CarWebScraping Report

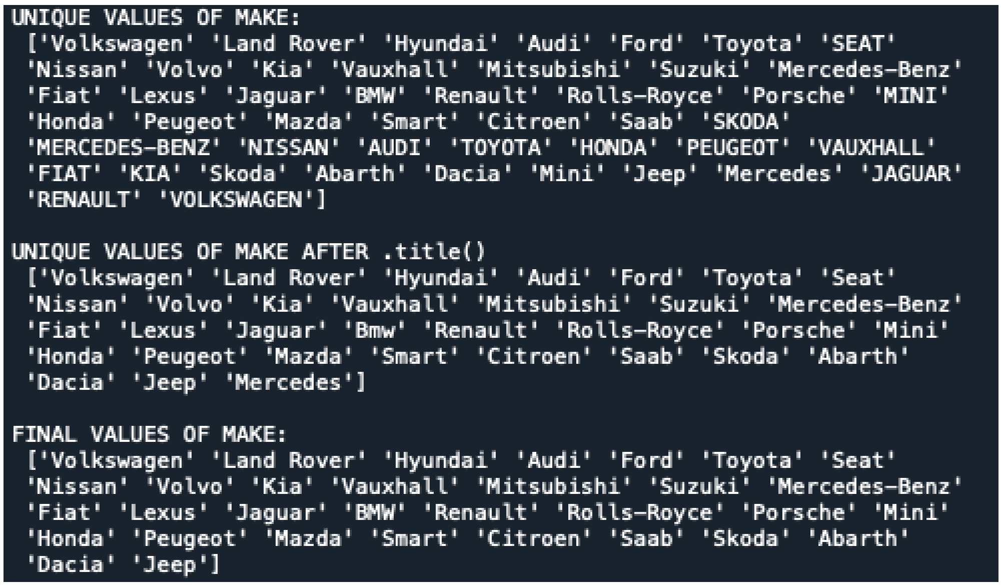
a. ***Collect 1000 items returned by search query of the website and save them into csv files.***

We cannot simply take all information from the listing of each car on the query page, as it's not all there. Need to use the individual link of each sale in order to access all the information. Each sales link is found from the query page, and then BeautifulSoup is utilised to access the data of this link. For all the data, the soup is searched using the find() function, and usually accessing the text attribute of each HTML tag gives the required data. It is noted that the distance is not possible to obtain from the individual page, and so instead is accessed on the query page and combined with the main DataFrame using .insert().
There are some pieces of data that are only present on some listings, so a function is written for these in order to use try, except, and fill the DataFrame with a null value if there is no accessible data. In the case of the delivery option, a try except is also used that returns true or false depending on whether it's able to access the delivery tag.
Another problem is that each query page only contains 20 sales, so we need to access 50 query pages to reach the total of 1000 sales. This is completed by making a list of 50 URLs and looping through them. The data generated for each is then combined using pd.concat(). Finally, saving the DataFrame to a .csv is simple with the pandas function.

b. Clean and transform the data into the correct data types.

There are two main tasks to undertake in the cleaning of this data. The first is to address the quantitative data by removing the units (putting them into the column titles) and converting them into integers. This is performed in a similar manner as to question 1 by using pd.to_numeric() in combination with replace(), on the columns price, mileage, year, doors, engine size, CO2 and review count.

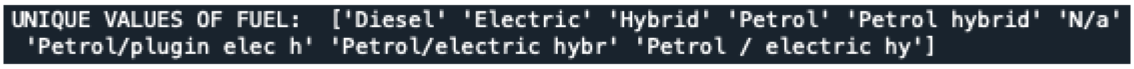
For the qualitative data, as in question 1, the unique function is used to test whether there are any non-legitimate entries or discrepancies such as typos of the same word. In the **'make'** column, we can see there are entries of 'MERCEDES-BENZ' and 'Mercedes-Benz', which we want to be represented as the same category. To address this, we use the .title() function. We can see this doesn't completely solve the problem though, we still have 'Mercedes-Benz' and 'Mercedes'. Therefore we utilise the replace() function to make the latter the same as the former. Additionally we replace 'Bmw' with 'BMW'.



```
UNIQUE VALUES OF MAKE:
 ['Volkswagen' 'Land Rover' 'Hyundai' 'Audi' 'Ford' 'Toyota' 'SEAT'
 'Nissan' 'Volvo' 'Kia' 'Vauxhall' 'Mitsubishi' 'Suzuki' 'Mercedes-Benz'
 'Fiat' 'Lexus' 'Jaguar' 'BMW' 'Renault' 'Rolls-Royce' 'Porsche' 'MINI'
 'Honda' 'Peugeot' 'Mazda' 'Smart' 'Citroen' 'Saab' 'SKODA'
 'MERCEDES-BENZ' 'NISSAN' 'AUDI' 'TOYOTA' 'HONDA' 'PEUGEOT' 'VAUXHALL'
 'FIAT' 'KIA' 'Skoda' 'Abarth' 'Dacia' 'Mini' 'Jeep' 'Mercedes' 'JAGUAR'
 'RENAULT' 'VOLKSWAGEN']

UNIQUE VALUES OF MAKE AFTER .title()
 ['Volkswagen' 'Land Rover' 'Hyundai' 'Audi' 'Ford' 'Toyota' 'Seat'
 'Nissan' 'Volvo' 'Kia' 'Vauxhall' 'Mitsubishi' 'Suzuki' 'Mercedes-Benz'
 'Fiat' 'Lexus' 'Jaguar' 'Bmw' 'Renault' 'Rolls-Royce' 'Porsche' 'Mini'
 'Honda' 'Peugeot' 'Mazda' 'Smart' 'Citroen' 'Saab' 'Skoda' 'Abarth'
 'Dacia' 'Jeep' 'Mercedes']

FINAL VALUES OF MAKE:
 ['Volkswagen' 'Land Rover' 'Hyundai' 'Audi' 'Ford' 'Toyota' 'Seat'
 'Nissan' 'Volvo' 'Kia' 'Vauxhall' 'Mitsubishi' 'Suzuki' 'Mercedes-Benz'
 'Fiat' 'Lexus' 'Jaguar' 'BMW' 'Renault' 'Rolls-Royce' 'Porsche' 'Mini'
 'Honda' 'Peugeot' 'Mazda' 'Smart' 'Citroen' 'Saab' 'Skoda' 'Abarth'
 'Dacia' 'Jeep']
```

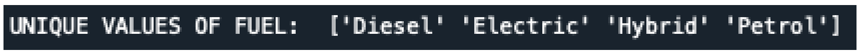
For the **model** column, groupby is used so that it can be checked whether there are any double occurrences of the same model for each make. This proves to be the case, with one example being the Mercedes ‘A-Class’ and ‘A Class’ (these were found by using the variable explorer in Spyder rather than using a print statement). Instances like this are few and replaced the same way as for the make column..
For the **variant** column, it is difficult to assess if any instances should be classed as the same without a deep knowledge of the variant structure of each model etc. The only obvious problem with this column is that there are some values where there is whitespace before or after the name, therefore in addition to the title() function, strip() is also applied.

The unique values of the **fuel** column are printed and it is found that there are various variations of hybrid cars.



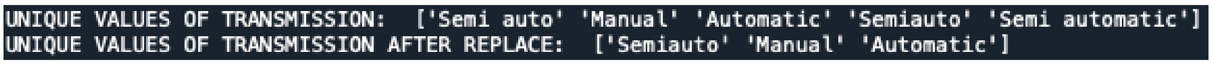
```
UNIQUE VALUES OF FUEL:  ['Diesel' 'Electric' 'Hybrid' 'Petrol' 'Petrol hybrid' 'N/a'
 'Petrol/plugin elec h' 'Petrol/electric hybr' 'Petrol / electric hy']
```

The values of ‘Petrol hybrid’, 'Petrol/plugin elec h', 'Petrol/electric hybr' and 'Petrol / electric hy' are all changed to the ‘Hybrid’ category using .replace(). Now we only have 4 standard fuel types.



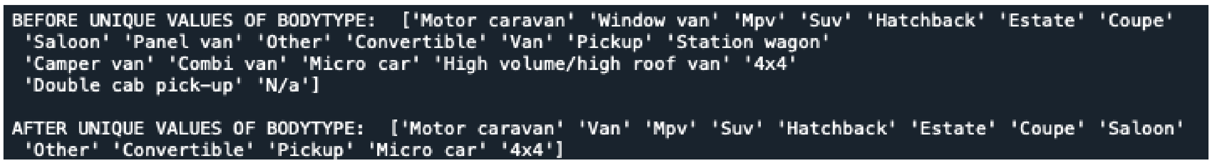
```
UNIQUE VALUES OF FUEL:  ['Diesel' 'Electric' 'Hybrid' 'Petrol']
```

We can do the same for the **transmission** column, and find that we need to standardize the semiautomatic values, which is completed using the replace() function.



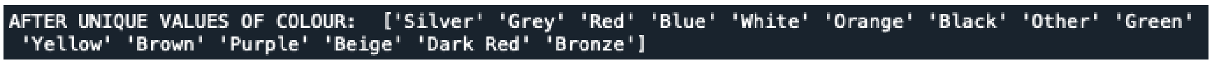
```
UNIQUE VALUES OF TRANSMISSION:  ['Semi auto' 'Manual' 'Automatic' 'Semiauto' 'Semi automatic']
UNIQUE VALUES OF TRANSMISSION AFTER REPLACE:  ['Semiauto' 'Manual' 'Automatic']
```

For **body type**, looking at all the categories, there are many that could be combined. We make the decision that ‘Window van’, ‘Panel van’, ‘Combi van’ and ‘High volume/high roof van’ can all be combined under ‘Van’. ‘Station wagon’ is another name for an estate so these categories will be combined, ‘Double cab pickup’ will be assigned to ‘Pickup’, and finally ‘Camper van’ will be changed to ‘Motor caravan’, all through the replace function.



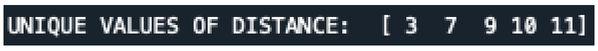
```
BEFORE UNIQUE VALUES OF BODYTYPE:  ['Motor caravan' 'Window van' 'Mpv' 'Suv' 'Hatchback' 'Estate' 'Coupe'
 'Saloon' 'Panel van' 'Other' 'Convertible' 'Van' 'Pickup' 'Station wagon'
 'Camper van' 'Combi van' 'Micro car' 'High volume/high roof van' '4x4'
 'Double cab pick-up' 'N/a']

AFTER UNIQUE VALUES OF BODYTYPE:  ['Motor caravan' 'Van' 'Mpv' 'Suv' 'Hatchback' 'Estate' 'Coupe' 'Saloon'
 'Other' 'Convertible' 'Pickup' 'Micro car' '4x4']
```

With the **colour** column, we have a multitude of different colour names given by car companies that we want to condense down to basic colour names, ‘red’, ‘green’ etc. A replace() function is used to do this, and a lot of the colours have to be googled in order to judge which ‘normal’ colour they are. Hence a large dictionary is used with all the changes. ‘N/a’ and ‘Standard’ colours are encompassed in the ‘Other’ category. We finish with the following colour categories:



```
AFTER UNIQUE VALUES OF COLOUR:  ['Silver' 'Grey' 'Red' 'Blue' 'White' 'Orange' 'Black' 'Other' 'Green'
 'Yellow' 'Brown' 'Purple' 'Beige' 'Dark Red' 'Bronze']
```

For the **distance**, the string in the form of ‘7 miles’ is taken, the number isolated by splitting the string, and converted to an int using .to_numeric(). We can see the unique values are all reasonable.

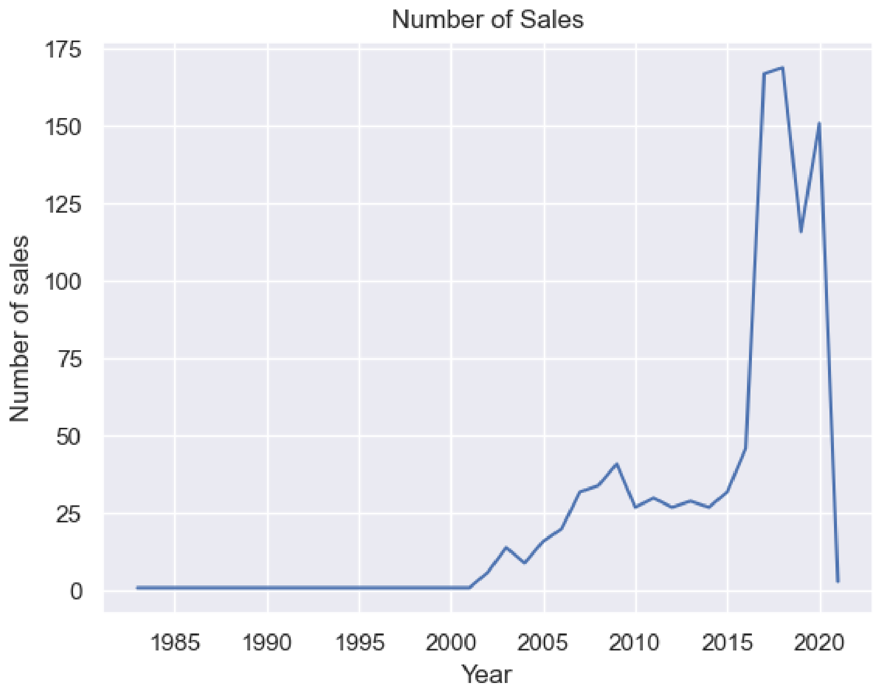


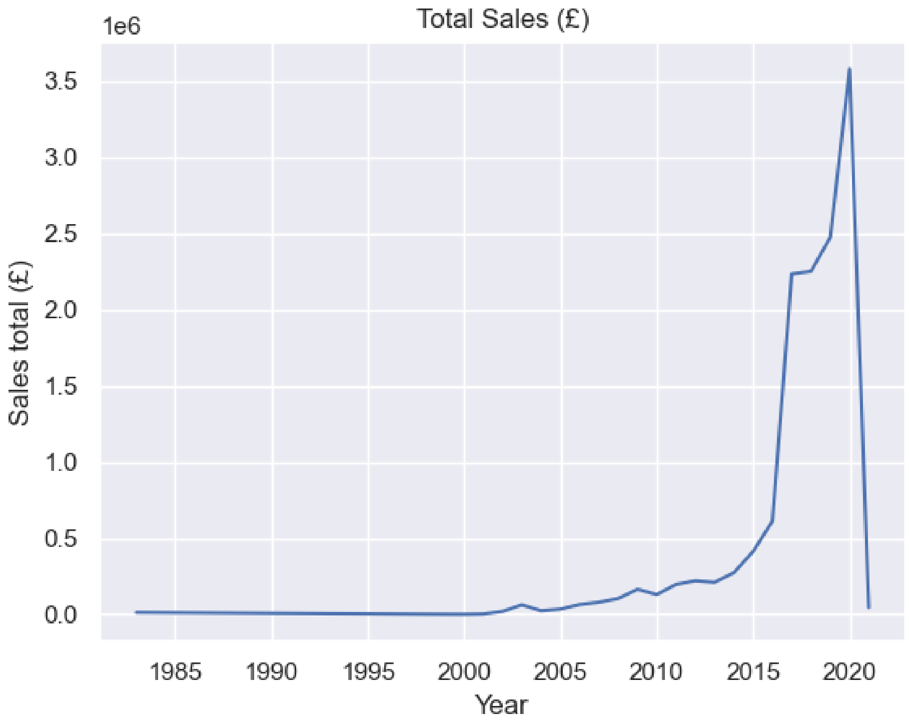
```
UNIQUE VALUES OF DISTANCE:  [ 3  7  9 10 11]
```

We know that **delivery** only has values of true or false because of the way the data is collected.

c. ***Calculate the total car sales based on the “registration year” feature***

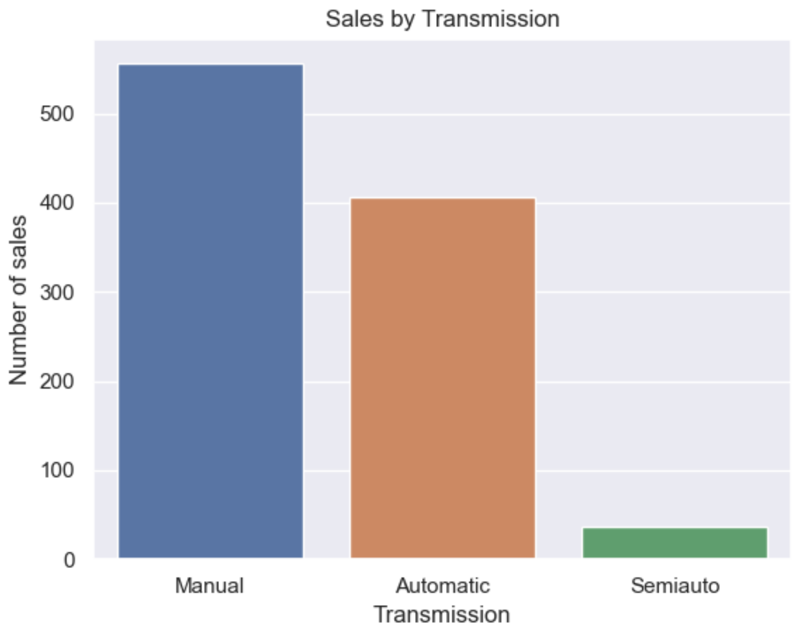
To complete this calculation, we only need to use the groupby function on the dataframe. By grouping on the year column, we can then use either size() to get the total number of sales, or can use sum() to get the total income from sales in £. Graphs for both are displayed below:


Number of Sales
Number of sales
175
150
125
100
75
50
25
0
1985
1990
1995
2000
2005
2010
2015
2020
Year

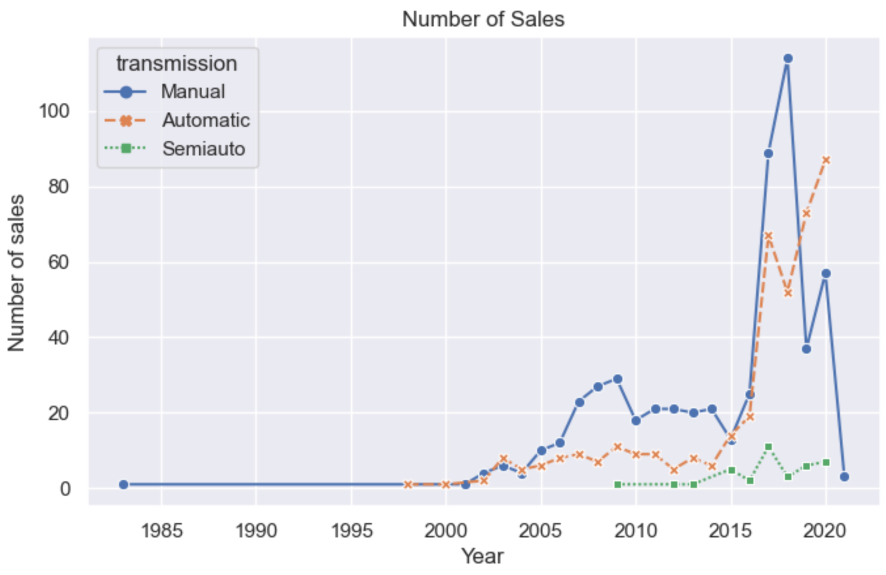

Total Sales (£)
1e6
Sales total (£)
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
1985
1990
1995
2000
2005
2010
2015
2020
Year

d. *Compare car sales on their transmission features*

This is completed very similarly to part c), apart from now we use groupby on year and transmission columns. The results can be seen below.

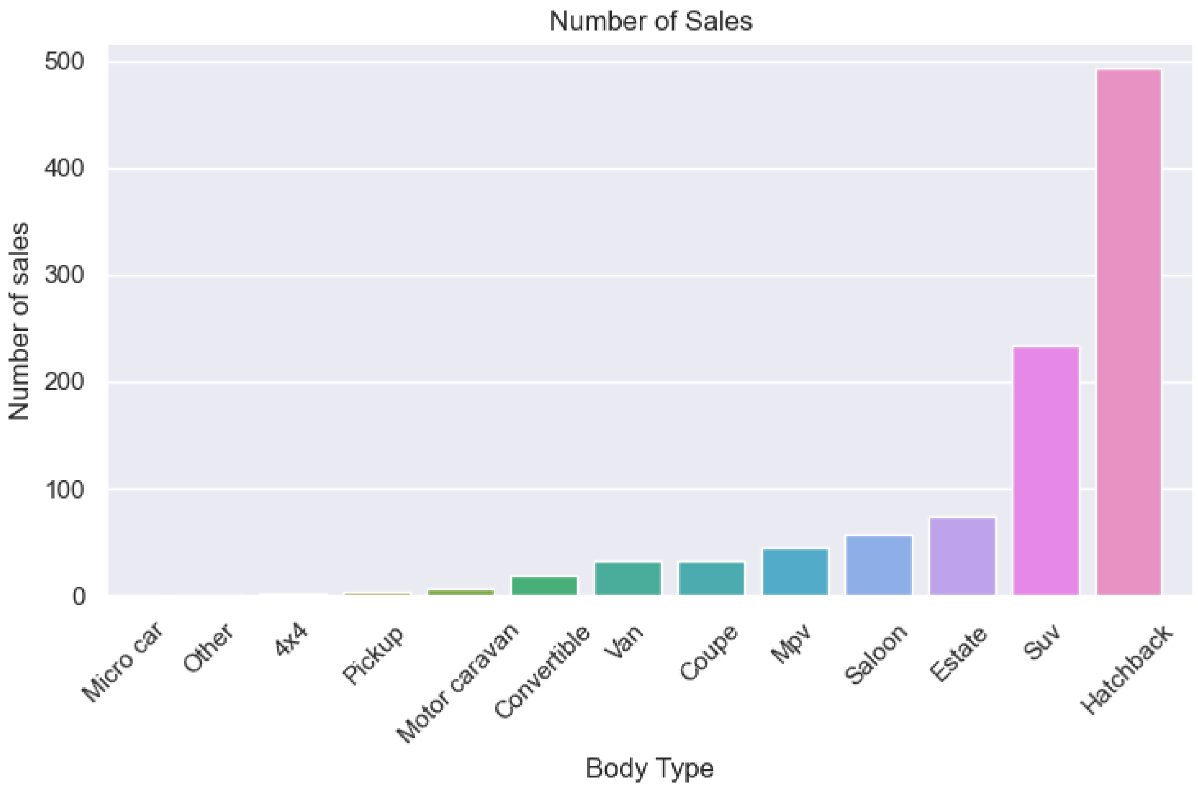


It is seen here that the most popular type of transmission is a manual, followed by automatic then semiauto. However, we find some interesting insight by plotting these three per year. It is seen that no automatics are for sale registered before the 90s, and no semiautos before 2009. In fact, for the cars registered in 2019 and 2020 there are more automatics for sale than manuals.



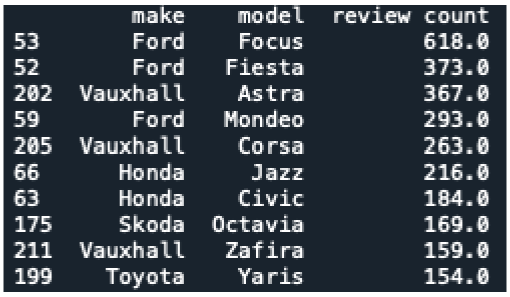
e. What are the most popular car sales based on the "Body Type"?

Here we groupby the body type column and once more use size() to get the number of cars for sale with each type. The results are seen below, with hatchback by far the most popular.



f. List top 10 cars having highest numbers of reviews.

groupby is called on make and model, then the resulting table sorted using sort_values(). Then it's as simple as selecting the correct columns and using head() to select the top 10.



```
         make    model  review count
53       Ford    Focus         618.0
52       Ford   Fiesta         373.0
202  Vauxhall    Astra         367.0
59       Ford   Mondeo         293.0
205  Vauxhall    Corsa         263.0
66      Honda     Jazz         216.0
63      Honda    Civic         184.0
175     Skoda  Octavia         169.0
211  Vauxhall   Zafira         159.0
199    Toyota    Yaris         154.0
```