Installing and importing packages

```
In [1]:  pip install requests beautifulsoup4 matplotlib pandas numpy sns
```

```
Requirement already satisfied: requests in c:\users\wikto\anaconda3\lib\site-packages (2.25.1)
Requirement already satisfied: beautifulsoup4 in c:\users\wikto\anaconda3\lib\site-packages (4.9.3)
Requirement already satisfied: matplotlib in c:\users\wikto\anaconda3\lib\site-packages (3.3.4)
Requirement already satisfied: pandas in c:\users\wikto\anaconda3\lib\site-packages (1.2.4)
Requirement already satisfied: numpy in c:\users\wikto\anaconda3\lib\site-packages (1.20.1)
Requirement already satisfied: sns in c:\users\wikto\anaconda3\lib\site-packages (0.1)
Requirement already satisfied: soupsieve>1.2 in c:\users\wikto\anaconda3\lib\site-packages (from beautifulsoup4) (2.2.1)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\wikto\anaconda3\lib\site-packages (from matplotlib) (2.8.1)
Requirement already satisfied: cycler>=0.10 in c:\users\wikto\anaconda3\lib\site-packages (from matplotlib) (0.10.0)
Requirement already satisfied: pillow>=6.2.0 in c:\users\wikto\anaconda3\lib\site-packages (from matplotlib) (8.2.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in c:\users\wikto\anaconda3\lib\site-packages (from matplotlib) (2.4.7)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\wikto\anaconda3\lib\site-packages (from matplotlib) (1.3.1)
Requirement already satisfied: six in c:\users\wikto\anaconda3\lib\site-packages (from cycler>=0.10->matplotlib) (1.15.0)
Requirement already satisfied: pytz>=2017.3 in c:\users\wikto\anaconda3\lib\site-packages (from pandas) (2021.1)
Note: you may need to restart the kernel to use updated packages.
Requirement already satisfied: certifi>=2017.4.17 in c:\users\wikto\anaconda3\lib\site-packages (from requests) (2020.12.5)
Requirement already satisfied: chardet<5,>=3.0.2 in c:\users\wikto\anaconda3\lib\site-packages (from requests) (4.0.0)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\wikto\anaconda3\lib\site-packages (from requests) (1.26.4)
Requirement already satisfied: idna<3,>=2.5 in c:\users\wikto\anaconda3\lib\site-packages (from requests) (2.10)
```

```
In [2]:  import requests
         from bs4 import BeautifulSoup
         import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt
         import seaborn as sns
```

Creating a scraping function

```
In [4]:  #creating a function which looks for tables on the website and scrapes them
         def scraping_function(table_url): #defining the function
             results_response = requests.get(table_url) #sending the get request to the specific url under which table is located
             results_soup = BeautifulSoup(results_response.text, 'html.parser') #creating an object in the BeautifulSoup library which contains the scraped html content

             race_tables = [] #creating an empty dataframe

             for table in results_soup.find_all('table'): #Looking for tables on the website
                 table_name = table.find_previous('h2').text.strip() #stripping table text which is located under the subheading
                 race_scraped = pd.read_html(str(table))[0] #saving the scraped table as a variable
                 race_tables.append(race_scraped) #appending the scraped table to the empty dataframe created previously
             final_df = pd.concat(race_tables, ignore_index = True) #creating final dataframe with the scraped table

             return final_df
```

Scraping the necessary datasets

Scraping races data

```
In [20]:  races = scraping_function('https://www.formula1.com/en/results.html/2023/races.html')#applying the scraping function created above to scrape the 'races' dataframe
          races
```

Out[20]:

	Unnamed: 0	Grand Prix	Date	Winner	Car	Laps	Time	Unnamed: 7
0	NaN	Bahrain	05 Mar 2023	Max Verstappen VER	Red Bull Racing Honda RBPT	57	1:33:56.736	NaN
1	NaN	Saudi Arabia	19 Mar 2023	Sergio Perez PER	Red Bull Racing Honda RBPT	50	1:21:14.894	NaN
2	NaN	Australia	02 Apr 2023	Max Verstappen VER	Red Bull Racing Honda RBPT	58	2:32:38.371	NaN
3	NaN	Azerbaijan	30 Apr 2023	Sergio Perez PER	Red Bull Racing Honda RBPT	51	1:32:42.436	NaN
4	NaN	Miami	07 May 2023	Max Verstappen VER	Red Bull Racing Honda RBPT	57	1:27:38.241	NaN
5	NaN	Monaco	28 May 2023	Max Verstappen VER	Red Bull Racing Honda RBPT	78	1:48:51.980	NaN
6	NaN	Spain	04 Jun 2023	Max Verstappen VER	Red Bull Racing Honda RBPT	66	1:27:57.940	NaN
7	NaN	Canada	18 Jun 2023	Max Verstappen VER	Red Bull Racing Honda RBPT	70	1:33:58.348	NaN
8	NaN	Austria	02 Jul 2023	Max Verstappen VER	Red Bull Racing Honda RBPT	71	1:25:33.607	NaN
9	NaN	Great Britain	09 Jul 2023	Max Verstappen VER	Red Bull Racing Honda RBPT	52	1:25:16.938	NaN
10	NaN	Hungary	23 Jul 2023	Max Verstappen VER	Red Bull Racing Honda RBPT	70	1:38:08.634	NaN
11	NaN	Belgium	30 Jul 2023	Max Verstappen VER	Red Bull Racing Honda RBPT	44	1:22:30.450	NaN
12	NaN	Netherlands	27 Aug 2023	Max Verstappen VER	Red Bull Racing Honda RBPT	72	2:24:04.411	NaN
13	NaN	Italy	03 Sep 2023	Max Verstappen VER	Red Bull Racing Honda RBPT	51	1:13:41.143	NaN
14	NaN	Singapore	17 Sep 2023	Carlos Sainz SAI	Ferrari	62	1:46:37.418	NaN
15	NaN	Japan	24 Sep 2023	Max Verstappen VER	Red Bull Racing Honda RBPT	53	1:30:58.421	NaN
16	NaN	Qatar	08 Oct 2023	Max Verstappen VER	Red Bull Racing Honda RBPT	57	1:27:39.168	NaN
17	NaN	United States	22 Oct 2023	Max Verstappen VER	Red Bull Racing Honda RBPT	56	1:35:21.362	NaN
18	NaN	Mexico	29 Oct 2023	Max Verstappen VER	Red Bull Racing Honda RBPT	71	2:02:30.814	NaN
19	NaN	Brazil	05 Nov 2023	Max Verstappen VER	Red Bull Racing Honda RBPT	71	1:56:48.894	NaN
20	NaN	Las Vegas	18 Nov 2023	Max Verstappen VER	Red Bull Racing Honda RBPT	50	1:29:08.289	NaN
21	NaN	Abu Dhabi	26 Nov 2023	Max Verstappen VER	Red Bull Racing Honda RBPT	58	1:27:02.624	NaN

Scraping race results for individual races

```
In [6]: Bahrain = scraping_function('https://www.formula1.com/en/results.html/2023/races/1141/bahrain/race-result.html')
        Saudi_arabia = scraping_function('https://www.formula1.com/en/results.html/2023/races/1142/saudi-arabia/race-result.html')
        Australia = scraping_function('https://www.formula1.com/en/results.html/2023/races/1143/australia/race-result.html')
        Azerbaijan = scraping_function('https://www.formula1.com/en/results.html/2023/races/1207/azerbaijan/race-result.html')
        Miami = scraping_function('https://www.formula1.com/en/results.html/2023/races/1208/miami/race-result.html')
        #note: the Emili_romagna race was cancelled hence there is no data to scrape hence it is not included here
        Monaco = scraping_function('https://www.formula1.com/en/results.html/2023/races/1210/monaco/race-result.html')
        Spain = scraping_function('https://www.formula1.com/en/results.html/2023/races/1211/spain/race-result.html')
        Canada = scraping_function('https://www.formula1.com/en/results.html/2023/races/1212/canada/race-result.html')
        Austria = scraping_function('https://www.formula1.com/en/results.html/2023/races/1213/austria/race-result.html')
        Great_britain = scraping_function('https://www.formula1.com/en/results.html/2023/races/1214/great-britain/race-result.html')
        Hungary = scraping_function('https://www.formula1.com/en/results.html/2023/races/1215/hungary/race-result.html')
        Belgium = scraping_function('https://www.formula1.com/en/results.html/2023/races/1216/belgium/race-result.html')
        Netherlands = scraping_function('https://www.formula1.com/en/results.html/2023/races/1217/netherlands/race-result.html')
        Italy = scraping_function('https://www.formula1.com/en/results.html/2023/races/1218/italy/race-result.html')
        Singapore = scraping_function('https://www.formula1.com/en/results.html/2023/races/1219/singapore/race-result.html')
        Japan = scraping_function('https://www.formula1.com/en/results.html/2023/races/1220/japan/race-result.html')
        Qatar = scraping_function('https://www.formula1.com/en/results.html/2023/races/1221/qatar/race-result.html')
        US = scraping_function('https://www.formula1.com/en/results.html/2023/races/1222/united-states/race-result.html')
        Mexico = scraping_function('https://www.formula1.com/en/results.html/2023/races/1223/mexico/race-result.html')
        Brazil = scraping_function('https://www.formula1.com/en/results.html/2023/races/1224/brazil/race-result.html')
        Las_vegas = scraping_function('https://www.formula1.com/en/results.html/2023/races/1225/las-vegas/race-result.html')
        Abu_dhabi = scraping_function('https://www.formula1.com/en/results.html/2023/races/1226/abu-dhabi/race-result.html')
```

Scraping qualifiers data

```
In [7]: Bahrain_qual = scraping_function('https://www.formula1.com/en/results.html/2023/races/1141/bahrain/qualifying.html')
        Saudi_arabia_qual = scraping_function('https://www.formula1.com/en/results.html/2023/races/1142/saudi-arabia/qualifying.html')
        Australia_qual = scraping_function('https://www.formula1.com/en/results.html/2023/races/1143/australia/qualifying.html')
        Azerbaijan_qual = scraping_function('https://www.formula1.com/en/results.html/2023/races/1207/azerbaijan/qualifying.html')
        Miami_qual = scraping_function('https://www.formula1.com/en/results.html/2023/races/1208/miami/qualifying.html')
        Monaco_qual = scraping_function('https://www.formula1.com/en/results.html/2023/races/1210/monaco/qualifying.html')
        Spain_qual = scraping_function('https://www.formula1.com/en/results.html/2023/races/1211/spain/qualifying.html')
        Canada_qual = scraping_function('https://www.formula1.com/en/results.html/2023/races/1212/canada/qualifying.html')
        Austria_qual = scraping_function('https://www.formula1.com/en/results.html/2023/races/1213/austria/qualifying.html')
        Great_britain_qual = scraping_function('https://www.formula1.com/en/results.html/2023/races/1214/great-britain/qualifying.html')
        Hungary_qual = scraping_function('https://www.formula1.com/en/results.html/2023/races/1215/hungary/qualifying.html')
        Belgium_qual = scraping_function('https://www.formula1.com/en/results.html/2023/races/1216/belgium/qualifying.html')
        Netherlands_qual = scraping_function('https://www.formula1.com/en/results.html/2023/races/1217/netherlands/qualifying.html')
        Italy_qual = scraping_function('https://www.formula1.com/en/results.html/2023/races/1218/italy/qualifying.html')
        Singapore_qual = scraping_function('https://www.formula1.com/en/results.html/2023/races/1219/singapore/qualifying.html')
        Japan_qual = scraping_function('https://www.formula1.com/en/results.html/2023/races/1220/japan/qualifying.html')
        Qatar_qual = scraping_function('https://www.formula1.com/en/results.html/2023/races/1221/qatar/qualifying.html')
        US_qual = scraping_function('https://www.formula1.com/en/results.html/2023/races/1222/united-states/qualifying.html')
        Mexico_qual = scraping_function('https://www.formula1.com/en/results.html/2023/races/1223/mexico/qualifying.html')
        Brazil_qual = scraping_function('https://www.formula1.com/en/results.html/2023/races/1224/brazil/qualifying.html')
        Las_vegas_qual = scraping_function('https://www.formula1.com/en/results.html/2023/races/1225/las-vegas/qualifying.html')
        Abu_dhabi_qual = scraping_function('https://www.formula1.com/en/results.html/2023/races/1226/abu-dhabi/qualifying.html')
```

```
In [8]: #rename 'Pos' column for qualifying dataset to PosQ to distniguish between qualifying and race positions in the final dataset I am building
        qual_data = [Bahrain_qual, Saudi_arabia_qual, Australia_qual, Azerbaijan_qual, Miami_qual,
                     Monaco_qual, Spain_qual, Canada_qual, Austria_qual, Great_britain_qual, Hungary_qual,
                     Belgium_qual, Netherlands_qual, Italy_qual, Singapore_qual, Japan_qual, Qatar_qual,
                     US_qual, Mexico_qual, Brazil_qual, Las_vegas_qual, Abu_dhabi_qual]

        for df in qual_data:
            df.rename(columns={'Pos': 'PosQ'}, inplace=True) #rename Pos to PosQ to idenitfy qualifying positions
```

```
In [9]: #Merging final results and qualifiers datasets for each race
        Bahrain = pd.merge(Bahrain, Bahrain_qual[['Driver', 'PosQ', 'Q1', 'Q2', 'Q3']], on = 'Driver',
                           how = 'inner')
        Saudi_arabia = pd.merge(Saudi_arabia, Saudi_arabia_qual[['Driver', 'PosQ', 'Q1', 'Q2', 'Q3']], on = 'Driver',
                           how = 'inner')
        Australia = pd.merge(Australia, Australia_qual[['Driver', 'PosQ', 'Q1', 'Q2', 'Q3']], on = 'Driver',
                           how = 'inner')
        Azerbaijan = pd.merge(Azerbaijan, Azerbaijan_qual[['Driver', 'PosQ', 'Q1', 'Q2', 'Q3']], on = 'Driver',
                           how = 'inner')
        Miami = pd.merge(Miami, Miami_qual[['Driver', 'PosQ', 'Q1', 'Q2', 'Q3']], on = 'Driver',
                           how = 'inner')
        Monaco = pd.merge(Monaco, Monaco_qual[['Driver', 'PosQ', 'Q1', 'Q2', 'Q3']], on = 'Driver',
                           how = 'inner')
        Spain = pd.merge(Spain, Spain_qual[['Driver', 'PosQ', 'Q1', 'Q2', 'Q3']], on = 'Driver',
                           how = 'inner')
        Canada = pd.merge(Canada, Canada_qual[['Driver', 'PosQ', 'Q1', 'Q2', 'Q3']], on = 'Driver',
                           how = 'inner')
        Austria = pd.merge(Austria, Austria_qual[['Driver', 'PosQ', 'Q1', 'Q2', 'Q3']], on = 'Driver',
                           how = 'inner')
        Great_britain = pd.merge(Great_britain, Great_britain_qual[['Driver', 'PosQ', 'Q1', 'Q2', 'Q3']], on = 'Driver',
                           how = 'inner')
        Hungary_qual = pd.merge(Hungary, Hungary_qual[['Driver', 'PosQ', 'Q1', 'Q2', 'Q3']], on = 'Driver',
                           how = 'inner')
        Belgium = pd.merge(Belgium, Belgium_qual[['Driver', 'PosQ', 'Q1', 'Q2', 'Q3']], on = 'Driver',
                           how = 'inner')
        Netherlands = pd.merge(Netherlands, Netherlands_qual[['Driver', 'PosQ', 'Q1', 'Q2', 'Q3']], on = 'Driver',
                           how = 'inner')
        Italy = pd.merge(Italy, Italy_qual[['Driver', 'PosQ', 'Q1', 'Q2', 'Q3']], on = 'Driver',
                           how = 'inner')
        Singapore = pd.merge(Singapore, Singapore_qual[['Driver', 'PosQ', 'Q1', 'Q2', 'Q3']], on = 'Driver',
                           how = 'inner')
        Japan = pd.merge(Japan, Japan_qual[['Driver', 'PosQ', 'Q1', 'Q2', 'Q3']], on = 'Driver',
                           how = 'inner')
        Qatar_qual = pd.merge(Qatar, Qatar_qual[['Driver', 'PosQ', 'Q1', 'Q2', 'Q3']], on = 'Driver',
                           how = 'inner')
        US = pd.merge(US, US_qual[['Driver', 'PosQ', 'Q1', 'Q2', 'Q3']], on = 'Driver',
                           how = 'inner')
        Mexico = pd.merge(Mexico, Mexico_qual[['Driver', 'PosQ', 'Q1', 'Q2', 'Q3']], on = 'Driver',
                           how = 'inner')
        Brazil = pd.merge(Brazil, Brazil_qual[['Driver', 'PosQ', 'Q1', 'Q2', 'Q3']], on = 'Driver',
                           how = 'inner')
        Las_vegas = pd.merge(Las_vegas, Las_vegas_qual[['Driver', 'PosQ', 'Q1', 'Q2', 'Q3']], on = 'Driver',
                           how = 'inner')
        Abu_dhabi = pd.merge(Abu_dhabi, Abu_dhabi_qual[['Driver', 'PosQ', 'Q1', 'Q2', 'Q3']], on = 'Driver',
                           how = 'inner')
```

Scraping pit stop data for each race

```
In [10]: Bahrain_pit = scraping_function('https://www.formula1.com/en/results.html/2023/races/1141/bahrain/pit-stop-summary.html')
         Saudi_arabia_pit = scraping_function('https://www.formula1.com/en/results.html/2023/races/1142/saudi-arabia/pit-stop-summary.html')
         Australia_pit = scraping_function('https://www.formula1.com/en/results.html/2023/races/1143/australia/pit-stop-summary.html')
         Azerbaijan_pit = scraping_function('https://www.formula1.com/en/results.html/2023/races/1207/azerbaijan/pit-stop-summary.html')
         Miami_pit = scraping_function('https://www.formula1.com/en/results.html/2023/races/1208/miami/pit-stop-summary.html')
         Monaco_pit = scraping_function('https://www.formula1.com/en/results.html/2023/races/1210/monaco/pit-stop-summary.html')
         Spain_pit = scraping_function('https://www.formula1.com/en/results.html/2023/races/1211/spain/pit-stop-summary.html')
         Canada_pit = scraping_function('https://www.formula1.com/en/results.html/2023/races/1212/canada/pit-stop-summary.html')
         Austria_pit = scraping_function('https://www.formula1.com/en/results.html/2023/races/1213/austria/pit-stop-summary.html')
         Great_britain_pit = scraping_function('https://www.formula1.com/en/results.html/2023/races/1214/great-britain/pit-stop-summary.html')
         Hungary_pit = scraping_function('https://www.formula1.com/en/results.html/2023/races/1215/hungary/pit-stop-summary.html')
         Belgium_pit = scraping_function('https://www.formula1.com/en/results.html/2023/races/1216/belgium/pit-stop-summary.html')
         Netherlands_pit = scraping_function('https://www.formula1.com/en/results.html/2023/races/1217/netherlands/pit-stop-summary.html')
         Italy_pit = scraping_function('https://www.formula1.com/en/results.html/2023/races/1218/italy/pit-stop-summary.html')
         Singapore_pit = scraping_function('https://www.formula1.com/en/results.html/2023/races/1219/singapore/pit-stop-summary.html')
         Japan_pit = scraping_function('https://www.formula1.com/en/results.html/2023/races/1220/japan/pit-stop-summary.html')
         Qatar_pit = scraping_function('https://www.formula1.com/en/results.html/2023/races/1221/qatar/pit-stop-summary.html')
         US_pit = scraping_function('https://www.formula1.com/en/results.html/2023/races/1222/united-states/pit-stop-summary.html')
         Mexico_pit = scraping_function('https://www.formula1.com/en/results.html/2023/races/1223/mexico/pit-stop-summary.html')
         Brazil_pit = scraping_function('https://www.formula1.com/en/results.html/2023/races/1224/brazil/pit-stop-summary.html')
         Las_vegas_pit = scraping_function('https://www.formula1.com/en/results.html/2023/races/1225/las-vegas/pit-stop-summary.html')
         Abu_dhabi_pit = scraping_function('https://www.formula1.com/en/results.html/2023/races/1226/abu-dhabi/pit-stop-summary.html')
```

Data Cleaning

Firstly I am adding identifier columns to all the dataframes so that I can join them up together

```
In [21]: races['RaceID'] = races.reset_index().index #adding identifier column to the races dataset
         races = races[['RaceID','Date', 'Grand Prix']] #reducing dataframe to  the columns needed for the analysis
         races.loc[:,'Date'] = pd.to_datetime(races['Date'], format='%d %b %Y') #converting date column to date format
         races
```

```
C:\Users\wikto\anaconda3\lib\site-packages\pandas\core\indexing.py:1676: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  self._setitem_single_column(ilocs[0], value, pi)
```

Out[21]:

	RaceID	Date	Grand Prix
0	0	2023-03-05	Bahrain
1	1	2023-03-19	Saudi Arabia
2	2	2023-04-02	Australia
3	3	2023-04-30	Azerbaijan
4	4	2023-05-07	Miami
5	5	2023-05-28	Monaco
6	6	2023-06-04	Spain
7	7	2023-06-18	Canada
8	8	2023-07-02	Austria
9	9	2023-07-09	Great Britain
10	10	2023-07-23	Hungary
11	11	2023-07-30	Belgium
12	12	2023-08-27	Netherlands
13	13	2023-09-03	Italy
14	14	2023-09-17	Singapore
15	15	2023-09-24	Japan
16	16	2023-10-08	Qatar
17	17	2023-10-22	United States
18	18	2023-10-29	Mexico
19	19	2023-11-05	Brazil
20	20	2023-11-18	Las Vegas
21	21	2023-11-26	Abu Dhabi

```
In [22]: #Adding identifier columns to individual races datasets
         Bahrain['RaceID'] = '0'
         Saudi_arabia['RaceID'] = '1'
         Australia['RaceID'] = '2'
         Azerbaijan['RaceID'] = '3'
         Miami['RaceID'] = '4'
         Monaco['RaceID'] = '5'
         Spain['RaceID'] = '6'
         Canada['RaceID'] = '7'
         Austria['RaceID'] = '8'
         Great_britain['RaceID'] = '9'
         Hungary['RaceID'] = '10'
         Belgium['RaceID'] = '11'
         Netherlands['RaceID'] = '12'
         Italy['RaceID'] = '13'
         Singapore['RaceID'] = '14'
         Japan['RaceID'] = '15'
         Qatar['RaceID'] = '16'
         US['RaceID'] = '17'
         Mexico['RaceID'] = '18'
         Brazil['RaceID'] = '19'
         Las_vegas['RaceID'] = '20'
         Abu_dhabi['RaceID'] = '21'
```

Merging race dataframes to create the final dataframe

```
In [23]:   races_dataframes = [Bahrain, Saudi_arabia, Australia, Azerbaijan, Miami, Monaco, Spain, Canada, Austria, Great_britain,
                               Hungary, Belgium, Netherlands, Italy, Singapore, Japan, Qatar, US, Mexico, Brazil, Las_vegas, Abu_dhabi]
           races_dataframes = pd.concat(races_dataframes, ignore_index=True) #joining up the races vertically
           races_dataframes['RaceID'] = races_dataframes['RaceID'].astype(int) #converting the RaceID column from 'object' type to integer to allow me to merge with the Races datafra
           F1_full = pd.merge(races_dataframes, races, on ='RaceID', how='left') #merging Races and Results dataframes
           F1_full = F1_full[['RaceID', 'Date', 'Grand Prix', 'Driver', 'Car', 'Pos','PTS', 'Laps', 'PosQ', 'Q1', 'Q2', 'Q3']] #reducing dataframe to the columns we need for analysis
```

```
In [24]:   #checking that all races have been merged together
           F1_full['Grand Prix'].unique()
```

```
Out[24]:   array(['Bahrain', 'Saudi Arabia', 'Australia', 'Azerbaijan', 'Miami',
                  'Monaco', 'Spain', 'Canada', 'Austria', 'Great Britain', 'Hungary',
                  'Belgium', 'Netherlands', 'Italy', 'Singapore', 'Japan', 'Qatar',
                  'United States', 'Mexico', 'Brazil', 'Las Vegas', 'Abu Dhabi'],
                 dtype=object)
```

```
In [133]:  #Checking for duplicate observations
           print(F1_full[F1_full.duplicated()])
```
```
           Empty DataFrame
           Columns: [RaceID, Date, Grand Prix, Driver, Car, Pos, PTS, Laps, PosQ, Q1, Q2, Q3]
           Index: []
```

```
In [25]:   #Checking for missing values
           F1_full.isnull().sum()
```

```
Out[25]:   RaceID          0
           Date            0
           Grand Prix      0
           Driver          0
           Car             0
           Pos             0
           PTS             0
           Laps            0
           PosQ           40
           Q1             40
           Q2            140
           Q3            239
           dtype: int64
```

Note: when checking for missing values we are only informed about the missing values in the qualifiers dataset but we know that there are observations under the 'Pos' column with values 'NC' which corresponds to not classified, and 'DQ' corresponding to disqualified. I therefore want to replace these with NaN values so that I can perform calculations on the 'Pos' column. I am keeping these in the final dataframe as these are real observations where drivers simply did not finish the race or did not qualify for the next round of qualifiers but we can still derive important insights from other columns corresponding to these observations

```
In [26]:   #replacing NC, DQ, and DNF values in 'Pos' column with 'Nan'
           F1_full.replace('NC', np.nan, inplace=True)
           F1_full.replace('DQ', np.nan, inplace=True)
           F1_full.replace('DNF', np.nan, inplace=True)
           F1_full.isnull().sum() #now we can clearly see the missing values in 'Pos' column corresponding to unclassified drivers
```

```
Out[26]:   RaceID          0
           Date            0
           Grand Prix      0
           Driver          0
           Car             0
           Pos            51
           PTS             0
           Laps            0
           PosQ           47
           Q1             44
           Q2            143
           Q3            243
           dtype: int64
```

Working with datatypes

```
In [27]:   #checking column datatypes
           print(F1_full.dtypes)
```
```
           RaceID               int32
           Date        datetime64[ns]
           Grand Prix          object
           Driver              object
           Car                 object
           Pos                 object
           PTS                  int64
           Laps                 int64
           PosQ                object
           Q1                  object
           Q2                  object
           Q3                  object
           dtype: object
```

```
In [28]:   #converting Pos and PosQ columns to integer datatype to allow us to perform calculations on this column
           F1_full['Pos'] = pd.to_numeric(F1_full['Pos'])
           F1_full['PosQ'] = pd.to_numeric(F1_full['PosQ'])
           print(F1_full.dtypes)
```
```
           RaceID               int32
           Date        datetime64[ns]
           Grand Prix          object
           Driver              object
           Car                 object
           Pos                float64
           PTS                  int64
           Laps                 int64
           PosQ               float64
           Q1                  object
           Q2                  object
           Q3                  object
           dtype: object
```

Cleaning the pit stop dataset

```
In [29]: #Adding identifier column to each pit stop dataframe
         Bahrain_pit['RaceID'] = '0'
         Saudi_arabia_pit['RaceID'] = '1'
         Australia_pit['RaceID'] = '2'
         Azerbaijan_pit['RaceID'] = '3'
         Miami_pit['RaceID'] = '4'
         Monaco_pit['RaceID'] = '5'
         Spain_pit['RaceID'] = '6'
         Canada_pit['RaceID'] = '7'
         Austria_pit['RaceID'] = '8'
         Great_britain_pit['RaceID'] = '9'
         Hungary_pit['RaceID'] = '10'
         Belgium_pit['RaceID'] = '11'
         Netherlands_pit['RaceID'] = '12'
         Italy_pit['RaceID'] = '13'
         Singapore_pit['RaceID'] = '14'
         Japan_pit['RaceID'] = '15'
         Qatar_pit['RaceID'] = '16'
         US_pit['RaceID'] = '17'
         Mexico_pit['RaceID'] = '18'
         Brazil_pit['RaceID'] = '19'
         Las_vegas_pit['RaceID'] = '20'
         Abu_dhabi_pit['RaceID'] = '21'
```

```
In [30]: #Combining all pit stop dataframes into 1 big dataframe
         pitstops = [Bahrain_pit, Saudi_arabia_pit,Australia_pit, Azerbaijan_pit, Miami_pit, Monaco_pit, Spain_pit, Canada_pit,
                     Austria_pit, Great_britain_pit, Hungary_pit, Belgium_pit, Netherlands_pit, Italy_pit, Singapore_pit,
                     Japan_pit, Qatar_pit, US_pit, Mexico_pit, Brazil_pit, Las_vegas_pit, Abu_dhabi_pit]
         F1_pit = pd.concat(pitstops) #joining up all pit stop dataframes vertically
```

```
In [31]: #modelling the dataset to keep only the columns necessary for analysis
         F1_pit = F1_pit [['RaceID', 'Driver', 'Car', 'Stops', 'Time', 'Total']]
```

```
In [32]: #renaming columns to make them clearer for my analysis
         F1_pit.rename(columns={'Stops': 'Pitstop_quantity', 'Time': 'Pit_time', 'Total': 'Total_Pit_time'}, inplace=True)
```

```
In [33]: #checking datatypes
         print(F1_pit.dtypes)

         RaceID              object
         Driver              object
         Car                 object
         Pitstop_quantity     int64
         Pit_time            object
         Total_Pit_time      object
         dtype: object
```

```
In [38]: #modifying datatypes - important error to note
         F1_pit['Pit_time'] = pd.to_numeric(F1_pit['Pit_time'])

         ---------------------------------------------------------------------------
         ValueError                                Traceback (most recent call last)
         pandas\_libs\lib.pyx in pandas._libs.lib.maybe_convert_numeric()

         ValueError: Unable to parse string "15:25.181"

         During handling of the above exception, another exception occurred:

         ValueError                                Traceback (most recent call last)
         <ipython-input-38-b99c81bf6f09> in <module>
               1 #modifying datatypes - important error to note
         ----> 2 F1_pit['Pit_time'] = pd.to_numeric(F1_pit['Pit_time'])

         ~\anaconda3\lib\site-packages\pandas\core\tools\numeric.py in to_numeric(arg, errors, downcast)
             152         coerce_numeric = errors not in ("ignore", "raise")
             153         try:
         --> 154             values = lib.maybe_convert_numeric(
             155                 values, set(), coerce_numeric=coerce_numeric
             156             )

         pandas\_libs\lib.pyx in pandas._libs.lib.maybe_convert_numeric()

         ValueError: Unable to parse string "15:25.181" at position 84
```

Note: the code above returns an error which gives us important information about the values in teh 'Pit_time' column. There are values in this column which cannot be turned into time format. Upon further investigation I found that Australia is the only race where this applies. Therefore I investigate the Australia dataframe further in the next step.

```
In [39]: F1_pit[F1_pit['RaceID'] == '2'] #investigating pit stop data for Austrialia race which is assigned to RaceID of 2
```
Out[39]:

	RaceID	Driver	Car	Pitstop_quantity	Pit_time	Total_Pit_time
0	2	Esteban Ocon OCO	Alpine Renault	1	18.056	18.056
1	2	Zhou Guanyu ZHO	Alfa Romeo Ferrari	1	18.951	18.951
2	2	Logan Sargeant SAR	Williams Mercedes	1	18.382	18.382
3	2	Sergio Perez PER	Red Bull Racing Honda RBPT	1	17.657	17.657
4	2	Valtteri Bottas BOT	Alfa Romeo Ferrari	1	21.659	21.659
...	...	...	...	...	...	...
60	2	Zhou Guanyu ZHO	Alfa Romeo Ferrari	5	31:04.998	61:25.244
61	2	Valtteri Bottas BOT	Alfa Romeo Ferrari	6	31:07.182	61:02.236
62	2	Sergio Perez PER	Red Bull Racing Honda RBPT	5	30:53.568	61:43.013
63	2	Fernando Alonso ALO	Aston Martin Aramco Mercedes	3	30:45.073	60:46.319
64	2	Lance Stroll STR	Aston Martin Aramco Mercedes	3	30:45.925	60:48.200

65 rows × 6 columns

Upon further investigating the Austrialia pit stop dataframe I found that there are observations where pit time is not the recorded pit stop duration but rather the time at which the pit stop occured which is not data we need. Therefore I proceed to use regular expressions to identify the observations with this type of input and drop them from the pit stop dataframe.

```
In [40]: #Fixing the error above by using regular expressions to match the values in the 'Pit_time' column which have a different pattern and drop these from the dataset as these a
         import re
         time_pattern = r'^\d{2}:\d{2}\.\d{3}$' #first string pattern to match
         F1_pit = F1_pit[~F1_pit['Pit_time'].astype(str).str.match(time_pattern)] #dropping the observations with matched string pattern
         time_pattern2 = r'^\d{1}:\d{2}\.\d{3}$' #second string patter match
         F1_pit = F1_pit[~F1_pit['Pit_time'].astype(str).str.match(time_pattern2)] #dropping observations wtih second string pattern
```

```
In [41]: F1_pit['Pit_time'] = pd.to_numeric(F1_pit['Pit_time']) #running the code to change the datatype again now that the anomalies have been removed
```

```
In [42]: #Repeating the procedure above for the total__pit_time column
         F1_pit = F1_pit[~F1_pit['Total_Pit_time'].astype(str).str.match(time_pattern)]
         F1_pit = F1_pit[~F1_pit['Total_Pit_time'].astype(str).str.match(time_pattern2)]
         F1_pit['Total_Pit_time'] = pd.to_numeric(F1_pit['Total_Pit_time']) #running the code to change the datatype again now that the anomalies have been removed
```

```
In [43]: F1_pit['RaceID'] = pd.to_numeric(F1_pit['RaceID']) #changing the raceID to numeric type to match the F1_full dataset raceID column
```

```
In [44]: F1_pit.dtypes #checking datatypes to ensure all necessary columns have been converted
```

```
Out[44]: RaceID              int64
         Driver              object
         Car                 object
         Pitstop_quantity    int64
         Pit_time            float64
         Total_Pit_time      float64
         dtype: object
```

Merging F1_full (race results) and F1_pit (pit stops) dataframes into one big dataframe to allow me to analyse pit stop against different race variables

```
In [46]: F1_pit_full = pd.merge(F1_full, F1_pit, on = ['RaceID', 'Driver'], how = 'inner') #merging F1_full and F1_pit dataframes
         F1_pit_full = F1_pit_full[['RaceID', 'Date', 'Grand Prix', 'Driver', 'Car_x', 'Pos', 'PTS', 'Laps',
                            'PosQ', 'Pitstop_quantity', 'Pit_time', 'Total_Pit_time' ]] #modelling the dataframes to the columns we want
         F1_pit_full.rename(columns={'Car_x': 'Car'}, inplace=True) #renaming columns which were duplicated in both dataframes
```

```
In [48]: F1_pit_full #viewing the final pit stop dataframe
```

Out[48]:

	RaceID	Date	Grand Prix	Driver	Car	Pos	PTS	Laps	PosQ	Pitstop_quantity	Pit_time	Total_Pit_time
0	0	2023-03-05	Bahrain	Max Verstappen VER	Red Bull Racing Honda RBPT	1.0	25	57	1.0	1	24.289	24.289
1	0	2023-03-05	Bahrain	Max Verstappen VER	Red Bull Racing Honda RBPT	1.0	25	57	1.0	2	24.910	49.199
2	0	2023-03-05	Bahrain	Sergio Perez PER	Red Bull Racing Honda RBPT	2.0	18	57	2.0	1	24.264	24.264
3	0	2023-03-05	Bahrain	Sergio Perez PER	Red Bull Racing Honda RBPT	2.0	18	57	2.0	2	25.091	49.355
4	0	2023-03-05	Bahrain	Fernando Alonso ALO	Aston Martin Aramco Mercedes	3.0	15	57	5.0	1	25.800	25.800
...	...	...	...	...	...	...	...	...	...	...	...	...
629	21	2023-11-26	Abu Dhabi	Carlos Sainz SAI	Ferrari	18.0	0	57	16.0	1	21.229	21.229
630	21	2023-11-26	Abu Dhabi	Carlos Sainz SAI	Ferrari	18.0	0	57	16.0	2	21.564	42.793
631	21	2023-11-26	Abu Dhabi	Valtteri Bottas BOT	Alfa Romeo Ferrari	19.0	0	57	18.0	1	22.665	22.665
632	21	2023-11-26	Abu Dhabi	Kevin Magnussen MAG	Haas Ferrari	20.0	0	57	17.0	1	22.764	22.764
633	21	2023-11-26	Abu Dhabi	Kevin Magnussen MAG	Haas Ferrari	20.0	0	57	17.0	2	22.163	44.927

634 rows × 12 columns

Saving the dataframes to CSV files

Now we have 2 dataframes: 1. Full race results, 2. Full pit stop data

```
In [49]: #saving a local copy of the dataframes in CSV format
         F1_full.to_csv('F1_full_scraped.csv', index = False)
         F1_pit_full.to_csv('F1_pit_scraped.csv', index=False)
```

Creating Insights

Team Performance

```
In [155]: #creating team leaderboard
          F1_full.groupby('Car')['PTS'].sum().sort_values(ascending = False)
          team_points = pd.DataFrame(data=F1_full.groupby('Car')['PTS'].sum().sort_values(ascending = False))
          team_points
```

Out[155]:

Car	PTS
Red Bull Racing Honda RBPT	790
Mercedes	374
Ferrari	363
Aston Martin Aramco Mercedes	266
McLaren Mercedes	266
Alpine Renault	110
Williams Mercedes	26
AlphaTauri Honda RBPT	22
Alfa Romeo Ferrari	16
Haas Ferrari	9

```
In [156]: F1_full['Car'].unique() #viewing all unique teams
```

```
Out[156]: array(['Red Bull Racing Honda RBPT', 'Aston Martin Aramco Mercedes',
                'Ferrari', 'Mercedes', 'Alfa Romeo Ferrari', 'Alpine Renault',
                'Williams Mercedes', 'AlphaTauri Honda RBPT', 'Haas Ferrari',
                'McLaren Mercedes'], dtype=object)
```

```
In [51]:  #setting team colours
          team_colours = {'Red Bull Racing Honda RBPT': '#FFCC00','Ferrari': '#DC0000',  'Mercedes' : '#00D2BE',
                          'Aston Martin Aramco Mercedes' : '#006F62' , 'McLaren Mercedes' : '#FF8000','Alpine Renault' : '#FD4BC7',
                          'AlphaTauri Honda RBPT' : '#00293F', 'Williams Mercedes' : '#00A3E0',
                          'Haas Ferrari' : '#F9F2F2', 'Alfa Romeo Ferrari': '#900000'}
          F1_full['Colours'] = F1_full['Car'].map(team_colours)
```

```
In [65]:  F1_full #checking that colours column has been added
```

Out[65]:

|	RaceID	Date	Grand Prix	Driver	Car	Pos	PTS	Laps	PosQ	Q1	Q2	Q3	Colours	Cumulative_PTS
0	0	2023-03-05	Bahrain	Max Verstappen VER	Red Bull Racing Honda RBPT	1.0	25	57	1.0	1:31.295	1:30.503	1:29.708	#FFCC00	25
1	0	2023-03-05	Bahrain	Sergio Perez PER	Red Bull Racing Honda RBPT	2.0	18	57	2.0	1:31.479	1:30.746	1:29.846	#FFCC00	43
2	0	2023-03-05	Bahrain	Fernando Alonso ALO	Aston Martin Aramco Mercedes	3.0	15	57	5.0	1:31.158	1:30.645	1:30.336	#006F62	23
3	0	2023-03-05	Bahrain	Carlos Sainz SAI	Ferrari	4.0	12	57	4.0	1:30.993	1:30.515	1:30.154	#DC0000	12
4	0	2023-03-05	Bahrain	Lewis Hamilton HAM	Mercedes	5.0	10	57	7.0	1:31.543	1:30.513	1:30.384	#00D2BE	16
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
434	21	2023-11-26	Abu Dhabi	Logan Sargeant SAR	Williams Mercedes	16.0	0	58	NaN	NaN	NaN	NaN	#00A3E0	26
435	21	2023-11-26	Abu Dhabi	Zhou Guanyu ZHO	Alfa Romeo Ferrari	17.0	0	58	19.0	1:25.159	NaN	NaN	#900000	16
436	21	2023-11-26	Abu Dhabi	Carlos Sainz SAI	Ferrari	18.0	0	57	16.0	1:24.738	NaN	NaN	#DC0000	345
437	21	2023-11-26	Abu Dhabi	Valtteri Bottas BOT	Alfa Romeo Ferrari	19.0	0	57	18.0	1:24.788	NaN	NaN	#900000	16
438	21	2023-11-26	Abu Dhabi	Kevin Magnussen MAG	Haas Ferrari	20.0	0	57	17.0	1:24.764	NaN	NaN	#F9F2F2	9

439 rows × 14 columns

```
In [66]:  F1_date_sort = F1_full.sort_values(by = 'Date') #sorting full F1 dataset by date
          F1_full['Cumulative_PTS'] = F1_date_sort.groupby('Car')['PTS'].cumsum() #creating column to hold cumulative values per car team
```

```
In [67]:  #sorting dataframe by points descending to help us see patterns more clearly
          F1_sorted = F1_full.sort_values(by='PTS', ascending = False).reset_index()
```

```
In [68]:  #creating a dictionary with each car assigned to its respective colour to apply to plots
          team_colours_func = dict(zip(F1_sorted['Car'], F1_sorted['Colours']))
```

```
In [69]:  #creating a time series linechart of cumulative points to investigate team performance over the season
          plt.figure(figsize = (15,12))
          sns.lineplot(data = F1_full, x = 'Date', y = 'Cumulative_PTS', hue = 'Car', marker = 'o', palette = team_colours_func)
          plt.xlabel('Date', fontsize = 12)
          plt.ylabel('Points per Team', fontsize = 12)
          plt.xticks(fontsize = 12)
          plt.yticks(fontsize = 12)
          plt.ylim(-10,800)
          plt.xlim(pd.Timestamp('2023-03'), pd.Timestamp('2023-12'))
          plt.title('Cumulative Points per Team Across Races ', fontsize = 16)
          plt.legend(title = 'Team', prop = {'size': 12}, fontsize = '10', title_fontsize = '12')
          plt.show()
```
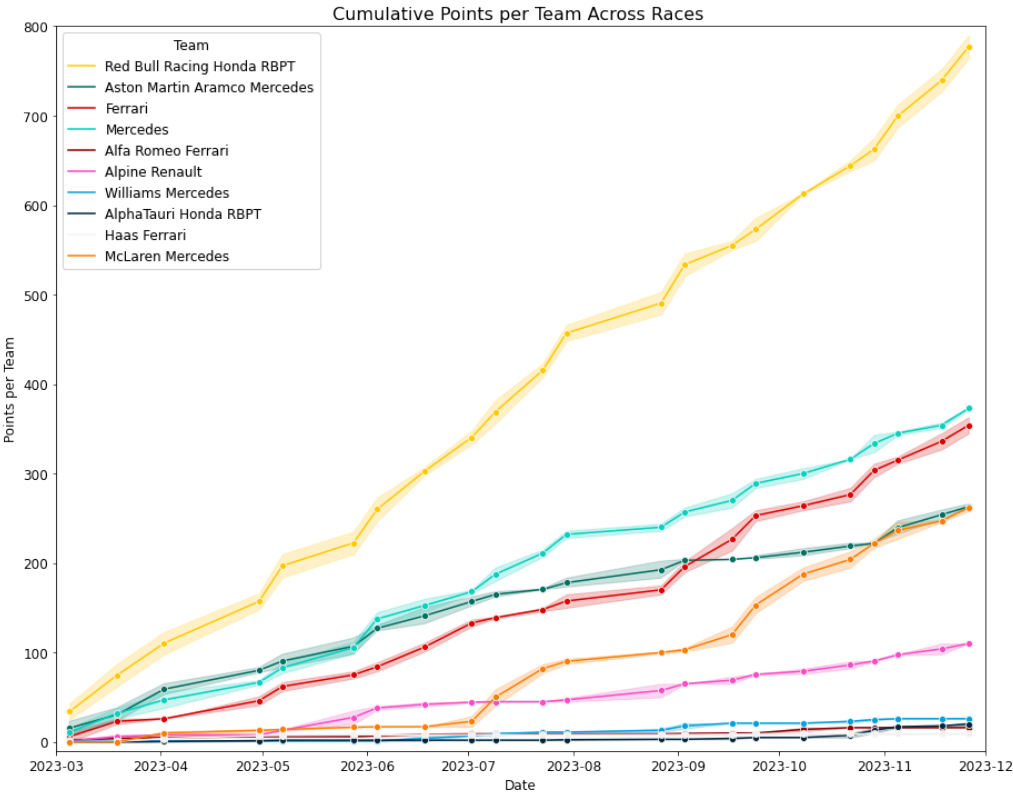
```
In [70]: #creating a boxplot to visualise the distribution of points scored per team
         plt.figure(figsize = (10,10))
         sns.boxplot(x = 'PTS', y = 'Car', data = F1_sorted, palette = team_colours_func)
         plt.title('Distribution of Points per Team ', fontsize = 16)
         plt.xlabel('Points per Team (per driver)', fontsize = 12)
         plt.ylabel('Team', fontsize = 12)
         plt.xticks(np.arange(0, 27, step=2),fontsize = 12)
         plt.yticks(fontsize = 12)
         plt.show()
```
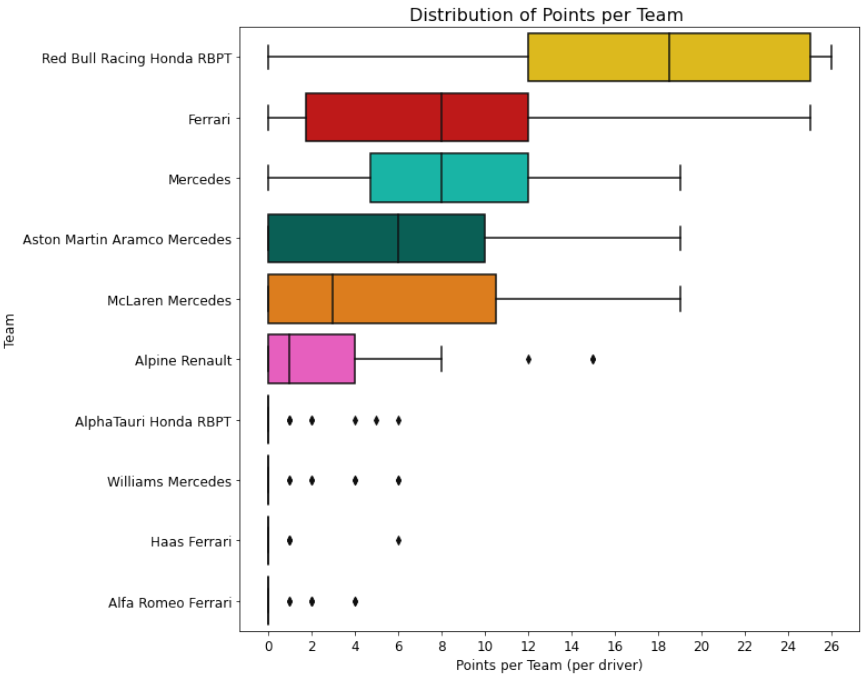

Driver Performance

```
In [71]: F1_full.groupby('Driver')['PTS'].sum().sort_values(ascending=False)
```

```
Out[71]: Driver
         Max Verstappen VER     530
         Sergio Perez PER       260
         Lewis Hamilton HAM     217
         Fernando Alonso ALO    198
         Charles Leclerc LEC    185
         Lando Norris NOR       184
         Carlos Sainz SAI       178
         George Russell RUS     157
         Oscar Piastri PIA       82
         Lance Stroll STR        68
         Esteban Ocon OCO        56
         Pierre Gasly GAS        54
         Alexander Albon ALB     25
         Yuki Tsunoda TSU        14
         Valtteri Bottas BOT     10
         Zhou Guanyu ZHO          6
         Nico Hulkenberg HUL      6
         Daniel Ricciardo RIC     6
         Kevin Magnussen MAG      3
         Liam Lawson LAW          2
         Logan Sargeant SAR       1
         Nyck De Vries DEV        0
         Name: PTS, dtype: int64
```

```
In [72]: F1_full.groupby('Driver')['PTS'].describe().round(2)
```

Out[72]:

Driver	count	mean	std	min	25%	50%	75%	max
Alexander Albon ALB	22.0	1.14	2.01	0.0	0.00	0.0	1.75	6.0
Carlos Sainz SAI	22.0	8.09	6.08	0.0	4.00	8.0	10.00	25.0
Charles Leclerc LEC	22.0	8.41	6.70	0.0	0.50	9.0	14.25	18.0
Daniel Ricciardo RIC	7.0	0.86	2.27	0.0	0.00	0.0	0.00	6.0
Esteban Ocon OCO	22.0	2.55	4.01	0.0	0.00	1.0	4.00	15.0
Fernando Alonso ALO	22.0	9.00	6.53	0.0	2.50	9.0	15.00	19.0
George Russell RUS	22.0	7.14	4.89	0.0	4.25	8.0	10.00	15.0
Kevin Magnussen MAG	22.0	0.14	0.35	0.0	0.00	0.0	0.00	1.0
Lance Stroll STR	21.0	3.24	4.11	0.0	0.00	1.0	6.00	12.0
Lando Norris NOR	22.0	8.36	7.47	0.0	0.50	7.0	17.25	19.0
Lewis Hamilton HAM	22.0	9.86	5.71	0.0	6.50	10.0	14.50	19.0
Liam Lawson LAW	5.0	0.40	0.89	0.0	0.00	0.0	0.00	2.0
Logan Sargeant SAR	22.0	0.05	0.21	0.0	0.00	0.0	0.00	1.0
Max Verstappen VER	22.0	24.09	3.78	10.0	25.00	25.0	26.00	26.0
Nico Hulkenberg HUL	22.0	0.27	1.28	0.0	0.00	0.0	0.00	6.0
Nyck De Vries DEV	10.0	0.00	0.00	0.0	0.00	0.0	0.00	0.0
Oscar Piastri PIA	22.0	3.73	5.47	0.0	0.00	0.5	5.50	18.0
Pierre Gasly GAS	22.0	2.45	3.90	0.0	0.00	0.5	3.50	15.0
Sergio Perez PER	22.0	11.82	7.40	0.0	8.25	12.0	17.25	25.0
Valtteri Bottas BOT	22.0	0.45	1.18	0.0	0.00	0.0	0.00	4.0
Yuki Tsunoda TSU	22.0	0.64	1.36	0.0	0.00	0.0	0.75	5.0
Zhou Guanyu ZHO	22.0	0.27	0.70	0.0	0.00	0.0	0.00	2.0

```
In [73]:   #Creating a dictionary to apply correct colour to each driver
           driver_colours_func = dict(zip(F1_sorted['Driver'], F1_sorted['Colours']))
```

```
In [74]:   F1_sorted #checking that the colours column has been applied
```

Out[74]:

	index	RaceID	Date	Grand Prix	Driver	Car	Pos	PTS	Laps	PosQ	Q1	Q2	Q3	Colours	Cumulative_PTS
0	419	21	2023-11-26	Abu Dhabi	Max Verstappen VER	Red Bull Racing Honda RBPT	1.0	26	58	1.0	1:24.160	1:23.740	1:23.445	#FFCC00	790
1	319	16	2023-10-08	Qatar	Max Verstappen VER	Red Bull Racing Honda RBPT	1.0	26	57	NaN	NaN	NaN	NaN	#FFCC00	612
2	160	8	2023-07-02	Austria	Max Verstappen VER	Red Bull Racing Honda RBPT	1.0	26	71	1.0	1:05.116	1:04.951	1:04.391	#FFCC00	333
3	299	15	2023-09-24	Japan	Max Verstappen VER	Red Bull Racing Honda RBPT	1.0	26	53	1.0	1:29.878	1:29.964	1:28.877	#FFCC00	586
4	200	10	2023-07-23	Hungary	Max Verstappen VER	Red Bull Racing Honda RBPT	1.0	26	70	NaN	NaN	NaN	NaN	#FFCC00	408
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
434	194	9	2023-07-09	Great Britain	Zhou Guanyu ZHO	Alfa Romeo Ferrari	15.0	0	52	17.0	1:30.123	NaN	NaN	#900000	9
435	193	9	2023-07-09	Great Britain	Lance Stroll STR	Aston Martin Aramco Mercedes	14.0	0	52	12.0	1:29.448	1:28.935	NaN	#006F62	162
436	192	9	2023-07-09	Great Britain	Nico Hulkenberg HUL	Haas Ferrari	13.0	0	52	11.0	1:29.603	1:28.896	NaN	#F9F2F2	8
437	191	9	2023-07-09	Great Britain	Valtteri Bottas BOT	Alfa Romeo Ferrari	12.0	0	52	NaN	1:29.798	NaN	NaN	#900000	9
438	438	21	2023-11-26	Abu Dhabi	Kevin Magnussen MAG	Haas Ferrari	20.0	0	57	17.0	1:24.764	NaN	NaN	#F9F2F2	9

439 rows × 15 columns

```
In [75]:   #creating a boxplot to visualise the distribution of points scored per driver
           plt.figure(figsize = (10,10))
           sns.boxplot(data = F1_sorted, x = 'PTS', y = 'Driver', palette = driver_colours_func)
           plt.xlabel('Distribution of Points Scored per Race per Driver', fontsize = 12)
           plt.ylabel('Driver', fontsize = 12)
           plt.title('Number of Points Scored per Race', fontsize = 16)
           plt.xticks(fontsize = 12)
           plt.yticks(fontsize = 12)
           plt.show()
```
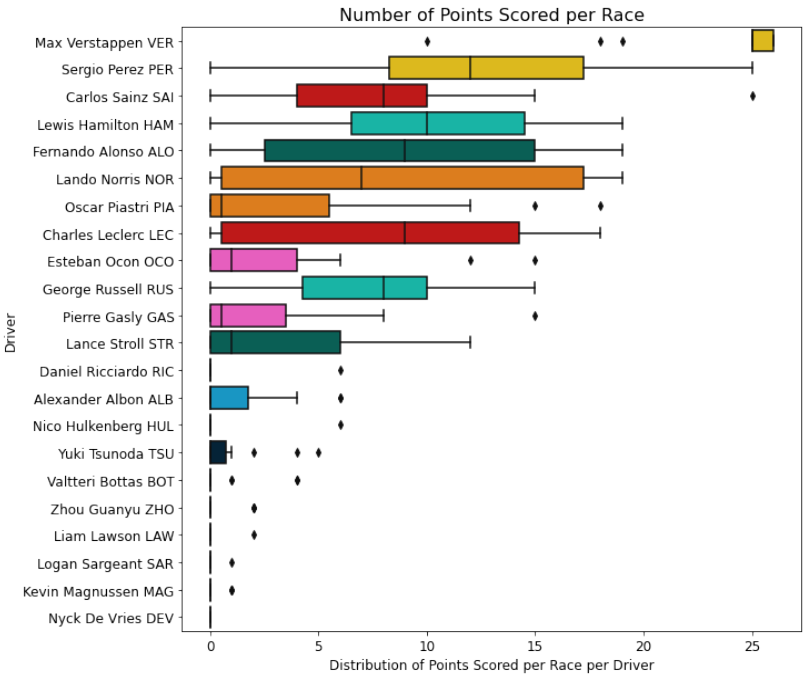
```
In [76]: #creating a heatmap of driver race finishing positions
         plt.figure(figsize = (10,10))
         driver_pivot = F1_sorted.pivot_table(index = 'Driver', columns = 'Pos', aggfunc = 'size', fill_value = 0)
         sns.heatmap(driver_pivot, annot = True,  cmap = 'YlOrRd', fmt = 'd', cbar_kws ={ 'label': 'Frequency'})
         plt.xlabel('Finishing Race Position', fontsize = 12)
         plt.yticks(fontsize = 12)
         plt.xticks(fontsize = 12, ticks = plt.xticks()[0], labels = [int(label) for label in plt.xticks()[0]], rotation = 0)
         plt.title('Distribution of Race Finish Positions per Driver ', fontsize = 16)
         plt.show()
```
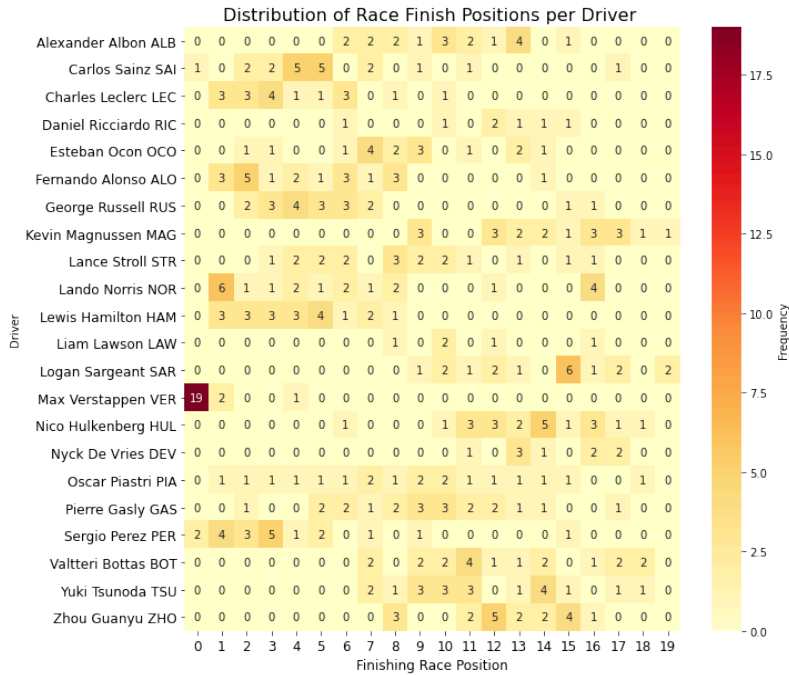

Qualifying Position vs Race Finish Position

```
In [77]: #Creating a measure for the difference between final race finishing positions and qualifying position
         F1_full['pos_diff'] = F1_full['PosQ'] - F1_full['Pos']
```

```
In [78]: #creating a scatterplot to visualise qualifying position vs finishing position
         plt.figure(figsize = (11,11))
         sns.regplot(data = F1_sorted, x = 'PosQ', y = 'Pos', color = 'red', marker = 'o')
         plt.xticks(np.arange(1, 21, 1), fontsize = 12)
         plt.yticks(np.arange(1, 21, 1), fontsize = 12)
         plt.xlim(0,21)
         plt.ylim(0,21)
         plt.xlabel('Qualifying Position', fontsize = 12)
         plt.ylabel('Finishing Position', fontsize = 12)
         plt.title('Relationship between qualifying position and race finish position', fontsize = 16)
         correlation_coef = F1_full['PosQ'].corr(F1_full['Pos']).round(2)
         plt.text(0.5, 20, f'Correlation Coefficient: {correlation_coef}', fontsize = 14, bbox = dict(facecolor = 'white'))
         plt.show()
```
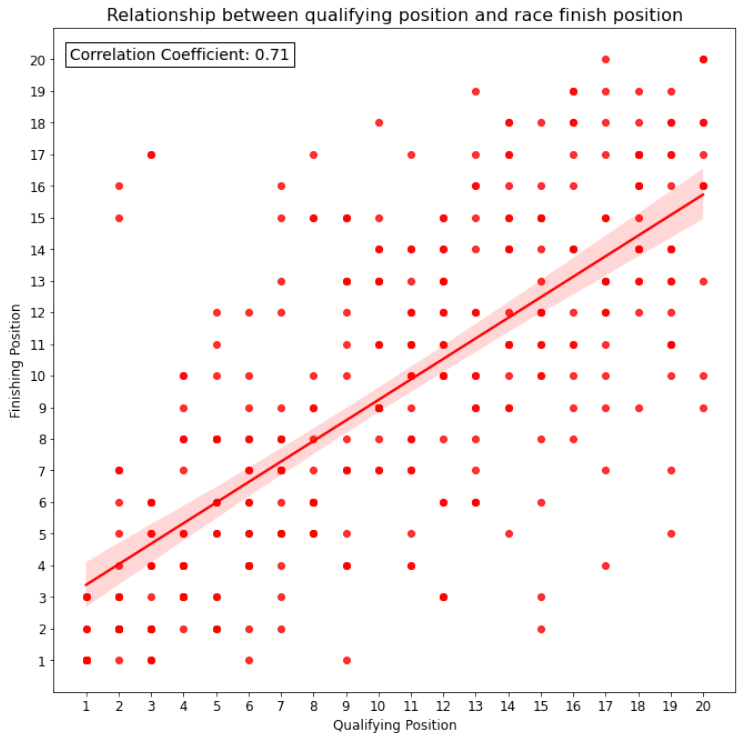
Pit Stops

```
In [79]: F1_pit.groupby('Car')['Pit_time'].describe().round(2)#pit stop summary statistics for each team
```

Out[79]:

Car	count	mean	std	min	25%	50%	75%	max
Alfa Romeo Ferrari	68.0	25.29	5.81	17.13	22.32	24.04	26.00	55.80
AlphaTauri Honda RBPT	63.0	24.45	4.20	16.25	21.97	23.49	26.34	42.64
Alpine Renault	58.0	23.95	3.24	16.38	22.05	23.74	25.49	31.80
Aston Martin Aramco Mercedes	65.0	23.63	2.84	16.60	22.37	23.44	24.82	30.67
Ferrari	63.0	23.62	3.27	16.38	21.85	23.15	24.42	32.89
Haas Ferrari	65.0	25.36	5.47	16.78	22.16	24.20	25.45	51.05
McLaren Mercedes	60.0	24.17	4.10	17.32	21.41	23.42	25.85	37.74
Mercedes	63.0	23.29	3.63	16.42	21.52	22.31	24.38	39.69
Red Bull Racing Honda RBPT	69.0	23.20	3.48	16.37	21.32	22.67	24.29	33.60
Williams Mercedes	60.0	24.92	5.65	16.74	21.91	23.94	25.62	50.14

```
In [82]: #creating a box plot to visualise the distribution of pit stop times per team
         plt.figure(figsize = (10,10))
         sns.boxplot(data = F1_pit_full, x = 'Pit_time', y = 'Car', color = 'red', palette = team_colours_func)
         plt.xlabel('Pit Time (overall per race, seconds)', fontsize = 12)
         plt.ylabel('Team',fontsize = 12)
         plt.xticks(fontsize = 12)
         plt.yticks(fontsize = 12)
         plt.title('Distribution of Pit Stop Times per Team ', fontsize = 16)
         plt.show()
```
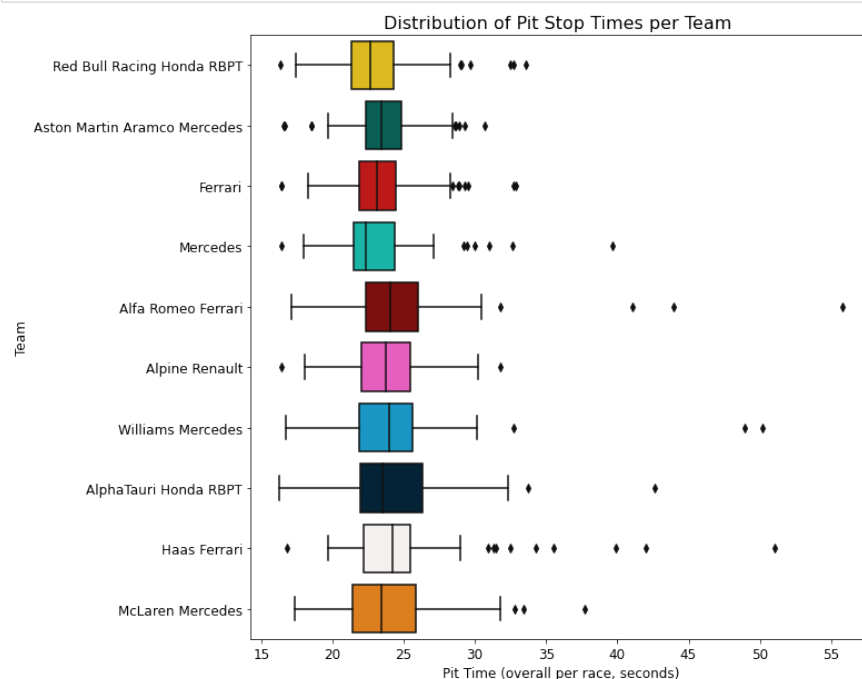

Regression

```
In [83]: #importing packages
         from statsmodels.regression.linear_model import OLS
         from statsmodels.tools import add_constant
```

```
In [84]: #setting independent and dependent variables
         F1_clean = F1_pit_full.dropna()#dropping columns with Nan values as these will prevent the regression from being carried out
         X = F1_clean[['PosQ','Pit_time' ]] #assigning independent variables
         Y = F1_clean['PTS'] #assigning dependent variable
```

```
In [85]: X = add_constant(X) #adding a constant term to the regression
```

```
In [86]: regression = OLS(Y, X)
         regression_results = regression.fit()
         print(regression_results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    PTS   R-squared:                       0.481
Model:                            OLS   Adj. R-squared:                  0.479
Method:                 Least Squares   F-statistic:                     241.5
Date:                Wed, 24 Apr 2024   Prob (F-statistic):           5.51e-75
Time:                        10:31:59   Log-Likelihood:                -1634.0
No. Observations:                 525   AIC:                             3274.
Df Residuals:                     522   BIC:                             3287.
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         17.2190      1.468     11.726      0.000      14.334      20.104
PosQ          -0.8956      0.042    -21.525      0.000      -0.977      -0.814
Pit_time      -0.0894      0.060     -1.480      0.139      -0.208       0.029
==============================================================================
Omnibus:                       23.331   Durbin-Watson:                   0.831
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               25.323
Skew:                           0.504   Prob(JB):                     3.17e-06
Kurtosis:                       3.375   Cond. No.                         163.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```