```
from google.colab import files
uploaded = files.upload()
```

Choose files Stores.csv

- **Stores.csv**(text/csv) - 22545 bytes, last modified: 04/09/2025 - 100% done

```
Saving Stores.csv to Stores.csv
```

```
import pandas as pd

df = pd.read_csv("Stores.csv")

print(df.head())
print(df.shape)
print(df.columns)
```

```
   Store ID   Store_Area  Items_Available  Daily_Customer_Count  Store_Sales
0         1        1659             1961                   530        66490
1         2        1461             1752                   210        39820
2         3        1340             1609                   720        54010
3         4        1451             1748                   620        53730
4         5        1770             2111                   450        46620
(896, 5)
Index(['Store ID ', 'Store_Area', 'Items_Available', 'Daily_Customer_Count',
       'Store_Sales'],
      dtype='object')
```

```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 896 entries, 0 to 895
Data columns (total 5 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Store ID              896 non-null    int64
 1   Store_Area            896 non-null    int64
 2   Items_Available       896 non-null    int64
 3   Daily_Customer_Count  896 non-null    int64
 4   Store_Sales           896 non-null    int64
dtypes: int64(5)
memory usage: 35.1 KB
None
```

```
print(df.describe(include="all").T)
```

```
                      count          mean           std      min       25%  \
Store ID              896.0    448.500000    258.797218      1.0    224.75
Store_Area            896.0   1485.409598    250.237011    775.0   1316.75
Items_Available       896.0   1782.035714    299.872053    932.0   1575.50
Daily_Customer_Count  896.0    786.350446    265.389281     10.0    600.00
Store_Sales           896.0  59351.305804  17190.741895  14920.0  46530.00

                          50%       75%       max
Store ID                448.5    672.25     896.0
Store_Area             1477.0   1653.50    2229.0
Items_Available        1773.5   1982.75    2667.0
Daily_Customer_Count    780.0    970.00    1560.0
Store_Sales           58605.0  71872.50  116320.0
```

```
print(df.isnull().sum())
```

```
Store ID                0
Store_Area              0
Items_Available         0
Daily_Customer_Count    0
Store_Sales             0
dtype: int64
```

```
for col in df.columns:
    print(f"{col} -> {df[col].nunique()} unique values")
```

```
Store ID  -> 896 unique values
Store_Area -> 583 unique values
Items_Available -> 616 unique values
Daily_Customer_Count -> 130 unique values
Store_Sales -> 816 unique values
```

```
print(df.skew())
```

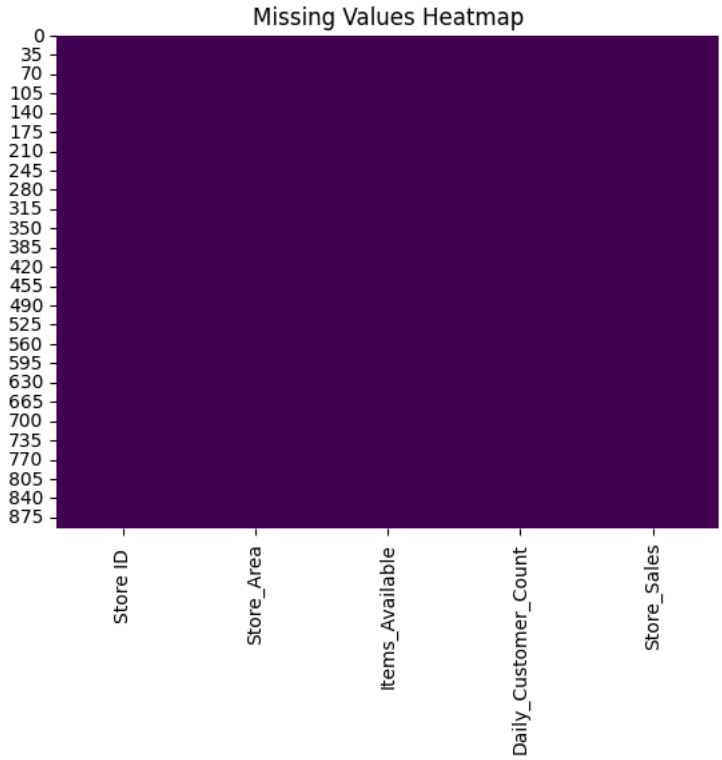
```
Store ID                0.000000
Store_Area              0.030367
Items_Available         0.034439
Daily_Customer_Count    0.074633
Store_Sales             0.148794
dtype: float64
```

```
for col in df.select_dtypes(include=['int64','float64']).columns:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5*IQR
    upper = Q3 + 1.5*IQR
    print(f"{col}: {((df[col] < lower) | (df[col] > upper)).sum()} outliers")
```

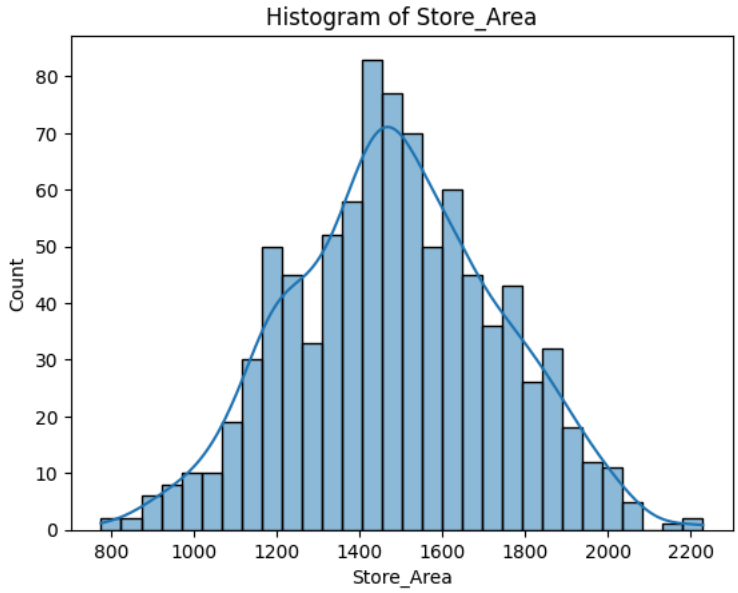
```
Store ID : 0 outliers
Store_Area: 5 outliers
Items_Available: 5 outliers
Daily_Customer_Count: 3 outliers
Store_Sales: 1 outliers
```

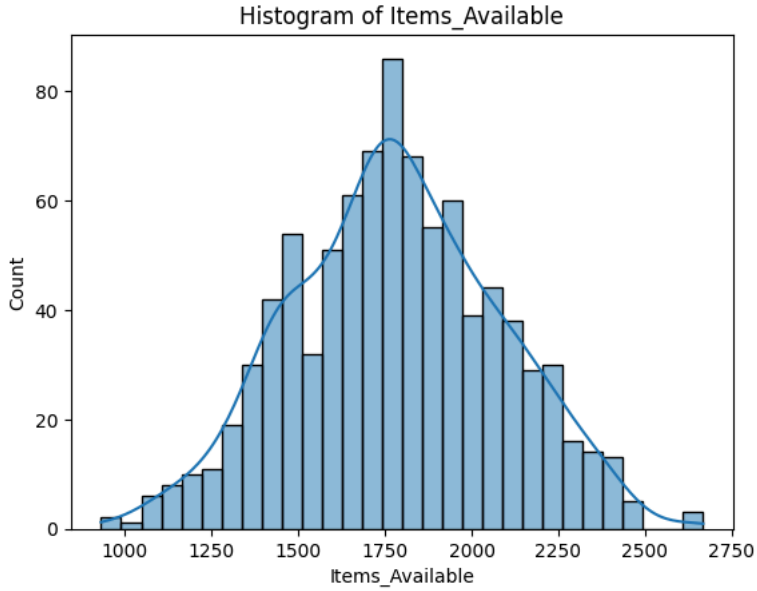
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv("Stores.csv")
```

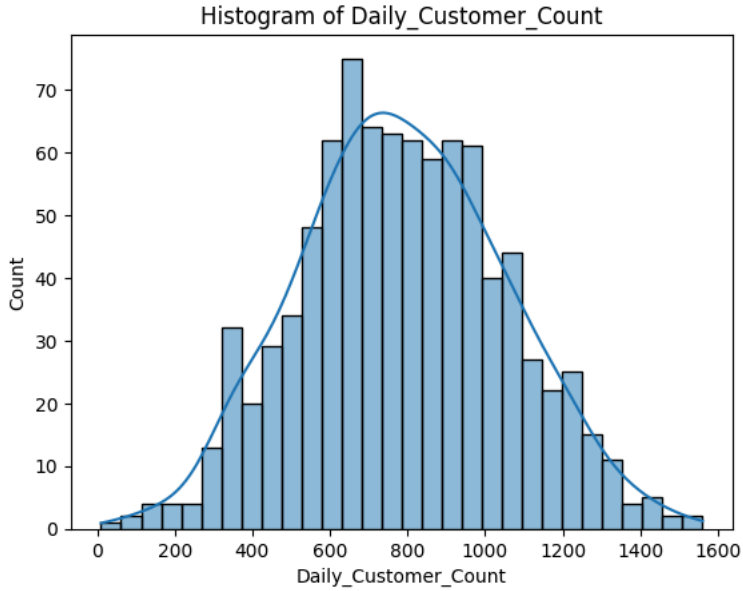
```
sns.heatmap(df.isnull(), cbar=False, cmap="viridis")
plt.title("Missing Values Heatmap")
plt.show()
```

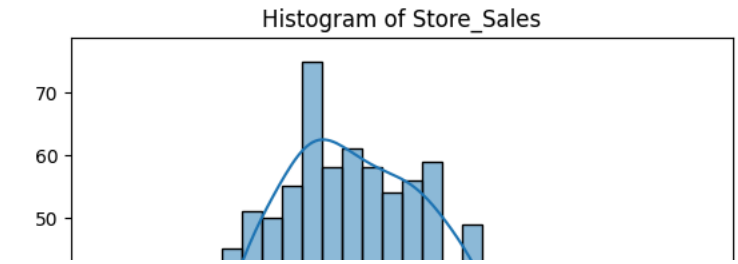


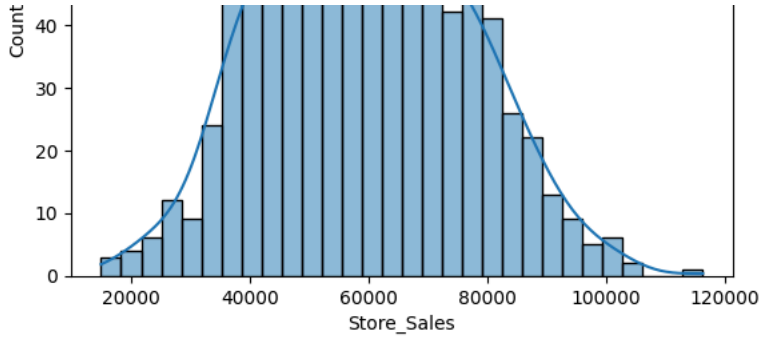

```
for col in df.columns[1:]:
    sns.histplot(df[col], kde=True, bins=30)
    plt.title(f"Histogram of {col}")
    plt.show()
```


Histogram of Store_Area
Count
Store_Area
0
10
20
30
40
50
60
70
80
800
1000
1200
1400
1600
1800
2000
2200

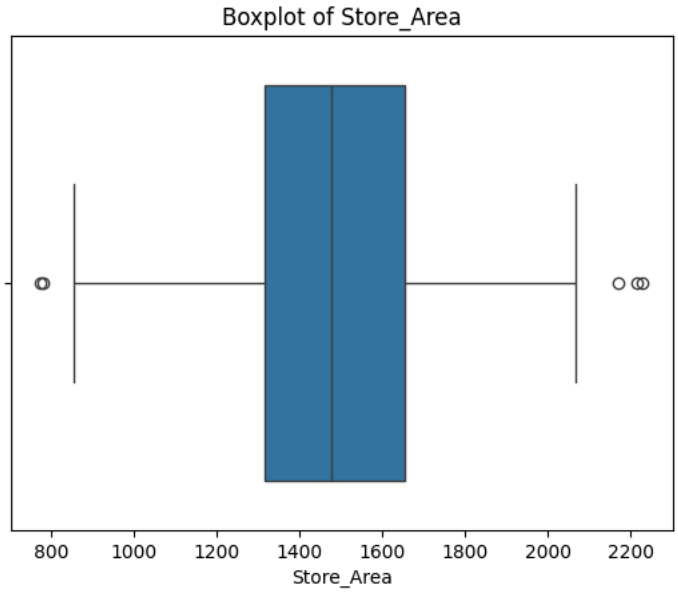

Histogram of Items_Available
Count
Items_Available
0
20
40
60
80
1000
1250
1500
1750
2000
2250
2500
2750

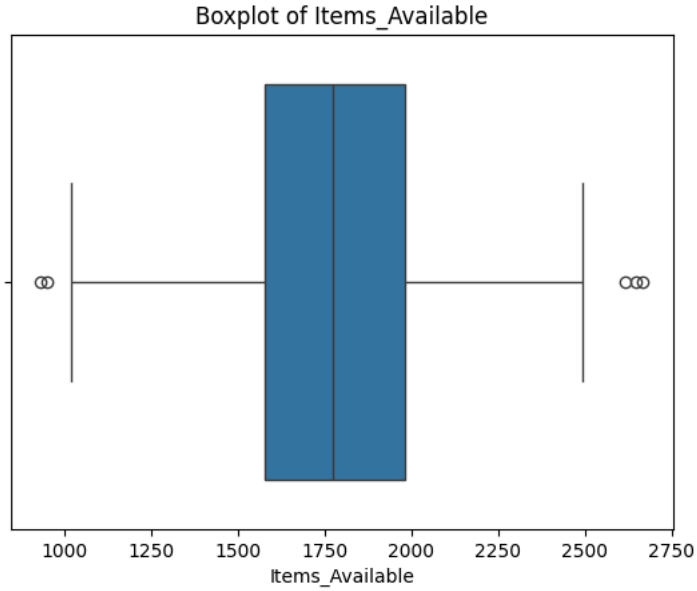

Histogram of Daily_Customer_Count
Count
Daily_Customer_Count
0
10
20
30
40
50
60
70
0
200
400
600
800
1000
1200
1400
1600

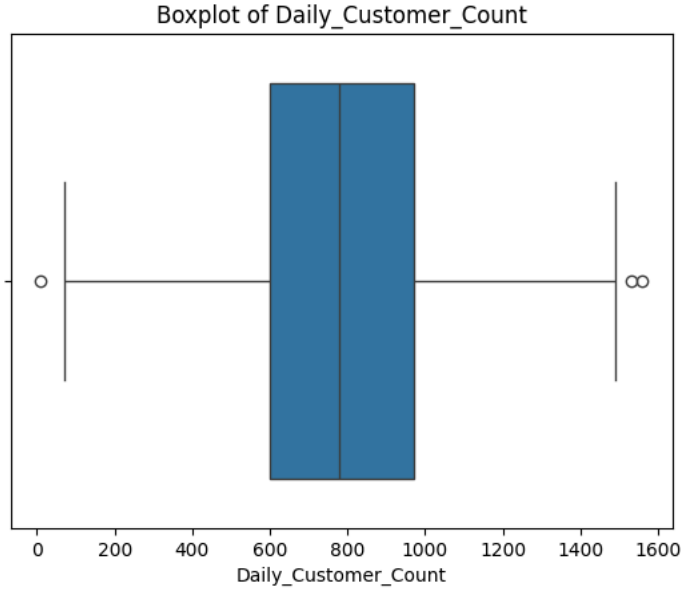

Histogram of Store_Sales
50
60
70

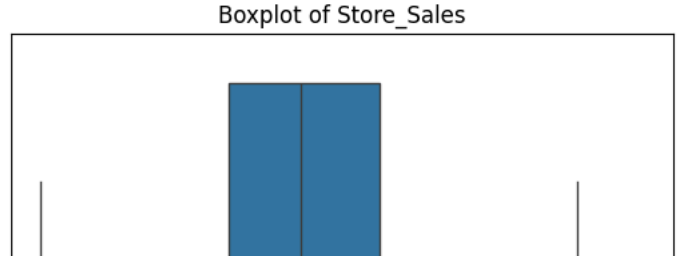


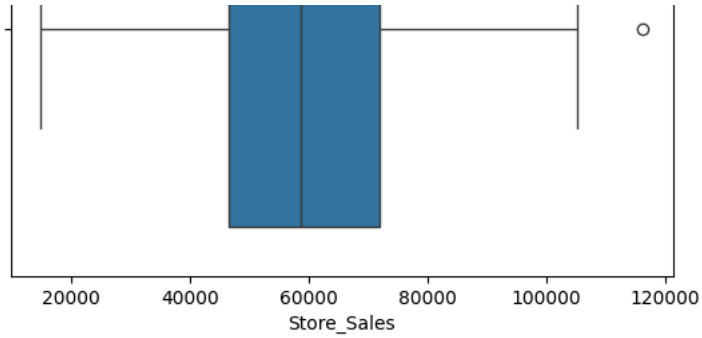

```
for col in df.columns[1:]:
    sns.boxplot(x=df[col])
    plt.title(f"Boxplot of {col}")
    plt.show()
```


Boxplot of Store_Area
800
1000
1200
1400
1600
1800
2000
2200
Store_Area


Boxplot of Items_Available
1000
1250
1500
1750
2000
2250
2500
2750
Items_Available

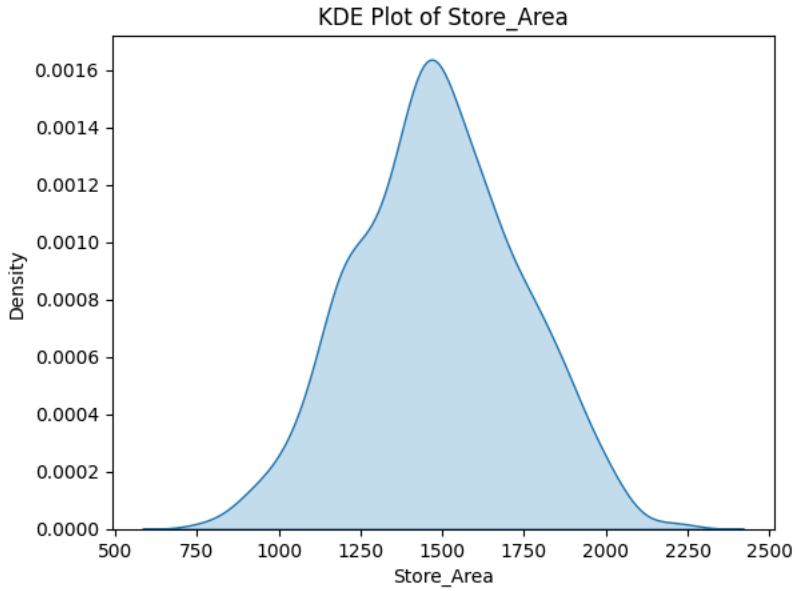

Boxplot of Daily_Customer_Count
0
200
400
600
800
1000
1200
1400
1600
Daily_Customer_Count


Boxplot of Store_Sales

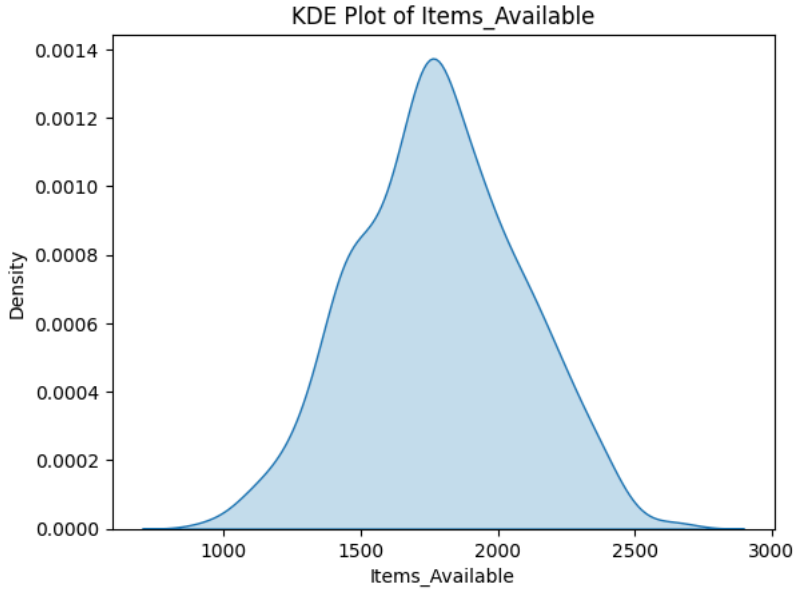



```
for col in df.columns[1:]:
    sns.kdeplot(df[col], shade=True)
    plt.title(f"KDE Plot of {col}")
    plt.show()
```

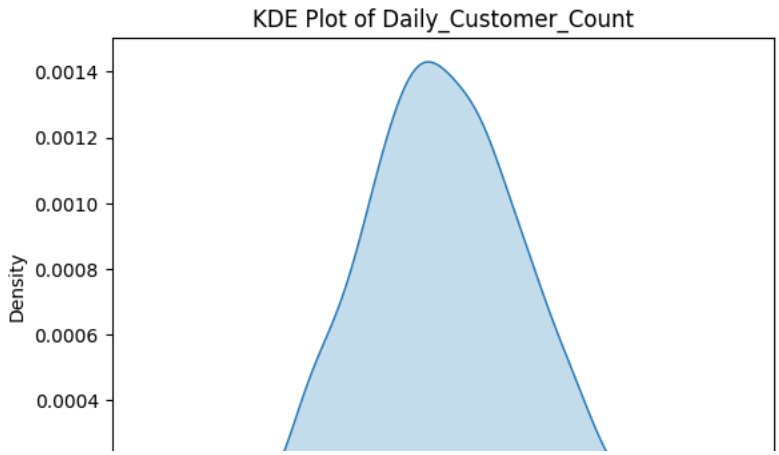
```
/tmp/ipython-input-1999266753.py:2: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(df[col], shade=True)
```

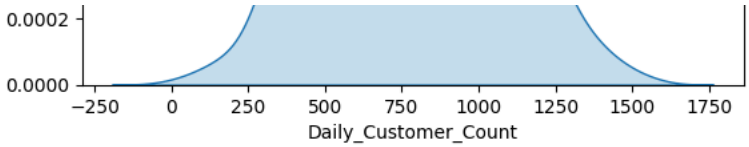



```
/tmp/ipython-input-1999266753.py:2: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(df[col], shade=True)
```

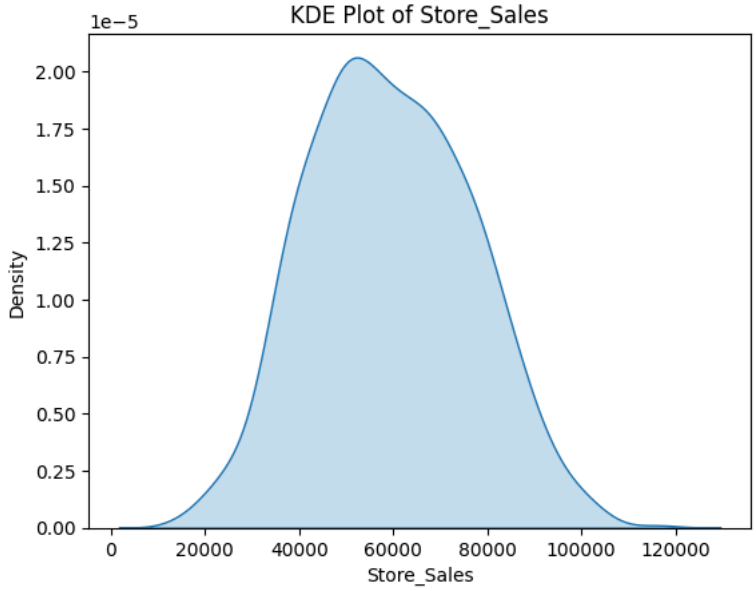



```
/tmp/ipython-input-1999266753.py:2: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(df[col], shade=True)
```



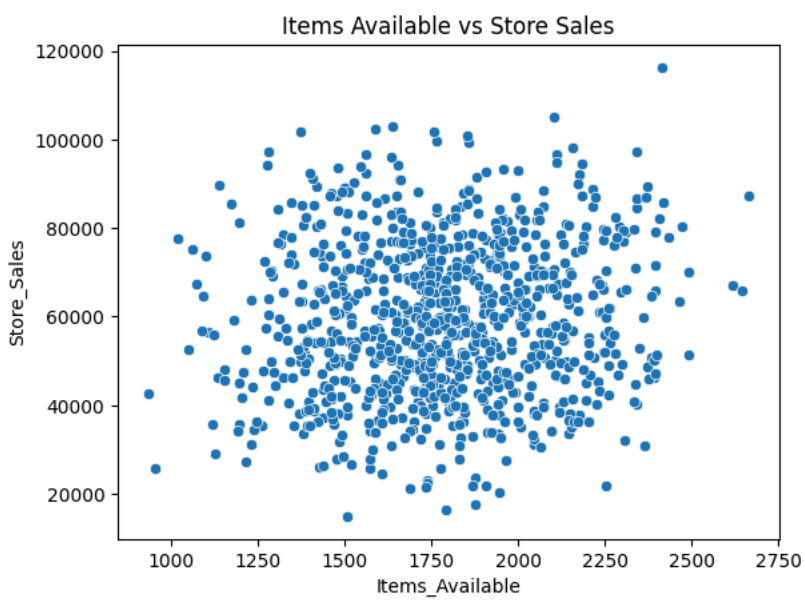



```
/tmp/ipython-input-1999266753.py:2: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(df[col], shade=True)
```



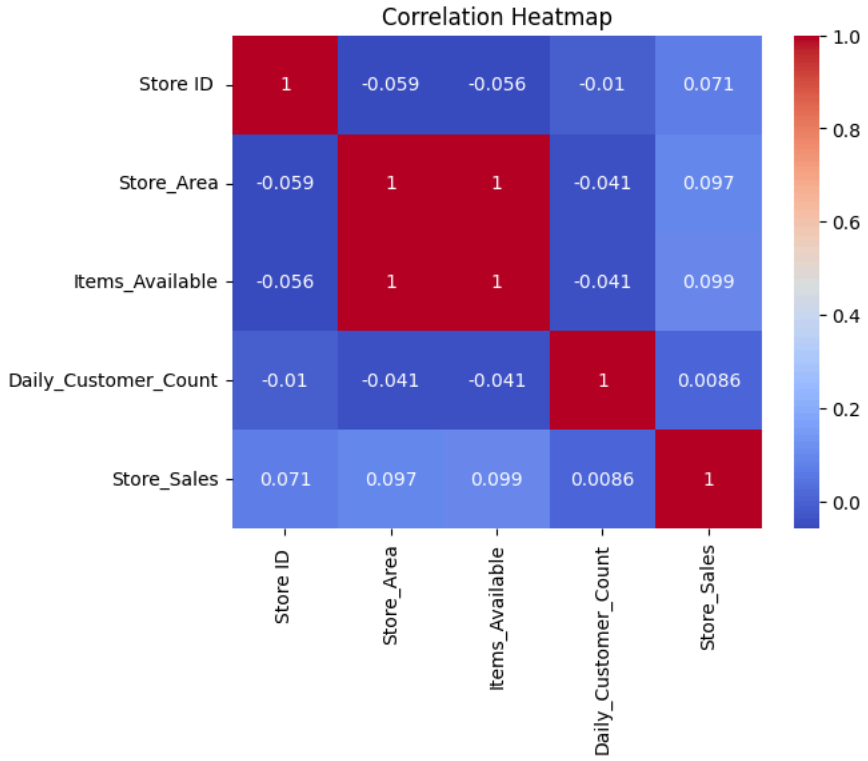

```
sns.scatterplot(x="Store_Area", y="Store_Sales", data=df)
plt.title("Store Area vs Store Sales")
plt.show()
```


```
sns.scatterplot(x="Items_Available", y="Store_Sales", data=df)
plt.title("Items Available vs Store Sales")
plt.show()
```

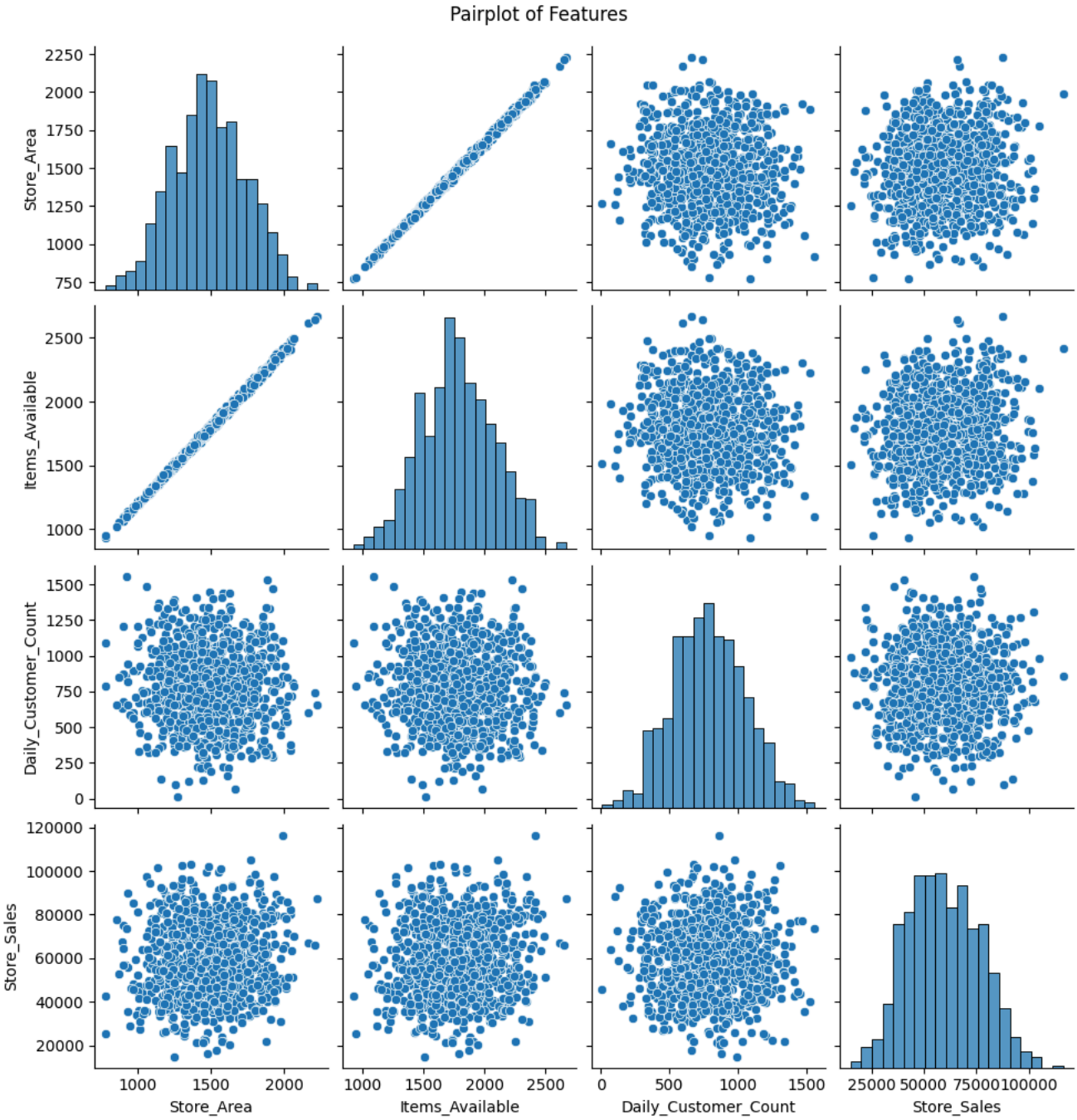



```
sns.scatterplot(x="Daily_Customer_Count", y="Store_Sales", data=df)
plt.title("Customers vs Store Sales")
plt.show()
```

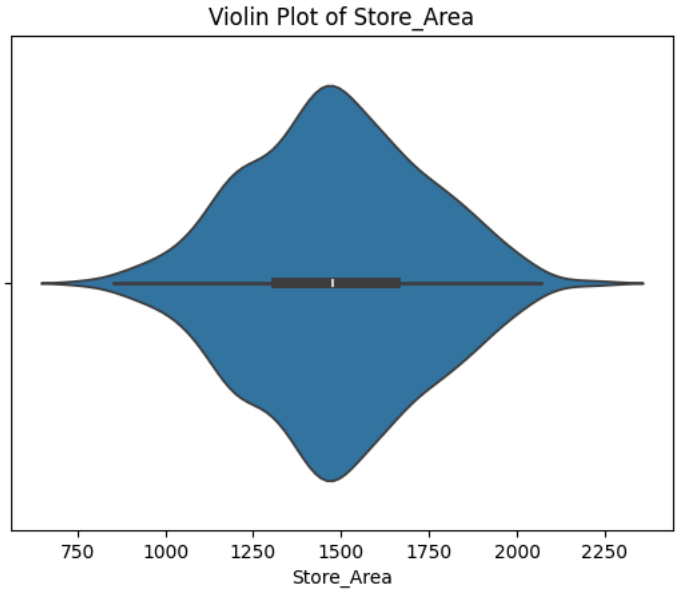

```
sns.heatmap(df.corr(), annot=True, cmap="coolwarm")
plt.title("Correlation Heatmap")
plt.show()
```

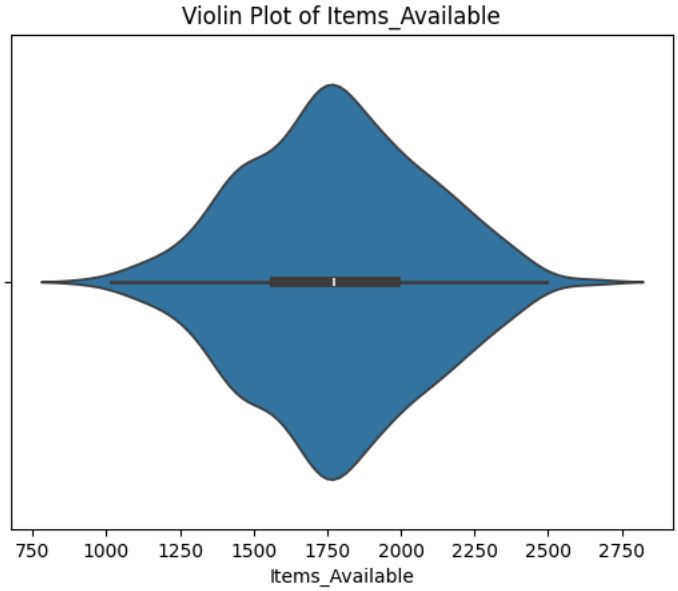


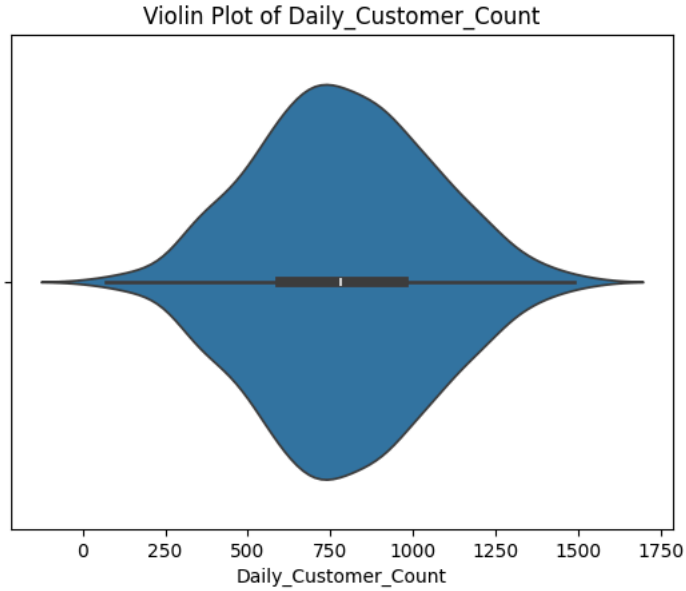

```
sns.pairplot(df.drop("Store ID ", axis=1))
plt.suptitle("Pairplot of Features", y=1.02)
plt.show()
```

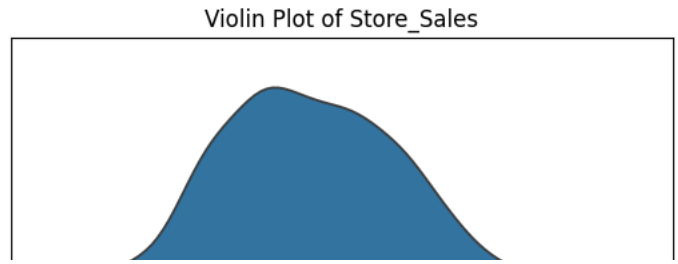


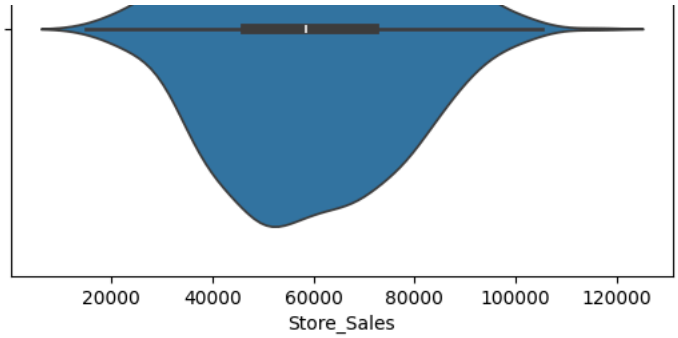

```
for col in ["Store_Area", "Items_Available", "Daily_Customer_Count", "Store_Sales"]:
    sns.violinplot(x=df[col])
    plt.title(f"Violin Plot of {col}")
    plt.show()
```


Violin Plot of Store_Area
750
1000
1250
1500
1750
2000
2250
Store_Area


Violin Plot of Items_Available
750
1000
1250
1500
1750
2000
2250
2500
2750
Items_Available

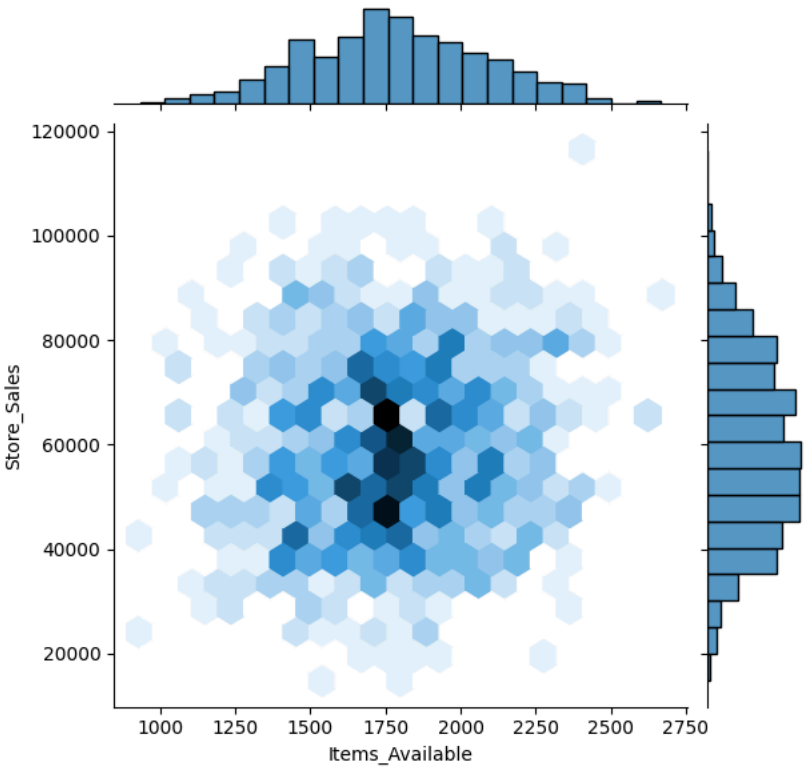

Violin Plot of Daily_Customer_Count
0
250
500
750
1000
1250
1500
1750
Daily_Customer_Count


Violin Plot of Store_Sales



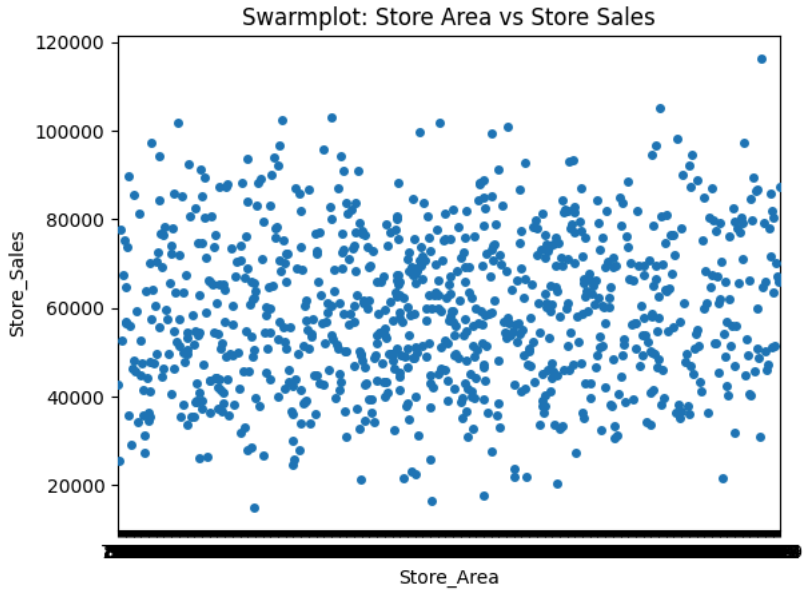

```
sns.regplot(x="Store_Area", y="Store_Sales", data=df)
plt.title("Regression Plot: Store Area vs Store Sales")
plt.show()
```


```
sns.jointplot(x="Items_Available", y="Store_Sales", data=df, kind="hex")
plt.show()
```



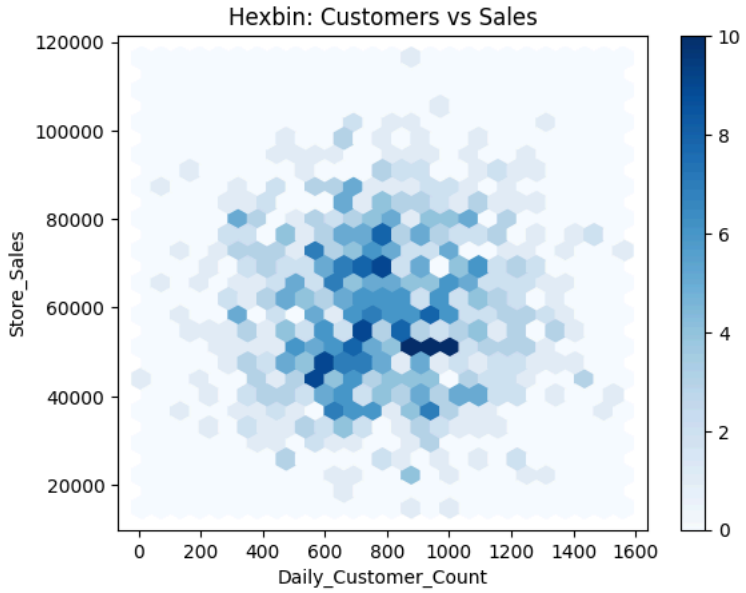
```
sns.swarmplot(x=df["Store_Area"], y=df["Store_Sales"])
plt.title("Swarmplot: Store Area vs Store Sales")
plt.show()
```

```
/usr/local/lib/python3.12/dist-packages/seaborn/categorical.py:3399: UserWarning: 50.0% of the points cannot be placed; you may want
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.12/dist-packages/seaborn/categorical.py:3399: UserWarning: 33.3% of the points cannot be placed; you may want
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.12/dist-packages/seaborn/categorical.py:3399: UserWarning: 20.0% of the points cannot be placed; you may want
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.12/dist-packages/seaborn/categorical.py:3399: UserWarning: 40.0% of the points cannot be placed; you may want
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.12/dist-packages/seaborn/categorical.py:3399: UserWarning: 25.0% of the points cannot be placed; you may want
  warnings.warn(msg, UserWarning)
```

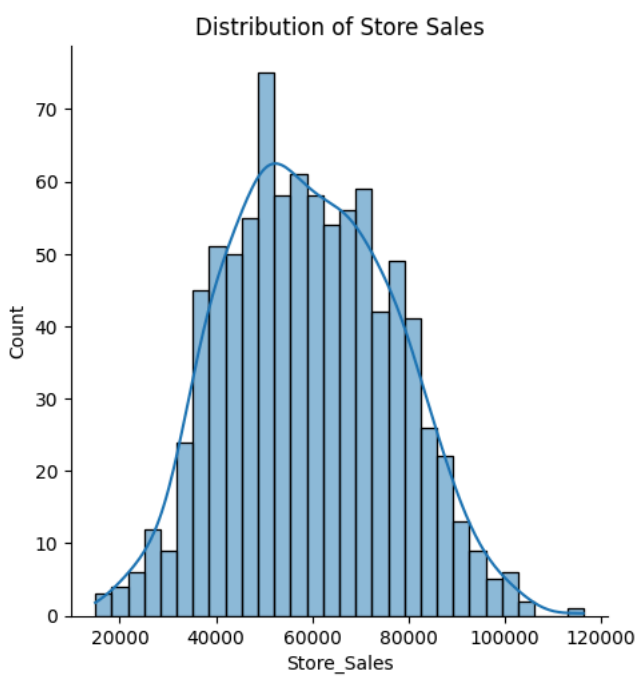



```
sns.stripplot(x=df["Daily_Customer_Count"], y=df["Store_Sales"])
plt.title("Stripplot: Customers vs Sales")
plt.show()
```

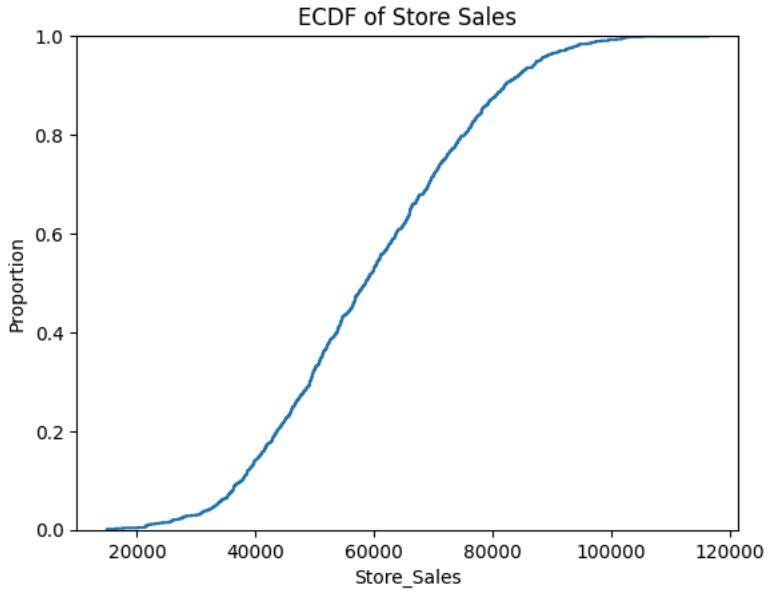

```
df.plot.hexbin(x="Daily_Customer_Count", y="Store_Sales", gridsize=25, cmap="Blues")
plt.title("Hexbin: Customers vs Sales")
plt.show()
```



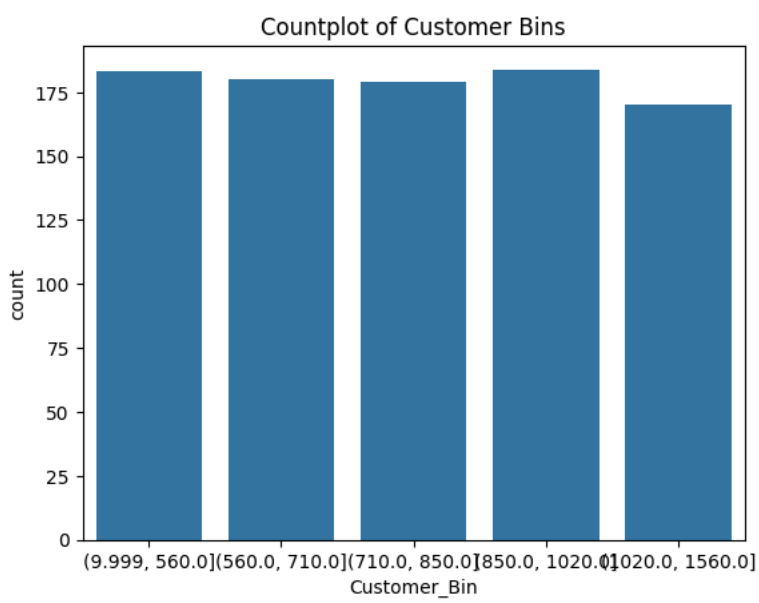

```
sns.displot(df["Store_Sales"], kde=True, bins=30)
plt.title("Distribution of Store Sales")
plt.show()
```




```
sns.ecdfplot(df["Store_Sales"])
plt.title("ECDF of Store Sales")
plt.show()
```

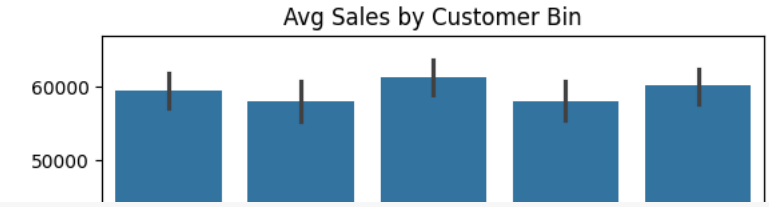



```
df["Customer_Bin"] = pd.qcut(df["Daily_Customer_Count"], q=5)
sns.countplot(x="Customer_Bin", data=df)
plt.title("Countplot of Customer Bins")
plt.show()
```




```
import numpy as np
```

```
sns.barplot(x="Customer_Bin", y="Store_Sales", data=df, estimator=np.mean)
plt.title("Avg Sales by Customer Bin")
plt.show()
```




```
sns.lineplot(x="Store_Area", y="Store_Sales", data=df.sort_values("Store_Area"))
plt.title("Lineplot: Store Area vs Store Sales")
plt.show()
```