In [1]:
```python
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 [2]:
```python
#read the data
toyoto_corrola=pd.read_csv('C:\\Users\\DELL\\Downloads\\toyoto_corrola.csv')
toyoto_corrola
```

Out[2]:

	Id	Model	Price	Age_08_04	KM	HP	Doors	Cylinders	Gears	Weight
0	1	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13500	23	46986	90	3	4	5	1165
1	2	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13750	23	72937	90	3	4	5	1165
2	3	ÊTOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13950	24	41711	90	3	4	5	1165
3	4	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	14950	26	48000	90	3	4	5	1165
4	5	TOYOTA Corolla 2.0 D4D HATCHB SOL 2/3-Doors	13750	30	38500	90	3	4	5	1170
...	...	...	...	...	...	...	...	...	...	...
1431	1438	TOYOTA Corolla 1.3 16V HATCHB G6 2/3-Doors	7500	69	20544	86	3	4	5	1025
1432	1439	TOYOTA Corolla 1.3 16V HATCHB LINEA TERRA 2/3-...	10845	72	19000	86	3	4	5	1015
1433	1440	TOYOTA Corolla 1.3 16V HATCHB LINEA TERRA 2/3-...	8500	71	17016	86	3	4	5	1015
1434	1441	TOYOTA Corolla 1.3 16V HATCHB LINEA TERRA 2/3-...	7250	70	16916	86	3	4	5	1015
1435	1442	TOYOTA Corolla 1.6 LB LINEA TERRA 4/5-Doors	6950	76	1	110	5	4	5	1114

1436 rows × 10 columns

In [3]:
```python
toyoto_corrola.head()
```

Out[3]:

	Id	Model	Price	Age_08_04	KM	HP	Doors	Cylinders	Gears	Weight
0	1	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13500	23	46986	90	3	4	5	1165
1	2	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13750	23	72937	90	3	4	5	1165
2	3	ÊTOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13950	24	41711	90	3	4	5	1165
3	4	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	14950	26	48000	90	3	4	5	1165
4	5	TOYOTA Corolla 2.0 D4D HATCHB SOL 2/3-Doors	13750	30	38500	90	3	4	5	1170

In [4]:
```python
toyoto_corrola.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1436 entries, 0 to 1435
Data columns (total 10 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Id         1436 non-null   int64
 1   Model      1436 non-null   object
 2   Price      1436 non-null   int64
 3   Age_08_04  1436 non-null   int64
 4   KM         1436 non-null   int64
 5   HP         1436 non-null   int64
 6   Doors      1436 non-null   int64
 7   Cylinders  1436 non-null   int64
 8   Gears      1436 non-null   int64
 9   Weight     1436 non-null   int64
dtypes: int64(9), object(1)
memory usage: 112.3+ KB
```

In [5]:
```python
#missing value
toyoto_corrola.isna().sum()
```

Out[5]:
```
Id           0
Model        0
Price        0
Age_08_04    0
KM           0
HP           0
Doors        0
Cylinders    0
Gears        0
Weight       0
dtype: int64
```

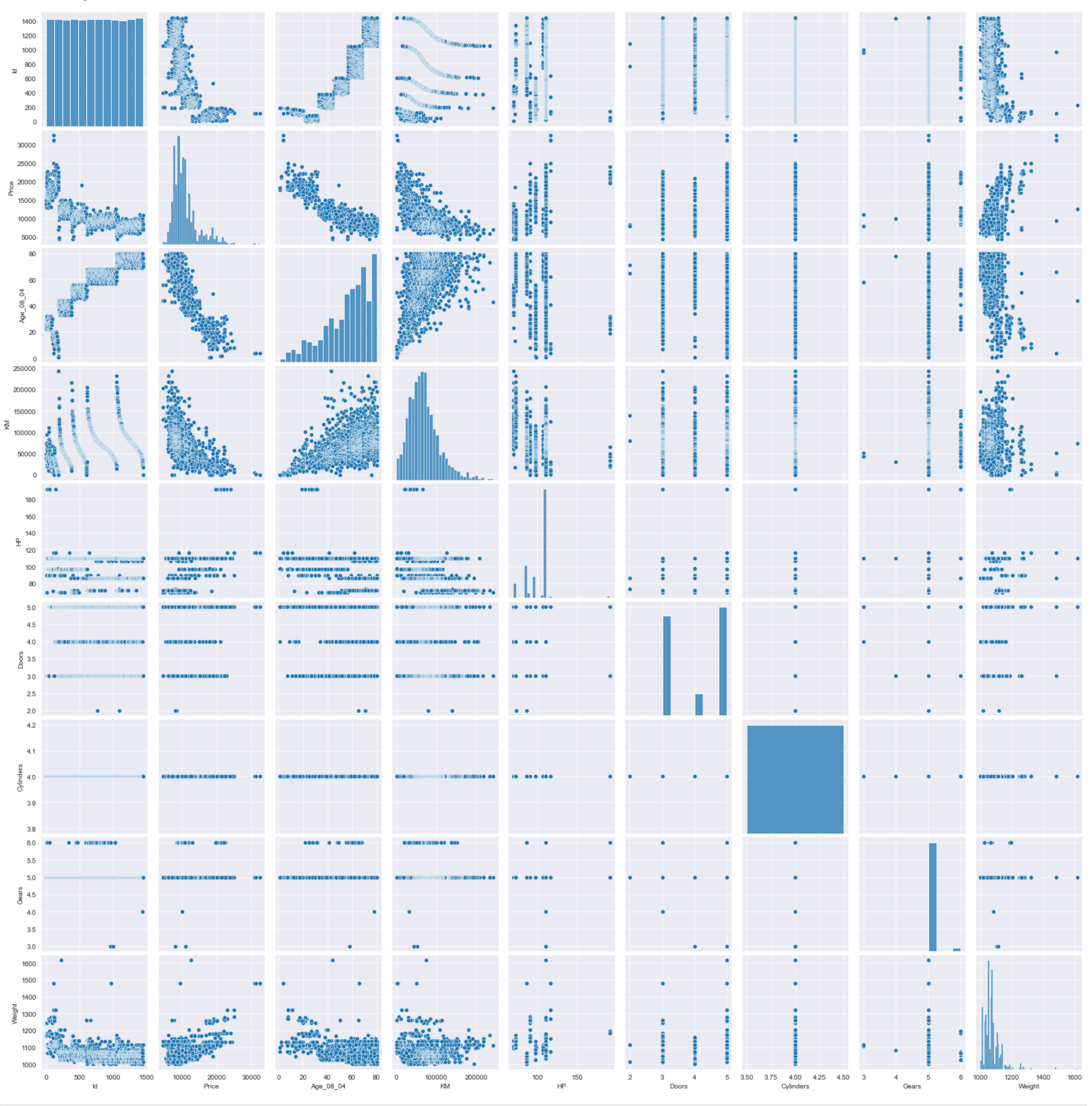
In [6]:
```python
#correlation
toyoto_corrola.corr()
```

Out[6]:

	Id	Price	Age_08_04	KM	HP	Doors	Cylinders	Gears	Weight
Id	1.000000	-0.738250	0.906132	0.273298	-0.109375	-0.130207	NaN	-0.043343	-0.414500
Price	-0.738250	1.000000	-0.876590	-0.569960	0.314990	0.185326	NaN	0.063104	0.581198
Age_08_04	0.906132	-0.876590	1.000000	0.505672	-0.156622	-0.148359	NaN	-0.005364	-0.470253
KM	0.273298	-0.569960	0.505672	1.000000	-0.333538	-0.036197	NaN	0.015023	-0.028598
HP	-0.109375	0.314990	-0.156622	-0.333538	1.000000	0.092424	NaN	0.209477	0.089614
Doors	-0.130207	0.185326	-0.148359	-0.036197	0.092424	1.000000	NaN	-0.160141	0.302618
Cylinders	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Gears	-0.043343	0.063104	-0.005364	0.015023	0.209477	-0.160141	NaN	1.000000	0.020613
Weight	-0.414500	0.581198	-0.470253	-0.028598	0.089614	0.302618	NaN	0.020613	1.000000

In [7]:
```python
#Format the plot background and scatter plots for all the variables
sns.set_style(style='darkgrid')
sns.pairplot(toyoto_corrola)
```

Out[7]: <seaborn.axisgrid.PairGrid at 0x21b4f77fa30>



In [8]:
```python
#Build model
import statsmodels.formula.api as smf
model=smf.ols('Price~KM+HP+Doors+Cylinders+Gears+Weight',data=toyoto_corrola).fit()
```

In [9]:
```python
#coefficients
model.params
```

Out[9]:
```
Intercept   -1945.606195
KM             -0.051054
HP             19.444021
Doors         -13.983691
Cylinders   -7782.424778
Gears         809.963361
Weight         38.525046
dtype: float64
```

In [10]:
```python
#t and p-Values
print(model.tvalues, '\n', model.pvalues)
```

```
Intercept   -17.526727
KM          -31.820210
HP            4.691072
Doors        -0.219993
Cylinders   -17.526727
Gears         2.581929
Weight       34.084708
dtype: float64
 Intercept     1.879922e-62
KM           1.884781e-168
HP            2.977620e-06
Doors         8.259077e-01
Cylinders     1.879922e-62
Gears         9.923706e-03
Weight       6.953130e-187
dtype: float64
```

In [11]:
```python
#R squared values
(model.rsquared,model.rsquared_adj)
```

Out[11]: (0.6531567201106192, 0.6519439813697472)

In [12]:
```python
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()
```

In [13]:
```python
list(np.where(model.resid>10))
```

Out[13]:
```
[array([   6,    7,    8,    9,   10,   11,   12,   13,   14,   15,   16,
          17,   18,   19,   20,   21,   22,   23,   24,   25,   26,   27,
          28,   29,   31,   32,   33,   34,   35,   36,   37,   38,   39,
          40,   41,   42,   43,   44,   45,   46,   47,   48,   49,   50,
          51,   52,   53,   54,   55,   56,   57,   58,   59,   60,   61,
          62,   63,   64,   65,   66,   67,   68,   69,   70,   71,   72,
          73,   74,   75,   76,   77,   78,   79,   80,   81,   82,   83,
          84,   85,   86,   87,   88,   89,   90,   91,   92,   93,   94,
          95,   96,   97,   98,   99,  100,  101,  102,  103,  104,  105,
         106,  107,  108,  109,  110,  111,  112,  113,  114,  115,  116,
         117,  118,  119,  120,  121,  122,  123,  124,  125,  126,  127,
         128,  129,  130,  131,  132,  133,  134,  135,  136,  137,  138,
         139,  140,  141,  142,  143,  144,  145,  146,  147,  148,  149,
         150,  151,  152,  153,  154,  155,  156,  157,  158,  159,  160,
         161,  162,  163,  164,  165,  166,  167,  168,  169,  170,  171,
         172,  173,  174,  175,  176,  177,  178,  179,  180,  181,  182,
         183,  184,  185,  186,  187,  188,  189,  190,  193,  194,  195,
         196,  197,  198,  199,  200,  201,  202,  203,  204,  205,  206,
         207,  208,  210,  212,  213,  215,  216,  217,  218,  219,  220,
         222,  223,  224,  225,  226,  227,  228,  229,  230,  231,  232,
         233,  234,  235,  237,  238,  239,  241,  242,  244,  245,  246,
         247,  248,  249,  250,  251,  252,  253,  255,  256,  257,  258,
         259,  260,  261,  262,  263,  264,  265,  266,  267,  269,  270,
         271,  272,  273,  274,  276,  277,  278,  279,  280,  281,  282,
         283,  285,  288,  289,  290,  291,  293,  294,  297,  298,  299,
         300,  301,  302,  303,  304,  305,  307,  308,  310,  311,  312,
         315,  316,  317,  322,  324,  326,  327,  328,  331,  333,  334,
         335,  336,  337,  338,  342,  345,  346,  347,  348,  353,  354,
         355,  356,  357,  359,  360,  361,  362,  363,  367,  371,  377,
         378,  379,  380,  381,  382,  383,  384,  385,  386,  387,  389,
         392,  394,  395,  396,  397,  398,  399,  400,  401,  403,  404,
         405,  406,  407,  408,  409,  410,  411,  413,  414,  415,  417,
         419,  420,  421,  423,  425,  426,  427,  428,  429,  430,  431,
         433,  434,  435,  438,  439,  440,  441,  442,  443,  444,  445,
         446,  448,  451,  452,  453,  454,  455,  456,  459,  460,  461,
         462,  466,  467,  468,  470,  471,  472,  473,  474,  477,  478,
         479,  481,  482,  484,  485,  488,  490,  492,  493,  494,  495,
         496,  497,  499,  504,  506,  507,  509,  510,  512,  513,  514,
         516,  517,  521,  523,  524,  526,  527,  529,  530,  534,  538,
         539,  541,  542,  543,  544,  546,  548,  549,  551,  557,  559,
         561,  563,  565,  569,  570,  572,  580,  587,  603,  604,  605,
         606,  607,  608,  610,  611,  612,  614,  616,  618,  620,  622,
         623,  625,  629,  632,  633,  636,  637,  638,  639,  640,  641,
         642,  643,  645,  646,  648,  649,  650,  651,  653,  656,  657,
         658,  659,  661,  662,  663,  664,  666,  667,  668,  669,  671,
         673,  674,  675,  676,  677,  678,  688,  690,  691,  692,  693,
         694,  695,  696,  698,  699,  701,  702,  703,  704,  706,  708,
         712,  716,  717,  723,  725,  726,  728,  729,  730,  731,  732,
         733,  735,  738,  740,  742,  745,  747,  749,  750,  751,  754,
         755,  759,  760,  761,  762,  763,  764,  765,  767,  769,  770,
         771,  773,  774,  775,  777,  778,  780,  781,  784,  788,  791,
         794,  796,  798,  803,  804,  809,  813,  815,  823,  825,  829,
         831,  834,  837,  839,  840,  841,  850,  855,  860,  865,  869,
         879,  880,  886,  887,  891,  894,  897,  898,  899,  902,  913,
         924,  925,  927,  929,  931,  934,  953,  957,  969,  970,  971,
         976,  987,  988,  999, 1044, 1045, 1046, 1049, 1050, 1051, 1052,
        1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1064, 1065,
        1066, 1067, 1068, 1069, 1075, 1076, 1077, 1079, 1081, 1082, 1084,
        1086, 1087, 1088, 1090, 1091, 1094, 1096, 1098, 1100, 1103, 1104,
        1105, 1106, 1110, 1117, 1120, 1121, 1123, 1125, 1131, 1133, 1136,
        1138, 1140, 1141, 1142, 1144, 1148, 1149, 1150, 1157, 1162, 1168,
        1179, 1184, 1188, 1189, 1196, 1198, 1210, 1211, 1214, 1224, 1233,
        1234, 1240, 1250, 1256, 1268, 1269, 1280, 1311, 1327, 1378, 1391,
        1402, 1432], dtype=int64)]
```

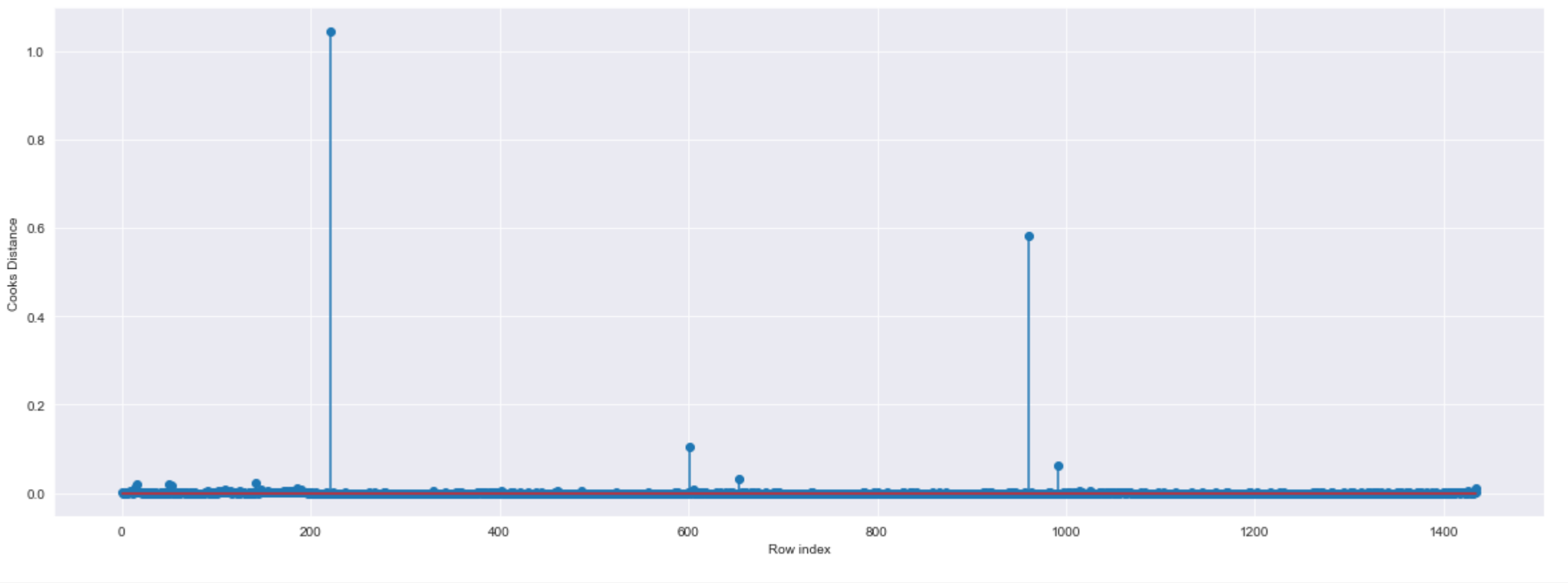
In [14]:
```python
def get_standardized_values( vals ):
    return (vals - vals.mean())/vals.std()
```

In [15]:
```python
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()
```

In [16]:
```python
model_influence = model.get_influence()
(c, _) = model_influence.cooks_distance
```

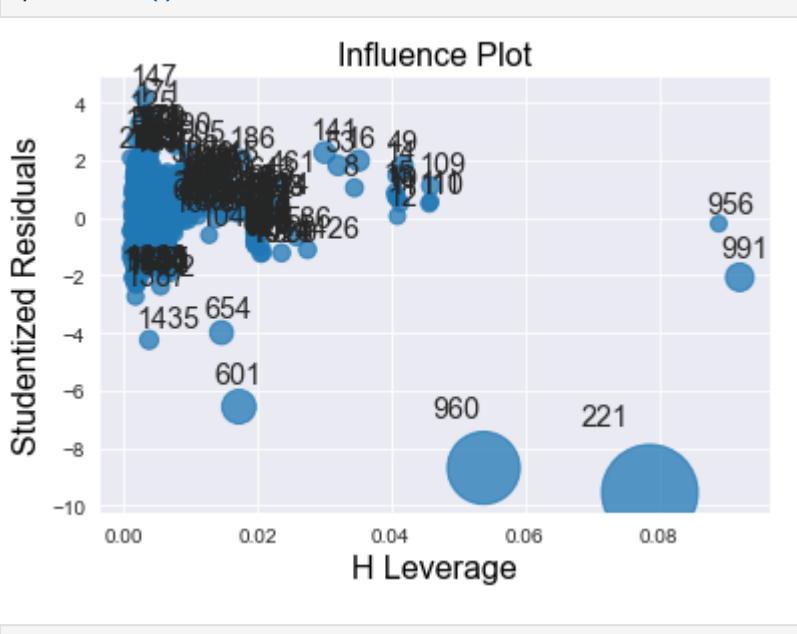
In [17]:
```python
#Plot the influencers values using stem plot
fig = plt.subplots(figsize=(20, 7))
plt.stem(np.arange(len(toyoto_corrola)), np.round(c, 3))
plt.xlabel('Row index')
plt.ylabel('Cooks Distance')
plt.show()
```



In [18]:
```python
#index and value of influencer where c is more than .5
(np.argmax(c),np.max(c))
```

Out[18]: (221, 1.0458156746423635)

In [19]:
```python
from statsmodels.graphics.regressionplots import influence_plot
influence_plot(model)
plt.show()
```



In [20]:
```python
k = toyoto_corrola.shape[1]
n = toyoto_corrola.shape[0]
leverage_cutoff = 3*((k + 1)/n)
```

In [21]:
```python
#from the above plot,it is evident that data point 956 and 991 are the influencers
```

In [22]:
```python
toyoto_corrola[toyoto_corrola.index.isin([956,991])]
```

Out[22]:

	Id	Model	Price	Age_08_04	KM	HP	Doors	Cylinders	Gears	Weight
956	960	TOYOTA Corolla 1.6 Linea Luna Aut. 4/5-Doors	10950	58	51421	110	5	4	3	1105
991	996	TOYOTA Corolla 1.6 Lin.Terra Aut. 4/5-Doors	7950	58	43000	110	4	4	3	1114

In [23]:
```python
#see the difference between price and other values
toyoto_corrola.head()
```

Out[23]:

	Id	Model	Price	Age_08_04	KM	HP	Doors	Cylinders	Gears	Weight
0	1	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13500	23	46986	90	3	4	5	1165
1	2	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13750	23	72937	90	3	4	5	1165
2	3	ÊTOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13950	24	41711	90	3	4	5	1165
3	4	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	14950	26	48000	90	3	4	5	1165
4	5	TOYOTA Corolla 2.0 D4D HATCHB SOL 2/3-Doors	13750	30	38500	90	3	4	5	1170

In [24]:
```python
#IMPROVING THE MODEL
```

In [25]:
```python
#load new data
tc_new=pd.read_csv('C:\\Users\\DELL\\Downloads\\toyoto_corrola.csv')
tc_new
```

Out[25]:

	Id	Model	Price	Age_08_04	KM	HP	Doors	Cylinders	Gears	Weight
0	1	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13500	23	46986	90	3	4	5	1165
1	2	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13750	23	72937	90	3	4	5	1165
2	3	ÊTOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13950	24	41711	90	3	4	5	1165
3	4	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	14950	26	48000	90	3	4	5	1165
4	5	TOYOTA Corolla 2.0 D4D HATCHB SOL 2/3-Doors	13750	30	38500	90	3	4	5	1170
...	...	...	...	...	...	...	...	...	...	...
1431	1438	TOYOTA Corolla 1.3 16V HATCHB G6 2/3-Doors	7500	69	20544	86	3	4	5	1025
1432	1439	TOYOTA Corolla 1.3 16V HATCHB LINEA TERRA 2/3-...	10845	72	19000	86	3	4	5	1015
1433	1440	TOYOTA Corolla 1.3 16V HATCHB LINEA TERRA 2/3-...	8500	71	17016	86	3	4	5	1015
1434	1441	TOYOTA Corolla 1.3 16V HATCHB LINEA TERRA 2/3-...	7250	70	16916	86	3	4	5	1015
1435	1442	TOYOTA Corolla 1.6 LB LINEA TERRA 4/5-Doors	6950	76	1	110	5	4	5	1114

1436 rows × 10 columns

In [26]:
```python
tc=tc_new.drop(columns=['Id'])
tc
```

Out[26]:

	Model	Price	Age_08_04	KM	HP	Doors	Cylinders	Gears	Weight
0	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13500	23	46986	90	3	4	5	1165
1	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13750	23	72937	90	3	4	5	1165
2	ÊTOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13950	24	41711	90	3	4	5	1165
3	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	14950	26	48000	90	3	4	5	1165
4	TOYOTA Corolla 2.0 D4D HATCHB SOL 2/3-Doors	13750	30	38500	90	3	4	5	1170
...	...	...	...	...	...	...	...	...	...
1431	TOYOTA Corolla 1.3 16V HATCHB G6 2/3-Doors	7500	69	20544	86	3	4	5	1025
1432	TOYOTA Corolla 1.3 16V HATCHB LINEA TERRA 2/3-...	10845	72	19000	86	3	4	5	1015
1433	TOYOTA Corolla 1.3 16V HATCHB LINEA TERRA 2/3-...	8500	71	17016	86	3	4	5	1015
1434	TOYOTA Corolla 1.3 16V HATCHB LINEA TERRA 2/3-...	7250	70	16916	86	3	4	5	1015
1435	TOYOTA Corolla 1.6 LB LINEA TERRA 4/5-Doors	6950	76	1	110	5	4	5	1114

1436 rows × 9 columns

In [27]:
```python
#buliding new data
import statsmodels.formula.api as smf
model=smf.ols('Price~KM+HP+Doors+Cylinders+Gears+Weight',data=tc).fit()
```

In [30]:
```python
finaldata=smf.ols('Price~KM+HP+Doors+Cylinders+Gears+Weight',data=tc).fit()
```

In [31]:
```python
finaldata.rsquared,finaldata.aic
```

Out[31]: (0.6531567201106192, 26104.982965425083)

In [32]:
```python
#prediction for new data
new_data=pd.DataFrame({"Age_08_04":24,"KM":17016,'HP':90,'Doors':3,'Cylinders':4,'Gears':5,'Weight':1165},index=[1])
```

In [33]:
```python
finaldata.predict(new_data)
```

Out[33]:
```
1    16695.469687
dtype: float64
```

In [34]:
```python
finaldata.predict(tc_new.iloc[0:5])
```

Out[34]:
```
0    15165.387407
1    13840.490334
2    15434.696183
3    15113.618857
4    15791.255154
dtype: float64
```

In [36]:
```python
pred_y=finaldata.predict(tc_new)
pred_y
```

Out[36]:
```
0       15165.387407
1       13840.490334
2       15434.696183
3       15113.618857
4       15791.255154
            ...
1431    11044.069345
1432    10737.645945
1433    10838.936677
1434    10844.042057
1435    15960.285711
Length: 1436, dtype: float64
```

In [37]:
```python
model.summary()
```

Out[37]:

OLS Regression Results

Dep. Variable:	Price	**R-squared:**	0.653
Model:	OLS	**Adj. R-squared:**	0.652
Method:	Least Squares	**F-statistic:**	538.6
Date:	Wed, 14 Apr 2021	**Prob (F-statistic):**	0.00
Time:	14:32:42	**Log-Likelihood:**	-13046.
No. Observations:	1436	**AIC:**	2.610e+04
Df Residuals:	1430	**BIC:**	2.614e+04
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-1945.6062	111.008	-17.527	0.000	-2163.362	-1727.850
KM	-0.0511	0.002	-31.820	0.000	-0.054	-0.048
HP	19.4440	4.145	4.691	0.000	11.313	27.575
Doors	-13.9837	63.564	-0.220	0.826	-138.673	110.705
Cylinders	-7782.4248	444.032	-17.527	0.000	-8653.448	-6911.401
Gears	809.9634	313.705	2.582	0.010	194.592	1425.334
Weight	38.5250	1.130	34.085	0.000	36.308	40.742

Omnibus:	372.072	**Durbin-Watson:**	0.938
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	4637.190
Skew:	-0.841	**Prob(JB):**	0.00
Kurtosis:	11.641	**Cond. No.**	5.44e+20

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 2.96e-29. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

In [ ]: