Pandas (Mini Project 1)

Analyze a Titanic dataset → clean, filter, analyze, visualize.

```
import pandas as pd
data=pd.read_csv("/content/drive/MyDrive/data/titanic.csv")
print(data)
```

```
     PassengerId  Survived  Pclass  \
0              1         0       3
1              2         1       1
2              3         1       3
3              4         1       1
4              5         0       3
..           ...       ...     ...
886          887         0       2
887          888         1       1
888          889         0       3
889          890         1       1
890          891         0       3

                                                  Name     Sex   Age
SibSp  \
0                              Braund, Mr. Owen Harris    male  22.0
1
1    Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0
1
2                               Heikkinen, Miss. Laina  female  26.0
0
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0
1
4                             Allen, Mr. William Henry    male  35.0
0
..                                                 ...     ...   ...
...
886                              Montvila, Rev. Juozas    male  27.0
0
887                       Graham, Miss. Margaret Edith  female  19.0
0
888           Johnston, Miss. Catherine Helen "Carrie"  female   NaN
1
889                              Behr, Mr. Karl Howell    male  26.0
0
890                                Dooley, Mr. Patrick    male  32.0
0

     Parch            Ticket     Fare Cabin Embarked
```

```
0         0         A/5 21171   7.2500   NaN        S
1         0          PC 17599  71.2833   C85        C
2         0  STON/O2. 3101282   7.9250   NaN        S
3         0            113803  53.1000  C123        S
4         0            373450   8.0500   NaN        S
..      ...               ...      ...   ...      ...
886       0            211536  13.0000   NaN        S
887       0            112053  30.0000   B42        S
888       2        W./C. 6607  23.4500   NaN        S
889       0            111369  30.0000  C148        C
890       0            370376   7.7500   NaN        Q

[891 rows x 12 columns]
```

```
print(data.head())
print("\n")
print(data.info())
print("\n")
print(data.describe())
```

```
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                 Name     Sex   Age
SibSp  \
0                             Braund, Mr. Owen Harris    male  22.0
1
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0
1
2                              Heikkinen, Miss. Laina  female  26.0
0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0
1
4                            Allen, Mr. William Henry    male  35.0
0

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S


<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
```

```
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
None


       PassengerId    Survived      Pclass         Age       SibSp  \
count   891.000000  891.000000  891.000000  714.000000  891.000000
mean    446.000000    0.383838    2.308642   29.699118    0.523008
std     257.353842    0.486592    0.836071   14.526497    1.102743
min       1.000000    0.000000    1.000000    0.420000    0.000000
25%     223.500000    0.000000    2.000000   20.125000    0.000000
50%     446.000000    0.000000    3.000000   28.000000    0.000000
75%     668.500000    1.000000    3.000000   38.000000    1.000000
max     891.000000    1.000000    3.000000   80.000000    8.000000

            Parch        Fare
count  891.000000  891.000000
mean     0.381594   32.204208
std      0.806057   49.693429
min      0.000000    0.000000
25%      0.000000    7.910400
50%      0.000000   14.454200
75%      0.000000   31.000000
max      6.000000  512.329200
```

```
# Data Cleaning
data["Age"]=data["Age"].fillna(data["Age"].median())
data.drop(['Cabin','Ticket','Name'],
axis=1,inplace=True,errors='ignore')
print(data)
```

```
     PassengerId  Survived  Pclass     Sex   Age  SibSp  Parch
Fare  \
0              1         0       3    male  22.0      1      0
7.2500
```

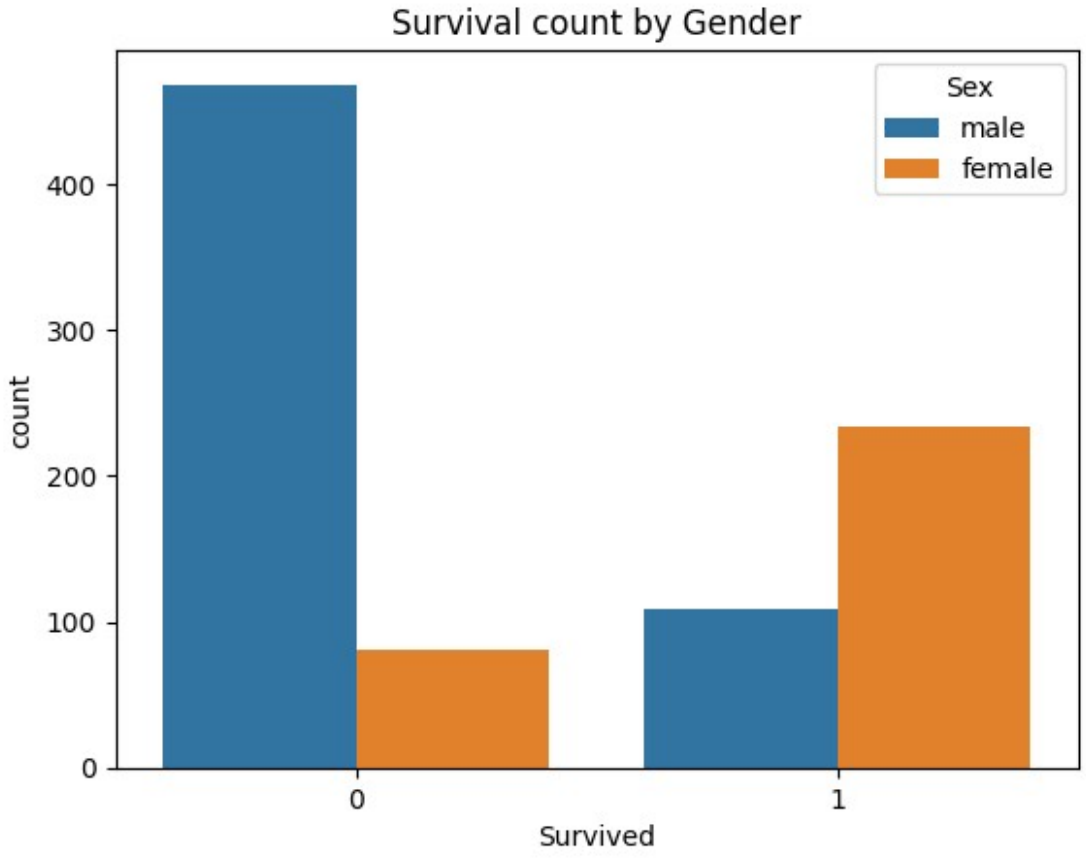
```
1              2         1       1  female  38.0      1      0
71.2833
2              3         1       3  female  26.0      0      0
7.9250
3              4         1       1  female  35.0      1      0
53.1000
4              5         0       3    male  35.0      0      0
8.0500
..           ...       ...     ...     ...   ...    ...    ...       ..
.
886          887         0       2    male  27.0      0      0
13.0000
887          888         1       1  female  19.0      0      0
30.0000
888          889         0       3  female  28.0      1      2
23.4500
889          890         1       1    male  26.0      0      0
30.0000
890          891         0       3    male  32.0      0      0
7.7500

    Embarked
0          S
1          C
2          S
3          S
4          S
..       ...
886        S
887        S
888        S
889        C
890        Q

[891 rows x 9 columns]
```

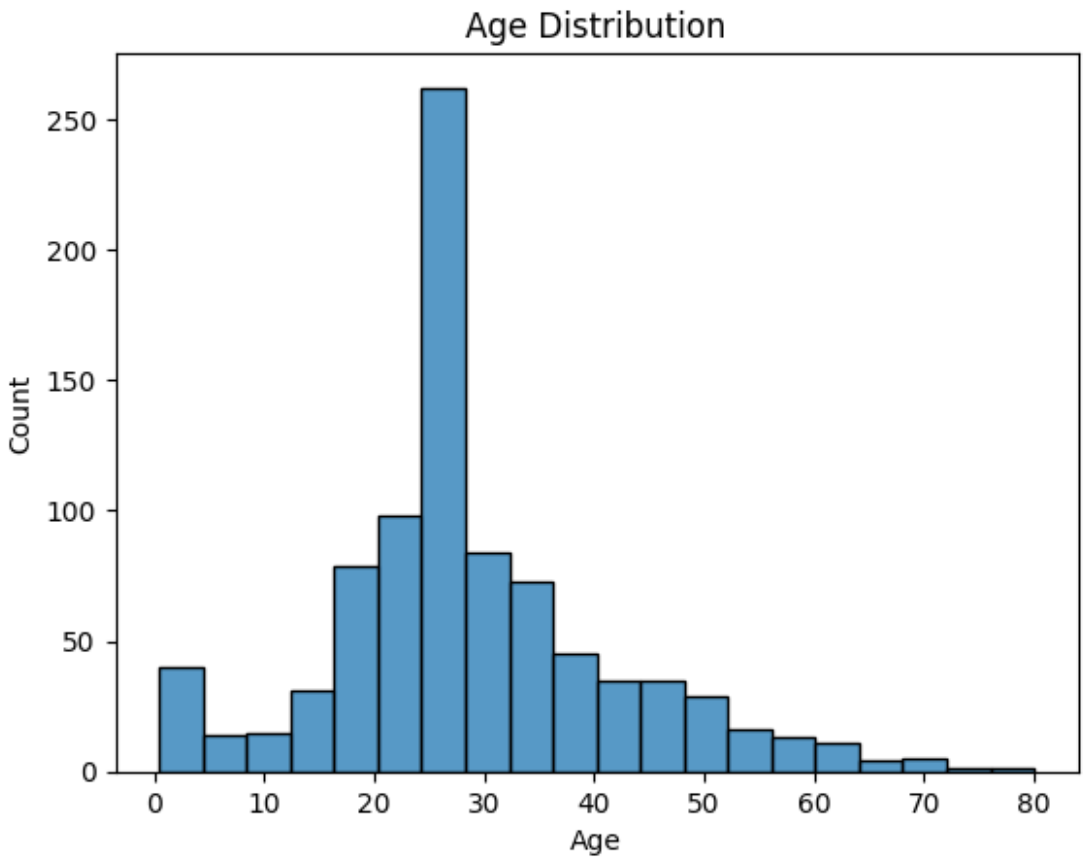
```python
#Analyze
#1.survival rate by gender
print(data.groupby('Sex')['Survived'].mean())
#2.Survival rate by class
print(data.groupby('Pclass')['Survived'].mean())
#3.Average age of survivors vs non-survivors
print(data.groupby('Survived')['Age'].mean())
```

```
Sex
female    0.742038
male      0.188908
Name: Survived, dtype: float64
Pclass
1    0.629630
```

```
2    0.472826
3    0.242363
Name: Survived, dtype: float64
Survived
0    30.028233
1    28.291433
Name: Age, dtype: float64
```

```
#Visualize
import matplotlib.pyplot as plt
import seaborn as sns

sns.countplot(x='Survived',hue='Sex',data=data)
plt.title("Survival count by Gender")
plt.show()

sns.histplot(data['Age'],bins=20)
plt.title("Age Distribution")
plt.show()
```




Age Distribution
Count
Age
0
50
100
150
200
250
0
10
20
30
40
50
60
70
80

Pandas (Mini Project 2)

Analyze a sales dataset → find top products, monthly sales

```
import pandas as pd
data=pd.read_csv("/content/drive/MyDrive/data/retail_store_sales.csv")
print(data)
```

```
      Transaction ID Customer ID       Category          Item  Price
Per Unit  \
0        TXN_6867343     CUST_09     Patisserie   Item_10_PAT
18.5
1        TXN_3731986     CUST_22  Milk Products  Item_17_MILK
29.0
2        TXN_9303719     CUST_02       Butchers   Item_12_BUT
21.5
3        TXN_9458126     CUST_06      Beverages   Item_16_BEV
27.5
4        TXN_4575373     CUST_05           Food   Item_6_FOOD
12.5
...              ...         ...            ...           ...
...
12570    TXN_9347481     CUST_18     Patisserie   Item_23_PAT
38.0
12571    TXN_4009414     CUST_03      Beverages    Item_2_BEV
6.5
12572    TXN_5306010     CUST_11       Butchers    Item_7_BUT
14.0
12573    TXN_5167298     CUST_04      Furniture    Item_7_FUR
14.0
12574    TXN_2407494     CUST_23           Food   Item_9_FOOD
17.0

       Quantity  Total Spent  Payment Method  Location Transaction
Date  \
0          10.0        185.0  Digital Wallet    Online       2024-04-
08
1           9.0        261.0  Digital Wallet    Online       2023-07-
23
2           2.0         43.0     Credit Card    Online       2022-10-
05
3           9.0        247.5     Credit Card    Online       2022-05-
07
4           7.0         87.5  Digital Wallet    Online       2022-10-
02
...         ...          ...             ...       ...              ..
.
```

```
12570       4.0        152.0     Credit Card  In-store       2023-09-
03
12571       9.0         58.5            Cash    Online       2022-08-
12
12572      10.0        140.0            Cash    Online       2024-08-
24
12573       6.0         84.0            Cash    Online       2023-12-
30
12574       3.0         51.0            Cash    Online       2022-08-
06

      Discount Applied
0                 True
1                 True
2                False
3                  NaN
4                False
...                ...
12570              NaN
12571            False
12572              NaN
12573             True
12574              NaN

[12575 rows x 11 columns]
```

```
print(data.head())
print("\n")
print(data.info())
print("\n")
print(data.describe())
```

```
  Transaction ID Customer ID       Category          Item  Price Per
Unit  \
0    TXN_6867343     CUST_09     Patisserie   Item_10_PAT
18.5
1    TXN_3731986     CUST_22  Milk Products  Item_17_MILK
29.0
2    TXN_9303719     CUST_02       Butchers   Item_12_BUT
21.5
3    TXN_9458126     CUST_06      Beverages   Item_16_BEV
27.5
4    TXN_4575373     CUST_05           Food   Item_6_FOOD
12.5

   Quantity  Total Spent  Payment Method Location Transaction Date  \
0      10.0        185.0  Digital Wallet   Online       2024-04-08
1       9.0        261.0  Digital Wallet   Online       2023-07-23
2       2.0         43.0     Credit Card   Online       2022-10-05
3       9.0        247.5     Credit Card   Online       2022-05-07
```

```
4       7.0         87.5  Digital Wallet   Online       2022-10-02
```

```
  Discount Applied
0             True
1             True
2            False
3              NaN
4            False
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12575 entries, 0 to 12574
Data columns (total 11 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Transaction ID    12575 non-null  object 
 1   Customer ID       12575 non-null  object 
 2   Category          12575 non-null  object 
 3   Item              11362 non-null  object 
 4   Price Per Unit    11966 non-null  float64
 5   Quantity          11971 non-null  float64
 6   Total Spent       11971 non-null  float64
 7   Payment Method    12575 non-null  object 
 8   Location          12575 non-null  object 
 9   Transaction Date  12575 non-null  object 
 10  Discount Applied  8376 non-null   object 
dtypes: float64(3), object(8)
memory usage: 1.1+ MB
None
```

```
       Price Per Unit      Quantity   Total Spent
count    11966.000000  11971.000000  11971.000000
mean        23.365912      5.536380    129.652577
std         10.743519      2.857883     94.750697
min          5.000000      1.000000      5.000000
25%         14.000000      3.000000     51.000000
50%         23.000000      6.000000    108.500000
75%         33.500000      8.000000    192.000000
max         41.000000     10.000000    410.000000
```

```
data.isna().sum()
```

```
Transaction ID         0
Customer ID            0
Category               0
Item                1213
Price Per Unit       609
Quantity             604
Total Spent          604
```

```
Payment Method          0
Location                0
Transaction Date        0
Discount Applied     4199
dtype: int64
```

Data Cleaning

```
#clean data
data['Discount Applied'].fillna(data['Discount Applied'].mode()[0],
inplace =True)
print(data['Discount Applied'])


0          True
1          True
2         False
3          True
4         False
          ...
12570      True
12571     False
12572      True
12573      True
12574      True
Name: Discount Applied, Length: 12575, dtype: bool

/tmp/ipython-input-3981063975.py:2: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  data['Discount Applied'].fillna(data['Discount Applied'].mode()[0],
inplace =True)
/tmp/ipython-input-3981063975.py:2: FutureWarning: Downcasting object
dtype arrays on .fillna, .ffill, .bfill is deprecated and will change
in a future version. Call result.infer_objects(copy=False) instead. To
opt-in to the future behavior, set
`pd.set_option('future.no_silent_downcasting', True)`
  data['Discount Applied'].fillna(data['Discount Applied'].mode()[0],
inplace =True)
```

```
print(data.info())           # ensure no missing values
print(data['Discount Applied'].value_counts())  # see counts of
True/False
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12575 entries, 0 to 12574
Data columns (total 11 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Transaction ID    12575 non-null  object
 1   Customer ID       12575 non-null  object
 2   Category          12575 non-null  object
 3   Item              11362 non-null  object
 4   Price Per Unit    11966 non-null  float64
 5   Quantity          11971 non-null  float64
 6   Total Spent       11971 non-null  float64
 7   Payment Method    12575 non-null  object
 8   Location          12575 non-null  object
 9   Transaction Date  12575 non-null  object
 10  Discount Applied  12575 non-null  bool
dtypes: bool(1), float64(3), object(7)
memory usage: 994.8+ KB
None
Discount Applied
True     8418
False    4157
Name: count, dtype: int64
```

```
#Item Item (Purchased Item Name)

# Type: Categorical / string
# Possible Issues:
# Typos or inconsistent capitalization (Laptop vs laptop)
# Leading/trailing space
# Missing values

data['Item']=data['Item'].str.strip()
data['Item']=data['Item'].str.title()
#data['Item'].fillna('unknown', inplace=True)
data['Item'] = data['Item'].fillna('unknown')
print(data['Item'].value_counts())
```

```
Item
unknown         1213
Item_2_Bev       126
Item_25_Fur      113
Item_11_Fur      110
Item_16_Milk     109
                ...
Item_5_Bev         7
```

```
Item_13_Bev         7
Item_13_Fur         7
Item_21_Pat         6
Item_3_Ehe          5
Name: count, Length: 201, dtype: int64
```

```
#Price per Unit
# Type: Numeric (float)
# Possible Issues:
# Missing values
# Incorrect negative or zero values

data['Price Per Unit']=data['Price Per Unit'].fillna(data['Price Per Unit'].median())
print(data['Price Per Unit'].value_counts())

data=data[data['Price Per Unit']>0]
print(data['Price Per Unit'].describe())
```

```
Price Per Unit
23.0    1117
33.5     678
20.0     634
21.5     630
41.0     593
29.0     554
15.5     554
38.0     542
6.5      527
12.5     522
26.0     507
11.0     497
27.5     497
32.0     466
36.5     451
24.5     437
14.0     421
5.0      419
9.5      414
39.5     387
8.0      380
35.0     371
30.5     355
17.0     335
18.5     287
Name: count, dtype: int64
count    12575.000000
mean        23.348191
std         10.480413
min          5.000000
```

```
25%         14.000000
50%         23.000000
75%         32.000000
max         41.000000
Name: Price Per Unit, dtype: float64
```

```
#Quantity
# Type: Numeric (integer)
# Possible Issues:
# Missing values
# Negative or zero quantity

data['Quantity']=data['Quantity'].fillna(data['Quantity'].median())
data=data[data['Quantity']>0]
print(data['Quantity'].describe())
```

```
count    12575.000000
mean         5.558648
std          2.790160
min          1.000000
25%          3.000000
50%          6.000000
75%          8.000000
max         10.000000
Name: Quantity, dtype: float64
```

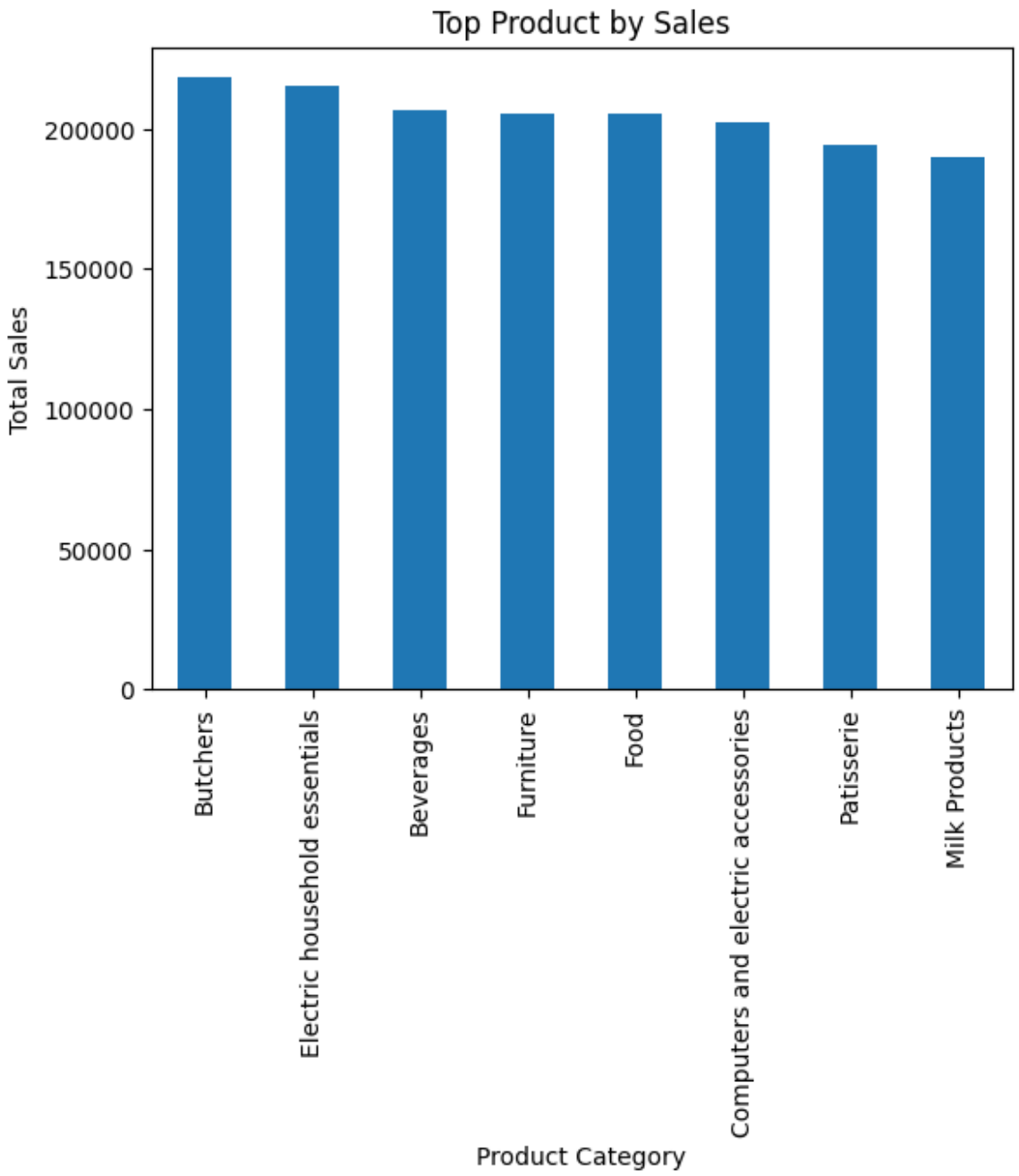
```
#Total Spent
#Type: Numeric (float)
# Possible Issues:
# Missing values
# Incorrect values (not equal to PricePerUnit * Quantity)

data['Total Spent']=data['Total Spent'].fillna(data['Price Per
Unit']*data['Quantity'])
print(data['Total Spent'].describe())
```

```
count    12575.000000
mean       130.208111
std         93.580667
min          5.000000
25%         52.000000
50%        110.000000
75%        192.000000
max        410.000000
Name: Total Spent, dtype: float64
```

```
#Top product
top_product=data.groupby('Category')['Total
Spent'].sum().sort_values(ascending=False)
print(top_product)
```

```
Category
Butchers                                  218153.0
Electric household essentials             215297.5
Beverages                                 206854.5
Furniture                                 205390.0
Food                                      205225.0
Computers and electric accessories        202023.5
Patisserie                                194531.5
Milk Products                             189892.0
Name: Total Spent, dtype: float64
```

```
import matplotlib.pyplot as plt

top_product.plot(kind='bar',title='Top Product by Sales')
plt.xlabel("Product Category")
plt.ylabel("Total Sales")
plt.show()
```




```
print(data.head(10))
```

```
  Transaction ID Customer ID       Category          Item  Price Per
Unit  \
0    TXN_6867343     CUST_09     Patisserie   Item_10_Pat
18.5
1    TXN_3731986     CUST_22  Milk Products  Item_17_Milk
29.0
2    TXN_9303719     CUST_02       Butchers   Item_12_But
21.5
```

```
3    TXN_9458126     CUST_06      Beverages   Item_16_Bev
27.5
4    TXN_4575373     CUST_05           Food   Item_6_Food
12.5
5    TXN_7482416     CUST_09     Patisserie       unknown
23.0
6    TXN_3652209     CUST_07           Food   Item_1_Food
5.0
7    TXN_1372952     CUST_21      Furniture       unknown
33.5
8    TXN_9728486     CUST_23      Furniture   Item_16_Fur
27.5
9    TXN_2722661     CUST_25       Butchers   Item_22_But
36.5

   Quantity  Total Spent  Payment Method  Location Transaction Date  \
0      10.0        185.0  Digital Wallet    Online       2024-04-08
1       9.0        261.0  Digital Wallet    Online       2023-07-23
2       2.0         43.0     Credit Card    Online       2022-10-05
3       9.0        247.5     Credit Card    Online       2022-05-07
4       7.0         87.5  Digital Wallet    Online       2022-10-02
5      10.0        200.0     Credit Card    Online       2023-11-30
6       8.0         40.0     Credit Card  In-store       2023-06-10
7       6.0        201.0  Digital Wallet  In-store       2024-04-02
8       1.0         27.5     Credit Card  In-store       2023-04-26
9       3.0        109.5            Cash    Online       2024-03-14

   Discount Applied
0              True
1              True
2             False
3              True
4             False
5              True
6              True
7              True
8             False
9             False
```

```
#Verifying the cleaned data:
print(data.info())        # no missing values, correct data types
print(data.describe())    # summary stats for numeric columns
print(data['Item'].value_counts())  # check categories
print(data['Discount Applied'].value_counts())  # 0/1 counts
```

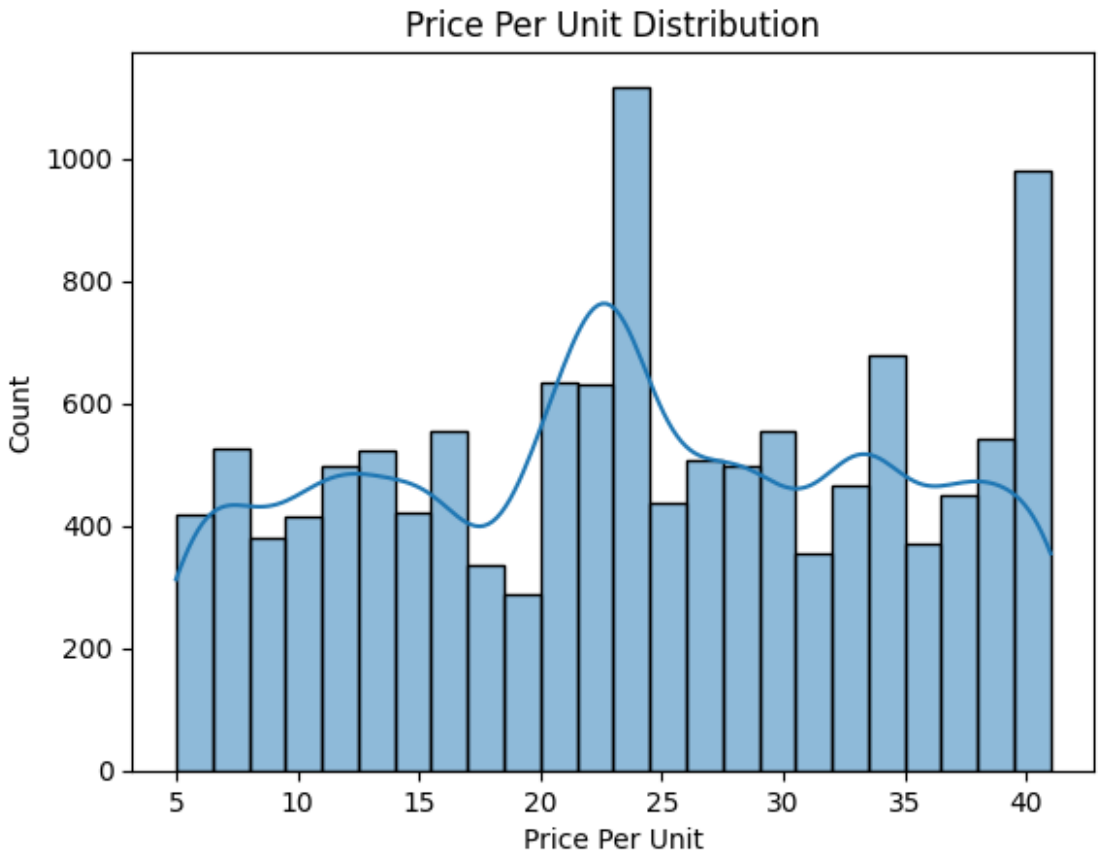
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12575 entries, 0 to 12574
Data columns (total 11 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
```

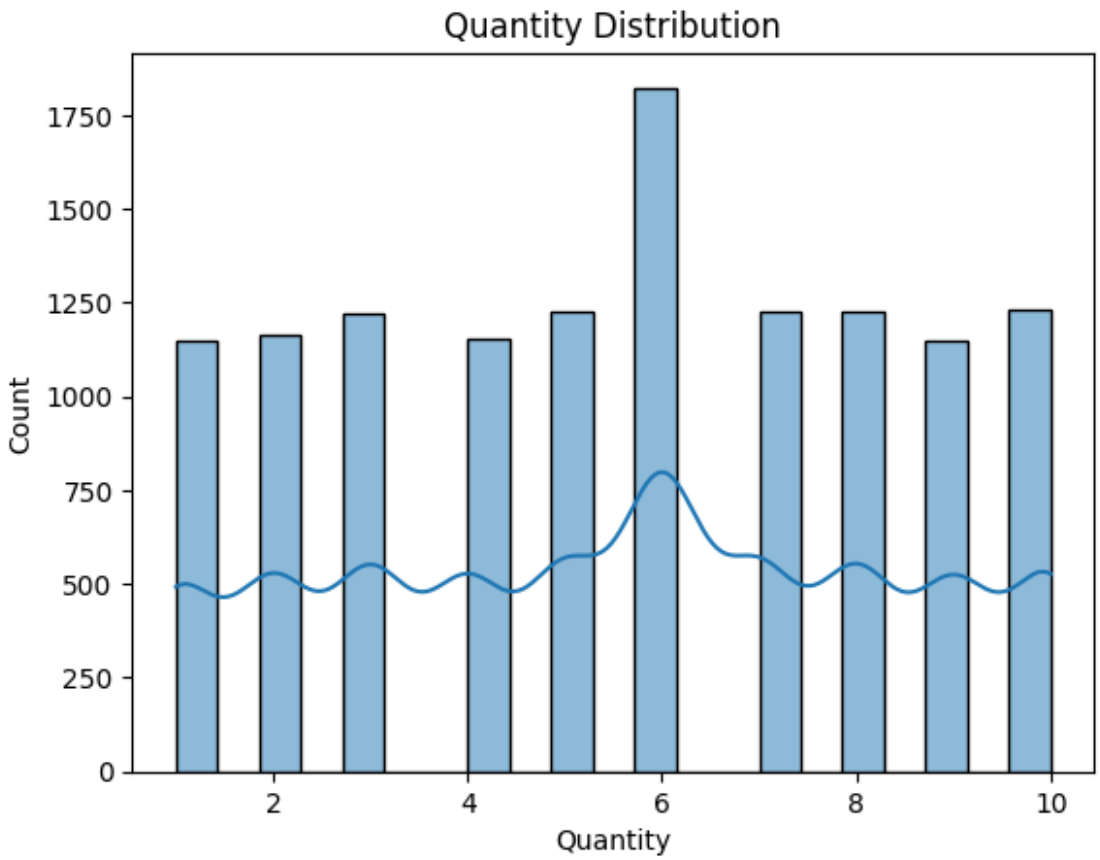
```
 0   Transaction ID    12575 non-null  object
 1   Customer ID       12575 non-null  object
 2   Category          12575 non-null  object
 3   Item              12575 non-null  object
 4   Price Per Unit    12575 non-null  float64
 5   Quantity          12575 non-null  float64
 6   Total Spent       12575 non-null  float64
 7   Payment Method    12575 non-null  object
 8   Location          12575 non-null  object
 9   Transaction Date  12575 non-null  object
 10  Discount Applied  12575 non-null  bool
dtypes: bool(1), float64(3), object(7)
memory usage: 994.8+ KB
None
       Price Per Unit      Quantity   Total Spent
count    12575.000000  12575.000000  12575.000000
mean        23.348191      5.558648    130.208111
std         10.480413      2.790160     93.580667
min          5.000000      1.000000      5.000000
25%         14.000000      3.000000     52.000000
50%         23.000000      6.000000    110.000000
75%         32.000000      8.000000    192.000000
max         41.000000     10.000000    410.000000
Item
unknown         1213
Item_2_Bev       126
Item_25_Fur      113
Item_11_Fur      110
Item_16_Milk     109
                ...
Item_5_Bev         7
Item_13_Bev        7
Item_13_Fur        7
Item_21_Pat        6
Item_3_Ehe         5
Name: count, Length: 201, dtype: int64
Discount Applied
True     8418
False    4157
Name: count, dtype: int64
```

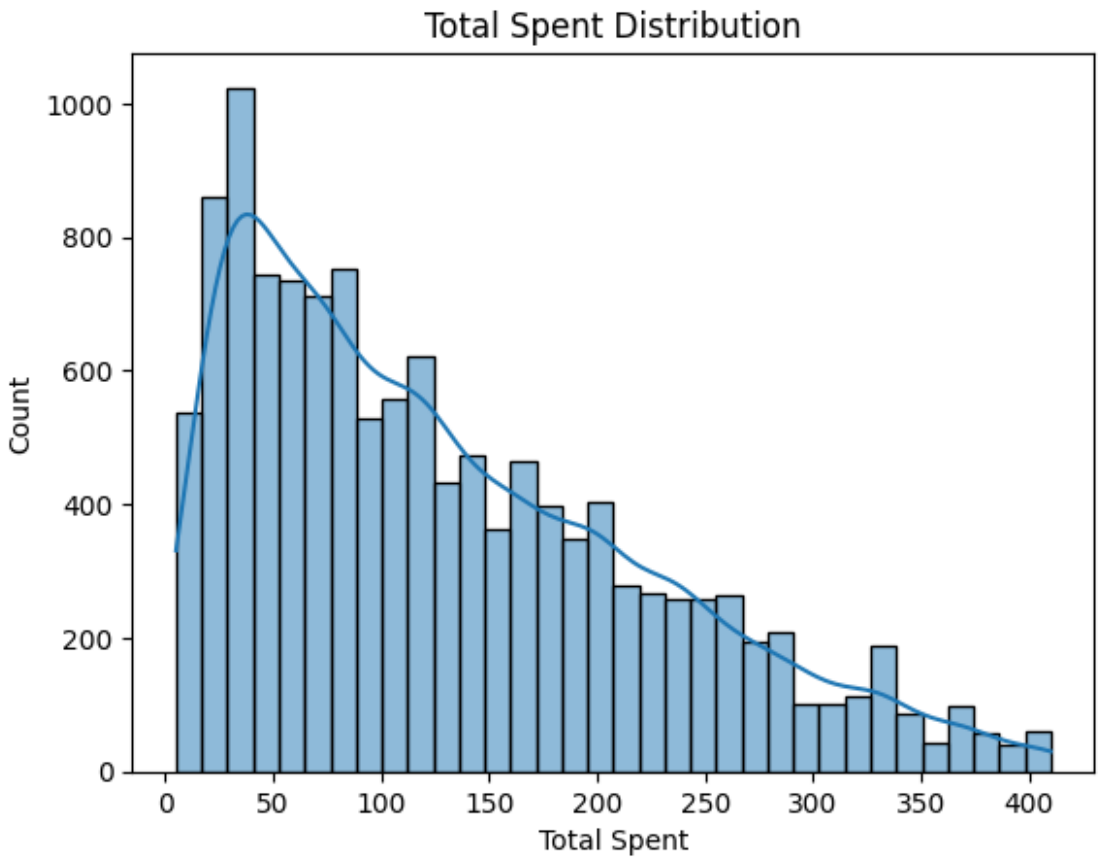
Univariate the Analysis (One column at a time)

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

#numeric analysis'
sns.histplot(data['Price Per Unit'],kde=True)
plt.title('Price Per Unit Distribution')
```

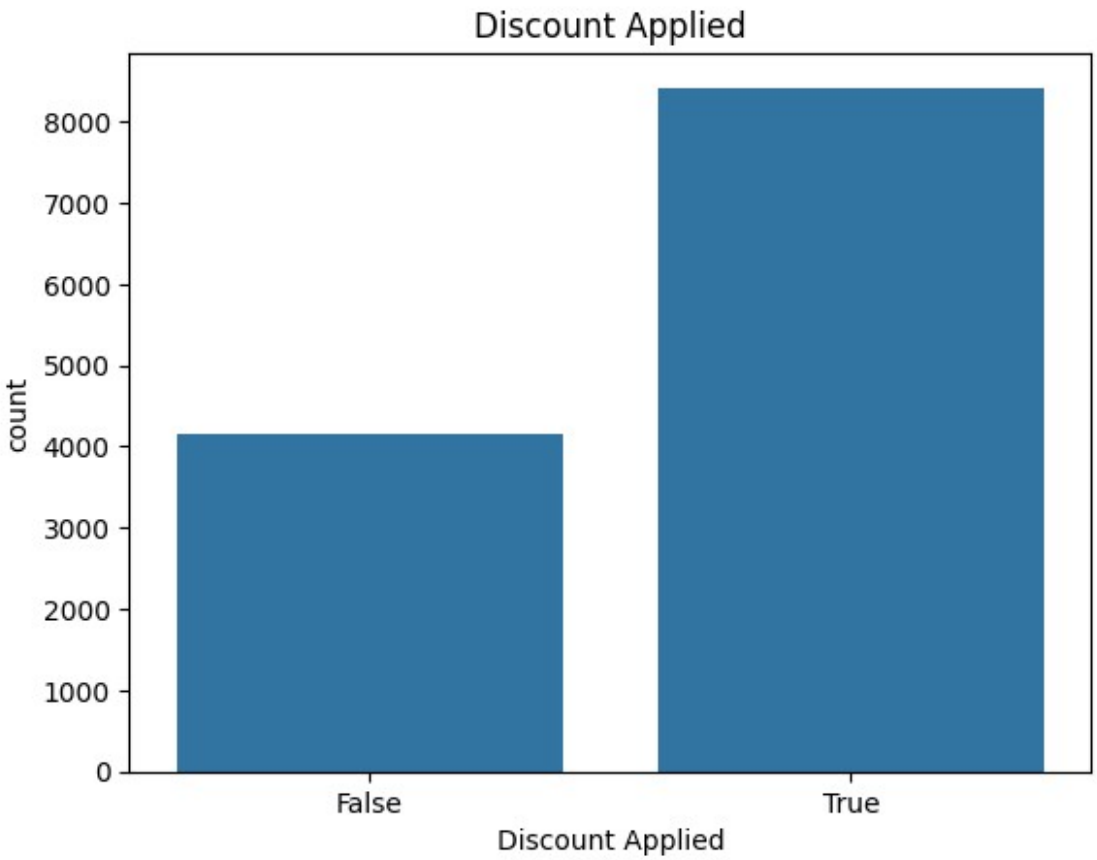
```
plt.show()
#print(data['Price Per Unit'].describe())

sns.histplot(data['Quantity'], kde=True)
plt.title("Quantity Distribution")
plt.show()

sns.histplot(data['Total Spent'], kde=True)
plt.title("Total Spent Distribution")
plt.show()
```

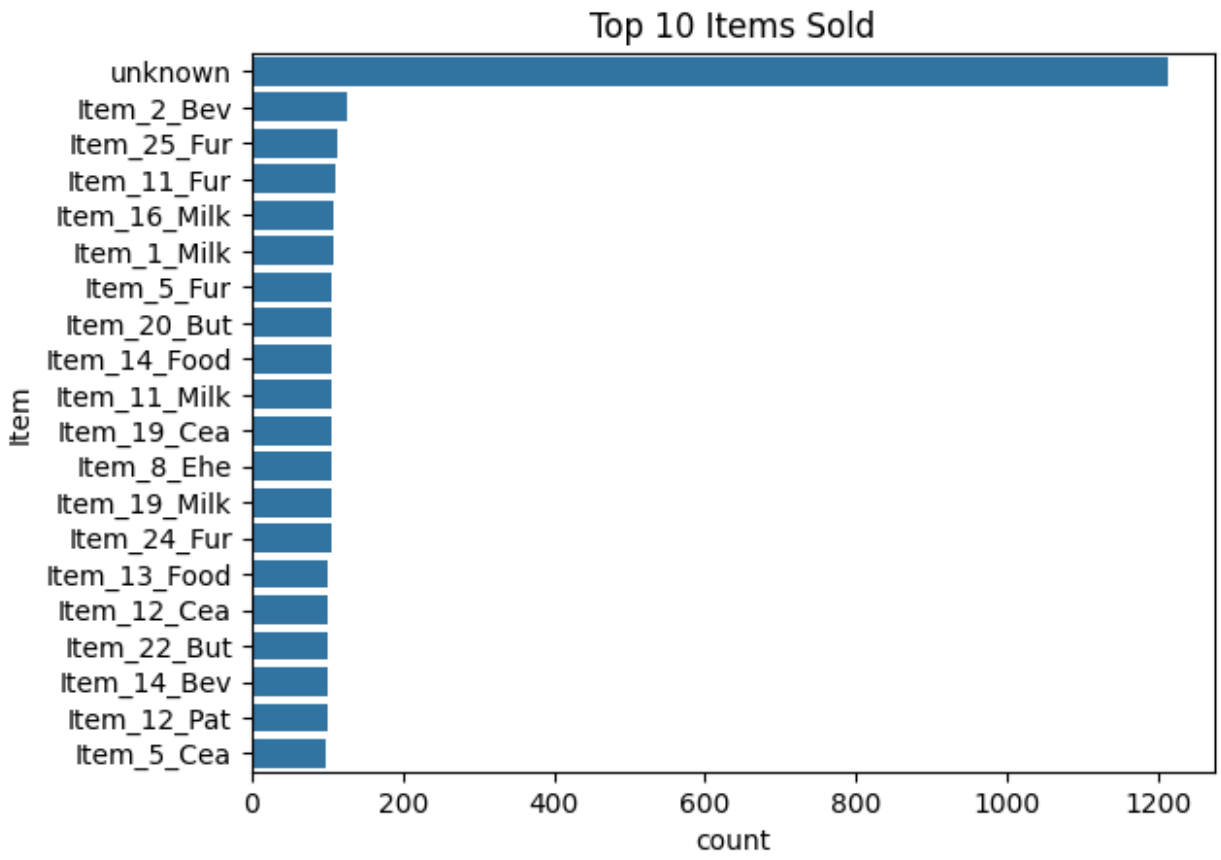



Quantity Distribution
Count
Quantity
0
250
500
750
1000
1250
1500
1750
2
4
6
8
10



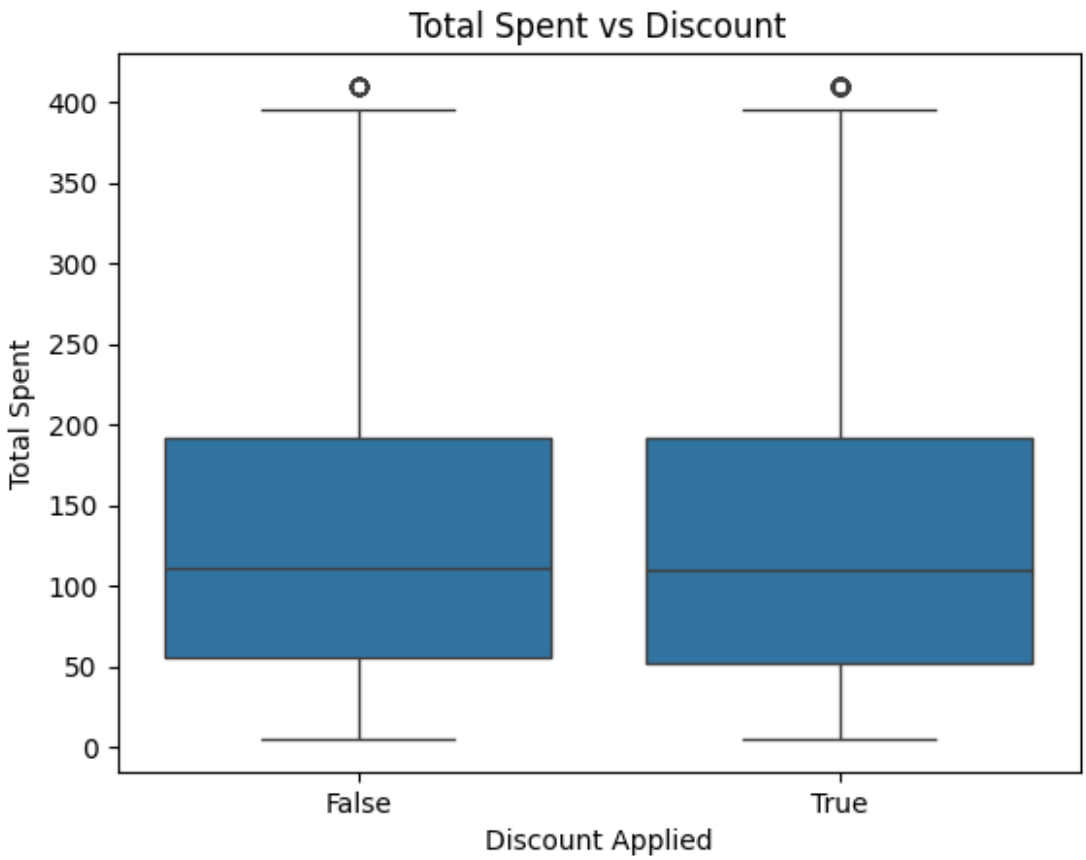

```
# Categorical distributions
sns.countplot(x='Discount Applied', data=data)
plt.title("Discount Applied")
plt.show()

sns.countplot(y='Item', data=data,
order=data['Item'].value_counts().head(20).index)
plt.title("Top 10 Items Sold")
plt.show()
```

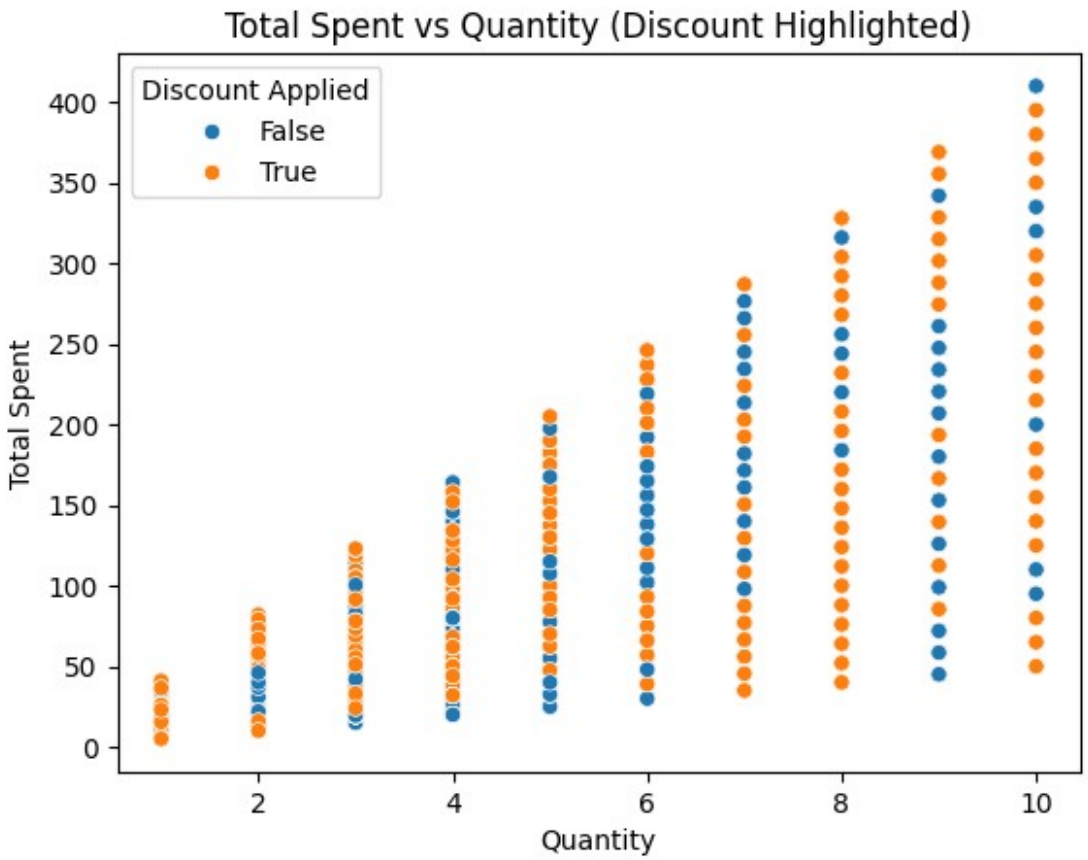

Discount Applied
8000
7000
6000
5000
4000
3000
2000
1000
0
count
False
True
Discount Applied



Bivariate Analysis(Compare two Columns): check relationship between variables

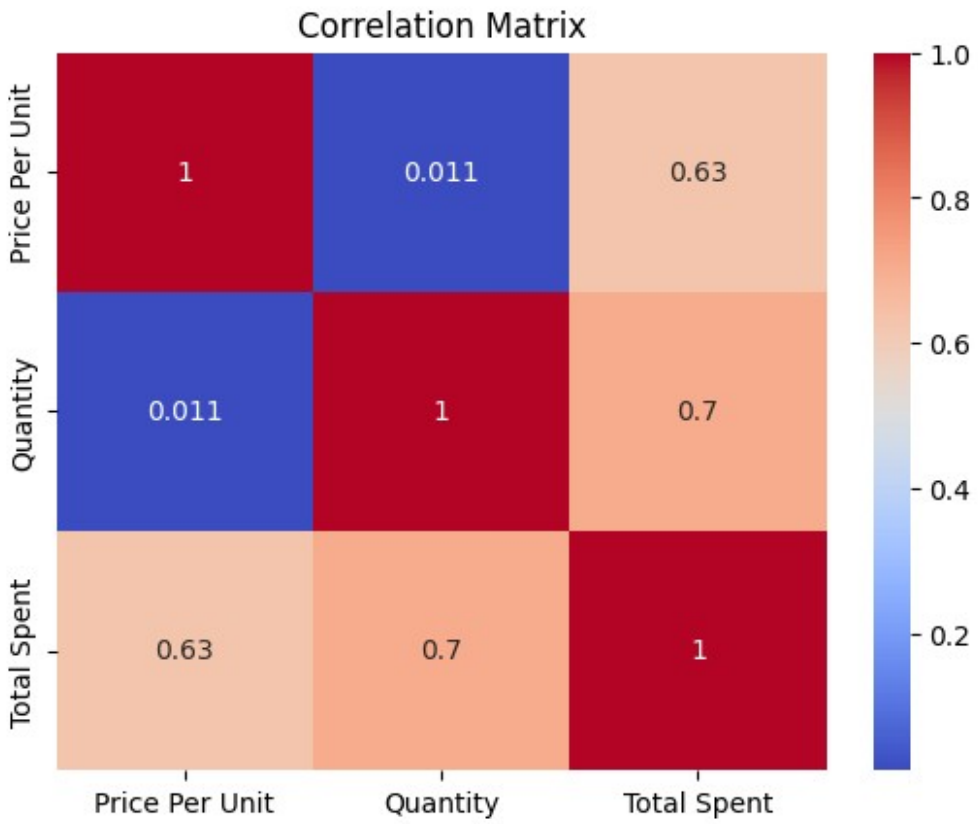
```
sns.boxplot(x='Discount Applied', y='Total Spent',data=data)
plt.title("Total Spent vs Discount")
plt.show()

sns.scatterplot(x='Quantity', y='Total Spent', hue='Discount Applied',
data=data)
plt.title("Total Spent vs Quantity (Discount Highlighted)")
plt.show()
```


Total Spent vs Discount
Total Spent
400
350
300
250
200
150
100
50
0
False
True
Discount Applied



Correlation Analysis

```
numeric_data = data.select_dtypes(include=['float64', 'int64'])
sns.heatmap(numeric_data.corr(), annot=True, cmap='coolwarm')
plt.title("Correlation Matrix")
plt.show()
```


Correlation Matrix
Price Per Unit
Quantity
Total Spent
1
0.011
0.63
0.011
1
0.7
0.63
0.7
1
Price Per Unit
Quantity
Total Spent
1.0
0.8
0.6
0.4
0.2