Predicting Fuel Consumption for cars

1. Importing the libraries

In [1]:

```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

import warnings
warnings.filterwarnings('ignore')
```

2. Load the Dataset

In [2]:

```
data=pd.read_csv("auto_mpg_dataset.csv",sep=",")
```

In [3]:

```
data.head()
```

Out[3]:

	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	car_name	mpg
0	8	307.0	130.0	3504.0	12.0	70	1	chevrolet chevelle malibu	18.0
1	8	350.0	165.0	3693.0	11.5	70	1	buick skylark 320	15.0
2	8	318.0	150.0	3436.0	11.0	70	1	plymouth satellite	18.0
3	8	304.0	150.0	3433.0	12.0	70	1	amc rebel sst	16.0
4	8	302.0	140.0	3449.0	10.5	70	1	ford torino	17.0

3. Exploratory Data Analysis (EDA)

In [4]:

```
data.tail()
```

Out[4]:

	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	car_name	mpg
393	4	140.0	86.0	2790.0	15.6	82	1	ford mustang gl	27.0
394	4	97.0	52.0	2130.0	24.6	82	2	vw pickup	44.0
395	4	135.0	84.0	2295.0	11.6	82	1	dodge rampage	32.0
396	4	120.0	79.0	2625.0	18.6	82	1	ford ranger	28.0
397	4	119.0	82.0	2720.0	19.4	82	1	chevy s-10	31.0

In [5]:

```
#shape of data
data.shape
```

Out[5]:

Out[5]:

```
(398, 9)
```

In [6]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
cylinders       398 non-null int64
displacement    398 non-null float64
horsepower      398 non-null float64
weight          398 non-null float64
acceleration    398 non-null float64
model_year      398 non-null int64
origin          398 non-null int64
car_name        398 non-null object
mpg             398 non-null float64
dtypes: float64(5), int64(3), object(1)
memory usage: 28.1+ KB
```

In [7]:

```
data.isna().sum()
```

Out[7]:

```
cylinders       0
displacement    0
horsepower      0
weight          0
acceleration    0
model_year      0
origin          0
car_name        0
mpg             0
dtype: int64
```

In [8]:

```
data.describe()
```

Out[8]:

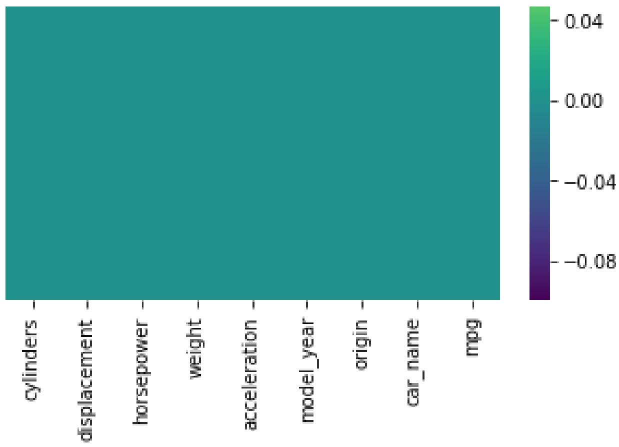
	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	mpg
count	398.000000	398.000000	398.000000	398.000000	398.000000	398.000000	398.000000	398.000000
mean	5.454774	193.425879	-1404.643216	2970.424623	15.568090	76.010050	1.572864	23.514573
std	1.701004	104.269838	12213.424763	846.841774	2.757689	3.697627	0.802055	7.815984
min	3.000000	68.000000	-100000.000000	1613.000000	8.000000	70.000000	1.000000	9.000000
25%	4.000000	104.250000	75.000000	2223.750000	13.825000	73.000000	1.000000	17.500000
50%	4.000000	148.500000	92.000000	2803.500000	15.500000	76.000000	1.000000	23.000000
75%	8.000000	262.000000	125.000000	3608.000000	17.175000	79.000000	2.000000	29.000000
max	8.000000	455.000000	230.000000	5140.000000	24.800000	82.000000	3.000000	46.600000

In [9]:

```
sns.heatmap(data.isnull(),yticklabels=False,cmap="viridis")
```

Out[9]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2a9c9cc4630>
```



We have zero null values in dataset

In [10]:

```
#columns of data
data.columns
```

Out[10]:

```
Index(['cylinders', 'displacement', 'horsepower', 'weight', 'acceleration',
       'model_year', 'origin', 'car_name', 'mpg'],
      dtype='object')
```

In [11]:

```
#checking for duplicated values
data.duplicated().sum()
```

Out[11]:

```
0
```

We have zero duplicated values

Checking for value counts in categorical columns

Categorical columns are cylinders, model_year, origin, car_name

In [12]:

```
data['cylinders'].value_counts()
```

Out[12]:

```
4    204
8    103
6     84
3      4
5      3
Name: cylinders, dtype: int64
```

In [13]:

```
data['model_year'].value_counts()
```

Out[13]:

```
73    40
78    36
76    34
82    31
75    30
81    29
80    29
79    29
70    29
```

```
77    28
72    28
71    28
74    27
Name: model_year, dtype: int64
```

In [14]:

```
data['origin'].value_counts()
```

Out[14]:

```
1    249
3     79
2     70
Name: origin, dtype: int64
```

In [15]:

```
data["car_name"].nunique()
```

Out[15]:

```
305
```

In [16]:

```
data["car_name"].value_counts().head(20)
```

Out[16]:

```
ford pinto                   6
toyota corolla               5
amc matador                  5
ford maverick                5
chevrolet chevette           4
toyota corona                4
amc gremlin                  4
chevrolet impala             4
peugeot 504                  4
amc hornet                   4
ford gran torino             3
plymouth duster              3
dodge colt                   3
volkswagen dasher            3
plymouth fury iii            3
chevrolet nova               3
chevrolet caprice classic    3
pontiac catalina             3
chevrolet citation           3
chevrolet vega               3
Name: car_name, dtype: int64
```

We found that car_name column in data is not useful for model since there are very few car names which are repeated

But we can extract some information from that column such as Brand of the car

In [17]:

```
data["car_name"].values[:10]
```

Out[17]:

```
array(['chevrolet chevelle malibu', 'buick skylark 320',
       'plymouth satellite', 'amc rebel sst', 'ford torino',
       'ford galaxie 500', 'chevrolet impala', 'plymouth fury iii',
       'pontiac catalina', 'amc ambassador dpl'], dtype=object)
```

We can see a pattern among those names

Brand name model variant

Let's extract Brand information from car_name column

In [18]:

```
data['brand'] = data["car_name"].str.extract('(^.*?)\s')
#brands=data["brand"]
#brands=data["brand"].unique().astype('str')
#brands
data["brand"].value_counts()[:10]
```

Out[18]:

```
ford          51
chevrolet     43
plymouth      31
dodge         28
amc           28
toyota        25
datsun        23
buick         17
pontiac       16
volkswagen    15
Name: brand, dtype: int64
```

There are few brand names which are repeated but in different letters

for example: chevrolet is repeated as chevy,chevroelt etc.

So, Let's replace those brand names as basic brand name

In [19]:

```
data['brand'] = data['brand'].replace(['volkswagen','vokswagen','vw'],'VW')
data['brand'] = data['brand'].replace(['chevrolet','chevy','chevroelt'],'chevrolet')
data['brand'] = data['brand'].replace('maxda','mazda')
data['brand'] = data['brand'].replace('toyouta','toyota')
data['brand'] = data['brand'].replace('mercedes','mercedes-benz')
data['brand'] = data['brand'].replace('nissan','datsun')
data['brand'] = data['brand'].replace('capri','ford')
data['brand'] = data['brand'].replace('nissan','datsun')
```

In [20]:

```
#Checking for any null values in brand
data[data['brand'].isnull()]
```

Out[20]:

	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	car_name	mpg	brand
150	4	108.0	93.0	2391.0	15.5	74	3	subaru	26.0	NaN
346	4	97.0	67.0	2065.0	17.8	81	3	subaru	32.3	NaN

We found that there are two null values

We can fill those values with their car name "subaru"

In [21]:

```
data['brand'].fillna(value = 'subaru',inplace=True)
```

In [22]:

```
data["brand"]=data["brand"].str.capitalize()
```

In [23]:

```
data.head()
```

Out[23]:

	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	car_name	mpg	brand
0	8	307.0	130.0	3504.0	12.0	70	1	chevrolet chevelle malibu	18.0	Chevrolet
1	8	350.0	165.0	3693.0	11.5	70	1	buick skylark 320	15.0	Buick
2	8	318.0	150.0	3436.0	11.0	70	1	plymouth satellite	18.0	Plymouth
3	8	304.0	150.0	3433.0	12.0	70	1	amc rebel sst	16.0	Amc
4	8	302.0	140.0	3449.0	10.5	70	1	ford torino	17.0	Ford

In [24]:

```
def country(x):
    if x==1:
        return "USA"
    elif x==2:
        return "Europe"
    elif x==3:
        return "Japan"
```

In [25]:

```
data["origin"]=data["origin"].apply(country)
```

In [26]:

```
data.head()
```

Out[26]:

	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	car_name	mpg	brand
0	8	307.0	130.0	3504.0	12.0	70	USA	chevrolet chevelle malibu	18.0	Chevrolet
1	8	350.0	165.0	3693.0	11.5	70	USA	buick skylark 320	15.0	Buick
2	8	318.0	150.0	3436.0	11.0	70	USA	plymouth satellite	18.0	Plymouth
3	8	304.0	150.0	3433.0	12.0	70	USA	amc rebel sst	16.0	Amc
4	8	302.0	140.0	3449.0	10.5	70	USA	ford torino	17.0	Ford

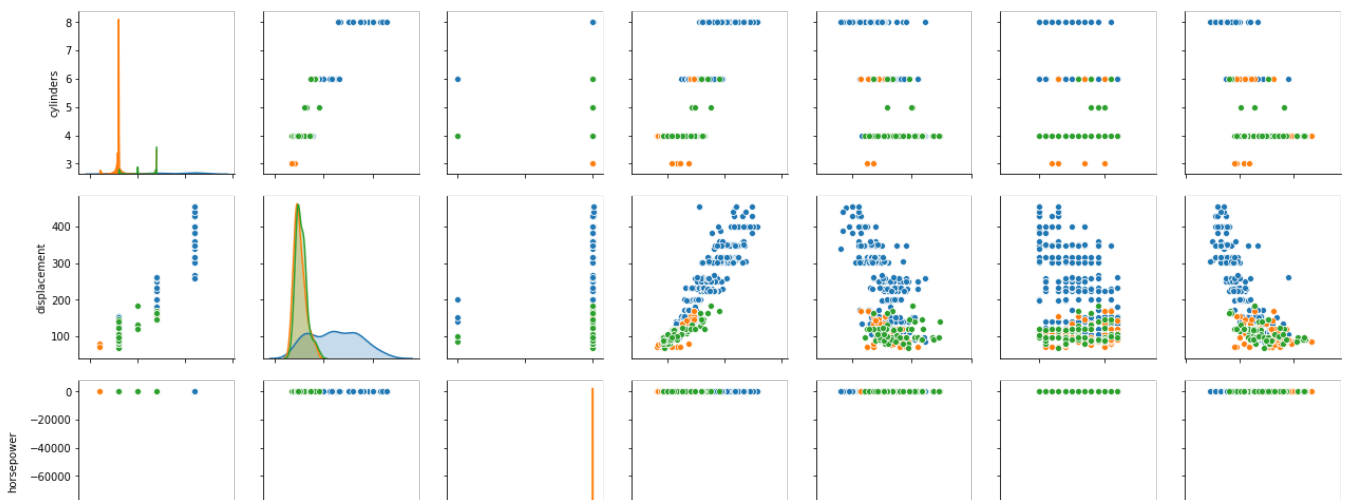
Ploting Pair Plots in order to visualize any outliers in dataset

In [27]:

```
sns.pairplot(data,hue="origin")
```

Out[27]:

<seaborn.axisgrid.PairGrid at 0x2a9c9d643c8>



There is some issue with "horsepower" column

Few data points are plotted at -10000

These are the outliers in our data

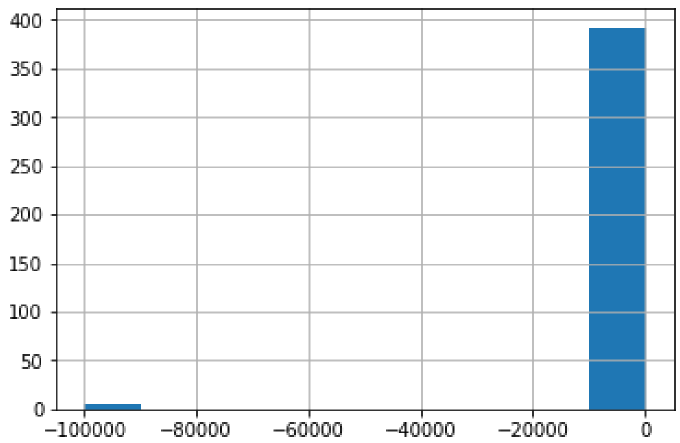
Let's visualize that column little more

In [28]:

```
#Distribution of horsepower for all cars
data["horsepower"].hist()
```

Out[28]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2a9c9743278>
```



In [29]:

```
#Range of horsepower
print("minimum value: {} \nmaximum value : {}".format(data["horsepower"].min(),data.hors
epower.max()))
```

```
minimum value: -100000.0
maximum value : 230.0
```

There are few data points which have horsepower as -10000

Let's get those data points

In [30]:

```
data[data["horsepower"]<=0]
```

Out[30]:

	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	car_name	mpg	brand
32	4	98.0	-100000.0	2046.0	19.0	71	USA	ford pinto	25.0	Ford
126	6	200.0	-100000.0	2875.0	17.0	74	USA	ford maverick	21.0	Ford
330	4	85.0	-100000.0	1835.0	17.3	80	Europe	renault lecar deluxe	40.9	Renault
336	4	140.0	-100000.0	2905.0	14.3	80	USA	ford mustang cobra	23.6	Ford
354	4	100.0	-100000.0	2320.0	15.8	81	Europe	renault 18i	34.5	Renault
374	4	151.0	-100000.0	3035.0	20.5	82	USA	amc concord dl	23.0	Amc

In [31]:

```
data[data["car_name"]=="ford pinto"]
```

Out[31]:

	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	car_name	mpg	brand
32	4	98.0	-100000.0	2046.0	19.0	71	USA	ford pinto	25.0	Ford
112	4	122.0	85.0	2310.0	18.5	73	USA	ford pinto	19.0	Ford
130	4	122.0	80.0	2451.0	16.5	74	USA	ford pinto	26.0	Ford
168	4	140.0	83.0	2639.0	17.0	75	USA	ford pinto	23.0	Ford
174	6	171.0	97.0	2984.0	14.5	75	USA	ford pinto	18.0	Ford
206	4	140.0	72.0	2565.0	13.6	76	USA	ford pinto	26.5	Ford

In [32]:

```
data[data["car_name"]=="ford maverick"]
```

Out[32]:

	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	car_name	mpg	brand
17	6	200.0	85.0	2587.0	16.0	70	USA	ford maverick	21.0	Ford
100	6	250.0	88.0	3021.0	16.5	73	USA	ford maverick	18.0	Ford
126	6	200.0	-100000.0	2875.0	17.0	74	USA	ford maverick	21.0	Ford
155	6	250.0	72.0	3158.0	19.5	75	USA	ford maverick	15.0	Ford
193	6	200.0	81.0	3012.0	17.6	76	USA	ford maverick	24.0	Ford

In [33]:

```
data[data["car_name"]=="renault 18i"]
```

Out[33]:

	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	car_name	mpg	brand
354	4	100.0	-100000.0	2320.0	15.8	81	Europe	renault 18i	34.5	Renault

We noticed that car_name as renault 18i has only one data point

So, for these situations we can see their brand

In [34]:

```
data[data["brand"]=="Renault"]
```

Out[34]:

	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	car_name	mpg	brand
79	4	96.0	69.0	2189.0	18.0	72	Europe	renault 12 (sw)	26.0	Renault
186	4	101.0	83.0	2202.0	15.3	76	Europe	renault 12tl	27.0	Renault
218	4	79.0	58.0	1825.0	18.6	77	Europe	renault 5 gtl	36.0	Renault
330	4	85.0	-100000.0	1835.0	17.3	80	Europe	renault lecar deluxe	40.9	Renault
354	4	100.0	-100000.0	2320.0	15.8	81	Europe	renault 18i	34.5	Renault

We can use these data points to find a median value for outlier

In [ ]:

```

```

We can fill the median values of each car horsepower for outliers

If there are any single outlier for a particular car, Let's go with the similar Brand's median value

Array of car_names which have horsepower < 0

In [35]:

```
cars=data[data["horsepower"]<=0]["car_name"].unique()
cars
```

Out[35]:

```
array(['ford pinto', 'ford maverick', 'renault lecar deluxe',
       'ford mustang cobra', 'renault 18i', 'amc concord dl'],
      dtype=object)
```

In [36]:

```
for car in cars:
    med=0
    brand=data[data["car_name"]==cars[3]]["brand"].values
    med=data.loc[(data["car_name"]==car) & (data["horsepower"]>0),"horsepower"].median()

    data.loc[(data["car_name"]==car) & (data["horsepower"]<=0),"horsepower"]=np.nan
    data.fillna(med,inplace=True)
    med_brand=data.loc[(data["brand"]==brand[0]) & (data["horsepower"]>0),"horsepower"].
median()
    data.fillna(med_brand,inplace=True)
```

In [ ]:

```

```

In [37]:

```
data[data["car_name"]=="ford maverick"]
```

Out[37]:

	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	car_name	mpg	brand
17	6	200.0	85.0	2587.0	16.0	70	USA	ford maverick	21.0	Ford
100	6	250.0	88.0	3021.0	16.5	73	USA	ford maverick	18.0	Ford
126	6	200.0	83.0	2875.0	17.0	74	USA	ford maverick	21.0	Ford

	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	car_name	mpg	brand
155	6	250.0	72.0	3158.0	19.5	75	USA	ford maverick	15.0	Ford
193	6	200.0	81.0	3012.0	17.6	76	USA	ford maverick	24.0	Ford

In [38]:

```
data[data["horsepower"]<=0]
#All outliers have been either filled with their car_name's median value
```

Out[38]:

cylinders	displacement	horsepower	weight	acceleration	model_year	origin	car_name	mpg	brand

In [39]:

```
#or if the car_name has only single row then it has been filled with the their Brand's me
dian value
data[data["car_name"]=="renault 18i"]
```

Out[39]:

	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	car_name	mpg	brand
354	4	100.0	89.0	2320.0	15.8	81	Europe	renault 18i	34.5	Renault

Cars with brand name as "renault"

In [40]:

```
data[data["brand"]=="Renault"]
```

Out[40]:

	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	car_name	mpg	brand
79	4	96.0	69.0	2189.0	18.0	72	Europe	renault 12 (sw)	26.0	Renault
186	4	101.0	83.0	2202.0	15.3	76	Europe	renault 12tl	27.0	Renault
218	4	79.0	58.0	1825.0	18.6	77	Europe	renault 5 gtl	36.0	Renault
330	4	85.0	89.0	1835.0	17.3	80	Europe	renault lecar deluxe	40.9	Renault
354	4	100.0	89.0	2320.0	15.8	81	Europe	renault 18i	34.5	Renault

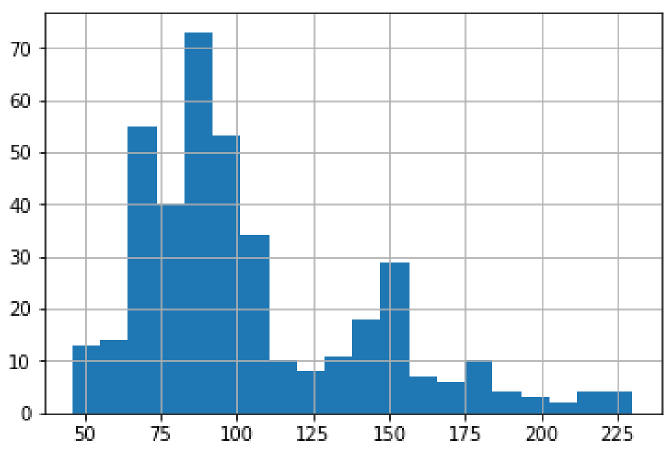
Visualizing the distribution of horsepower

In [41]:

```
data["horsepower"].hist(bins=20)
```

Out[41]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2a9ce718e10>
```



Number of brands from each origin

In [42]:

```
plt.figure(figsize=(20,8))
brands_USA=data[data["origin"]=="USA"]["brand"]
brands_Europe=data[data["origin"]=="Europe"]["brand"]
brands_Japan=data[data["origin"]=="Japan"]["brand"]
brand_origin=pd.DataFrame([["USA",brands_USA.nunique()],["Europe",brands_Europe.nunique()
],["Japan",brands_Japan.nunique()]],
                          columns=["Origin Country","Total no. of brands"])
brand_origin
```

Out[42]:

	Origin Country	Total no. of brands
0	USA	12
1	Europe	11
2	Japan	5

```
<Figure size 1440x576 with 0 Axes>
```

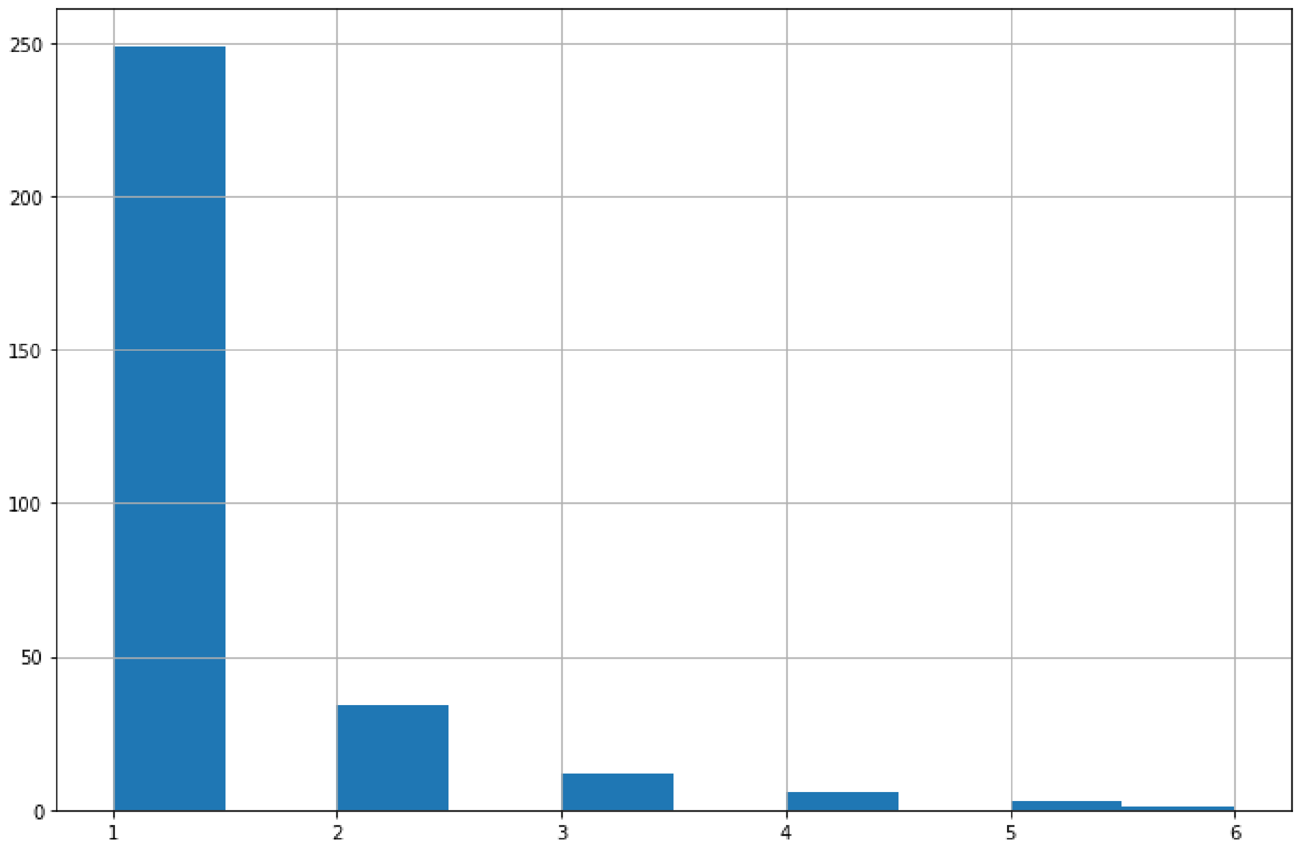
Number of unique car_names

In [43]:

```
plt.figure(figsize=(12,8))
data["car_name"].value_counts().hist()
```

Out[43]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2a9ce7ad438>
```



In [44]:

```
data["car_name"].nunique()
```

Out[44]:

```
305
```

Most of the car_names are unique. So there is no useful information in that column. Let's drop that column

In [45]:

```
data.drop("car_name",axis=1,inplace=True)
```

Correlation of data

In [46]:

```
data.corr()
```

Out[46]:

	cylinders	displacement	horsepower	weight	acceleration	model_year	mpg
cylinders	1.000000	0.950721	0.841842	0.896017	-0.505419	-0.348746	-0.775396
displacement	0.950721	1.000000	0.896334	0.932824	-0.543684	-0.370164	-0.804203
horsepower	0.841842	0.896334	1.000000	0.863053	-0.687740	-0.413533	-0.773662
weight	0.896017	0.932824	0.863053	1.000000	-0.417457	-0.306564	-0.831741
acceleration	-0.505419	-0.543684	-0.687740	-0.417457	1.000000	0.288137	0.420289
model_year	-0.348746	-0.370164	-0.413533	-0.306564	0.288137	1.000000	0.579267
mpg	-0.775396	-0.804203	-0.773662	-0.831741	0.420289	0.579267	1.000000

4. Data visualization

In [47]:

```
#Setting palette
sns.set_palette("bright")
```

In [48]:

```
data.head()
```

Out[48]:

	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	mpg	brand
0	8	307.0	130.0	3504.0	12.0	70	USA	18.0	Chevrolet
1	8	350.0	165.0	3693.0	11.5	70	USA	15.0	Buick
2	8	318.0	150.0	3436.0	11.0	70	USA	18.0	Plymouth
3	8	304.0	150.0	3433.0	12.0	70	USA	16.0	Amc
4	8	302.0	140.0	3449.0	10.5	70	USA	17.0	Ford

4.1 Plots for categorical features

1. Number of cars belong to each Origin(country)

Reference:

Origin 1: USA

Origin 2: Europe

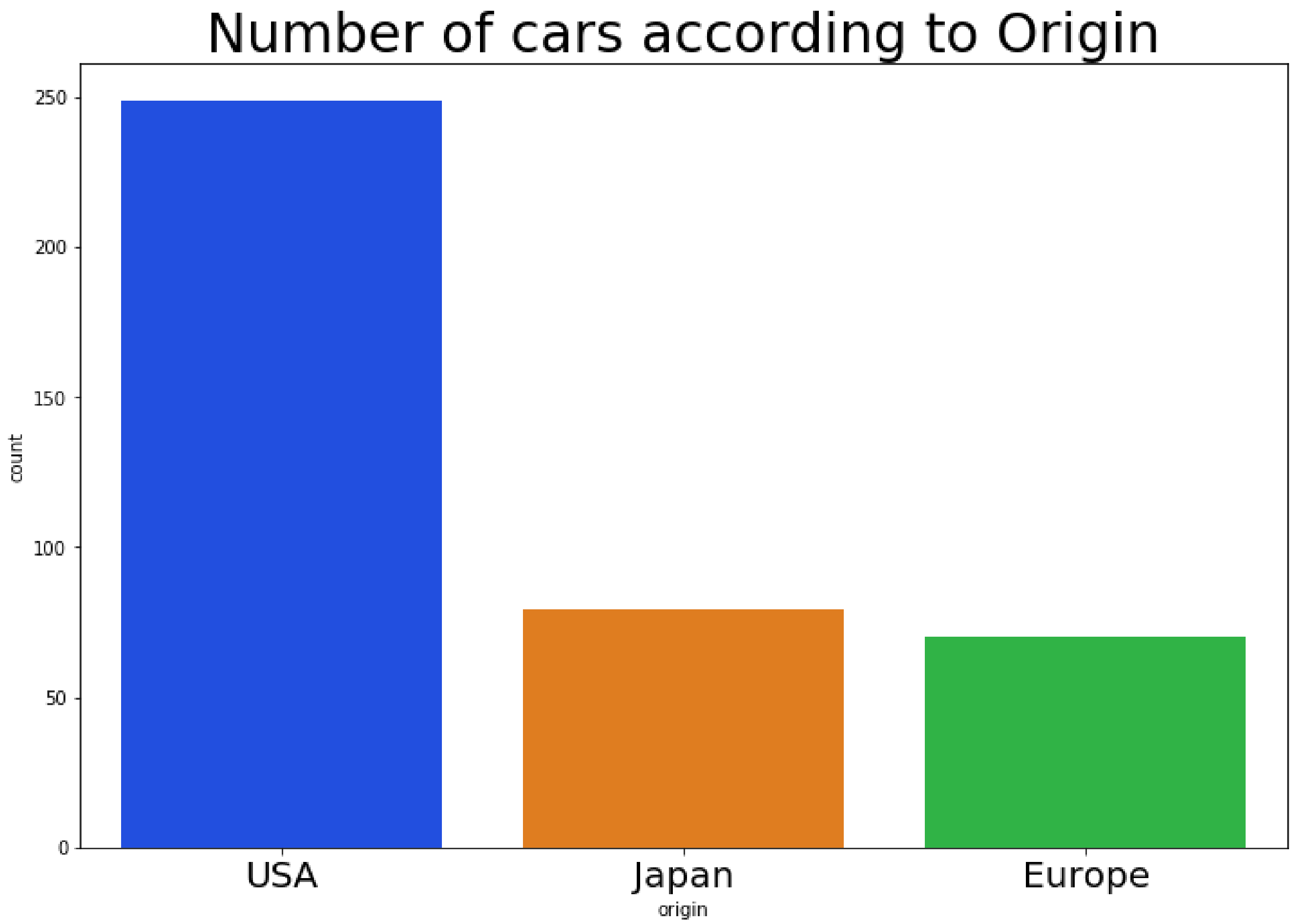
Origin 3: Japan

In [49]:

```
plt.figure(figsize=(12,8))
plt.title("Number of cars according to Origin",fontsize=30)
plt.xticks(fontsize=20)
sns.countplot(data["origin"])
```

Out[49]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2a9ceb343c8>
```



USA has the most number of cars

Japan and Europe have almost same number of cars

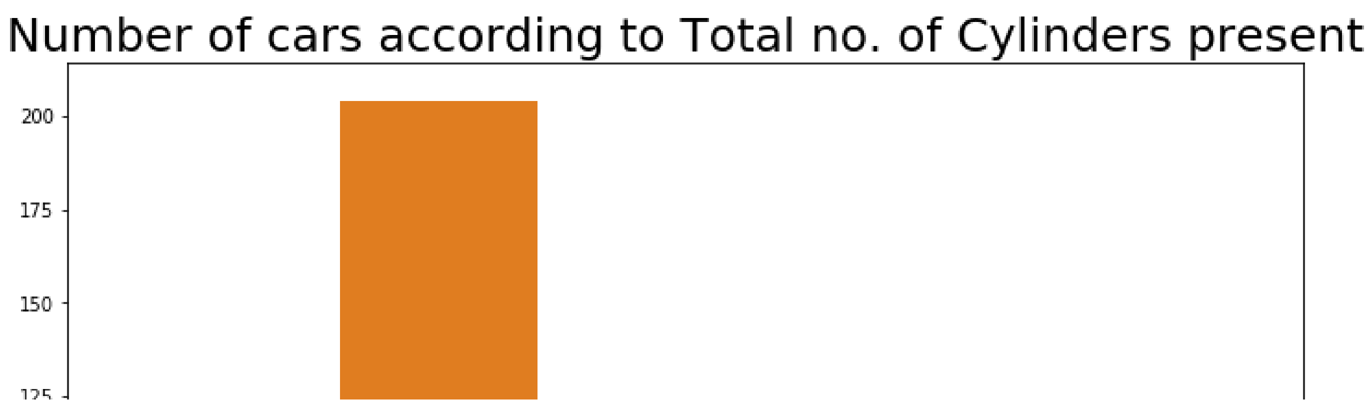
2. Number of cars belong to Total number of Cylinders present

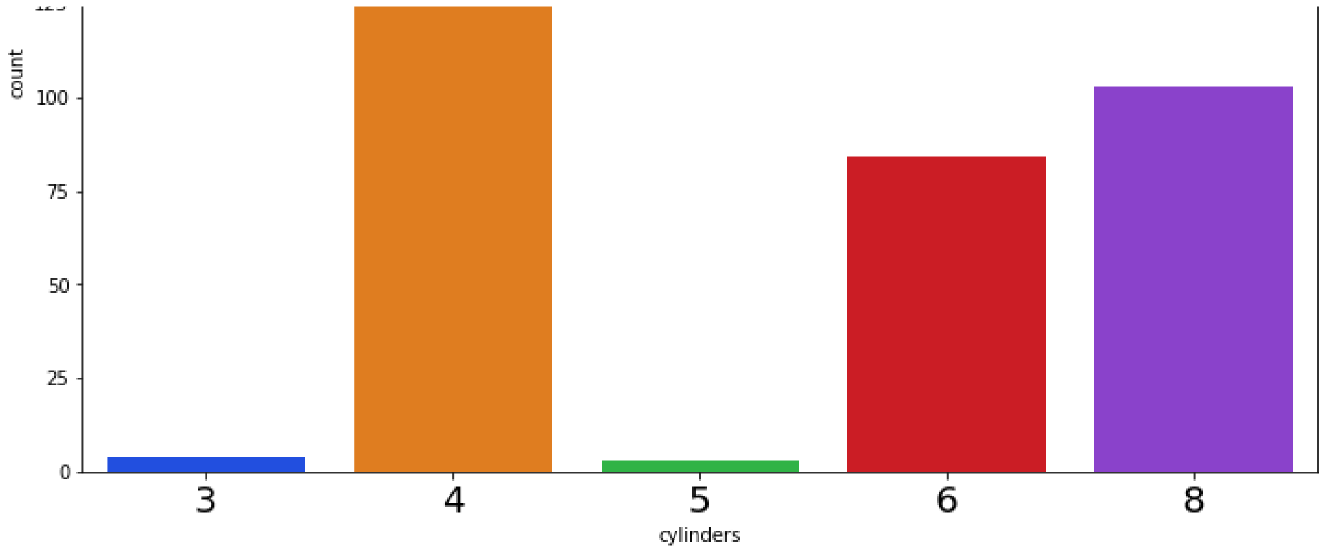
In [50]:

```
plt.figure(figsize=(12,8))
plt.title("Number of cars according to Total no. of Cylinders present",fontsize=25)
plt.xticks(fontsize=20)
sns.countplot(data["cylinders"])
```

Out[50]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2a9ce9eaa20>
```





Cars with 4 cylinders have the most number of cars

Cars with 3 and 5 cyliners have the least number of cars

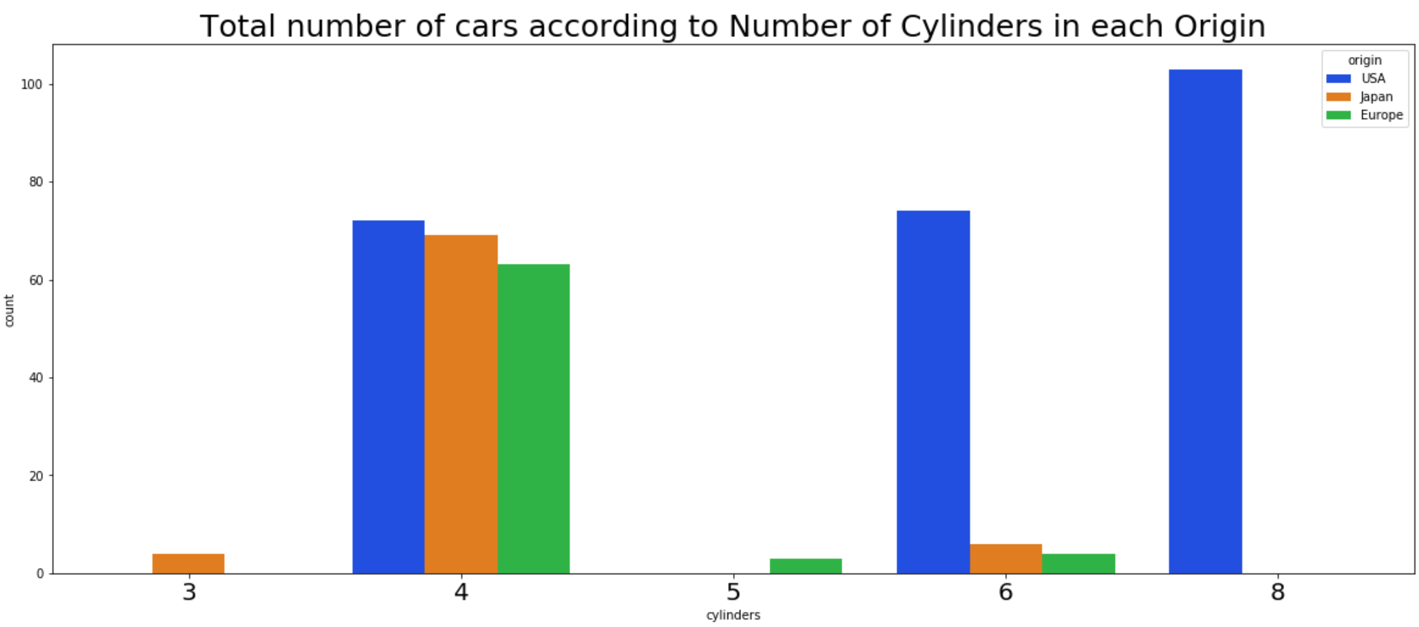
3. Number of cars belong to Total number of Cylinders present in each Origin(country)

In [51]:

```
plt.figure(figsize=(20,8))
plt.title("Total number of cars according to Number of Cylinders in each Origin",fontsize
=25)
plt.xticks(fontsize=20)
sns.countplot(x="cylinders",data=data,hue="origin")
```

Out[51]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2a9cea3dbe0>
```



Only USA cars have 8 cylinders

Similarly Ony Japan cars have 3 cylinders and Europe cars have 5 cylinders

Most common number of cylinders is 4

4. Number of cars belong to each Model year

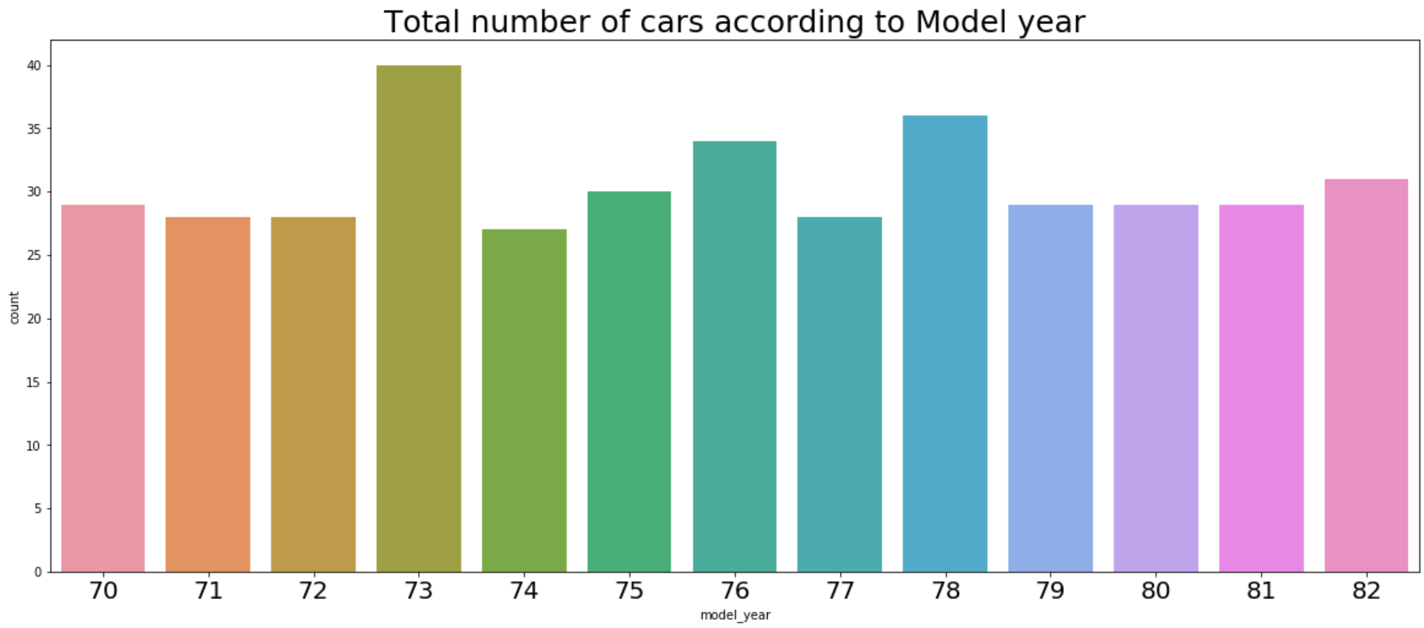
In [52]:

```
plt.figure(figsize=(20,8))
```

```
plt.figure(figsize=(20,8))
plt.title("Total number of cars according to Model year",fontsize=25)
plt.xticks(fontsize=20)

sns.countplot(data["model_year"])
```

Out[52]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2a9ceaa8cf8>
```



Cars of model_year 73 has the highest number of cars

Other model_years are almost distributed similarly

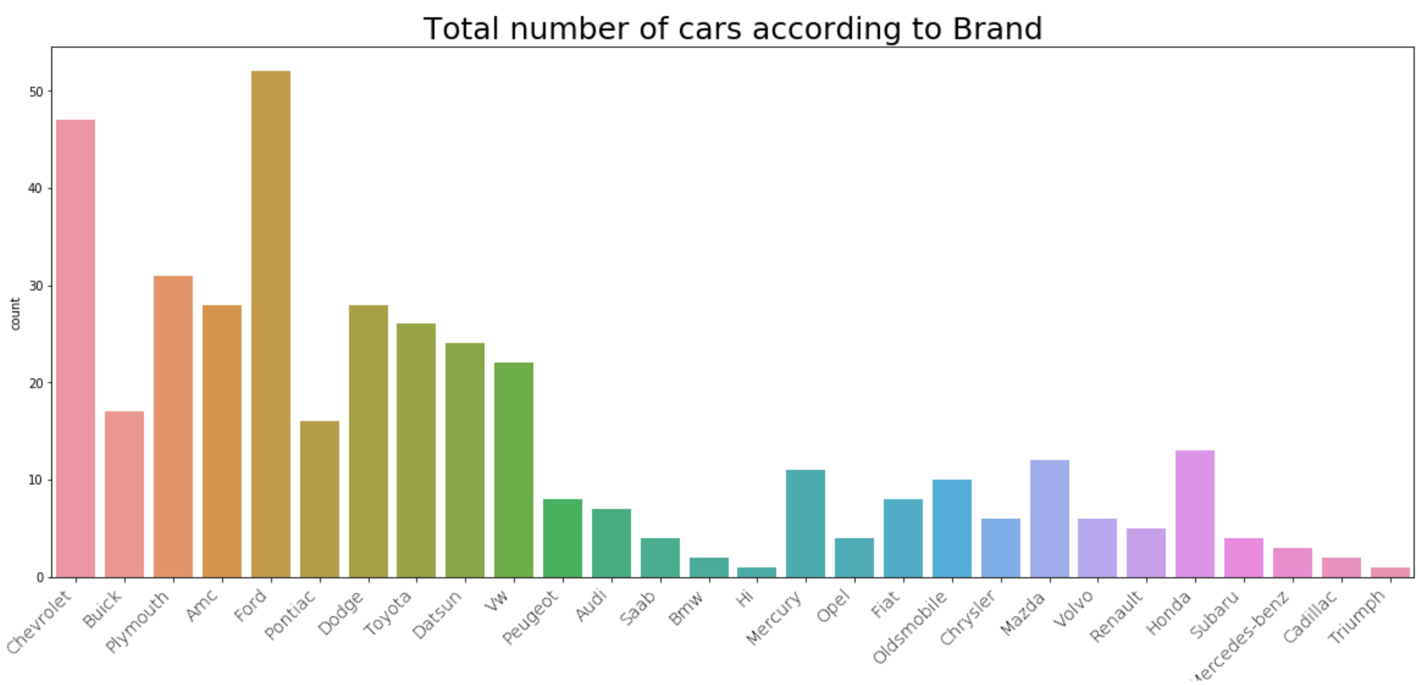
3. Number of cars belong to each Brand

In [53]:

```
plt.figure(figsize=(20,8))
plt.title("Total number of cars according to Brand",fontsize=25)
plt.xticks(rotation=45, horizontalalignment='right',fontweight='light',fontsize='x-large'
)

sns.countplot(data["brand"])
```

Out[53]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2a9cece2940>
```



brand

Ford has the most number of cars, followed by Chevrolet

Hi and Triumph have the least number of cars

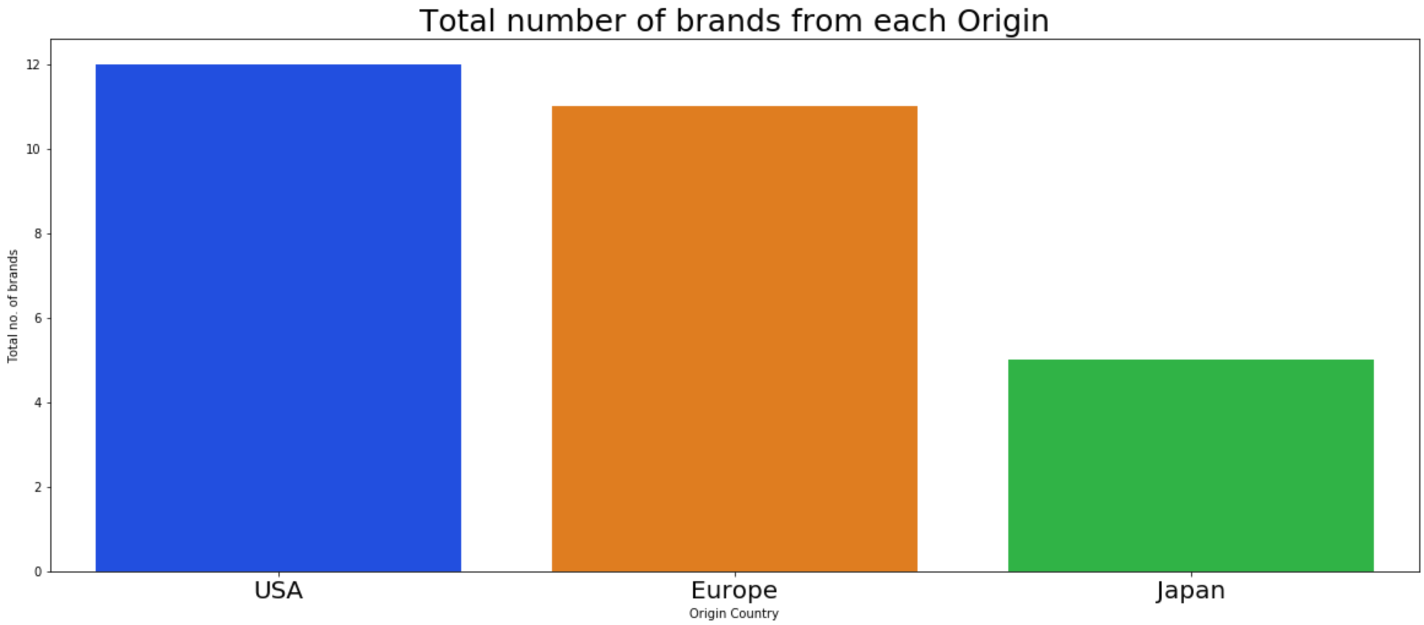
4. Number of brands from each origin

In [54]:

```
plt.figure(figsize=(20,8))
plt.title("Total number of brands from each Origin",fontsize=25)
plt.xticks(fontsize=20)

sns.barplot(data=brand_origin ,x="Origin Country",y="Total no. of brands")
```

Out[54]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2a9ceda9320>
```



USA and Europe have almost equal number of brands

Japan has ony 5 brands

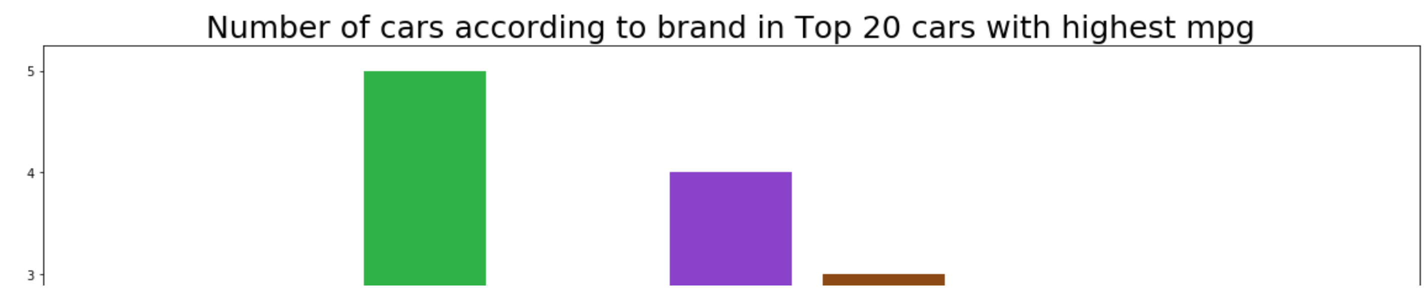
5. Number of cars according to brand in Top 20 cars with highest mpg

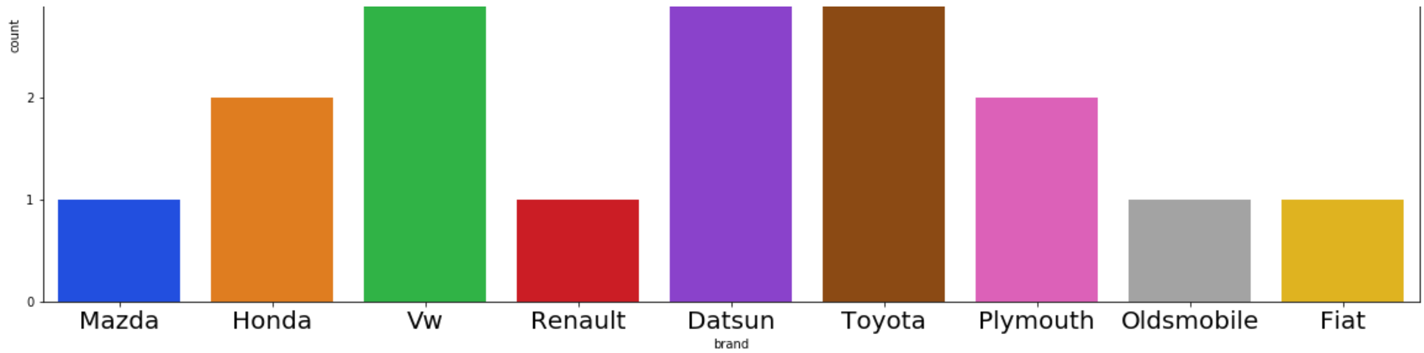
In [55]:

```
plt.figure(figsize=(20,8))
plt.title("Number of cars according to brand in Top 20 cars with highest mpg",fontsize=25
)
plt.xticks(fontsize=20)

sns.countplot(data.sort_values(by="mpg",ascending=False)["brand"][:20])
```

Out[55]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2a9cee16e48>
```





Volkswagen has 5 cars in top 20 highest mpg followed by Datsun with 4 cars

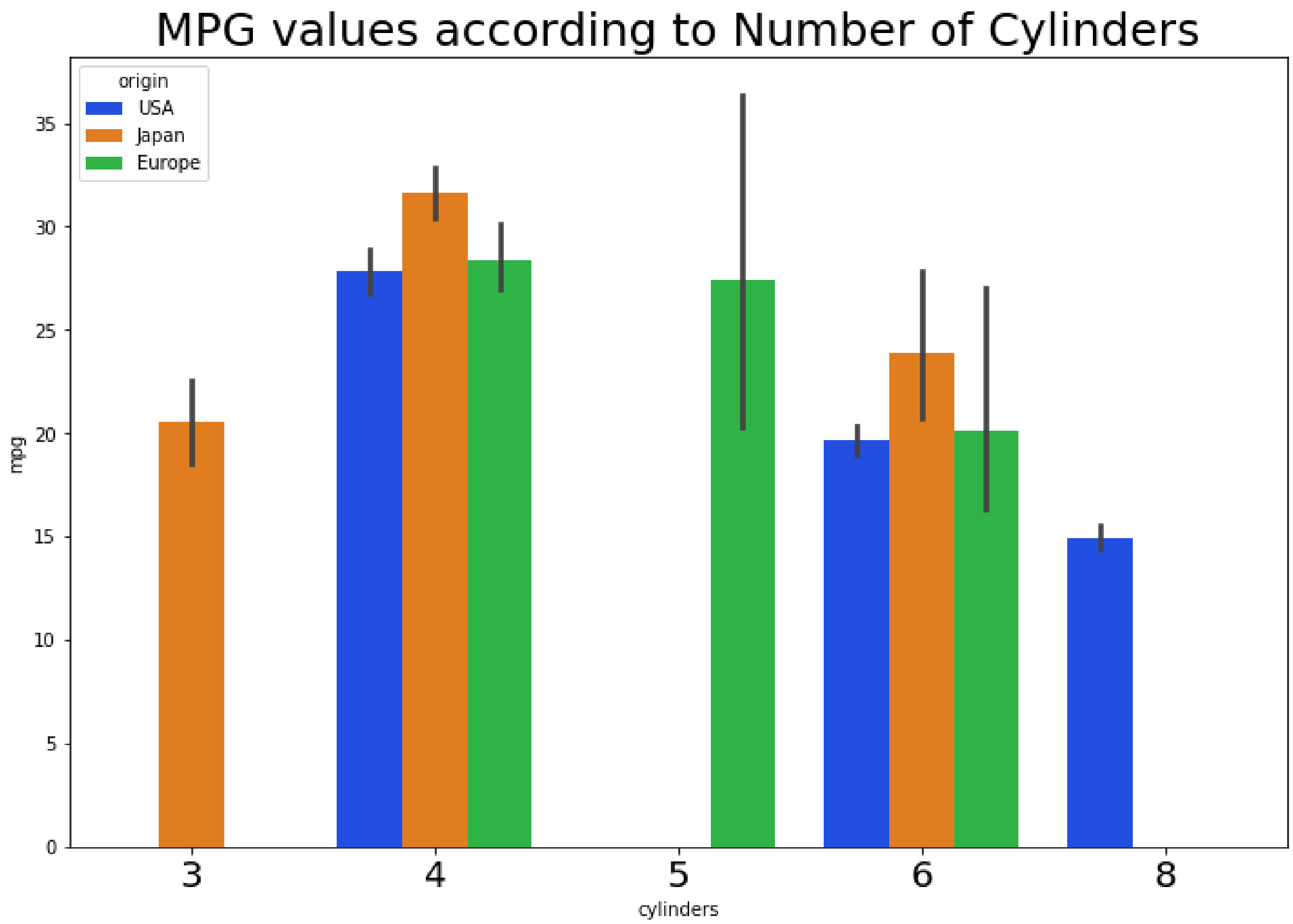
6.Average mpg values of cars in each cylinders from each origin

In [56]:

```
plt.figure(figsize=(12,8))
plt.title("MPG values according to Number of Cylinders",fontsize=25)
plt.xticks(fontsize=20)
sns.barplot(x="cylinders",y="mpg",data=data,hue="origin")
```

Out[56]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2a9cf048eb8>
```



Cars with 8 cylinders has the least Average mpg value

Overall USA has low Average mpg value

6.Average mpg values of cars in each Model year from each origin

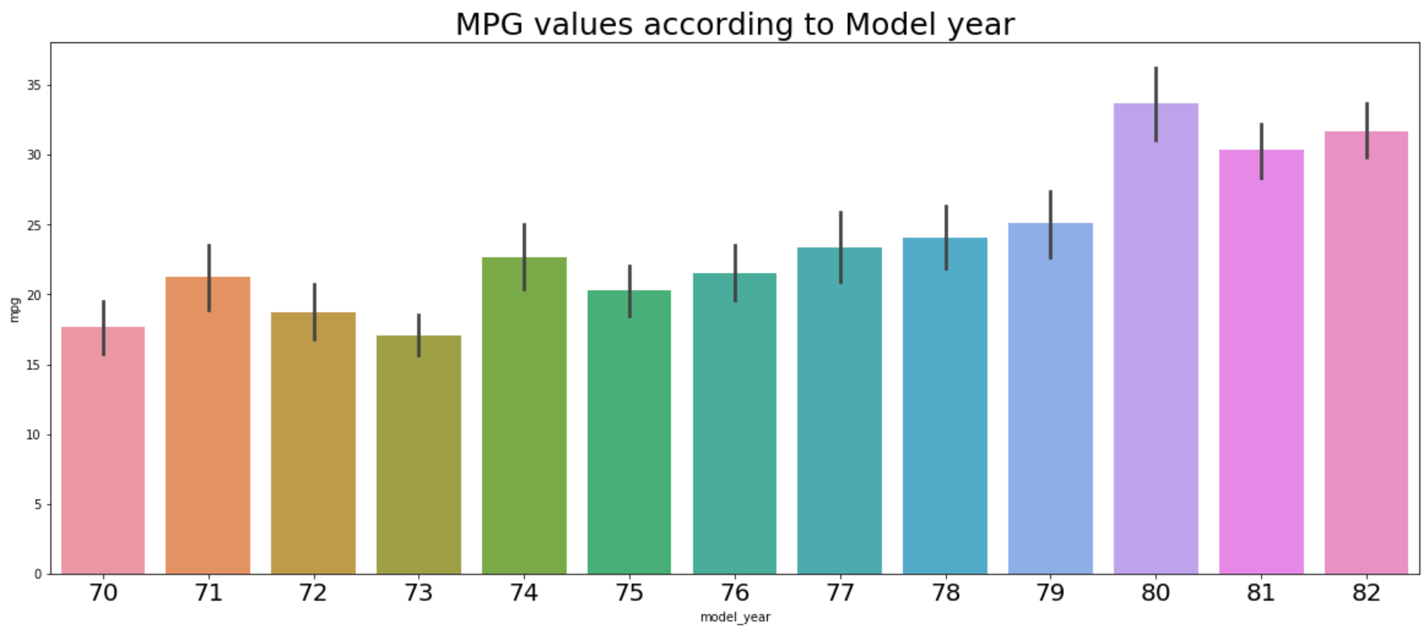
In [57]:

```
plt.figure(figsize=(20,8))
```

```
plt.title("MPG values according to Model year",fontsize=25)
plt.xticks(fontsize=20)
sns.barplot(y="mpg",x="model_year",data=data)
```

Out[57]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2a9cff35eb8>
```



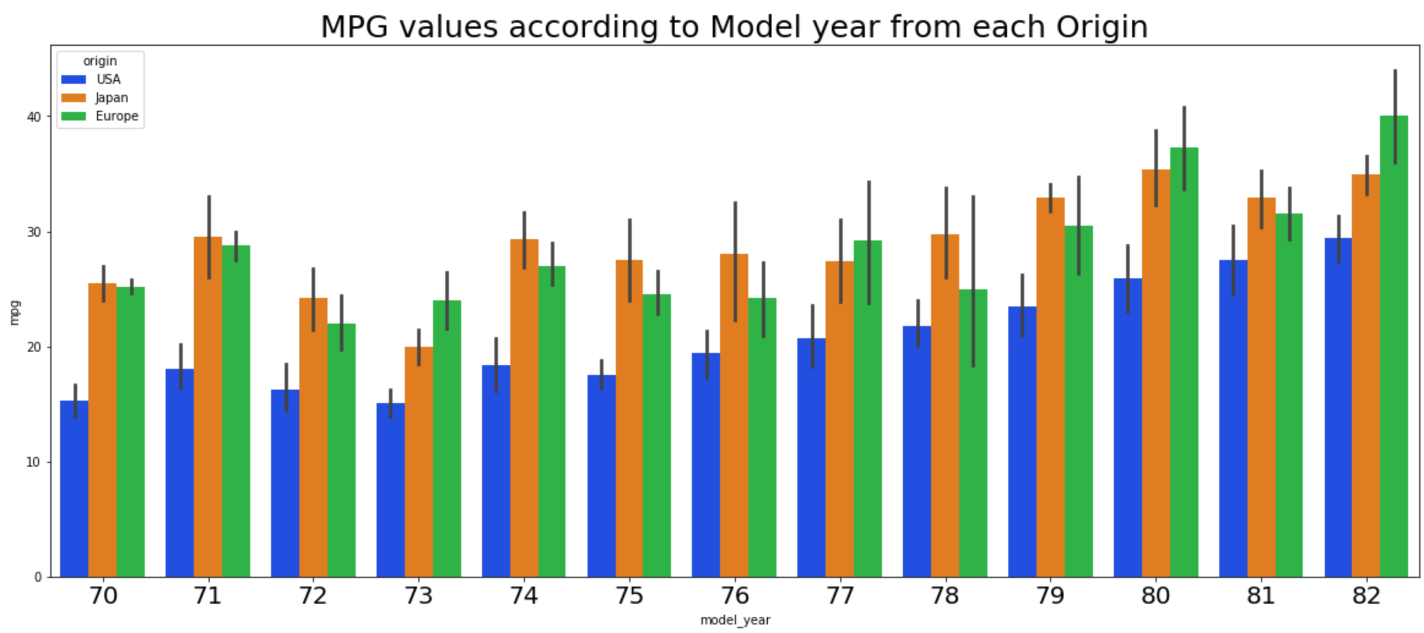
Cars of model year 80 has the highest average MPG value

In [58]:

```
plt.figure(figsize=(20,8))
plt.title("MPG values according to Model year from each Origin",fontsize=25)
plt.xticks(fontsize=20)
sns.barplot(x="model_year",y="mpg",data=data,hue="origin")
```

Out[58]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2a9cedaff28>
```



As we can see Average mpg values have been clearly improved as years passed by

Japan cars have the most number of highest averages of each year

Europe cars have improved average mpg value much better in 82

7. MPG values of each brand

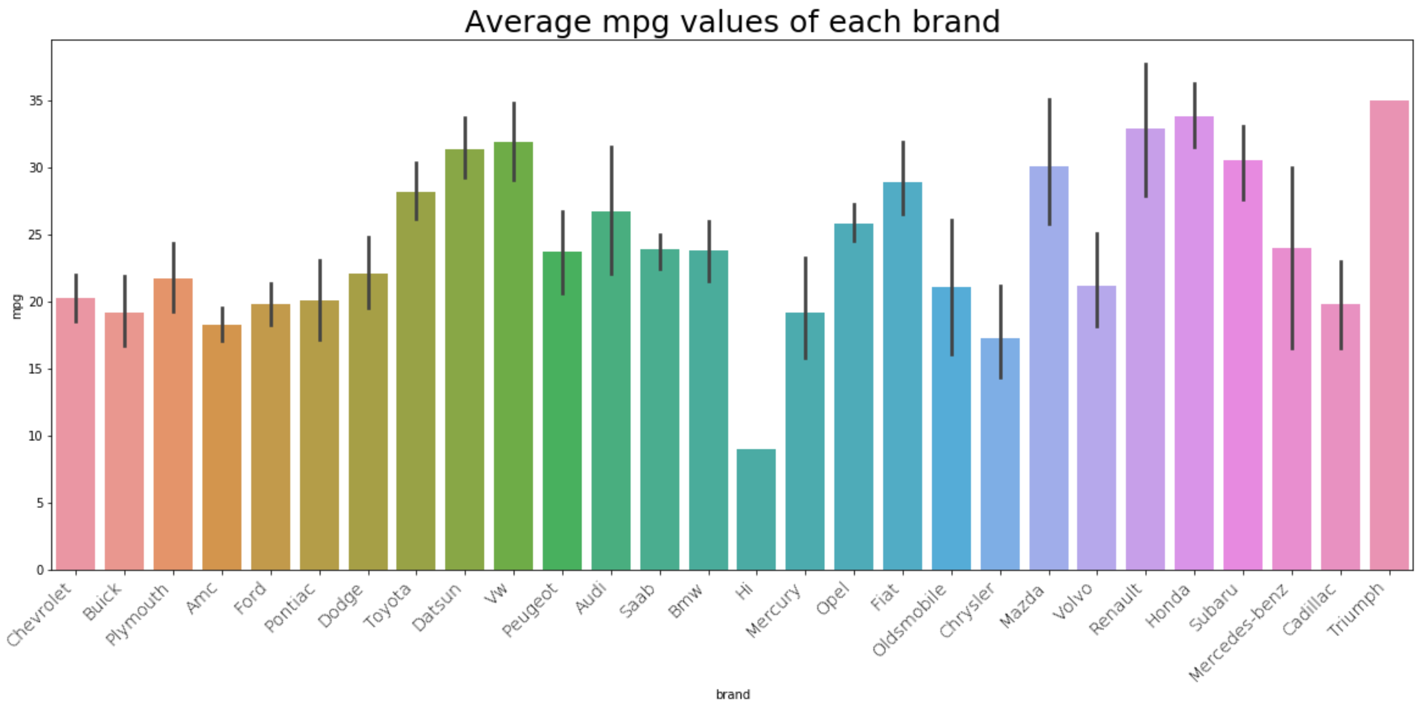
Average mpg values of each Brand

In [59]:

```
plt.figure(figsize=(20,8))
plt.title("Average mpg values of each brand",fontsize=25)
plt.xticks(rotation=45, horizontalalignment='right',fontweight='light',fontsize='x-large'
)


sns.barplot(x="brand",y="mpg",data=data)
```

Out[59]:

<matplotlib.axes._subplots.AxesSubplot at 0x2a9cfe659b0>



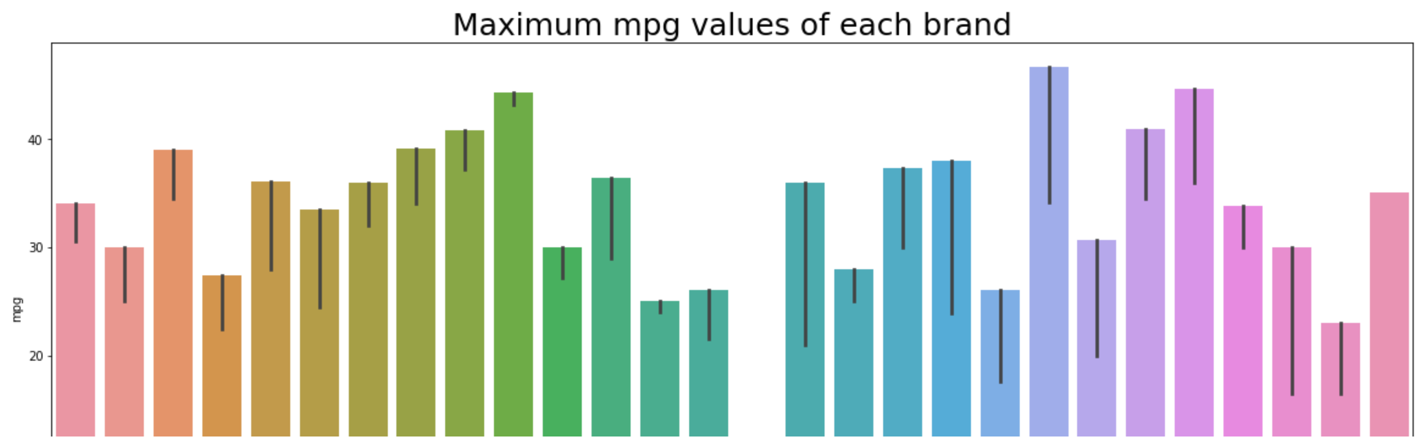
Maximum mpg value of each Brand

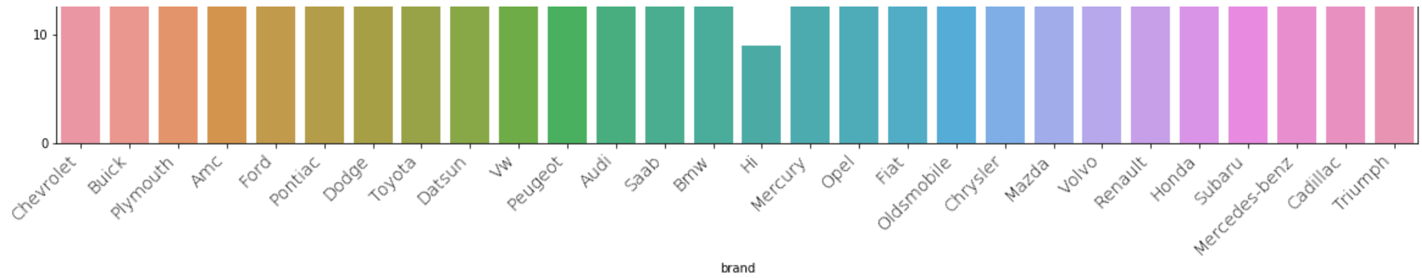
In [60]:

```
plt.figure(figsize=(20,8))
plt.title("Maximum mpg values of each brand",fontsize=25)
plt.xticks(rotation=45, horizontalalignment='right',fontweight='light',fontsize='x-large'
)


sns.barplot(x="brand",y="mpg",data=data,estimator=max)
```

Out[60]:

<matplotlib.axes._subplots.AxesSubplot at 0x2a9d00fa400>





As we can see Mazda has Highest mpg value and Hi has the lowest mgp value

Renault cars have the highest average mpg value

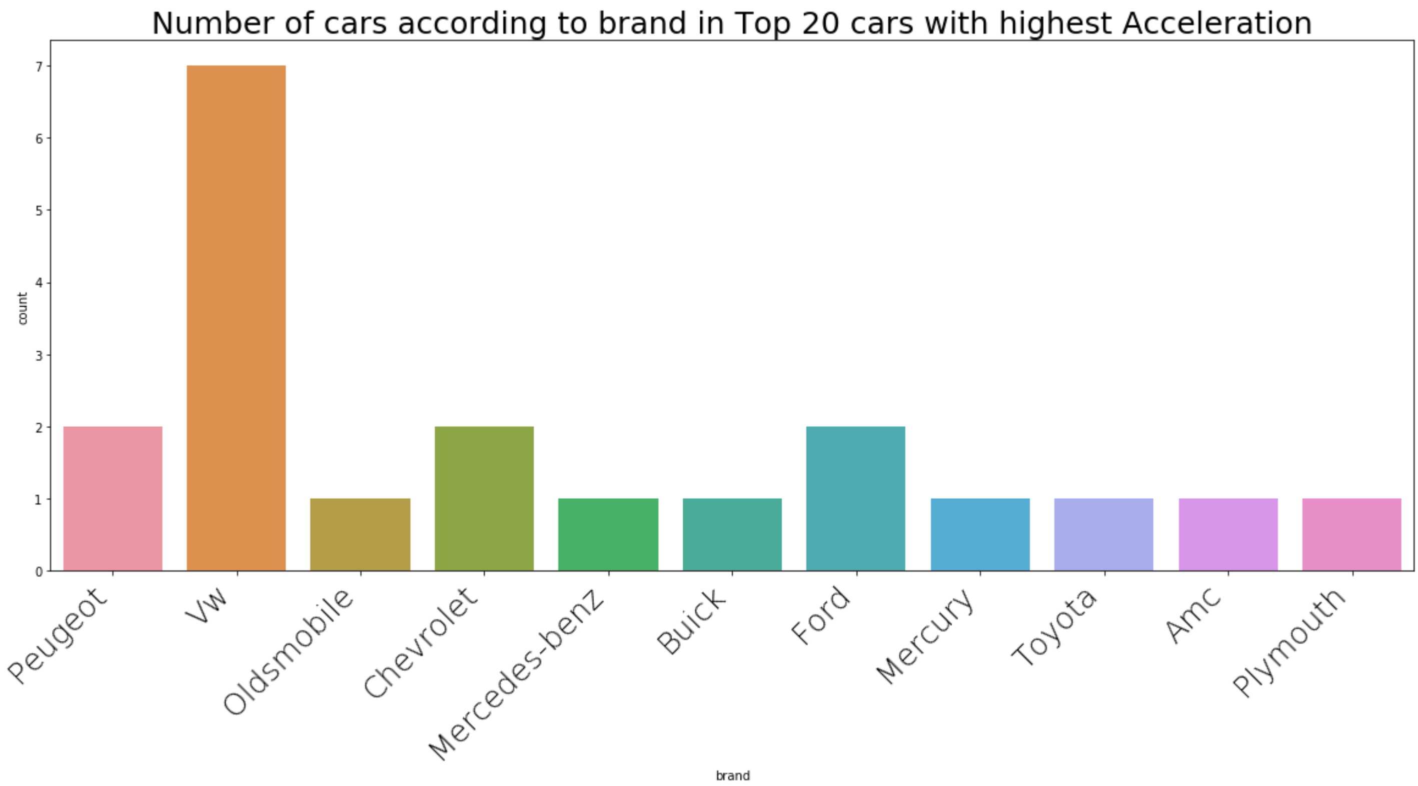
8. Number of cars according to brand in Top 20 cars with highest Acceleration

In [61]:

```
plt.figure(figsize=(20,8))
plt.title("Number of cars according to brand in Top 20 cars with highest Acceleration",fo
ntsize=25)
plt.xticks(rotation=45, horizontalalignment='right',fontweight='light',fontsize=25)

sns.countplot(data.sort_values(by="acceleration",ascending=False)["brand"][:20])
```

Out[61]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2a9d02c2da0>
```



Volkswagen has the most number of cars in Top 20 Cars with Highest Acceleration

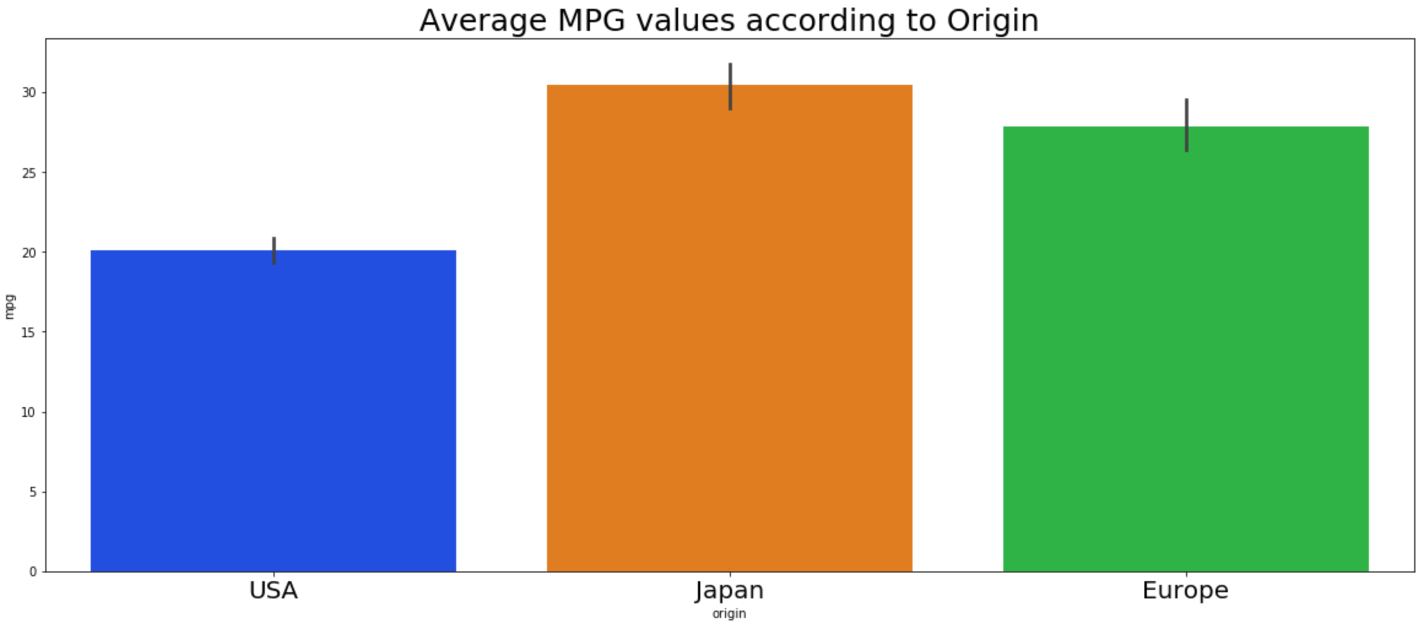
8. Average of Mpg values according to origin

In [62]:

```
plt.figure(figsize=(20,8))
plt.title("Average MPG values according to Origin",fontsize=25)
plt.xticks(fontsize=20)
sns.barplot(y="mpg",x="origin",data=data)
```

Out[62]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2a9d0342ac8>
```



Japan has highest average mpg value and USA has least

4.2 Plots for Numerical features

1. Distribution plots

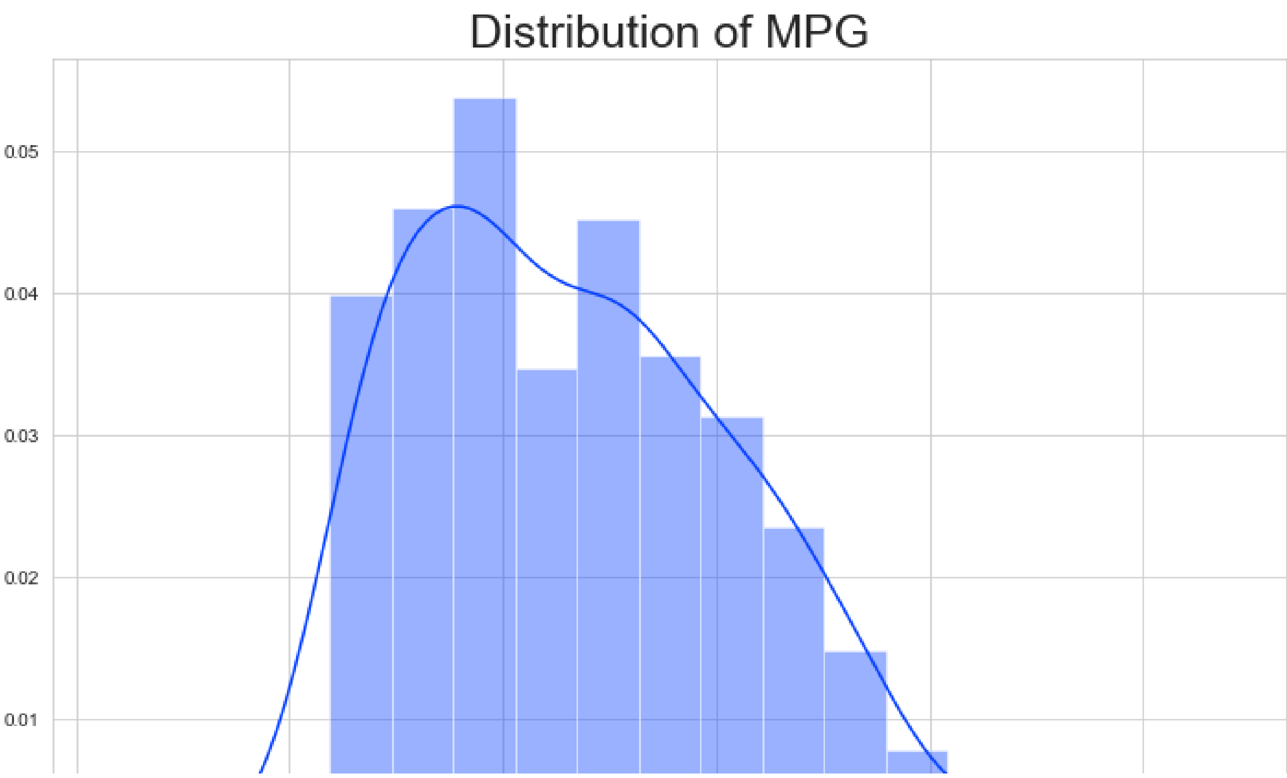
Distribution of mpg values

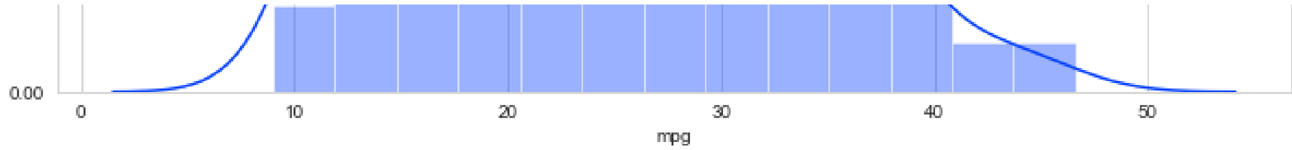
In [63]:

```
sns.set_style("whitegrid")

plt.figure(figsize=(12,8))
plt.title("Distribution of MPG",fontsize=25)

sns.distplot(data["mpg"])
```

Out[63]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2a9d038b630>
```





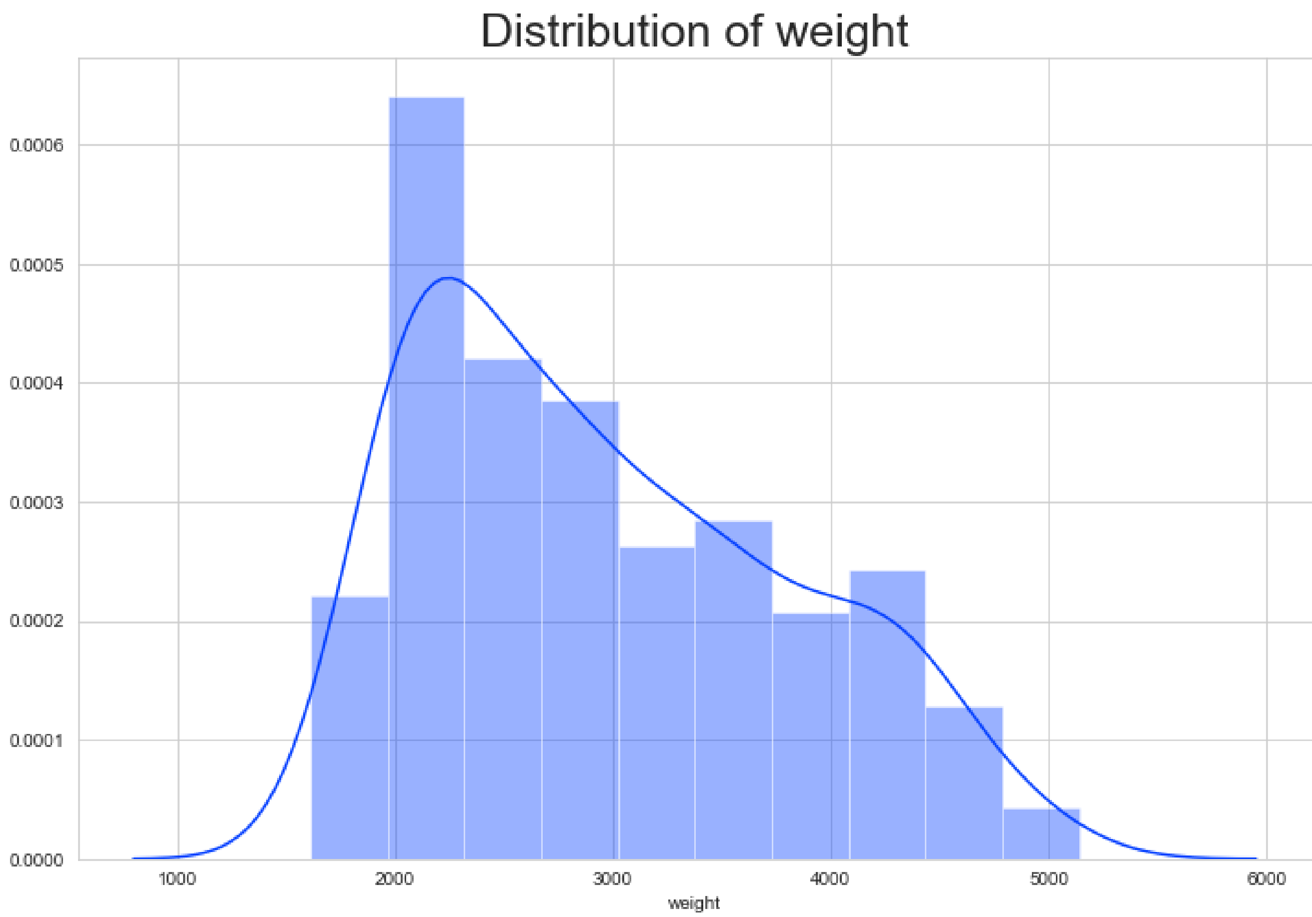
Distribution of weight of cars

In [64]:

```
plt.figure(figsize=(12,8))
plt.title("Distribution of weight",fontsize=25)

sns.distplot(data["weight"])
```

Out[64]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2a9d0408208>
```



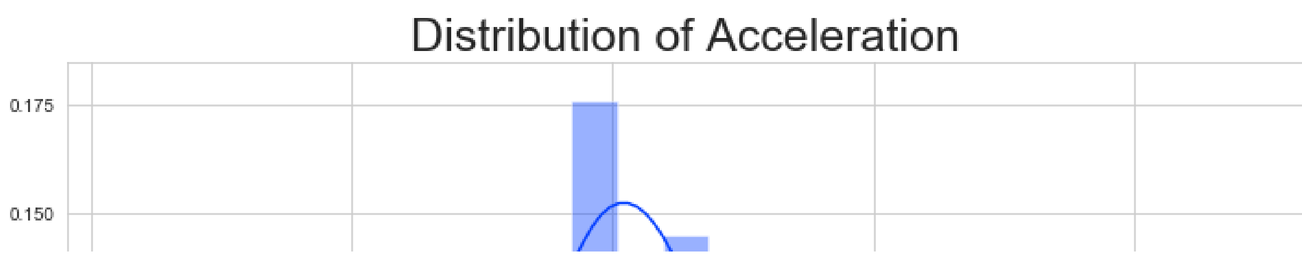
Distribution of Acceleration of cars

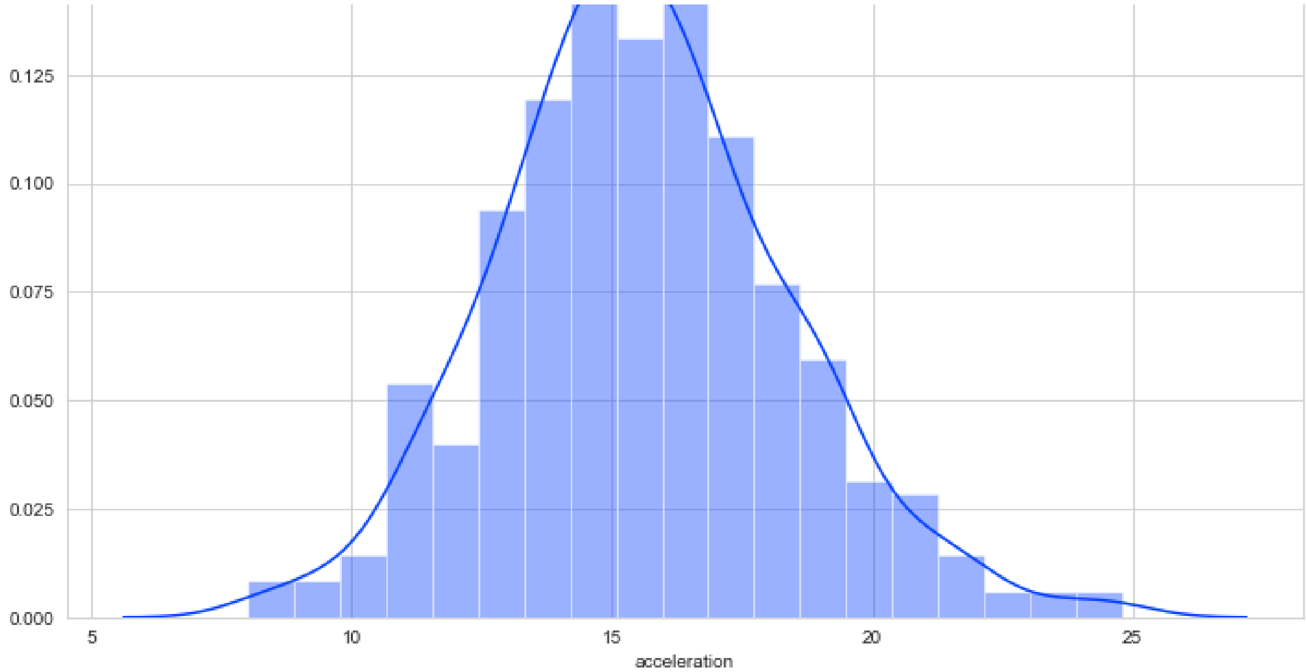
In [65]:

```
plt.figure(figsize=(12,8))
plt.title("Distribution of Acceleration",fontsize=25)

sns.distplot(data["acceleration"])
```

Out[65]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2a9d0b37ba8>
```





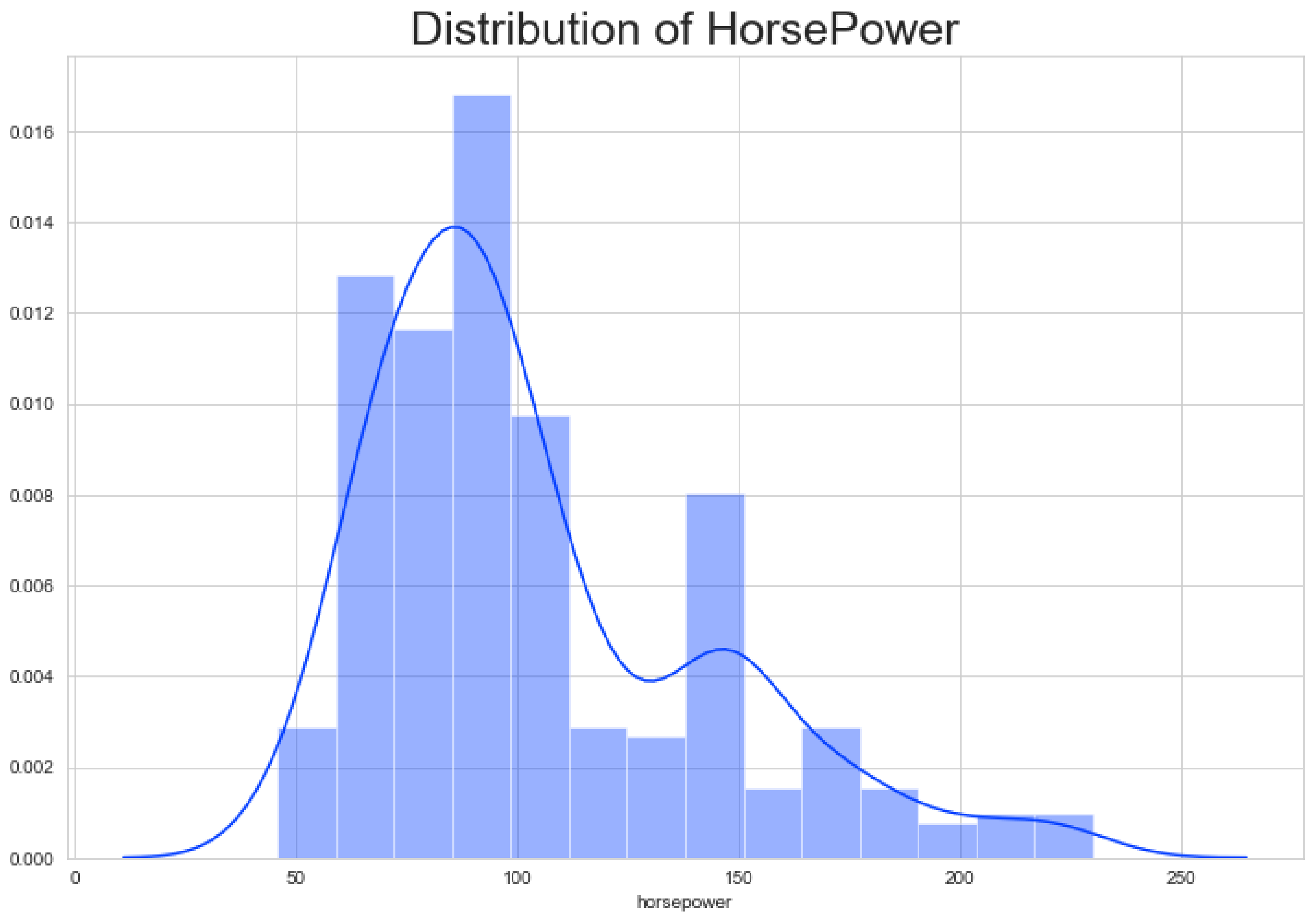
Distribution of Horsepower of cars

In [66]:

```
plt.figure(figsize=(12,8))
plt.title("Distribution of HorsePower",fontsize=25)

sns.distplot(data["horsepower"])
```

Out[66]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2a9d0b37a20>
```



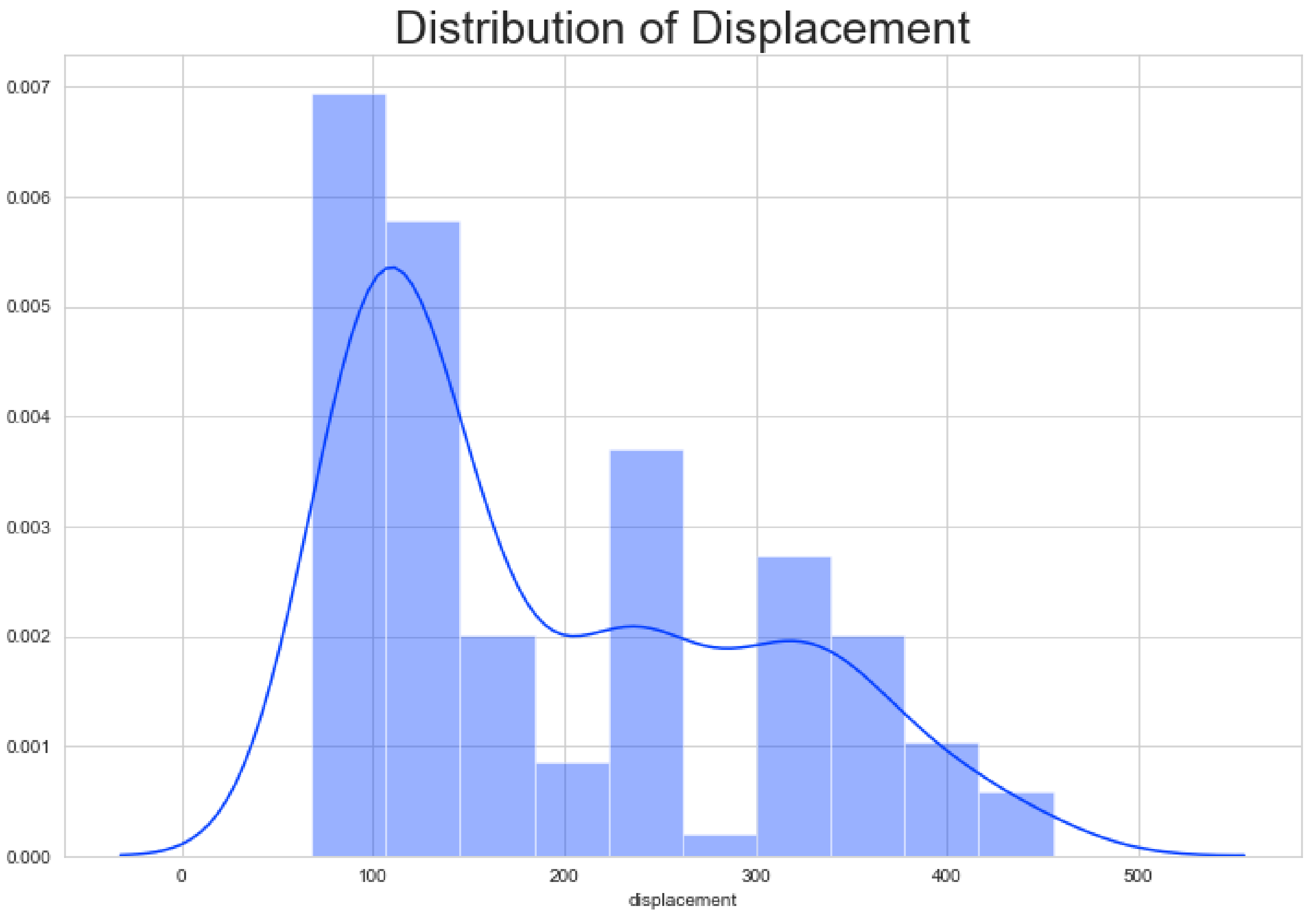
Distribution of Displacement of cars

In [67]:

```
plt.figure(figsize=(12,8))
plt.title("Distribution of Displacement",fontsize=25)

sns.distplot(data["displacement"])
```

Out[67]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2a9d0bc1e10>
```



2. Joint plots

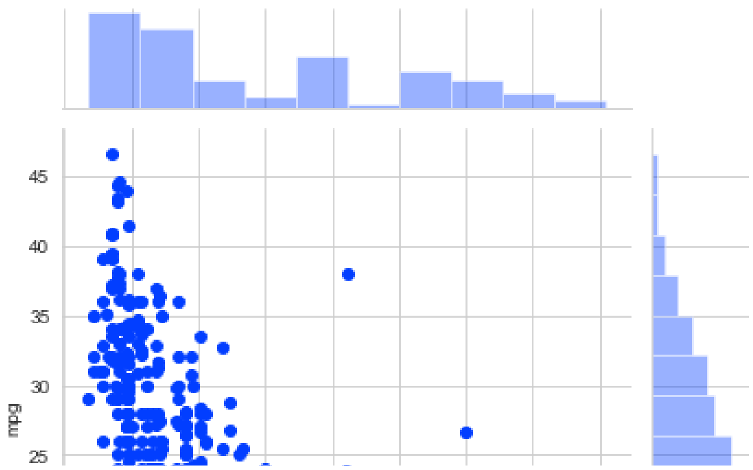
Mpg vs Displacement

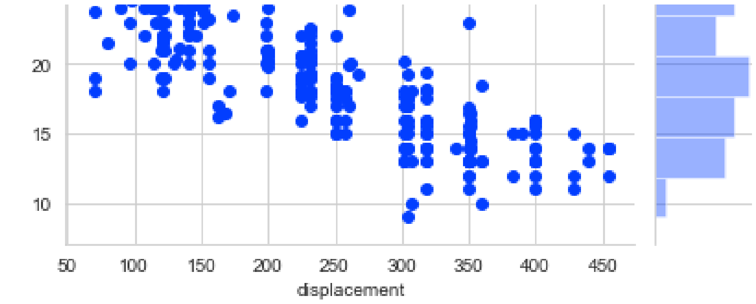
In [68]:

```
plt.figure(figsize=(12,8))
sns.jointplot("displacement","mpg",data)
```

Out[68]:

```
<seaborn.axisgrid.JointGrid at 0x2a9d1d3b2b0>

<Figure size 864x576 with 0 Axes>
```





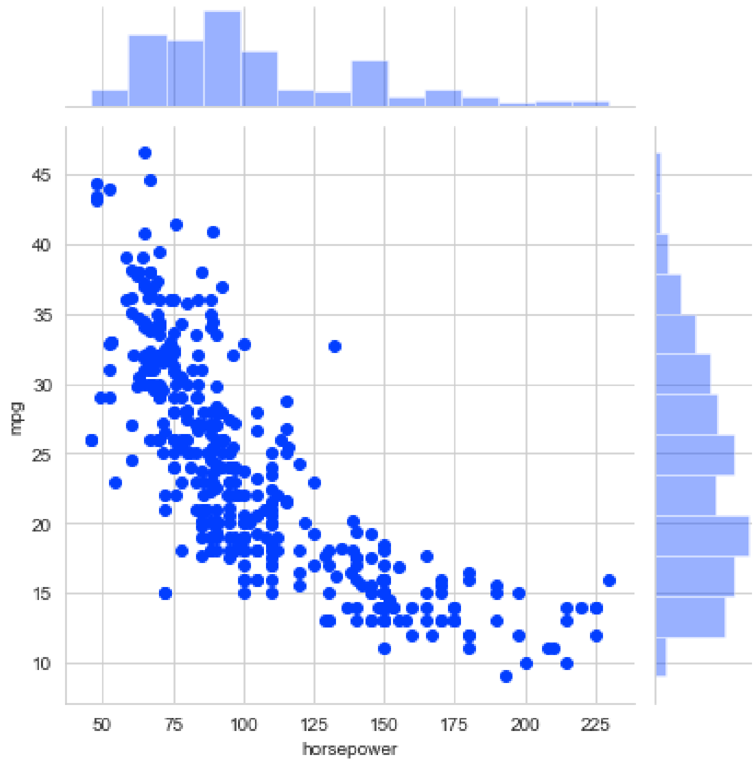
Mpg vs Horsepower

In [69]:

```
#joint plot for horsepower and weight of car
sns.jointplot("horsepower","mpg",data)
```

Out[69]:

```
<seaborn.axisgrid.JointGrid at 0x2a9d1ab9e80>
```



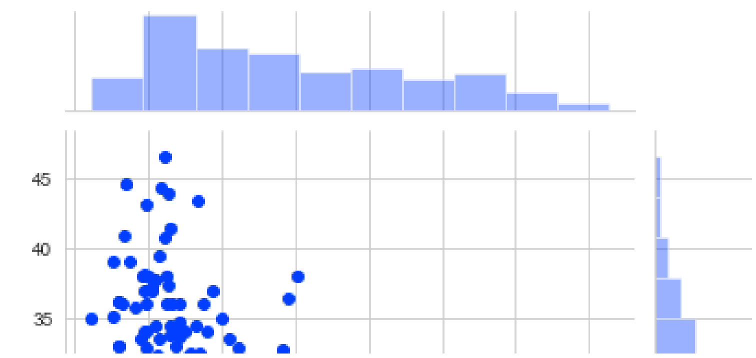
Mpg vs Weight

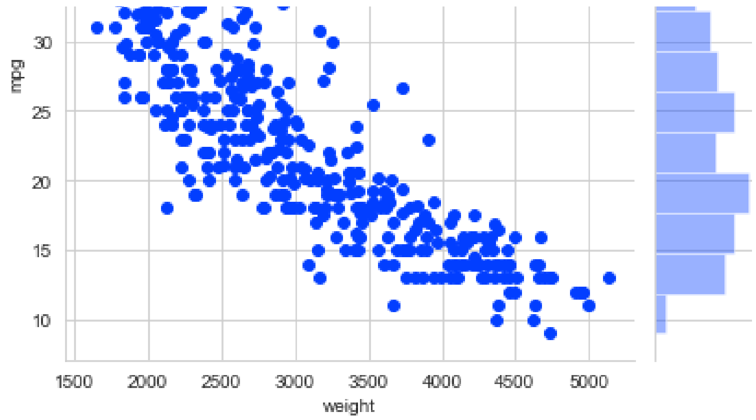
In [70]:

```
#joint plot for horsepower and weight of car
sns.jointplot("weight","mpg",data)
```

Out[70]:

```
<seaborn.axisgrid.JointGrid at 0x2a9d1b9ca20>
```





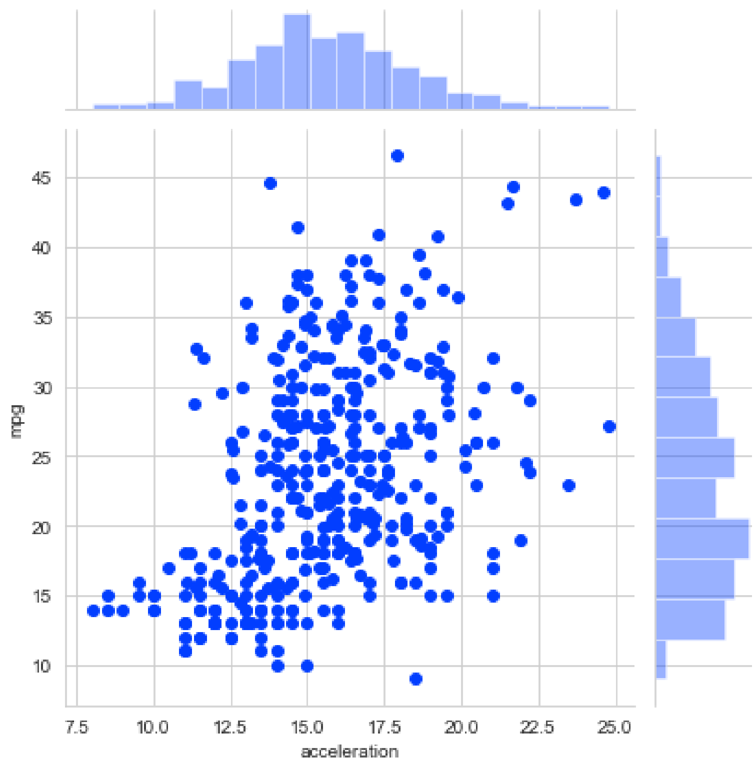
Mpg vs Acceleration

In [71]:

```
#joint plot for horsepower and weight of car
sns.jointplot("acceleration","mpg",data)
```

Out[71]:

```
<seaborn.axisgrid.JointGrid at 0x2a9d31a72b0>
```



As we see MPG value decreases as we increase weight or displacement or horsepower of car

Mpg value only increases slightly when we increase Acceleration of car

3. Violin and box plots

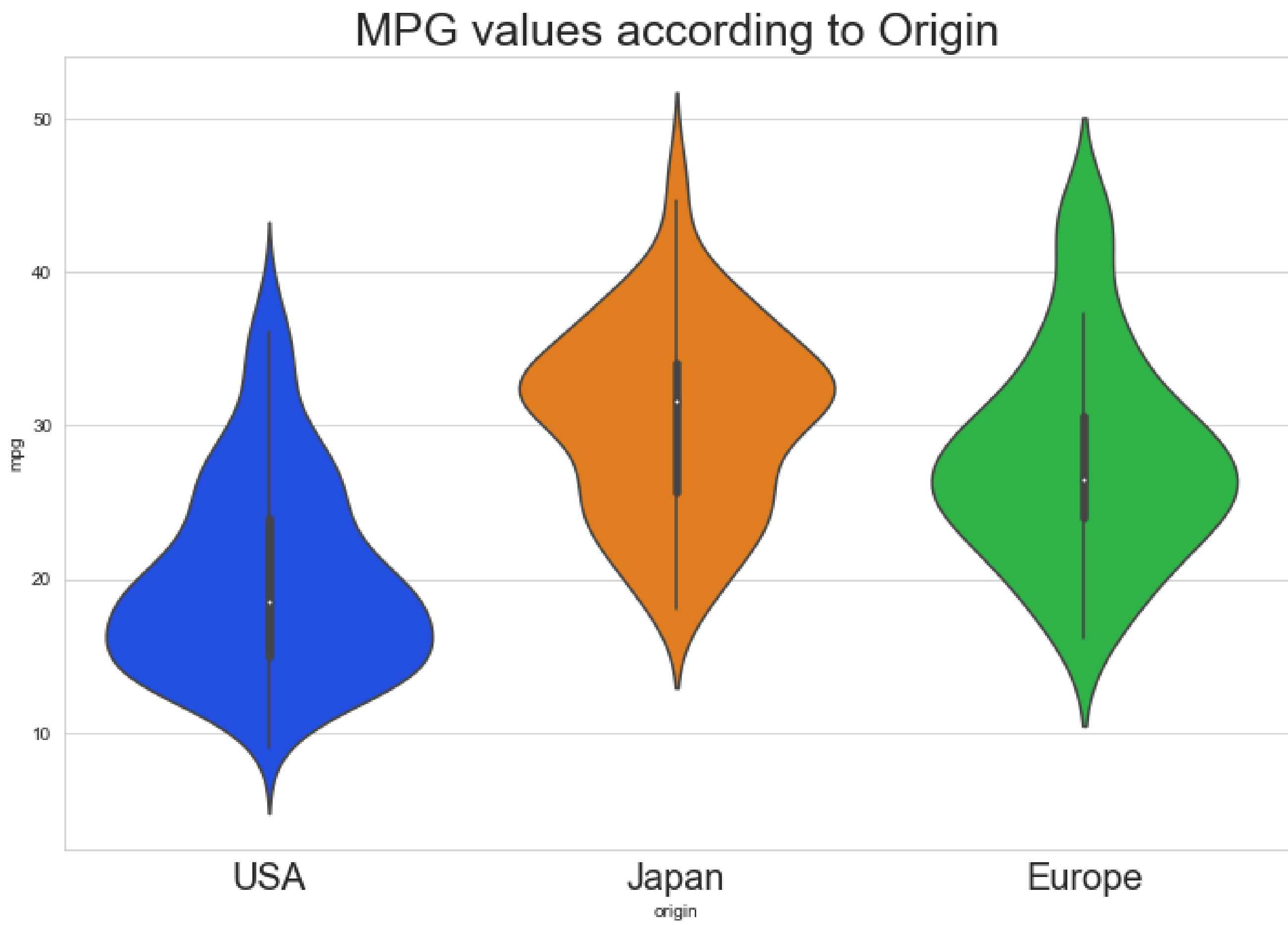
MPG vs Origin

In [72]:

```
plt.figure(figsize=(12,8))
plt.title("MPG values according to Origin",fontsize=25)
plt.xticks(fontsize=20)

sns.violinplot(x="origin",y="mpg",data=data)
```

Out[72]:

<matplotlib.axes._subplots.AxesSubplot at 0x2a9d31cac18>

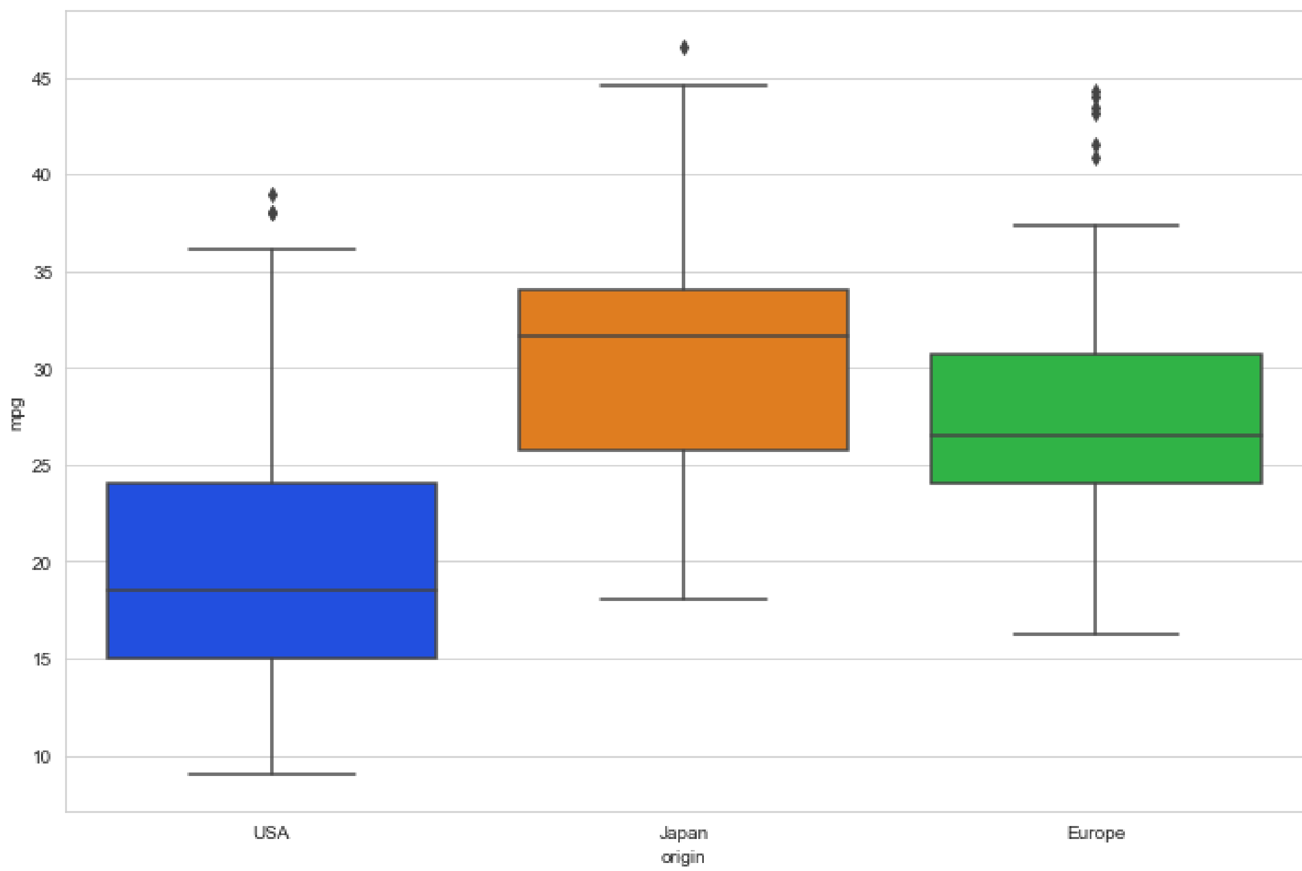


In [73]:

```
plt.figure(figsize=(12,8))
sns.boxplot(x="origin",y="mpg",data=data)
```

Out[73]:

<matplotlib.axes._subplots.AxesSubplot at 0x2a9d331c320>



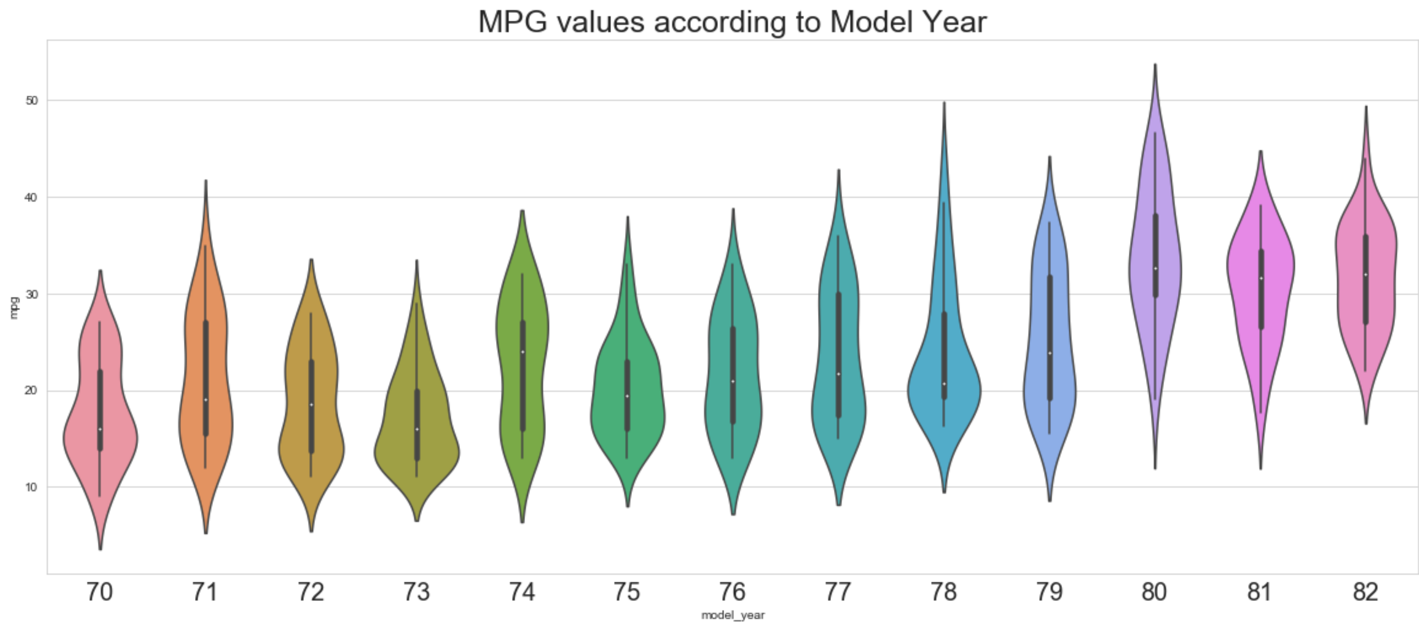
MPG vs Model Year

In [74]:

```
plt.figure(figsize=(20,8))
plt.title("MPG values according to Model Year",fontsize=25)
plt.xticks(fontsize=20)

sns.violinplot(x="model_year",y="mpg",data=data)
```

Out[74]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2a9d341fe80>
```

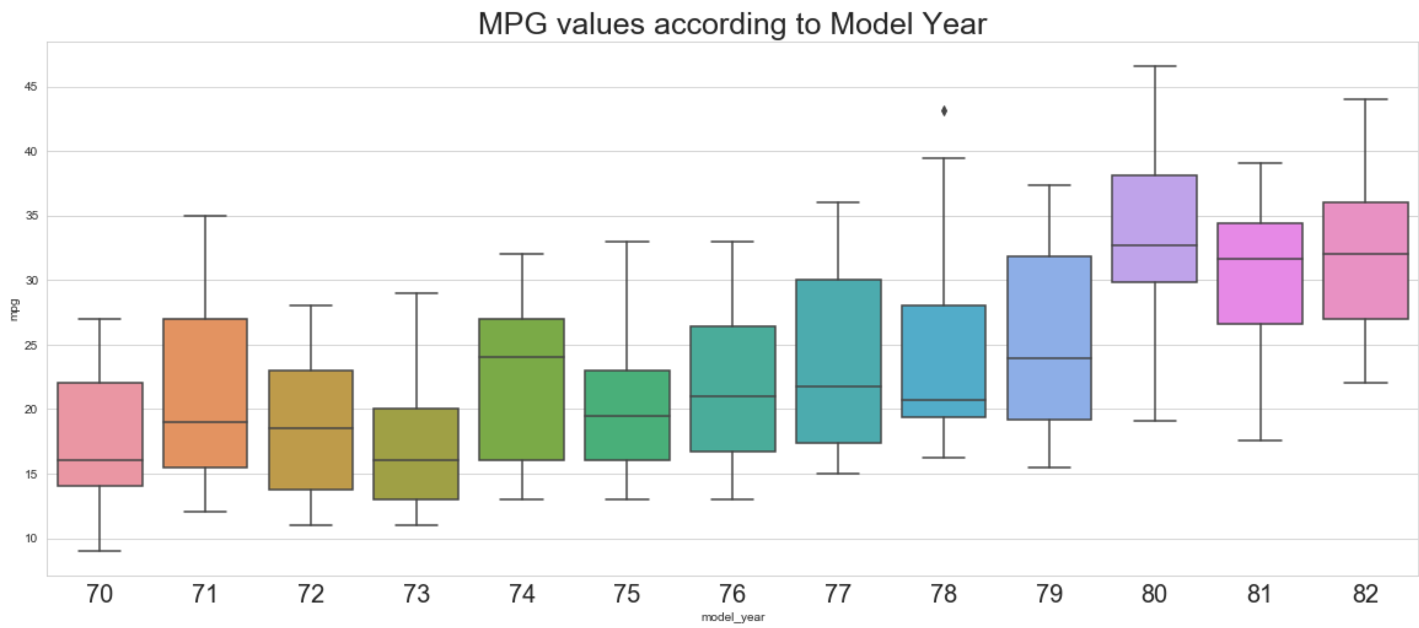


In [75]:

```
plt.figure(figsize=(20,8))
plt.title("MPG values according to Model Year",fontsize=25)
plt.xticks(fontsize=20)

sns.boxplot(x="model_year",y="mpg",data=data)
```

Out[75]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2a9d37df908>
```



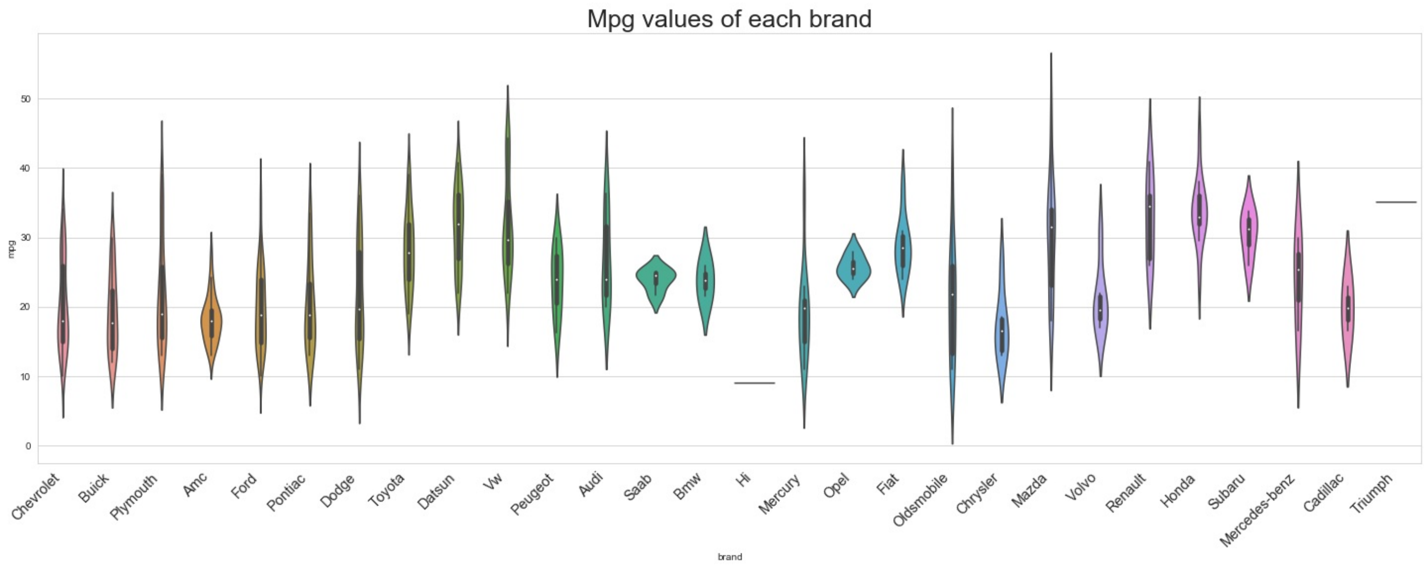
MPG vs Brand

In [76]:

```
plt.figure(figsize=(25,8))
plt.title("Mpg values of each brand",fontsize=25)
plt.xticks(rotation=45, horizontalalignment='right',fontweight='light',fontsize='x-large'
)


sns.violinplot(x="brand",y="mpg",data=data)
```

Out[76]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2a9d38f1d68>
```

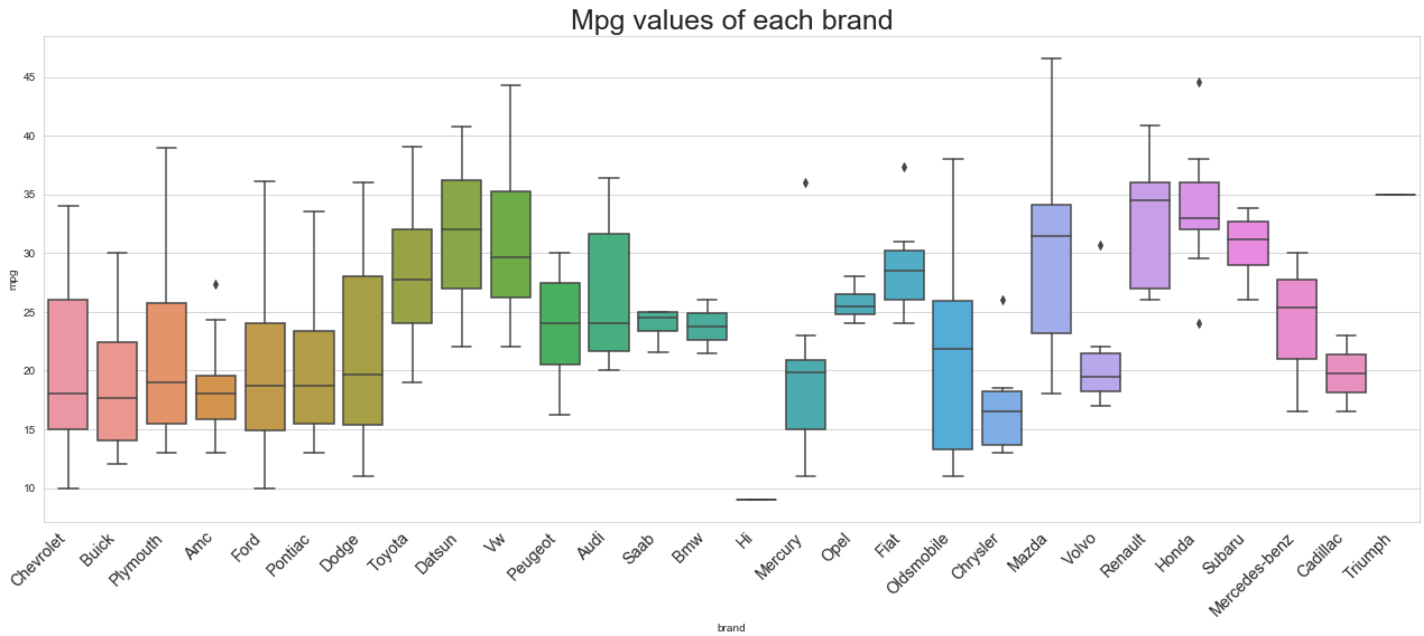


In [77]:

```
plt.figure(figsize=(22,8))
plt.title("Mpg values of each brand",fontsize=25)
plt.xticks(rotation=45, horizontalalignment='right',fontweight='light',fontsize='x-large'
)


sns.boxplot(x="brand",y="mpg",data=data)
```

Out[77]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2a9d38f1320>
```



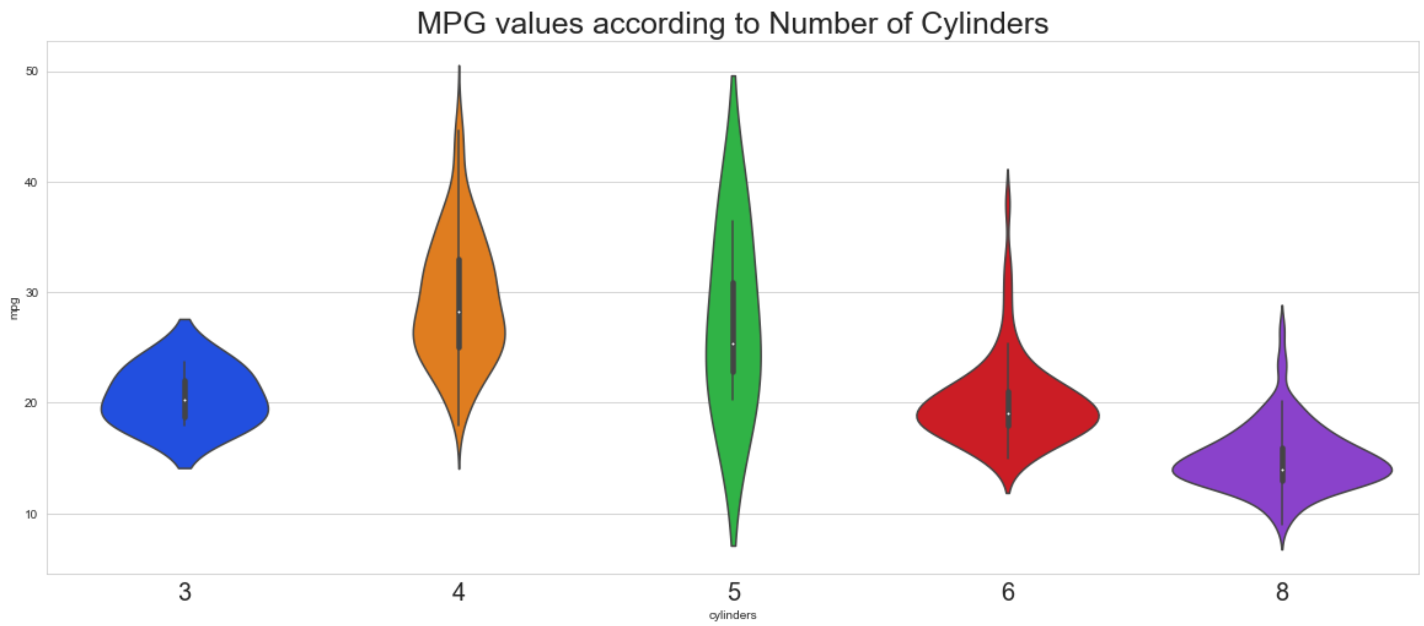
MPG vs Cylinders

In [78]:

```
plt.figure(figsize=(20,8))
plt.title("MPG values according to Number of Cylinders",fontsize=25)
plt.xticks(fontsize=20)

sns.violinplot(x="cylinders",y="mpg",data=data)
```

Out[78]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2a9d3aab3c8>
```

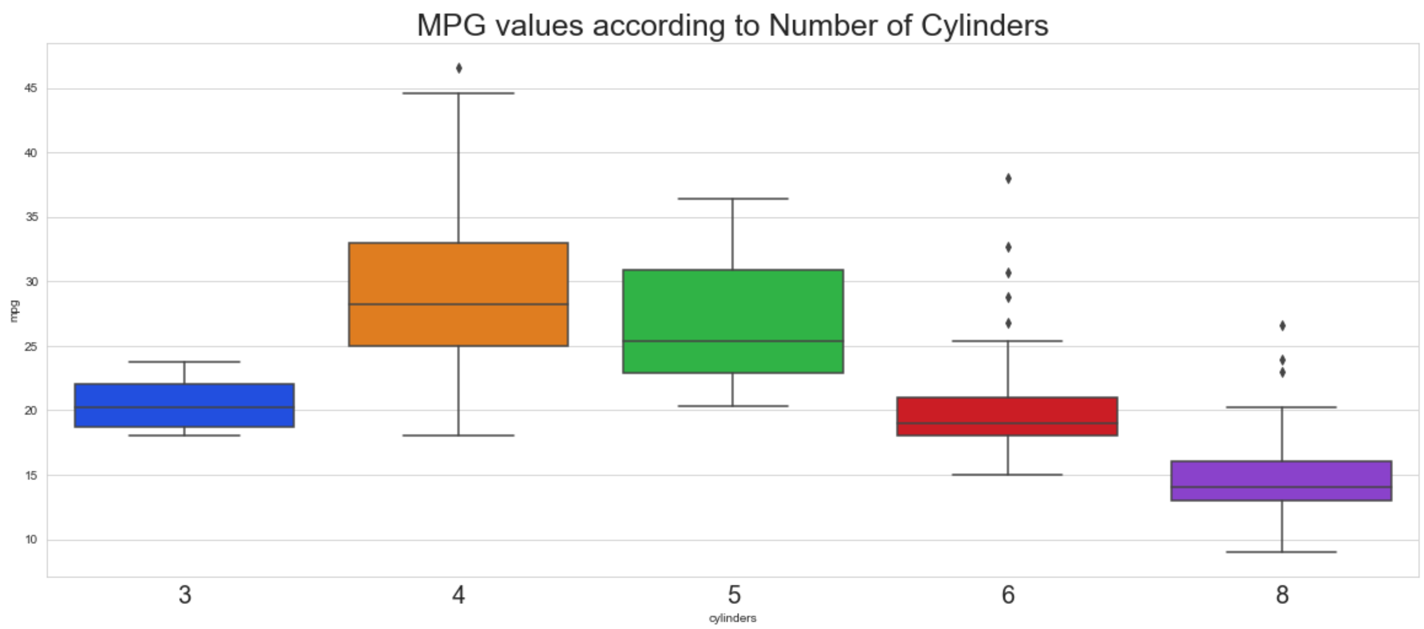


In [79]:

```
plt.figure(figsize=(20,8))
plt.title("MPG values according to Number of Cylinders",fontsize=25)
plt.xticks(fontsize=20)

sns.boxplot(x="cylinders",y="mpg",data=data)
```

Out[79]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2a9d3417390>
```



4.3 Heatmaps

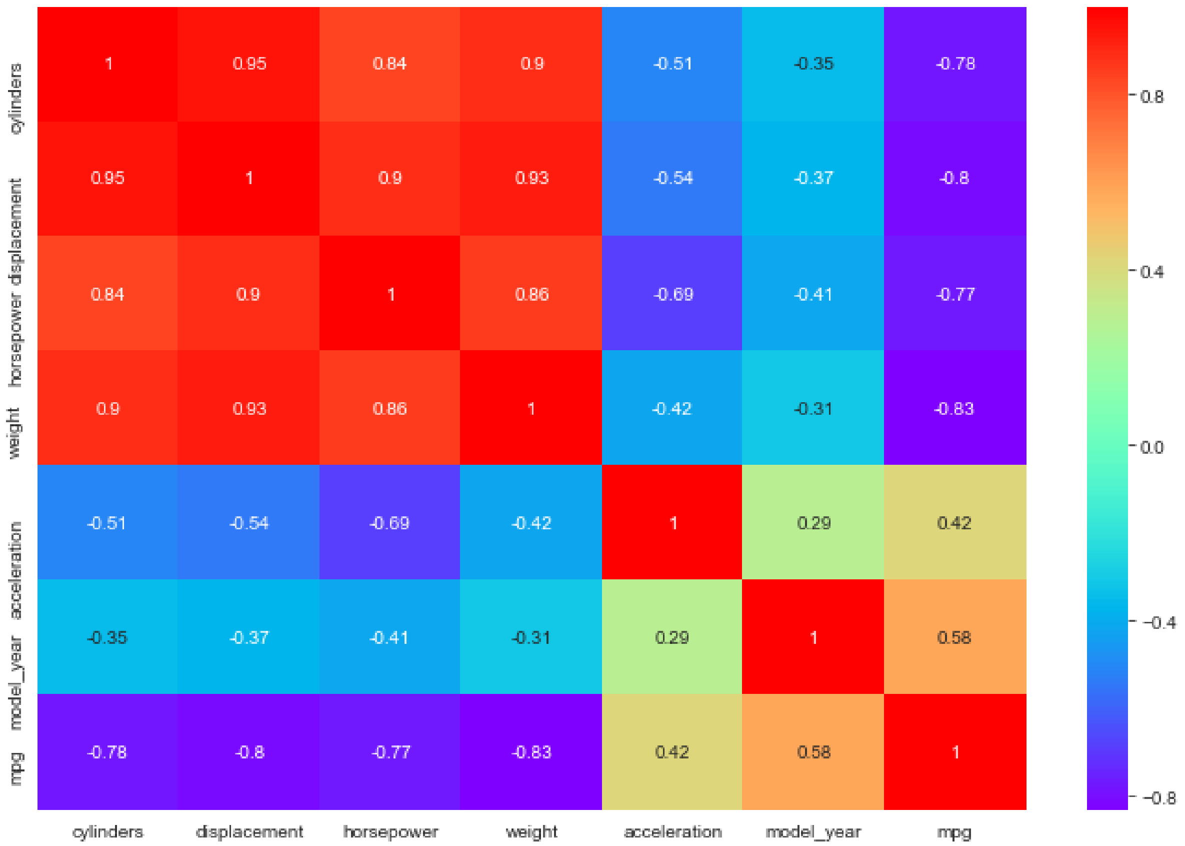
Heat map of correlation of data

In [80]:

```
plt.figure(figsize=(12,8))
sns.heatmap(data.corr(),annot=True,cmap="rainbow")
```

Out[80]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2a9d43e3860>
```



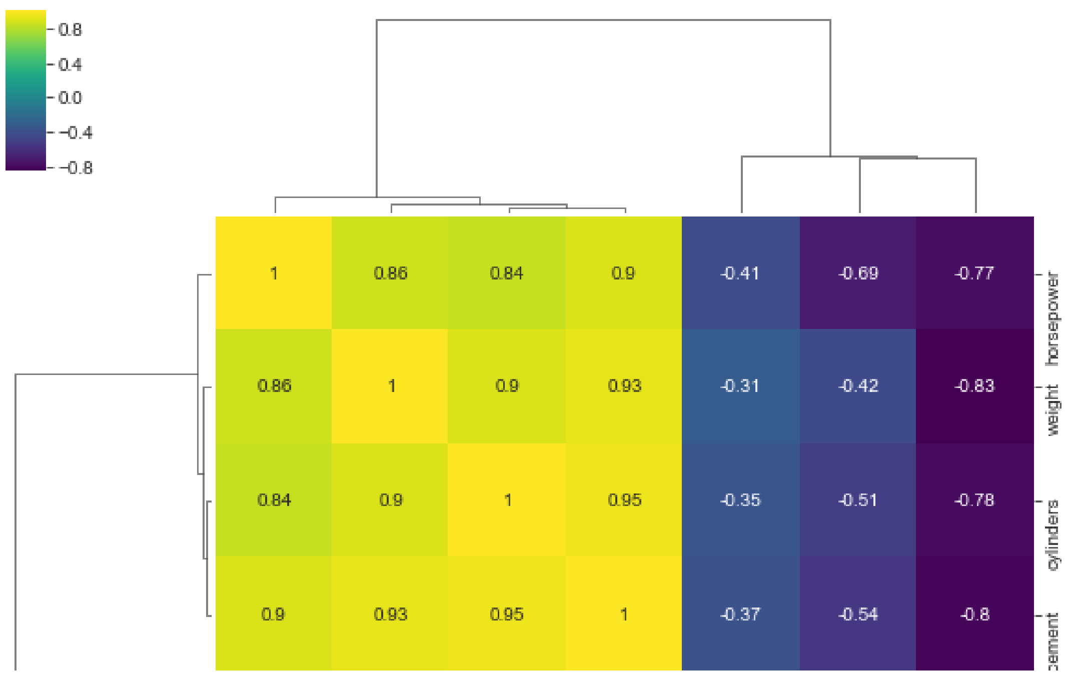
Cluster Map

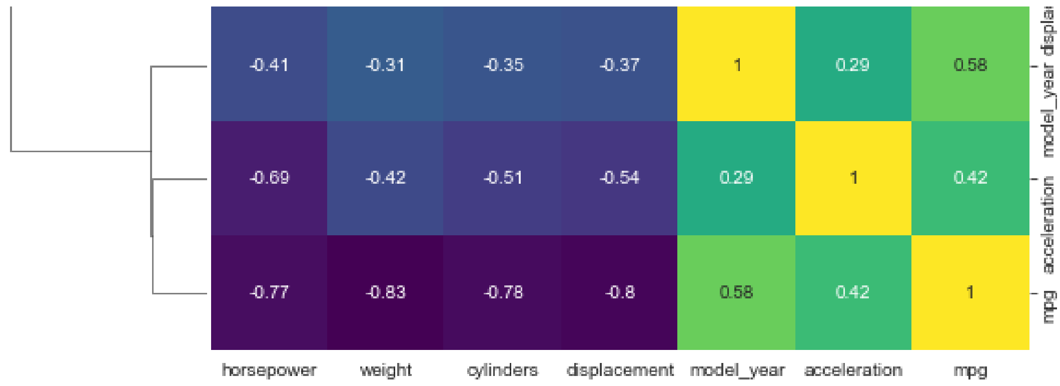
In [81]:

```
plt.figure(figsize=(12,8))
sns.clustermap(data.corr(),cmap="viridis",annot=True)
```

Out[81]:

```
<seaborn.matrix.ClusterGrid at 0x2a9d4e9ca20>
<Figure size 864x576 with 0 Axes>
```





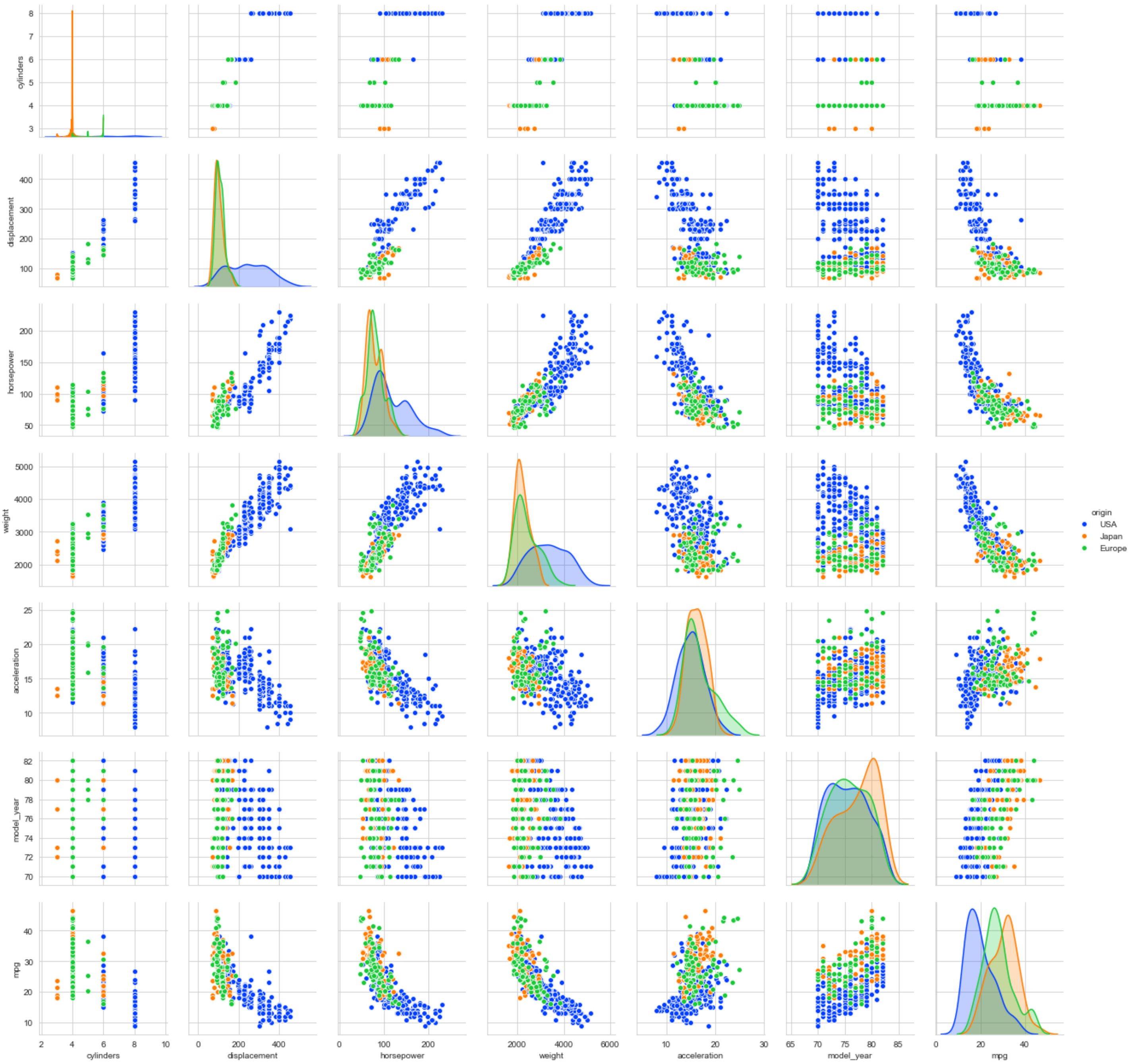
4.4 Pair plot

In [82]:

```
sns.pairplot(data,hue="origin")
```

Out[82]:

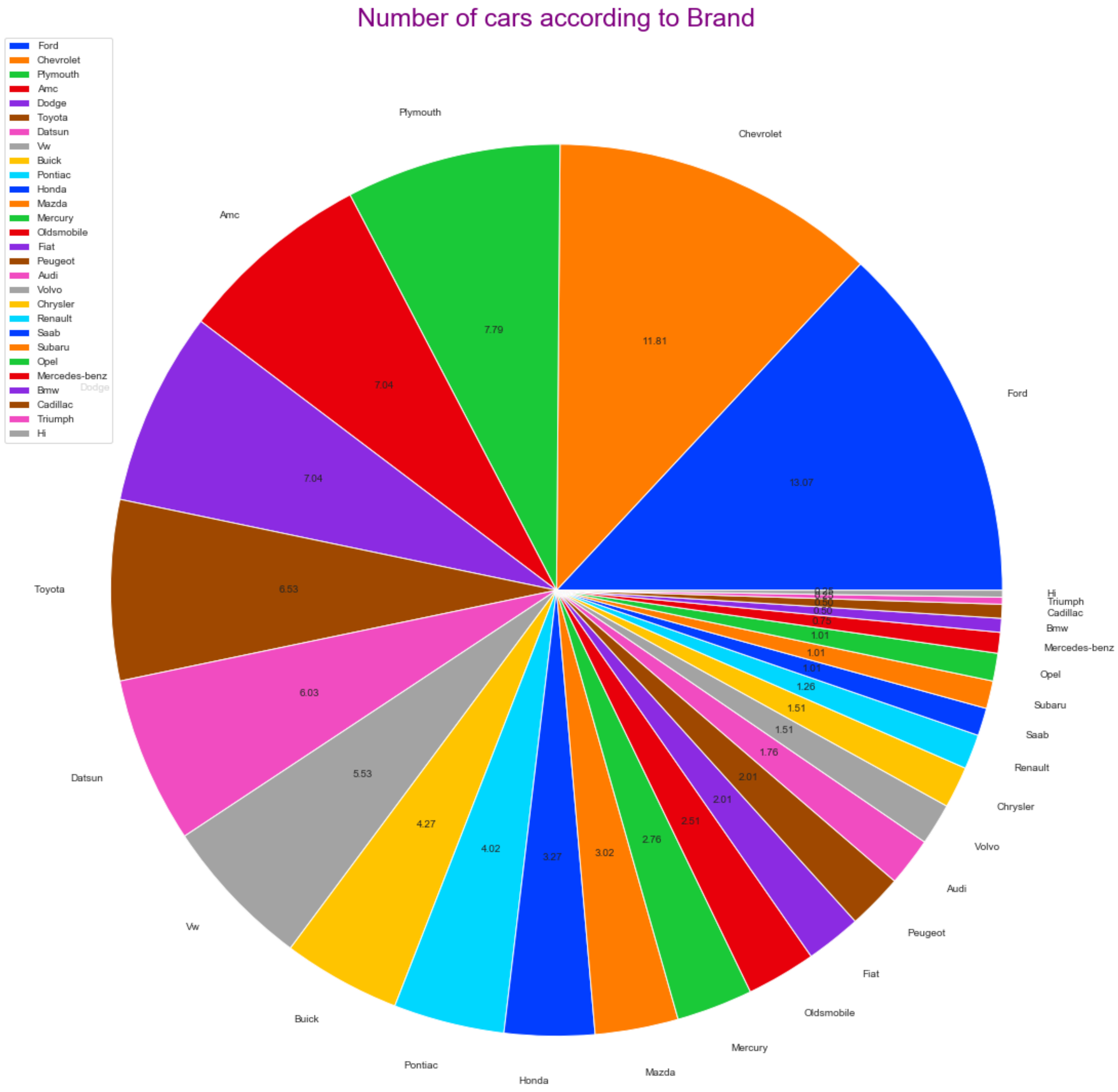
<seaborn.axisgrid.PairGrid at 0x2a9d4d25860>



4.5 Pie charts

In [83]:

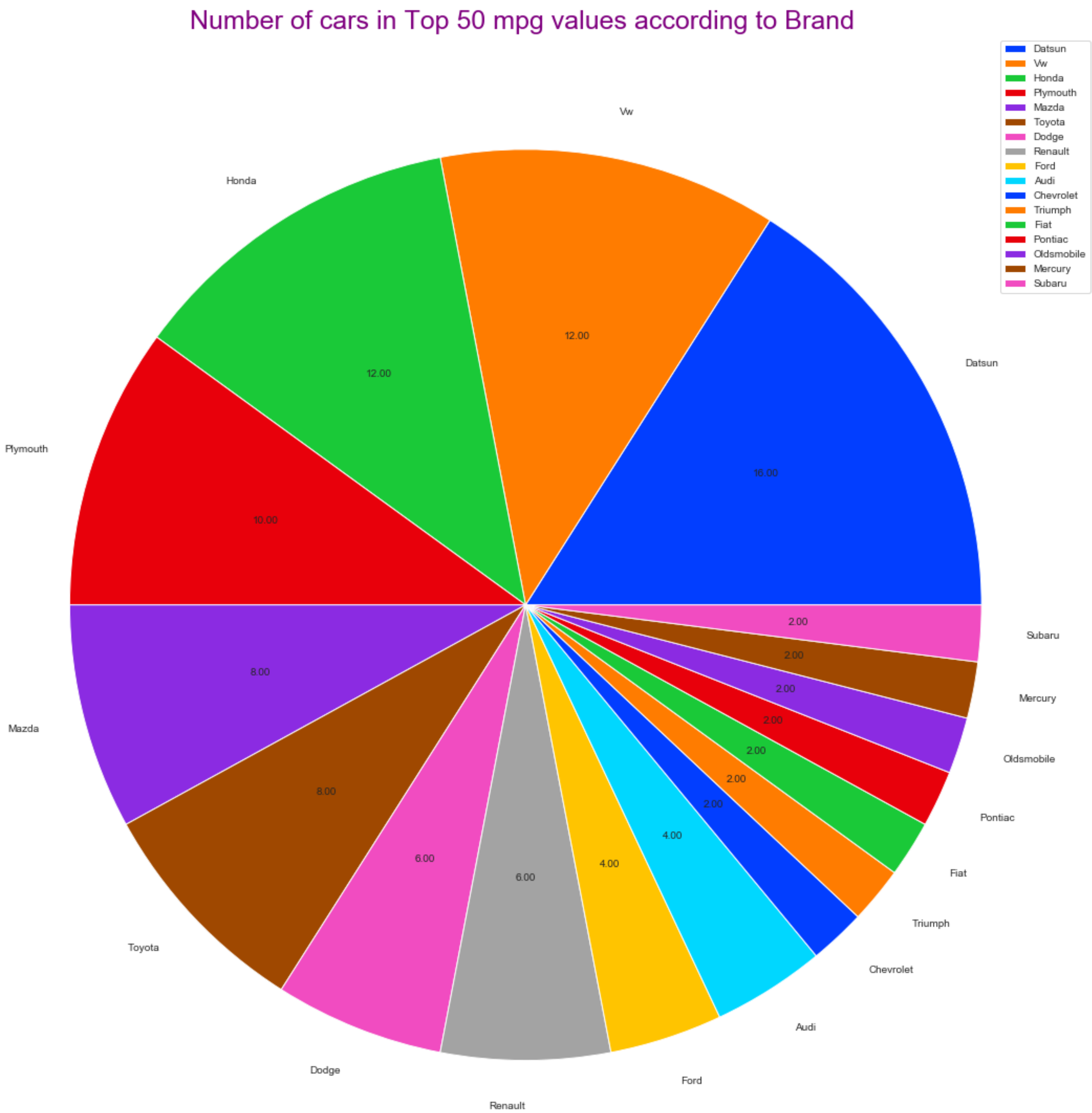
```
plt.figure(figsize=(20,20))
ax =data["brand"].value_counts()
labels=data["brand"].value_counts().index
plt.pie(ax,labels=labels,autopct='%.2f')
plt.title("Number of cars according to Brand",fontsize=25,color='purple')
plt.legend()
plt.show()
```



Ford has the most number of cars

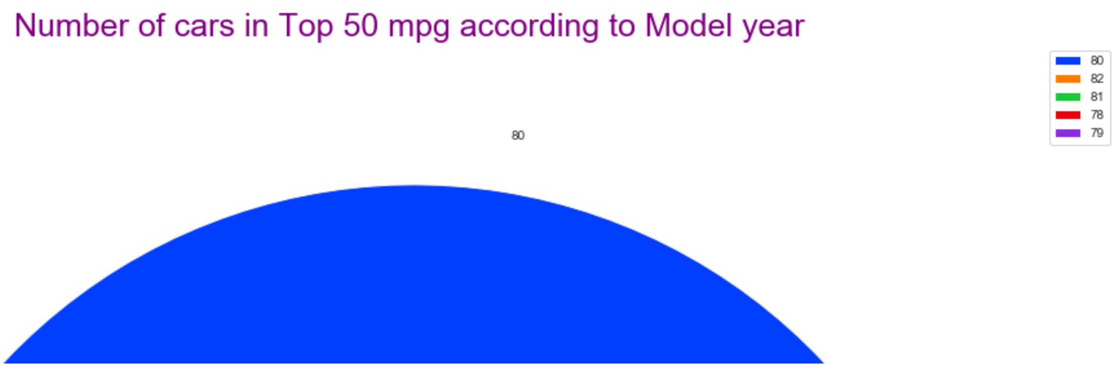
In [84]:

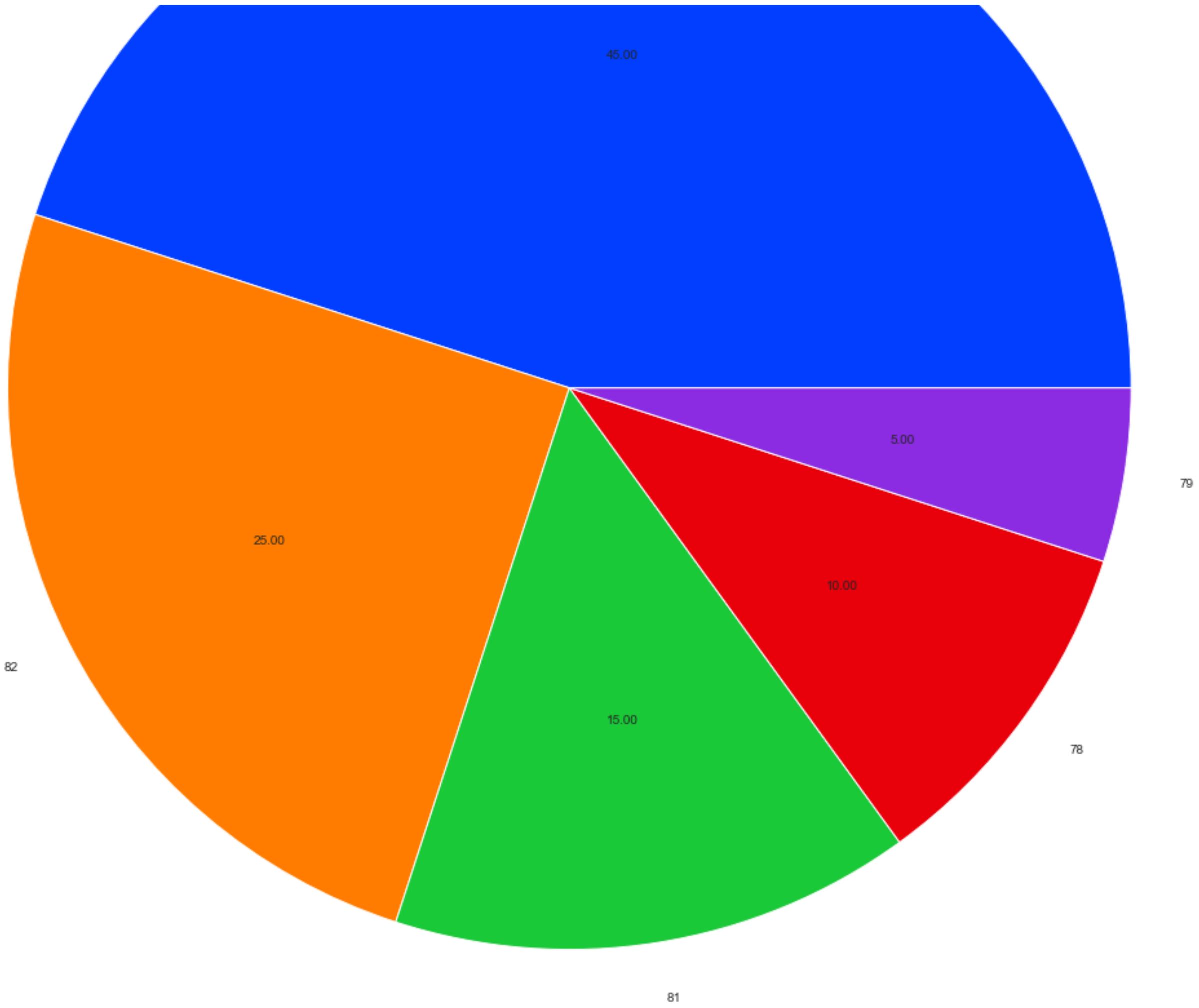
```
plt.figure(figsize=(20,20))
ax =data.sort_values(by="mpg",ascending=False)["brand"][:50].value_counts()
labels=data.sort_values(by="mpg",ascending=False)["brand"][:50].value_counts().index
plt.pie(ax,labels=labels,autopct='%.2f')
plt.title("Number of cars in Top 50 mpg values according to Brand ",fontsize=25,color='pu
rple')
plt.legend()
plt.show()
```



In [85]:

```
plt.figure(figsize=(20,20))
ax =data.sort_values(by="mpg",ascending=False)["model_year"][:20].value_counts()
labels=data.sort_values(by="mpg",ascending=False)["model_year"][:20].value_counts().inde
x
plt.pie(ax,labels=labels,autopct='%.2f')
plt.title("Number of cars in Top 50 mpg according to Model year ",fontsize=25,color='purp
le')
plt.legend()
plt.show()
```

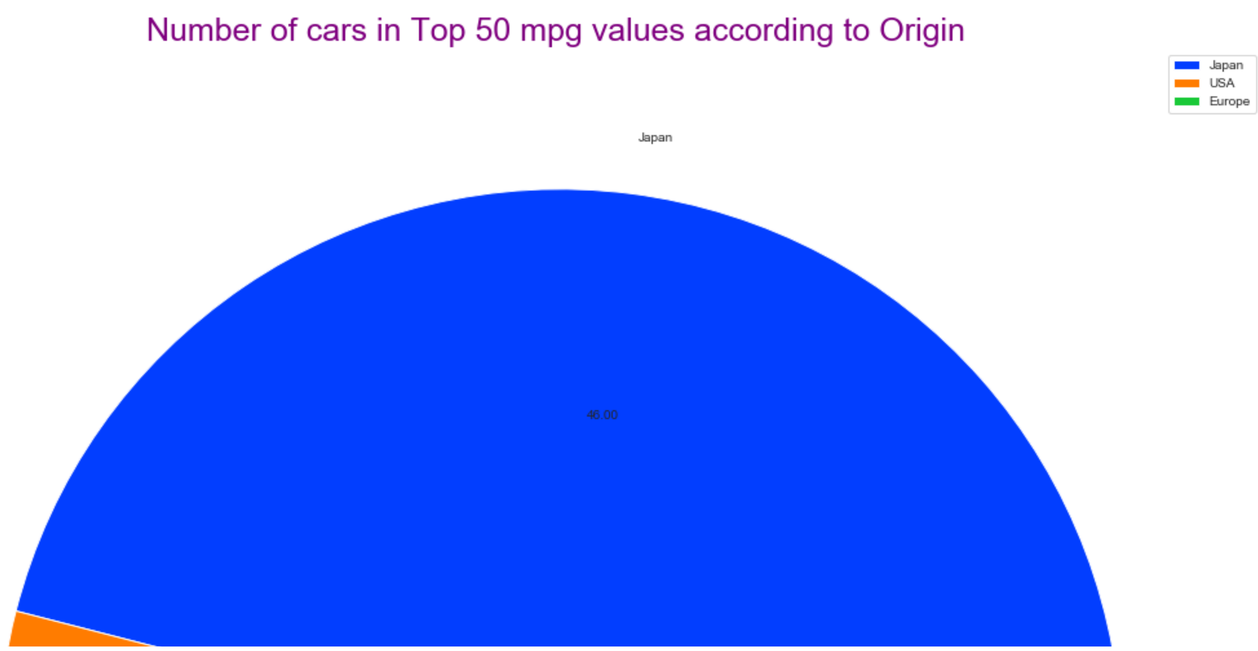


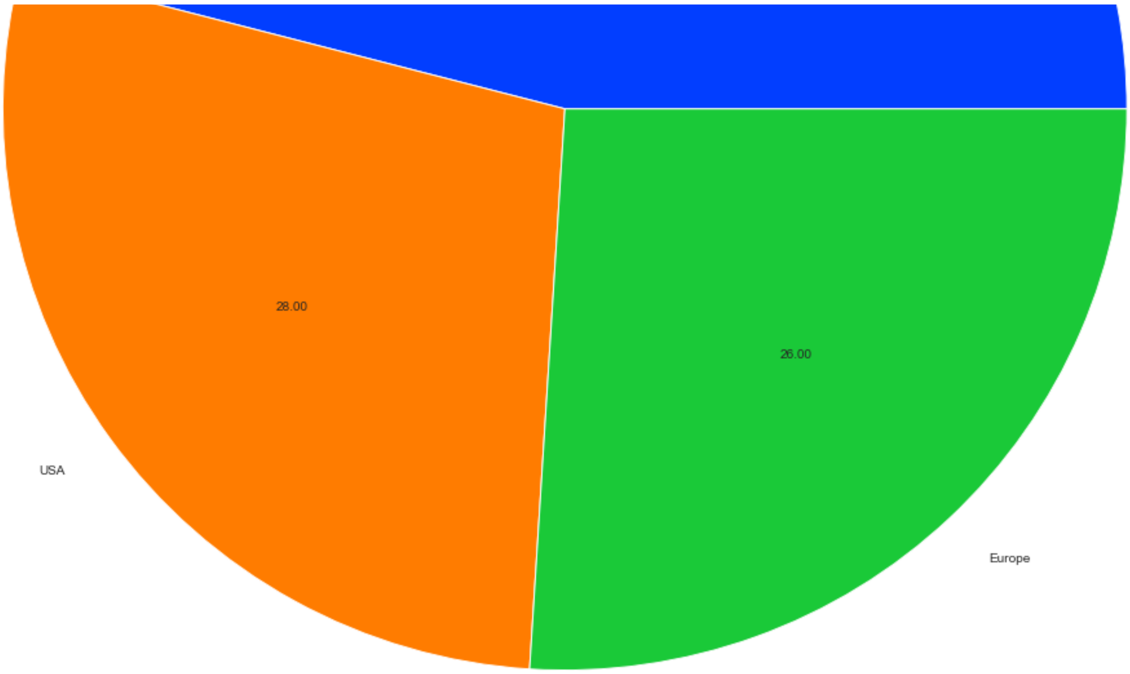


Cars of Model year 80 has the most number of cars in Top 50 cars with highest mpg values

In [86]:

```
plt.figure(figsize=(20,20))
ax =data.sort_values(by="mpg",ascending=False)["origin"][:50].value_counts()
labels=data.sort_values(by="mpg",ascending=False)["origin"][:50].value_counts().index

plt.pie(ax,labels=labels,autopct='%.2f')
plt.title("Number of cars in Top 50 mpg values according to Origin ",fontsize=25,color='p
urple')
plt.legend()
plt.show()
```

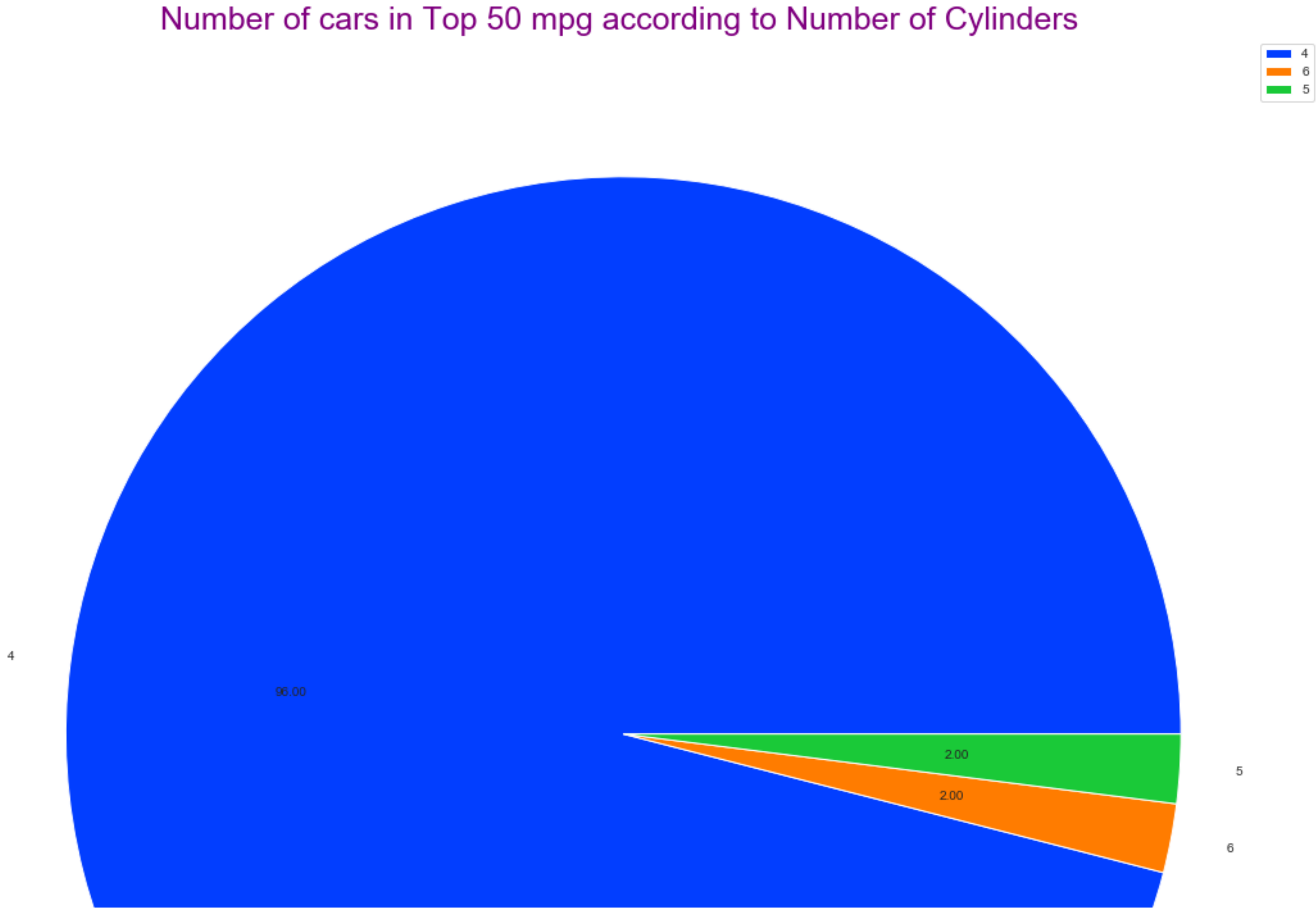




Japan cars have the most number of cars in Top 50 cars with highest Mpg values

In [87]:

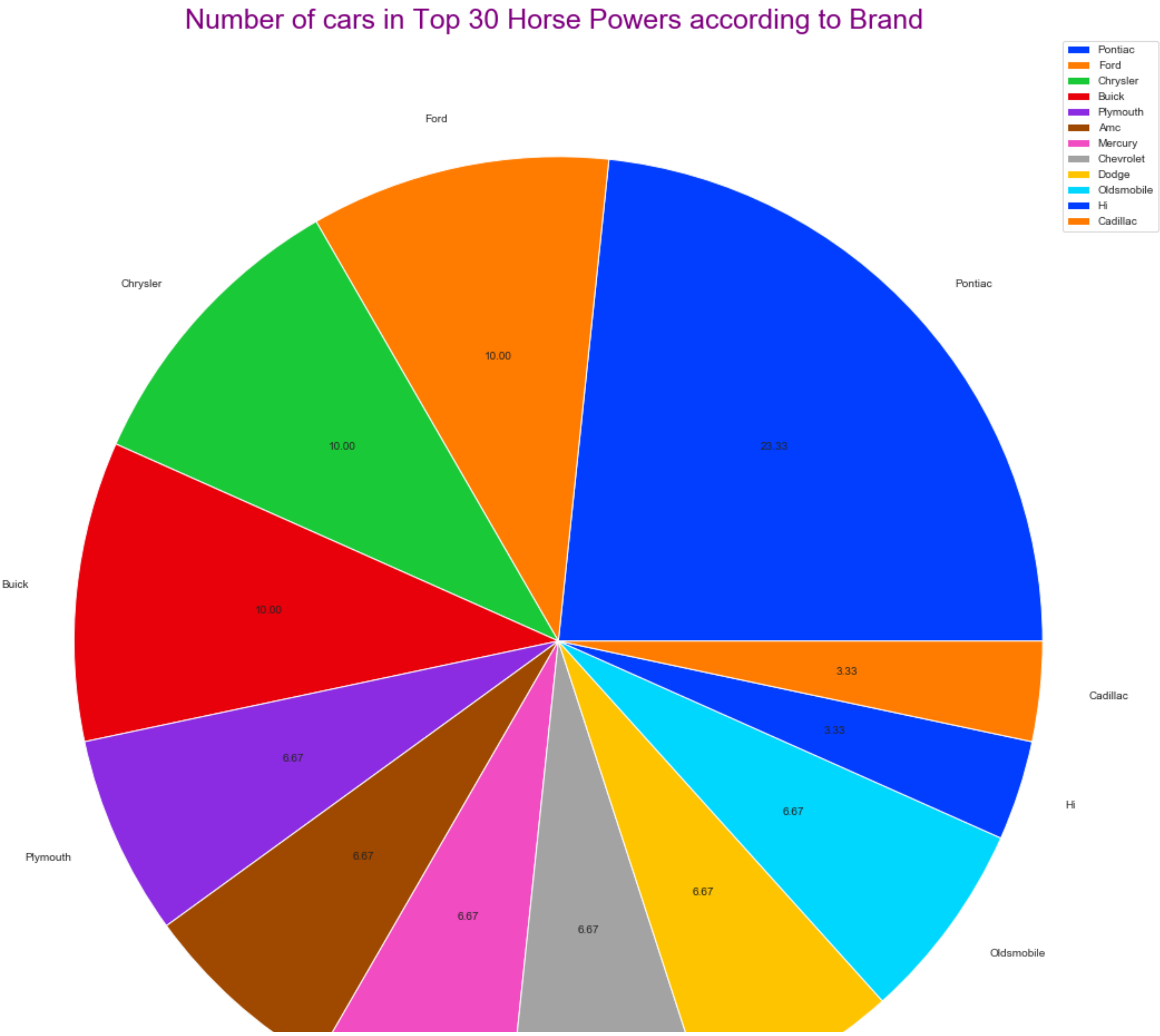
```
plt.figure(figsize=(20,20))
ax =data.sort_values(by="mpg",ascending=False)["cylinders"][:50].value_counts()
labels=data.sort_values(by="mpg",ascending=False)["cylinders"][:50].value_counts().index

plt.pie(ax,labels=labels,autopct='%.2f')
plt.title("Number of cars in Top 50 mpg according to Number of Cylinders ",fontsize=25,co
lor='purple')
plt.legend()
plt.show()
```

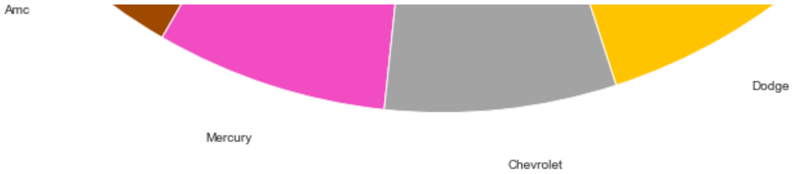


Cars with 4 cylinders have the most number of cars(48 cars out of 50) in Top 50 cars with highest Mpg values

In [88]:

```
plt.figure(figsize=(20,20))
ax =data.sort_values(by="horsepower",ascending=False)["brand"][:30].value_counts()
labels=data.sort_values(by="horsepower",ascending=False)["brand"][:30].value_counts().in
dex
plt.pie(ax,labels=labels,autopct='%.2f')
plt.title("Number of cars in Top 30 Horse Powers according to Brand ",fontsize=25,color='
purple')
plt.legend()
plt.show()
```





Pontiac cars have the most number of cars in Top 30 cars with Highest Horsepower

5. Data Preprocessing

In [89]:

```
data.head()
```

Out[89]:

	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	mpg	brand
0	8	307.0	130.0	3504.0	12.0	70	USA	18.0	Chevrolet
1	8	350.0	165.0	3693.0	11.5	70	USA	15.0	Buick
2	8	318.0	150.0	3436.0	11.0	70	USA	18.0	Plymouth
3	8	304.0	150.0	3433.0	12.0	70	USA	16.0	Amc
4	8	302.0	140.0	3449.0	10.5	70	USA	17.0	Ford

5.1 X and Y values

mpg column is target variable

In [90]:

```
#Dependent variable

y=data.iloc[:,7].values
```

In [91]:

```
#Independent variable

x=data.drop("mpg",axis=1).values
```

In [127]:

```
x[:10]
```

Out[127]:

```
array([[8, 307.0, 130.0, 3504.0, 12.0, 70, 'USA', 'Chevrolet'],
       [8, 350.0, 165.0, 3693.0, 11.5, 70, 'USA', 'Buick'],
       [8, 318.0, 150.0, 3436.0, 11.0, 70, 'USA', 'Plymouth'],
       [8, 304.0, 150.0, 3433.0, 12.0, 70, 'USA', 'Amc'],
       [8, 302.0, 140.0, 3449.0, 10.5, 70, 'USA', 'Ford'],
       [8, 429.0, 198.0, 4341.0, 10.0, 70, 'USA', 'Ford'],
       [8, 454.0, 220.0, 4354.0, 9.0, 70, 'USA', 'Chevrolet'],
       [8, 440.0, 215.0, 4312.0, 8.5, 70, 'USA', 'Plymouth'],
       [8, 455.0, 225.0, 4425.0, 10.0, 70, 'USA', 'Pontiac'],
       [8, 390.0, 190.0, 3850.0, 8.5, 70, 'USA', 'Amc']], dtype=object)
```

In [93]:

```
y[:10]
```

Out[93]:

Out[93]:

```
array([18., 15., 18., 16., 17., 15., 14., 14., 14., 15.])
```

In [94]:

```
x.shape
```

Out[94]:

```
(398, 8)
```

5.2 Encoding Categorical Data

OneHotEncoding

1.origin x[:,6]

2.cylinders x[:,0]

In [126]:

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct=ColumnTransformer(transformers=[('encode',OneHotEncoder(),[0,6])],
                    remainder="passthrough")
x_s=np.array(ct.fit_transform(x))
x_s[29]
```

Out[126]:

```
array([0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 97.0, 88.0, 2130.0, 14.5,
       71, 'Datsun'], dtype=object)
```

In [96]:

```
x_s[:,8:]
```

Out[96]:

```
array([[307.0, 130.0, 3504.0, 12.0, 70, 'Chevrolet'],
       [350.0, 165.0, 3693.0, 11.5, 70, 'Buick'],
       [318.0, 150.0, 3436.0, 11.0, 70, 'Plymouth'],
       ...,
       [135.0, 84.0, 2295.0, 11.6, 82, 'Dodge'],
       [120.0, 79.0, 2625.0, 18.6, 82, 'Ford'],
       [119.0, 82.0, 2720.0, 19.4, 82, 'Chevrolet']], dtype=object)
```

LabelEncoding

3.model_year x[:,12]

4.Brand x[:,13]

In [183]:

```
from sklearn.preprocessing import LabelEncoder
le_brand=LabelEncoder()
le_year=LabelEncoder()
x_s[:,13]=le_brand.fit_transform(x_s[:,13].astype(str))
x_s[:,12]=le_year.fit_transform(x_s[:,12])
x_s[:2]
```

Out[183]:

```
array([[0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 307.0, 130.0, 3504.0,
        12.0, 0, 16],
       [0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 350.0, 165.0, 3693.0,
        11.5, 0, 14]], dtype=object)
```

In [181]:

```
newdata=pd.DataFrame(x_s,columns=["3","4","5","6","8","Europe","Japan","USA","displacement","horsepower","weight","acceleration","model_year","brand"])
newdata.head()
```

Out[181]:

	3	4	5	6	8	Europe	Japan	USA	displacement	horsepower	weight	acceleration	model_year	brand
0	0	0	0	0	1	0	0	1	307	130	3504	12	0	5
1	0	0	0	0	1	0	0	1	350	165	3693	11.5	0	3
2	0	0	0	0	1	0	0	1	318	150	3436	11	0	19
3	0	0	0	0	1	0	0	1	304	150	3433	12	0	0
4	0	0	0	0	1	0	0	1	302	140	3449	10.5	0	10

The first eight columns have binary values

The last two columns have labelled values

5.3 Train_test_split

In [143]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x_s,y,test_size=0.3,random_state=101)
```

In [144]:

```
x_train
```

Out[144]:

```
array([[0.0, 0.0, 0.0, ..., 13.0, 1, 19],
       [0.0, 1.0, 0.0, ..., 14.0, 1, 24],
       [0.0, 1.0, 0.0, ..., 14.5, 8, 8],
       ...,
       [0.0, 1.0, 0.0, ..., 17.0, 10, 12],
       [0.0, 0.0, 0.0, ..., 8.0, 0, 19],
       [0.0, 1.0, 0.0, ..., 16.2, 11, 10]], dtype=object)
```

In [145]:

```
x_test.shape
```

Out[145]:

```
(120, 14)
```

5.4 Feature Scaling

In [146]:

```
x_train[:2]
```

Out[146]:

```
array([[0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 318.0, 150.0, 4096.0,
        13.0, 1, 19],
       [0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 113.0, 95.0, 2228.0,
        14.0, 1, 24]], dtype=object)
```

In X

numerical columns starts from 8 and ends at 11

column 8 is displacement

column 9 is horsepower

column 10 is weight

column 11 is acceleration

x[:,8:-1]

In [147]:

```
x_train_scaled=x_train
x_test_scaled=x_test
```

In [148]:

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train_scaled[:,8:12]=sc.fit_transform(x_train_scaled[:,8:12])
x_test_scaled[:,8:12]=sc.transform(x_test_scaled[:,8:12])
x_train_scaled
```

Out[148]:

```
array([[0.0, 0.0, 0.0, ..., -0.9477638194514814, 1, 19],
       [0.0, 1.0, 0.0, ..., -0.5852454625341561, 1, 24],
       [0.0, 1.0, 0.0, ..., -0.4039862840754934, 8, 8],
       ...,
       [0.0, 1.0, 0.0, ..., 0.5023096082178199, 10, 12],
       [0.0, 0.0, 0.0, ..., -2.760355604038108, 0, 19],
       [0.0, 1.0, 0.0, ..., 0.21229492268395941, 11, 10]], dtype=object)
```

In [149]:

```
x_train
```

Out[149]:

```
array([[0.0, 0.0, 0.0, ..., -0.9477638194514814, 1, 19],
       [0.0, 1.0, 0.0, ..., -0.5852454625341561, 1, 24],
       [0.0, 1.0, 0.0, ..., -0.4039862840754934, 8, 8],
       ...,
       [0.0, 1.0, 0.0, ..., 0.5023096082178199, 10, 12],
       [0.0, 0.0, 0.0, ..., -2.760355604038108, 0, 19],
       [0.0, 1.0, 0.0, ..., 0.21229492268395941, 11, 10]], dtype=object)
```

6. Machine Learning Models

6.1 Linear Regression

Training the model

In [150]:

```
from sklearn.linear_model import LinearRegression
Linreg=LinearRegression()
Linreg_fs=LinearRegression()
Linreg.fit(x_train,y_train)
```

Out[150]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

Model Prediction

In [151]:

```
P_linreg=Linreg.predict(x_test)
```

```
P_linreg=Linreg.predict(x_test)
```

Evaluting the model

In [152]:

```
from sklearn.metrics import mean_squared_error,r2_score
mse_linreg=mean_squared_error(y_test,P_linreg)

print(np.sqrt(mse_linreg))
```

```
3.043589961027452
```

In [153]:

```
lin_score=r2_score(y_test,P_linreg)*100
print(lin_score)
```

```
85.21942054278513
```

6.2 Support Vector Regression

Scaling y values

In [154]:

```
sc_y=StandardScaler()
y_train_svm=y_train.reshape(len(y_train),1)
ys_train=sc_y.fit_transform(y_train_svm)
y_test_svm=y_test.reshape(len(y_test),1)
ys_test=sc_y.transform(y_test_svm)
```

In [155]:

```
#ys_train
```

Training the model

In [156]:

```
from sklearn.svm import SVR
svr=SVR(kernel='rbf')
svr.fit(x_train_scaled,ys_train)
```

Out[156]:

```
SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1,
  gamma='auto_deprecated', kernel='rbf', max_iter=-1, shrinking=True,
  tol=0.001, verbose=False)
```

Model Prediction

In [157]:

```
Ps_svr=svr.predict(x_test_scaled)
P_svr=sc_y.inverse_transform(Ps_svr)
```

Evaluting the model

In [158]:

```
mse_svr=mean_squared_error(y_test,P_svr)
print(np.sqrt(mse_svr))
```

```
2.8502619663034263
```

In [159]:

```
svr_score=r2_score(y_test,P_svr)*100
print(svr_score)
```

```
87.03750118021506
```

Train test split once again

In [160]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x_s,y,test_size=0.3,random_state=101)
```

6.3 Random Forest Regression

Training the model

In [161]:

```
from sklearn.ensemble import RandomForestRegressor
randomforest=RandomForestRegressor(n_estimators=100,random_state=101)
randomforest.fit(x_train,y_train)
```

Out[161]:

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None,
           oob_score=False, random_state=101, verbose=0, warm_start=False)
```

Model Prediction

In [162]:

```
P_forest=randomforest.predict(x_test)
```

Evaluting the model

In [163]:

```
forest_score=r2_score(y_test,P_forest)*100
print(forest_score)
```

```
91.06342937079626
```

6.4 Lasso

Training the model

In [164]:

```
from sklearn.linear_model import Lasso
lass=Lasso()
lass.fit(x_train,y_train)
```

Out[164]:

```
Lasso(alpha=1.0, copy_X=True, fit_intercept=True, max_iter=1000,
   normalize=False, positive=False, precompute=False, random_state=None,
   selection='cyclic', tol=0.0001, warm_start=False)
```

Model Prediction

In [165]:

```
P_lasso=lass.predict(x_test)
```

Evaluting the model

In [166]:

```
lasso_score=r2_score(y_test,P_lasso)*100
print(lasso_score)
```

```
81.08347246593218
```

6.5 Ridge Regression

Training the model

In [167]:

```
from sklearn.linear_model import Ridge
ridge=Ridge()
ridge.fit(x_train,y_train)
```

Out[167]:

```
Ridge(alpha=1.0, copy_X=True, fit_intercept=True, max_iter=None,
   normalize=False, random_state=None, solver='auto', tol=0.001)
```

Model Prediction

In [168]:

```
P_ridge=ridge.predict(x_test)
```

Evaluting the model

In [169]:

```
ridge_score=r2_score(y_test,P_ridge)*100
print(ridge_score)
```

```
85.23246723113603
```

6.6 Elastic net Regression

Training the model

In [170]:

```
from sklearn.linear_model import ElasticNet
elastic=ElasticNet()
elastic.fit(x_train,y_train)
```

Out[170]:

```
ElasticNet(alpha=1.0, copy_X=True, fit_intercept=True, l1_ratio=0.5,
      max_iter=1000, normalize=False, positive=False, precompute=False,
      random_state=None, selection='cyclic', tol=0.0001, warm_start=False)
```

Model Prediction

In [171]:

```
P_elastic=elastic.predict(x_test)
```

Evaluting the model

In [172]:

```
elastic_score=r2_score(y_test,P_elastic)*100
print(elastic_score)
```

```
81.2184476795696
```

7. Model Selection

Score comparison

In [173]:

```
Score=pd.DataFrame({"Model_name":["Linear Regression","Support Vector Regression", "Rand
om Forest Regression","Lasso Regression",
                              "Ridge Regression","Elastic Net Regression"],
                   "Accuracy_score":[lin_score,svr_score,forest_score,lasso_score,ridge_
score,elastic_score]})
```

In [174]:

```
Score
```

Out[174]:

	Model_name	Accuracy_score
0	Linear Regression	85.219421
1	Support Vector Regression	87.037501
2	Random Forest Regression	91.063429
3	Lasso Regression	81.083472
4	Ridge Regression	85.232467
5	Elastic Net Regression	81.218448

As you can see from above data "Random Forest Regressor" Given the highest Accuracy score

Feature Importance

In [177]:

```
Feature_importance=pd.DataFrame(randomforest.feature_importances_,index=["3","4","5","6",
"8","Europe","Japan","USA","displacement","horsepower","weight","acceleration","model_yea
r","brand"])
Feature_importance[8:]
```

Out[177]:

	0
displacement	0.325921
horsepower	0.184914
weight	0.215085

acceleration	0.0222900
model_year	0.133949
brand	0.011489

We found that Fuel consumption of a car is mostly affected by Displacement and Weight of the car

In [178]:

```
sample=pd.DataFrame({"Actual mpg":y_test,
                     "Predicted mpg":np.round(P_forest,2)})
```

In [179]:

```
sample
```

Out[179]:

	Actual mpg	Predicted mpg
0	26.0	24.59
1	17.5	19.80
2	46.6	38.22
3	12.0	12.59
4	13.0	12.98
5	21.5	25.30
6	44.0	38.41
7	23.8	22.83
8	26.0	27.41
9	23.0	24.34
10	19.8	22.36
11	26.4	24.12
12	32.3	34.83
13	31.0	31.49
14	16.9	16.24
15	13.0	14.72
16	15.0	17.97
17	15.0	18.16
18	28.0	23.12
19	26.0	25.04
20	13.0	12.94
21	17.6	22.13
22	30.5	30.62
23	18.2	17.68
24	16.0	14.90
25	33.0	31.64
26	26.5	25.51
27	27.2	31.42
28	15.5	17.60
29	14.0	11.59
...	...	...
90	31.0	27.74

	Actual mpg	Predicted mpg
91	[illegible]	[illegible]
92	20.0	19.51
93	44.3	41.47
94	25.4	27.41
95	15.0	14.06
96	14.0	12.88
97	12.0	12.19
98	26.6	20.91
99	15.0	14.12
100	38.0	36.03
101	15.0	14.56
102	34.5	34.27
103	31.0	36.46
104	16.5	18.28
105	29.0	30.17
106	18.0	20.19
107	36.0	35.18
108	19.0	18.62
109	16.0	13.45
110	21.0	21.32
111	15.0	15.77
112	29.0	28.74
113	17.5	16.87
114	17.7	18.94
115	26.8	26.05
116	29.0	30.82
117	30.0	31.27
118	25.0	25.47
119	12.0	13.20

120 rows × 2 columns

AS you can see Our model is working good its predicted Mpg is Nearly to the Actual Mpg

Save Mpg sample as a csv file

In [180]:

```
sample.to_csv("Model Prediction Sample.csv",index=False)
```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```
```