In [1]:

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
```

In [3]:

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

Out[3]:

	Id	Model	Price	Age_08_04	Mfg_Month	Mfg_Year	KM	Fuel_Type	HP	Me
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	Me
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 [4]:

```
toyota_new = toyota_data.loc[:,['Price','Age_08_04', 'KM', 'HP','cc','Doors','Gears', 'Quar
toyota_new
```

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]:

```
toyota_new[toyota_new.duplicated()]
```

Out[5]:

	Price	Age_08_04	KM	HP	cc	Doors	Gears	Quarterly_Tax	Weight
113	24950	8	13253	116	2000	5	5	234	1320

In [6]:

```
toyota_2 = toyota_new.drop_duplicates().reset_index(drop = True)
toyota_2
```

Out[6]:

	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
...	...	...	...	...	...	...	...	...	...
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 [7]:

```
toyota_2.describe()
```

Out[7]:

	Price	Age_08_04	KM	HP	cc	Doors	
count	1435.000000	1435.000000	1435.000000	1435.000000	1435.000000	1435.000000	143
mean	10720.915679	55.980488	68571.782578	101.491986	1576.560976	4.032753	5
std	3608.732978	18.563312	37491.094553	14.981408	424.387533	0.952667	(
min	4350.000000	1.000000	1.000000	69.000000	1300.000000	2.000000	3
25%	8450.000000	44.000000	43000.000000	90.000000	1400.000000	3.000000	5
50%	9900.000000	61.000000	63451.000000	110.000000	1600.000000	4.000000	5
75%	11950.000000	70.000000	87041.500000	110.000000	1600.000000	5.000000	5
max	32500.000000	80.000000	243000.000000	192.000000	16000.000000	5.000000	6

In [12]:

```
toyota_2.dtypes
```

Out[12]:

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

In [13]:

```
toyota_2.shape
```

Out[13]:

```
(1435, 9)
```

In [14]:

```
toyota_2.isna().sum()
```

Out[14]:

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

In [9]:

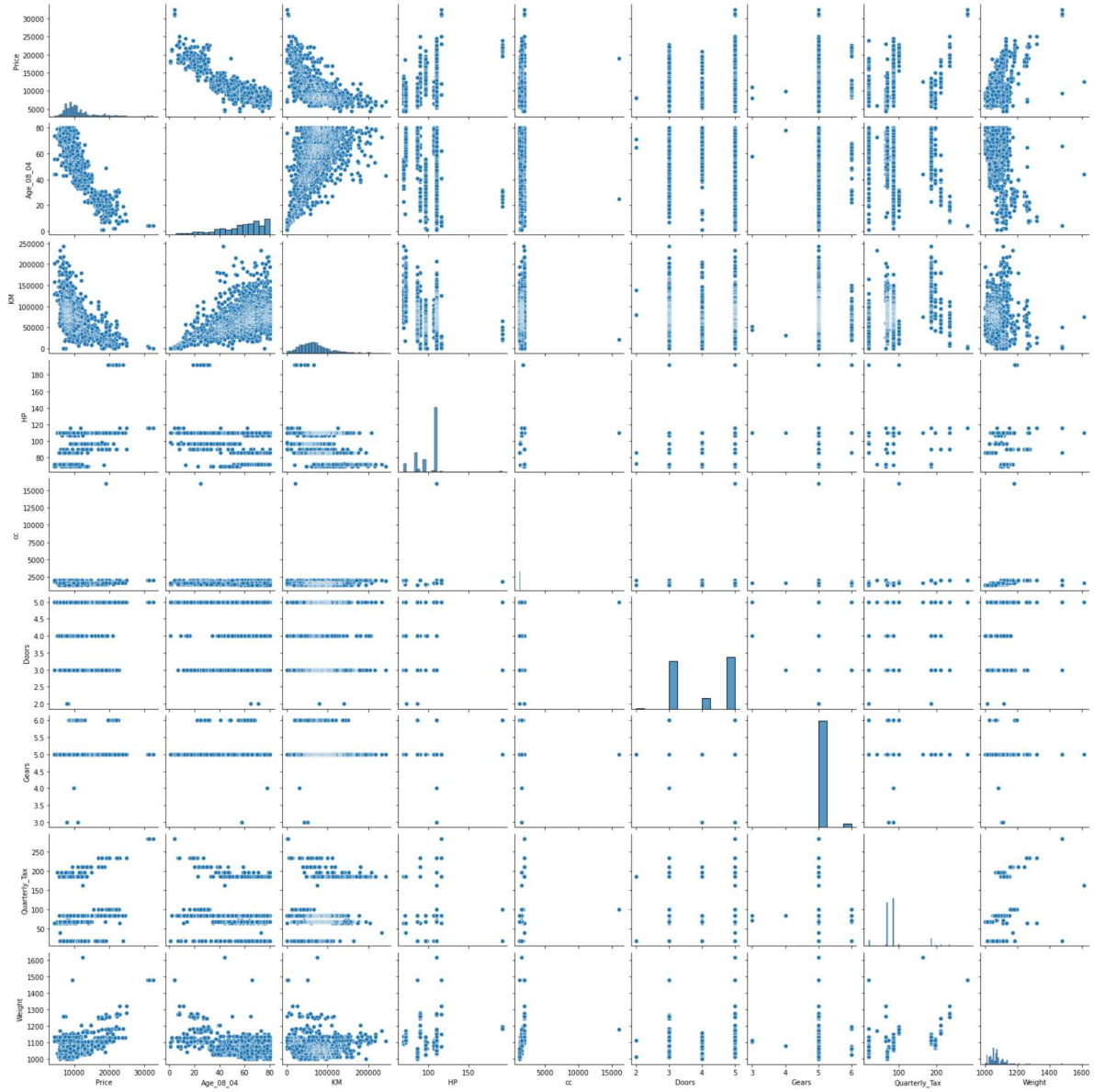
```
#Correlation Analysis
toyota_2.corr()
```

Out[9]:

	Price	Age_08_04	KM	HP	cc	Doors	Gears	Qua
Price	1.000000	-0.876273	-0.569420	0.314134	0.124375	0.183604	0.063831	
Age_08_04	-0.876273	1.000000	0.504575	-0.155293	-0.096549	-0.146929	-0.005629	
KM	-0.569420	0.504575	1.000000	-0.332904	0.103822	-0.035193	0.014890	
HP	0.314134	-0.155293	-0.332904	1.000000	0.035207	0.091803	0.209642	
cc	0.124375	-0.096549	0.103822	0.035207	1.000000	0.079254	0.014732	
Doors	0.183604	-0.146929	-0.035193	0.091803	0.079254	1.000000	-0.160101	
Gears	0.063831	-0.005629	0.014890	0.209642	0.014732	-0.160101	1.000000	
Quarterly_Tax	0.211508	-0.193319	0.283312	-0.302287	0.305982	0.107353	-0.005125	
Weight	0.575869	-0.466484	-0.023969	0.087143	0.335077	0.301734	0.021238	

In [10]:

```
sns.pairplot(toyota_2)
plt.show()
```



Model Building

In [16]:

```
import statsmodels.formula.api as smf
```

In [20]:

```
model=smf.ols('Price~Age_08_04+KM+HP+cc+Doors+Gears+Quarterly_Tax+Weight', data = toyota_2)
print('R2 score                :', model.rsquared)
print('Adjusted R2 score       :', model.rsquared_adj)
print('AIC score               :', model.aic)
print('BIC score               :', model.bic)
```

```
R2 score                : 0.8625200256947
Adjusted R2 score       : 0.8617487495415146
AIC score               : 24750.407207346932
BIC score               : 24797.827488500676
```

In [21]:

```
model.summary()
```

Out[21]:

OLS Regression Results

Dep. Variable:	Price	**R-squared:**	0.863
Model:	OLS	**Adj. R-squared:**	0.862
Method:	Least Squares	**F-statistic:**	1118.
Date:	Sun, 24 Oct 2021	**Prob (F-statistic):**	0.00
Time:	13:44:35	**Log-Likelihood:**	-12366.
No. Observations:	1435	**AIC:**	2.475e+04
Df Residuals:	1426	**BIC:**	2.480e+04
Df Model:	8		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-5472.5404	1412.169	-3.875	0.000	-8242.692	-2702.389
Age_08_04	-121.7139	2.615	-46.552	0.000	-126.843	-116.585
KM	-0.0207	0.001	-16.552	0.000	-0.023	-0.018
HP	31.5846	2.818	11.210	0.000	26.058	37.112
cc	-0.1186	0.090	-1.316	0.188	-0.295	0.058
Doors	-0.9202	39.988	-0.023	0.982	-79.362	77.522
Gears	597.7159	196.969	3.035	0.002	211.335	984.097
Quarterly_Tax	3.8588	1.311	2.944	0.003	1.288	6.430
Weight	16.8555	1.069	15.761	0.000	14.758	18.953

Omnibus:	149.666	**Durbin-Watson:**	1.544
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	1000.538
Skew:	-0.204	**Prob(JB):**	5.44e-218
Kurtosis:	7.070	**Cond. No.**	3.13e+06

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.13e+06. This might indicate that there are
strong multicollinearity or other numerical problems.

Model Testing

In [22]:

```
# Finding Coefficient parameters
model.params
```

Out[22]:

```
Intercept       -5472.540368
Age_08_04        -121.713891
KM                 -0.020737
HP                 31.584612
cc                 -0.118558
Doors              -0.920189
Gears             597.715894
Quarterly_Tax       3.858805
Weight             16.855470
dtype: float64
```

In [24]:

```
# Finding tvalues and pvalues
model.tvalues, round(model.pvalues,5)
```

Out[24]:

```
(Intercept        -3.875273
 Age_08_04       -46.551876
 KM              -16.552424
 HP               11.209719
 cc               -1.316436
 Doors            -0.023012
 Gears             3.034563
 Quarterly_Tax     2.944198
 Weight           15.760663
 dtype: float64,
 Intercept        0.00011
 Age_08_04        0.00000
 KM               0.00000
 HP               0.00000
 cc               0.18824
 Doors            0.98164
 Gears            0.00245
 Quarterly_Tax    0.00329
 Weight           0.00000
 dtype: float64)
```

In [ ]:

```
# Build SLR and MLR models for insignificant variables 'CC' and 'Doors'
# Also find their tvalues and pvalues
```

In [26]:

```
model_1 = smf.ols('Price~cc', data=toyota_2).fit()
model_1.tvalues , model_1.pvalues #cc has significant pvalue
```

Out[26]:

```
(Intercept    24.879592
 cc            4.745039
 dtype: float64,
 Intercept    7.236022e-114
 cc            2.292856e-06
 dtype: float64)
```

In [27]:

```
model_2 = smf.ols('Price~Doors', data=toyota_2).fit()
model_2.tvalues , model_2.pvalues #Door has significant pvalue
```

Out[27]:

```
(Intercept    19.421546
 Doors         7.070520
 dtype: float64,
 Intercept    8.976407e-75
 Doors        2.404166e-12
 dtype: float64)
```

In [28]:

```
model_3 = smf.ols('Price~Doors+cc', data=toyota_2).fit()
model_3.tvalues , model_2.pvalues #cc & Door have significant pvalue
```

Out[28]:

```
(Intercept    12.786341
 Doors         6.752236
 cc            4.268006
 dtype: float64,
 Intercept    8.976407e-75
 Doors        2.404166e-12
 dtype: float64)
```

Understanding R2 and Adjusted R2.

In [30]:

```
model_4 = smf.ols('Price~Age_08_04', data = toyota_2).fit()
print('R2 score                :', model_4.rsquared)
print('Adjusted R2 score       :', model_4.rsquared_adj)
print('AIC score               :', model_4.aic)
print('BIC score               :', model_4.bic)
```

```
R2 score                : 0.7678550373053038
Adjusted R2 score       : 0.7676930380291735
AIC score               : 25488.18038953394
BIC score               : 25498.718229790327
```

In [31]:

```
model_5=smf.ols('Price~Age_08_04+KM', data = toyota_2).fit()
print('R2 score                :', model_5.rsquared)
print('Adjusted R2 score       :', model_5.rsquared_adj)
print('AIC score               :', model_5.aic)
print('BIC score               :', model_5.bic)
```

```
R2 score                : 0.7895865358054202
Adjusted R2 score       : 0.7892926622520757
AIC score               : 25349.13776446717
BIC score               : 25364.94452485175
```

In [32]:

```
model_6=smf.ols('Price~Age_08_04+KM+HP', data = toyota_2).fit()
print('R2 score                :', model_6.rsquared)
print('Adjusted R2 score       :', model_6.rsquared_adj)
print('AIC score               :', model_6.aic)
print('BIC score               :', model_6.bic)
```

```
R2 score                : 0.8099643382575972
Adjusted R2 score       : 0.8095659406438814
AIC score               : 25204.96475523018
BIC score               : 25226.040435742954
```

In [33]:

```
model_7=smf.ols('Price~Age_08_04+KM+HP+cc', data = toyota_2).fit()
print('R2 score                :', model_7.rsquared)
print('Adjusted R2 score       :', model_7.rsquared_adj)
print('AIC score               :', model_7.aic)
print('BIC score               :', model_7.bic)
```

```
R2 score                : 0.8131215637158065
Adjusted R2 score       : 0.8125988268310954
AIC score               : 25182.923603541938
BIC score               : 25209.268204182907
```

In [34]:

```
model_8=smf.ols('Price~Age_08_04+KM+HP+cc+Doors', data = toyota_2).fit()
print('R2 score                :', model_8.rsquared)
print('Adjusted R2 score       :', model_8.rsquared_adj)
print('AIC score               :', model_8.aic)
print('BIC score               :', model_8.bic)
```

```
R2 score                : 0.8153256441556611
Adjusted R2 score       : 0.8146794777601246
AIC score               : 25167.898337691102
BIC score               : 25199.511858460264
```

In [35]:

```
model_9=smf.ols('Price~Age_08_04+KM+HP+cc+Doors+Gears', data = toyota_2).fit()
print('R2 score                :', model_9.rsquared)
print('Adjusted R2 score       :', model_9.rsquared_adj)
print('AIC score               :', model_9.aic)
print('BIC score               :', model_9.bic)
```

```
R2 score                : 0.8169449510404598
Adjusted R2 score       : 0.8161758121792853
AIC score               : 25157.26012943683
BIC score               : 25194.142570334185
```

In [36]:

```
model_10=smf.ols('Price~Age_08_04+KM+HP+cc+Doors+Gears+Quarterly_Tax', data = toyota_2).fit
print('R2 score                :', model_10.rsquared)
print('Adjusted R2 score       :', model_10.rsquared_adj)
print('AIC score               :', model_10.aic)
print('BIC score               :', model_10.bic)
```

```
R2 score                : 0.8385720455246246
Adjusted R2 score       : 0.8377801774928604
AIC score               : 24978.840477013637
BIC score               : 25020.991838039186
```

In [37]:

```
model_4.summary()
```

Out[37]:

OLS Regression Results

Dep. Variable:	Price	**R-squared:**	0.768
Model:	OLS	**Adj. R-squared:**	0.768
Method:	Least Squares	**F-statistic:**	4740.
Date:	Sun, 24 Oct 2021	**Prob (F-statistic):**	0.00
Time:	14:07:51	**Log-Likelihood:**	-12742.
No. Observations:	1435	**AIC:**	2.549e+04
Df Residuals:	1433	**BIC:**	2.550e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	2.026e+04	145.926	138.818	0.000	2e+04	2.05e+04
Age_08_04	-170.3487	2.474	-68.847	0.000	-175.202	-165.495

Omnibus:	363.006	**Durbin-Watson:**	1.223
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	2878.814
Skew:	0.951	**Prob(JB):**	0.00
Kurtosis:	9.673	**Cond. No.**	187.

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

In [39]:

```
model_5.summary()
```

Out[39]:

OLS Regression Results

Dep. Variable:	Price	**R-squared:**	0.790
Model:	OLS	**Adj. R-squared:**	0.789
Method:	Least Squares	**F-statistic:**	2687.
Date:	Sun, 24 Oct 2021	**Prob (F-statistic):**	0.00
Time:	14:08:07	**Log-Likelihood:**	-12672.
No. Observations:	1435	**AIC:**	2.535e+04
Df Residuals:	1432	**BIC:**	2.536e+04
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	2.045e+04	139.846	146.207	0.000	2.02e+04	2.07e+04
Age_08_04	-153.6003	2.729	-56.276	0.000	-158.954	-148.246
KM	-0.0164	0.001	-12.161	0.000	-0.019	-0.014

Omnibus:	453.671	**Durbin-Watson:**	1.302
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	3455.391
Skew:	1.259	**Prob(JB):**	0.00
Kurtosis:	10.173	**Cond. No.**	2.50e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.5e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

In [40]:

```
model_6.summary()
```

Out[40]:

OLS Regression Results

Dep. Variable:	Price	**R-squared:**	0.810
Model:	OLS	**Adj. R-squared:**	0.810
Method:	Least Squares	**F-statistic:**	2033.
Date:	Sun, 24 Oct 2021	**Prob (F-statistic):**	0.00
Time:	14:08:17	**Log-Likelihood:**	-12598.
No. Observations:	1435	**AIC:**	2.520e+04
Df Residuals:	1431	**BIC:**	2.523e+04
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1.643e+04	350.302	46.908	0.000	1.57e+04	1.71e+04
Age_08_04	-154.1011	2.595	-59.382	0.000	-159.192	-149.010
KM	-0.0115	0.001	-8.512	0.000	-0.014	-0.009
HP	36.4703	2.944	12.387	0.000	30.695	42.246

Omnibus:	421.706	**Durbin-Watson:**	1.361
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	3120.040
Skew:	1.163	**Prob(JB):**	0.00
Kurtosis:	9.839	**Cond. No.**	6.59e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 6.59e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

In [41]:

```
model_7.summary()
```

Out[41]:

OLS Regression Results

Dep. Variable:	Price	**R-squared:**	0.813
Model:	OLS	**Adj. R-squared:**	0.813
Method:	Least Squares	**F-statistic:**	1556.
Date:	Sun, 24 Oct 2021	**Prob (F-statistic):**	0.00
Time:	14:08:40	**Log-Likelihood:**	-12586.
No. Observations:	1435	**AIC:**	2.518e+04
Df Residuals:	1430	**BIC:**	2.521e+04
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1.574e+04	375.144	41.950	0.000	1.5e+04	1.65e+04
Age_08_04	-151.8506	2.615	-58.075	0.000	-156.980	-146.721
KM	-0.0127	0.001	-9.366	0.000	-0.015	-0.010
HP	35.3430	2.930	12.064	0.000	29.596	41.090
cc	0.4893	0.100	4.915	0.000	0.294	0.685

Omnibus:	404.525	**Durbin-Watson:**	1.364
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	2991.152
Skew:	1.107	**Prob(JB):**	0.00
Kurtosis:	9.717	**Cond. No.**	7.11e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 7.11e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

In [42]:

```
model_8.summary()
```

Out[42]:

OLS Regression Results

Dep. Variable:	Price	**R-squared:**	0.815
Model:	OLS	**Adj. R-squared:**	0.815
Method:	Least Squares	**F-statistic:**	1262.
Date:	Sun, 24 Oct 2021	**Prob (F-statistic):**	0.00
Time:	14:08:42	**Log-Likelihood:**	-12578.
No. Observations:	1435	**AIC:**	2.517e+04
Df Residuals:	1429	**BIC:**	2.520e+04
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1.508e+04	405.423	37.200	0.000	1.43e+04	1.59e+04
Age_08_04	-150.3257	2.626	-57.239	0.000	-155.477	-145.174
KM	-0.0131	0.001	-9.645	0.000	-0.016	-0.010
HP	34.3275	2.924	11.741	0.000	28.592	40.063
cc	0.4678	0.099	4.719	0.000	0.273	0.662
Doors	180.9328	43.812	4.130	0.000	94.991	266.875

Omnibus:	400.723	**Durbin-Watson:**	1.352
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	3000.146
Skew:	1.091	**Prob(JB):**	0.00
Kurtosis:	9.739	**Cond. No.**	7.73e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 7.73e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

In [43]:

```
model_9.summary()
```

Out[43]:

OLS Regression Results

Dep. Variable:	Price	**R-squared:**	0.817
Model:	OLS	**Adj. R-squared:**	0.816
Method:	Least Squares	**F-statistic:**	1062.
Date:	Sun, 24 Oct 2021	**Prob (F-statistic):**	0.00
Time:	14:08:45	**Log-Likelihood:**	-12572.
No. Observations:	1435	**AIC:**	2.516e+04
Df Residuals:	1428	**BIC:**	2.519e+04
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1.119e+04	1167.571	9.582	0.000	8897.584	1.35e+04
Age_08_04	-149.8834	2.619	-57.238	0.000	-155.020	-144.747
KM	-0.0136	0.001	-9.996	0.000	-0.016	-0.011
HP	31.7027	3.004	10.553	0.000	25.810	37.595
cc	0.4669	0.099	4.729	0.000	0.273	0.661
Doors	210.8740	44.440	4.745	0.000	123.699	298.049
Gears	805.7003	226.692	3.554	0.000	361.016	1250.385

Omnibus:	410.365	**Durbin-Watson:**	1.348
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	3172.449
Skew:	1.112	**Prob(JB):**	0.00
Kurtosis:	9.936	**Cond. No.**	2.27e+06

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.27e+06. This might indicate that there are
strong multicollinearity or other numerical problems.

In [44]:

```
model_10.summary()
```

Out[44]:

OLS Regression Results

Dep. Variable:	Price	**R-squared:**	0.839
Model:	OLS	**Adj. R-squared:**	0.838
Method:	Least Squares	**F-statistic:**	1059.
Date:	Sun, 24 Oct 2021	**Prob (F-statistic):**	0.00
Time:	14:08:49	**Log-Likelihood:**	-12481.
No. Observations:	1435	**AIC:**	2.498e+04
Df Residuals:	1427	**BIC:**	2.502e+04
Df Model:	7		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	9990.3469	1100.231	9.080	0.000	7832.103	1.21e+04
Age_08_04	-136.3300	2.648	-51.483	0.000	-141.524	-131.136
KM	-0.0200	0.001	-14.715	0.000	-0.023	-0.017
HP	43.2997	2.944	14.708	0.000	37.525	49.075
cc	0.1085	0.096	1.127	0.260	-0.080	0.297
Doors	157.5018	41.925	3.757	0.000	75.260	239.744
Gears	625.4171	213.353	2.931	0.003	206.898	1043.936
Quarterly_Tax	15.9298	1.152	13.827	0.000	13.670	18.190

Omnibus:	218.215	**Durbin-Watson:**	1.374
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	813.088
Skew:	0.705	**Prob(JB):**	2.75e-177
Kurtosis:	6.408	**Cond. No.**	2.28e+06

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.28e+06. This might indicate that there are
strong multicollinearity or other numerical problems.

Model Improvement Techniques

Log Transformation

In [47]:

```
toyota_2
```

Out[47]:

	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
...	...	...	...	...	...	...	...	...	...
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 [48]:

```
toyota_2['log_Age_08_04'] = np.log(toyota_2['Age_08_04'])
toyota_2['log_KM'] = np.log(toyota_2['KM'])
toyota_2['log_HP'] = np.log(toyota_2['HP'])
toyota_2['log_cc'] = np.log(toyota_2['cc'])
toyota_2['log_Doors'] = np.log(toyota_2['Doors'])
toyota_2['log_Gears'] = np.log(toyota_2['Gears'])
toyota_2['log_Quarterly_Tax'] = np.log(toyota_2['Quarterly_Tax'])
toyota_2['log_Weight'] = np.log(toyota_2['Weight'])
toyota_2
```

Out[48]:

	Price	Age_08_04	KM	HP	cc	Doors	Gears	Quarterly_Tax	Weight	log_Age_08_(
0	13500	23	46986	90	2000	3	5	210	1165	3.1354
1	13750	23	72937	90	2000	3	5	210	1165	3.1354
2	13950	24	41711	90	2000	3	5	210	1165	3.1780
3	14950	26	48000	90	2000	3	5	210	1165	3.2580
4	13750	30	38500	90	2000	3	5	210	1170	3.4011
...	...	...	...	...	...	...	...	...	...	
1430	7500	69	20544	86	1300	3	5	69	1025	4.2341(
1431	10845	72	19000	86	1300	3	5	69	1015	4.2766(
1432	8500	71	17016	86	1300	3	5	69	1015	4.2626(
1433	7250	70	16916	86	1300	3	5	69	1015	4.2484
1434	6950	76	1	110	1600	5	5	19	1114	4.3307

1435 rows × 17 columns

In [54]:

```
toyota_log = toyota_2.loc[:,['Price','log_Age_08_04','log_KM','log_HP','log_cc','log_Doors'
toyota_log
```

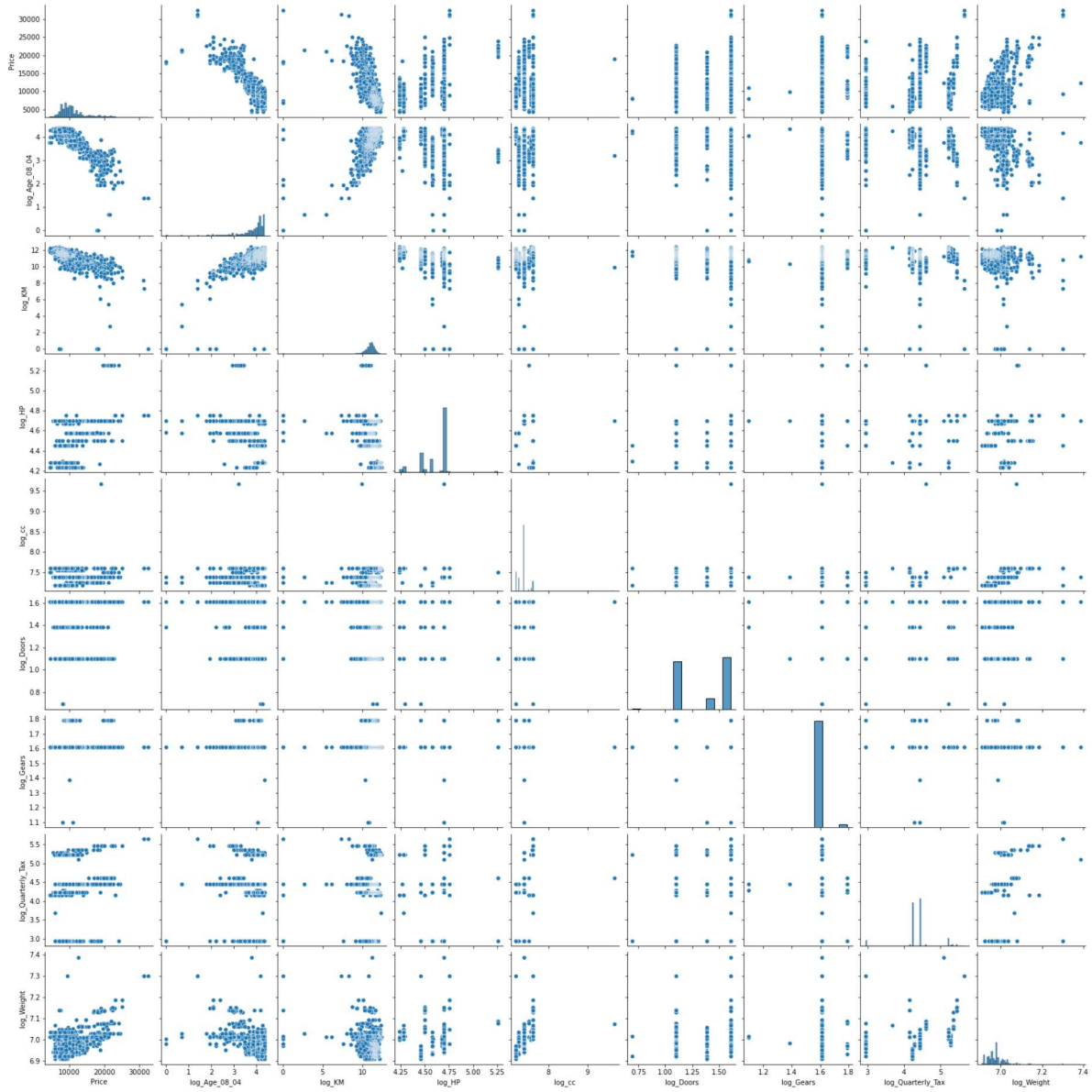
Out[54]:

	Price	log_Age_08_04	log_KM	log_HP	log_cc	log_Doors	log_Gears	log_Quarter
0	13500	3.135494	10.757605	4.499810	7.600902	1.098612	1.609438	5.3
1	13750	3.135494	11.197351	4.499810	7.600902	1.098612	1.609438	5.3
2	13950	3.178054	10.638520	4.499810	7.600902	1.098612	1.609438	5.3
3	14950	3.258097	10.778956	4.499810	7.600902	1.098612	1.609438	5.3
4	13750	3.401197	10.558414	4.499810	7.600902	1.098612	1.609438	5.3
...	...	...	...	...	...	...	...	
1430	7500	4.234107	9.930324	4.454347	7.170120	1.098612	1.609438	4.2
1431	10845	4.276666	9.852194	4.454347	7.170120	1.098612	1.609438	4.2
1432	8500	4.262680	9.741909	4.454347	7.170120	1.098612	1.609438	4.2
1433	7250	4.248495	9.736015	4.454347	7.170120	1.098612	1.609438	4.2
1434	6950	4.330733	0.000000	4.700480	7.377759	1.609438	1.609438	2.9

1435 rows × 9 columns

In [55]:

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



Understanding R2 and Adjusted R2

In [56]:

```
model_11 = smf.ols('Price~log_Age_08_04', data = toyota_log).fit()
print('R2 score                :', model_11.rsquared)
print('Adjusted R2 score       :', model_11.rsquared_adj)
print('AIC score               :', model_11.aic)
print('BIC score               :', model_11.bic)
```

```
R2 score                : 0.765867523101288
Adjusted R2 score       : 0.7657041368647921
AIC score               : 25500.41387935763
BIC score               : 25510.95171961402
```

In [57]:

```
model_12=smf.ols('Price~log_Age_08_04+log_KM', data = toyota_log).fit()
print('R2 score                :', model_12.rsquared)
print('Adjusted R2 score       :', model_12.rsquared_adj)
print('AIC score               :', model_12.aic)
print('BIC score               :', model_12.bic)
```

```
R2 score                : 0.7680962768829682
Adjusted R2 score       : 0.7677723890015198
AIC score               : 25488.688395974998
BIC score               : 25504.49515635958
```

In [58]:

```
model_13=smf.ols('Price~log_Age_08_04+log_KM+log_HP', data = toyota_log).fit()
print('R2 score                :', model_13.rsquared)
print('Adjusted R2 score       :', model_13.rsquared_adj)
print('AIC score               :', model_13.aic)
print('BIC score               :', model_13.bic)
```

```
R2 score                : 0.8002964361804118
Adjusted R2 score       : 0.7998777704281694
AIC score               : 25276.17281054707
BIC score               : 25297.248491059843
```

In [59]:

```
model_14=smf.ols('Price~log_Age_08_04+log_KM+log_HP+log_cc', data = toyota_log).fit()
print('R2 score                :', model_14.rsquared)
print('Adjusted R2 score       :', model_14.rsquared_adj)
print('AIC score               :', model_14.aic)
print('BIC score               :', model_14.bic)
```

```
R2 score                : 0.8023466668997267
Adjusted R2 score       : 0.8017937904435022
AIC score               : 25263.364424364947
BIC score               : 25289.709025005915
```

In [61]:

```
model_15=smf.ols('Price~log_Age_08_04+log_KM+log_HP+log_cc+log_Doors', data = toyota_log).f
print('R2 score                :', model_15.rsquared)
print('Adjusted R2 score       :', model_15.rsquared_adj)
print('AIC score               :', model_15.aic)
print('BIC score               :', model_15.bic)
```

```
R2 score                : 0.8029732657035262
Adjusted R2 score       : 0.8022838789495148
AIC score               : 25260.807974150273
BIC score               : 25292.421494919436
```

In [62]:

```
model_16=smf.ols('Price~log_Age_08_04+log_KM+log_HP+log_cc+log_Doors+log_Gears', data = toy
print('R2 score                :', model_16.rsquared)
print('Adjusted R2 score       :', model_16.rsquared_adj)
print('AIC score               :', model_16.aic)
print('BIC score               :', model_16.bic)
```

```
R2 score                : 0.80467330166521
Adjusted R2 score       : 0.8038526012520386
AIC score               : 25250.37241646597
BIC score               : 25287.254857363325
```

In [63]:

```
model_17=smf.ols('Price~log_Age_08_04+log_KM+log_HP+log_cc+log_Doors+log_Gears+log_Quarterl
print('R2 score                :', model_17.rsquared)
print('Adjusted R2 score       :', model_17.rsquared_adj)
print('AIC score               :', model_17.aic)
print('BIC score               :', model_17.bic)
```

```
R2 score                : 0.8124799146426903
Adjusted R2 score       : 0.8115600543781485
AIC score               : 25193.842251817387
BIC score               : 25235.993612842936
```

In [64]:

```
model_18=smf.ols('Price~log_Age_08_04+log_KM+log_HP+log_cc+log_Doors+log_Gears+log_Quarterl
print('R2 score                :', model_18.rsquared)
print('Adjusted R2 score       :', model_18.rsquared_adj)
print('AIC score               :', model_18.aic)
print('BIC score               :', model_18.bic)
```

```
R2 score                : 0.8334918891624676
Adjusted R2 score       : 0.8325577623134492
AIC score               : 25025.30406848275
BIC score               : 25072.724349636494
```

Using sklearn library

In [65]:

```
toyota_3 = toyota_2.iloc[:,:9]
toyota_3
```

Out[65]:

	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
...	...	...	...	...	...	...	...	...	...
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 [67]:

```
X = toyota_3.drop(labels='Price', axis=1)
y=toyota_3[['Price']]
```

In [69]:

```
from sklearn.linear_model import LinearRegression
```

In [70]:

```
linear_model=LinearRegression()
linear_model.fit(X,y)
```

Out[70]:

```
LinearRegression()
```

In [71]:

```
y_pred=linear_model.predict(X)
y_pred
```

Out[71]:

```
array([[16791.95887083],
       [16253.80041357],
       [16779.63521013],
       ...,
       [ 8455.43440191],
       [ 8579.22204117],
       [10396.0875261 ]])
```

In [72]:

```
error=y-y_pred
error
```

Out[72]:

	Price
0	-3291.958871
1	-2503.800414
2	-2829.635210
3	-1455.789389
4	-2450.217277
...	...
1430	-1294.255037
1431	2552.422658
1432	44.565598
1433	-1329.222041
1434	-3446.087526

1435 rows × 1 columns

In [73]:

```
from sklearn.metrics import mean_squared_error
```

In [74]:

```
mean_squared_error(y,y_pred)
```

Out[74]:

```
1789147.6788508205
```

In [75]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.20,random_state=12)
```

In [76]:

```
# Training data
X_train.shape , y_train.shape
```

Out[76]:

```
((1148, 8), (1148, 1))
```

In [94]:

```
# Testing data
y_test.shape , y_test.shape
```

Out[94]:

```
((287, 1), (287, 1))
```

In [95]:

```
from sklearn.linear_model import LinearRegression
```

In [96]:

```
linear_model_2 = LinearRegression()
linear_model_2.fit(X_train,y_train)
```

Out[96]:

```
LinearRegression()
```

In [99]:

```
# For Training data
y_train_pred = linear_model_2.predict(X_train)
y_train_pred
```

Out[99]:

```
array([[8599.92992111],
       [7750.10324152],
       [9364.037886  ],
       ...,
       [7978.8440309 ],
       [8950.04215709],
       [8762.89539268]])
```

In [101]:

```
mean_squared_error(y_train,y_train_pred)
```

Out[101]:

```
2229396.167451788
```

In [102]:

```
# For Testing data
y_test_pred = linear_model_2.predict(X_test)
y_test_pred
```

Out[102]:

```
array([[11540.50831185],
       [ 9026.89075291],
       [ 9724.63237261],
       [17667.87901072],
       [14706.92451751],
       [ 9469.05768515],
       [ 9855.80914989],
       [ 9128.95200787],
       [ 9115.70253678],
       [ 9839.38199006],
       [ 8560.93585712],
       [ 9130.79232794],
       [ 8879.27583137],
       [10888.49935351],
       [ 8962.50975712],
       [ 7903.26448258],
       [ 9280.36214369],
       [ 7590.2237871 ],
```

In [103]:

```
mean_squared_error(y_test,y_test_pred)
```

Out[103]:

```
1959825.7380623019
```

MSE for Log Transformation inputs

In [104]:

```
X= toyota_log.drop (labels = 'Price', axis=1)
y=toyota_log[['Price']]
```

In [105]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test= train_test_split(X,y,test_size=0.20,random_state=12)
```

In [106]:

```
#Training data
X_train.shape , y_train.shape
```

Out[106]:

```
((1148, 8), (1148, 1))
```

In [107]:

```
#Testing data
X_test.shape , y_test.shape
```

Out[107]:

```
((287, 8), (287, 1))
```

In [108]:

```
from sklearn.linear_model import LinearRegression
```

In [111]:

```
linear_model_3 = LinearRegression()
linear_model_3.fit(X_train, y_train)
```

Out[111]:

```
LinearRegression()
```

In [112]:

```
y_train_pred = linear_model_3.predict(X_train)
y_train_pred
```

Out[112]:

```
array([[8599.92992111],
       [7750.10324152],
       [9364.037886  ],
       ...,
       [7978.8440309 ],
       [8950.04215709],
       [8762.89539268]])
```

In [113]:

```
from sklearn.metrics import mean_squared_error
```

In [114]:

```
mean_squared_error(y_train,y_train_pred)
```

Out[114]:

```
2229396.167451788
```

In [115]:

```
y_test_pred = linear_model_3.predict(X_test)
y_test_pred
```

Out[115]:

```
array([[11540.50831185],
       [ 9026.89075291],
       [ 9724.63237261],
       [17667.87901072],
       [14706.92451751],
       [ 9469.05768515],
       [ 9855.80914989],
       [ 9128.95200787],
       [ 9115.70253678],
       [ 9839.38199006],
       [ 8560.93585712],
       [ 9130.79232794],
       [ 8879.27583137],
       [10888.49935351],
       [ 8962.50975712],
       [ 7903.26448258],
       [ 9280.36214369],
       [ 7590.2237871 ],
```

In [116]:

```
mean_squared_error(y_test,y_test_pred)
```

Out[116]:

```
1959825.7380623019
```

In [ ]: