Student Name: Joshua Cayetano Student ID: s3722151

Data Preparation

With Task 1 we needed to combine 11 separate CSV files together into 1 file called "cleaned_car_buyers.CSV". I would then need to later get rid of any potential issues such as white spaces or commas which will be explained later.

- I first started downloading all the associated files and putting them into 1 directory so I could access them later in Jupyter Notebook.
- I would import 'pandas' and matplot to read between different data structures and plot graphs from my data later.
- I read all the CSV files and later combined them into one file ("Mainfile") by creating a new column for each variable and calling upon them in different CSV files. "*Task – Read Files"*
- We do not need to worry about bad lines, as there is no extra value in the CSV value. *"Task – bad lines"*

Identification of issues

I looked at the contents of each CSV file. I determined potential issues were white spaces, commas, missing values (NaN), impossible values and outliers.

NOTE: *I used the sort-function in Excel from Microsoft (2022) as well as functions in Python to identify these values.*

For each File:

- Manufacturer: This is a nominal data type. There are 52 car manufacturers. All are valid.
- Model: This is a nominal data type. There is only one white space which we need to remove. ***I'm going to assume that all models are valid for the purpose of the assignment. It would be intensive, time consuming and outside the scope of the course to create another file containing all model types then use it to compare missing values. The reason we would compare our data to the models that are allowed is because there could be a filled in observation which does not match the specifications (E.g. Grease Lightning).***
- Price: This is a discrete data type. There are 20 values that are above the price range. We have 2 values less then 0. We also have 5 blank observations which we should remove or change.
- Transmission: This is a continuous data type. There are 129 observations that are <=0 which we must remove. We can assume this is because they were just bought buy haven't been used yet. We also have 2 values over 10 which we need to remove. We also have 4 white space values which we must remove or change.
- Power: This is a continuous data type. We have 4 values which are greater than 500 which we need to remove. We have 3 observations which are less than 0 which we need to remove. We also again see 5 observations which are blank and we need to remove.

- Engine CC: This is a continuous data type. There are 17 values which are equal to 0 which violates our conditions which we need to remove.
- Fuel: This is a nominal data type. There are 5 values which are empty which we need to remove.
- Male: This is a discrete data type. There are 4 whitespaces which we need to replace with 0. We also must remove the commas.
- Female: This is a discrete data type. There are 4 whitespaces which we need to replace with 0. We also must remove the commas.
- Unknown: This is a discrete data type. There are 5 whitespaces which we need to replace with 0. We also must remove the commas.
- Total: The data type is discrete. There are 4 whitespaces which we need to remove or replace with 0. We also must remove the commas.

Python ways to check values:
"Task - Look at blank excel value." **+** "Task - Python Check"

Removal of issues

Error 1: Redundant Index

I had originally used the "Unnamed: 0" column to link the tables together. However, it became redundant as there was an automatic index created. So, what I did to solve this is delete it using the drop function. By removing this we also remove one of the major sources of error in Machine Learning – "Leakage".

```
#2.We the clean the file
 #Rename the unamed column then delete it - It is redundant information.
list(Mainfile)
Mainfile.rename(columns = {'Unnamed: 0': 'Index'}, inplace =True)
Mainfile = Mainfile.drop(columns=['Index'])
Mainfile
```

	Manufacturer	Model	Price	Transmission	Power	Engine CC	Fuel	Female	Male	Unknown	Total
0	Ford	Focus	30.619322	5.966102	-94.033898	1497.169492	petrol	422731	814172	56,487	1293390
1	Ford	Fiesta	18.532143	5000.714286	68.571429	1166.142857	petrol	631666	554879	54,057	1240602
2	Volkswagen	Golf	31.242154	6.164835	-89.461538	1537.406593	petrol	310604	483216	47,563	841383

"Task – Drop Index"

Error 2: Mistakes during data entry

In this case it was being the use of commas in our integer values (Male, Female, Unknown, Total) and us having the need to round the decimals for, Price, Transmission, Power, and Engine. We need to remove commas to sum our values later and do calculations. We round our decimals to make it easier to read the data and standardise the variables.
I aimed to do this by updating each column and using the round and string replace command.
"Task – Rounding and comma removal"

Error 3: Redundant White Space

We need to remove white space as while it is easy to remove it is difficult to spot and can cause errors. I aimed to solve this by using the strip function which would remove any white

space for any string characters. This can also be used on discrete values. It can be noted that around 50 observations were removed based on the shape outputted.
"Task – White Space"
"Task – Python Check"

Error 4: Impossible values/ outliers

Outliers can affect our data results in the data modelling process later. We also have to remove those that cannot be possible as for instance a car could not be priced over $650. I used drop function and used as references, examples from StackOverflow (2013) as well as examples from Data Science Parichay (2022).
"Task – Impossible Values"
"Task - Checking drop function"
"Task – Outliers"

Error 5: Missing values

I wanted to remove values that were empty as it could affect my descriptive statistics and modelling in the next section by skewing the results. I first attempted to find empty values via "Task – Impossible values" but found there were no more empty areas. I also just in case aimed to fill in empty spaces with the fill.mean function.
"Task – Impossible values".
"Task – Attempting to fill in missing spaces"

After cleaning my data:
Finally, I wrote the file to a new CSV file with the following code:
Mainfile.to_csv('**cleaned_car_buyers.csv**', index=False)

Data Exploration

Task 2.1

The data columns we are using are Male, Female, Unknown, Total and Model.

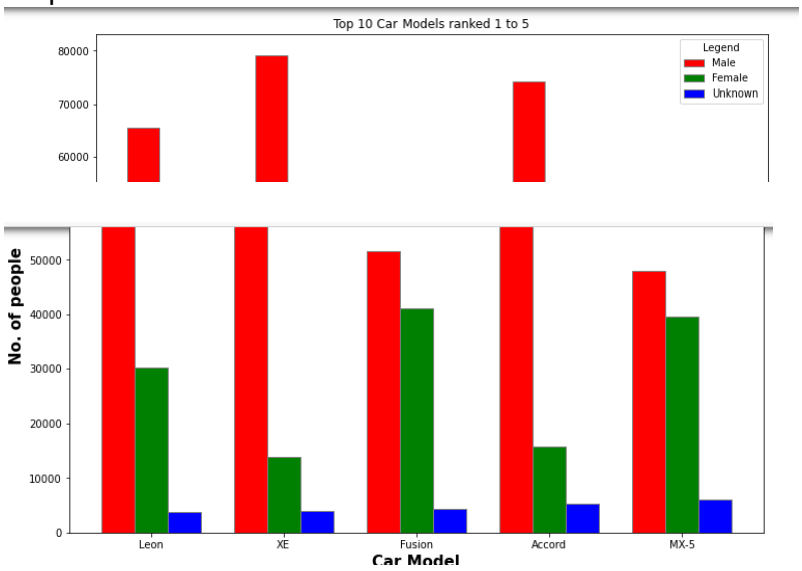
- Model: This is a nominal data type. We represent this best with a pie chart, bar chart or stacked column. Our level of measurement is Frequency and Proportions.
- Male, Female, Unknown, Total: These are all Discrete data types. We represent these with bar charts or histograms. This also looks at frequency and proportions for level of measurements.

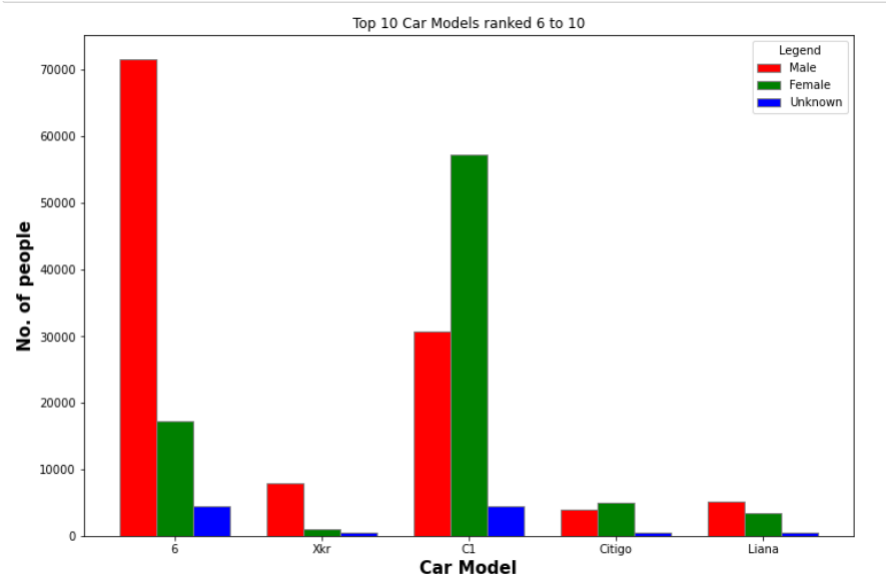
While I identified that I need to have my data for Model into a Bar Chart, I realised I need to manipulate the data into the top 10 Models. This is because one manufacturer can have many car models, but a model can only belong to one manufacturer. I also realised I needed to leave the values of gender by itself instead of summing it, based on the large quantity of people involved and with the numbers already being provided in the total column.
"Task 2.1 – Model conversion needed"

I then obtained my data as shown through "Task 2.1 – Top 10 Cars by Total".

NOTE: While the firsts 6 observations are correct in being sorted in descending order, 3 observations ranging at the 9000 total amount, seem to be outliers. This could be because I grouped the data by Model. I did not group by Manufacturer as Manufacturer could appear twice for 2 different car models while also compromising the descending order of Total.

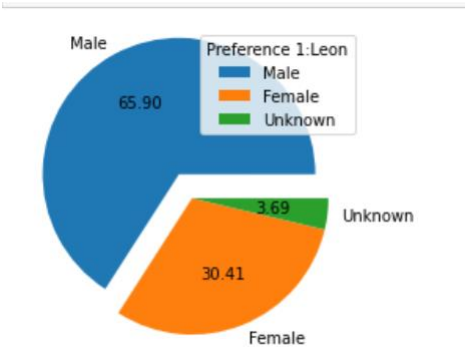
I first mapped the overall data of Top 10 cars into a bar chart. **NOTE:** Python cut off the output.



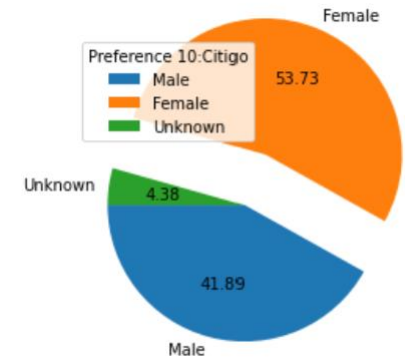


From our Bar Chart data, we can determine that for most car models, majority of them are mostly used by Men (e.g. Leon, 6 and the Accord). People with unknown gender are primarily small stakeholders in the types of car models they get. We can ascertain that Females like the C1 model as Females outnumber the males who buy the car model.

For the bar charts, it would be meaningful to look at only the 1st and 10th observations based on the total of number of people who bought the car model. From the 1st observation, we can ascertain that the people who bought the Leon Model, 65.90% of which were 30.91% of Females and 3.69% of people of Unknown Gender.



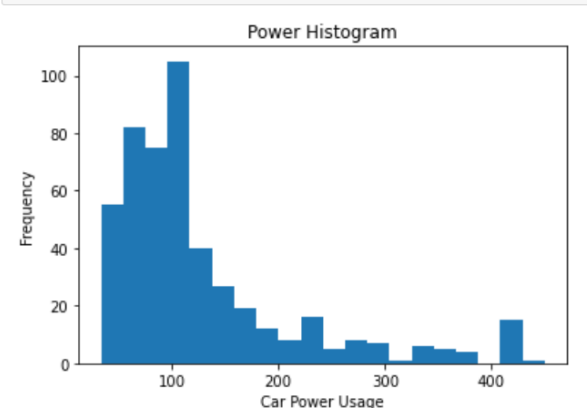
For the 10th observation, we can observe that of those that bought the Citigo Model, 53.73% were Females, 41.89% were Male and 4.38% were those of an Unknown gender.

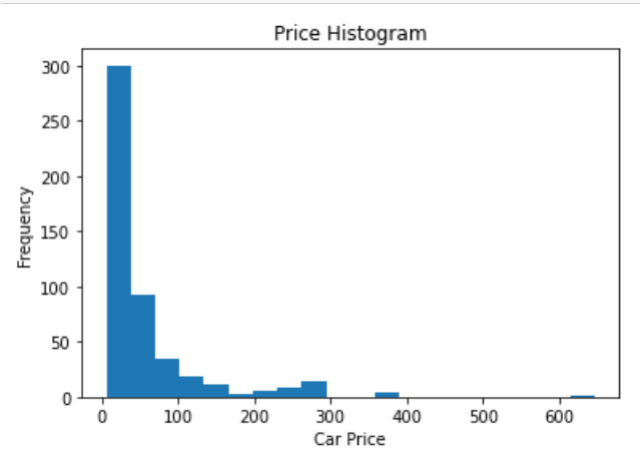


Task 2.2

The data columns that we are using is Manufacturer, Model, Price and Power. The reason I added Model is to give context what the variables Price and Power belong to.
Since we are focusing on Price and Power which are both continuous data types, the level of measurement we use can be either be represented as a bar chart or a histogram. It is preferable to use a histogram in this case as we are dealing with numerical data (Price & Power), and we want to see the shape of the distribution.

From our histograms:





We can see for both histograms, most of the values are on the left side which means the data is positively skewed. For Power the average power for cars would range between 80 – 120. We also note that the observations are fairly spread out with a possible outlier past 400 Car Power. Meanwhile for Car Price, the most predominant price for Cars is Car Prices being approximately $20.

From the histograms we can identify that as an issue there are outliers which can affect our results which can be removed.

However, another issue which isn't drawn from our histograms is that there are some mismatches of Prices and Power in relation to Cars. This can be determined from my appendix "Task 2.2 – Sorted Power". For instance, an Aston-Martin has a higher price of 360.76 and generates 410 for a car. In comparison a Bentleigh Continental has a lower Price of 285.25 that generates more power at 428.20. While some of this could be explained due to the difference of new and older models – this is very ambiguous.

Task 2.3

Here I have decided to explore the relationship between the number of Male Owners to Model, Transmission and Fuel. I would look at the top 5 results.
For the level of measurement:

- Male is a discrete data type, so I need to use either a histogram or a bar chart.
- Transmission is a continuous data type, so I need to use either a histogram or a bar chart.
- Fuel is a nominal data type, and I would normally represent this with a bar chart or pir chart. However, the results all have the fuel type so it would be redundant to visualise it.
- Model is a nominal data type so I would represent this as a pie chart or bar chart.

From the appendix "Task 2.3 – Males sorted", we can determine that the number of males in the top 5 observations all use petrol. Furthermore, the highest transmission rate is 6.44 while the lowest transmission rate is 1.75. The highest number of males is 408,016.

From the appendix "Task 2.3 – Redundant values - Fuel", it can be determined that the L Range has the highest transmission rate from using diesel. However, it must be noted that there are different number of Males for each observation.

Appendix

Task 1: Data preparation

"Task 1 - Python Check"

```
In [8]: Mainfile ['Price'] = (file2['Price'])

In [9]: Mainfile ['Transmission'] = (file3['Transmission'])

In [10]: Mainfile ['Power'] = (file4['Power'])

In [11]: Mainfile ['Engine CC'] = (file5['Engine CC'])

In [12]: Mainfile ['Fuel'] = (file6['Fuel'])

In [13]: Mainfile ['Female'] = (file7['Female'])

In [14]: Mainfile ['Male'] = (file8['Male'])

In [15]: Mainfile ['Unknown'] = (file9['Unknown'])

In [16]: Mainfile ['Total'] = (file10['Total'])

In [17]: #

In [18]: #

In [19]: Mainfile
```

Out[19]:

	Unnamed: 0	Manufacturer	Model	Price	Transmission	Power	Engine CC	Fuel	Female	Male	Unknown	Total
0	0	Ford	Focus	30.619322	5.966102	-94.033898	1497.169492	petrol	422731	814172	56,487	1293390
1	1	Ford	Fiesta	18.532143	5000.714286	68.571429	1166.142857	petrol	631666	554879	54,057	1240602
2	2	Volkswagen	Golf	31.242154	6.164835	-89.461538	1537.406593	petrol	310604	483216	47,563	841383
3	3	Renault	Clio	22.100000	5.615385	75.576923	1219.653846	petrol	312556	241287	28,004	581847
4	4	BMW	320i	47.848370	6.444444	126.111111	1995.777778	petrol	115843	408016	29,125	552984
...	...	...	...	...	...	...	...	...	...	...	...	...
6097	6097	Land-Rover	Defender	108.747195	7.853659	207.609756	2304.975610	diesel	150	1,012	80	1,242
6098	6098	Toyota	RAV4	43.548516	1.354839	137.774193	2261.193548	petrol	482	670	66	1,218
6099	6099	Alfa-Romeo	Spider	55.200000	6.000000	163.500000	2696.500000	petrol	247	790	81	1,118

“Task 1 - Bad Lines”

```
 #This I now can read the file. I can now read the other files.This is from the Week 2 slides - Missing Values
file1 = pd.read_csv('Model.csv',sep=',', decimal='.',error_bad_lines=False)
file2 = pd.read_csv('Price.csv',sep=',', decimal='.')
file3 = pd.read_csv('Transmission.csv',sep=',', decimal='.')
file4 = pd.read_csv('Power.csv',sep=',', decimal='.')
file5 = pd.read_csv('Engine CC.csv',sep=',', decimal='.')
file6 = pd.read_csv('Fuel.csv',sep=',', decimal='.')
file7 = pd.read_csv('Female.csv',sep=',', decimal='.')
file8 = pd.read_csv('Male.csv',sep=',', decimal='.')
file9 = pd.read_csv('Unknown.csv',sep=',', decimal='.')
file10 = pd.read_csv('Total.csv',sep=',', decimal='.')
```

```
C:\Users\User\anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3444: FutureWarning: The error_bad_lines argument ha
s been deprecated and will be removed in a future version.


  exec(code_obj, self.user_global_ns, self.user_ns)
```

“Task 1 - Drop Index”

```
#2.We the clean the file
 #Rename the unamed column then delete it - It is redundant information.
list(Mainfile)
Mainfile.rename(columns = {'Unnamed: 0': 'Index'}, inplace =True)
Mainfile = Mainfile.drop(columns=['Index'])
Mainfile
```

	Manufacturer	Model	Price	Transmission	Power	Engine CC	Fuel	Female	Male	Unknown	Total
0	Ford	Focus	30.619322	5.966102	-94.033898	1497.169492	petrol	422731	814172	56,487	1293390
1	Ford	Fiesta	18.532143	5000.714286	68.571429	1166.142857	petrol	631666	554879	54,057	1240602
2	Volkswagen	Golf	31.242154	6.164835	-89.461538	1537.406593	petrol	310604	483216	47,563	841383

“Task 1 - Rounding and comma removal”

```
#Make sure to round all values with decimals to 4 places.
Mainfile['Price'] = Mainfile['Price'].round(2)
Mainfile['Transmission'] = Mainfile['Transmission'].round(2)
Mainfile['Power'] = Mainfile['Power'].round(2)
Mainfile['Engine CC'] = Mainfile['Engine CC'].round(2)
```

```
#Getting rid of comma's in our decimal values
Mainfile['Female'] = Mainfile['Female'].str.replace(',', '')
Mainfile['Male'] = Mainfile['Male'].str.replace(',', '')
Mainfile['Unknown'] = Mainfile['Unknown'].str.replace(',', '')
Mainfile['Total'] = Mainfile['Total'].str.replace(',', '')
Mainfile
```

“Task 1 - White Space”

In [20]:
```
#We strip white spaces with values that use strings.
 #This doesn't work on Price, Power and Engine CC.
 #We notice that we cannot remove reundant spaces of values
Mainfile['Manufacturer'] = Mainfile['Manufacturer'].str.strip
Mainfile['Model'] = Mainfile['Model'].str.strip()
#Mainfile['Price'] = Mainfile['Price'].str.strip()
#Mainfile['Power'] = Mainfile['Power'].str.strip()
#Mainfile['Engine CC'] = Mainfile['Engine CC'].str.strip()
Mainfile['Fuel'] = Mainfile['Fuel'].str.strip()
Mainfile['Male'] = Mainfile['Male'].str.strip()
Mainfile['Female'] = Mainfile['Female'].str.strip()
Mainfile['Unknown'] = Mainfile['Unknown'].str.strip()
Mainfile['Total'] = Mainfile['Total'].str.strip()
Mainfile.shape
```

Out[20]: (5961, 11)

“Task 1 - Impossible values”

In [29]:
```
#Impossible values
 #We identify NULL values and try to substitute them.
mask_TransmissionNULL = Mainfile['Transmission']=='NaN'
mask_TransmissionNULL.sort_values(ascending=True)
mask_TransmissionNULL
```

Out[29]:
```
3       False
4       False
5       False
6       False
7       False
        ...
6097    False
6098    False
6099    False
6100    False
6101    False
Name: Transmission, Length: 5961, dtype: bool
```

“”Task 1 - Outliers””

In [22]:
```
Mainfile.loc[Mainfile['Engine CC'] < 0]
Mainfile.loc[Mainfile['Engine CC'] > 6500]
#Inspiration from https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.loc.html
```

Out[22]:

Manufacturer	Model	Price	Transmission	Power	Engine CC	Fuel	Female	Male	Unknown	Total

"Task 1 - Fill in missing spaces"

In [38]:
```
# Attempting to fill in empty spaces 2
Mainfile['Price'].fillna(Mainfile['Price'].mean(axis=0))
Mainfile['Transmission'].fillna(Mainfile['Transmission'].mean(axis=0))
Mainfile['Power'].fillna(Mainfile['Power'].mean(axis=0))
Mainfile['Engine CC'].fillna(Mainfile['Engine CC'].mean(axis=0))
#Mainfile['Male'].fillna(Mainfile['Male'].mean(axis=0))
```

Out[38]:
```
3       1219.65
4       1995.78
5       1408.06
6       1631.50
7       1998.50
          ...
6097    2304.98
6098    2261.19
6099    2696.50
6100    2254.00
6101    1817.32
Name: Engine CC, Length: 5961, dtype: float64
```

"Task 1 - Previous result of filling empty values"

In [10]:
```
# Test code
Mainfile["Transmission"].fillna(-1)
Mainfile.sort_values(['Transmission'],ascending=True)
#Mainfile = Mainfile.drop(Mainfile[(Mainfile.Transmission <= 0.0)].index)
#Mainfile = Mainfile.drop(Mainfile[(Mainfile.Transmission > 10.0)].index)
```

Out[10]:

	Manufacturer	Model	Price	Transmission	Power	Engine CC	Fuel	Female	Male	Unknown	Total
2314	Lexus	GS	78.48	0.89	224.78	3196.56	petrol	93	941	72	1106
1931	Lexus	GS	78.48	0.89	224.78	3196.56	petrol	139	1283	92	1514
1543	Lexus	GS	78.48	0.89	224.78	3196.56	petrol	176	1468	112	1756
743	Lexus	GS	78.48	0.89	224.78	3196.56	petrol	233	1857	149	2239
340	Lexus	GS	78.48	0.89	224.78	3196.56	petrol	252	2034	166	2452
...	...	...	...	...	...	...	...	...	...	...	...
3406	Land-Rover	Range	107.67	8.28	196.34	2521.09	diesel	8608	44779	2190	55577
23	Ford	Focus	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
32	Seat	Ibiza	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
54	Toyota	Aygo	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
65	Ford	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

5961 rows × 11 columns

"Task 1 - Look at blank excel value."

In [34]:
```
#This was a way I could verify in PYTHON an empty field after searching manually. where the observation 23 has an empty field.
test = Mainfile['Fuel']
test[23]
```

"Task 1 - Checking Drop function"

```
# Test code
Mainfile = Mainfile.drop(Mainfile[(Mainfile.Price < 0.0) & (Mainfile.Price > 650.0)].index)
Mainfile.sort_values(['Price'],ascending=False)
Mainfile


#Mainfile = Mainfile.drop(Mainfile[(Mainfile.Transmission <= 0.0)].index)
#Mainfile = Mainfile.drop(Mainfile[(Mainfile.Transmission > 10.0)].index)
```

	Manufacturer	Model	Price	Transmission	Power	Engine CC	Fuel	Female	Male	Unknown	Total
41	Citroen	C5	290050.26	5.95	0.12	1462.36	petrol	42130	79604	5547	127281
1934	Porsche	Carrera	646.60	6.00	450.00	5733.00	petrol	104	1228	60	1392
5906	Porsche	Carrera	646.60	6.00	450.00	5733.00	petrol	159	1178	45	1382
3930	Porsche	Carrera	646.60	6.00	450.00	5733.00	petrol	136	1455	56	1647
3637	Porsche	Carrera	646.60	6.00	450.00	5733.00	petrol	137	1460	55	1652
...	...	...	...	...	...	...	...	...	...	...	...

In [15]:
```
# Test code
Mainfile = Mainfile.drop(Mainfile[(Mainfile.Power < 0.0)].index)
Mainfile = Mainfile.drop(Mainfile[(Mainfile.Power >500.0)].index)
Mainfile.sort_values(['Power'], ascending=True)

#Mainfile = Mainfile.drop(Mainfile[(Mainfile.Transmission <= 0.0)].index)
#Mainfile = Mainfile.drop(Mainfile[(Mainfile.Transmission > 10.0)].index)
```

Out[15]:

	Manufacturer	Model	Price	Transmission	Power	Engine CC	Fuel	Female	Male	Unknown	Total
4957	Skoda	120	3.33	4.60	15.6	469.60	petrol	1314	3254	100	4668
5387	Skoda	120	3.33	4.60	15.6	469.60	petrol	3519	7990	218	11727
4500	Skoda	120	3.33	4.60	15.6	469.60	petrol	392	1091	38	1521
5593	Skoda	120	3.33	4.60	15.6	469.60	petrol	4801	10763	275	15839
5982	Skoda	120	3.33	4.60	15.6	469.60	petrol	7364	16722	368	24454
...	...	...	...	...	...	...	...	...	...	...	...
23	Ford	Focus	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
32	Seat	Ibiza	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
54	Toyota	Aygo	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
65	Ford	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

```
Mainfile.fillna(0)
Mainfile.sort_values(['Transmission'],ascending=True)

#Mainfile = Mainfile.drop(Mainfile[(Mainfile.Transmission <= 0.0)].index)
#Mainfile = Mainfile.drop(Mainfile[(Mainfile.Transmission > 10.0)].index)
```

Out[31]:

	Manufacturer	Model	Price	Transmission	Power	Engine CC	Fuel	Female	Male	Unknown	Total
2314	Lexus	GS	78.48	0.89	224.78	3196.56	petrol	93	941	72	1106
1931	Lexus	GS	78.48	0.89	224.78	3196.56	petrol	139	1283	92	1514
1543	Lexus	GS	78.48	0.89	224.78	3196.56	petrol	176	1468	112	1756
743	Lexus	GS	78.48	0.89	224.78	3196.56	petrol	233	1857	149	2239
340	Lexus	GS	78.48	0.89	224.78	3196.56	petrol	252	2034	166	2452
...	...	...	...	...	...	...	...	...	...	...	...
3406	Land-Rover	Range	107.67	8.28	196.34	2521.09	diesel	8608	44779	2190	55577
23	Ford	Focus	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
32	Seat	Ibiza	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
54	Toyota	Aygo	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
65	Ford	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

"Task 1 - Python Check"

In [11]:
```
#Other miscellaneous functions - APPENDIX
#1) We check the name of the columns - specifically the unnamed Index.
#file2.columns
#Y = file2[['Unnamed: 0', 'Price']]
#Y

#2) How to look at a specific value
#This was a way I could verify in PYTHON an empty field after searching manually. where the observation 23 has an empty field.
#test = Mainfile['Fuel']
#test[23]

#3) Python way to check
#Alternate way of sorting values to find outliers - Run individually.
# Price
#Mainfile.sort_values(['Price'],ascending=True).head(20)
#Mainfile.sort_values(['Price'],ascending=False).head(20)

# For Transmission
#Mainfile.sort_values(['Transmission'],ascending=True).head(20)
#Mainfile.sort_values(['Transmission'],ascending=False).true(20)

# For Power
#Mainfile.sort_values(['Power'],ascending=True).head(20)
#Mainfile.sort_values(['Power'],ascending=False).true(20)
```

Task 2: Data Exploration

2.1

"Task 2.1 – Model conversion needed"

```
~\anaconda3\lib\site-packages\pandas\plotting\_matplotlib\core.py in generate(self)
    284     def generate(self):
    285         self._args_adjust()
--> 286         self._compute_plot_data()
    287         self._setup_subplots()
    288         self._make_plot()

~\anaconda3\lib\site-packages\pandas\plotting\_matplotlib\core.py in _compute_plot_dat
    451         # no non-numeric frames or series allowed
    452         if is_empty:
--> 453             raise TypeError("no numeric data to plot")
    454
    455         self.data = numeric_data.apply(self._convert_to_ndarray)

TypeError: no numeric data to plot
```

"Task 2.1 – Top 10 Cars by Total"

Out[35]:

	Model	Male	Female	Unknown	Total
56	Leon	65505	30221	3778	99504
57	XE	79176	13946	3968	97090
58	Fusion	51517	41027	4395	96939
59	Accord	74157	15889	5274	95320
60	MX-5	47900	39519	5986	93405
61	6	71632	17216	4536	93384
632	XKR	7853	981	428	9262
62	C1	30691	57228	4503	92422
228	Citigo	3867	4960	404	9231
229	Liana	5205	3467	543	9215

2.2

Task 2.2 – Different Prices and Models”

Out[16]:

	Manufacturer	Price	Power
3	Renault	22.10	75.58
4	BMW	47.85	126.11
5	Volkswagen	18.19	60.96
6	Peugeot	20.03	71.33
7	Ford	39.97	130.25
...	...	...	...
4746	Lancia	29.04	105.07
4764	Morris	5.76	43.00
4779	Morris	8.12	51.50
4999	Morris	5.76	43.00
5017	Morris	8.12	51.50

255 rows × 3 columns

“Task 2.2 – Same highest values from different businesses.

Out[43]:

	Manufacturer	Model	Price	Power
4219	Porsche	Carrera	646.6	450.0
3930	Porsche	Carrera	646.6	450.0
5906	Porsche	Carrera	646.6	450.0
2648	Porsche	Carrera	646.6	450.0
2996	Porsche	Carrera	646.6	450.0
1176	Porsche	Carrera	646.6	450.0
791	Porsche	Carrera	646.6	450.0
6095	Porsche	Carrera	646.6	450.0
3637	Porsche	Carrera	646.6	450.0
399	Porsche	Carrera	646.6	450.0

"Task 2.2 – Sorted Power"

```
#Looking at Price
#PriceandPower = PriceandPower.groupby('Manufacturer')[['Manufacturer','Model','Price','Power']].head(10)
PriceandPower.sort_values(['Power'],ascending=False).head(10)
```

Out[49]:

	Manufacturer	Model	Price	Power
4219	Porsche	Carrera	646.6	450.0
3930	Porsche	Carrera	646.6	450.0
5906	Porsche	Carrera	646.6	450.0
2648	Porsche	Carrera	646.6	450.0
2996	Porsche	Carrera	646.6	450.0
1176	Porsche	Carrera	646.6	450.0
791	Porsche	Carrera	646.6	450.0
6095	Porsche	Carrera	646.6	450.0
3637	Porsche	Carrera	646.6	450.0
399	Porsche	Carrera	646.6	450.0

```
#Looking at Price
PriceandPower = PriceandPower.groupby('Manufacturer')[['Manufacturer','Model','Price','Power']].head(10)
PriceandPower.sort_values(['Power'],ascending=False).head(10)
```

Out[50]:

	Manufacturer	Model	Price	Power
399	Porsche	Carrera	646.60	450.0
1428	Bentley	Continental	285.25	428.2
1823	Bentley	Continental	285.25	428.2
2193	Bentley	Continental	285.25	428.2
230	Bentley	Continental	285.25	428.2
2572	Bentley	Continental	285.25	428.2
1038	Bentley	Continental	285.25	428.2
2938	Bentley	Continental	285.25	428.2
3287	Bentley	Continental	285.25	428.2
3638	Bentley	Continental	285.25	428.2

```
#Looking at Power
PriceandPower = PriceandPower.groupby('Manufacturer')[['Manufacturer','Model','Price','Power']].head(1)
PriceandPower.sort_values(['Power'],ascending=False).head(10)
```

Out[61]:

	Manufacturer	Model	Price	Power
230	Bentley	Continental	285.25	428.20
283	Aston-Martin	Vantage	360.76	410.00
1191	Ferrari	F430	260.29	360.00
391	Maserati	GranCabrio	245.17	338.00
121	Porsche	911	158.79	293.53
1954	TVR	Tuscan	107.70	274.00
340	Lexus	GS	78.48	224.78
57	Jaguar	XE	65.57	167.25
3948	Daimler	4	92.80	166.00
14	Audi	A4	57.93	149.71

"Task 2.2 – Sorted Price"

```
#Looking at Price
PriceandPower = PriceandPower.groupby('Manufacturer')[['Manufacturer','Model','Price','Power']].head(1)
PriceandPower.sort_values(['Price'],ascending=False).head(10)
```

Out[62]:

	Manufacturer	Model	Price	Power
283	Aston-Martin	Vantage	360.76	410.00
230	Bentley	Continental	285.25	428.20
1191	Ferrari	F430	260.29	360.00
391	Maserati	GranCabrio	245.17	338.00
121	Porsche	911	158.79	293.53
1954	TVR	Tuscan	107.70	274.00
3948	Daimler	4	92.80	166.00
340	Lexus	GS	78.48	224.78
57	Jaguar	XE	65.57	167.25
14	Audi	A4	57.93	149.71

```
#Looking at Price
PriceandPower = PriceandPower.groupby('Manufacturer')[['Manufacturer','Model','Price','Power']].head(10)
PriceandPower.sort_values(['Power'],ascending=False).head(10)
```

Out[21]:

	Manufacturer	Model	Price	Power
399	Porsche	Carrera	646.60	450.0
1428	Bentley	Continental	285.25	428.2
1823	Bentley	Continental	285.25	428.2
2193	Bentley	Continental	285.25	428.2
230	Bentley	Continental	285.25	428.2
2572	Bentley	Continental	285.25	428.2
1038	Bentley	Continental	285.25	428.2
2938	Bentley	Continental	285.25	428.2
3287	Bentley	Continental	285.25	428.2
3638	Bentley	Continental	285.25	428.2

2.3

"Task 2.3 – Redundant values - Fuel"

Out[42]:

	Manufacturer	Model	Transmission	Fuel	Male
752	Land-Rover	Range	8.28	diesel	1847
34	Land-Rover	Range	8.28	diesel	107520
846	Land-Rover	Range	8.28	diesel	83312
445	Land-Rover	Range	8.28	diesel	94411
377	Land-Rover	Range	8.28	diesel	1532
954	BMW	740i	8.00	petrol	16520
1776	Jaguar	XF	8.00	diesel	10686
1765	Volkswagen	Touareg	8.00	diesel	10197
199	Porsche	Cayenne	8.00	petrol	10034
201	BMW	640i	8.00	petrol	11265

"Task 2.3 – Males sorted"

```
#To Sort data by male numbers
MaleData = MaleData[['Manufacturer','Model','Transmission','Fuel','Male']].head(10)

#Looking at Model
MaleData.sort_values(['Male'],ascending=False).head(10)
```

Out[58]:

	Manufacturer	Model	Transmission	Fuel	Male
4	BMW	320i	6.44	petrol	408016
7	Ford	Mondeo	1.75	petrol	357452
9	Honda	Civic	4.50	petrol	242188
3	Renault	Clio	5.62	petrol	241287
5	Volkswagen	Polo	5.07	petrol	216333
11	Renault	Megane	5.41	petrol	209379
6	Peugeot	206	4.83	petrol	178698
8	Nissan	Micra	4.87	petrol	143218
12	Toyota	Yaris	3.52	petrol	137907
10	Ford	Ka+	5.00	petrol	102708

"Task 2.3 – Transmission sorted"

```
#To Sort data by transmission
MaleData = MaleData.groupby('Model')[['Manufacturer','Model','Transmission','Fuel','Male']].head(10)

#Looking at Model
MaleData.sort_values(['Transmission'],ascending=False).head(10)
```

Out[34]:

	Manufacturer	Model	Transmission	Fuel	Male
1886	Land-Rover	Range	8.28	diesel	2951
1243	Land-Rover	Range	8.28	diesel	76869
377	Land-Rover	Range	8.28	diesel	1532
846	Land-Rover	Range	8.28	diesel	83312
752	Land-Rover	Range	8.28	diesel	1847
1633	Land-Rover	Range	8.28	diesel	70504
1126	Land-Rover	Range	8.28	diesel	2123
1504	Land-Rover	Range	8.28	diesel	2540
34	Land-Rover	Range	8.28	diesel	107520
445	Land-Rover	Range	8.28	diesel	94411

"Task 2.3 – Fuel for Males"

	Manufacturer	Model	Transmission	Fuel	Male
4	BMW	320i	6.44	petrol	408016
7	Ford	Mondeo	1.75	petrol	357452
9	Honda	Civic	4.50	petrol	242188
3	Renault	Clio	5.62	petrol	241287
5	Volkswagen	Polo	5.07	petrol	216333
11	Renault	Megane	5.41	petrol	209379
6	Peugeot	206	4.83	petrol	178698
8	Nissan	Micra	4.87	petrol	143218
12	Toyota	Yaris	3.52	petrol	137907
10	Ford	Ka+	5.00	petrol	102708

"Task 2.3 – Model"

```
#Looking at Model
MaleData.sort_values(['Model'],ascending=False).head(10)
```

Out[67]:

	Manufacturer	Model	Transmission	Fuel	Male
401	Opel	Zafira	6.00	petrol	675
269	Citroen	ZX	4.92	petrol	3972
216	MG	ZT	5.00	petrol	10660
132	MG	ZR	5.00	petrol	19477
135	BMW	Z4	6.75	petrol	19343
189	BMW	Z3	4.60	petrol	9579
166	Skoda	Yeti	6.30	petrol	13365
12	Toyota	Yaris	3.52	petrol	137907
20	Citroen	Xsara	4.88	petrol	145377
796	Mazda	Xedos	4.40	petrol	844

References

All based on APA 7.

ASQ. (2022). *WHAT IS A HISTOGRAM?.* ASQ. https://asq.org/quality-resources/histogram#:~:text=When%20to%20Use%20a%20Histogram,can%20meet%20the%20customer's%20requirements

Data Science Parichay. (2022). *Pandas – Delete rows based on column values*. Data Science Parochay. https://datascienceparichay.com/article/pandas-delete-rows-based-on-column-values/

Bibliography

Adam Smith. (2022). *How to sum rows of a pandas DataFrame in Python.* Adam Smith. https://www.adamsmith.haus/python/answers/how-to-sum-rows-of-a-pandas-dataframe-in-python

GeeksforGeeks. (2022). *Bar Plot in Matplotlib.* GeeksforGeeks. https://www.geeksforgeeks.org/bar-plot-in-matplotlib/

Lynn, S. (2022). *Use Pandas Groupby to Group and Summarise DataFrames.* Shanelynn. https://www.shanelynn.ie/summarising-aggregation-and-grouping-data-in-python-pandas/

Nightingale, N. (2022). *How to remove comma from integer in python.* Grepper. https://www.codegrepper.com/code-examples/python/how+to+remove+comma+from+integer+in+python

Nic, Dr.(2022). *Types of Data: Nominal, Ordinal, Interval/Ratio - Statistics Help*. Dr Nic's Maths and Stats. https://www.youtube.com/watch?v=hZxnzfnt5v8#t=03m15s

Microsoft. (2022). *Split text into different columns with the Convert Text to Columns Wizard.* Microsoft. https://support.microsoft.com/en-us/office/split-text-into-different-columns-with-the-convert-text-to-columns-wizard-30b14928-5550-41f5-97ca-7a3e9c363ed7
pandas. (2022). *Panda.DataFrame.dropna.* Pandas. https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.dropna.html

Python Guides. (2022). *Python Pandas Drop Rows Example*. Python Guides. https://pythonguides.com/python-pandas-drop-rows-example/#:~:text=Python%20pandas%20drop%20rows%20by%20index,-In%20this%20section&text=To%20remove%20the%20rows%20by,the%20index%20number%20or%20name.

Ren,Y. (2022, April 9). *Practical Data Science: Data Curation* [Lecture Material]. Canvas@RMIT University. https://rmit.instructure.com/courses/90503/files/23226842?module_item_id=3941935

Ren,Y. (2022, April 9). *Practical Data Science: Data Summarisation: Descriptive Statistics and Visualisation*. [Lecture Material]. Canvas@RMIT University.

RMIT University. (2022). *Easy Cite referencing tool -RMIT University.* RMIT University. https://www.lib.rmit.edu.au/easy-cite/

Stackoverflow. (2013). *How to delete rows from a pandas DataFrame based on a conditional expression [duplicate],* Stackoverflow. https://stackoverflow.com/questions/13851535/how-to-delete-rows-from-a-pandas-dataframe-based-on-a-conditional-expression

Statology. (2022). *How to Change the Position of a Legend in Matplotlib*. Statology. https://www.statology.org/matplotlib-legend-position/#:~:text=To%20change%20the%20position%20of%20a%20legend%20in%20Matplotlib%2C%20you,legend()%20function.&text=The%20default%20location%20is%20%E2%80%9Cbest,avoids%20covering%20any%20data%20points.

TutorialsPoint. (2022). *Matplotlib – Bar Plot.* Tutorialspoint. https://www.tutorialspoint.com/matplotlib/matplotlib_bar_plot.htm

3schools. (2022). *Matplotlib Pie Charts.* 3schools. https://www.w3schools.com/python/matplotlib_pie_charts.asp