In [6]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from statsmodels.graphics.regressionplots import influence_plot
import statsmodels.formula.api as smf
import numpy as np
```

In [7]:
```
#Read the data
cars = pd.read_csv("Cars.csv")
cars.head()
```

Out[7]:

| | HP | MPG | VOL | SP | WT |
|---|---|---|---|---|---|
| **0** | 49 | 53.700681 | 89 | 104.185353 | 28.762059 |
| **1** | 55 | 50.013401 | 92 | 105.461264 | 30.466833 |
| **2** | 55 | 50.013401 | 92 | 105.461264 | 30.193597 |
| **3** | 70 | 45.696322 | 92 | 113.461264 | 30.632114 |
| **4** | 53 | 50.504232 | 92 | 104.461264 | 29.889149 |

In [8]:
```
cars.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 81 entries, 0 to 80
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   HP      81 non-null     int64  
 1   MPG     81 non-null     float64
 2   VOL     81 non-null     int64  
 3   SP      81 non-null     float64
 4   WT      81 non-null     float64
dtypes: float64(3), int64(2)
memory usage: 3.3 KB
```

In [9]:
```
#check for missing values
cars.isna().sum()
```

Out[9]:
```
HP     0
MPG    0
VOL    0
SP     0
WT     0
dtype: int64
```

# Correlation Matrix

In [10]:
```
cars.corr()
```

Out[10]:

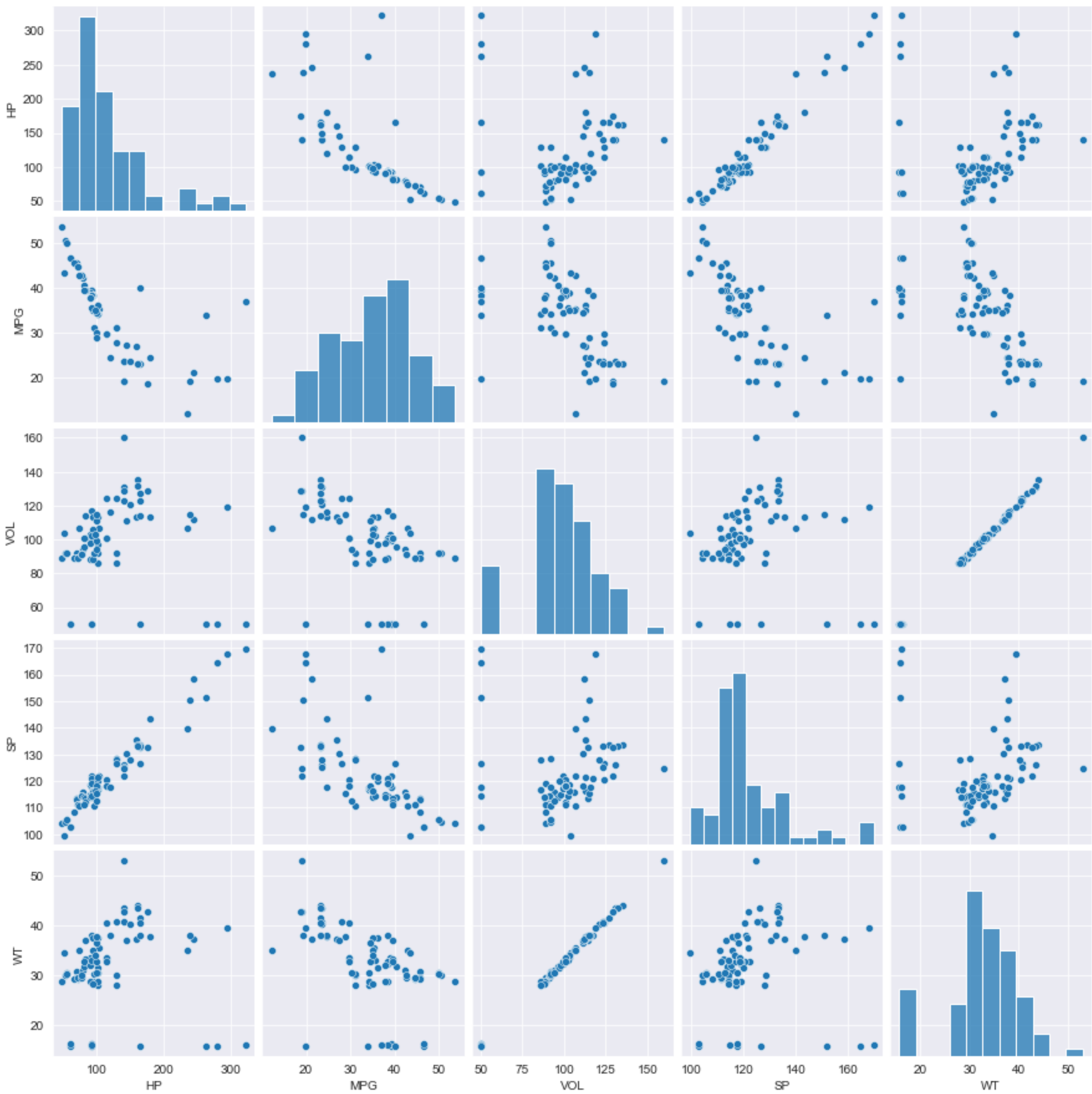
| | HP | MPG | VOL | SP | WT |
|---|---|---|---|---|---|
| **HP** | 1.000000 | -0.725038 | 0.077459 | 0.973848 | 0.076513 |
| **MPG** | -0.725038 | 1.000000 | -0.529057 | -0.687125 | -0.526759 |
| **VOL** | 0.077459 | -0.529057 | 1.000000 | 0.102170 | 0.999203 |
| **SP** | 0.973848 | -0.687125 | 0.102170 | 1.000000 | 0.102439 |
| **WT** | 0.076513 | -0.526759 | 0.999203 | 0.102439 | 1.000000 |

# Scatterplot between variables along with histograms

In [11]:

```
#Format the plot background and scatter plots for all the variables
sns.set_style(style='darkgrid')
sns.pairplot(cars)
```

Out[11]: <seaborn.axisgrid.PairGrid at 0x249bf3a9d00>




# Preparing a model

In [12]:
```
#Build model
import statsmodels.formula.api as smf
model = smf.ols('MPG~WT+VOL+SP+HP',data=cars).fit()
```

In [13]:
```
model.rsquared
```

Out[13]:
```
0.7705372737359844
```

In [14]:
```
model.summary()
```

Out[14]:

OLS Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | MPG | **R-squared:** | 0.771 |
| **Model:** | OLS | **Adj. R-squared:** | 0.758 |
| **Method:** | Least Squares | **F-statistic:** | 63.80 |
| **Date:** | Tue, 29 Nov 2022 | **Prob (F-statistic):** | 1.54e-23 |
| **Time:** | 13:08:50 | **Log-Likelihood:** | -233.96 |
| **No. Observations:** | 81 | **AIC:** | 477.9 |
| **Df Residuals:** | 76 | **BIC:** | 489.9 |
| **Df Model:** | 4 | | |
| **Covariance Type:** | nonrobust | | |

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **Intercept** | 30.6773 | 14.900 | 2.059 | 0.043 | 1.001 | 60.354 |
| **WT** | 0.4006 | 1.693 | 0.237 | 0.814 | -2.972 | 3.773 |
| **VOL** | -0.3361 | 0.569 | -0.591 | 0.556 | -1.469 | 0.796 |
| **SP** | 0.3956 | 0.158 | 2.500 | 0.015 | 0.080 | 0.711 |
| **HP** | -0.2054 | 0.039 | -5.239 | 0.000 | -0.284 | -0.127 |

| | | | |
|---|---|---|---|
| **Omnibus:** | 10.780 | **Durbin-Watson:** | 1.403 |
| **Prob(Omnibus):** | 0.005 | **Jarque-Bera (JB):** | 11.722 |
| **Skew:** | 0.707 | **Prob(JB):** | 0.00285 |
| **Kurtosis:** | 4.215 | **Cond. No.** | 6.09e+03 |

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 6.09e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

In [15]:
```
#Coefficients
model.params
```

Out[15]:
```
Intercept    30.677336
WT            0.400574
VOL          -0.336051
SP            0.395627
HP           -0.205444
dtype: float64
```

In [16]:
```
#t and p-Values
print(model.tvalues, '\n', model.pvalues)
```

```
Intercept    2.058841
WT           0.236541
VOL         -0.590970
SP           2.499880
HP          -5.238735
dtype: float64
 Intercept    0.042936
WT           0.813649
VOL          0.556294
SP           0.014579
HP           0.000001
dtype: float64
```

In [17]:
```
#R squared values
(model.rsquared,model.rsquared_adj)
```

Out[17]:
```
(0.7705372737359844, 0.7584602881431415)
```

# Simple Linear Regression Models

In [18]:
```
ml_v=smf.ols('MPG~VOL',data = cars).fit()
#t and p-Values
print(ml_v.tvalues, '\n', ml_v.pvalues)
```

```
Intercept    14.106056
VOL          -5.541400
dtype: float64
 Intercept    2.753815e-23
VOL          3.822819e-07
dtype: float64
```

In [19]:
```
ml_v.summary()
```

Out[19]:

OLS Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | MPG | **R-squared:** | 0.280 |
| **Model:** | OLS | **Adj. R-squared:** | 0.271 |
| **Method:** | Least Squares | **F-statistic:** | 30.71 |
| **Date:** | Tue, 29 Nov 2022 | **Prob (F-statistic):** | 3.82e-07 |
| **Time:** | 13:08:53 | **Log-Likelihood:** | -280.28 |
| **No. Observations:** | 81 | **AIC:** | 564.6 |
| **Df Residuals:** | 79 | **BIC:** | 569.4 |
| **Df Model:** | 1 | | |
| **Covariance Type:** | nonrobust | | |

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **Intercept** | 55.8171 | 3.957 | 14.106 | 0.000 | 47.941 | 63.693 |
| **VOL** | -0.2166 | 0.039 | -5.541 | 0.000 | -0.294 | -0.139 |

| | | | |
|---|---|---|---|
| **Omnibus:** | 2.691 | **Durbin-Watson:** | 0.566 |
| **Prob(Omnibus):** | 0.260 | **Jarque-Bera (JB):** | 1.997 |
| **Skew:** | -0.263 | **Prob(JB):** | 0.368 |
| **Kurtosis:** | 3.562 | **Cond. No.** | 462. |

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

In [20]:

```
ml_w=smf.ols('MPG~WT',data = cars).fit()
print(ml_w.tvalues, '\n', ml_w.pvalues)
```

```
Intercept    14.248923
WT           -5.508067
dtype: float64
 Intercept    1.550788e-23
WT           4.383467e-07
dtype: float64
```

In [21]:

```
ml_w.summary()
```

Out[21]:

OLS Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | MPG | **R-squared:** | 0.277 |
| **Model:** | OLS | **Adj. R-squared:** | 0.268 |
| **Method:** | Least Squares | **F-statistic:** | 30.34 |
| **Date:** | Tue, 29 Nov 2022 | **Prob (F-statistic):** | 4.38e-07 |
| **Time:** | 13:08:54 | **Log-Likelihood:** | -280.42 |
| **No. Observations:** | 81 | **AIC:** | 564.8 |
| **Df Residuals:** | 79 | **BIC:** | 569.6 |
| **Df Model:** | 1 | | |
| **Covariance Type:** | nonrobust | | |

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **Intercept** | 55.2296 | 3.876 | 14.249 | 0.000 | 47.514 | 62.945 |
| **WT** | -0.6420 | 0.117 | -5.508 | 0.000 | -0.874 | -0.410 |

| | | | |
|---|---|---|---|
| **Omnibus:** | 2.735 | **Durbin-Watson:** | 0.555 |
| **Prob(Omnibus):** | 0.255 | **Jarque-Bera (JB):** | 2.045 |
| **Skew:** | -0.263 | **Prob(JB):** | 0.360 |
| **Kurtosis:** | 3.573 | **Cond. No.** | 149. |

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

In [22]:

```
ml_wv=smf.ols('MPG~WT+VOL',data = cars).fit()
print(ml_wv.tvalues, '\n', ml_wv.pvalues)
```

```
Intercept    12.545736
WT            0.489876
VOL          -0.709604
dtype: float64
 Intercept    2.141975e-20
WT           6.255966e-01
VOL          4.800657e-01
dtype: float64
```

In [23]:

```
ml_wv.summary()
```

Out[23]:

OLS Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | MPG | **R-squared:** | 0.282 |
| **Model:** | OLS | **Adj. R-squared:** | 0.264 |
| **Method:** | Least Squares | **F-statistic:** | 15.33 |
| **Date:** | Tue, 29 Nov 2022 | **Prob (F-statistic):** | 2.43e-06 |
| **Time:** | 13:08:55 | **Log-Likelihood:** | -280.16 |
| **No. Observations:** | 81 | **AIC:** | 566.3 |
| **Df Residuals:** | 78 | **BIC:** | 573.5 |
| **Df Model:** | 2 | | |
| **Covariance Type:** | nonrobust | | |

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **Intercept** | 56.8847 | 4.534 | 12.546 | 0.000 | 47.858 | 65.912 |
| **WT** | 1.4349 | 2.929 | 0.490 | 0.626 | -4.397 | 7.266 |
| **VOL** | -0.6983 | 0.984 | -0.710 | 0.480 | -2.658 | 1.261 |

| | | | |
|---|---|---|---|
| **Omnibus:** | 2.405 | **Durbin-Watson:** | 0.591 |
| **Prob(Omnibus):** | 0.300 | **Jarque-Bera (JB):** | 1.712 |
| **Skew:** | -0.251 | **Prob(JB):** | 0.425 |
| **Kurtosis:** | 3.506 | **Cond. No.** | 597. |

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

# Calculating VIF

In [24]:
```
hp = smf.ols('HP~WT+VOL+SP',data = cars).fit().rsquared
```

In [25]:
```
vif_hp1 = 1/(1 - hp)
```

In [26]:
```
vif_hp1
```

Out[26]:
```
19.926588974998563
```

In [27]:
```
rsq_hp = smf.ols('HP~WT+VOL+SP',data=cars).fit().rsquared
vif_hp = 1/(1-rsq_hp) # 19

rsq_wt = smf.ols('WT~HP+VOL+SP',data=cars).fit().rsquared
vif_wt = 1/(1-rsq_wt) # 625

rsq_vol = smf.ols('VOL~WT+SP+HP',data=cars).fit().rsquared
vif_vol = 1/(1-rsq_vol) #  624

rsq_sp = smf.ols('SP~WT+VOL+HP',data=cars).fit().rsquared
vif_sp = 1/(1-rsq_sp) #  20
```

```
# Storing vif values in a data frame
d1 = {'Variables':['Hp','WT','VOL','SP'],'VIF':[vif_hp,vif_wt,vif_vol,vif_sp]}
Vif_frame = pd.DataFrame(d1)
Vif_frame
```

Out[27]:

| | Variables | VIF |
|---|---|---|
| **0** | Hp | 19.926589 |
| **1** | WT | 639.533818 |
| **2** | VOL | 638.806084 |
| **3** | SP | 20.007639 |

# Residual Analysis

## Test for Normality of Residuals (Q-Q Plot)

In [28]:
```
cars
```

Out[28]:

| | HP | MPG | VOL | SP | WT |
|---|---|---|---|---|---|
| **0** | 49 | 53.700681 | 89 | 104.185353 | 28.762059 |
| **1** | 55 | 50.013401 | 92 | 105.461264 | 30.466833 |
| **2** | 55 | 50.013401 | 92 | 105.461264 | 30.193597 |
| **3** | 70 | 45.696322 | 92 | 113.461264 | 30.632114 |
| **4** | 53 | 50.504232 | 92 | 104.461264 | 29.889149 |
| **...** | ... | ... | ... | ... | ... |
| **76** | 322 | 36.900000 | 50 | 169.598513 | 16.132947 |
| **77** | 238 | 19.197888 | 115 | 150.576579 | 37.923113 |
| **78** | 263 | 34.000000 | 50 | 151.598513 | 15.769625 |
| **79** | 295 | 19.833733 | 119 | 167.944460 | 39.423099 |
| **80** | 236 | 12.101263 | 107 | 139.840817 | 34.948615 |

81 rows × 5 columns

In [29]:
```
cars.iloc[:,[0,2,3,4]]
```

Out[29]:

| | HP | VOL | SP | WT |
|---|---|---|---|---|
| **0** | 49 | 89 | 104.185353 | 28.762059 |
| **1** | 55 | 92 | 105.461264 | 30.466833 |
| **2** | 55 | 92 | 105.461264 | 30.193597 |
| **3** | 70 | 92 | 113.461264 | 30.632114 |
| **4** | 53 | 92 | 104.461264 | 29.889149 |
| **...** | ... | ... | ... | ... |
| **76** | 322 | 50 | 169.598513 | 16.132947 |
| **77** | 238 | 115 | 150.576579 | 37.923113 |
| **78** | 263 | 50 | 151.598513 | 15.769625 |
| **79** | 295 | 119 | 167.944460 | 39.423099 |
| **80** | 236 | 107 | 139.840817 | 34.948615 |

81 rows × 4 columns

In [30]:

```
cars["MPG"]-model.predict(cars.iloc[:,[0,2,3,4]])
```

Out[30]:

```
0     10.258747
1      7.624608
2      7.734060
3      3.157963
4      8.331584
        ...
76    15.617904
77     1.298838
78     7.863547
79     7.517122
80    -3.458218
Length: 81, dtype: float64
```

In [31]:

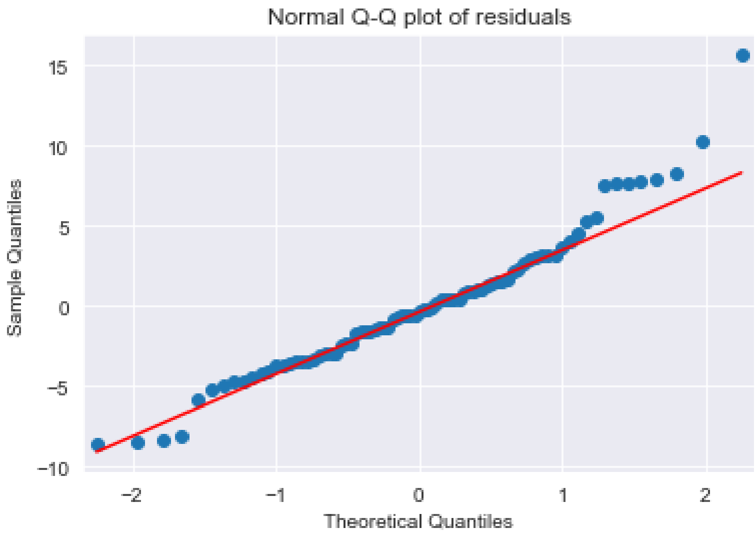
```
model.resid
```

Out[31]:

```
0     10.258747
1      7.624608
2      7.734060
3      3.157963
4      8.331584
        ...
76    15.617904
77     1.298838
78     7.863547
79     7.517122
80    -3.458218
Length: 81, dtype: float64
```

In [32]:

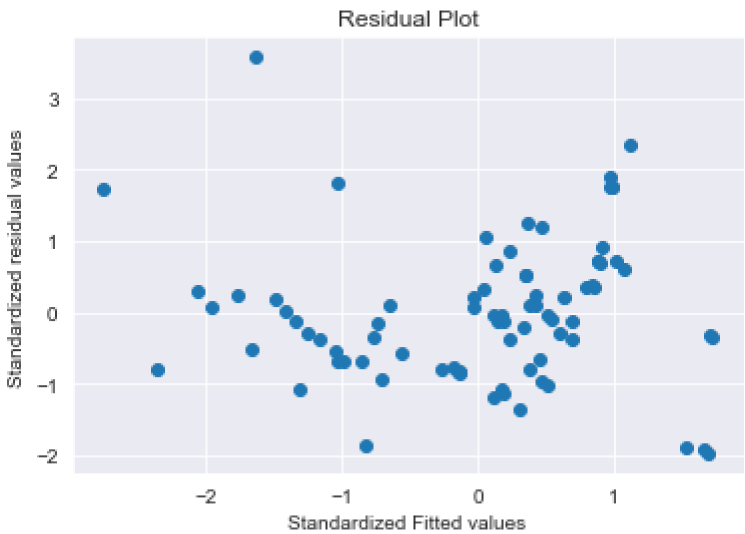
```
import statsmodels.api as sm
qqplot=sm.qqplot(model.resid,line='q') # line = 45 to draw the diagnoal line
plt.title("Normal Q-Q plot of residuals")
plt.show()
```




# Residual Plot for Homoscedasticity

In [33]:
```
def get_standardized_values( vals ):
    return (vals - vals.mean())/vals.std()
```

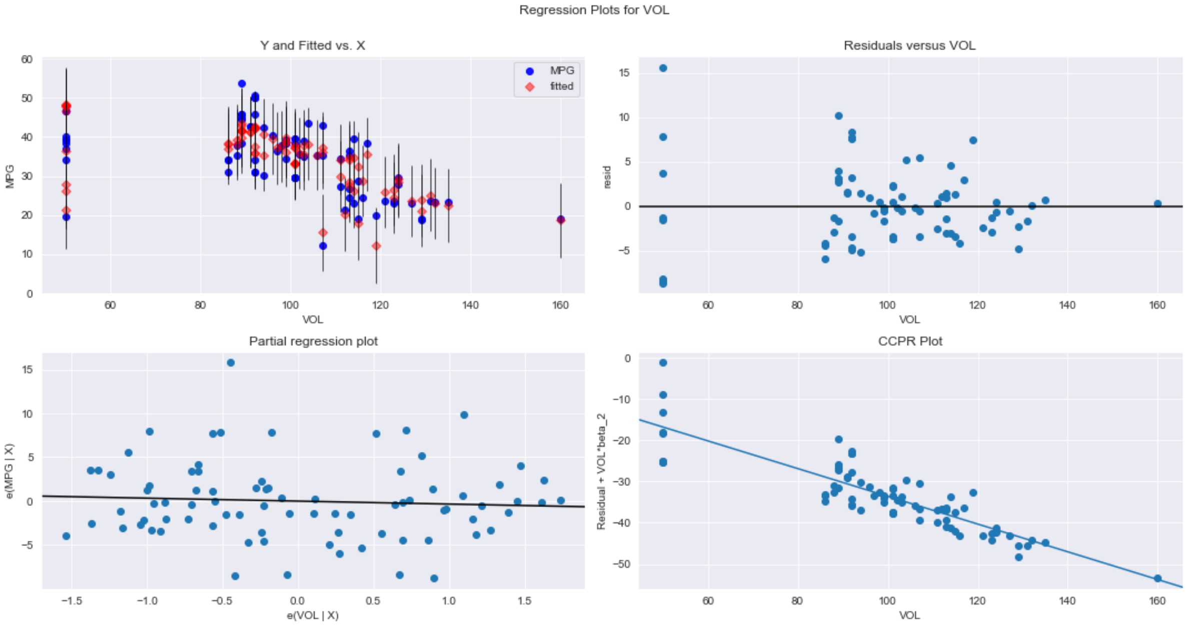
In [34]:
```
plt.scatter(get_standardized_values(model.fittedvalues),
            get_standardized_values(model.resid))

plt.title('Residual Plot')
plt.xlabel('Standardized Fitted values')
plt.ylabel('Standardized residual values')
plt.show()
```




# Residual Vs Regressors

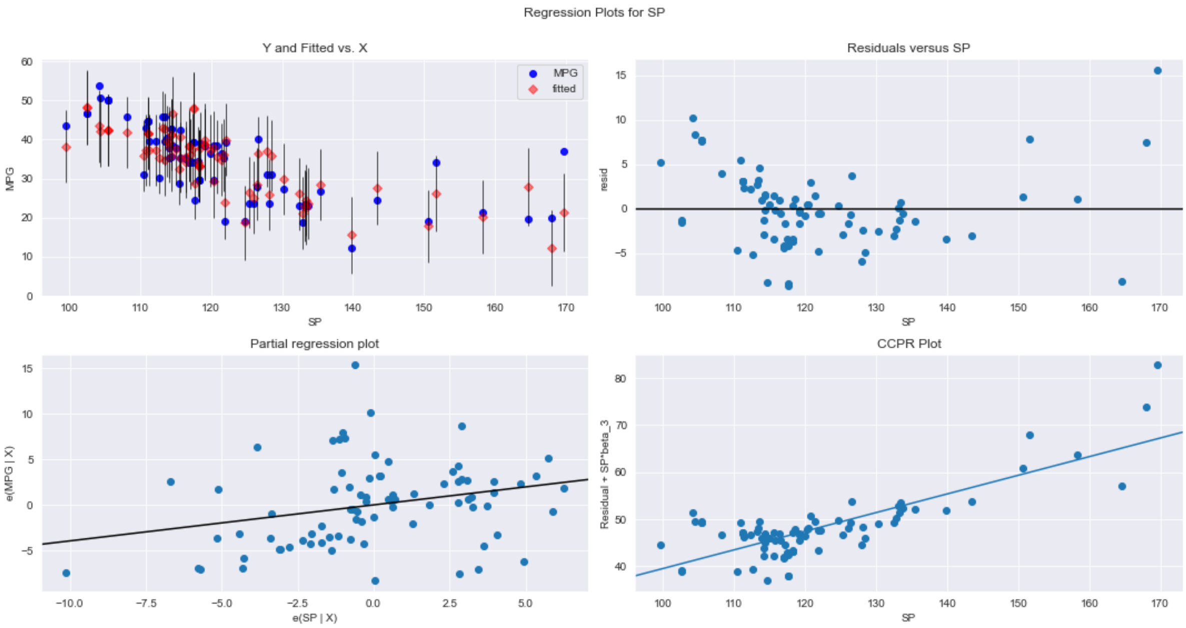
In [35]:
```
fig = plt.figure(figsize=(15,8))
fig = sm.graphics.plot_regress_exog(model, "VOL", fig=fig)
plt.show()
```

```
eval_env: 1
```



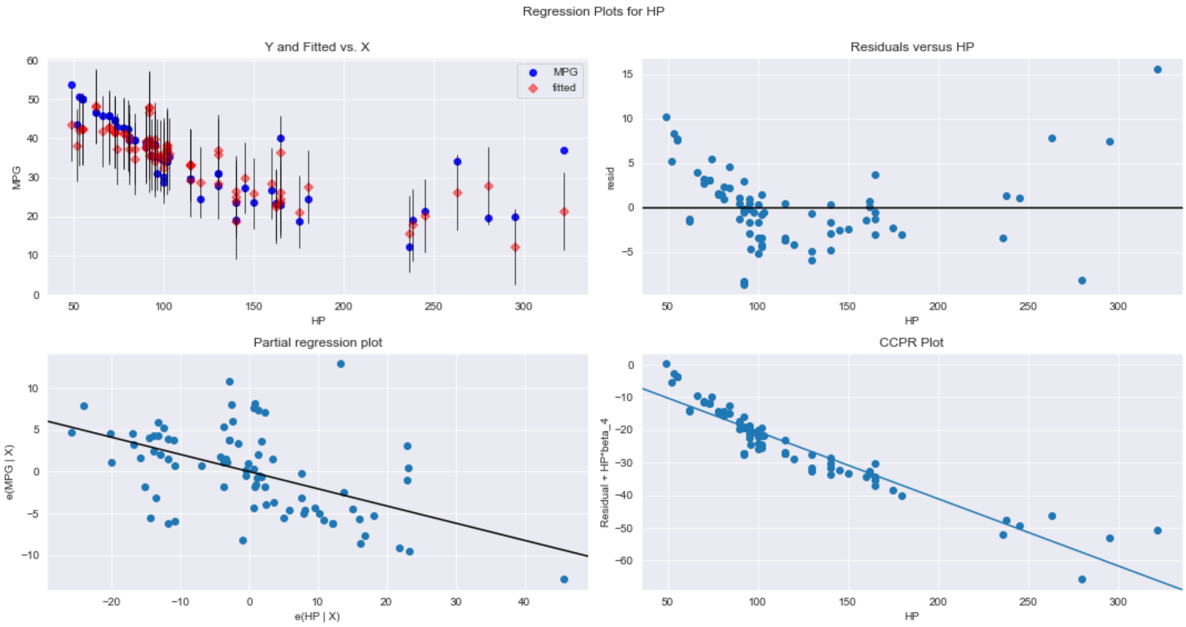

In [36]:
```python
fig = plt.figure(figsize=(15,8))
fig = sm.graphics.plot_regress_exog(model, "SP", fig=fig)
plt.show()
```

```
eval_env: 1
```



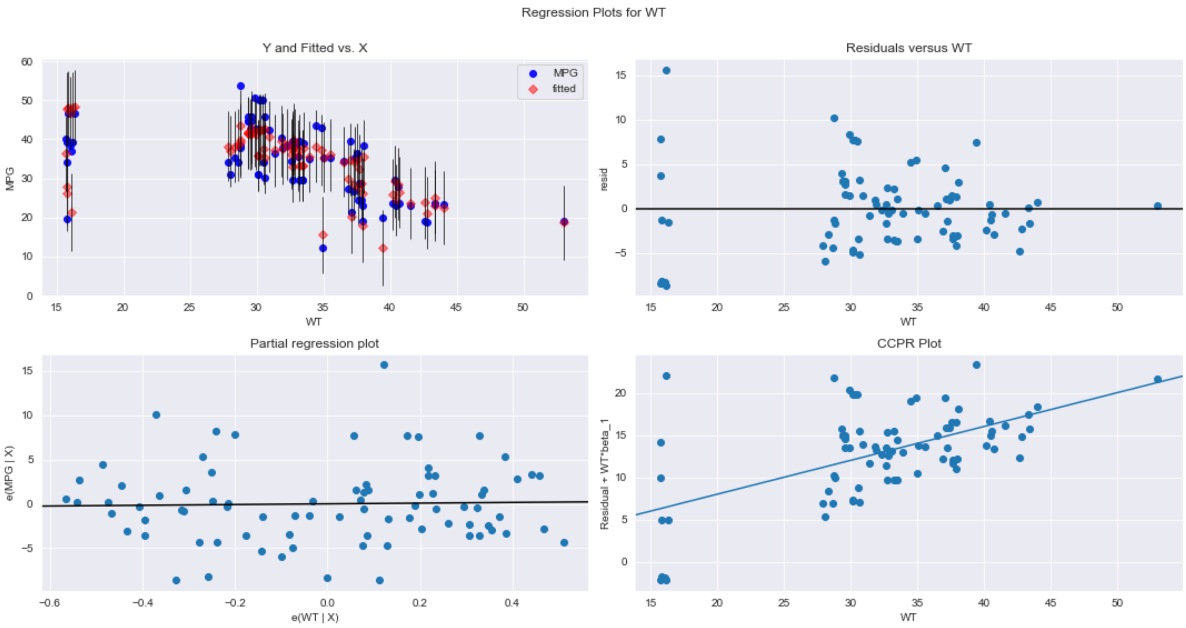

In [37]:
```python
fig = plt.figure(figsize=(15,8))
fig = sm.graphics.plot_regress_exog(model, "HP", fig=fig)
plt.show()
```

```
eval_env: 1
```




In [38]:
```python
fig = plt.figure(figsize=(15,8))
fig = sm.graphics.plot_regress_exog(model, "WT", fig=fig)
plt.show()
```

```
eval_env: 1
```




# Model Deletion Diagnostics

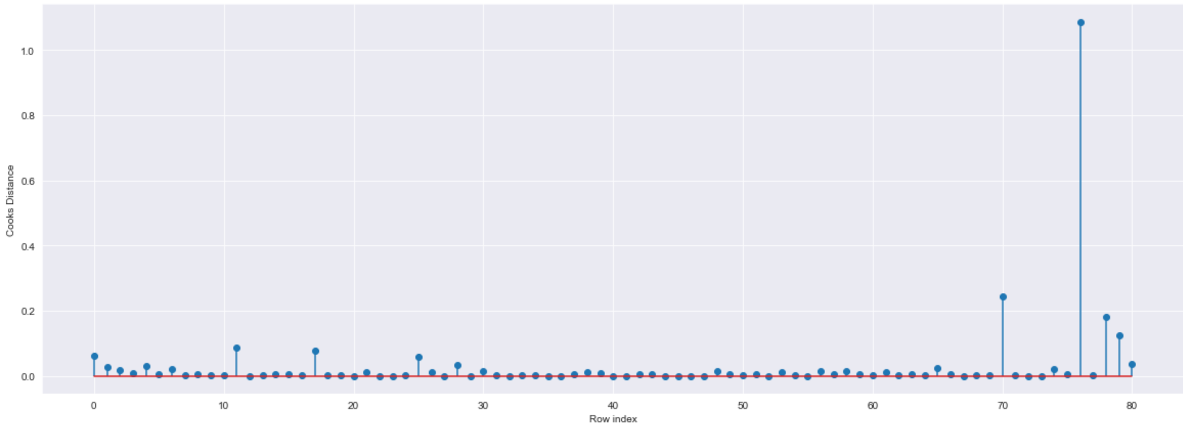
## Detecting Influencers/Outliers

## Cook’s Distance

In [31]:
```python
model_influence = model.get_influence()
(c, _) = model_influence.cooks_distance
```

In [32]:
```python
#Plot the influencers values using stem plot
fig = plt.subplots(figsize=(20, 7))
plt.stem(np.arange(len(cars)), np.round(c, 3))
```

```
plt.xlabel('Row index')
plt.ylabel('Cooks Distance')
plt.show()
```

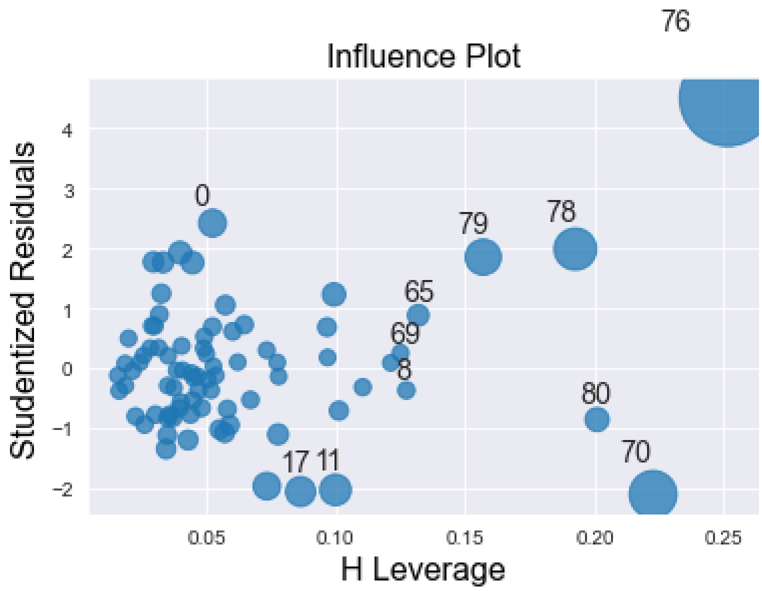



In [33]:
```
#index and value of influencer where c is more than .5
(np.argmax(c),np.max(c))
```

Out[33]:
```
(76, 1.0865193998179947)
```

# High Influence points

In [34]:
```
from statsmodels.graphics.regressionplots import influence_plot
influence_plot(model)
plt.show()
```




In [1]:
```
k = cars.shape[1]
n = cars.shape[0]
leverage_cutoff = 3*((k + 1)/n)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-7296e6debee6> in <module>
----> 1 k = cars.shape[1]
      2 n = cars.shape[0]
      3 leverage_cutoff = 3*((k + 1)/n)

NameError: name 'cars' is not defined
```

### From the above plot, it is evident that data point 70 and 76 are the influencers

In [36]:
```
cars[cars.index.isin([68, 74])]
```

Out[36]:

| | HP | MPG | VOL | SP | WT |
|---|---|---|---|---|---|
| **68** | 165 | 23.103172 | 123 | 133.312342 | 40.472042 |
| **74** | 140 | 19.086341 | 129 | 121.864163 | 42.618698 |

In [37]:
```
#See the differences in HP and other variable values
cars.head()
```

Out[37]:

| | HP | MPG | VOL | SP | WT |
|---|---|---|---|---|---|
| **0** | 49 | 53.700681 | 89 | 104.185353 | 28.762059 |
| **1** | 55 | 50.013401 | 92 | 105.461264 | 30.466833 |
| **2** | 55 | 50.013401 | 92 | 105.461264 | 30.193597 |
| **3** | 70 | 45.696322 | 92 | 113.461264 | 30.632114 |
| **4** | 53 | 50.504232 | 92 | 104.461264 | 29.889149 |

# Improving the model

In [38]:
```
#Load the data
cars_new = pd.read_csv("Cars.csv")
```

In [39]:
```
#Discard the data points which are influencers and reasign the row number (reset_in
car1=cars_new.drop(cars_new.index[[68,74]],axis=0).reset_index()
```

In [40]:
```
#Drop the original index
car1=car1.drop(['index'],axis=1)
```

In [41]:
```
car1
```

Out[41]:

| | HP | MPG | VOL | SP | WT |
|---|---|---|---|---|---|
| **0** | 49 | 53.700681 | 89 | 104.185353 | 28.762059 |
| **1** | 55 | 50.013401 | 92 | 105.461264 | 30.466833 |
| **2** | 55 | 50.013401 | 92 | 105.461264 | 30.193597 |
| **3** | 70 | 45.696322 | 92 | 113.461264 | 30.632114 |
| **4** | 53 | 50.504232 | 92 | 104.461264 | 29.889149 |
| **...** | ... | ... | ... | ... | ... |
| **74** | 322 | 36.900000 | 50 | 169.598513 | 16.132947 |
| **75** | 238 | 19.197888 | 115 | 150.576579 | 37.923113 |
| **76** | 263 | 34.000000 | 50 | 151.598513 | 15.769625 |
| **77** | 295 | 19.833733 | 119 | 167.944460 | 39.423099 |
| **78** | 236 | 12.101263 | 107 | 139.840817 | 34.948615 |

79 rows × 5 columns

# Build Model

In [42]:
```
#Exclude variable "WT" and generate R-Squared and AIC values
final_ml_V= smf.ols('MPG~VOL+SP+HP',data = car1).fit()
```

In [43]:
```
(final_ml_V.rsquared,final_ml_V.aic)
```

Out[43]:
```
(0.7609747264109348, 465.0378229302144)
```

In [44]:
```
#Exclude variable "VOL" and generate R-Squared and AIC values
final_ml_W= smf.ols('MPG~WT+SP+HP',data = car1).fit()
```

In [45]:
```
(final_ml_W.rsquared,final_ml_W.aic)
```

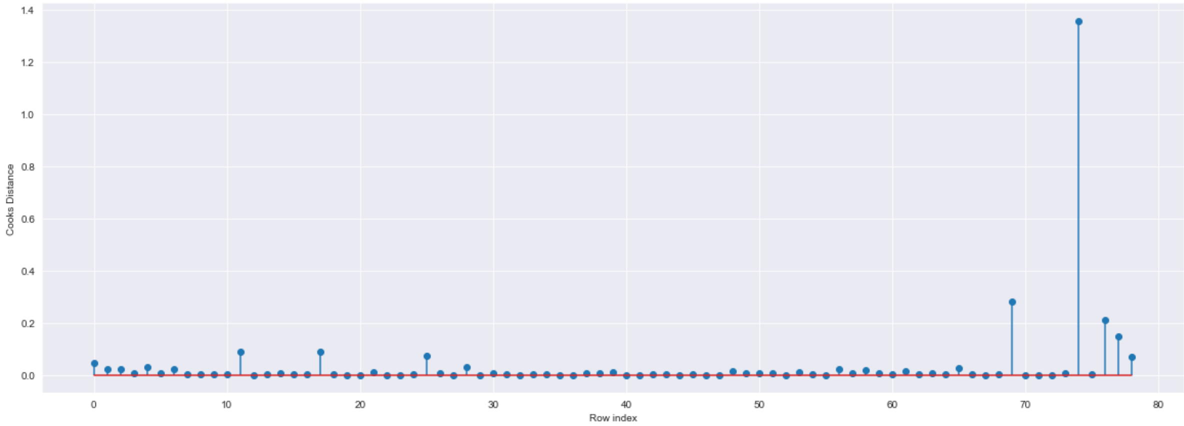
Out[45]:
```
(0.7599704098655541, 465.36906316731984)
```

**Comparing above R-Square and AIC values, model 'final_ml_V' has high R- square and low AIC value hence include variable 'VOL' so that multi collinearity problem would be resolved.**

# Cook's Distance

In [46]:
```
model_influence_V = final_ml_V.get_influence()
(c_V, _) = model_influence_V.cooks_distance
```

In [47]:
```
fig= plt.subplots(figsize=(20,7))
plt.stem(np.arange(len(car1)),np.round(c_V,3));
plt.xlabel('Row index')
plt.ylabel('Cooks Distance');
```




In [48]:
```
#index of the data points where c is more than .5
(np.argmax(c_V),np.max(c_V))
```

Out[48]:
```
(74, 1.3577278867500056)
```

In [49]:
```
#Drop 76 and 77 observations
car2=car1.drop(car1.index[[74]],axis=0)
```

In [50]:
```
car2
```

Out[50]:

| | HP | MPG | VOL | SP | WT |
|---|---|---|---|---|---|
| **0** | 49 | 53.700681 | 89 | 104.185353 | 28.762059 |
| **1** | 55 | 50.013401 | 92 | 105.461264 | 30.466833 |
| **2** | 55 | 50.013401 | 92 | 105.461264 | 30.193597 |
| **3** | 70 | 45.696322 | 92 | 113.461264 | 30.632114 |
| **4** | 53 | 50.504232 | 92 | 104.461264 | 29.889149 |
| **...** | ... | ... | ... | ... | ... |
| **73** | 175 | 18.762837 | 129 | 132.864163 | 42.778219 |
| **75** | 238 | 19.197888 | 115 | 150.576579 | 37.923113 |
| **76** | 263 | 34.000000 | 50 | 151.598513 | 15.769625 |
| **77** | 295 | 19.833733 | 119 | 167.944460 | 39.423099 |
| **78** | 236 | 12.101263 | 107 | 139.840817 | 34.948615 |

78 rows × 5 columns

In [51]:
```
#Reset the index and re arrange the row values
car3=car2.reset_index()
```

In [52]:
```
car4=car3.drop(['index'],axis=1)
```

In [53]:
```
car4
```

Out[53]:

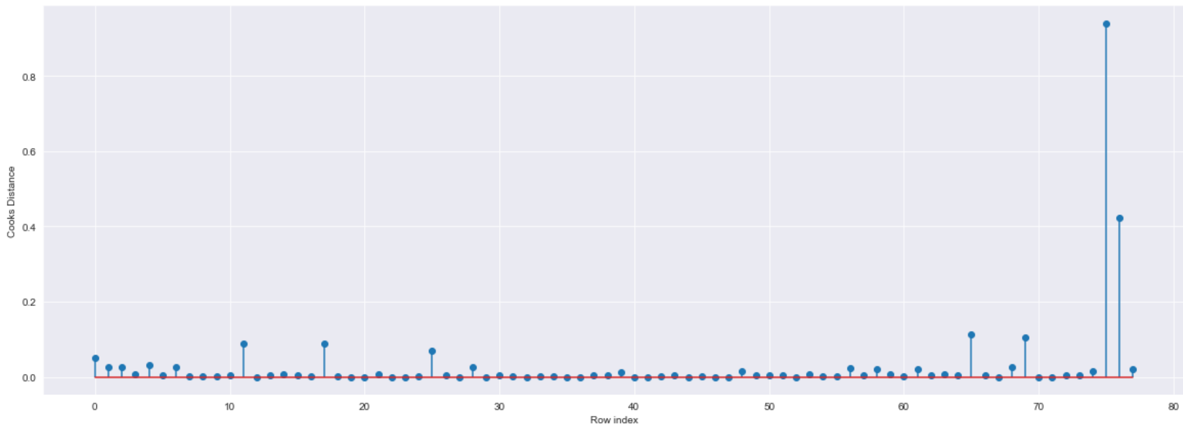
| | HP | MPG | VOL | SP | WT |
|---|---|---|---|---|---|
| **0** | 49 | 53.700681 | 89 | 104.185353 | 28.762059 |
| **1** | 55 | 50.013401 | 92 | 105.461264 | 30.466833 |
| **2** | 55 | 50.013401 | 92 | 105.461264 | 30.193597 |
| **3** | 70 | 45.696322 | 92 | 113.461264 | 30.632114 |
| **4** | 53 | 50.504232 | 92 | 104.461264 | 29.889149 |
| **...** | ... | ... | ... | ... | ... |
| **73** | 175 | 18.762837 | 129 | 132.864163 | 42.778219 |
| **74** | 238 | 19.197888 | 115 | 150.576579 | 37.923113 |
| **75** | 263 | 34.000000 | 50 | 151.598513 | 15.769625 |
| **76** | 295 | 19.833733 | 119 | 167.944460 | 39.423099 |
| **77** | 236 | 12.101263 | 107 | 139.840817 | 34.948615 |

78 rows × 5 columns

In [54]:
```
#Build the model on the new data
final_ml_V= smf.ols('MPG~VOL+SP+HP',data = car4).fit()
```

In [55]:
```
#Again check for influencers
model_influence_V = final_ml_V.get_influence()
(c_V, _) = model_influence_V.cooks_distance
```

In [56]:
```
fig= plt.subplots(figsize=(20,7))
plt.stem(np.arange(len(car4)),np.round(c_V,3));
plt.xlabel('Row index')
plt.ylabel('Cooks Distance');
```




In [57]:
```
#index of the data points where c is more than .5
(np.argmax(c_V),np.max(c_V))
```

Out[57]:
```
(75, 0.9407391391291708)
```

## Since the value is <1 , we can stop the diagnostic process and finalize the model

In [58]:
```
#Check the accuracy of the mode
final_ml_V= smf.ols('MPG~VOL+SP+HP',data = car4).fit()
```

In [59]:
```
(final_ml_V.rsquared,final_ml_V.aic)
```

Out[59]:
```
(0.8128580840170883, 441.1019267870809)
```

# Predicting for new data

In [60]:
```
#New data for prediction
new_data=pd.DataFrame({'HP':40,"VOL":95,"SP":102,"WT":35},index=[1])
```

In [61]:
```
final_ml_V.predict(new_data)
```

Out[61]:
```
1    44.376113
dtype: float64
```

In [62]:
```
final_ml_V.predict(cars_new.iloc[0:5,])
```

Out[62]:
```
0    44.245744
1    42.918036
2    42.918036
3    42.662430
4    42.977162
dtype: float64
```

In [63]:
```
pred_y = final_ml_V.predict(cars_new)
```

In [64]:
```
pred_y
```

Out[64]:
```
0     44.245744
1     42.918036
2     42.918036
3     42.662430
4     42.977162
        ...
76    16.003514
77    16.228852
78    22.068019
79     9.691507
80    13.970818
Length: 81, dtype: float64
```

In [65]:
```
final_ml_V.summary()
```

Out[65]:

OLS Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | MPG | **R-squared:** | 0.813 |
| **Model:** | OLS | **Adj. R-squared:** | 0.805 |
| **Method:** | Least Squares | **F-statistic:** | 107.1 |
| **Date:** | Wed, 01 Dec 2021 | **Prob (F-statistic):** | 7.37e-27 |
| **Time:** | 08:23:44 | **Log-Likelihood:** | -216.55 |
| **No. Observations:** | 78 | **AIC:** | 441.1 |
| **Df Residuals:** | 74 | **BIC:** | 450.5 |
| **Df Model:** | 3 | | |
| **Covariance Type:** | nonrobust | | |

| | **coef** | **std err** | **t** | **P>\|t\|** | **[0.025** | **0.975]** |
|---|---|---|---|---|---|---|
| **Intercept** | 30.6697 | 13.155 | 2.331 | 0.022 | 4.458 | 56.881 |
| **VOL** | -0.1675 | 0.022 | -7.750 | 0.000 | -0.211 | -0.124 |
| **SP** | 0.3757 | 0.142 | 2.640 | 0.010 | 0.092 | 0.659 |
| **HP** | -0.2174 | 0.036 | -6.114 | 0.000 | -0.288 | -0.147 |

| | | | |
|---|---|---|---|
| **Omnibus:** | 9.478 | **Durbin-Watson:** | 1.101 |
| **Prob(Omnibus):** | 0.009 | **Jarque-Bera (JB):** | 9.191 |
| **Skew:** | 0.775 | **Prob(JB):** | 0.0101 |
| **Kurtosis:** | 3.653 | **Cond. No.** | 5.76e+03 |

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 5.76e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

In [ ]:

In [ ]:

In [ ]: