Import Libraries

In [1]:

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
import statsmodels.formula.api as smf
import statsmodels.api as sm
from statsmodels.graphics.regressionplots import influence_plot
```

1. Import data set

In [2]:

```
car_data = pd.read_csv('ToyotaCorolla.csv')
car_data
```

Out[2]:

	Id	Model	Price	Age_08_04	Mfg_Month	Mfg_Year	KM	Fuel_Type	HP	Met_(
0	1	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13500	23	10	2002	46986	Diesel	90	
1	2	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13750	23	10	2002	72937	Diesel	90	
2	3	�TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13950	24	9	2002	41711	Diesel	90	
3	4	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	14950	26	7	2002	48000	Diesel	90	
4	5	TOYOTA Corolla 2.0 D4D HATCHB SOL 2/3-Doors	13750	30	3	2002	38500	Diesel	90	
...	...	...	...	...	...	...	...	...	...	
1431	1438	TOYOTA Corolla 1.3 16V HATCHB G6 2/3-Doors	7500	69	12	1998	20544	Petrol	86	
1432	1439	TOYOTA Corolla 1.3 16V HATCHB LINEA TERRA 2/3-...	10845	72	9	1998	19000	Petrol	86	
1433	1440	TOYOTA Corolla 1.3 16V HATCHB LINEA TERRA 2/3-...	8500	71	10	1998	17016	Petrol	86	

	Id	Model	Price	Age_08_04	Mfg_Month	Mfg_Year	KM	Fuel_Type	HP	Met_(
1434	1441	TOYOTA Corolla 1.3 16V HATCHB LINEA TERRA 2/3-...	7250	70	11	1998	16916	Petrol	86	
1435	1442	TOYOTA Corolla 1.6 LB LINEA TERRA 4/5-Doors	6950	76	5	1998	1	Petrol	110	

1436 rows × 38 columns

In [3]:

```
car_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1436 entries, 0 to 1435
Data columns (total 38 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   Mfg_Month         1436 non-null   int64 
 5   Mfg_Year          1436 non-null   int64 
 6   KM                1436 non-null   int64 
 7   Fuel_Type         1436 non-null   object
 8   HP                1436 non-null   int64 
 9   Met_Color         1436 non-null   int64 
 10  Color             1436 non-null   object
 11  Automatic         1436 non-null   int64 
 12  cc                1436 non-null   int64 
 13  Doors             1436 non-null   int64 
 14  Cylinders         1436 non-null   int64 
 15  Gears             1436 non-null   int64 
 16  Quarterly_Tax     1436 non-null   int64 
 17  Weight            1436 non-null   int64 
 18  Mfr_Guarantee     1436 non-null   int64 
 19  BOVAG_Guarantee   1436 non-null   int64 
 20  Guarantee_Period  1436 non-null   int64 
 21  ABS               1436 non-null   int64 
 22  Airbag_1          1436 non-null   int64 
 23  Airbag_2          1436 non-null   int64 
 24  Airco             1436 non-null   int64 
 25  Automatic_airco   1436 non-null   int64 
 26  Boardcomputer     1436 non-null   int64 
 27  CD_Player         1436 non-null   int64 
 28  Central_Lock      1436 non-null   int64 
 29  Powered_Windows   1436 non-null   int64 
 30  Power_Steering    1436 non-null   int64 
 31  Radio             1436 non-null   int64 
 32  Mistlamps         1436 non-null   int64 
 33  Sport_Model       1436 non-null   int64 
 34  Backseat_Divider  1436 non-null   int64 
 35  Metallic_Rim      1436 non-null   int64 
 36  Radio_cassette    1436 non-null   int64 
 37  Tow_Bar           1436 non-null   int64 
dtypes: int64(35), object(3)
memory usage: 426.4+ KB
```

In [4]:

```
car_data=pd.DataFrame(data=car_data,columns=["Price","Age_08_04","KM","HP","cc","Doors","Ge
car_data
```

Out[4]:

	Price	Age_08_04	KM	HP	cc	Doors	Gears	Quarterly_Tax	Weight
0	13500	23	46986	90	2000	3	5	210	1165
1	13750	23	72937	90	2000	3	5	210	1165
2	13950	24	41711	90	2000	3	5	210	1165
3	14950	26	48000	90	2000	3	5	210	1165
4	13750	30	38500	90	2000	3	5	210	1170
...	...	...	...	...	...	...	...	...	...
1431	7500	69	20544	86	1300	3	5	69	1025
1432	10845	72	19000	86	1300	3	5	69	1015
1433	8500	71	17016	86	1300	3	5	69	1015
1434	7250	70	16916	86	1300	3	5	69	1015
1435	6950	76	1	110	1600	5	5	19	1114

1436 rows × 9 columns

In [5]:

```
car_data.isna().sum()
```

Out[5]:

```
Price            0
Age_08_04        0
KM               0
HP               0
cc               0
Doors            0
Gears            0
Quarterly_Tax    0
Weight           0
dtype: int64
```

In [6]:

```
car_data.dtypes
```

Out[6]:

```
Price            int64
Age_08_04        int64
KM               int64
HP               int64
cc               int64
Doors            int64
Gears            int64
Quarterly_Tax    int64
Weight           int64
dtype: object
```

In [7]:

```
car_data = car_data.rename({'Age_08_04':'Age','cc':'CC','Quarterly_Tax':'QT'},axis=1)
car_data
```

Out[7]:

	Price	Age	KM	HP	CC	Doors	Gears	QT	Weight
0	13500	23	46986	90	2000	3	5	210	1165
1	13750	23	72937	90	2000	3	5	210	1165
2	13950	24	41711	90	2000	3	5	210	1165
3	14950	26	48000	90	2000	3	5	210	1165
4	13750	30	38500	90	2000	3	5	210	1170
...	...	...	...	...	...	...	...	...	...
1431	7500	69	20544	86	1300	3	5	69	1025
1432	10845	72	19000	86	1300	3	5	69	1015
1433	8500	71	17016	86	1300	3	5	69	1015
1434	7250	70	16916	86	1300	3	5	69	1015
1435	6950	76	1	110	1600	5	5	19	1114

1436 rows × 9 columns

In [8]:

```
car_data.head()
```

Out[8]:

	Price	Age	KM	HP	CC	Doors	Gears	QT	Weight
0	13500	23	46986	90	2000	3	5	210	1165
1	13750	23	72937	90	2000	3	5	210	1165
2	13950	24	41711	90	2000	3	5	210	1165
3	14950	26	48000	90	2000	3	5	210	1165
4	13750	30	38500	90	2000	3	5	210	1170

In [9]:

```
car_data.shape
```

Out[9]:

(1436, 9)

In [10]:

```
car_data.describe(include ='all')
```

Out[10]:

	Price	Age	KM	HP	CC	Doors	
count	1436.000000	1436.000000	1436.000000	1436.000000	1436.00000	1436.000000	1436.
mean	10730.824513	55.947075	68533.259749	101.502089	1576.85585	4.033426	5.
std	3626.964585	18.599988	37506.448872	14.981080	424.38677	0.952677	0.
min	4350.000000	1.000000	1.000000	69.000000	1300.00000	2.000000	3.
25%	8450.000000	44.000000	43000.000000	90.000000	1400.00000	3.000000	5.
50%	9900.000000	61.000000	63389.500000	110.000000	1600.00000	4.000000	5.
75%	11950.000000	70.000000	87020.750000	110.000000	1600.00000	5.000000	5.
max	32500.000000	80.000000	243000.000000	192.000000	16000.00000	5.000000	6.

In [11]:

```
car_data.corr()
```

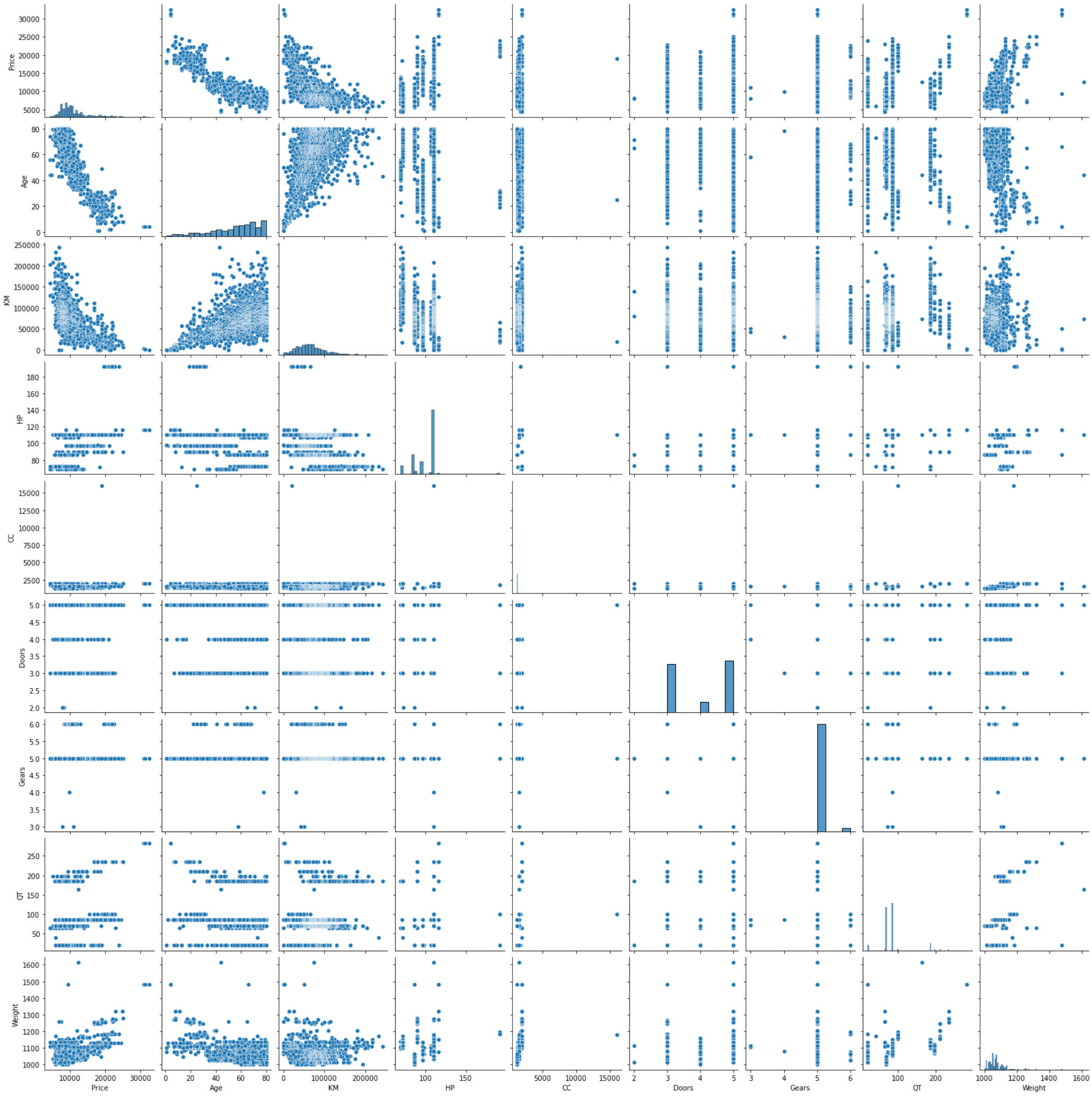
Out[11]:

	Price	Age	KM	HP	CC	Doors	Gears	QT
Price	1.000000	-0.876590	-0.569960	0.314990	0.126389	0.185326	0.063104	0.219197
Age	-0.876590	1.000000	0.505672	-0.156622	-0.098084	-0.148359	-0.005364	-0.198431
KM	-0.569960	0.505672	1.000000	-0.333538	0.102683	-0.036197	0.015023	0.278165
HP	0.314990	-0.156622	-0.333538	1.000000	0.035856	0.092424	0.209477	-0.298432
CC	0.126389	-0.098084	0.102683	0.035856	1.000000	0.079903	0.014629	0.306996
Doors	0.185326	-0.148359	-0.036197	0.092424	0.079903	1.000000	-0.160141	0.109363
Gears	0.063104	-0.005364	0.015023	0.209477	0.014629	-0.160141	1.000000	-0.005452
QT	0.219197	-0.198431	0.278165	-0.298432	0.306996	0.109363	-0.005452	1.000000
Weight	0.581198	-0.470253	-0.028598	0.089614	0.335637	0.302618	0.020613	0.626134

2. check for Linearity

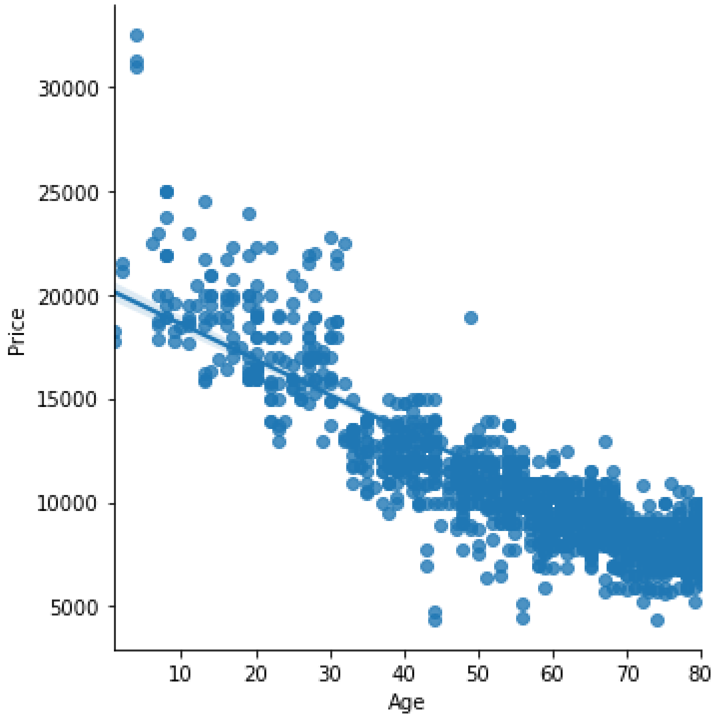
In [12]:

```
sns.pairplot(car_data)
plt.show()
```



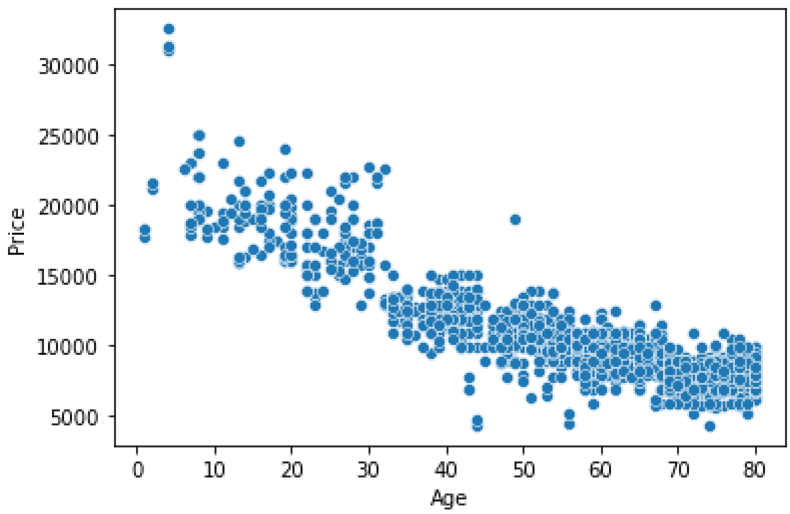
In [13]:

```
sns.lmplot(x='Age', y='Price', data=car_data)
plt.show()
```



In [14]:

```
sns.scatterplot( x='Age',y='Price',data=car_data)
plt.show()
```



3.Model Building

In [15]:

```
model = smf.ols("Price~Age+KM+HP+CC+Doors+Gears+QT+Weight", data=car_data).fit()
model
```

Out[15]:

```
<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x222aacfba
30>
```

4. Model Testing

In [16]:

```
# finding p and t values
np.round(model.pvalues,5),model.tvalues
```

Out[16]:

```
(Intercept    0.00008
 Age          0.00000
 KM           0.00000
 HP           0.00000
 CC           0.17909
 Doors        0.96777
 Gears        0.00261
 QT           0.00262
 Weight       0.00000
 dtype: float64,
 Intercept    -3.948666
 Age         -46.511852
 KM          -16.621622
 HP           11.241018
 CC           -1.344222
 Doors        -0.040410
 Gears         3.016007
 QT            3.014535
 Weight       15.879803
 dtype: float64)
```

In [17]:

```
model.rsquared, model.rsquared_adj
```

Out[17]:

```
(0.8637627463428192, 0.8629989775766963)
```

In [18]:

```
model_2 = smf.ols('Price~CC', data=car_data).fit()
np.round(model_2.pvalues), model_2.tvalues # CC has Significant pvalue
```

Out[18]:

```
(Intercept    0.0
 CC           0.0
 dtype: float64,
 Intercept    24.694090
 CC            4.824822
 dtype: float64)
```

In [19]:

```
model_3 = smf.ols('Price~Doors', data=car_data).fit()
model_3
```

Out[19]:

```
<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x222ab6ff4
90>
```

In [20]:

```
model_3.pvalues,model_3.tvalues # Doors has Significant pvalue
```

Out[20]:

```
(Intercept    1.094732e-73
 Doors        1.461237e-12
 dtype: float64,
 Intercept    19.258097
 Doors         7.141657
 dtype: float64)
```

In [21]:

```
model_4 = smf.ols('Price~CC+Doors', data=car_data).fit()
model_4.pvalues, model_4.tvalues #CC and Doors have significant pvalues
```

Out[21]:

```
(Intercept    1.056885e-34
 CC           1.521992e-05
 Doors        1.373469e-11
 dtype: float64,
 Intercept    12.620704
 CC            4.340400
 Doors         6.816153
 dtype: float64)
```

Model Validation

In [22]:

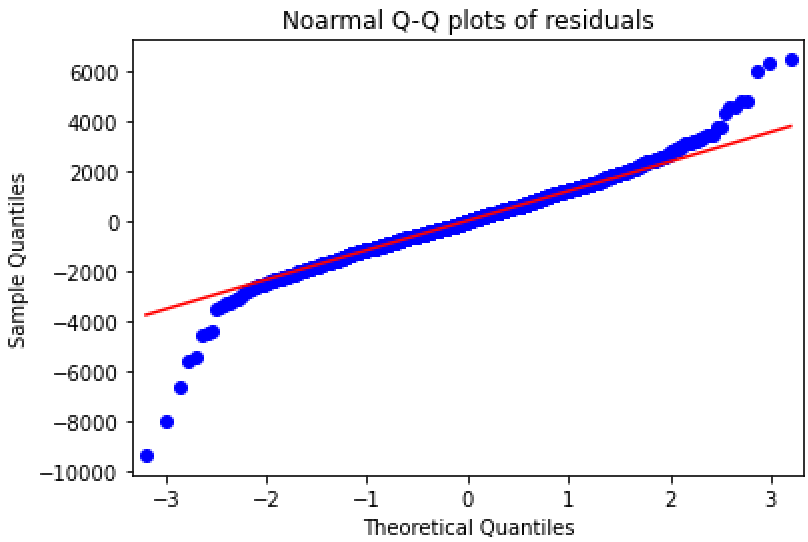
```
# Collinearity Check
rsq_age = smf.ols('Age~KM+HP+CC+Doors+Gears+QT+Weight',data=car_data).fit().rsquared
vif_age=1/(1-rsq_age)

rsq_km = smf.ols('KM~Age+HP+CC+Doors+Gears+QT+Weight',data=car_data).fit().rsquared
vif_km=1/(1-rsq_km)

rsq_hp = smf.ols('HP~KM+Age+CC+Doors+Gears+QT+Weight',data=car_data).fit().rsquared
vif_hp=1/(1-rsq_hp)

rsq_cc = smf.ols('CC~HP+KM+Age+Doors+Gears+QT+Weight', data=car_data).fit().rsquared
vif_cc=1/(1-rsq_cc)

rsq_doors = smf.ols('Doors~CC+HP+KM+Age+Gears+QT+Weight',data=car_data).fit().rsquared
vif_doors=1/(1-rsq_doors)

rsq_gears = smf.ols('Gears~Doors+CC+HP+KM+Age+QT+Weight',data=car_data).fit().rsquared
vif_gears=1/(1-rsq_gears)

rsq_qt = smf.ols('QT~Gears+Doors+CC+HP+KM+Age+Weight', data=car_data).fit().rsquared
vif_qt=1/(1-rsq_qt)

rsq_weight = smf.ols('Weight~QT+Gears+Doors+CC+HP+KM+Age',data=car_data).fit().rsquared
vif_weight=1/(1-rsq_weight)

df={'Variables':['Age','KM','HP','CC','Doors','Gears','QT','Weight'],
   'vif':[vif_age,vif_cc,vif_doors,vif_gears,vif_hp,vif_km,vif_qt,vif_weight,]}
vif=pd.DataFrame(df)
vif
```

Out[22]:

	Variables	vif
0	Age	1.884620
1	KM	1.163894
2	HP	1.156575
3	CC	1.098723
4	Doors	1.419422
5	Gears	1.756905
6	QT	2.311431
7	Weight	2.516420

In [23]:

```
# Residual Analysis
sm.qqplot(model.resid,line='q')
plt.title('Noarmal Q-Q plots of residuals')
plt.show()
```



In [24]:

```
list(np.where(model.resid>6000))
```

Out[24]:

```
[array([147, 523], dtype=int64)]
```

In [25]:

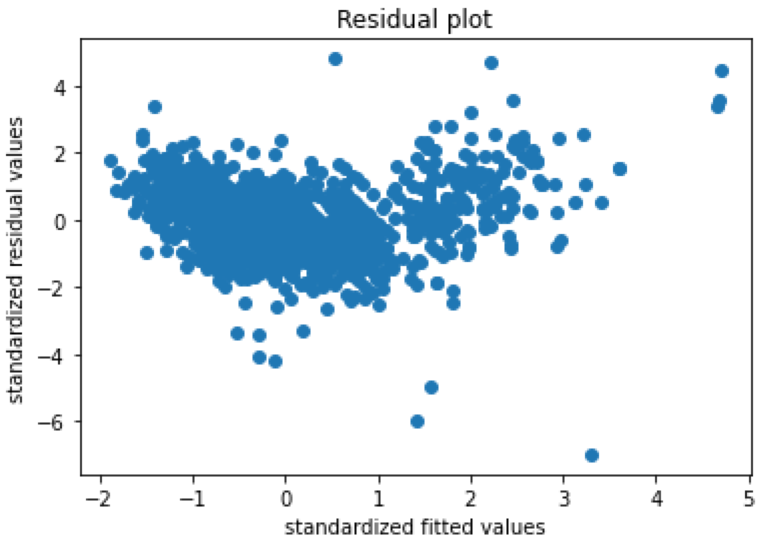
```
list(np.where(model.resid<-6000))
```

Out[25]:

```
[array([221, 601, 960], dtype=int64)]
```

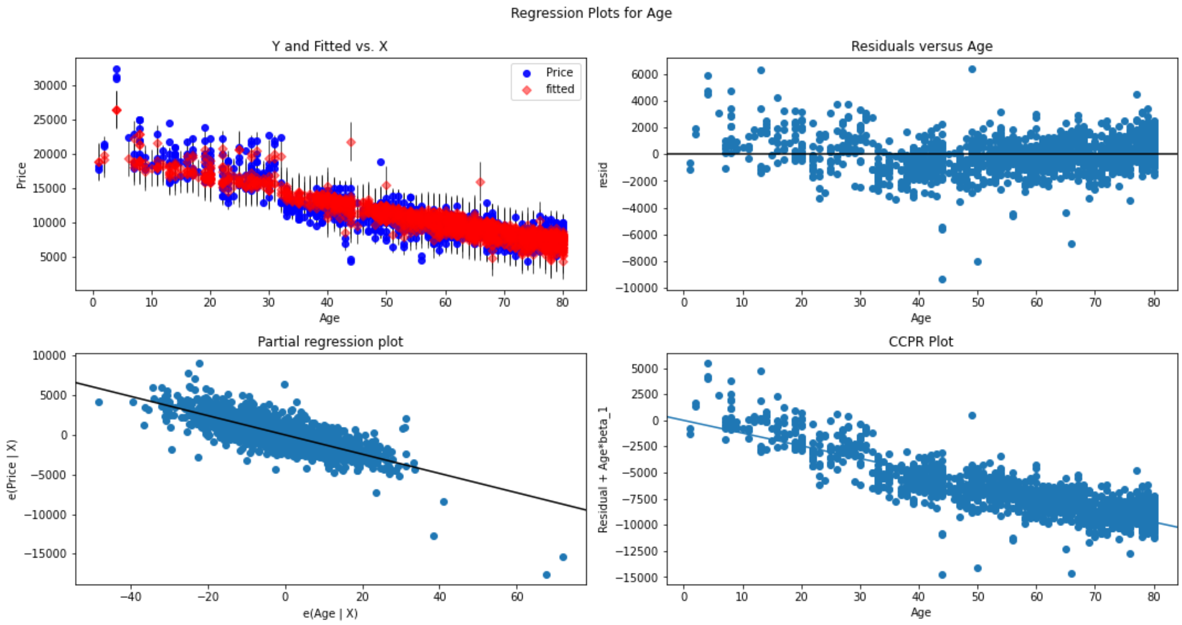
In [26]:

```
def standard_values(vals) : return (vals-vals.mean())/vals.std()
plt.scatter(standard_values(model.fittedvalues),standard_values(model.resid))
plt.title('Residual plot')
plt.xlabel('standardized fitted values')
plt.ylabel('standardized residual values')
plt.show()
```



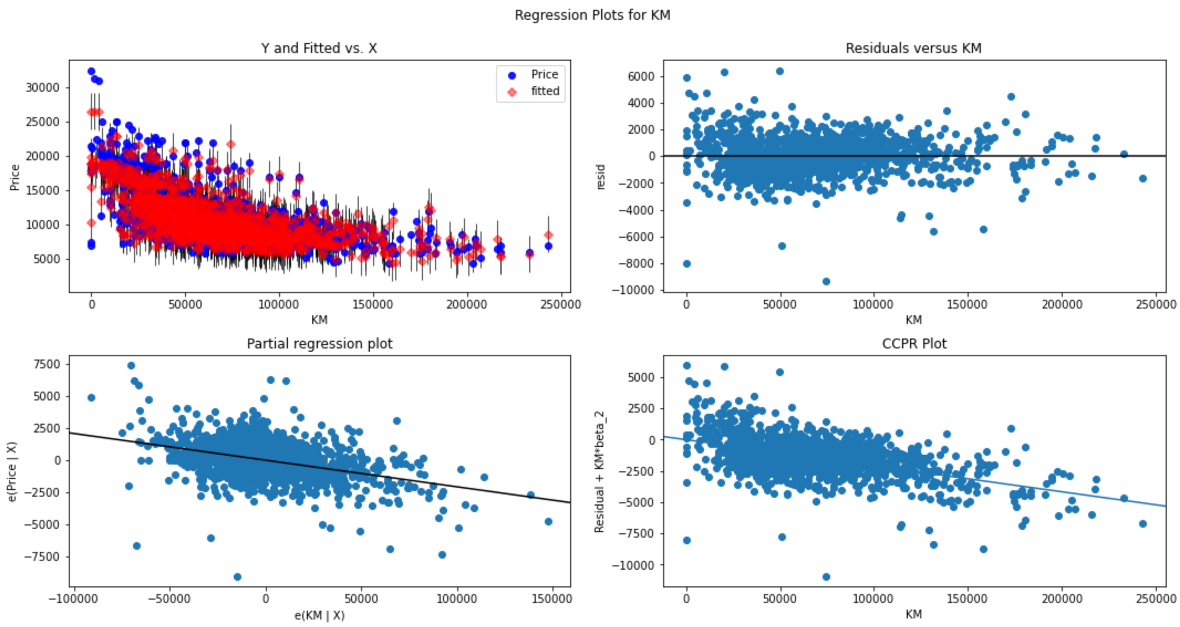
In [27]:

```
#residual plots
fig=plt.figure(figsize=(15,8))
sm.graphics.plot_regress_exog(model,'Age',fig=fig)
plt.show()
```



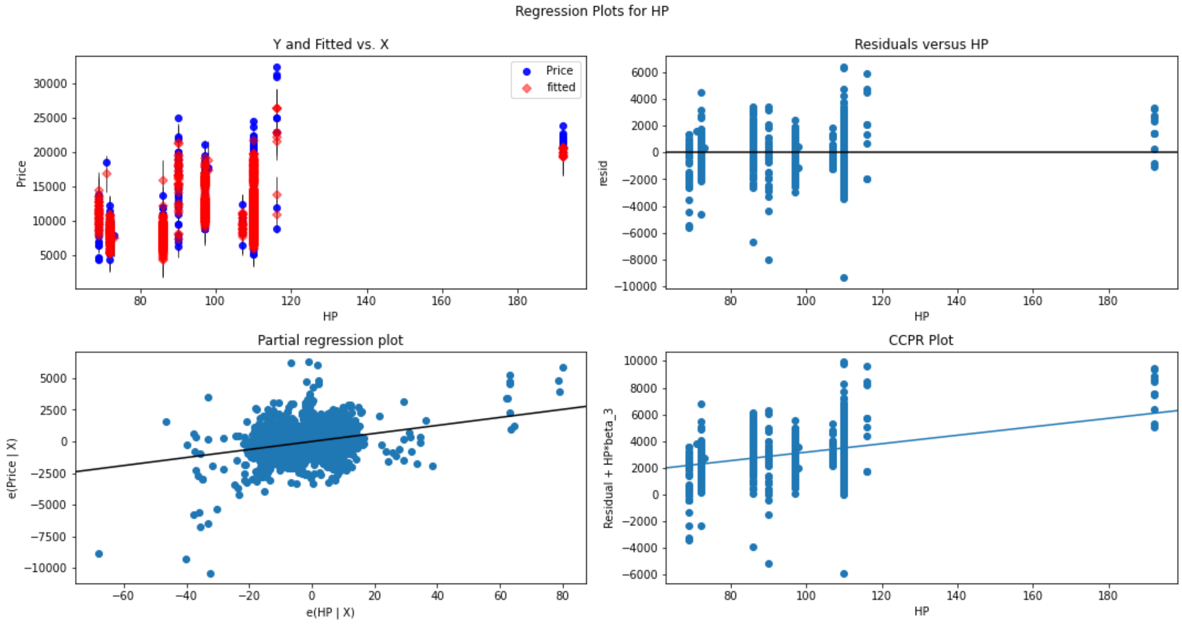
In [28]:

```
fig=plt.figure(figsize=(15,8))
sm.graphics.plot_regress_exog(model,'KM',fig=fig)
plt.show()
```



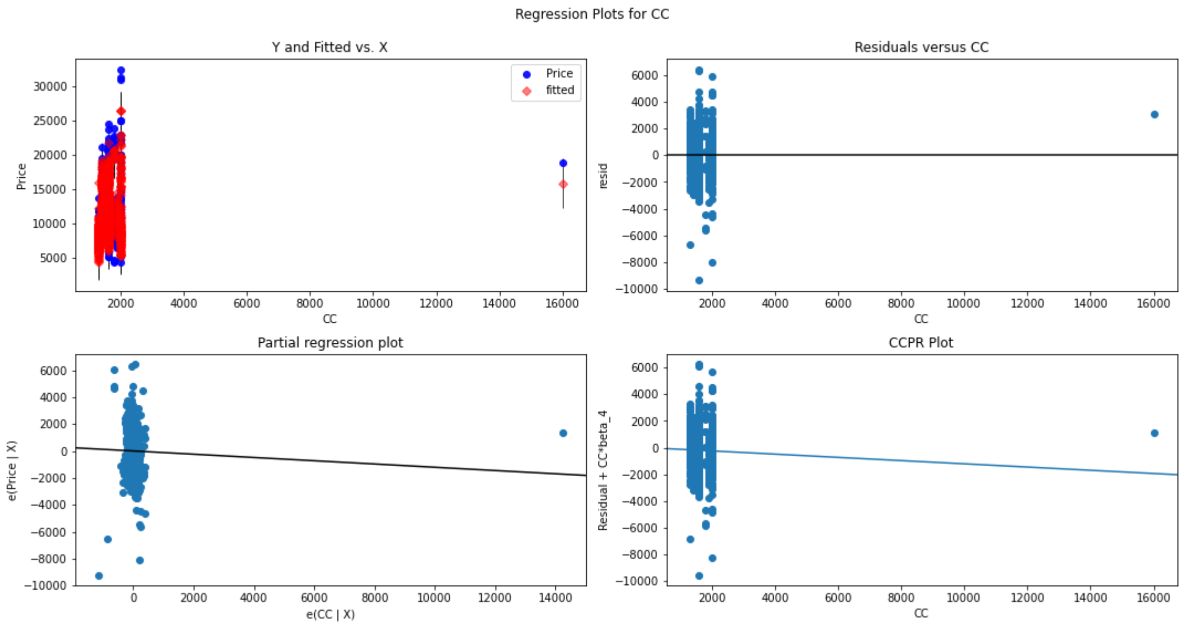
In [29]:

```
fig=plt.figure(figsize=(15,8))
sm.graphics.plot_regress_exog(model,'HP',fig=fig)
plt.show()
```



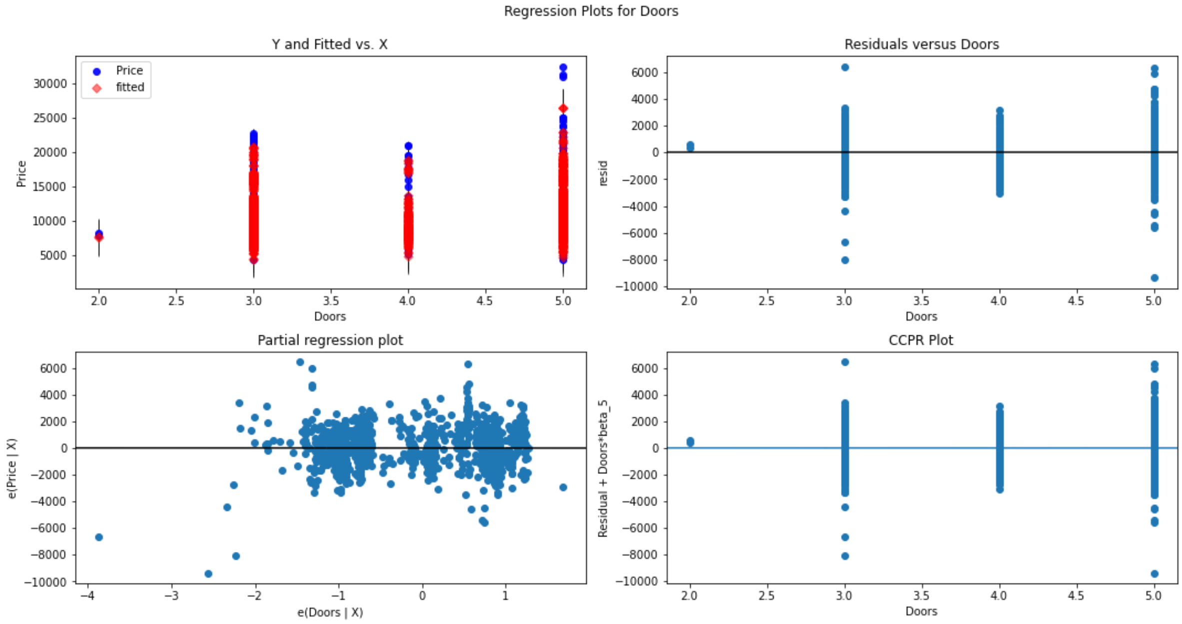
In [30]:

```
fig=plt.figure(figsize=(15,8))
sm.graphics.plot_regress_exog(model,'CC',fig=fig)
plt.show()
```



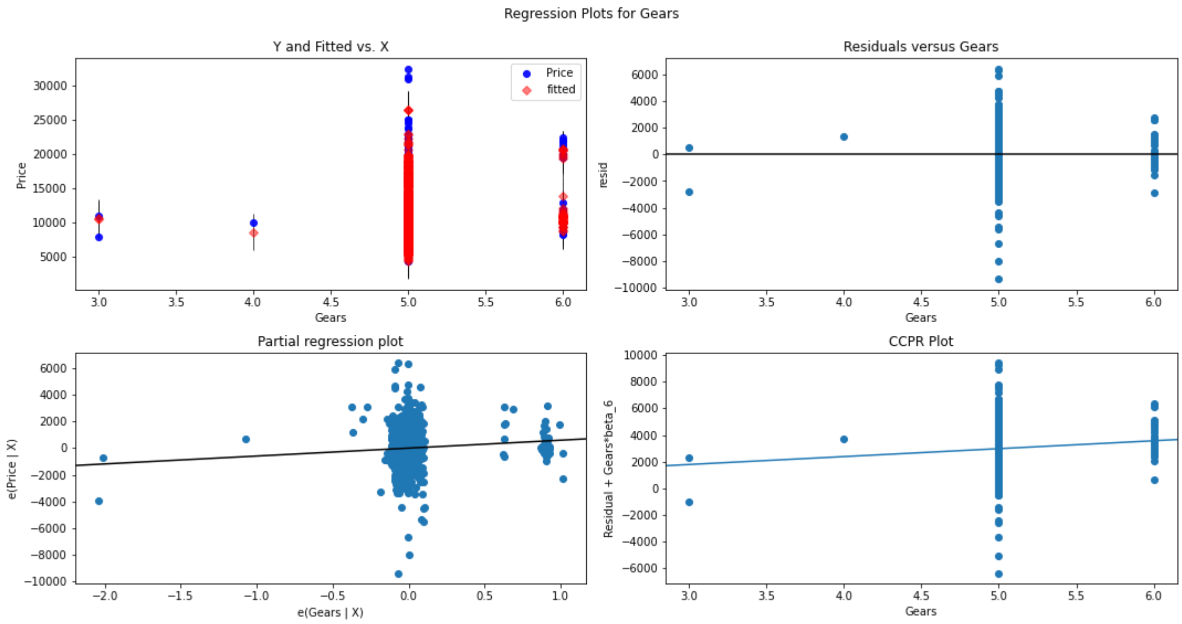
In [31]:

```
fig=plt.figure(figsize=(15,8))
sm.graphics.plot_regress_exog(model,'Doors',fig=fig)
plt.show()
```



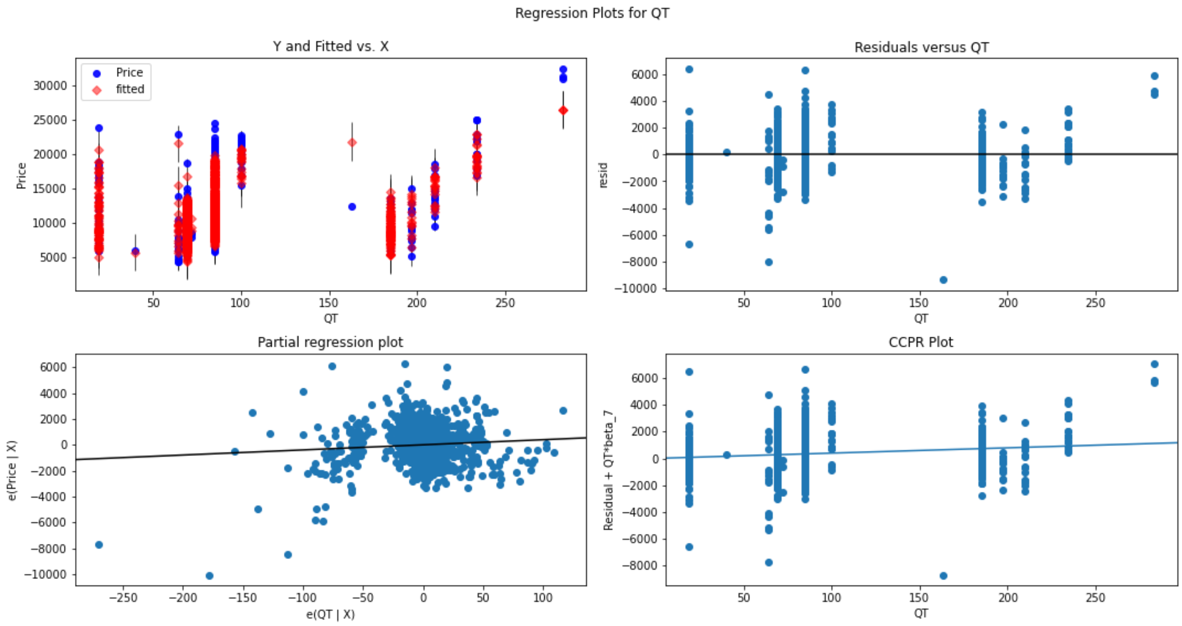
In [32]:

```
fig=plt.figure(figsize=(15,8))
sm.graphics.plot_regress_exog(model,'Gears',fig=fig)
plt.show()
```



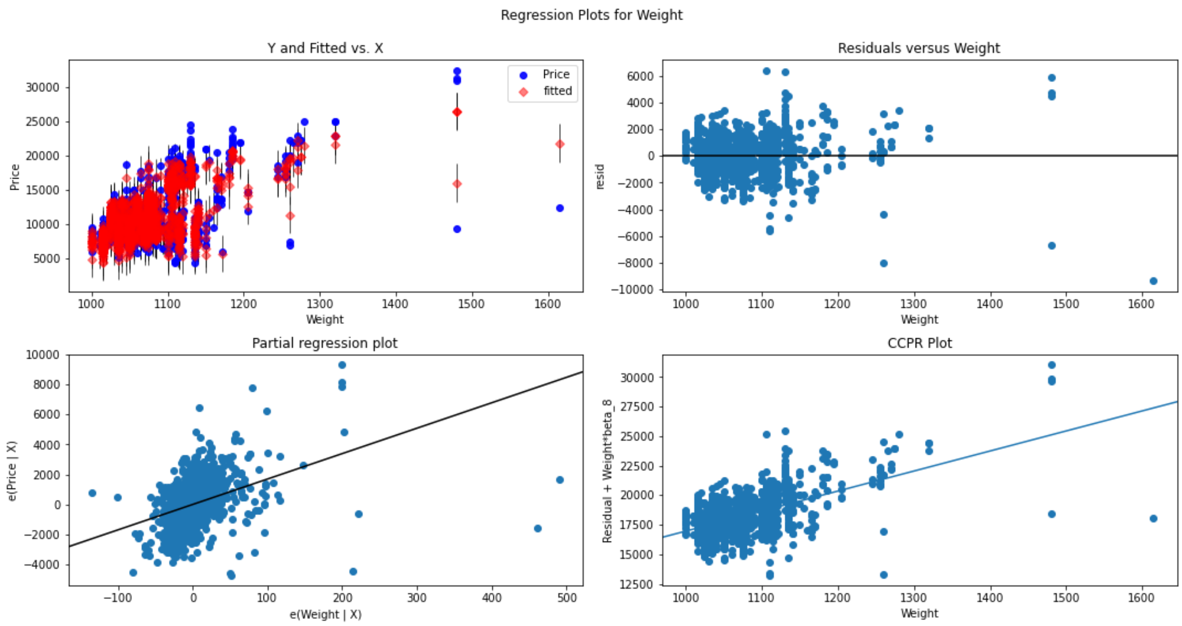
In [33]:

```
fig=plt.figure(figsize=(15,8))
sm.graphics.plot_regress_exog(model,'QT',fig=fig)
plt.show()
```



In [34]:

```
fig=plt.figure(figsize=(15,8))
sm.graphics.plot_regress_exog(model,'Weight',fig=fig)
plt.show()
```



In [35]:

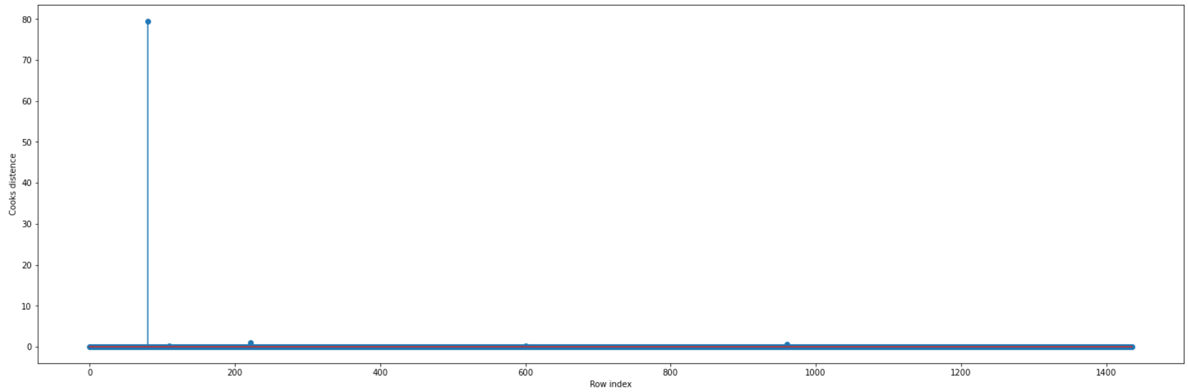
```
# Model delegation diagotics
#1. cooks distence
(C,_)=model.get_influence().cooks_distance
C
```

Out[35]:

```
array([7.23682667e-03, 3.96793393e-03, 5.46476784e-03, ...,
       8.44762355e-07, 6.97878368e-04, 1.08627724e-02])
```

In [36]:

```
#plot the influencers using stem plot
plt.figure(figsize=(25,8))
plt.stem(np.arange(len(car_data)),np.round(C,3))
plt.xlabel('Row index')
plt.ylabel('Cooks distence')
plt.show()
```



In [37]:

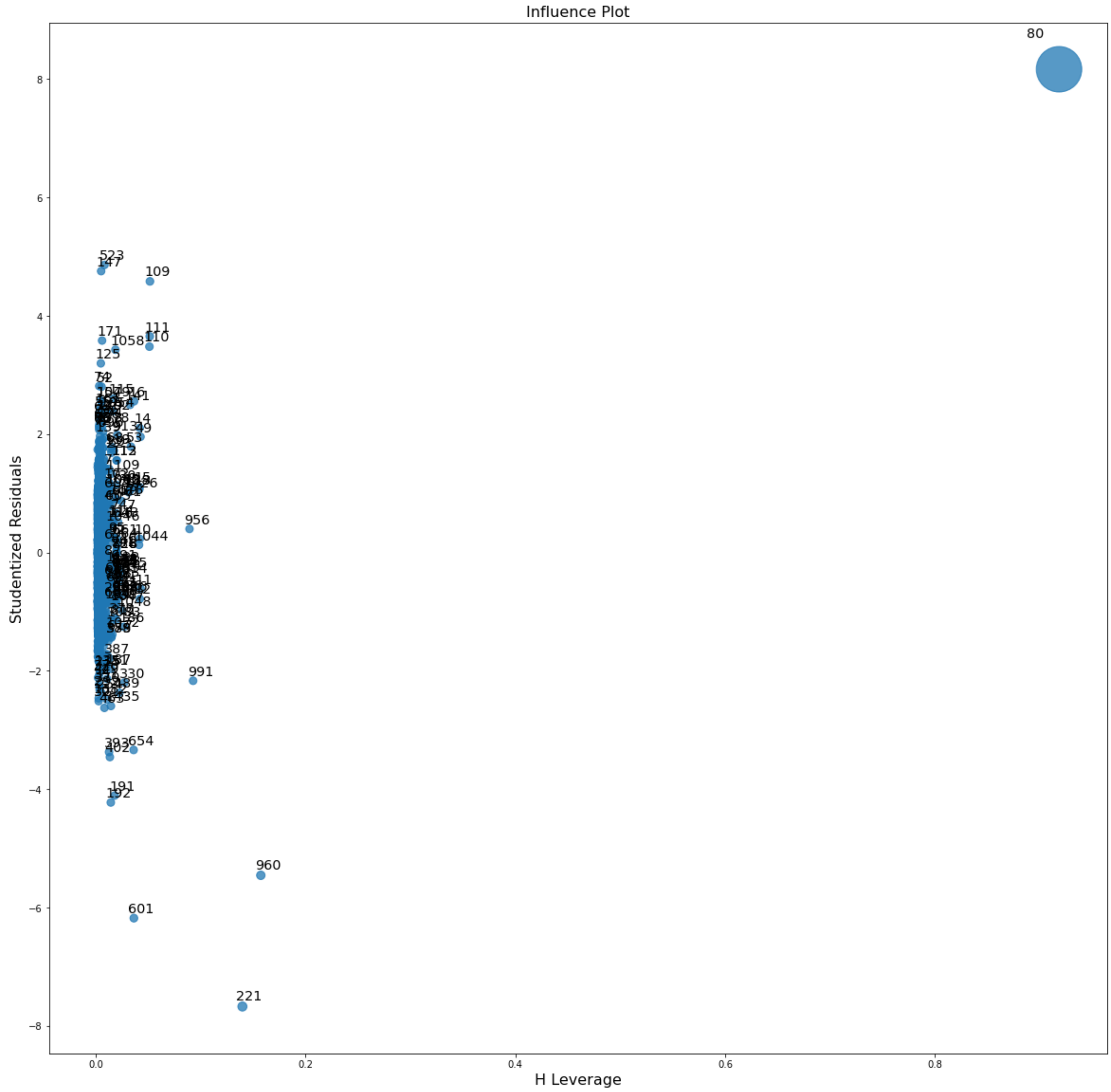
```
np.argmax(C),np.max(C)
```

Out[37]:

```
(80, 79.52010624138181)
```

In [38]:

```
#2.Leverage value using high influence points
fig,ax=plt.subplots(figsize=(20,20))
fig=influence_plot(model, ax = ax)
```



In [39]:

```
# Levarage Cutoff values
k=car_data.shape[1]
n=car_data.shape[0]
levarage_cutoff=(3*(k+1))/n
levarage_cutoff
```

Out[39]:

```
0.020891364902506964
```

In [40]:

```
car_data[car_data.index.isin([80])]
```

Out[40]:

	Price	Age	KM	HP	CC	Doors	Gears	QT	Weight
80	18950	25	20019	110	16000	5	5	100	1180

In [41]:

```
# improving model
car_new = car_data.copy()
car_new
```

Out[41]:

	Price	Age	KM	HP	CC	Doors	Gears	QT	Weight
0	13500	23	46986	90	2000	3	5	210	1165
1	13750	23	72937	90	2000	3	5	210	1165
2	13950	24	41711	90	2000	3	5	210	1165
3	14950	26	48000	90	2000	3	5	210	1165
4	13750	30	38500	90	2000	3	5	210	1170
...	...	...	...	...	...	...	...	...	...
1431	7500	69	20544	86	1300	3	5	69	1025
1432	10845	72	19000	86	1300	3	5	69	1015
1433	8500	71	17016	86	1300	3	5	69	1015
1434	7250	70	16916	86	1300	3	5	69	1015
1435	6950	76	1	110	1600	5	5	19	1114

1436 rows × 9 columns

In [42]:

```
car_data1 = car_new.drop(car_new.index[[80]], axis=0).reset_index()
car_data1
```

Out[42]:

	index	Price	Age	KM	HP	CC	Doors	Gears	QT	Weight
0	0	13500	23	46986	90	2000	3	5	210	1165
1	1	13750	23	72937	90	2000	3	5	210	1165
2	2	13950	24	41711	90	2000	3	5	210	1165
3	3	14950	26	48000	90	2000	3	5	210	1165
4	4	13750	30	38500	90	2000	3	5	210	1170
...	...	...	...	...	...	...	...	...	...	...
1430	1431	7500	69	20544	86	1300	3	5	69	1025
1431	1432	10845	72	19000	86	1300	3	5	69	1015
1432	1433	8500	71	17016	86	1300	3	5	69	1015
1433	1434	7250	70	16916	86	1300	3	5	69	1015
1434	1435	6950	76	1	110	1600	5	5	19	1114

1435 rows × 10 columns

In [43]:

```
car_data1 = car_data1.drop(['index'], axis=1)
car_data1
```

Out[43]:

	Price	Age	KM	HP	CC	Doors	Gears	QT	Weight
0	13500	23	46986	90	2000	3	5	210	1165
1	13750	23	72937	90	2000	3	5	210	1165
2	13950	24	41711	90	2000	3	5	210	1165
3	14950	26	48000	90	2000	3	5	210	1165
4	13750	30	38500	90	2000	3	5	210	1170
...	...	...	...	...	...	...	...	...	...
1430	7500	69	20544	86	1300	3	5	69	1025
1431	10845	72	19000	86	1300	3	5	69	1015
1432	8500	71	17016	86	1300	3	5	69	1015
1433	7250	70	16916	86	1300	3	5	69	1015
1434	6950	76	1	110	1600	5	5	19	1114

1435 rows × 9 columns

In [55]:

```
while np.max(C) > 0.5:
    model = smf.ols('Price~Age+KM+HP+CC+Doors+Gears+QT+Weight',data=car_data1).fit()
    (C,_)=model.get_influence().cooks_distance
    C
    np.argmax(C),np.max(C)
    car_data1 = car_data1.drop(car_data1.index[[np.argmax(C)]], axis=1).reset_index()
    car_data1
else:
    final_model = smf.ols('Price~Age+KM+HP+CC+Doors+Gears+QT+Weight',data=car_data1).fit()
    final_model.rsquared,final_model.aic
    print('Thus model accuracy is improved to', final_model.rsquared)
```

```
Thus model accuracy is improved to 0.8851845904421739
```

In [57]:

```
if np.max(C)>0.5 :
    model = smf.ols('Price~Age+KM+HP+CC+Doors+Gears+QT+Weight',data=car_data1).fit()
    (C,_)=model.get_influence().cooks_distance
    C
    np.argmax(C),np.max(C)
    car_data1=car_data1.drop(car_data1.index[[np.argmax(C)]],axis=1).reset_index()
    car_data1
elif np.max(C)<0.5:
    final_model=smf.ols('Price~Age+KM+HP+CC+Doors+Gears+QT+Weight',data=car_data1).fit()
    final_model.rsquared,final_model.aic
    print('Thus model accuracy is improved to',final_model.rsquared)
```

```
Thus model accuracy is improved to 0.8851845904421739
```

In [58]:

```
final_model.rsquared
```

Out[58]:

```
0.8851845904421739
```

In [59]:

```
car_data1
```

Out[59]:

	level_0	index	Price	Age	KM	HP	CC	Doors	Gears	QT	Weight
0	0	0	13500	23	46986	90	2000	3	5	210	1165
1	1	1	13750	23	72937	90	2000	3	5	210	1165
2	2	2	13950	24	41711	90	2000	3	5	210	1165
3	3	3	14950	26	48000	90	2000	3	5	210	1165
4	4	4	13750	30	38500	90	2000	3	5	210	1170
...	...	...	...	...	...	...	...	...	...	...	...
1428	1429	1430	7500	69	20544	86	1300	3	5	69	1025
1429	1430	1431	10845	72	19000	86	1300	3	5	69	1015
1430	1431	1432	8500	71	17016	86	1300	3	5	69	1015
1431	1432	1433	7250	70	16916	86	1300	3	5	69	1015
1432	1433	1434	6950	76	1	110	1600	5	5	19	1114

1433 rows × 11 columns

In [70]:

```
# Model prediction for new data
new_data = pd.DataFrame({'Age':15,'KM':50000,'HP':90,'CC':1400,'Doors':4,'Gears':5,'QT':210
new_data
```

Out[70]:

	Age	KM	HP	CC	Doors	Gears	QT	Weight
0	15	50000	90	1400	4	5	210	1165

In [72]:

```
final_model.predict(new_data)
```

Out[72]:

```
0    19332.917337
dtype: float64
```

In [74]:

```
pred_y = final_model.predict(car_data1)
pred_y
```

Out[74]:

```
0       16333.273814
1       15892.326850
2       16310.886081
3       15979.990390
4       15846.536733
            ...
1428     9115.435074
1429     8499.218117
1430     8644.947302
1431     8758.664462
1432    10641.521002
Length: 1433, dtype: float64
```

In [ ]: