## Data Description

The dataset contains information about 81 cars, including various attributes related to their performance and characteristics. the vehicle.

1. MPG (Average Miles per Gallon): This column represents the average number of miles a car can travel per gallon of fuel. It provides an indication of the fuel efficiency of the vehicles.
2. HP (Engine Horsepower): This column provides the horsepower of the car's engine, which is a measure of its power and performance capabilities.
3. VOL (Cubic Feet of Cab Space): This column denotes the volume of the car's cab space in cubic feet. It reflects the size and spaciousness of the vehicle's interior.
4. SP (Top Speed in Miles per Hour): This column specifies the top speed of the car, indicating the maximum speed the vehicle can achieve in miles per hour.
5. WT (Vehicle Weight in 100 lbs.): This column represents the weight of the car in units of 100 lbs. It provides an insight into the heaviness or lightness of

In [2]:
```
import pandas as pd
```

In [17]:
```
df= pd.read_csv ("Table 11.7 Passenger Car Milage Data.csv ")
df
```

Out[17]:

| | Observation | MPG | SP | HP | VOL | WT |
|---|---|---|---|---|---|---|
| 0 | 1 | 65.4 | 96 | 49 | 89 | 17.5 |
| 1 | 2 | 56.0 | 97 | 55 | 92 | 20.0 |
| 2 | 3 | 55.9 | 97 | 55 | 92 | 20.0 |
| 3 | 4 | 49.0 | 105 | 70 | 92 | 20.0 |
| 4 | 5 | 46.5 | 96 | 53 | 92 | 20.0 |
| ... | ... | ... | ... | ... | ... | ... |
| 76 | 77 | 18.1 | 165 | 322 | 50 | 45.0 |
| 77 | 78 | 17.2 | 140 | 238 | 115 | 45.0 |
| 78 | 79 | 17.0 | 147 | 263 | 50 | 45.0 |
| 79 | 80 | 16.7 | 157 | 295 | 119 | 45.0 |
| 80 | 81 | 13.2 | 130 | 236 | 107 | 55.0 |

81 rows × 6 columns

In [18]:
```
df.head()
```

Out[18]:

| | Observation | MPG | SP | HP | VOL | WT |
|---|---|---|---|---|---|---|
| 0 | 1 | 65.4 | 96 | 49 | 89 | 17.5 |
| 1 | 2 | 56.0 | 97 | 55 | 92 | 20.0 |
| 2 | 3 | 55.9 | 97 | 55 | 92 | 20.0 |
| 3 | 4 | 49.0 | 105 | 70 | 92 | 20.0 |
| 4 | 5 | 46.5 | 96 | 53 | 92 | 20.0 |

## Estimating the parameters of the model

In [44]:
```
import statsmodels.formula.api as smf
reg = smf.ols('MPG ~ SP+HP+WT', data=df).fit()
print(reg.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    MPG   R-squared:                       0.883
Model:                            OLS   Adj. R-squared:                  0.878
Method:                 Least Squares   F-statistic:                     193.5
Date:                Tue, 09 Jul 2024   Prob (F-statistic):           9.28e-36
Time:                        18:55:11   Log-Likelihood:                -214.54
No. Observations:                  81   AIC:                             437.1
Df Residuals:                      77   BIC:                             446.7
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    189.9597     22.529      8.432      0.000     145.099     234.820
SP            -1.2717      0.233     -5.455      0.000      -1.736      -0.808
HP             0.3904      0.076      5.121      0.000       0.239       0.542
WT            -1.9033      0.186    -10.259      0.000      -2.273      -1.534
==============================================================================
Omnibus:                       20.525   Durbin-Watson:                   1.024
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               29.909
Skew:                           1.068   Prob(JB):                     3.20e-07
Kurtosis:                       5.074   Cond. No.                     1.00e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large,  1e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

MPG is positively related to HP and negatively related to SP and WT. R squared value is 0.883, suggesting that the model explains 88.3% of the variance in MPG. The model showcases strong signs of multicollinearity. Since this data is a cross-sectional data involving a diversity of cars, we can also expect heteroscedasticity.

## Test for Multicollinearity

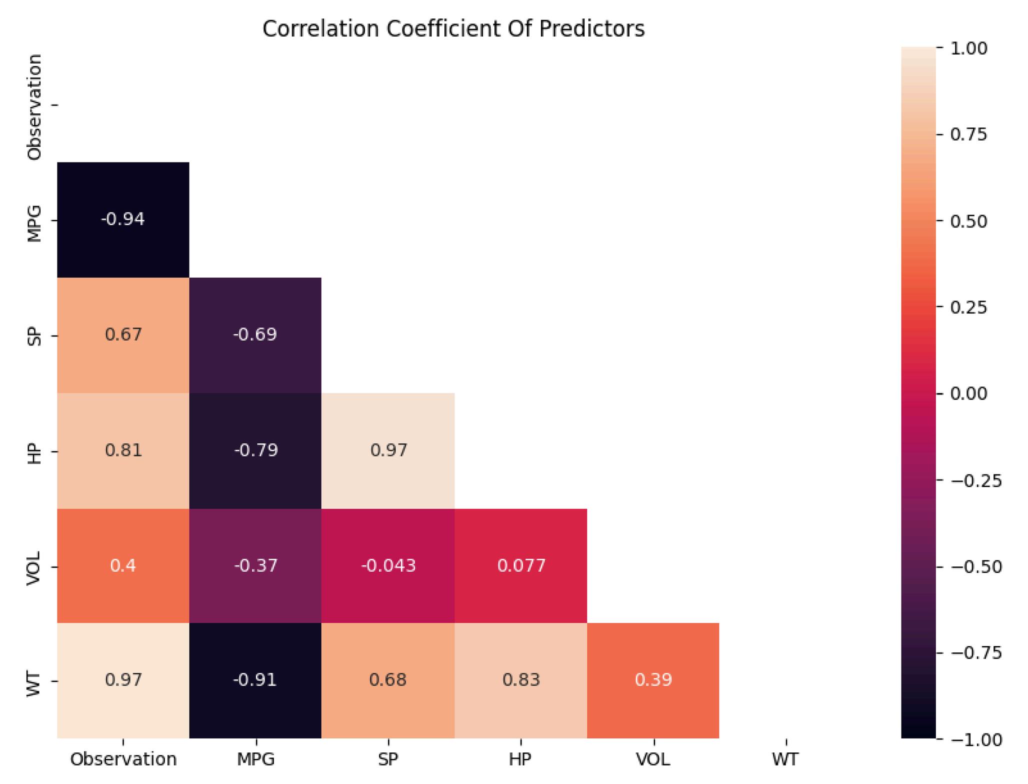
In [22]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor
df1 = df[['SP','HP','WT']]
vif_data = pd.DataFrame()
vif_data["feature"] = df1.columns

 # calculating VIF for each feature
vif_data["VIF"] = [variance_inflation_factor(df1.values, i)
for i in range(len(df1.columns))]
print(vif_data)
```

```
  feature        VIF
0      SP  26.898818
1      HP  13.665893
2      WT  38.926431
```

VIF or variance inflation factor is an indicator of multicollinearity which has been used by a number of authors. If the VIF exceeds 10, then variables are highly collinear. In our case, the variables are indeed highly collinear and hence, multicollinearity is present.

Correlation Coefficient of Predictors

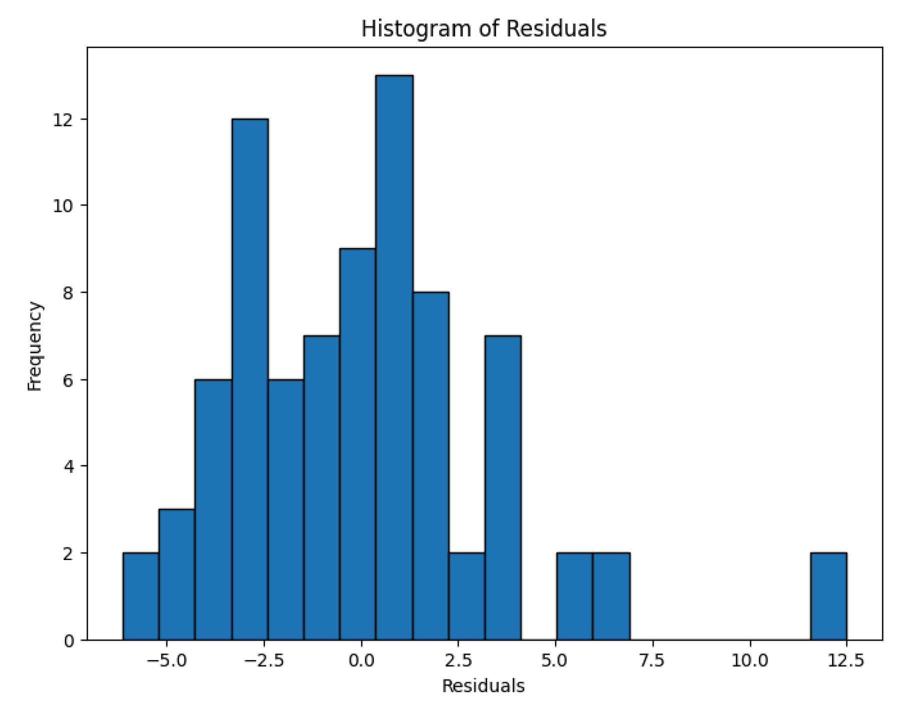
In [24]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
 # set figure size
plt.figure(figsize=(10,7))
mask = np.triu(np.ones_like(df.corr(), dtype=bool))
 # generate heatmap
sns.heatmap(df.corr(), annot=True, mask=mask, vmin=-1, vmax=1)
plt.title('Correlation Coefficient Of Predictors')
plt.show()
```



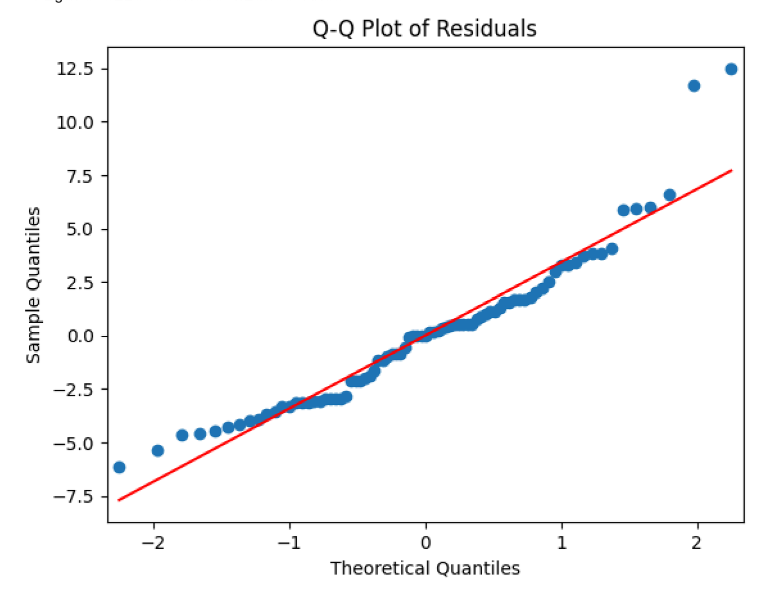

## Test for Heteroscedasticity

The test that we are using, that is, BPG, is an asymptotic large sample test and is sensitive to the assumption of normal distribution of disturbances

In [27]:
```
import statsmodels.api as sm
X = df[['SP', 'HP', 'WT']].values
y = df['MPG'].values
 # Perform multiple regression to obtain predicted values
regression_model = sm.OLS(y, sm.add_constant(X))
regression_results = regression_model.fit()
y_pred = regression_results.predict(sm.add_constant(X))
 # Calculate the residuals
residuals = y - y_pred
 # Plot histogram of residuals
plt.figure(figsize=(8, 6))
plt.hist(residuals, bins=20, edgecolor='black')
plt.xlabel('Residuals')
plt.ylabel('Frequency')
plt.title('Histogram of Residuals')
plt.show()
 # Plot Q-Q plot of residuals
plt.figure(figsize=(8, 6))
sm.qqplot(residuals, line='s')
plt.xlabel('Theoretical Quantiles')
plt.ylabel('Sample Quantiles')
plt.title('Q-Q Plot of Residuals')
plt.show()
```




<Figure size 800x600 with 0 Axes>




We can assume that the residuals follow normal distribution, and hence, normality assumption is fulfilled.

## BPG Test

In [29]:
```
import statsmodels.formula.api as smf
from statsmodels.compat import lzip
import statsmodels.stats.api as sms
 # Fit the regression model
fit = smf.ols('MPG ~ SP+HP+WT',
data=df).fit()
 # Conduct the Breusch-Pagan test
names = ['Lagrange multiplier statistic', 'p-value', 'f-value', 'f p-value']
 # Get the test result
test_result = sms.het_breuschpagan(fit.resid, fit.model.exog)
lzip(names, test_result)For this test, the null and alternate hypothesis are:
 H0: There is no heteroscedasticity present
 H1: There is presence of heteroscedasticity
```

Out[29]:
```
[('Lagrange multiplier statistic', np.float64(25.491410654647773)),
 ('p-value', np.float64(1.218650354151728e-05)),
 ('f-value', np.float64(11.786996352317546)),
 ('f p-value', np.float64(1.9545433210143356e-06))]
```

For this test, the null and alternate hypothesis are: H0: There is no heteroscedasticity present H1: There is presence of heteroscedasticity

- The p value obtained from the test show a highly significant result confirming the presence of heteroscedasticity with level of significance equal to 0.05

## Remedial Measures for Multicollinearity

Multicollinearity is more of a data deficiency problem or micro-numerosity and sometimes we have no choice over the data we have available for empirical analysis. Multicollinearity is often said to be God’s will, and hence, not a problem with OLS or statistical technique in general.

- For this data, we decide to do nothing

## Remedial Measures for Heteroscedasticity

Log transformation of the variables

In [37]:
```
df2 = pd.read_csv ("Log Transformed_mayank.csv")
df2
```

Out[37]:

| | Observation | MPG | SP | HP | VOL | WT | Log_MPG | Log_SP | Log_HP | Log_VOL | Log_WT |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 65.4 | 96 | 49 | 89 | 17.5 | 1.815578 | 1.982271 | 1.690196 | 1.949390 | 1.243038 |
| 1 | 2 | 56.0 | 97 | 55 | 92 | 20.0 | 1.748188 | 1.986772 | 1.740363 | 1.963788 | 1.301030 |
| 2 | 3 | 55.9 | 97 | 55 | 92 | 20.0 | 1.747412 | 1.986772 | 1.740363 | 1.963788 | 1.301030 |
| 3 | 4 | 49.0 | 105 | 70 | 92 | 20.0 | 1.690196 | 2.021189 | 1.845098 | 1.963788 | 1.301030 |
| 4 | 5 | 46.5 | 96 | 53 | 92 | 20.0 | 1.667453 | 1.982271 | 1.724276 | 1.963788 | 1.301030 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 76 | 77 | 18.1 | 165 | 322 | 50 | 45.0 | 1.257679 | 2.217484 | 2.507856 | 1.698970 | 1.653213 |
| 77 | 78 | 17.2 | 140 | 238 | 115 | 45.0 | 1.235528 | 2.146128 | 2.376577 | 2.060698 | 1.653213 |
| 78 | 79 | 17.0 | 147 | 263 | 50 | 45.0 | 1.230449 | 2.167317 | 2.419956 | 1.698970 | 1.653213 |
| 79 | 80 | 16.7 | 157 | 295 | 119 | 45.0 | 1.222716 | 2.195900 | 2.469822 | 2.075547 | 1.653213 |
| 80 | 81 | 13.2 | 130 | 236 | 107 | 55.0 | 1.120574 | 2.113943 | 2.372912 | 2.029384 | 1.740363 |

81 rows × 11 columns

In [38]:
```
df2.head()
```

Out[38]:

| | Observation | MPG | SP | HP | VOL | WT | Log_MPG | Log_SP | Log_HP | Log_VOL | Log_WT |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 65.4 | 96 | 49 | 89 | 17.5 | 1.815578 | 1.982271 | 1.690196 | 1.949390 | 1.243038 |
| 1 | 2 | 56.0 | 97 | 55 | 92 | 20.0 | 1.748188 | 1.986772 | 1.740363 | 1.963788 | 1.301030 |
| 2 | 3 | 55.9 | 97 | 55 | 92 | 20.0 | 1.747412 | 1.986772 | 1.740363 | 1.963788 | 1.301030 |
| 3 | 4 | 49.0 | 105 | 70 | 92 | 20.0 | 1.690196 | 2.021189 | 1.845098 | 1.963788 | 1.301030 |
| 4 | 5 | 46.5 | 96 | 53 | 92 | 20.0 | 1.667453 | 1.982271 | 1.724276 | 1.963788 | 1.301030 |

BPG test for transformed variables

In [41]:
```
 # Fit the regression model
fit = smf.ols('Log_MPG ~ Log_SP+Log_HP+Log_WT',
data=df2).fit()
 # Conduct the Breusch-Pagan test
names = ['Lagrange multiplier statistic', 'p-value', 'f-value', 'f p-value']
 # Get the test result
test_result = sms.het_breuschpagan(fit.resid, fit.model.exog)
lzip(names, test_result)
```

Out[41]:
```
[('Lagrange multiplier statistic', np.float64(5.321352570703248)),
 ('p-value', np.float64(0.14972284042055495)),
 ('f-value', np.float64(1.8047545415720079)),
 ('f p-value', np.float64(0.15332450852500787))]
```

We are successful in removing heteroscedasticity from our model using log transformation with an insignificant p value

## Estimating the parameters for new model

In [45]:
```
import statsmodels.formula.api as smf
reg = smf.ols('Log_MPG ~ Log_SP+Log_HP+Log_WT', data=df2).fit()
print(reg.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                Log_MPG   R-squared:                       0.929
Model:                            OLS   Adj. R-squared:                  0.926
Method:                 Least Squares   F-statistic:                     336.6
Date:                Tue, 09 Jul 2024   Prob (F-statistic):           3.75e-44
Time:                        19:54:26   Log-Likelihood:                 156.23
No. Observations:                  81   AIC:                            -304.5
Df Residuals:                      77   BIC:                            -294.9
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      2.2010      1.104      1.993      0.050       0.002       4.400
Log_SP         0.5732      0.675      0.849      0.398      -0.771       1.917
Log_HP        -0.5052      0.292     -1.731      0.088      -1.087       0.076
Log_WT        -0.5687      0.220     -2.580      0.012      -1.008      -0.130
==============================================================================
Omnibus:                        2.939   Durbin-Watson:                   0.906
Prob(Omnibus):                  0.230   Jarque-Bera (JB):                2.538
Skew:                           0.165   Prob(JB):                        0.281
Kurtosis:                       3.802   Cond. No.                     1.14e+03
==============================================================================

Notes:
```

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.14e+03. This might indicate that there are
strong multicollinearity or other numerical problems.