ASSIGNMENT MULTIPLE REGRESSION

In [81]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn import linear_model
df = pd.read_csv("cars.csv")
df
```

Out[81]:

	Dimensions.Height	Dimensions.Length	Dimensions.Width	Engine Information.Driveline	Information.
0	140	143	202	All-wheel drive	Audi 3.2L 6 (250hp 2
1	140	143	202	Front-wheel drive	Audi 2.0L 4 (200 hp 2(
2	140	143	202	Front-wheel drive	Audi 2.0L 4 (200 hp 2(
3	140	143	202	All-wheel drive	Audi 2.0L 4 (200 hp 2(
4	140	143	202	All-wheel drive	Audi 2.0L 4 (200 hp 2(
...	...	...	...	...	
5071	13	253	201	Front-wheel drive	Hond Cylinder 250
5072	141	249	108	All-wheel drive	Lamborghini cylinder 552
5073	160	249	108	All-wheel drive	Lamborghini cylinder 552
5074	200	210	110	Rear-wheel drive	BMW cylinder 315 ft-lb
5075	200	94	110	Rear-wheel drive	BMW cylinder 315 ft-lb

5076 rows × 18 columns

In [82]:
```
1 df.describe()
```

Out[82]:

	Dimensions.Height	Dimensions.Length	Dimensions.Width	Engine Information.Number of Forward Gears	Informatic
count	5076.000000	5076.000000	5076.000000	5076.000000	5076.0
mean	145.632191	127.825847	144.012411	5.519110	17.2
std	62.125026	77.358295	79.925899	0.845637	4.4
min	1.000000	2.000000	1.000000	4.000000	8.0
25%	104.000000	60.000000	62.000000	5.000000	14.0
50%	152.000000	128.000000	158.000000	6.000000	17.0
75%	193.000000	198.000000	219.000000	6.000000	20.0
max	255.000000	255.000000	254.000000	8.000000	38.0

In [83]:
```
1 dimension=df.shape
2 print("DIMENSION=",dimension)
```

```
DIMENSION= (5076, 18)
```

In [84]:
```
numerical_columns=df.select_dtypes(include=['number'])
count=numerical_columns.shape[1]
print(count)
numerical_columns
```

```
9
```

Out[84]:

	Dimensions.Height	Dimensions.Length	Dimensions.Width	Engine Information.Number of Forward Gears	Informatior
0	140	143	202	6	
1	140	143	202	6	
2	140	143	202	6	
3	140	143	202	6	
4	140	143	202	6	
...	...	...	...	...	
5071	13	253	201	5	
5072	141	249	108	6	
5073	160	249	108	6	
5074	200	210	110	6	
5075	200	94	110	6	

5076 rows × 9 columns

In [85]:
```
categorical_columns=df.select_dtypes(include=['object', 'category'])
count=categorical_columns.shape
print(count)
categorical_columns
```

```
(5076, 8)
```

Out[85]:

	Engine Information.Driveline	Engine Information.Engine Type	Engine Information.Transmission	Fuel Information.Fuel Type	Identi
0	All-wheel drive	Audi 3.2L 6 cylinder 250hp 236ft-lbs	6 Speed Automatic Select Shift	Gasoline	
1	Front-wheel drive	Audi 2.0L 4 cylinder 200 hp 207 ft-lbs Turbo	6 Speed Automatic Select Shift	Gasoline	
2	Front-wheel drive	Audi 2.0L 4 cylinder 200 hp 207 ft-lbs Turbo	6 Speed Manual	Gasoline	
3	All-wheel drive	Audi 2.0L 4 cylinder 200 hp 207 ft-lbs Turbo	6 Speed Automatic Select Shift	Gasoline	
4	All-wheel drive	Audi 2.0L 4 cylinder 200 hp 207 ft-lbs Turbo	6 Speed Automatic Select Shift	Gasoline	
...	...	...	...	...	
5071	Front-wheel drive	Honda 3.5L 6 Cylinder 250 hp 253 ft-lbs	5 Speed Automatic	Gasoline	
5072	All-wheel drive	Lamborghini 5.2L 10 cylinder 552 hp 398 ft-lbs	6 Speed Manual	Gasoline	
5073	All-wheel drive	Lamborghini 5.2L 10 cylinder 552 hp 398 ft-lbs	6 Speed Manual	Gasoline	
5074	Rear-wheel drive	BMW 3.0L 6 cylinder 315hp 330 ft-lbs Turbo	6 Speed Automatic Select Shift	Gasoline	
5075	Rear-wheel drive	BMW 3.0L 6 cylinder 315hp 330 ft-lbs Turbo	6 Speed Automatic Select Shift	Gasoline	

5076 rows × 8 columns

In [86]:
```
missing_values = df.isnull().sum()
print("Missing values in each column:\n", missing_values)

```

```
Missing values in each column:
 Dimensions.Height                                  0
Dimensions.Length                                   0
Dimensions.Width                                    0
Engine Information.Driveline                        0
Engine Information.Engine Type                      0
Engine Information.Hybrid                           0
Engine Information.Number of Forward Gears          0
Engine Information.Transmission                     0
Fuel Information.City mpg                           0
Fuel Information.Fuel Type                          0
Fuel Information.Highway mpg                        0
Identification.Classification                       0
Identification.ID                                   0
Identification.Make                                 0
Identification.Model Year                           0
Identification.Year                                 0
Engine Information.Engine Statistics.Horsepower     0
Engine Information.Engine Statistics.Torque         0
dtype: int64
```

In [87]:
```
duplicates = df.duplicated()
num_duplicates = duplicates.sum()
print(f"Number of duplicate rows: {num_duplicates}")

duplicate_rows = df[duplicates]
print("Duplicate rows:\n", duplicate_rows)

```

```
(
4                                        2009 Audi A3                200
9
121                               2011 Nissan 370Z Coupe                201
1
1389  2010 Chevrolet Corvette Grand Sport Convertible                201
0
2203                                    2011 GMC Canyon                201
1
2343                                    2011 GMC Terrain                201
1
3569                                  2011 Toyota Tundra                201
1
3570                                  2011 Toyota Tundra                201
1
3640                               2010 Chevrolet Colorado                201
0
3703                                    2011 Ford Ranger                201
1
3704                            2010 Mercedes-Benz R-Class                201
1
```

In [88]:
```
# Remove duplicate rows and keep the first occurrence
df = df.drop_duplicates()
```

In [89]:
```
df
```

Out[89]:

	Dimensions.Height	Dimensions.Length	Dimensions.Width	Engine Information.Driveline	Informatio
0	140	143	202	All-wheel drive	Audi 3.2L 250hp
1	140	143	202	Front-wheel drive	Audi 2.0L 200 hp
2	140	143	202	Front-wheel drive	Audi 2.0L 200 hp
3	140	143	202	All-wheel drive	Audi 2.0L 200 hp
5	91	17	62	All-wheel drive	Audi 3.2L 265hp
...	...	...	...	...	
5071	13	253	201	Front-wheel drive	Hor Cylinder 2
5072	141	249	108	All-wheel drive	Lamborghi cylinder 5
5073	160	249	108	All-wheel drive	Lamborghi cylinder 5
5074	200	210	110	Rear-wheel drive	BM cylinder 3 ft
5075	200	94	110	Rear-wheel drive	BM cylinder 3 ft

5058 rows × 18 columns

In [90]:
```
df.dtypes
```

Out[90]:
```
Dimensions.Height                                 int64
Dimensions.Length                                 int64
Dimensions.Width                                  int64
Engine Information.Driveline                     object
Engine Information.Engine Type                   object
Engine Information.Hybrid                          bool
Engine Information.Number of Forward Gears        int64
Engine Information.Transmission                  object
Fuel Information.City mpg                         int64
Fuel Information.Fuel Type                       object
Fuel Information.Highway mpg                      int64
Identification.Classification                    object
Identification.ID                                object
Identification.Make                              object
Identification.Model Year                        object
Identification.Year                               int64
Engine Information.Engine Statistics.Horsepower   int64
Engine Information.Engine Statistics.Torque       int64
dtype: object
```

In [91]:
```
num_cols=['Dimensions.Height','Dimensions.Length','Dimensions.Width',
          'Engine Information.Number of Forward Gears','Fuel Information.
         'Fuel Information.Highway mpg','Identification.Year','Engine Inf
         'Engine Information.Engine Statistics.Torque']

cat_cols=['Engine Information.Driveline','Engine Information.Engine Type'
         'Engine Information.Transmission','Fuel Information.Fuel Type',
         'Identification.Classification','Identification.ID','Identificat

boolean_cols=['Engine Information.Hybrid']
```

In [92]:
```
df
```

Out[92]:

	Dimensions.Height	Dimensions.Length	Dimensions.Width	Engine Information.Driveline	Inform
0	140	143	202	All-wheel drive	Audi 2
1	140	143	202	Front-wheel drive	Audi 20
2	140	143	202	Front-wheel drive	Audi 20
3	140	143	202	All-wheel drive	Audi 20
5	91	17	62	All-wheel drive	Audi 2(

In [93]:
```
# import seaborn as sns
# sns.histplot(df[num_cols].values,kde=True)
```

In [94]:
```
#Checking outliers
for i in num_cols:
    q1=df[i].quantile(0.25)
    q3=df[i].quantile(0.75)
    iqr=q3-q1
    lowerbound=q1-1.5*iqr
    upperbound=q3+1.5*iqr
    df=df[(df[i]>=lowerbound) & (df[i]<=upperbound)]
df# outlier rows removed
```

Out[94]:

	Dimensions.Height	Dimensions.Length	Dimensions.Width	Engine Information.Driveline	Informatio
0	140	143	202	All-wheel drive	Audi 3.2L 250hp
1	140	143	202	Front-wheel drive	Audi 2.0L 200 hp
2	140	143	202	Front-wheel drive	Audi 2.0L 200 hp
3	140	143	202	All-wheel drive	Audi 2.0L 200 hp
5	91	17	62	All-wheel drive	Audi 3.2L 265hp
...	...	...	...	...	
5069	3	253	201	Front-wheel drive	Hor Cylinder 2
5070	3	253	201	Four-wheel drive	Hor Cylinder 2
5071	13	253	201	Front-wheel drive	Hor Cylinder 2
5074	200	210	110	Rear-wheel drive	BM cylinder 3 ft
5075	200	94	110	Rear-wheel drive	BM cylinder 3 ft

4794 rows × 18 columns

In [95]:
```
1 df.nunique()
```

Out[95]:
```
Dimensions.Height                                189
Dimensions.Length                                202
Dimensions.Width                                 130
Engine Information.Driveline                       4
Engine Information.Engine Type                   468
Engine Information.Hybrid                          1
Engine Information.Number of Forward Gears         4
Engine Information.Transmission                   10
Fuel Information.City mpg                         22
Fuel Information.Fuel Type                         4
Fuel Information.Highway mpg                      30
Identification.Classification                      2
Identification.ID                               4768
Identification.Make                               43
Identification.Model Year                        828
Identification.Year                                4
Engine Information.Engine Statistics.Horsepower  188
Engine Information.Engine Statistics.Torque      193
dtype: int64
```

In [96]:
```
df=df.drop(columns=['Engine Information.Hybrid'])
df
```

Out[96]:

	Dimensions.Height	Dimensions.Length	Dimensions.Width	Engine Information.Driveline	Informatio
0	140	143	202	All-wheel drive	Audi 3.2L 250hp
1	140	143	202	Front-wheel drive	Audi 2.0L 200 hp
2	140	143	202	Front-wheel drive	Audi 2.0L 200 hp
3	140	143	202	All-wheel drive	Audi 2.0L 200 hp
5	91	17	62	All-wheel drive	Audi 3.2L 265hp
...	...	...	...	...	
5069	3	253	201	Front-wheel drive	Hor Cylinder 2
5070	3	253	201	Four-wheel drive	Hor Cylinder 2
5071	13	253	201	Front-wheel drive	Hor Cylinder 2
5074	200	210	110	Rear-wheel drive	BM cylinder 3 ft
5075	200	94	110	Rear-wheel drive	BM cylinder 3 ft

4794 rows × 17 columns

In [97]:
```
df[cat_cols].nunique()
```

Out[97]:
```
Engine Information.Driveline          4
Engine Information.Engine Type      468
Engine Information.Transmission      10
Fuel Information.Fuel Type            4
Identification.Classification         2
Identification.ID                  4768
Identification.Make                  43
Identification.Model Year           828
dtype: int64
```

In [98]:

```
#Now we will encode categorical data .
#target encoding for the 4 highest values
target='Fuel Information.City mpg'
target_columns = [
    'Engine Information.Engine Type',
    'Engine Information.Transmission',
    'Identification.Model Year',
    'Identification.ID',
    'Identification.Make'
]

for i in target_columns:
    df[i]=df.groupby(i)[target].transform('mean')

# onehot encoding for low cardinality features

onehot_columns = ['Engine Information.Driveline',
    'Fuel Information.Fuel Type',
    'Identification.Classification']
df=pd.get_dummies(df,columns=onehot_columns)
```

In [104]:

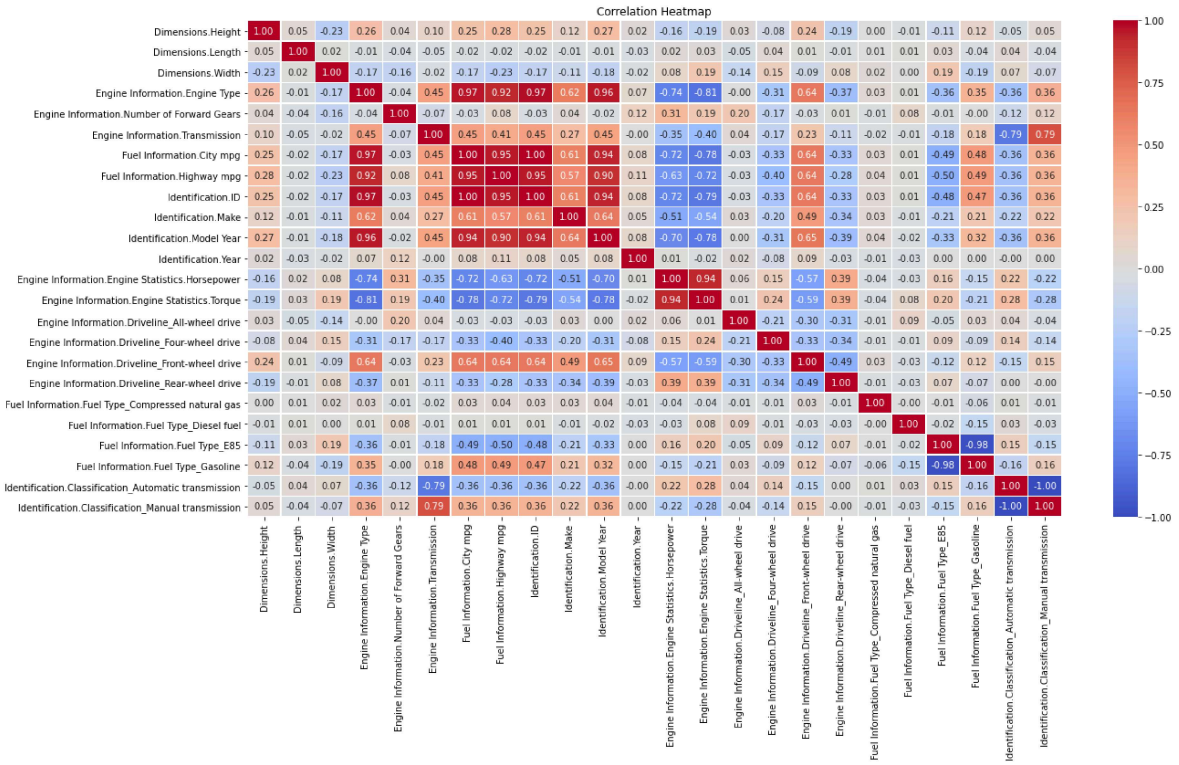
```
bool_to_num_cols=['Engine Information.Driveline_All-wheel drive',
                  'Engine Information.Driveline_Four-wheel drive',
                 'Engine Information.Driveline_Front-wheel drive',
                 'Engine Information.Driveline_Rear-wheel drive',
                 'Fuel Information.Fuel Type_Compressed natural gas',
                 'Fuel Information.Fuel Type_Diesel fuel',
                 'Fuel Information.Fuel Type_E85',
                 'Fuel Information.Fuel Type_Gasoline',
                 'Identification.Classification_Automatic transmission',
                 'Identification.Classification_Manual transmission']
df[bool_to_num_cols]=df[bool_to_num_cols].astype(int)
```

In [105]:
```
1 df
```

Out[105]:

Engine ɔn.Driveline_Front-wheel drive	Engine Information.Driveline_Rear-wheel drive	Fuel Information.Fuel Type_Compressed natural gas	Fuel Information.Fuel Type_Diesel fuel	Fue Information.Fue Type_E8
0	0	0	0	(
1	0	0	0	(
1	0	0	0	(
0	0	0	0	(
0	0	0	0	(
...	...	...	...	..
1	0	0	0	(
0	0	0	0	(
1	0	0	0	(
0	1	0	0	(
0	1	0	0	(

In [109]:
```
import seaborn as sns
corr_matrix=df.corr()
plt.figure(figsize=(20,10))
sns.heatmap(corr_matrix,annot=True,cmap='coolwarm',fmt='.2f',linewidths=0
plt.title('Correlation Heatmap')
plt.show()

```



FEATURE SCALING

In [110]:
```
from sklearn.preprocessing import MinMaxScaler
```

In [115]:
```
scaler=MinMaxScaler()
df[df.select_dtypes(include='number').columns]=scaler.fit_transform(df[df
```

In [116]:
```
1 df
```

Out[116]:

	Dimensions.Height	Dimensions.Length	Dimensions.Width	Engine Information.Engine Type	Information. of Forwa
0	0.545455	0.557312	0.794466	0.463415	(
1	0.545455	0.557312	0.794466	0.617886	(
2	0.545455	0.557312	0.794466	0.617886	(
3	0.545455	0.557312	0.794466	0.617886	(
5	0.351779	0.059289	0.241107	0.414634	(
...	...	...	...	...	
5069	0.003953	0.992095	0.790514	0.404878	(
5070	0.003953	0.992095	0.790514	0.404878	(
5071	0.043478	0.992095	0.790514	0.404878	(
5074	0.782609	0.822134	0.430830	0.414634	(
5075	0.782609	0.363636	0.430830	0.414634	(

4794 rows × 24 columns

Feature Selection

In [117]:
```
1 df.columns
```

Out[117]:
```
Index(['Dimensions.Height', 'Dimensions.Length', 'Dimensions.Width',
       'Engine Information.Engine Type',
       'Engine Information.Number of Forward Gears',
       'Engine Information.Transmission', 'Fuel Information.City mpg',
       'Fuel Information.Highway mpg', 'Identification.ID',
       'Identification.Make', 'Identification.Model Year',
       'Identification.Year',
       'Engine Information.Engine Statistics.Horsepower',
       'Engine Information.Engine Statistics.Torque',
       'Engine Information.Driveline_All-wheel drive',
       'Engine Information.Driveline_Four-wheel drive',
       'Engine Information.Driveline_Front-wheel drive',
       'Engine Information.Driveline_Rear-wheel drive',
       'Fuel Information.Fuel Type_Compressed natural gas',
       'Fuel Information.Fuel Type_Diesel fuel',
       'Fuel Information.Fuel Type_E85', 'Fuel Information.Fuel Type_Gasolin
e',
       'Identification.Classification_Automatic transmission',
       'Identification.Classification_Manual transmission'],
      dtype='object')
```

Using RFE method

In [118]:
```
from sklearn.feature_selection import RFE
from sklearn.linear_model import Ridge

X = df.drop('Fuel Information.City mpg', axis=1)# storing independent var
y = df['Fuel Information.City mpg'] #target variable

#creating objects for ridge and rfe
ridge = Ridge()
rfe = RFE(ridge, n_features_to_select=5)  # Select top 5 features
rfe.fit(X, y)

selected_features = X.columns[rfe.support_]
print("Selected Features by RFE:", selected_features)
```

```
Selected Features by RFE: Index(['Engine Information.Engine Type', 'Fuel Inf
ormation.Highway mpg',
       'Identification.ID', 'Identification.Model Year',
       'Fuel Information.Fuel Type_E85'],
      dtype='object')
```

In [132]:
```
#vif
from statsmodels.stats.outliers_influence import variance_inflation_facto
selected_features = ['Engine Information.Engine Type', 'Fuel Information.
       'Identification.ID',
       'Identification.Model Year',
       'Fuel Information.Fuel Type_E85']

X = df[selected_features]

# Create a DataFrame to store VIF values
vif_data = pd.DataFrame()
vif_data["Feature"] = X.columns
vif_data["VIF"] = [variance_inflation_factor(X.values, i) for i in range(

print(vif_data)
```

```
                         Feature         VIF
0  Engine Information.Engine Type  153.036549
1    Fuel Information.Highway mpg   62.119350
2              Identification.ID  179.245052
3       Identification.Model Year   51.423572
4  Fuel Information.Fuel Type_E85    1.282206
```

In [ ]:
```
#here Fuel Information.Fuel Type_E85 has vif value near to 1, ie, less mu
#So now VIF values of other 4 features should be reduced using PCA
```

In [133]:

```
from sklearn.decomposition import PCA
# Extract features with high VIF
high_vif_features = ['Engine Information.Engine Type', 'Fuel Information.


# Apply PCA
pca = PCA(n_components=1)  # Reduce to a single component
X_pca = pca.fit_transform(df[high_vif_features])

# Create a new column in the DataFrame for the PCA component
df['PCA_Enginetype_highway'] = X_pca

# Drop the original correlated features
df = df.drop(columns=high_vif_features)

print("Transformed DataFrame with PCA feature added:")
df.head()

```

```
Transformed DataFrame with PCA feature added:
```

Out[133]:

	Dimensions.Height	Dimensions.Length	Dimensions.Width	Engine Information.Number of Forward Gears	Information.Tra
0	0.545455	0.557312	0.794466	0.666667	
1	0.545455	0.557312	0.794466	0.666667	
2	0.545455	0.557312	0.794466	0.666667	
3	0.545455	0.557312	0.794466	0.666667	
5	0.351779	0.059289	0.241107	0.666667	

5 rows × 23 columns

In [134]:

```
from sklearn.decomposition import PCA
# Extract features with high VIF
high_vif_features1 = ['Identification.ID', 'Identification.Model Year']


# Apply PCA
pca = PCA(n_components=1)  # Reduce to a single component
X_pca = pca.fit_transform(df[high_vif_features1])

# Create a new column in the DataFrame for the PCA component
df['PCA_ID_ModelYear'] = X_pca

# Drop the original correlated features
df = df.drop(columns=high_vif_features1)

print("Transformed DataFrame with PCA feature added:")
df.head()

```

```
Transformed DataFrame with PCA feature added:
```

Out[134]:

Engine Information.Number of Forward Gears	Engine Information.Transmission	Fuel Information.City mpg	Identification.Make	Identification.Ye
0.666667	0.240283	0.476190	0.424175	0
0.666667	0.240283	0.666667	0.424175	0
0.666667	0.489321	0.619048	0.424175	0
0.666667	0.240283	0.619048	0.424175	0
0.666667	0.489321	0.380952	0.424175	0

In [135]:
```
#vif
from statsmodels.stats.outliers_influence import variance_inflation_facto
selected_features = ['PCA_Enginetype_highway',
       'PCA_ID_ModelYear',
       'Fuel Information.Fuel Type_E85']

X = df[selected_features]

# Create a DataFrame to store VIF values
vif_data = pd.DataFrame()
vif_data["Feature"] = X.columns
vif_data["VIF"] = [variance_inflation_factor(X.values, i) for i in range(

print(vif_data)
```

```
                         Feature        VIF
0          PCA_Enginetype_highway  23.623078
1               PCA_ID_ModelYear  23.140748
2  Fuel Information.Fuel Type_E85   1.208917
```

In [ ]:
```
#Now again do pca as 1st 2 have high VIF values
```

In [136]:
```
from sklearn.decomposition import PCA
# Extract features with high VIF
high_vif_features2 = ['PCA_Enginetype_highway', 'PCA_ID_ModelYear']


# Apply PCA
pca = PCA(n_components=1)  # Reduce to a single component
X_pca = pca.fit_transform(df[high_vif_features2])

# Create a new column in the DataFrame for the PCA component
df['PCA_final'] = X_pca

# Drop the original correlated features
df = df.drop(columns=high_vif_features2)

print("Transformed DataFrame with PCA feature added:")
df.head()

```

```
Transformed DataFrame with PCA feature added:
```

Out[136]:

	Dimensions.Height	Dimensions.Length	Dimensions.Width	Engine Information.Number of Forward Gears	Information.Tra
0	0.545455	0.557312	0.794466	0.666667	
1	0.545455	0.557312	0.794466	0.666667	
2	0.545455	0.557312	0.794466	0.666667	
3	0.545455	0.557312	0.794466	0.666667	
5	0.351779	0.059289	0.241107	0.666667	

5 rows × 21 columns

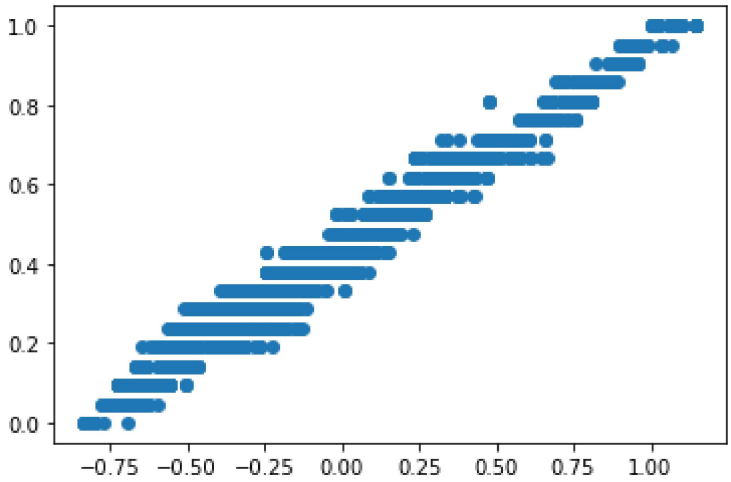
In [137]:
```
#vif
from statsmodels.stats.outliers_influence import variance_inflation_facto
selected_features = ['Fuel Information.Fuel Type_E85',
       'PCA_final']

X = df[selected_features]

# Create a DataFrame to store VIF values
vif_data = pd.DataFrame()
vif_data["Feature"] = X.columns
vif_data["VIF"] = [variance_inflation_factor(X.values, i) for i in range(

print(vif_data)

```

```
                         Feature       VIF
0  Fuel Information.Fuel Type_E85  1.192972
1                       PCA_final  1.192972
```

In [ ]:
```
#Since VIF is near to 1, it shows less multicollinearity(independence)
#Now check the linearity
```

In [142]:
```
x=df['PCA_final']
y=df['Fuel Information.City mpg']
plt.scatter(x,y)
```

Out[142]:
```
<matplotlib.collections.PathCollection at 0x2a8e7987b50>
```



In [144]:
```
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
import seaborn as sns

model=LinearRegression()

X=df[['PCA_final','Fuel Information.Fuel Type_E85']]
Y=df[['Fuel Information.City mpg']]

model.fit(X,Y)
pred=model.predict(X)
```

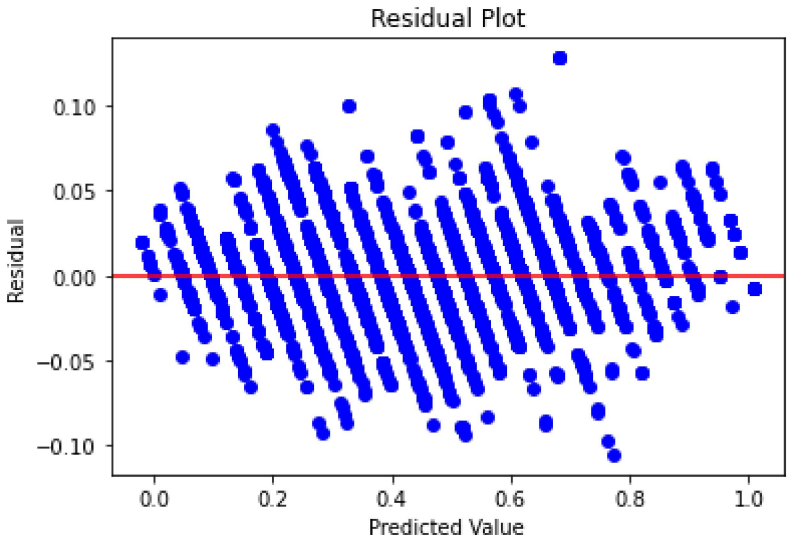
In [145]:
```
from sklearn.metrics import r2_score
r2_score_value = r2_score(Y, pred)
print(r2_score_value) #high r^2 means better model
```

```
0.9814751698519412
```

In [ ]:
```
#checking for homoscedasticity
```

In [146]:
```
error=Y-pred
plt.scatter(pred,error,color='b')
plt.xlabel('Predicted Value')
plt.ylabel('Residual')
plt.title('Residual Plot')
plt.axhline(y=0,color='r')

```

Out[146]: <matplotlib.lines.Line2D at 0x2a8e78f4640>



Conclusion

Hence multiple linear regression is performed with Fuel Information.City mpg as the target variable. Assumptions of Linear Regression model like multicollinearity,linearity and homoscedasticity are verified and a model is trained.

In [ ]:
```
1
```

In [ ]:
```
1
```

In [ ]:
```
1
```

In [ ]:
```
1
```

In [ ]:
```
1
```