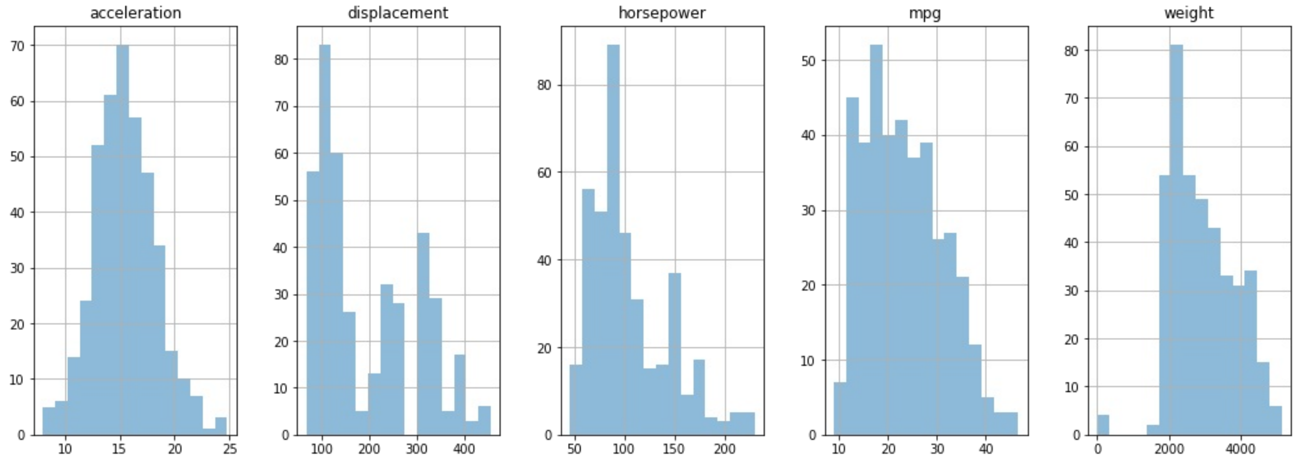
In [1]:

```
%matplotlib inline
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import plotly.graph_objects as go
import statistics
from sklearn.impute import KNNImputer
import seaborn as sns

adf = pd.read_csv('auto-mpg.csv')

numerical =['mpg','displacement','horsepower','acceleration','weight']
categorical =['cylinders','year','origin','carname']

print("Data Quality Report".center(100));
print("-------------------".center(100));

print('\033[1m' + 'ABT for Auto-mpg dataset');
#display(adf)


print('\033[1m' + 'Numerical Features');
display(adf[numerical].describe())


fig, ax=plt.subplots(1,5,figsize=(18,6))
adf_for_hist=adf[numerical]
adf_for_hist.hist(bins=15, alpha=0.5, ax=ax)
plt.show();


print('\033[1m' + 'Categorical Features');
display(adf[categorical].describe())



sns.countplot(adf['cylinders']);
plt.title('No of Cylinders');
plt.show();

sns.countplot(adf['year']);
plt.title('Year')
plt.show();

plt.title('Origin');
sns.countplot(adf['origin']);
plt.show();

#For handling outliers and missing values
#Finding the outlier

Q1 = adf.quantile(0.25)
Q3 = adf.quantile(0.75)
IQR = Q3 - Q1

#Identifying lower and upper bound outliers for attributes

Lower_Fence = Q1 - (1.5 * IQR)

Upper_Fence = Q3 + (1.5 * IQR)

#Removing the outlier

adf_out = adf[~((adf < (Q1 - 1.5 * IQR)) |(adf > (Q3 + 1.5 * IQR))).any(axis=1)]

#impute missing values
knn_imputer = KNNImputer(n_neighbors=3)
impute_copy = adf[['mpg', 'cylinders', 'displacement', 'horsepower', 'weight','acceleration','year','origin']].co
py()

df_filled = knn_imputer.fit_transform(impute_copy)

#Converting to Dataframe
new_df=pd.DataFrame(df_filled)
new_df.columns=['mpg', 'cylinders', 'displacement', 'horsepower', 'weight','acceleration','year','origin']

print('\033[1m' + 'Data Quality table after removing outliers and imputing missing values');

print('\033[1m' + 'Continuous Features');
```

```
fig = go.Figure(data=[go.Table(header=dict(values=['Feature', 'Count', '%Missing Value','Cardinality','Min','1st
Qrt.','Median','3rd Qrt.','Max','Std.dev']),
                cells=dict(values=[['mpg','displacement','horsepower','weight','acceleration'], [new_df['mpg'].c
ount(), new_df['displacement'].count(), new_df['horsepower'].count(), new_df['acceleration'].count(),new_df['weig
ht'].count()],[new_df['mpg'].isnull().sum(),new_df['displacement'].isnull().sum(),new_df['horsepower'].isnull().s
um(),new_df['weight'].isnull().sum(),new_df['acceleration'].isnull().sum()],[new_df['mpg'].unique().size,new_df['
displacement'].unique().size,new_df['horsepower'].unique().size,new_df['weight'].unique().size,new_df['accelerati
on'].unique().size],[new_df['mpg'].min(),new_df['displacement'].min(),new_df['horsepower'].min(),new_df['weight']
.min(),new_df['acceleration'].min()],[new_df['mpg'].quantile(0.25),new_df['displacement'].quantile(0.25),new_df['
horsepower'].quantile(0.25),new_df['weight'].quantile(0.25),new_df['acceleration'].quantile(0.25)],[new_df['mpg']
.median(),new_df['displacement'].median(),new_df['horsepower'].median(),new_df['weight'].median(),new_df['acceler
ation'].median()],[new_df['mpg'].quantile(0.75),new_df['displacement'].quantile(0.75),new_df['horsepower'].quanti
le(0.75),new_df['weight'].quantile(0.75),new_df['acceleration'].quantile(0.75)],[new_df['mpg'].max(),new_df['disp
lacement'].max(),new_df['horsepower'].max(),new_df['weight'].max(),new_df['acceleration'].max()],[new_df['mpg'].s
td(),new_df['displacement'].std(),new_df['horsepower'].std(),new_df['weight'].std(),new_df['acceleration'].std()]
]))
                    ])
fig.update_layout(width=1500, height=400)
fig.show()


print('\033[1m' + 'Categorical Features');

fig1 = go.Figure(data=[go.Table(header=dict(values=['Feature', 'Count', '%Missing Value','Cardinality','Mode']),
                cells=dict(values=[['Cylinders','Origin','Year'], [new_df['cylinders'].count(), new_df['origin']
.count(), new_df['year'].count()],[new_df['cylinders'].isnull().sum(),new_df['origin'].isnull().sum(),new_df['yea
r'].isnull().sum()],[new_df['cylinders'].unique().size,new_df['origin'].unique().size,new_df['year'].unique().siz
e],[statistics.mode(adf['cylinders']),statistics.mode(adf['origin']),statistics.mode(adf['year'])]]))
                    ])

fig1.show()
```

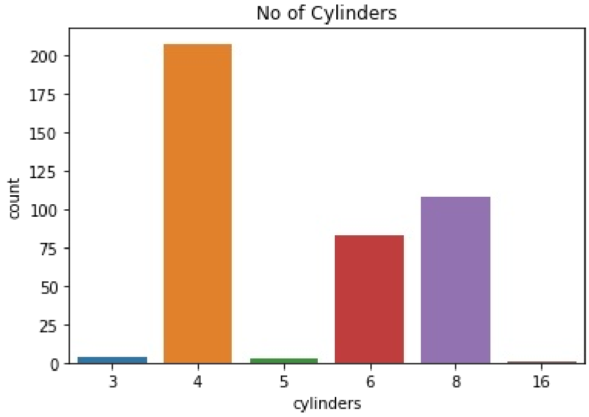
```
                                          Data Quality Report
                                          -------------------

ABT for Auto-mpg dataset
Numerical Features
```

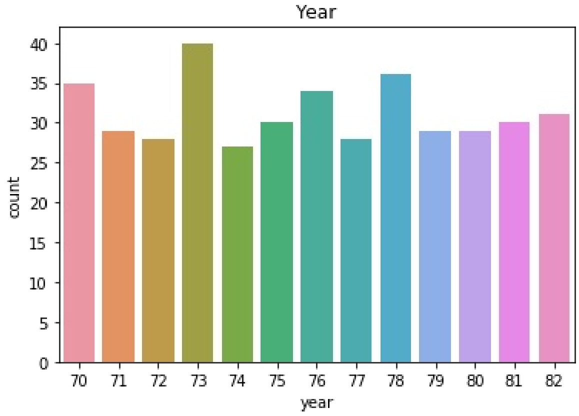
	mpg	displacement	horsepower	acceleration	weight
count	398.000000	406.000000	400.000000	406.000000	406.000000
mean	23.514573	194.779557	105.082500	15.519704	2952.305419
std	7.815984	104.922458	38.768779	2.803359	891.587329
min	9.000000	68.000000	46.000000	8.000000	19.000000
25%	17.500000	105.000000	75.750000	13.700000	2220.000000
50%	23.000000	151.000000	95.000000	15.500000	2811.000000
75%	29.000000	302.000000	130.000000	17.175000	3612.000000
max	46.600000	455.000000	230.000000	24.800000	5140.000000

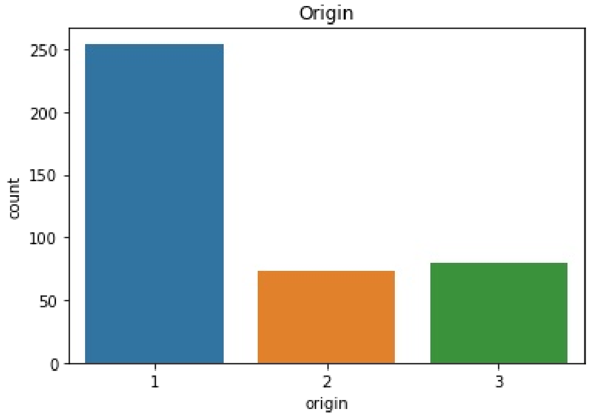


```
Categorical Features
```

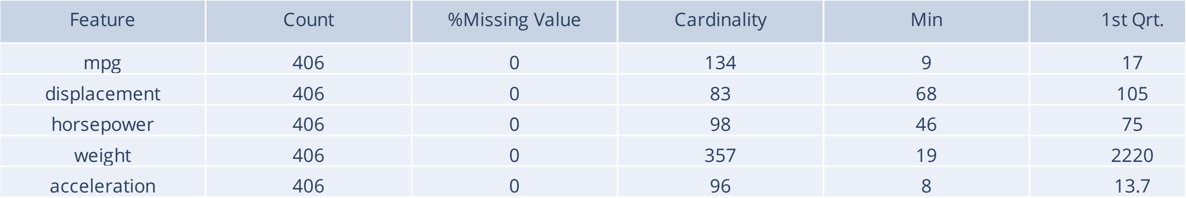
	cylinders	year	origin
count	406.000000	406.000000	406.000000
mean	5.500000	75.921182	1.568966
std	1.789889	3.748737	0.797479
min	3.000000	70.000000	1.000000
25%	4.000000	73.000000	1.000000
50%	4.000000	76.000000	1.000000
75%	8.000000	79.000000	2.000000
max	16.000000	82.000000	3.000000





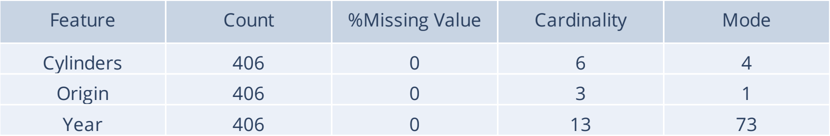


Data Quality table after removing outliers and imputing missing values
Continuous Features



Feature	Count	%Missing Value	Cardinality	Min	1st Qrt.
mpg	406	0	134	9	17
displacement	406	0	83	68	105
horsepower	406	0	98	46	75
weight	406	0	357	19	2220
acceleration	406	0	96	8	13.7

Categorical Features



Feature	Count	%Missing Value	Cardinality	Mode
Cylinders	406	0	6	4
Origin	406	0	3	1
Year	406	0	13	73

In [10]:

In [ ]: