Sports Car Project

By: Tilak Muley

Background Of The Data Chosen

Two datasets were used. One of the data used was: "Sports Car Prices dataset" which was name SportCar for this project. The dataset was found on kaggle.com(https://www.kaggle.com/datasets/rkiattisak/sports-car-prices-dataset/code). This contains data regarding Sport cars so the dataset has many variables: Car Make, Car Model, Year, Engine Size (L), Horsepower, Torque (lb-ft), 0-60 MPH Time (seconds) and Price (in USD). Car Make is the make/company of the car. Car Model is the model/version/variant of the sport car. Year of the sport car is the year the sport car was produced. Engine Size (L) is the size of the cars engine in liters and in some sport cars cases it could be a electric motor. Horsepower is the power output of the sport car engines. Torque (lb-ft) is the force produced by the sport cars engine and determines the acceleration of the car. 0-60 MPH Time (seconds) is the time it takes the sport car to accelerate from 0 MPH to 60 MPH. Price (in USD) is the purchasing price of the sport car in USD.

The second dataset used was Auto Groups which was named AutoGroups for this project this was a dataset created by me. The dataset includes Car Make, Country and Car Group. Car Make being the company, Country being the origin of the Car Make and Car group being what group the Car Make is apart of.

Goal

Do some data analysis to better understand both the datasets, with some EDA, data cleaning, data visualization, data wrangling, data imputations, and data aggregation. As well as find some answers to questions such as: What Country makes the most powerful car determined by horsepower in each price bracket? As well as what is each Car Groups fastest car based on 0-60 MPH Time? Also how much correlation is between horsepower, torque and 0-60 MPH Time?

Loading Dataset and Libraries

In [1]:
```
import pandas as pd
import seaborn as sns
import numpy as np
import datetime
import statsmodels.api as stat
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

```
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeRegressor
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
from matplotlib import pyplot as plt
```

Load Dataset

In [2]:
```
 #dataset assigned Sc
Sc= pd.read_csv('SportCar.csv',encoding= 'latin1')
```

In [3]:
```
AG= pd.read_csv('AutoGroups.csv',encoding= 'latin1')
```

Exploratory Data Analysis on AutoGroups Data

In [4]:
```
#First five of data
AG.head(5)
```

Out[4]:

	Car Make	Country	Car Group
0	Acura	Japan	Honda
1	Alfa Romeo	Italy	FCA
2	Alpine	France	Renault Group
3	Ariel	England	Ariel Motor Car Company
4	Aston Martin	England	Premier Automotive Group

In [5]:
```
#First five of data
AG.tail(5)
```

Out[5]:

	Car Make	Country	Car Group
33	TVR	England	TVR
34	Tesla	America	Tesla
35	Toyota	Japan	Toyota Motor Corporation
36	Ultima	England	the Subaru Corporation
37	W Motors	Dubai	W Motors

In [6]:
```
#Shape of data
AG.shape
```

Out[6]:
```
(38, 3)
```

In [7]:
```
#Information on data
AG.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 38 entries, 0 to 37
Data columns (total 3 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Car Make   38 non-null     object
 1   Country    38 non-null     object
 2   Car Group  38 non-null     object
dtypes: object(3)
memory usage: 1.0+ KB
```

In [8]:
```
#check for missing values and print each column and the sum of each column
AG.isnull().sum()
```

Out[8]:
```
Car Make     0
Country      0
Car Group    0
dtype: int64
```

Exploratory Data Analysis on SportCar Data

In [9]:
```
#First five of data
Sc.head(5)
```

Out[9]:

	Car Make	Car Model	Year	Engine Size (L)	Horsepower	Torque (lb-ft)	0-60 MPH Time (seconds)	Price (in USD)
0	Porsche	911	2022	3	379	331	4	101,200
1	Lamborghini	Huracan	2021	5.2	630	443	2.8	274,390
2	Ferrari	488 GTB	2022	3.9	661	561	3	333,750
3	Audi	R8	2022	5.2	562	406	3.2	142,700
4	McLaren	720S	2021	4	710	568	2.7	298,000

In [10]:
```
#Last five of data
Sc.tail(5)
```

Out[10]:

	Car Make	Car Model	Year	Engine Size (L)	Horsepower	Torque (lb-ft)	0-60 MPH Time (seconds)	Price (in USD)
1002	Koenigsegg	Jesko	2022	5	1280	1106	2.5	3,000,000
1003	Lotus	Evija	2021	Electric Motor	1972	1254	2	2,000,000
1004	McLaren	Senna	2021	4	789	590	2.7	1,000,000
1005	Pagani	Huayra	2021	6	764	738	3	2,600,000
1006	Rimac	Nevera	2021	Electric Motor	1888	1696	1.85	2,400,000

In [11]:
```
#Shape of data
Sc.shape
```

Out[11]:
```
(1007, 8)
```

In [12]:
```
Sc.columns.tolist()
```

Out[12]:
```
['Car Make',
 'Car Model',
 'Year',
 'Engine Size (L)',
 'Horsepower',
 'Torque (lb-ft)',
 '0-60 MPH Time (seconds)',
 'Price (in USD)']
```

All of the column names were printed in a list to get an idea of what the original data set has.

In [13]:
```
#Describing the data
Sc.describe()
```

Out[13]:

	Year
count	1007.000000
mean	2021.201589
std	2.019802
min	1965.000000
25%	2021.000000
50%	2021.000000
75%	2022.000000
max	2023.000000

With only year coming up when using .describe() that means the other columns/variable are not being recognized as numerical data. Which will be checked below to see what data type each column/variable is assigned.

In [14]:
```
#Information on data
Sc.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1007 entries, 0 to 1006
Data columns (total 8 columns):
 #   Column                   Non-Null Count  Dtype 
---  ------                   --------------  ----- 
 0   Car Make                 1007 non-null   object
 1   Car Model                1007 non-null   object
 2   Year                     1007 non-null   int64 
 3   Engine Size (L)          997 non-null    object
 4   Horsepower               1007 non-null   object
 5   Torque (lb-ft)           1004 non-null   object
 6   0-60 MPH Time (seconds)  1007 non-null   object
 7   Price (in USD)           1007 non-null   object
dtypes: int64(1), object(7)
memory usage: 63.1+ KB
```

Above all of the variables in the original data set have a data type of object except for year which is a int64. This is not ideal as columns like 'Engine Size (L)', 'Horsepower', 'Torque (lb-ft)', '0-60 MPH Time (seconds)', and 'Price (in USD)' should be represented as numeric types such as int or float. The data types will be modified later on.

In [15]:
```
#check for missing values and print each column and the sum of each column
Sc.isnull().sum()
```

Out[15]:
```
Car Make                   0
Car Model                  0
Year                       0
Engine Size (L)           10
Horsepower                 0
Torque (lb-ft)             3
0-60 MPH Time (seconds)    0
Price (in USD)             0
dtype: int64
```

First it is checked for missing values and then returns the sum of null values for each column. This gives us a feel for the data set and how many missing values we may have to deal with.

In [16]:
```
#checking duplicate values
Sc.nunique()
```

Out[16]:
```
Car Make                    38
Car Model                  176
Year                         9
Engine Size (L)             45
Horsepower                 124
Torque (lb-ft)              93
0-60 MPH Time (seconds)     43
Price (in USD)             367
dtype: int64
```

.unique() gives us the amount of unique values in each column of this dataset. Which helps us get an idea of exactly how many unique values exists and the variety.

In [17]:
```
#Filter rows with missing values
rows_missing_values = Sc[Sc.isnull().any(axis=1)]

#Print rows with missing values
rows_missing_values
```

Out[17]:

	Car Make	Car Model	Year	Engine Size (L)	Horsepower	Torque (lb-ft)	0-60 MPH Time (seconds)	Price (in USD)
168	Rimac	C_Two	2022	NaN	1914	1696	1.9	2,400,000
171	Tesla	Model S Plaid	2021	NaN	1020	1050	1.98	131,190
222	Porsche	Taycan Turbo S	2021	NaN	750	774	2.6	185,000
247	Tesla	Model S Plaid	2022	NaN	1020	1050	1.9	131,190
387	Rimac	C_Two	2022	NaN	1888	1696	1.8	2,400,000
389	Tesla	Roadster	2022	NaN	10000+	0	1.9	200,000
642	Tesla	Model S Plaid	2021	Electric	1020	NaN	1.9	139,990
686	Rimac	C_Two	2022	NaN	1914	1696	1.85	2,400,000
697	Lotus	Evija	2022	NaN	1972	1254	2.5	2,700,000
752	Porsche	Taycan	2022	NaN	469	479	3.8	79,900
878	Maserati	GranTurismo	2021	Electric	550	NaN	2.8	200,000
916	Tesla	Roadster	2022	NaN	10,000+	NaN	1.9	200,000

In [18]:
```
#Sum of the number of rows with missing values
rows_missing_values_count = Sc.isnull().any(axis=1).sum()

print("Number of rows with missing values:", rows_missing_values_count)
```

```
Number of rows with missing values: 12
```

Number of rows with missing values and the rows that are missing values are displayed. This will help decide if the rows with missing values will be completly removed or the missing values will be replaced with a value to help keep the row.

No plots can be made as of right now until data types are changed

Data Imputations

In [19]:
```
#New data set is made by copying original data set
Sc2=Sc.copy()
```

In [20]:
```
#Replace missing values in 'Engine Size (L)' column with 'Electric'
Sc2['Engine Size (L)'].fillna('Electric', inplace=True)
```

After futher review it was found that all rows missing a value in the 'Engine Size (L)' column were all cars with 'Electric' engines so the missing values in the column were replaced with 'Electric'

In [21]:
```
#Filter rows with missing values
rows_missing_values = Sc2[Sc2.isnull().any(axis=1)]

#Print rows with missing values
rows_missing_values
```

Out[21]:

	Car Make	Car Model	Year	Engine Size (L)	Horsepower	Torque (lb-ft)	0-60 MPH Time (seconds)	Price (in USD)
642	Tesla	Model S Plaid	2021	Electric	1020	NaN	1.9	139,990
878	Maserati	GranTurismo	2021	Electric	550	NaN	2.8	200,000
916	Tesla	Roadster	2022	Electric	10,000+	NaN	1.9	200,000

In [22]:
```
#Rows with missing values dropped
Sc2.dropna()
```

Out[22]:

	Car Make	Car Model	Year	Engine Size (L)	Horsepower	Torque (lb-ft)	0-60 MPH Time (seconds)	Price (in USD)
0	Porsche	911	2022	3	379	331	4	101,200
1	Lamborghini	Huracan	2021	5.2	630	443	2.8	274,390
2	Ferrari	488 GTB	2022	3.9	661	561	3	333,750
3	Audi	R8	2022	5.2	562	406	3.2	142,700
4	McLaren	720S	2021	4	710	568	2.7	298,000
...	...	...	...	...	...	...	...	...
1002	Koenigsegg	Jesko	2022	5	1280	1106	2.5	3,000,000
1003	Lotus	Evija	2021	Electric Motor	1972	1254	2	2,000,000
1004	McLaren	Senna	2021	4	789	590	2.7	1,000,000
1005	Pagani	Huayra	2021	6	764	738	3	2,600,000
1006	Rimac	Nevera	2021	Electric Motor	1888	1696	1.85	2,400,000

1004 rows × 8 columns

In [23]:
```
Sc2.dropna(inplace=True)
```

Rows with missing values in the 'Torque (lb-ft)' were removed as averaging all the values in this column would not make sense as each car has a different torque value and a solution without looking up values couldn't be found.

In [24]:
```
#Replace Electric/Electric Motor with 0 in the 'Engine Size (L)' column
Sc2['Engine Size (L)'] = Sc2['Engine Size (L)'].replace(['Electric', 'Electric Motor']
```

To help clean up the dataset and get the 'Engine Size (L)' column ready for a data type change any Electric/ Electric motor was changed to a 0 as Electric

vehicles don't have a engine size.

Data Cleaning

In [25]:
```
Sc2.drop_duplicates(inplace= True)
Sc2
```

Out[25]:

	Car Make	Car Model	Year	Engine Size (L)	Horsepower	Torque (lb-ft)	0-60 MPH Time (seconds)	Price (in USD)
0	Porsche	911	2022	3	379	331	4	101,200
1	Lamborghini	Huracan	2021	5.2	630	443	2.8	274,390
2	Ferrari	488 GTB	2022	3.9	661	561	3	333,750
3	Audi	R8	2022	5.2	562	406	3.2	142,700
4	McLaren	720S	2021	4	710	568	2.7	298,000
...	...	...	...	...	...	...	...	...
999	Nissan	370Z	2021	3.7	332	270	5.1	30,090
1002	Koenigsegg	Jesko	2022	5	1280	1106	2.5	3,000,000
1003	Lotus	Evija	2021	0	1972	1254	2	2,000,000
1005	Pagani	Huayra	2021	6	764	738	3	2,600,000
1006	Rimac	Nevera	2021	0	1888	1696	1.85	2,400,000

714 rows × 8 columns

To start cleaning the data any duplicates in the dataset was removed to help get the data ready and only have a single row for each car model so no data is repeating and skewing any results. So data drops from 1004 to 717 observations.

In [26]:
```
Sc2['Price (in USD)']= Sc2['Price (in USD)'].str.replace(',','',regex= True)
```

To get some of the columns ready to convert to a numerical value for the data type any non-numeric characters need to be removed so the comma in the column 'Price (in USD)' needs to be removed first

In [27]:
```
#All relevant columns were converted to numeric data types
numeric_columns = ['Engine Size (L)', 'Horsepower', 'Torque (lb-ft)', '0-60 MPH Time (
Sc2[numeric_columns] = Sc2[numeric_columns].apply(pd.to_numeric, errors='coerce')
```

Exploratory Data Analysis for New SportCar Data

In [28]:
```
# Display updated data types
print(Sc2.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 714 entries, 0 to 1006
Data columns (total 8 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   Car Make                 714 non-null    object 
 1   Car Model                714 non-null    object 
 2   Year                     714 non-null    int64  
 3   Engine Size (L)          704 non-null    float64
 4   Horsepower               706 non-null    float64
 5   Torque (lb-ft)           709 non-null    float64
 6   0-60 MPH Time (seconds)  713 non-null    float64
 7   Price (in USD)           714 non-null    int64  
dtypes: float64(4), int64(2), object(2)
memory usage: 50.2+ KB
None
```

Columns: Engine Size (L),Horsepower,Torque (lb-ft),0-60 MPH Time (seconds) were all changed to float64 data types while column: Price (in USD) was changed to int64 data type. Car Make, Car Model can remain objects because they are just letters or words. As with Year it can remain int64 data type.

Data Wrangling

In [29]:
```
#Merging the two datasets together
Scag= Sc2.merge(AG,on= 'Car Make')
Scag
```

Out[29]:

	Car Make	Car Model	Year	Engine Size (L)	Horsepower	Torque (lb-ft)	0-60 MPH Time (seconds)	Price (in USD)	Country	Car Grou
0	Porsche	911	2022	3.0	379.0	331.0	4.0	101200	Germany	Volkswage Grou
1	Porsche	Boxster	2021	2.0	300.0	280.0	4.7	62000	Germany	Volkswage Grou
2	Porsche	Cayman GT4	2022	4.0	414.0	309.0	3.8	100200	Germany	Volkswage Grou
3	Porsche	Taycan 4S	2022	0.0	562.0	479.0	3.8	104000	Germany	Volkswage Grou
4	Porsche	Cayman	2021	2.0	300.0	280.0	5.1	58900	Germany	Volkswage Grou
...	...	...	...	...	...	...	...	...	...	
709	Pininfarina	Battista	2022	0.0	1874.0	1696.0	1.9	2500000	Itlay	Pininfarir
710	Pininfarina	Battista	2021	0.0	1872.0	1696.0	1.9	2500000	Itlay	Pininfarir
711	Kia	Stinger	2022	3.3	368.0	376.0	4.7	52200	South Korea	Hyund Mot Grou
712	Alpine	A110	2021	1.8	288.0	236.0	4.4	71500	France	Renau Grou
713	Ultima	RS	2021	6.2	1200.0	1300.0	2.3	220000	England	the Subar Corporatic

714 rows × 10 columns

AutoGroups Data and SportCar data were merged together on the column Car Make

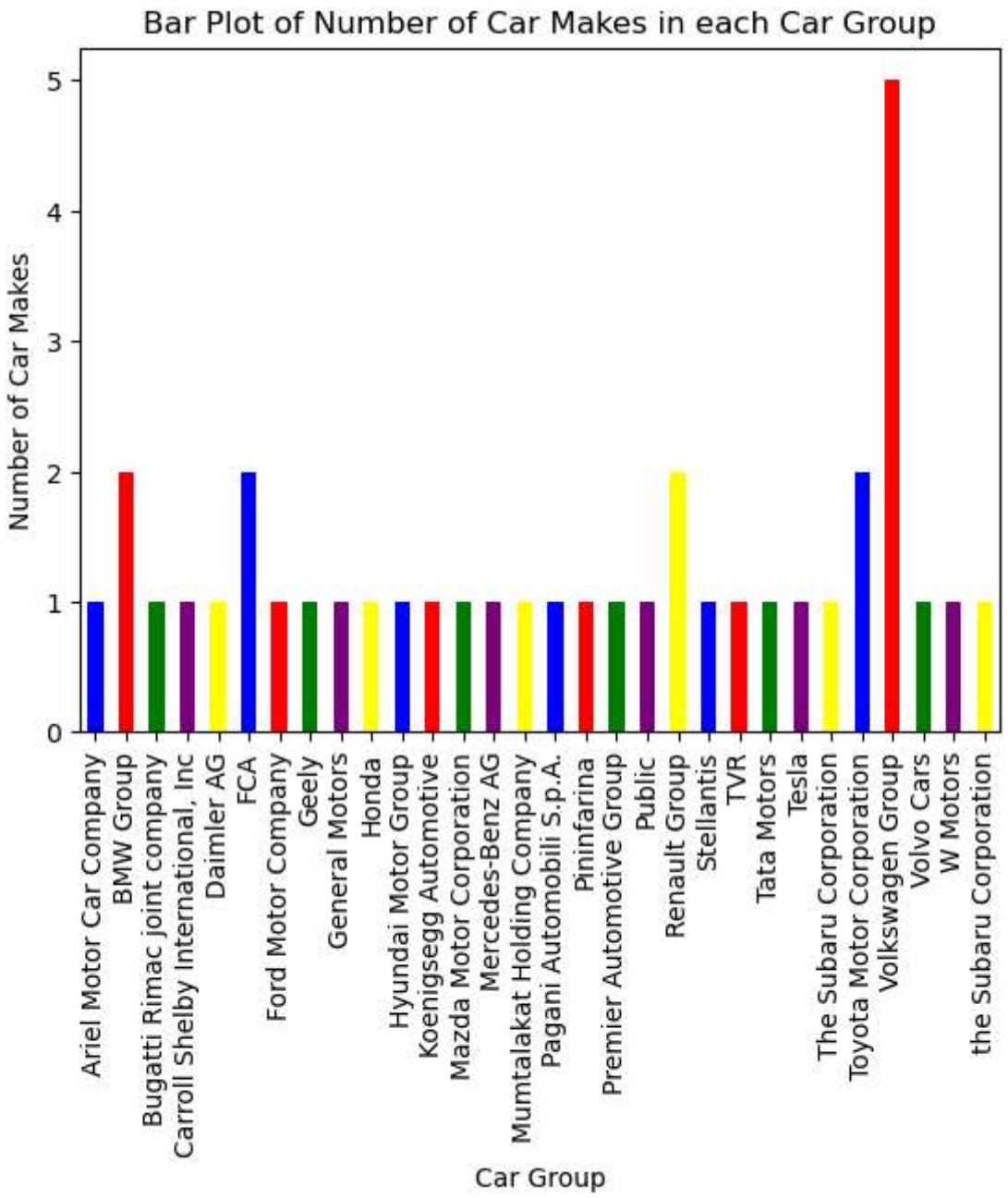
Exploratory Data Analysis for Merged Data

Data Visualization

In [30]:

```
#Grouped
grouped= Scag.groupby("Car Group")["Car Make"].nunique()

#bar plot
colors = ['blue', 'red','green','purple','yellow'] * (len(grouped) // 2 + 1)

# Create a bar plot with alternating colors
grouped.plot(kind='bar', color=colors[:len(grouped)])

#Tittle and labels
plt.title('Bar Plot of Number of Car Makes in each Car Group')
plt.xlabel('Car Group')
```

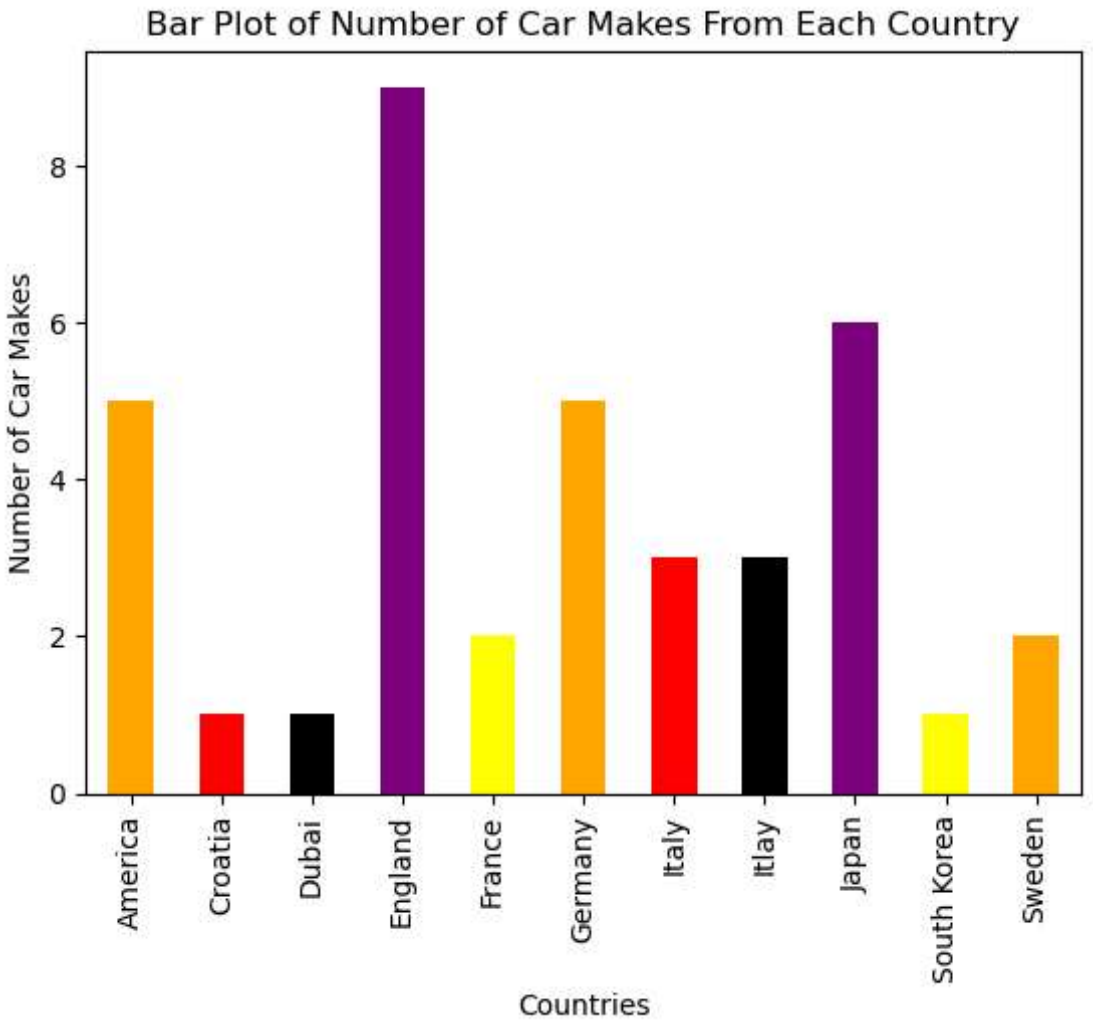
```
plt.ylabel('Number of Car Makes')

#plot
plt.show()
```



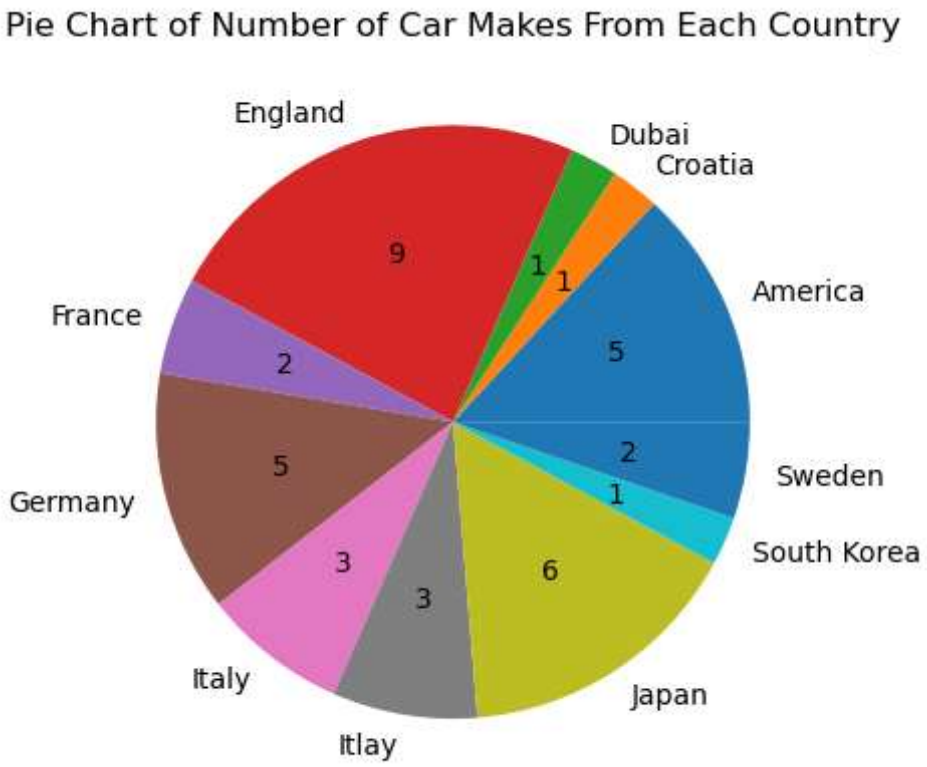
Bar graph above helps visualize how many Car Makes are in each Car Group. The .groupby() was used to make this. With Volkswagen Group leading the way with five.

In [31]:
```
#Grouped
grouped= Scag.groupby("Country")["Car Make"].nunique()


#Bar plot with alternating colors
colors = ['orange', 'red','black','purple','yellow'] * (len(grouped) // 2 + 1)

grouped.plot(kind='bar', color=colors[:len(grouped)])
```

```
#Title and labels
plt.title('Bar Plot of Number of Car Makes From Each Country')
plt.xlabel('Countries')
plt.ylabel('Number of Car Makes')

#plot
plt.show()
```



In [32]:
```
fig, ax = plt.subplots()

ax.pie(grouped, labels = grouped.index, autopct=lambda p: '{:.0f}'.format(p * sum(grou

plt.title('Pie Chart of Number of Car Makes From Each Country')
plt.show()
```



Bar plot and the pie chart above help visualize how many Car Makes from each Country. The .groupby() was used to make this. With England leading the way with 8 Car Makes from its country.

Data Aggregation

Most Powerful Car in Each Price Bracket by Horsepower

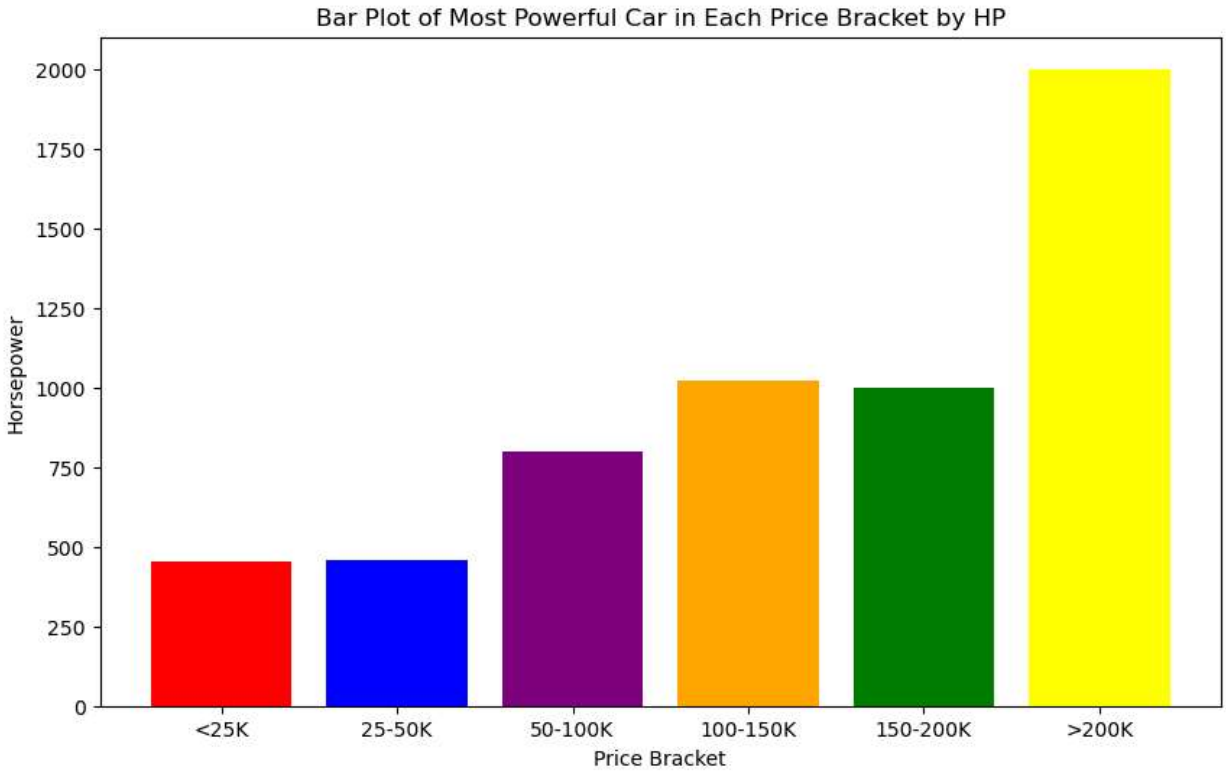
In [33]:
```
#price brackets
bins = [0, 25000, 50000, 100000, 150000, 200000, float('inf')]
labels = ['<25K', '25-50K','50-100K', '100-150K', '150-200K', '>200K']
Scag['Price Bracket'] = pd.cut(Scag['Price (in USD)'], bins=bins, labels=labels)

#Powerful car in each price bracket
most_powerful_price = Scag.loc[Scag.groupby('Price Bracket')['Horsepower'].idxmax()]

most_powerful_price#
```

Out[33]:

	Car Make	Car Model	Year	Engine Size (L)	Horsepower	Torque (lb-ft)	0-60 MPH Time (seconds)	Price (in USD)	Country	Car Group
351	Chevrolet	Camaro	2021	6.2	455.0	455.0	4.00	25000	America	General Motors
367	Ford	Mustang GT	2022	5.0	460.0	420.0	4.30	39000	America	Ford Motor Company
473	Dodge	Challenger SRT Hellcat Redeye	2022	6.2	797.0	707.0	3.40	78595	America	FCA
681	Tesla	Model S Plaid	2021	0.0	1020.0	1050.0	1.98	131190	America	Tesla
694	Tesla	Roadster	2022	0.0	1000.0	737.0	1.90	200000	America	Tesla
567	Lotus	Evija	2021	0.0	2000.0	1254.0	2.80	2800000	England	Geely

In [34]:

```
#bar plot
plt.figure(figsize=(10,6))
c= 'red','blue','purple','orange','green','yellow'
plt.bar(most_powerful_price['Price Bracket'], most_powerful_price['Horsepower'], color
plt.xlabel('Price Bracket')
plt.ylabel('Horsepower')
plt.title('Bar Plot of Most Powerful Car in Each Price Bracket by HP')

#plot
plt.show()
```



6 Price brackets were created to see which country makes the most powerful car determined by horsepower in each price bracket. The bracket labels ended up being '<25K', '25-50K','50-100K', '100-150K', '150-200K', '>200K'. The results ended up being Car Makes made in America holding 5 of the first 6 spots ('<25K', '25-50K','50-100K', '100-150K', '150-200K',) and England getting the last spot (>200k). Then the bar plot made of Price Bracket and Horsepower was a bit interesting to see as rice increases so does horsepower but 100-150k price bracket actually has a car with more horsepower then the one in the 150-200k bracket.

Fastest Car in Each Car Group

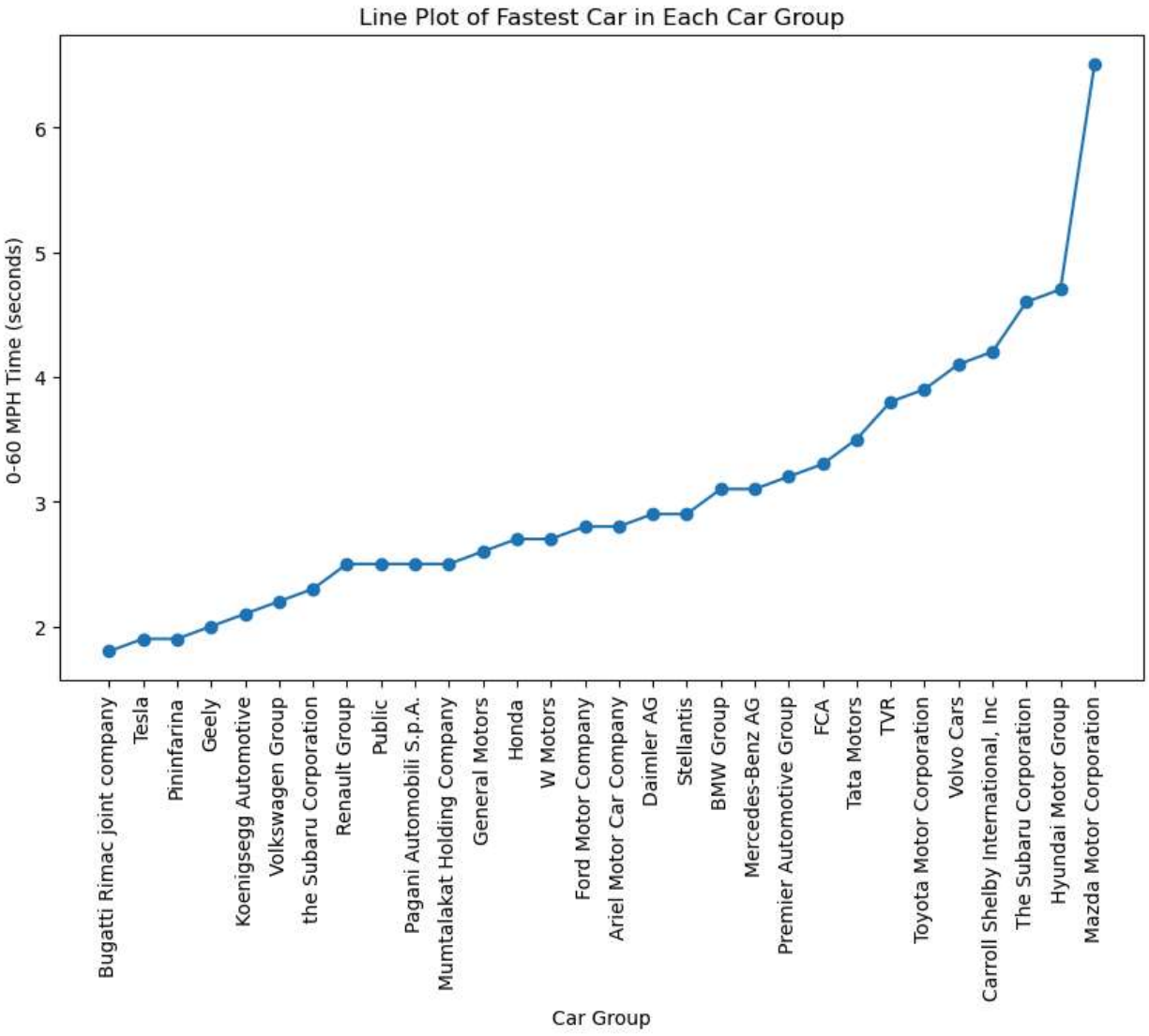
In [35]:
```
fastest_cars = Scag.loc[Scag.groupby('Car Group')['0-60 MPH Time (seconds)'].idxmin()]
fastest_cars = fastest_cars.sort_values('0-60 MPH Time (seconds)')
fastest_cars
```

Out[35]:

	Car Make	Car Model	Year	Engine Size (L)	Horsepower	Torque (lb-ft)	0-60 MPH Time (seconds)	Price (in USD)	Country	C
657	Rimac	C_Two	2022	0.0	1888.0	1696.0	1.8	2400000	Croatia	Ri
680	Tesla	Roadster	2022	0.0	NaN	737.0	1.9	200000	America	
709	Pininfarina	Battista	2022	0.0	1874.0	1696.0	1.9	2500000	Itlay	P
572	Lotus	Evija	2021	0.0	1972.0	1254.0	2.0	2000000	England	
536	Koenigsegg	Jesko Absolut	2022	5.0	1600.0	1106.0	2.1	2800000	Sweden	Ko Au
50	Porsche	918 Spyder	2015	4.6	887.0	944.0	2.2	1800000	Germany	Vo
713	Ultima	RS	2021	6.2	1200.0	1300.0	2.3	220000	England	th Co
387	Nissan	GT-R Nismo	2021	3.8	600.0	481.0	2.5	212000	Japan	
114	Ferrari	SF90 Stradale	2021	4.0	986.0	590.0	2.5	625000	Italy	
644	Pagani	Huayra Roadster BC	2021	6.0	791.0	774.0	2.5	3500000	Itlay	Au
221	McLaren	Speedtail	2021	4.0	1035.0	848.0	2.5	2300000	England	Mu
350	Chevrolet	Corvette Z06	2023	5.5	625.0	650.0	2.6	85000	America	
665	Acura	NSX	2021	3.5	573.0	476.0	2.7	157500	Japan	
700	W Motors	Fenyr Supersport	2022	3.8	800.0	723.0	2.7	1700000	Dubai	W
370	Ford	GT	2022	3.5	660.0	550.0	2.8	500000	America	Fo
606	Ariel	Atom	2021	2.0	320.0	243.0	2.8	75000	England	Ar
311	Mercedes-Benz	SLS AMG	2021	4.0	730.0	590.0	2.9	250000	Germany	Da
590	Maserati	MC20	2021	3.0	621.0	538.0	2.9	210000	Itlay	
244	BMW	M8	2022	4.4	617.0	553.0	3.1	130000	Germany	BM
636	Mercedes-AMG	GT Black Series	2021	4.0	720.0	590.0	3.1	325000	Germany	M

	Car Make	Car Model	Year	Engine Size (L)	Horsepower	Torque (lb-ft)	0-60 MPH Time (seconds)	Price (in USD)	Country	C
423	Aston Martin	DBS Superleggera	2021	5.2	715.0	664.0	3.2	304995	England	Au
476	Dodge	Viper	2017	8.4	645.0	600.0	3.3	120000	America	
513	Jaguar	F-Type	2022	5.0	575.0	516.0	3.5	103200	England	Tat
704	TVR	Griffith	2022	5.0	500.0	479.0	3.8	123500	England	
697	Toyota	Supra	2022	3.0	382.0	368.0	3.9	43090	Japan	Co
652	Polestar	1	2021	2.0	600.0	738.0	4.1	155000	Sweden	V
703	Shelby	Cobra	1965	7.0	435.0	440.0	4.2	1000000	America	Inte
706	Subaru	WRX STI	2021	2.5	310.0	290.0	4.6	38170	Japan	Th Co
711	Kia	Stinger	2022	3.3	368.0	376.0	4.7	52200	South Korea	Mot
670	Mazda	MX-5 Miata	2021	2.0	181.0	151.0	6.5	26830	Japan	Co

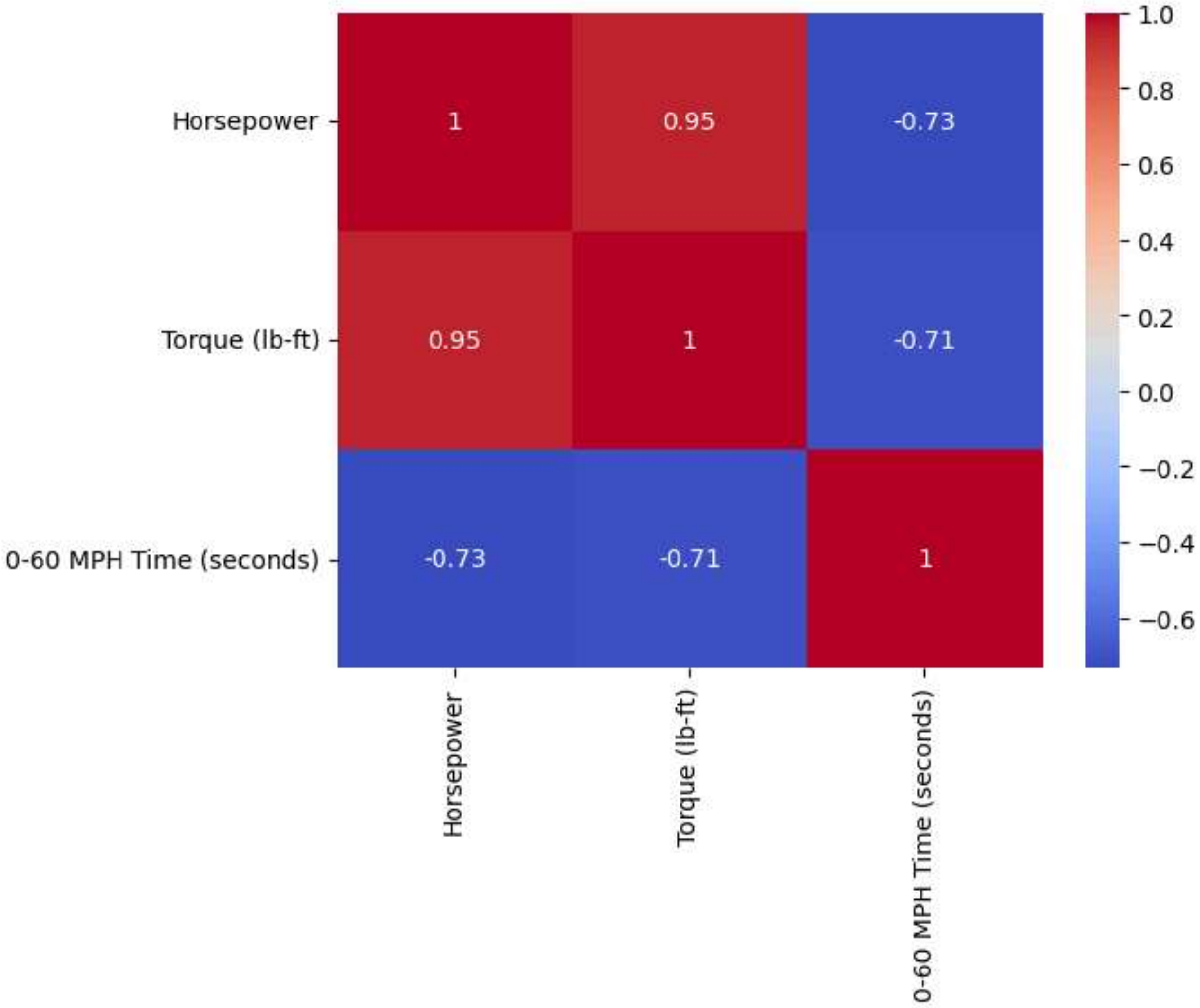
In [36]:
```
#Line plot
plt.figure(figsize=(10,6))
plt.plot(fastest_cars['Car Group'], fastest_cars['0-60 MPH Time (seconds)'], marker='c

plt.xlabel('Car Group')
plt.ylabel('0-60 MPH Time (seconds)')
plt.title('Line Plot of Fastest Car in Each Car Group')
#Rotate x-axis labels
plt.xticks(rotation=90)
#Plot
plt.show()
```



Fastest car for each Car Group based on '0-60 MPH Time (seconds)' was found above. Then it was sorted from fastest time to slowest time. Fastest 0-60 MPH Time turned out to be 1.8 seconds by the Bugatti Rimac joint company with the Rimac C_two. While the slowest fastest 0-60 MPH Time was 6.5 seconds by theMazda Motor Corporation with the Mazda MX-5 Miata. Creating a line plot to help visualize the fastest 0-60 MPH Time for each Car Group was very interesting as the gap isn't to big as you go from fastest to fastest slowest 0-60 from the next Group until you get to Hyundai Motor Group the gap between them and Mazda Motor Coperation is quite big when visualization over a 1 second difference.

Correlation Matrix

In [37]:
```
corr = Scag[["Horsepower", "Torque (lb-ft)", "0-60 MPH Time (seconds)"]].corr()

# Generate a heatmap
sns.heatmap(corr, annot=True, cmap='coolwarm')

# Show the plot
plt.show()
```



A Correlation matrix was created to show the correlation between Horsepower, Torque and 0-60MPH Time. 1 indicates a strong positive relationship, -1 indicates a strong negative relationship, and 0 indicates no relationship. So for example 0-60 MPH time and horsepower are connected to the box -0.73 which means as horsepower increases 0-60 MPH Time decreases which is expected as faster car equals faster 0-60 MPH Time. while another example Torque and Horsepower sharing a box with 0.95 that mean as horsepower increases torque increase as well. Overall these results are consistent with what we know. More horsepower equals faster 0-60 MPH time and increase in torque while more torque equals faster 0-60MPH time and increase in horsepower. So overall yes there is a correlation between 0-60 MPH Time and Horsepower and Torque.