In [1]:
```
import pandas as pd
FS=pd.read_csv('fs.csv')
```

In [2]:
```
#loding the columns of a csv file ##
FS.columns=["Open", "Close", "High", "Low", "Volume"]
```

In [3]:
```
#displaying the colums with some data ##
FS.sample(50)
```

Out[3]:

	Open	Close	High	Low	Volume
9554	415.250000	420.820007	415.089996	420.549988	22032800
6001	31.000000	31.180000	30.889999	31.170000	25384000
9570	413.959991	414.250000	400.640015	402.649994	26919200
1863	2.445313	2.476563	2.429688	2.445313	65801600
3448	46.375000	47.250000	45.968750	46.281250	46349000
7077	39.750000	39.930000	39.200001	39.820000	35918600
2955	16.812500	16.976563	16.679688	16.859375	60094400
1913	2.585938	2.648438	2.578125	2.601563	86931200
1054	0.920139	0.923611	0.909722	0.913194	47232000
8538	163.779999	165.759995	163.070007	165.460007	24899900
2207	3.921875	3.953125	3.875000	3.953125	37715200
3236	35.218750	35.839844	34.718750	34.750000	34942800
5149	24.490000	24.639999	24.340000	24.440001	44405700
9534	386.000000	390.679993	380.380005	384.630005	27850800
8901	279.399994	280.690002	277.149994	279.929993	23260000
3122	29.546875	29.875000	29.203125	29.250000	45794800
4571	27.400000	27.719999	27.340000	27.540001	258269000
8453	137.500000	139.960007	136.029999	139.360001	21382000
3177	25.625000	25.906250	24.343750	24.406250	76550000
7635	49.810001	50.939999	49.520000	49.830002	133503000
2376	5.710938	5.867188	5.703125	5.867188	77705600
8810	232.080002	234.589996	227.880005	228.990005	39542200
1148	0.840278	0.843750	0.826389	0.836806	39268800
1110	0.909722	0.927083	0.892361	0.916667	91771200
1330	1.579861	1.593750	1.562500	1.576389	29419200
8928	289.480011	292.899994	289.299988	292.850006	18249000
2759	12.296875	12.562500	12.187500	12.546875	62448000
1323	1.565972	1.569444	1.534722	1.543403	32680800
7460	46.650002	47.099998	46.529999	46.680000	24697800
3370	46.875000	47.500000	46.250000	47.468750	37985200
3943	30.250000	30.809999	29.809999	30.660000	79141400
8800	245.000000	245.919998	240.889999	242.820007	22186700
1161	0.802083	0.805556	0.770833	0.784722	130334400
1754	2.539063	2.585938	2.468750	2.550781	138553600
5441	29.660000	29.850000	29.600000	29.840000	30265400
2306	4.984375	5.015625	4.945313	4.976563	44305600
5252	29.650000	30.100000	29.530001	29.930000	50220200
9284	235.259995	239.899994	233.559998	238.509995	27269500
7663	56.799999	57.520000	56.669998	57.389999	26587700
7305	43.070000	43.240002	42.820000	43.110001	23193500
8509	153.000000	154.889999	152.830002	154.529999	23845400
704	0.364583	0.368056	0.359375	0.361111	74592000
4839	25.309999	25.500000	25.250000	25.459999	39983200

1097	1.093750	1.121528	1.090278	1.100694	48729600
681	0.321181	0.324653	0.317708	0.323785	34430400
3012	19.656250	19.937500	19.593750	19.906250	48099200
8456	138.050003	139.220001	137.779999	139.029999	17280900
929	0.586806	0.612847	0.581597	0.611979	178905600
1341	1.484375	1.505208	1.416667	1.419271	43920000
4332	26.040001	26.200001	25.889999	25.990000	44501900

In [ ]:
```
#pip install scikit-learn
```

In [4]:
```
from sklearn.preprocessing import normalize
```

In [5]:
```
from sklearn.preprocessing import MinMaxScaler
min_max = MinMaxScaler()
min_max.fit_transform(FS[['Open','Close','Volume']])
taxi_min_max = pd.DataFrame(min_max.fit_transform(FS[['Open','Close','Volume']]), columns = ['Open','Close','Vo
taxi_min_max.sample(40)
```

Out[5]:

	Open	Close	Volume
8283	0.244943	0.246704	0.022895
7691	0.130062	0.130420	0.021135
233	0.000371	0.000405	0.178174
3760	0.073213	0.074325	0.080312
0	0.000000	0.000022	1.000000
6889	0.079124	0.080086	0.026571
1304	0.003108	0.003184	0.071875
5913	0.055076	0.056119	0.064797
296	0.000772	0.000789	0.056006
8635	0.445391	0.450944	0.039100
4744	0.061914	0.061932	0.044732
6679	0.069607	0.070213	0.030448
1527	0.005992	0.005987	0.056435
3706	0.081948	0.083264	0.198007
833	0.000662	0.000653	0.053572
5535	0.065033	0.065277	0.119867
887	0.000802	0.000821	0.073687
1355	0.003068	0.003068	0.032339
6510	0.061752	0.063155	0.075972
2792	0.027264	0.027505	0.088991
5476	0.076653	0.077295	0.079529
8337	0.276453	0.277407	0.013767
2868	0.040908	0.042895	0.105794
8874	0.580024	0.579892	0.021489
896	0.000858	0.000867	0.065014
7176	0.103172	0.103546	0.014914
5935	0.059673	0.060340	0.056626
1295	0.003012	0.003064	0.083604
6727	0.062723	0.062739	0.071669
4822	0.057663	0.058033	0.075556
7386	0.103080	0.102946	0.032678
1172	0.001801	0.001846	0.048824
4321	0.058449	0.059625	0.052437
260	0.000497	0.000515	0.065378
2480	0.012663	0.012781	0.058819
213	0.000277	0.000306	0.111425
1269	0.002972	0.003024	0.078596
5969	0.066557	0.067330	0.061756
4571	0.063092	0.063731	0.248634
2171	0.007863	0.007934	0.024875

In [10]:

```
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np



# Replace infinite values with NaN
FS.replace([np.inf, -np.inf], np.nan, inplace=True)

# Drop rows with NaN values
FS.dropna(inplace= True)

#Plot KDE plot
sns.kdeplot(data=FS[['Open', 'Close', 'Volume']])
```

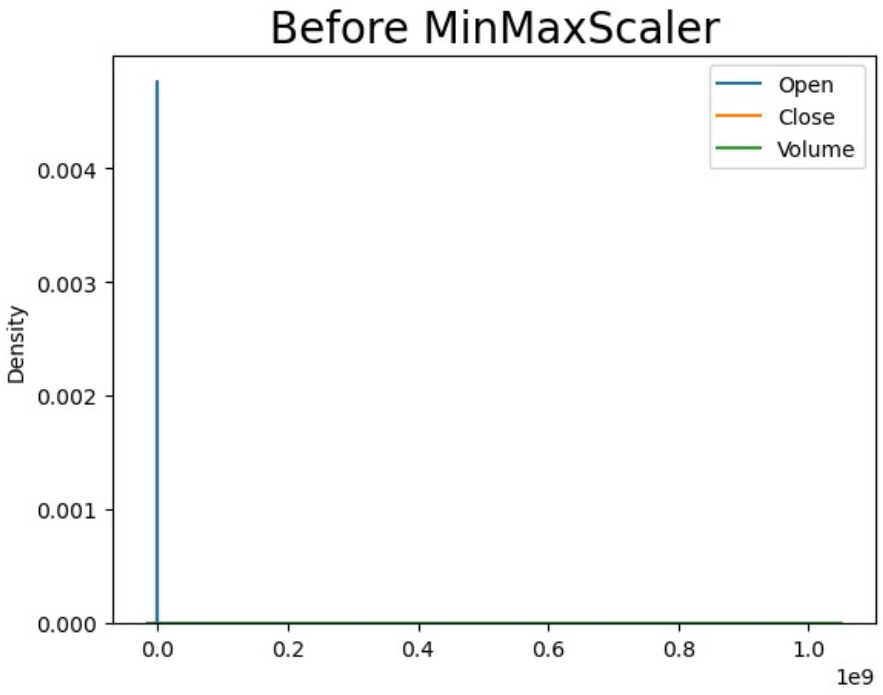
```
plt.title('Before MinMaxScaler', size=20)
plt.show()
```

In [13]:
```
# Check data types of columns
print(FS[['Open', 'Close', 'Volume']].dtypes)

# Check for NaN values
print(FS[['Open', 'Close', 'Volume']].isnull().sum())
```

```
Open      float64
Close     float64
Volume      int64
dtype: object
Open      0
Close     0
Volume    0
dtype: int64
```

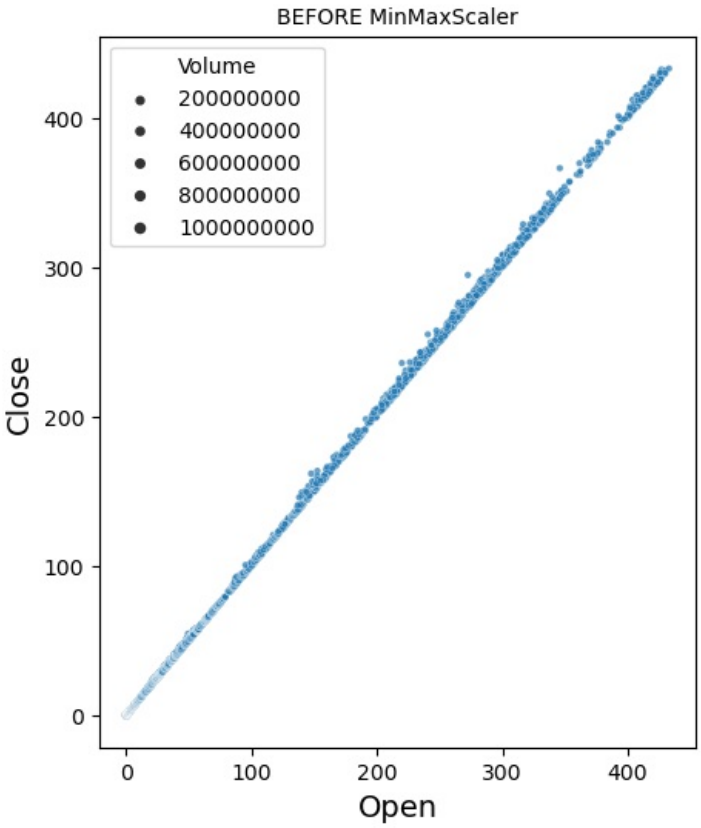
In [14]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import warnings

# Suppress specific warning
warnings.filterwarnings("ignore", message="use_inf_as_na is deprecated")

# Replace infinite values with NaN
FS.replace([np.inf, -np.inf], np.nan, inplace=True)

# Plot KDE plot
sns.kdeplot(data=FS[['Open', 'Close', 'Volume']])
plt.title('Before MinMaxScaler', size=20)
plt.show()
```

```
C:\Users\AkhilPokuri\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option i
s deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



In [23]:
```
# Assuming FS is your DataFrame containing the required data

import seaborn as sns
import matplotlib.pyplot as plt

# Create a heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(data=FS[['Open', 'Close', 'Volume']], cmap='YlGnBu', annot=True, fmt=".2f")
plt.title('BEFORE MinMaxScaler', size=20)
plt.show()
```

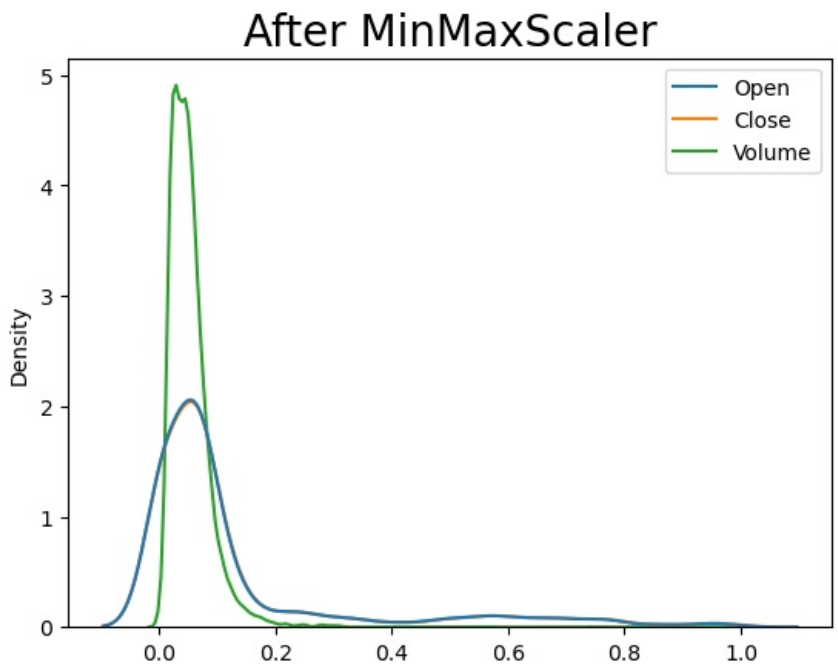
In [22]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming FS is your DataFrame containing the required data

# Create a scatter plot
plt.figure(figsize=(5, 6))
sns.scatterplot(data=FS, x='Open', y='Close', size='Volume', sizes=(10, 20), alpha=0.7)
plt.title('BEFORE MinMaxScaler', size=10)
plt.xlabel('Open', size=14)
plt.ylabel('Close', size=14)
plt.show()
```



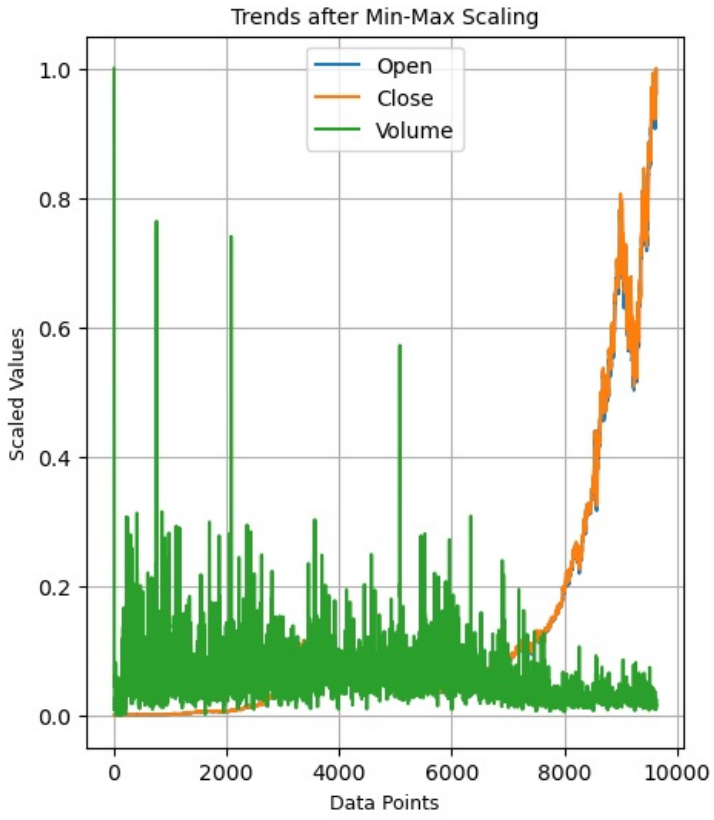
In [16]:
```
sns.kdeplot(taxi_min_max[['Open','Close','Volume']])
plt.title('After MinMaxScaler', size = 20)
```

```
C:\Users\AkhilPokuri\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option i
s deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

Out[16]: Text(0.5, 1.0, 'After MinMaxScaler')



In [26]:
```
plt.figure(figsize=(5, 6))
plt.plot(taxi_min_max['Open'], label='Open')
plt.plot(taxi_min_max['Close'], label='Close')
plt.plot(taxi_min_max['Volume'], label='Volume')
plt.title('Trends after Min-Max Scaling', size=10)
plt.xlabel('Data Points', size=9)
plt.ylabel('Scaled Values', size=9)
plt.legend()
plt.grid(True)
plt.show()
```



In [ ]:

In [ ]: