In [2]:

```
import pandas as pd
import numpy as np
import seaborn as sns
```

In [44]:

```
#EDA for two wheeler
data = pd.read_excel(r'C:/Users/HP/Documents/EV _report/ev_two_wheeler.xlsx')
```

In [9]:

```
data.head()
```

Out[9]:

	Model Name	Used it for	Owned for	Ridden for	rating	Visual Appeal	Reliability	Performance	Service Experience	F
0	TVS iQube	Daily Commute	Never owned	NaN	1	3.0	4.0	NaN	NaN	
1	TVS iQube	Everything	> 1 yr	< 5000 kms	1	3.0	1.0	NaN	1.0	
2	TVS iQube	Daily Commute	< 3 months	< 5000 kms	3	4.0	4.0	NaN	2.0	
3	TVS iQube	Daily Commute	6 months-1 yr	5000-10000 kms	1	1.0	1.0	NaN	1.0	
4	TVS iQube	Daily Commute	6 months-1 yr	< 5000 kms	1	3.0	4.0	NaN	1.0	

In [19]:

```
data.tail()
```

Out[19]:

	Model Name	Used it for	Owned for	Ridden for	rating	Visual Appeal	Reliability	Performance	Servic Experienc
839	Gemopai Ryder	Daily Commute	> 1 yr	< 5000 kms	2	2.0	2.0	NaN	2.
840	Gemopai Ryder	Everything	< 3 months	< 5000 kms	5	5.0	5.0	5.0	5.
841	Gemopai Ryder	Daily Commute	> 1 yr	5000-10000 kms	2	2.0	2.0	4.0	2.
842	Gemopai Ryder	Daily Commute	3-6 months	< 5000 kms	5	4.0	4.0	NaN	Nal
843	Gemopai Ryder	Daily Commute	3-6 months	> 15000 kms	4	3.0	4.0	NaN	4.

In [10]:

```
data.describe()
```

Out[10]:

	rating	Visual Appeal	Reliability	Performance	Service Experience	Extra Features	Comfort
count	844.000000	739.000000	716.000000	345.000000	703.000000	185.000000	530.000000
mean	3.363744	3.779432	3.314246	3.527536	3.145092	2.935135	3.664151
std	1.689873	1.350021	1.585024	1.507721	1.637871	1.630587	1.387371
min	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
25%	1.000000	3.000000	2.000000	2.000000	1.000000	1.000000	3.000000
50%	4.000000	4.000000	4.000000	4.000000	4.000000	3.000000	4.000000
75%	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000
max	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000

In [11]:

```
data.shape
```

Out[11]:

```
(844, 13)
```

In [12]:

```
data.columns
```

Out[12]:

```
Index(['Model Name', 'Used it for', 'Owned for', 'Ridden for', 'rating',
       'Visual Appeal', 'Reliability', 'Performance', 'Service Experienc
e',
       'Extra Features', 'Comfort', 'Maintenance cost', 'Value for Mone
y'],
      dtype='object')
```

In [13]:

```
data.nunique()
```

Out[13]:

```
Model Name            39
Used it for            5
Owned for              6
Ridden for             5
rating                 5
Visual Appeal          5
Reliability            5
Performance            5
Service Experience     5
Extra Features         5
Comfort                5
Maintenance cost       5
Value for Money        5
dtype: int64
```

In [14]:

```
data['Model Name'].unique()
```

Out[14]:

```
array(['TVS iQube', 'Revolt RV 400', 'Bajaj Chetak', 'OLA S1 Pro',
       'Ather 450X', 'Hero Electric Optima', 'Tork Kratos', 'OLA S1',
       'Bounce Infinity E1', 'Hero Electric Optima CX',
       'Hero Electric Flash', 'Ampere Magnus EX', 'Revolt RV 300',
       'Hero Electric Photon', 'Okinawa Praise', 'Benling Aura',
       'Ampere Magnus Pro', 'PURE EV EPluto 7G', 'Ampere REO',
       'Odysse Evoqis', 'Hero Electric NYX HX', 'Okinawa i-Praise',
       'Joy e-bike Monster', 'PURE EV ETrance Neo', 'Evolet Polo',
       'Okinawa Ridge Plus', 'Ampere Zeal', 'Hero Electric Atria',
       'Okinawa Lite', 'Hero Electric NYX', 'Okinawa R30', 'Yo Drift',
       'BGauss B8', 'Joy e-bike Wolf', 'Gemopai Astrid Lite',
       'Techo Electra Emerge', 'Techo Electra Raptor', 'e-bike Gen Nxt',
       'Gemopai Ryder'], dtype=object)
```

In [27]:

```
#CLEANING DATA
```

In [15]:

```
data.isnull().sum()
```

Out[15]:

```
Model Name              0
Used it for             0
Owned for               0
Ridden for            176
rating                  0
Visual Appeal         105
Reliability           128
Performance           499
Service Experience    141
Extra Features        659
Comfort               314
Maintenance cost      664
Value for Money       454
dtype: int64
```

In [19]:

```
data['Comfort'].isnull().sum()
```

Out[19]:

```
314
```

In [46]:

```
data['Comfort'].mean()
```

Out[46]:

```
3.6641509433962263
```

In [4]:

```
updated_data = data['Comfort'].replace(np.NaN, data['Comfort'].mean())
print(updated_data.head(844))
```

```
0      4.000000
1      3.000000
2      5.000000
3      1.000000
4      3.000000
         ...
839    2.000000
840    3.664151
841    3.664151
842    4.000000
843    4.000000
Name: Comfort, Length: 844, dtype: float64
```

In [25]:

```
data['Visual Appeal'].isnull().sum()
```

Out[25]:

```
105
```

In [26]:

```
data['Visual Appeal'].mean()
```

Out[26]:

```
3.7794316644113666
```

In [27]:

```
updated_data = data['Visual Appeal'].replace(np.NaN, data['Visual Appeal'].mean())
print(updated_data.head(844))
```

```
0      3.0
1      3.0
2      4.0
3      1.0
4      3.0
      ...
839    2.0
840    5.0
841    2.0
842    4.0
843    3.0
Name: Visual Appeal, Length: 844, dtype: float64
```

In [28]:

```
data['Reliability'].isnull().sum()
data['Reliability'].mean()
updated_data = data['Reliability'].replace(np.NaN, data['Reliability'].mean())
print(updated_data.head(844))
```

```
0      4.0
1      1.0
2      4.0
3      1.0
4      4.0
      ...
839    2.0
840    5.0
841    2.0
842    4.0
843    4.0
Name: Reliability, Length: 844, dtype: float64
```

In [29]:

```
data['Performance'].isnull().sum()
data['Performance'].mean()
updated_data = data['Performance'].replace(np.NaN, data['Performance'].mean())
print(updated_data.head(844))
```

```
0      3.527536
1      3.527536
2      3.527536
3      3.527536
4      3.527536
         ...
839    3.527536
840    5.000000
841    4.000000
842    3.527536
843    3.527536
Name: Performance, Length: 844, dtype: float64
```

In [30]:

```
data['Service Experience'].isnull().sum()
data['Service Experience'].mean()
updated_data = data['Service Experience'].replace(np.NaN, data['Service Experience'].mean
print(updated_data.head(844))
```

```
0      3.145092
1      1.000000
2      2.000000
3      1.000000
4      1.000000
         ...
839    2.000000
840    5.000000
841    2.000000
842    3.145092
843    4.000000
Name: Service Experience, Length: 844, dtype: float64
```

In [31]:

```
data['Extra Features'].isnull().sum()
data['Extra Features'].mean()
updated_data = data['Extra Features'].replace(np.NaN, data['Extra Features'].mean())
print(updated_data.head(844))
```

```
0      2.935135
1      2.935135
2      2.935135
3      2.935135
4      2.935135
         ...
839    2.935135
840    5.000000
841    3.000000
842    2.935135
843    2.935135
Name: Extra Features, Length: 844, dtype: float64
```

In [32]:

```
data['Maintenance cost'].isnull().sum()
data['Maintenance cost'].mean()
updated_data = data['Maintenance cost'].replace(np.NaN, data['Maintenance cost'].mean())
print(updated_data.head(844))
```

```
0      3.394444
1      3.394444
2      3.394444
3      3.394444
4      3.394444
         ...
839    3.394444
840    5.000000
841    1.000000
842    3.394444
843    3.394444
Name: Maintenance cost, Length: 844, dtype: float64
```

In [33]:

```
data['Value for Money'].isnull().sum()
data['Value for Money'].mean()
updated_data = data['Value for Money'].replace(np.NaN, data['Value for Money'].mean())
print(updated_data.head(844))
```

```
0      1.000000
1      3.000000
2      2.000000
3      1.000000
4      2.000000
         ...
839    3.000000
840    3.382051
841    3.382051
842    5.000000
843    4.000000
Name: Value for Money, Length: 844, dtype: float64
```

In [14]:

```
#Droping redandant data
EV= data.drop(['Service Experience','Visual Appeal','Maintenance cost','Performance','Ext
```

In [15]:

```
EV.head()
```

Out[15]:

	Model Name	Used it for	Owned for	Ridden for	rating	Reliability	Comfort	Value for Money
0	TVS iQube	Daily Commute	Never owned	NaN	1	4.0	4.0	1.0
1	TVS iQube	Everything	> 1 yr	< 5000 kms	1	1.0	3.0	3.0
2	TVS iQube	Daily Commute	< 3 months	< 5000 kms	3	4.0	5.0	2.0
3	TVS iQube	Daily Commute	6 months-1 yr	5000-10000 kms	1	1.0	1.0	1.0
4	TVS iQube	Daily Commute	6 months-1 yr	< 5000 kms	1	4.0	3.0	2.0

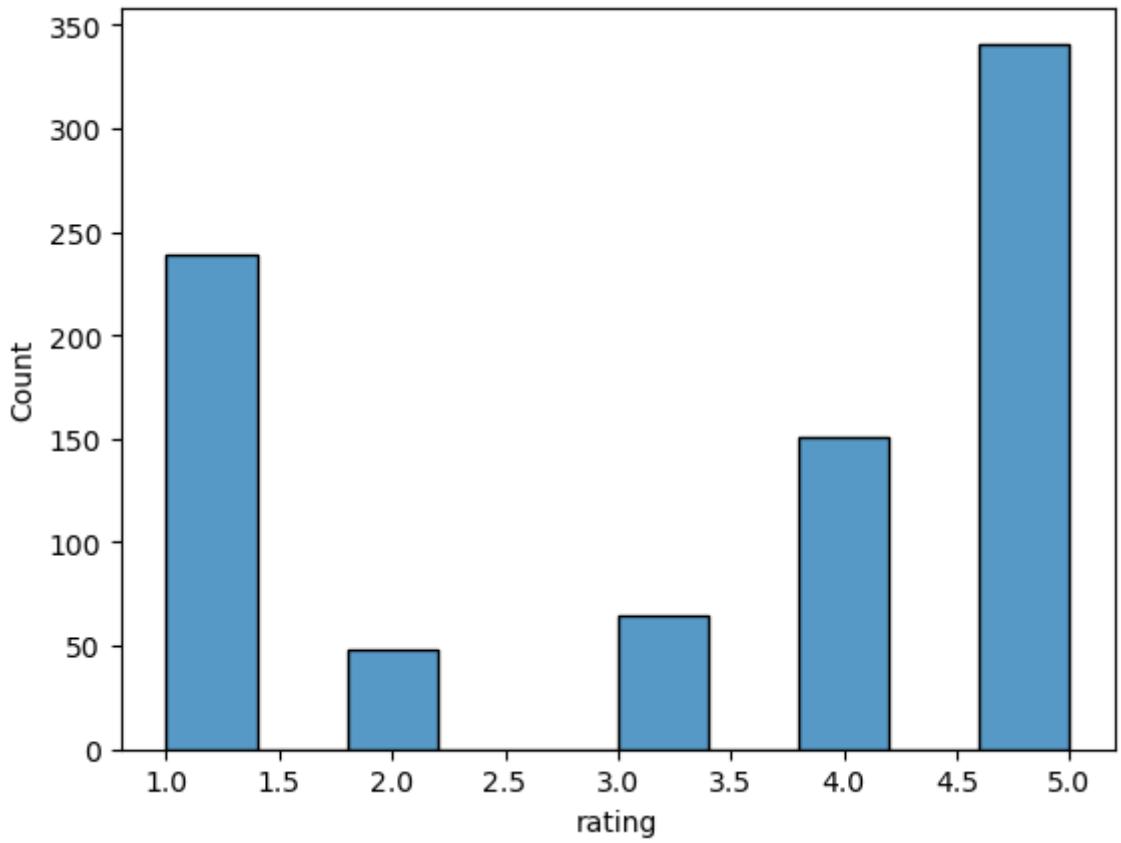
In [6]:

```
#realationship analysis
```

In [45]:

```
#rating for twowheeler
sns.histplot(data['rating'], bins=10)

# Show the plot
plt.show()
```

```
---------------------------------------------------------------------------
-
NameError                                 Traceback (most recent call las
t)
Cell In[45], line 4
      1 sns.histplot(data['rating'], bins=10)
      3 # Show the plot
----> 4 plt.show()

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



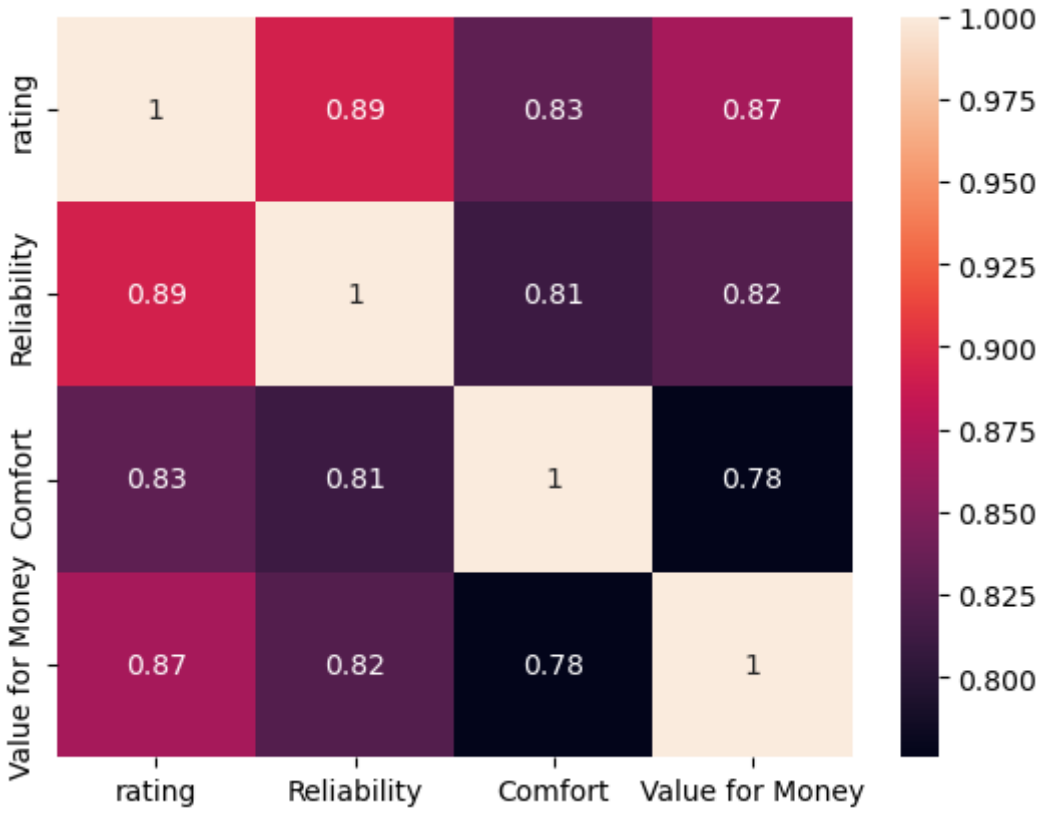
In [20]:

```
import seaborn as sns

correlation = EV.corr()
sns.heatmap(correlation, xticklabels=correlation.columns, yticklabels=correlation.columns
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_11892\2114391969.py:3: FutureWarn
ing: The default value of numeric_only in DataFrame.corr is deprecated. In
a future version, it will default to False. Select only valid columns or s
pecify the value of numeric_only to silence this warning.
  correlation = EV.corr()
```

Out[20]:

```
<Axes: >
```

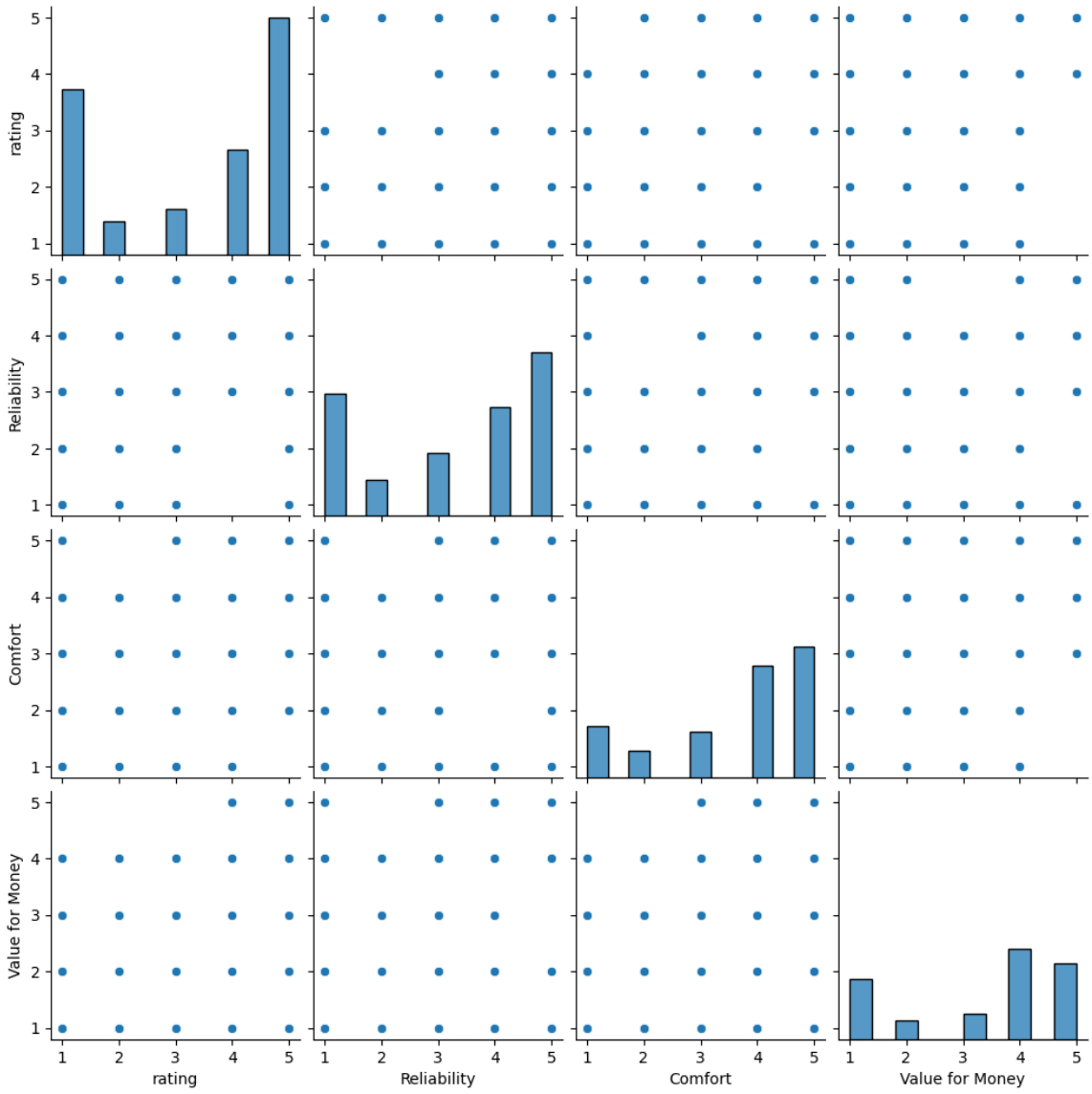


In [21]:

```
sns.pairplot(EV)
```

Out[21]:

```
<seaborn.axisgrid.PairGrid at 0x270f9cfaf50>
```

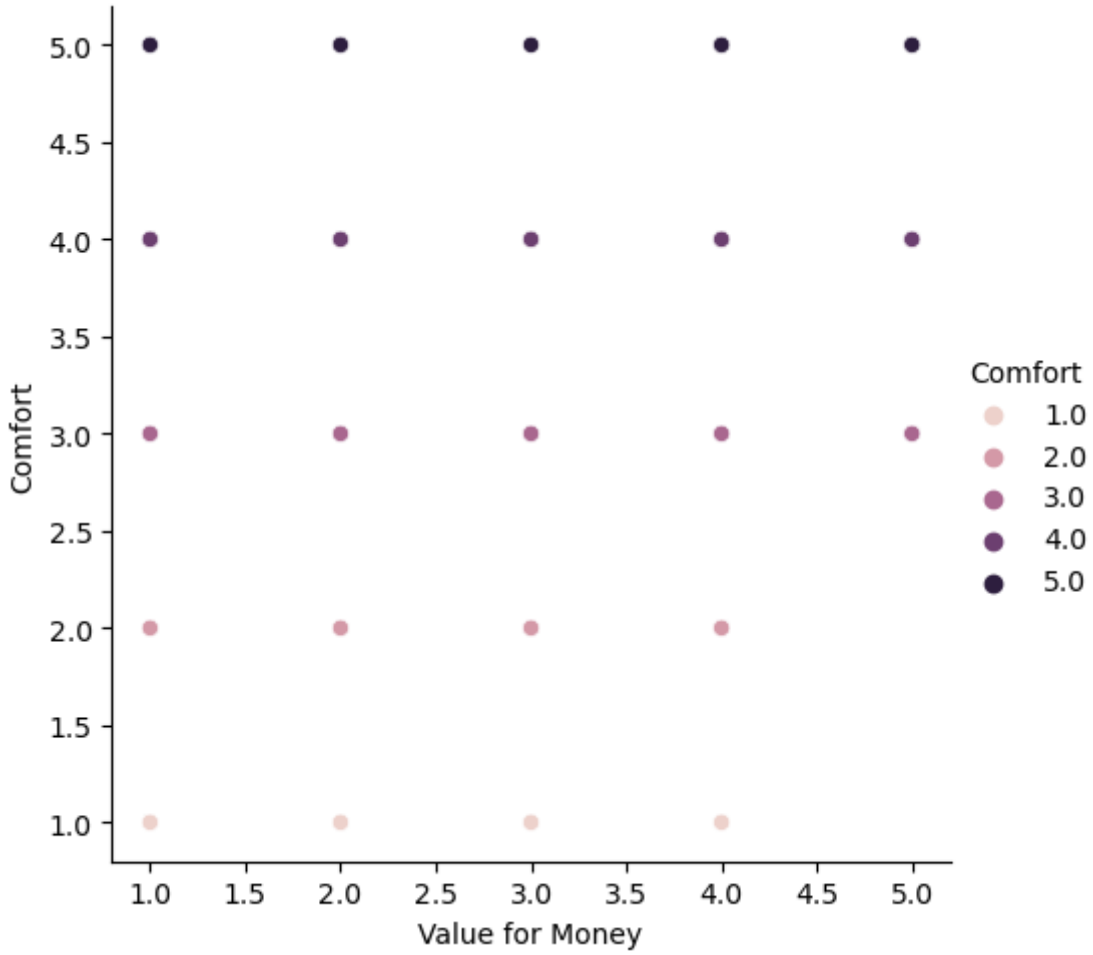


In [24]:

```
sns.relplot(x= 'Value for Money',y='Comfort',hue='Comfort',data=EV)
```

Out[24]:

```
<seaborn.axisgrid.FacetGrid at 0x270faee6140>
```



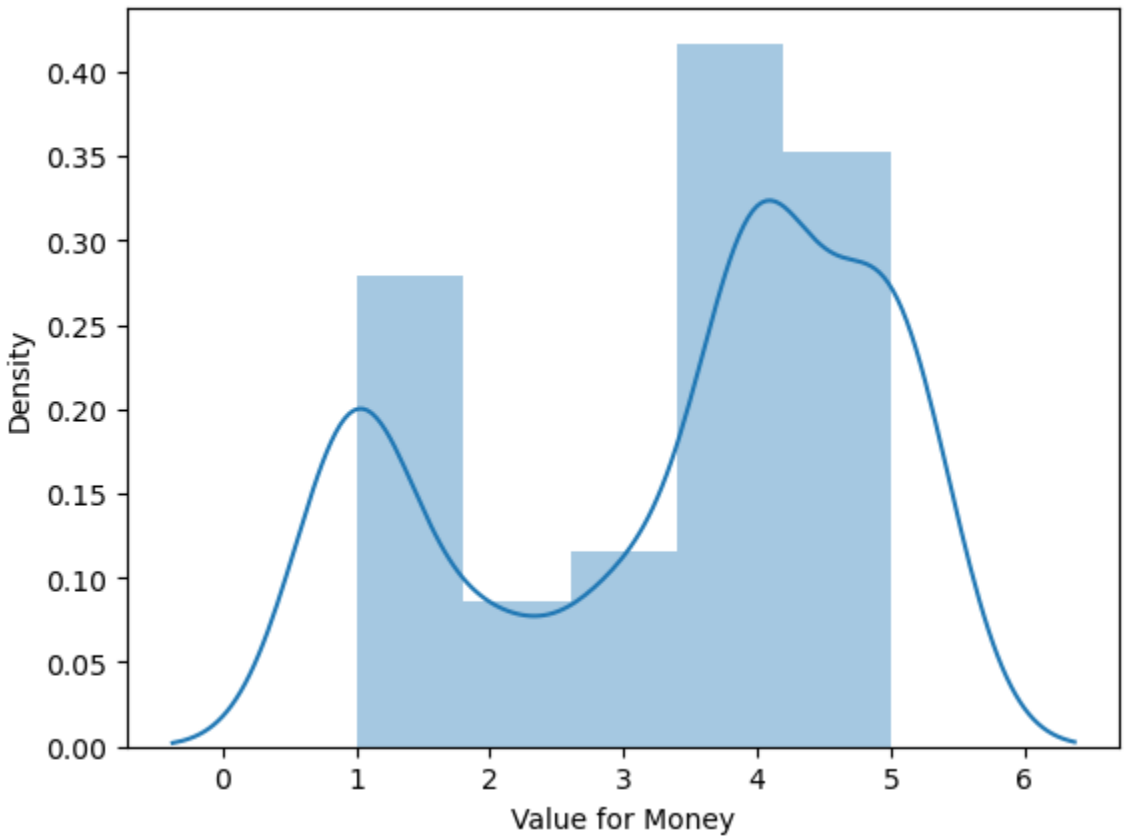
In [28]:

```
sns.distplot(EV['Value for Money'],bins=5)
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_11892\344180319.py:1: UserWarnin
g:

`distplot` is a deprecated function and will be removed in seaborn v0.14.
0.

Please adapt your code to use either `displot` (a figure-level function wi
th
similar flexibility) or `histplot` (an axes-level function for histogram
s).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://
gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(EV['Value for Money'],bins=5)
```

Out[28]:

```
<Axes: xlabel='Value for Money', ylabel='Density'>
```

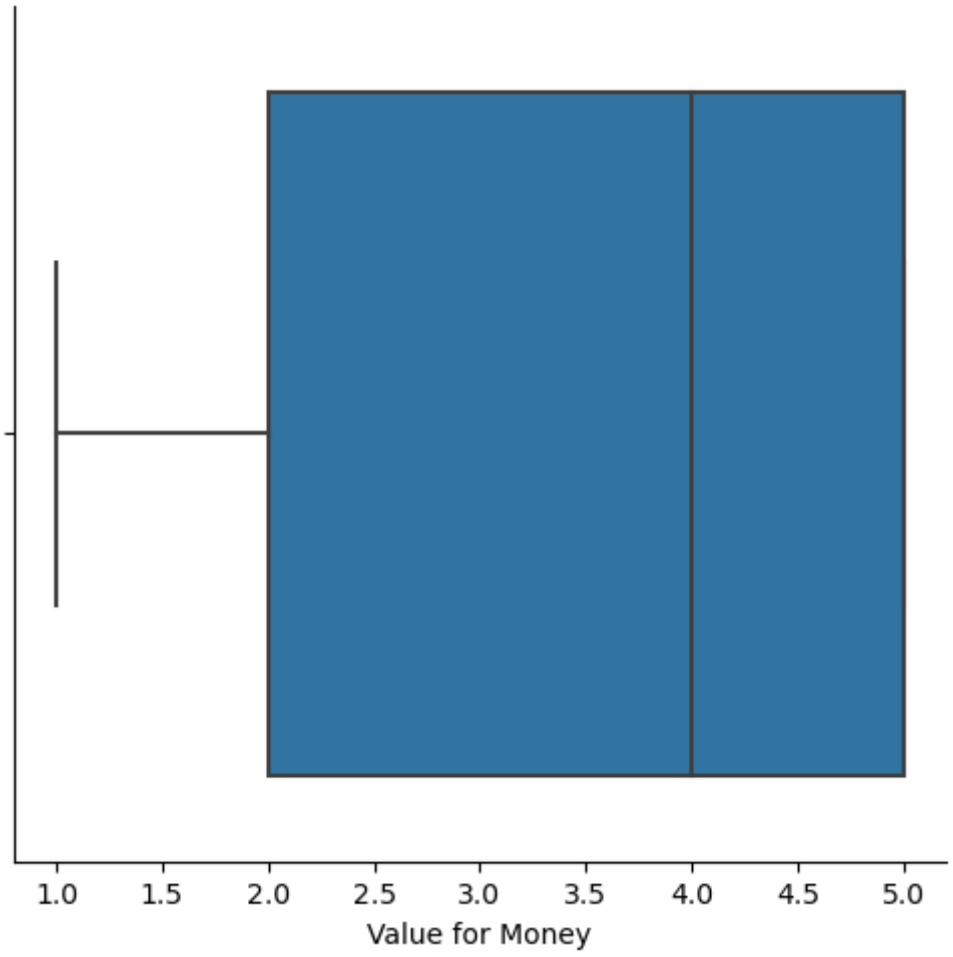


In [31]:

```
sns.catplot(x='Value for Money',kind ='box' ,data =EV)
```

Out[31]:

```
<seaborn.axisgrid.FacetGrid at 0x270fc7494e0>
```



In [46]:

```
#EDA for two wheeler
data = pd.read_excel(r'C:/Users/HP/Documents/EV _report/ev_four_wheeler.xlsx')
```

In [36]:

```
data.head()
```

Out[36]:

	model_name	Exterior	Comfort	Performance	Fuel Economy	Value for Money	Condition	driven	rati
0	hyundai kona	5	4	5	5	5	New	Few hundred kilometers	
1	hyundai kona	1	1	1	1	1	New	Haven't driven it	
2	hyundai kona	4	5	5	5	4	New	Few thousand kilometers	
3	hyundai kona	5	5	5	5	5	New	Few thousand kilometers	
4	hyundai kona	4	4	5	3	2	Not Purchased	Haven't driven it	

In [37]:

```
data.tail()
```

Out[37]:

	model_name	Exterior	Comfort	Performance	Fuel Economy	Value for Money	Condition	driven	r
124	tata tigor ev	5	4	4	4	3	New	Did a short drive once	
125	tata tigor ev	5	5	5	5	5	Not Purchased	Did a short drive once	
126	tata tigor ev	5	5	5	5	5	Not Purchased	Did a short drive once	
127	tata tigor ev	4	4	4	5	5	Not Purchased	Haven't driven it	
128	tata tigor ev	5	5	5	3	2	Not Purchased	Few hundred kilometers	

In [38]:

```
data.describe()
```

Out[38]:

	Exterior	Comfort	Performance	Fuel Economy	Value for Money	rating
count	129.000000	129.000000	129.000000	129.000000	129.000000	129.000000
mean	4.472868	4.418605	4.418605	4.418605	4.162791	4.341085
std	0.968871	1.036051	1.150392	1.122899	1.345076	1.389110
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	4.000000	4.000000	4.000000	4.000000	4.000000	4.000000
50%	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000
75%	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000
max	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000

In [39]:

```
data.isnull().sum()
```

Out[39]:

```
model_name         0
Exterior           0
Comfort            0
Performance        0
Fuel Economy       0
Value for Money    0
Condition          0
driven             0
rating             0
dtype: int64
```

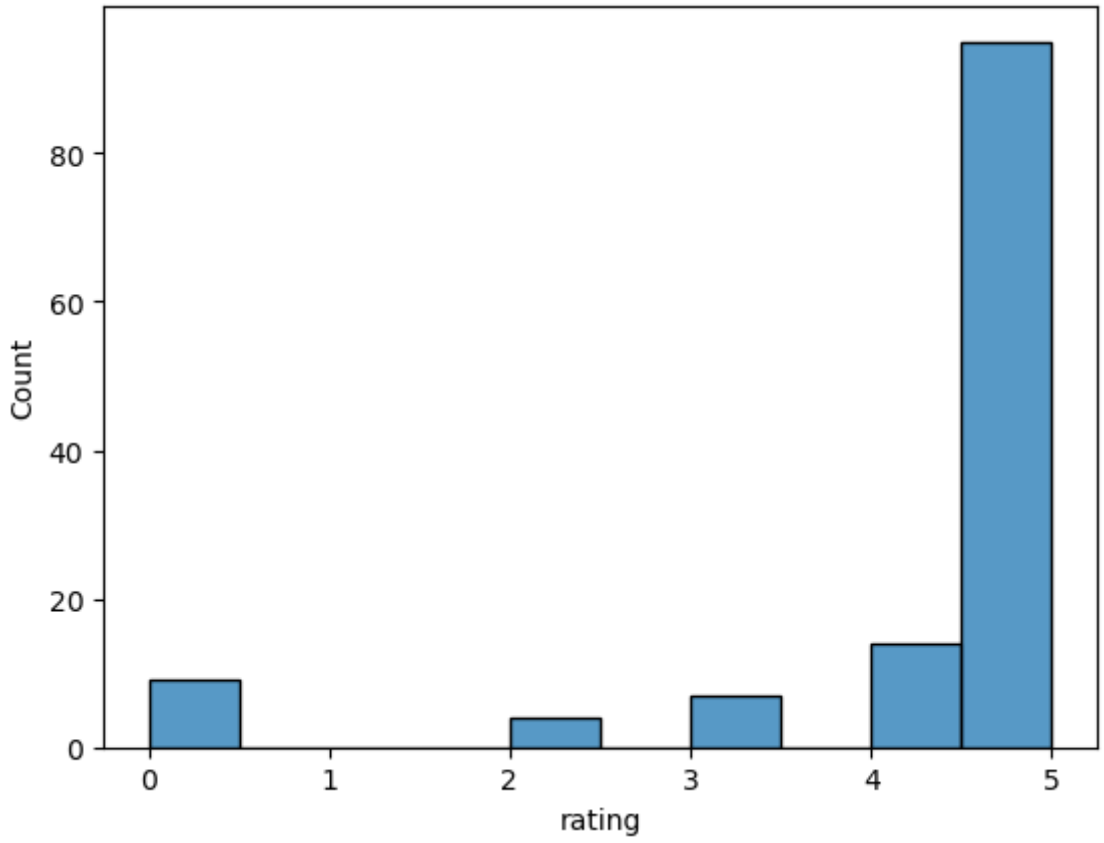
In [41]:

```
#Droping redandant data
EV= data.drop(['driven','Condition'],axis=1)
```

In [42]:

```
sns.histplot(data['rating'], bins=10)

# Show the plot
plt.show()
```

```
---------------------------------------------------------------------------
-
NameError                                 Traceback (most recent call las
t)
Cell In[42], line 4
      1 sns.histplot(data['rating'], bins=10)
      3 # Show the plot
----> 4 plt.show()

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



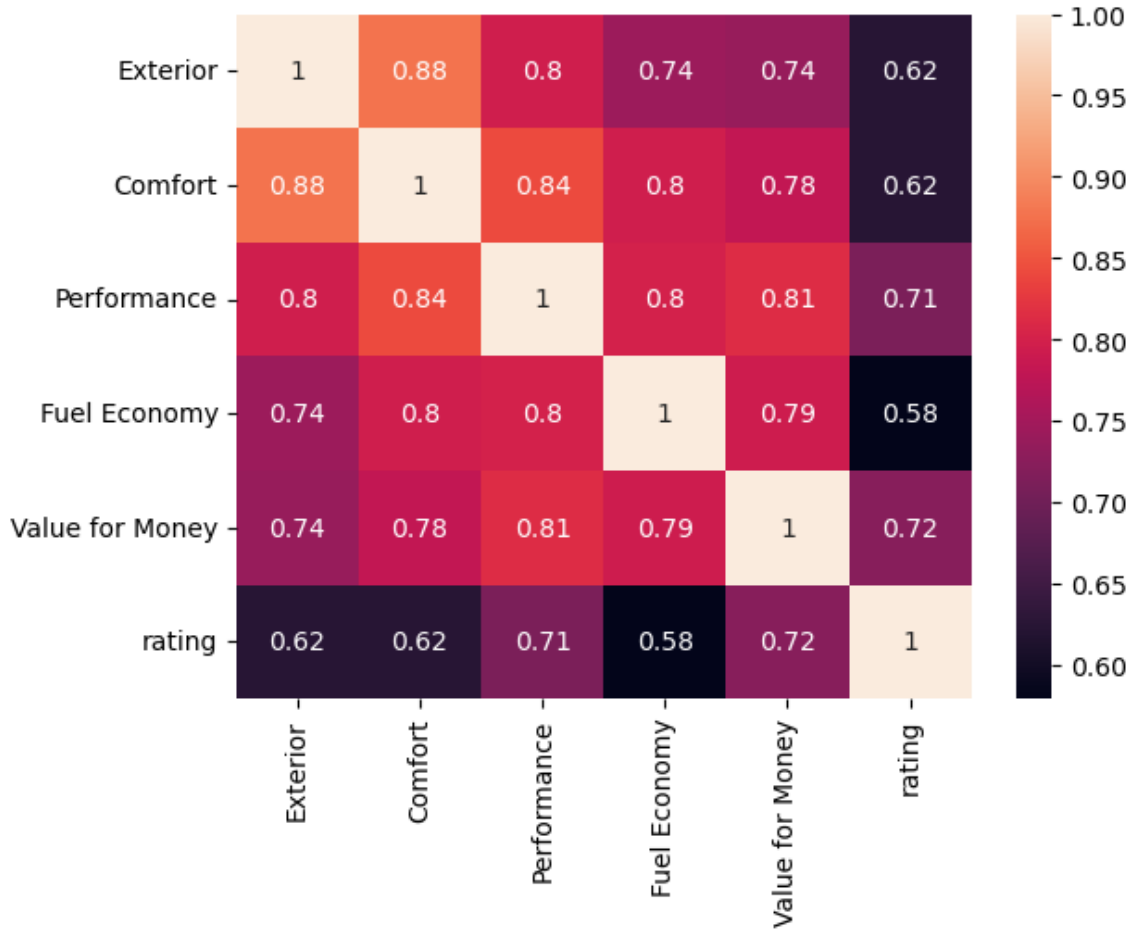
In [47]:

```
import seaborn as sns

correlation = EV.corr()
sns.heatmap(correlation, xticklabels=correlation.columns, yticklabels=correlation.columns
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_11892\2114391969.py:3: FutureWarn
ing: The default value of numeric_only in DataFrame.corr is deprecated. In
a future version, it will default to False. Select only valid columns or s
pecify the value of numeric_only to silence this warning.
  correlation = EV.corr()
```

Out[47]:

```
<Axes: >
```

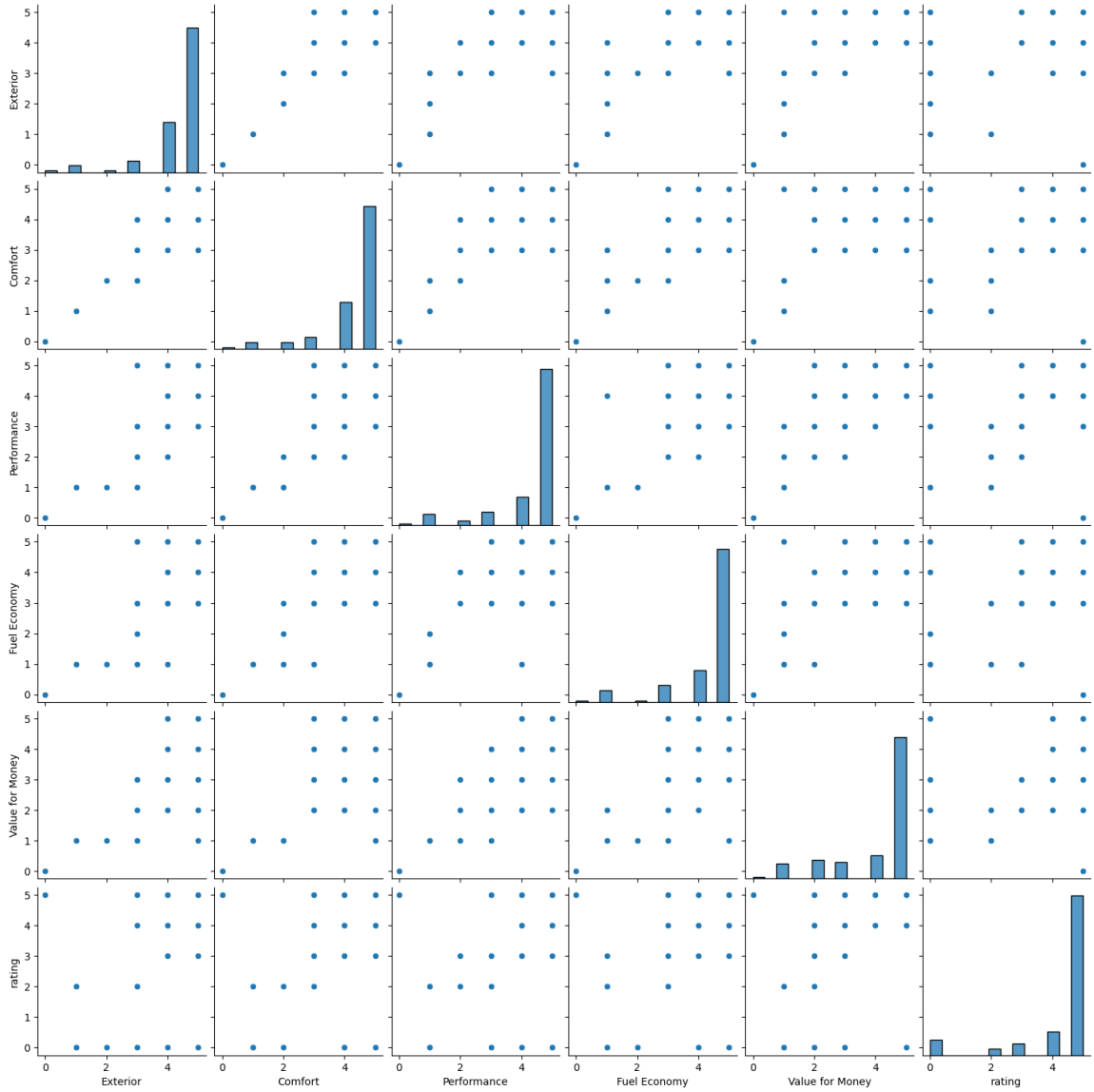


In [48]:

```
sns.pairplot(EV)
```

Out[48]:

```
<seaborn.axisgrid.PairGrid at 0x27080c08af0>
```



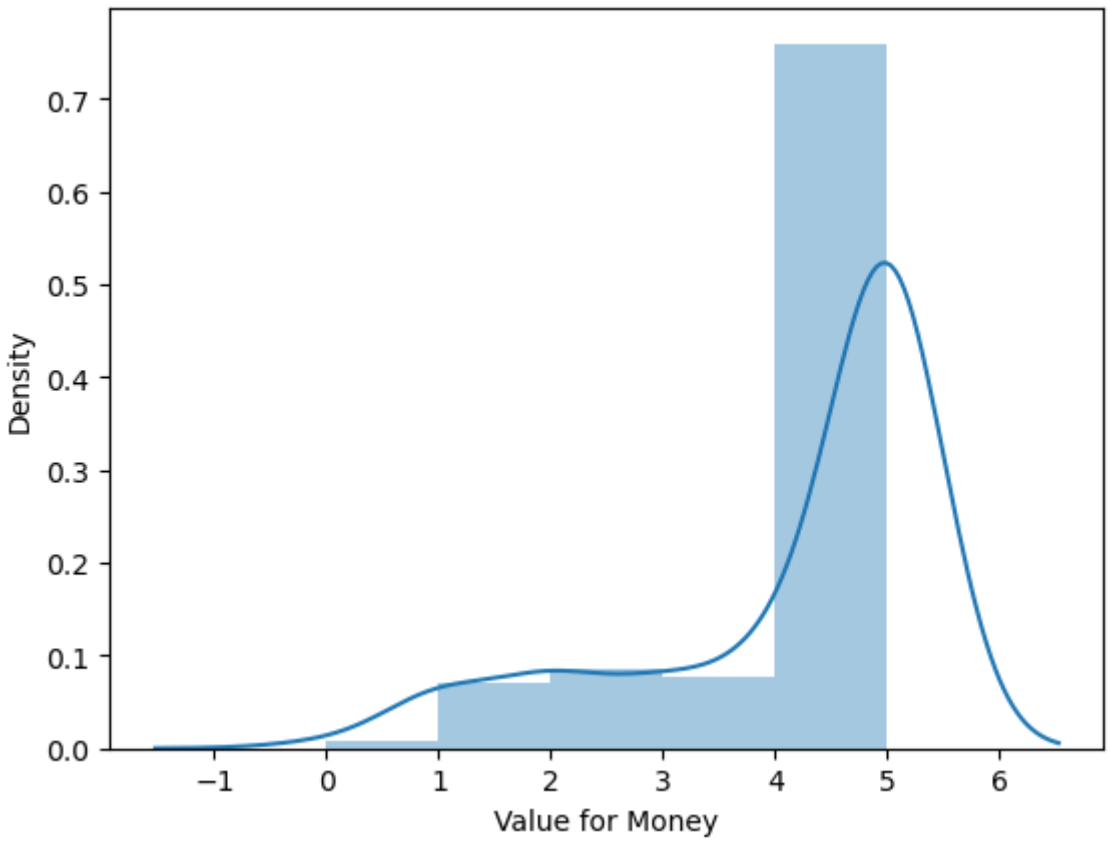
In [49]:

```
sns.distplot(EV['Value for Money'],bins=5)
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_11892\344180319.py:1: UserWarnin
g:

`distplot` is a deprecated function and will be removed in seaborn v0.14.
0.

Please adapt your code to use either `displot` (a figure-level function wi
th
similar flexibility) or `histplot` (an axes-level function for histogram
s).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://
gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(EV['Value for Money'],bins=5)
```

Out[49]:

```
<Axes: xlabel='Value for Money', ylabel='Density'>
```

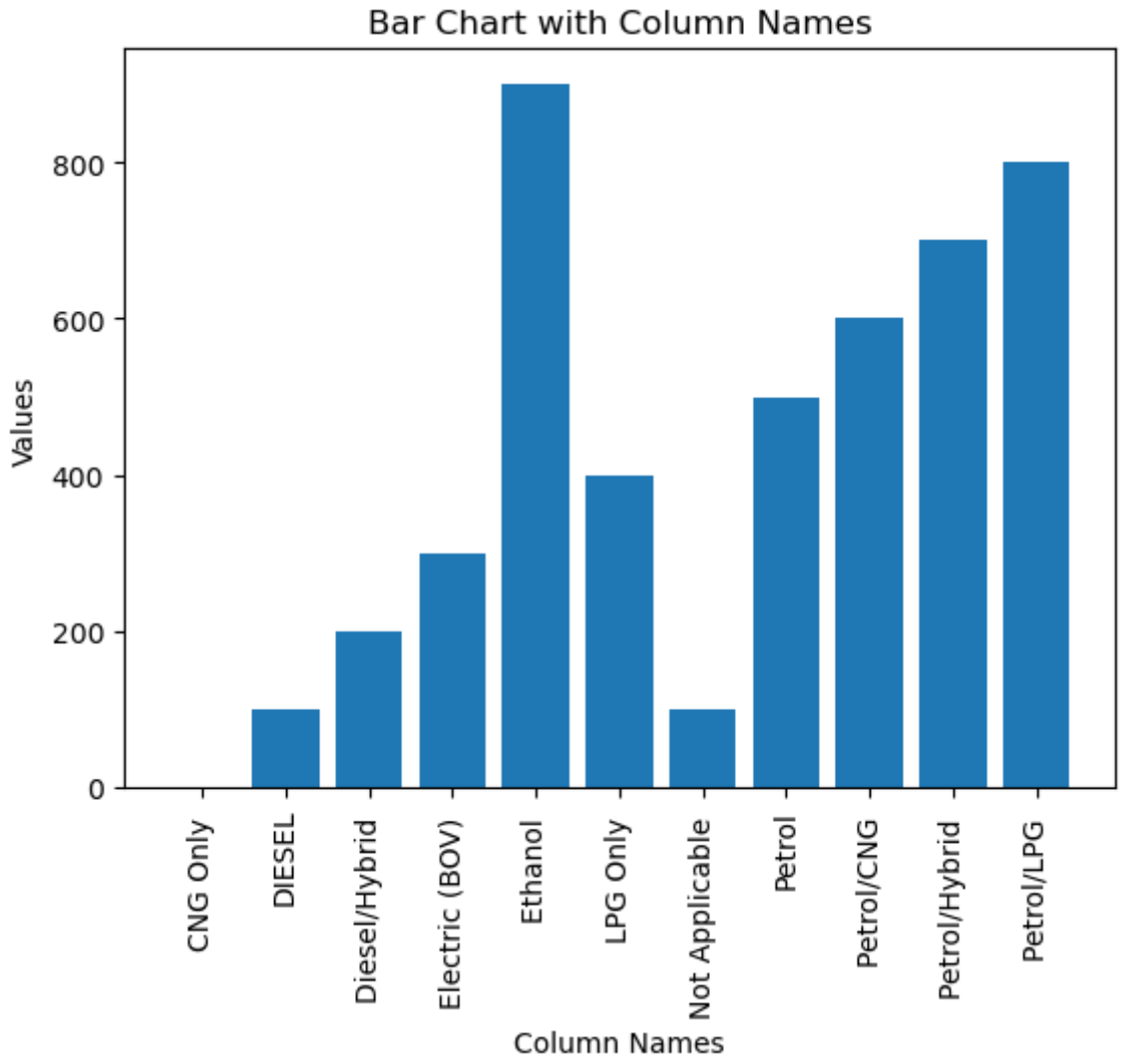


In [53]:

```
#registration of vehicles in karnataka
#EDA for two wheeler
data = pd.read_excel(r'C:/Users/HP/Documents/EV _report/registration.xlsx')
```

In [72]:

```
import matplotlib.pyplot as plt

# Assuming 'data' is your pandas DataFrame
data = pd.DataFrame({'Column': ['CNG Only ', 'DIESEL', 'Diesel/Hybrid ', 'Electric (BOV)

# Set the first column as the base
base_column = data['Column'].iloc[0]
data_sorted = data.sort_values(by='Column')

# Create the bar chart
plt.bar(data_sorted['Column'], data_sorted['Values'], bottom=data_sorted['Column'].apply(

# Add labels and title
plt.xlabel('Column Names')
plt.ylabel('Values')
plt.title('Bar Chart with Column Names')

# Rotate x-axis labels for better visibility
plt.xticks(rotation=90)

# Show the plot
plt.show()
```



In [74]:

```
#dataset of No. of Electric Vehicle (EV) Chargers Sanctioned

data = pd.read_excel(r'C:/Users/HP/Documents/EV _report/sanctioned_charges.xlsx')
```

In [76]:

```
print(data)
```

```
                          State/UT  \
0                      Maharashtra
1                   Andhra Pradesh
2                       Tamil Nadu
3                          Gujarat
4                    Uttar Pradesh
5                        Rajasthan
6                        Karnataka
7                   Madhya Pradesh
8                      West Bengal
9                        Telangana
10                          Kerala
11                           Delhi
12                      Chandigarh
13                         Haryana
14                       Meghalaya
15                           Bihar
16                          Sikkim
17                Jammu and Kashmir
18                     Chhattisgarh
19                            Assam
20                           Odisha
21                      Uttarakhand
22                       Puducherry
23  Andaman and Nicobar (Port Blair)
24                  Himachal Pradesh
25                             Total

    No. of Electric Vehicle (EV) Chargers Sanctioned
0                                                317
1                                                266
2                                                281
3                                                278
4                                                207
5                                                205
6                                                172
7                                                235
8                                                141
9                                                138
10                                               211
11                                                72
12                                                70
13                                                50
14                                                40
15                                                37
16                                                29
17                                                25
18                                                25
19                                                20
20                                                18
21                                                10
22                                                10
23                                                10
24                                                10
25                                              2877
```

In [ ]:

```

```