	sedan	0	6.75	Petrol	yes	2016	Mulsanne	rear
7914	Bentley	499999.0	crossover	0	6.00	Petrol	yes	2016	Bentayga	full
1611	Bentley	499999.0	crossover	0	6.00	Petrol	yes	2016	Bentayga	full
4134	Bentley	449999.0	crossover	0	6.00	Petrol	yes	2016	Bentayga	full
4325	Mercedes-Benz	300000.0	sedan	68	6.00	Petrol	yes	2011	S 600	NaN
5849	Mercedes-Benz	300000.0	other	37	5.00	Petrol	yes	2012	G 500	full
1891	Mercedes-Benz	295000.0	sedan	29	6.00	Petrol	yes	2011	S 600	rear
2165	Mercedes-Benz	295000.0	sedan	29	6.00	Petrol	yes	2011	S-Guard	rear
8205	Land Rover	285000.0	crossover	0	5.00	Petrol	yes	2016	Range Rover	full
1478	Bentley	259000.0	sedan	0	6.00	Petrol	yes	2014	Flying Spur	full

```
CarSales_Data.groupby('car')['price'].count().sort_values(ascending=False)
```

Creating a copy...

```
---------------------------------------------------------------------------
                                          Traceback (most recent call last)
<ipython-input-1-7b9c0be95ef8> in <cell line: 1>()
----> 1 CarSales_Data.groupby('car')['price'].count().sort_values(ascending=False)

NameError: name 'CarSales_Data' is not defined
```

SEARCH STACK OVERFLOW

```
CarSales_Data['car'].value_counts().head()0
```

```
Volkswagen       936
Mercedes-Benz    921
BMW              694
Toyota           541
VAZ              489
Name: car, dtype: int64
```

```
CarSales_Data['car'].value_counts(normalize=True) * 100
```

```
Volkswagen       9.774436
Mercedes-Benz    9.617794
BMW              7.247285
Toyota           5.649541
VAZ              5.106516
                   ...
ZX               0.010443
Other-Retro      0.010443
Mercury          0.010443
Maserati         0.010443
Buick            0.010443
Name: car, Length: 87, dtype: float64
```

It has been observed that top 3 selling cars are : Volkswagen , Mercedes-Benz & BMW

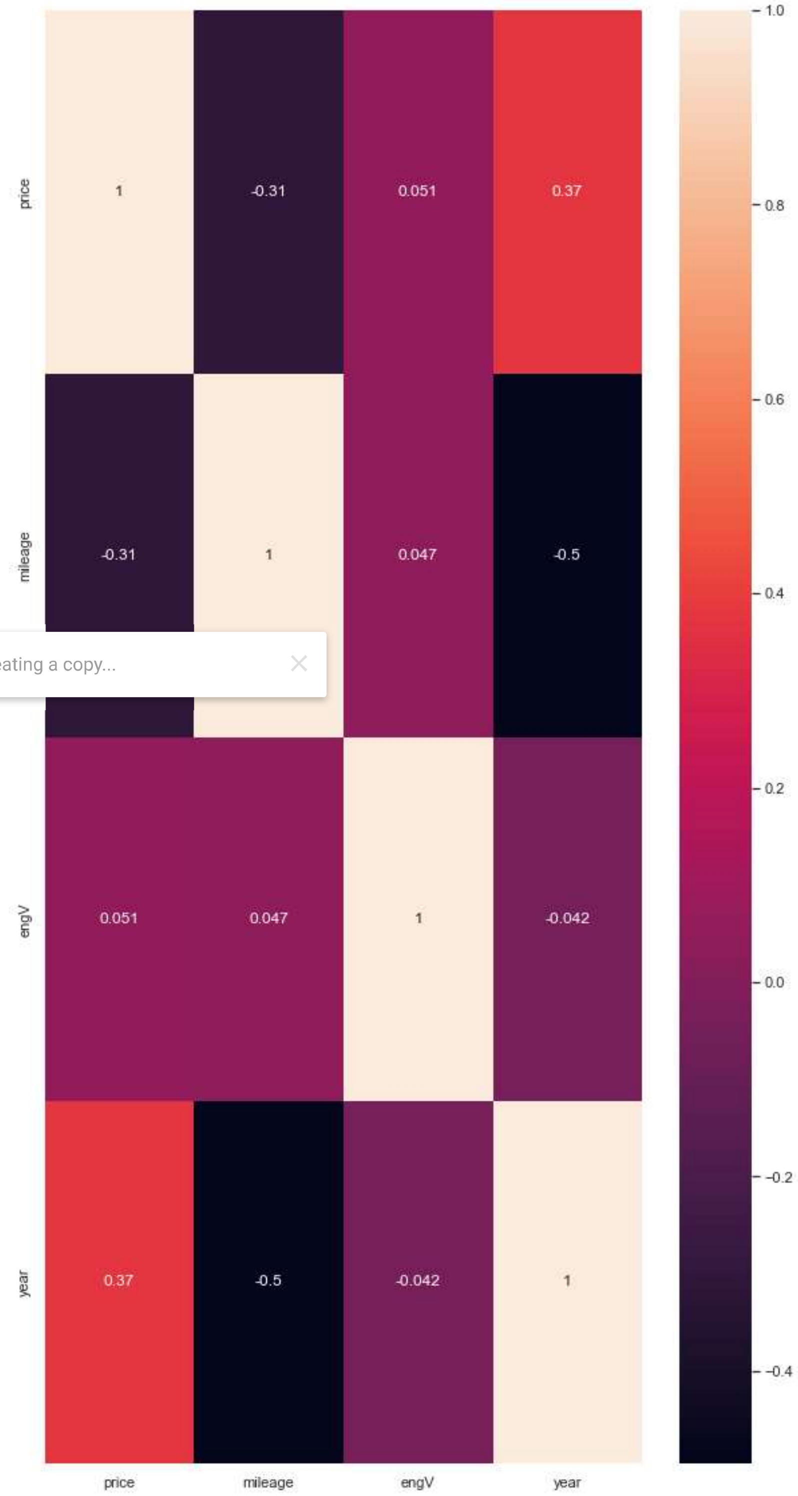
```
CarSales_Data.corr()
```

	price	mileage	engV	year
price	1.000000	-0.312415	0.051070	0.370379
mileage	-0.312415	1.000000	0.047070	-0.495599
engV	0.051070	0.047070	1.000000	-0.042251
year	0.370379	-0.495599	-0.042251	1.000000

Double-click (or enter) to edit

```
import seaborn as sns                                   # Provides a high level interface for drawing attractive and informati
sns.set()
plt.subplots(figsize=(10,20))
sns.heatmap(CarSales_Data.corr(),annot=True)
```

```
<AxesSubplot:>
```



Creating a copy...

```
sns.boxplot(data=CarSales_Data.engV)
```

```
<AxesSubplot:>
```

```
..0tail(5)

# This will print the last n rows of the Data Frame
```

	car	price	body	mileage	engV	engType	registration	year	model	drive
9571	Hyundai	14500.0	crossover	140	2.0	Gas	yes	2011	Tucson	front
9572	Volkswagen	2200.0	vagon	150	1.6	Petrol	yes	1986	Passat B2	front
9573	Mercedes-Benz	18500.0	crossover	180	3.5	Petrol	yes	2008	ML 350	full
9574	Lexus	16999.0	sedan	150	3.5	Gas	yes	2008	ES 350	front
9575	Audi	22500.0	other	71	3.6	Petrol	yes	2007	Q7	full

```
CarSales_Data.info()                                    # This will give Index, Datatype and Memory information
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9576 entries, 0 to 9575
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   car           9576 non-null   object
 1   price         9576 non-null   float64
 2   body          9576 non-null   object
 3   mileage       9576 non-null   int64
 4   engV          9142 non-null   float64
 5   engType       9576 non-null   object
 6   registration  9576 non-null   object
 7   year          9576 non-null   int64
                                l   object
                                l   object
dtypes: float64(2), int64(2), object(6)
memory usage: 748.2+ KB
```

Creating a copy...

```
CarSales_Data.isnull().sum()
```

```
car               0
price             0
body              0
mileage           0
engV            434
engType           0
registration      0
year              0
model             0
drive           511
dtype: int64
```

From the above output we can see that **engV** and **drive** columns contains **maximum null values**. We will see how to deal with them.

1. Fill missing
2. Sort()according to price (Asending)
3. Group via drive
4. Dummy

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
pip install pandas_profiling    ##Installing pandas_profiling packages
```

```
Note: you may need to restart the kernel to use updated packages.
ERROR: Invalid requirement: '#Installing'
```

Now performing **pandas profiling** to understand data better.

```
profile = pandas_profiling.ProfileReport(CarSales_Data)
profile
```

```
Summarize dataset:   0%|          | 0/5 [00:00<?, ?it/s]
Generate report structure:   0%|          | 0/1 [00:00<?, ?it/s]
Render HTML:   0%|          | 0/1 [00:00<?, ?it/s]
```

Overview

Dataset statistics

Number of variables	10
Number of observations	9576
Missing cells	945
Missing cells (%)	1.0%
Duplicate rows	110
Duplicate rows (%)	1.1%
Total size in memory	748.2 KiB
Average record size in memory	80.0 B

Variable types

Categorical	4
Numeric	4
Boolean	1
Unsupported	1

Alerts

Dataset has 110 (1.1%) duplicate rows	Duplicates
7 distinct values	High cardinality
`price` is highly overall correlated with `year`	High correlation
`mileage` is highly overall correlated with `year`	High correlation
`year` is highly overall correlated with `price` and 1 other fields (price, mileage)	High correlation
`car` is highly overall correlated with `drive`	High correlation
`body` is highly overall correlated with `drive`	High correlation

Creating a copy...

```
profile.to_file(output_file="CarSales_before_preprocessing.html")
```

```
Export report to file:   0%|          | 0/1 [00:00<?, ?it/s]
```

3.3 Preprocessing

- Dealing with duplicate rows
  - Find number of duplicate rows in the dataset.
  - Print the duplicate entries and analyze.
  - Drop the duplicate entries from the dataset.

```
miss1 = CarSales_Data.isnull().sum()
miss = (CarSales_Data.isnull().sum()/len(CarSales_Data))*100
miss_data = pd.concat([miss1,miss],axis=1, keys=['Total', '%'])
print(miss_data)
```

```
              Total         %
car               0  0.000000
price             0  0.000000
body              0  0.000000
mileage           0  0.000000
engV            434  4.532164
engType           0  0.000000
registration      0  0.000000
year              0  0.000000
model             0  0.000000
drive           511  5.336257
```

```
CarSales_Data.duplicated().sum()
```

```
113
```

```
CarSales_Data.loc[CarSales_Data.duplicated(), :]
```

	car	price	body	mileage	engV	engType	registration	year	model	drive
18	Nissan	16600.0	crossover	83	2.0	Petrol	yes	2013	X-Trail	full
42	Mercedes-Benz	20400.0	sedan	190	1.8	Gas	yes	2011	E-Class	rear
70	Mercedes-Benz	0.0	crossover	0	3.0	Diesel	yes	2016	GLE-Class	full
86	Toyota	103999.0	crossover	0	4.5	Diesel	yes	2016	Land Cruiser 200	full
98	Mercedes-Benz	20400.0	sedan	190	1.8	Gas	yes	2011	E-Class	rear
...	...	...	...	...	...	...	...	...	...	...
9156	Volkswagen	15700.0	sedan	110	1.8	Petrol	yes	2011	Passat B7	front
9163	Mercedes-Benz	20500.0	sedan	222	5.5	Petrol	yes	2006	S 500	rear
9164	VAZ	3900.0	hatch	121	1.4	Petrol	yes	2008	1119	front
9169	Hyundai	12900.0	crossover	49	2.7	Petrol	yes	2008	Tucson	full

```
CarSales_Data_copy.drop_duplicates(inplace=True)
```

```
CarSales_Data_copy.loc[CarSales_Data.duplicated(), :]
```

car	price	body	mileage	engV	engType	registration	year	model	drive

Creating a copy...

	car	price	body	mileage	engV	engType	registration	year	model	drive
0	Ford	15500.0	crossover	68	2.5	Gas	yes	2010	Kuga	full
1	Mercedes-Benz	20500.0	sedan	173	1.8	Gas	yes	2011	E-Class	rear
2	Mercedes-Benz	35000.0	other	135	5.5	Petrol	yes	2008	CL 550	rear
3	Mercedes-Benz	17800.0	van	162	1.8	Diesel	yes	2012	B 180	front
4	Mercedes-Benz	33000.0	vagon	91	NaN	Other	yes	2013	E-Class	NaN
...	...	...	...	...	...	...	...	...	...	...
9571	Hyundai	14500.0	crossover	140	2.0	Gas	yes	2011	Tucson	front
9572	Volkswagen	2200.0	vagon	150	1.6	Petrol	yes	1986	Passat B2	front
9573	Mercedes-Benz	18500.0	crossover	180	3.5	Petrol	yes	2008	ML 350	full
9574	Lexus	16999.0	sedan	150	3.5	Gas	yes	2008	ES 350	front
9575	Audi	22500.0	other	71	3.6	Petrol	yes	2007	Q7	full

9463 rows × 10 columns

```
b=CarSales_Data_copy["drive"].mode()
b
```

```
0    front
Name: drive, dtype: object
```

```
CarSales_Data_copy["drive"]=CarSales_Data_copy["drive"].fillna("front")
CarSales_Data_copy.isnull().sum()
```

```
car             0
price           0
body            0
mileage         0
engV            0
engType         0
registration    0
year            0
model           0
drive           0
dtype: int64
```

```
print(CarSales_Data_copy.duplicated().sum())
```

```
0
```

```
CarSales_Data.loc[CarSales_Data.duplicated(keep=False), :]
```

	car	price	body	mileage	engV	engType	registration	year	model	drive
0	Ford	15500.0	crossover	68	2.5	Gas	yes	2010	Kuga	full
1	Mercedes-Benz	20500.0	sedan	173	1.8	Gas	yes	2011	E-Class	rear
3	Mercedes-Benz	17800.0	van	162	1.8	Diesel	yes	2012	B 180	front
5	Nissan	16600.0	crossover	83	2.0	Petrol	yes	2013	X-Trail	full
7	Renault	10500.0	vagon	185	1.5	Diesel	yes	2011	Megane	front
...	...	...	...	...	...	...	...	...	...	...
9156	Volkswagen	15700.0	sedan	110	1.8	Petrol	yes	2011	Passat B7	front
9163	Mercedes-Benz	20500.0	sedan	222	5.5	Petrol	yes	2006	S 500	rear
9164	VAZ	3900.0	hatch	121	1.4	Petrol	yes	2008	1119	front
9169	Hyundai	12900.0	crossover	49	2.7	Petrol	yes	2008	Tucson	full
9477	BMW	77777.0	sedan	8	4.4	Petrol	yes	2014	750	full

201 rows × 10 columns

```
CarSales_Data.drop_duplicates(keep='first').shape
```

```
(9463, 10)
```


- Dealing with missing values
    - **434** missing entries of **engV.** Replace it with **median** value of engV from the same Car and body group of cars.
    - **511** missing entries of **drive.** Replace it with **most common** value of drive from the same Car and body group of cars.
    - Drop entries having **price** is 0 or less than 0.

```
CarSales_Data_copy.groupby(['car', 'body'])['engV'].head()
```

```
0        2.5
1        1.8
2        5.5
3        1.8
4        2.3
        ...
9499     3.7
9501     1.2
9508    19.0
9539     1.5
9566     NaN
Name: engV, Length: 1018, dtype: float64
```

```
CarSales_Data_copy['engV'] = CarSales_Data_copy.groupby(['car', 'body'])['engV'].transform(lambda x: x.fillna(x.median()))
```

Now let's check if the missing values of **engV** has been replaced.

```
CarSales_Data_copy.isnull().sum()
```

```
car              0
price            0
body             0
mileage          0
engV            10
engType          0
registration     0
year             0
model            0
drive            0
dtype: int64
```

424 missing values of **engV** has been replaced however, still **10** entries are left as missing. Let's see the missing value data.

```
CarSales_Data_copy[CarSales_Data_copy.engV.isnull()]
```

	car	price	body	mileage	engV	engType	registration	year	model	drive
319	Tesla	58000.0	hatch	52	NaN	Other	yes	2013	Model S	front
1437	Tesla	178500.0	crossover	0	NaN	Other	yes	2016	Model X	full
2486	Tesla	185000.0	crossover	1	NaN	Other	yes	2016	Model X	full
5084	GAZ	0.0	crossover	1	NaN	Petrol	yes	1963	69	full
6773	UAZ	3000.0	other	1	NaN	Other	yes	1985	3303	full
8569	Tesla	176900.0	crossover	0	NaN	Other	yes	2016	Model X	full
8824	Fisker	0.0	other	100	NaN	Other	yes	2001	Karma	front
8905	Changan	6028.0	crossover	101	NaN	Other	yes	2005	Ideal	front
9360	Barkas	5500.0	van	80	NaN	Petrol	yes	2015	B1000	front
9566	UAZ	850.0	van	255	NaN	Other	yes	1981	3962	front

Replacing NaN values of **drive** with most common values of drive from Car and body group.

```
def f(x):
    if x.count()<=0:
        return np.nan
    return x.value_counts().index[0]

CarSales_Data_copy['drive'] = CarSales_Data_copy['drive'].fillna(CarSales_Data_copy.groupby(['car','body'])['drive'].transform(f))
#CarSales_Data[CarSales_Data.drive.isnull()]
```

```
CarSales_Data_copy[CarSales_Data_copy.drive.isnull()]
```

Creating a copy...

gV	engType	registration	year	model	drive

Let's check the count of NaN values of **engV** and **drive**.

```
CarSales_Data_copy.isnull().sum()
```

```
car              0
price            0
body             0
mileage          0
engV            10
engType          0
registration     0
year             0
model            0
drive            0
dtype: int64
```

Dropping remaining NaN values of **engV** and **drive**.

```
CarSales_Data_copy.dropna(subset=['engV'],inplace=True)
CarSales_Data_copy.dropna(subset=['drive'],inplace=True)
CarSales_Data_copy.isnull().sum()
```

```
car             0
price           0
body            0
mileage         0
engV            0
engType         0
registration    0
year            0
model           0
drive           0
dtype: int64
```

Dropping entries with **price <= 0** .

```
CarSales_Data_copy = CarSales_Data_copy.drop(CarSales_Data_copy[CarSales_Data_copy.price <= 0 ].index)
```

```
CarSales_Data_copy.price[CarSales_Data_copy.price ==0].count()
```

```
0
```

```
b = CarSales_Data_copy["mileage"].median()
CarSales_Data_copy["mileage"]=CarSales_Data_copy["mileage"].replace(0,b)
```

```
CarSales_Data_copy[CarSales_Data_copy.mileage == 0]
```

car	price	body	mileage	engV	engType	registration	year	model	drive

```
CarSales_Data_copy
```

	car	price	body	mileage	engV	engType	registration	year	model	drive
0	Ford	15500.0	crossover	68	2.5	Gas	yes	2010	Kuga	full
1	Mercedes-Benz	20500.0	sedan	173	1.8	Gas	yes	2011	E-Class	rear
2	Mercedes-Benz	35000.0	other	135	5.5	Petrol	yes	2008	CL 550	rear
3	Mercedes-Benz	17800.0	van	162	1.8	Diesel	yes	2012	B 180	front
4	Mercedes-Benz	33000.0	vagon	91	2.3	Other	yes	2013	E-Class	front
...	...	...	...	...	...	...	...	...	...	...
9571	Hyundai	14500.0	crossover	140	2.0	Gas	yes	2011	Tucson	front
9572	Volkswagen	2200.0	vagon	150	1.6	Petrol	yes	1986	Passat B2	front
9573	Mercedes-Benz	18500.0	crossover	180	3.5	Petrol	yes	2008	ML 350	full
9574	Lexus	16999.0	sedan	150	3.5	Gas	yes	2008	ES 350	front
9575	Audi	22500.0	other	71	3.6	Petrol	yes	2007	Q7	full

9215 rows × 10 columns

Creating a copy... ×

```
profile_cleaned = pandas_profiling.ProfileReport(CarSales_Data_copy)
profile_cleaned.to_file(output_file="CarSales_post_preprocessing.html")
```

```
Summarize dataset:   0%|          | 0/5 [00:00<?, ?it/s]
Generate report structure:   0%|          | 0/1 [00:00<?, ?it/s]
Render HTML:   0%|          | 0/1 [00:00<?, ?it/s]
Export report to file:   0%|          | 0/1 [00:00<?, ?it/s]
```

The data are processed now. The dataset doesnot contain missing and zero values. The pandas profiling report generated after processing the data giving us more clear data. We can compare the two reports.

4. Questions