```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')

df = pd.read_csv('retail_sales_dataset.csv')
df
```

```
     Transaction ID        Date Customer ID  Gender  Age Product
Category  \
0                 1  2023-11-24     CUST001    Male   34
Beauty
1                 2  2023-02-27     CUST002  Female   26
Clothing
2                 3  2023-01-13     CUST003    Male   50
Electronics
3                 4  2023-05-21     CUST004    Male   37
Clothing
4                 5  2023-05-06     CUST005    Male   30
Beauty
..              ...         ...         ...     ...  ...
...
995             996  2023-05-16     CUST996    Male   62
Clothing
996             997  2023-11-17     CUST997    Male   52
Beauty
997             998  2023-10-29     CUST998  Female   23
Beauty
998             999  2023-12-05     CUST999  Female   36
Electronics
999            1000  2023-04-12    CUST1000    Male   47
Electronics

     Quantity  Price per Unit  Total Amount
0           3              50           150
1           2             500          1000
2           1              30            30
3           1             500           500
4           2              50           100
..        ...             ...           ...
995         1              50            50
996         3              30            90
997         4              25           100
998         3              50           150
999         4              30           120

[1000 rows x 9 columns]
```

```
df1 = pd.read_csv('RS_Session_265_AU_845_B_i.csv')
df1
```

	Sl. No.	State/UT	Total Fair Price Shops	Operational with ePOS devices	% FPS Automation
0	1	Andaman and Nicobar Islands	416.0	416.0	100.0
1	2	Andhra Pradesh	29791.0	29791.0	100.0
2	3	Arunachal Pradesh	1680.0	1680.0	100.0
3	4	Assam			
4	5	Bihar			
5	6	Chandigarh			
6	7	Chhattisgarh			
7	8	Dadra and Nagar Haveli and Daman and Diu			
8	9	Delhi			
9	10	Goa			
10	11	Gujarat			
11	12	Haryana			
12	13	Himachal Pradesh			
13	14	Jammu and Kashmir			
14	15	Jharkhand			
15	16	Karnataka			
16	17	Kerala			
17	18	Ladakh			
18	19	Lakshadweep			
19	20	Madhya Pradesh			
20	21	Maharashtra			
21	22	Manipur			
22	23	Meghalaya			
23	24	Mizoram			
24	25	Nagaland			
25	26	Odisha			
26	27	Puducherry			
27	28	Punjab			
28	29	Rajasthan			
29	30	Sikkim			
30	31	Tamil Nadu			
31	32	Telangana			
32	33	Tripura			
33	34	Uttarakhand			
34	35	Uttar Pradesh			
35	36	West Bengal			
36	National Summary	National Summary			

```
3                   34300.0                          34286.0
100.0
4                   50951.0                          50951.0
100.0
5                       NaN                              NaN
NaN
6                   13675.0                          13675.0
100.0
7                     114.0                            114.0
100.0
8                    1993.0                           1993.0
100.0
9                     452.0                            452.0
100.0
10                  16949.0                          16949.0
100.0
11                   9434.0                           9434.0
100.0
12                   5219.0                           5155.0
99.0
13                   6737.0                           6737.0
100.0
14                  25228.0                          25228.0
100.0
15                  20403.0                          20325.0
100.0
16                  13913.0                          13905.0
100.0
17                    404.0                            404.0
100.0
18                     39.0                             39.0
100.0
19                  27377.0                          27127.0
99.0
20                  52642.0                          52642.0
100.0
21                   2339.0                           2339.0
100.0
22                   4735.0                           4727.0
100.0
23                   1258.0                           1258.0
100.0
24                   1783.0                           1774.0
99.0
25                  12044.0                          12044.0
100.0
26                      NaN                              NaN
NaN
27                  18150.0                          18150.0
```

```
100.0
28                  27062.0                          25579.0
95.0
29                   1312.0                           1312.0
100.0
30                  34805.0                          34805.0
100.0
31                  17246.0                          17246.0
100.0
32                   2057.0                           2057.0
100.0
33                   9059.0                           9059.0
100.0
34                  79216.0                          79216.0
100.0
35                  20476.0                          20476.0
100.0
36                 543259.0                         541345.0
99.6
```

```
df.shape
```

```
(1000, 9)
```

```
df.head()
```

```
   Transaction ID        Date Customer ID  Gender  Age Product
Category  \
0               1  2023-11-24     CUST001    Male   34
Beauty
1               2  2023-02-27     CUST002  Female   26
Clothing
2               3  2023-01-13     CUST003    Male   50
Electronics
3               4  2023-05-21     CUST004    Male   37
Clothing
4               5  2023-05-06     CUST005    Male   30
Beauty

   Quantity  Price per Unit  Total Amount
0         3              50           150
1         2             500          1000
2         1              30            30
3         1             500           500
4         2              50           100
```

```
df.tail()
```

```
     Transaction ID        Date Customer ID  Gender  Age Product
Category  \
995             996  2023-05-16     CUST996    Male   62
```

```
Clothing
996             997  2023-11-17     CUST997    Male   52
Beauty
997             998  2023-10-29     CUST998  Female   23
Beauty
998             999  2023-12-05     CUST999  Female   36
Electronics
999            1000  2023-04-12    CUST1000    Male   47
Electronics

     Quantity  Price per Unit  Total Amount
995         1              50            50
996         3              30            90
997         4              25           100
998         3              50           150
999         4              30           120

df.columns

Index(['Transaction ID', 'Date', 'Customer ID', 'Gender', 'Age',
       'Product Category', 'Quantity', 'Price per Unit', 'Total
Amount'],
      dtype='object')

df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 9 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Transaction ID    1000 non-null   int64
 1   Date              1000 non-null   object
 2   Customer ID       1000 non-null   object
 3   Gender            1000 non-null   object
 4   Age               1000 non-null   int64
 5   Product Category  1000 non-null   object
 6   Quantity          1000 non-null   int64
 7   Price per Unit    1000 non-null   int64
 8   Total Amount      1000 non-null   int64
dtypes: int64(5), object(4)
memory usage: 70.4+ KB

df.describe()

       Transaction ID         Age     Quantity  Price per Unit  Total
Amount
count     1000.000000  1000.00000  1000.000000     1000.000000
1000.000000
mean       500.500000    41.39200     2.514000      179.890000
456.000000
```

```
std         288.819436    13.68143     1.132734      189.681356
559.997632
min           1.000000    18.00000     1.000000       25.000000
25.000000
25%         250.750000    29.00000     1.000000       30.000000
60.000000
50%         500.500000    42.00000     3.000000       50.000000
135.000000
75%         750.250000    53.00000     4.000000      300.000000
900.000000
max        1000.000000    64.00000     4.000000      500.000000
2000.000000
```

```
df.describe(include='object')
```

```
              Date Customer ID  Gender Product Category
count         1000        1000    1000             1000
unique         345        1000       2                3
top     2023-05-16     CUST001  Female         Clothing
freq            11           1     510              351
```

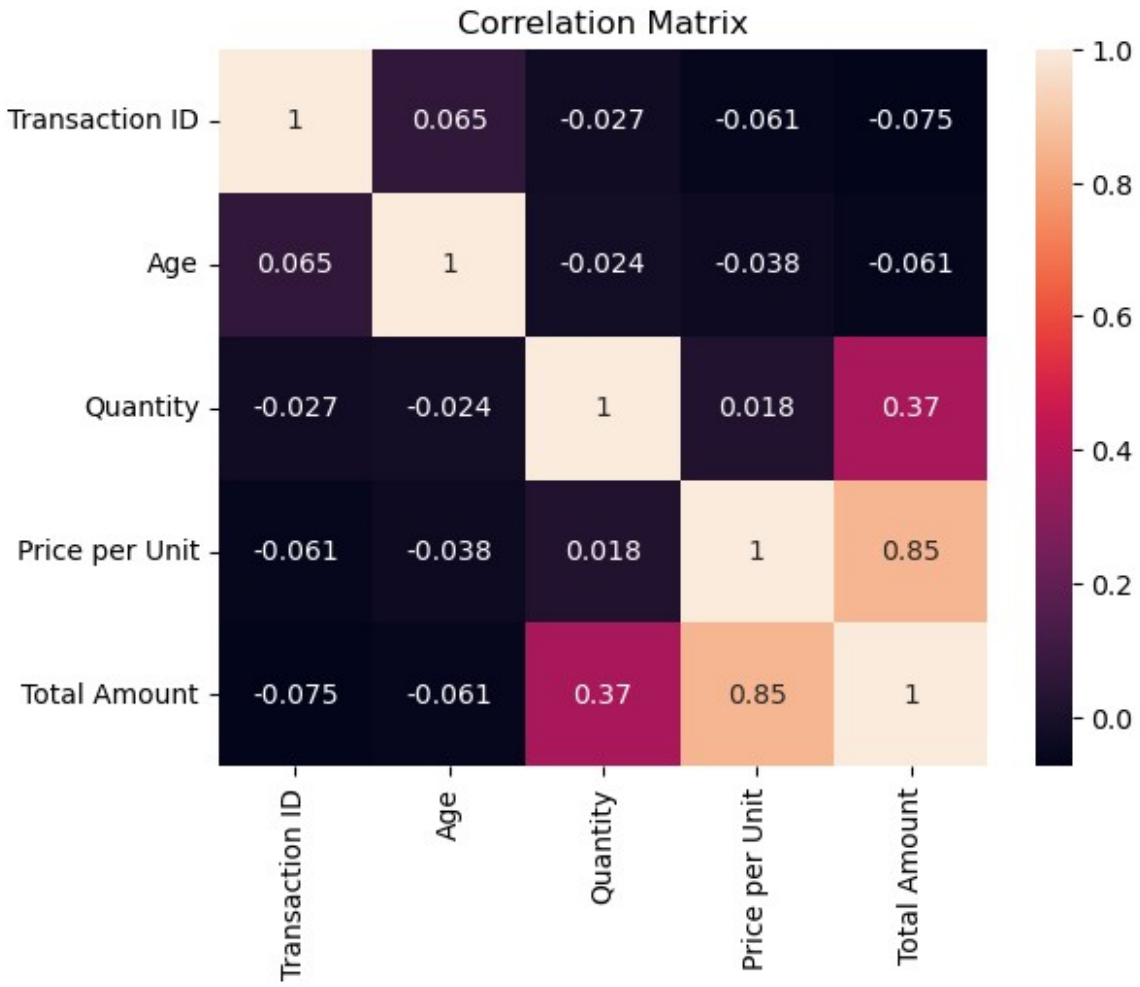
```
df.isnull().sum()
```

```
Transaction ID      0
Date                0
Customer ID         0
Gender              0
Age                 0
Product Category    0
Quantity            0
Price per Unit      0
Total Amount        0
dtype: int64
```

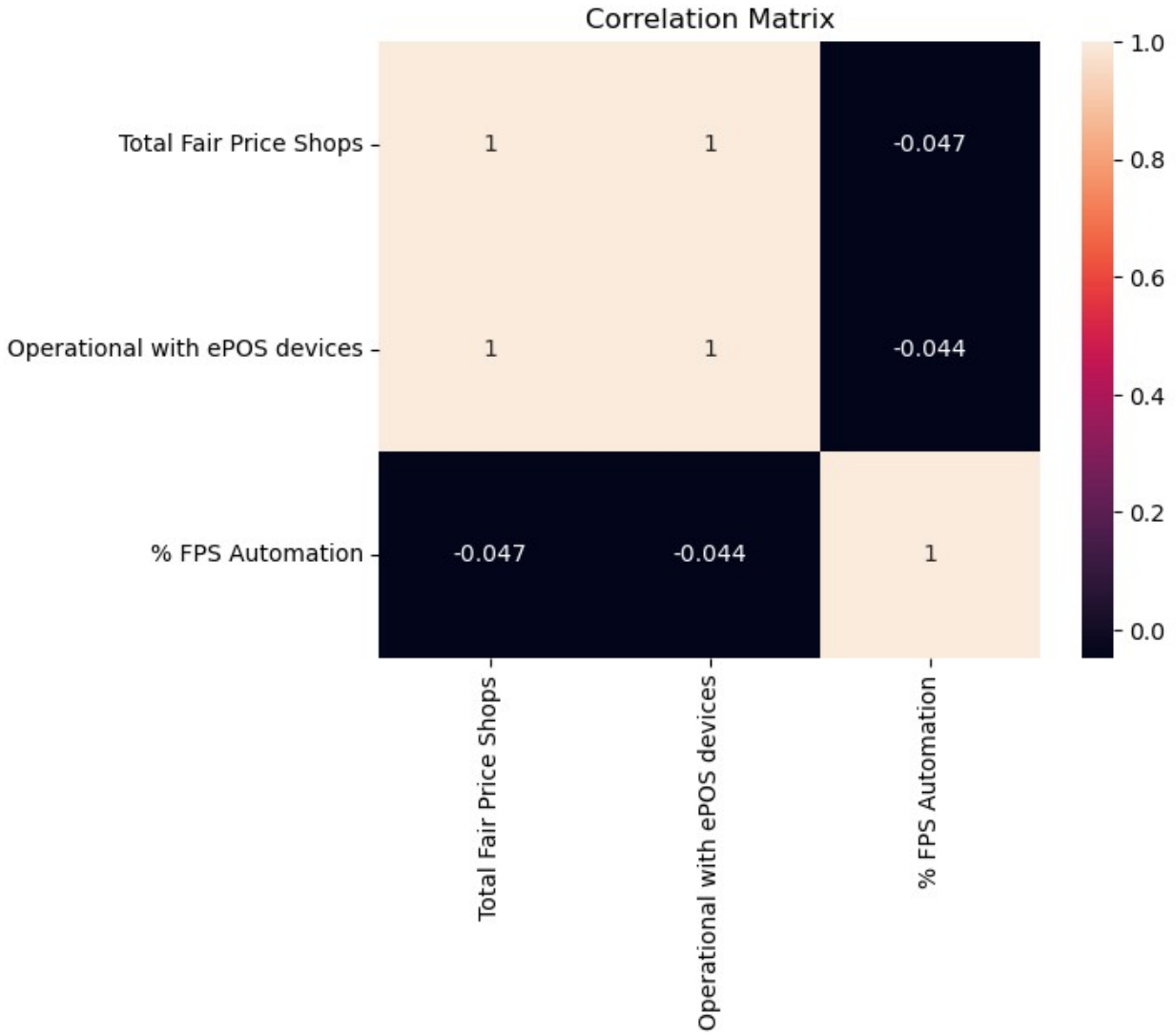
```
df1.isnull().sum()
```

```
Sl. No.                           0
State/UT                          0
Total Fair Price Shops            2
Operational with ePOS devices     2
% FPS Automation                  2
dtype: int64
```

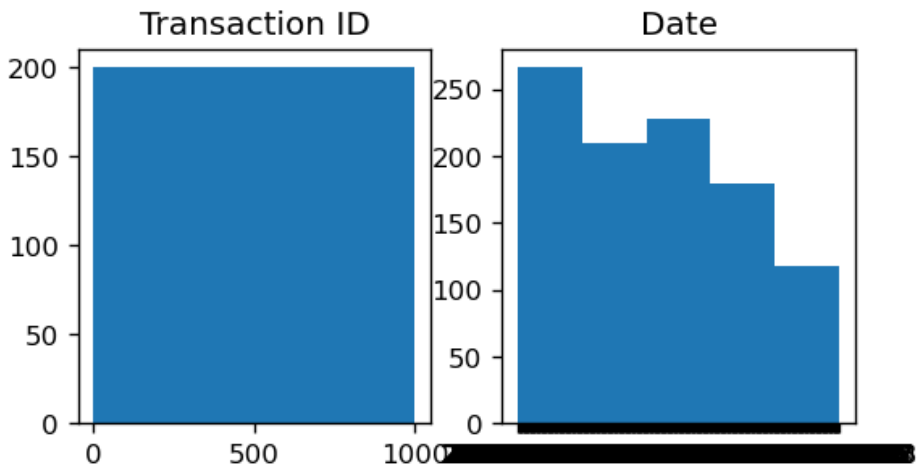
```
sns.heatmap(df.corr(numeric_only=True),annot=True)
sns.color_palette("viridis", as_cmap=True)
plt.title('Correlation Matrix')
plt.show()
```

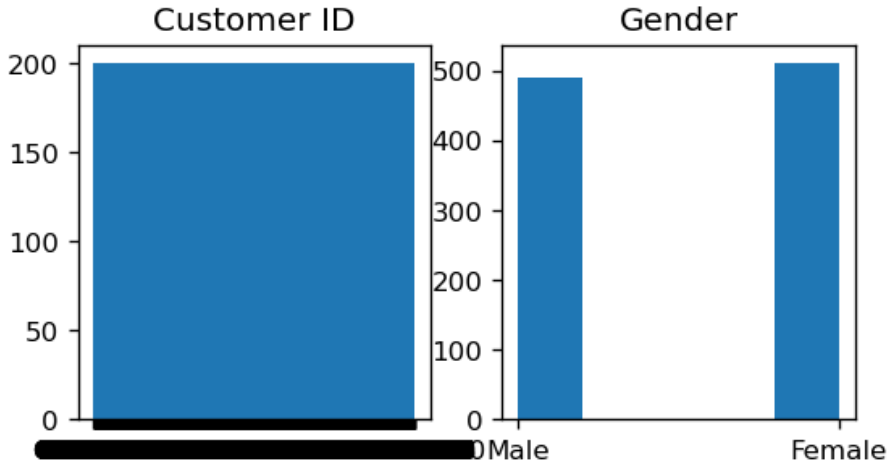


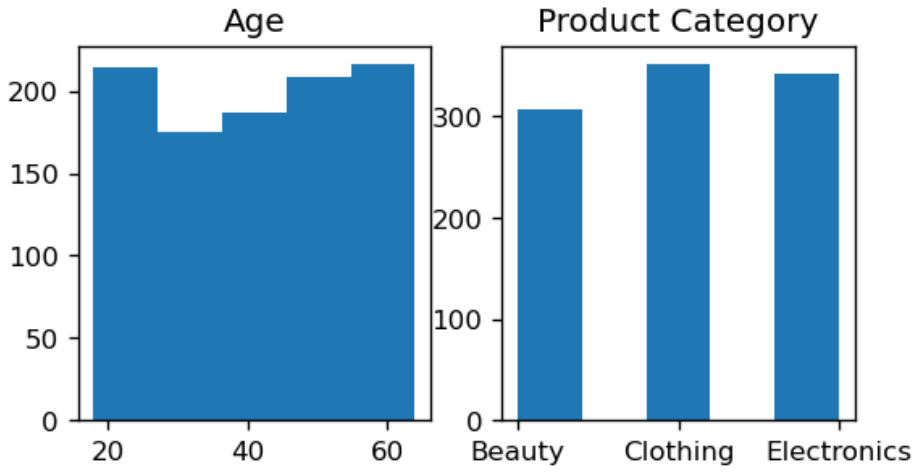

```
sns.heatmap(df1.corr(numeric_only=True),annot=True)
sns.color_palette("viridis", as_cmap=True)
plt.title('Correlation Matrix')
plt.show()
```

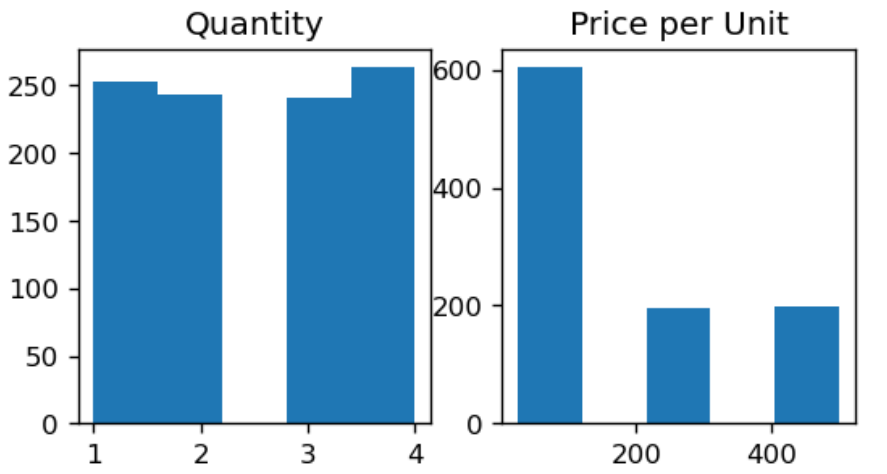


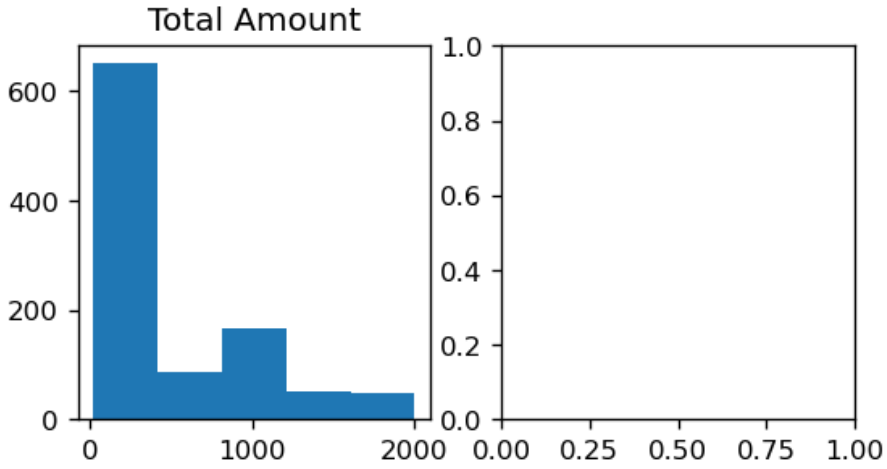

```
i = 0
while i < len(df.columns):
    try:
        fig = plt.figure(figsize=(8,2.5),dpi=120)
        plt.subplot(1,3,1)
        plt.hist(df[df.columns[i]],bins=5)
        plt.title(df.columns[i])
        i+=1
        plt.subplot(1,3,2)
        plt.hist(df[df.columns[i]],bins=5)
        plt.title(df.columns[i])
        i+=1
    except:
        continue

plt.show()
```


Transaction ID
Date
200
150
100
50
0
0
500
250
200
150
100
50
0

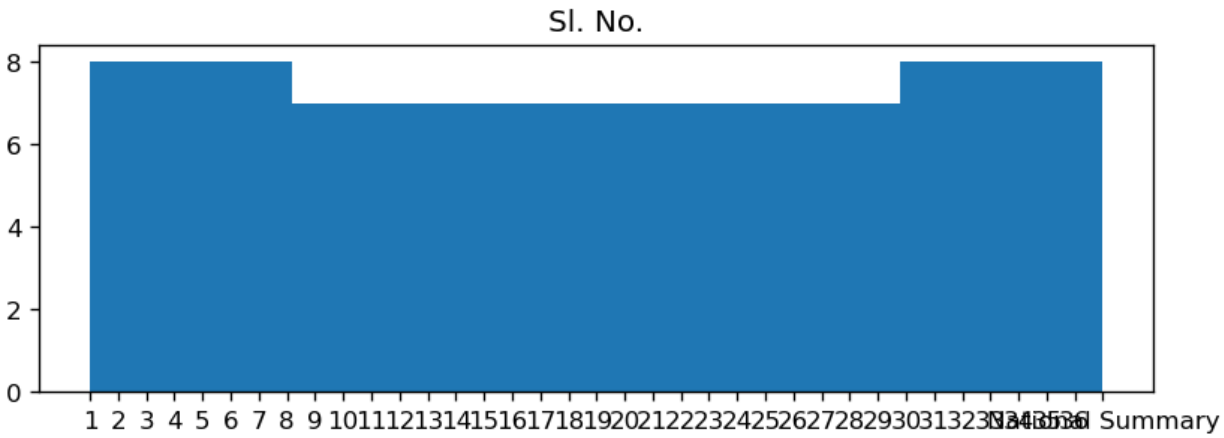

Customer ID
Gender
200
150
100
50
0
500
400
300
200
100
0
Male
Female

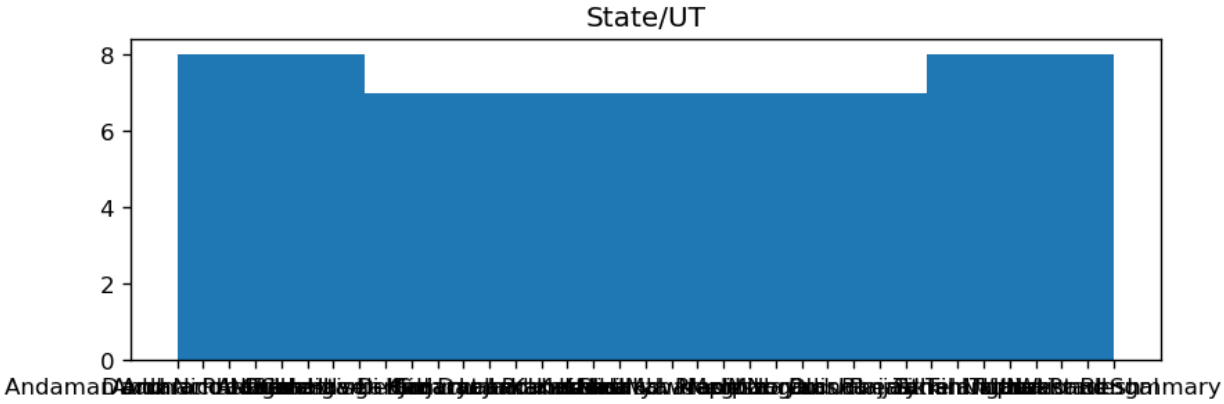

Age
Product Category
200
150
100
50
0
20
40
60
300
200
100
0
Beauty
Clothing
Electronics



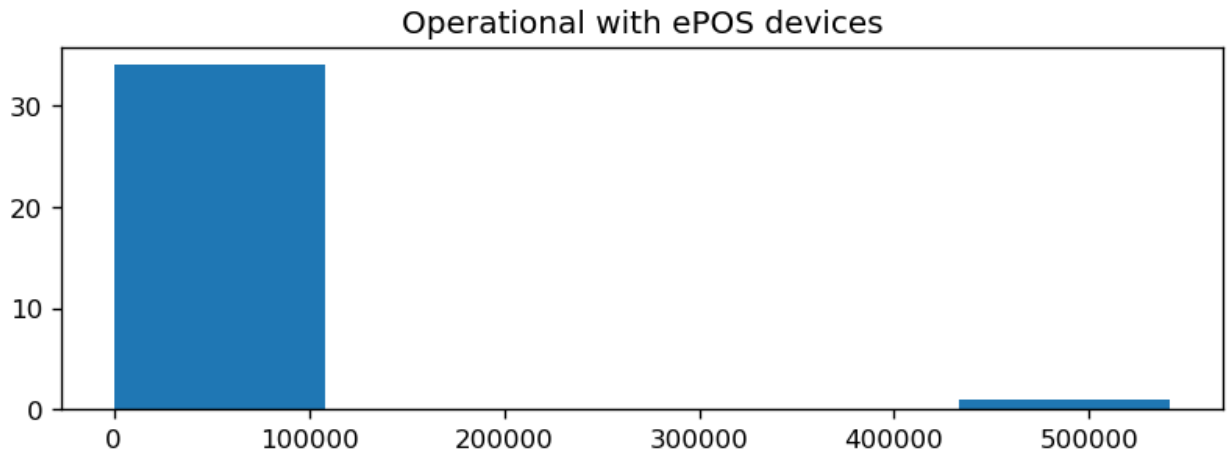


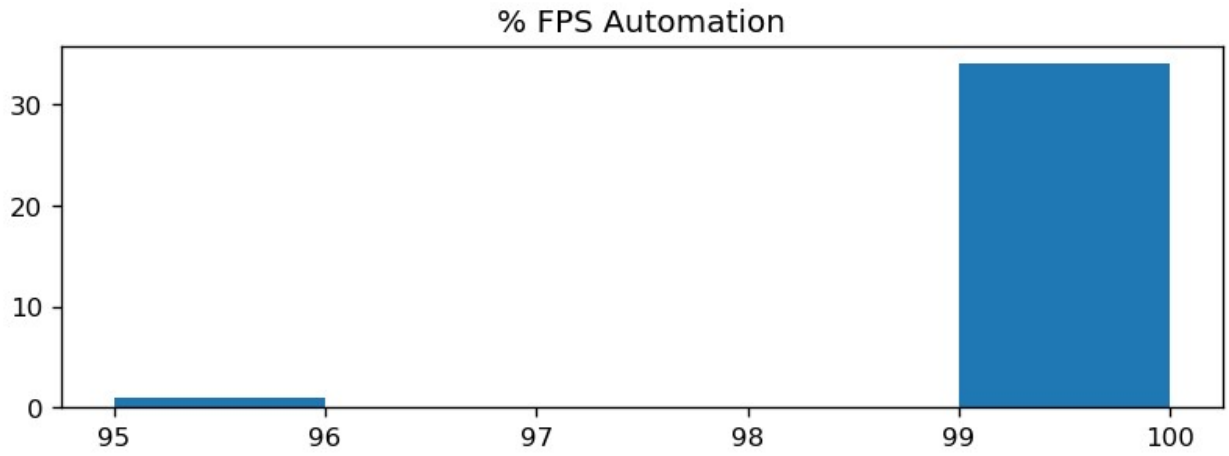

```
i = 0
while i < len(df1.columns):
    try:
        fig = plt.figure(figsize=(8,2.5),dpi=120)
        plt.subplot(1,1,1)
        plt.hist(df1[df1.columns[i]],bins=5)
        plt.title(df1.columns[i])
        i+=1
        plt.subplot(1,1,2)
        plt.hist(df1[df1.columns[i]],bins=5)
        plt.title(df1.columns[i])
        i+=1
    except:
        continue

plt.show()
```


Sl. No.
8
6
4
2
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 Summary


State/UT
8
6
4
2
0
Andaman Summary


Total Fair Price Shops
30
20
10
0
0
100000
200000
300000
400000
500000



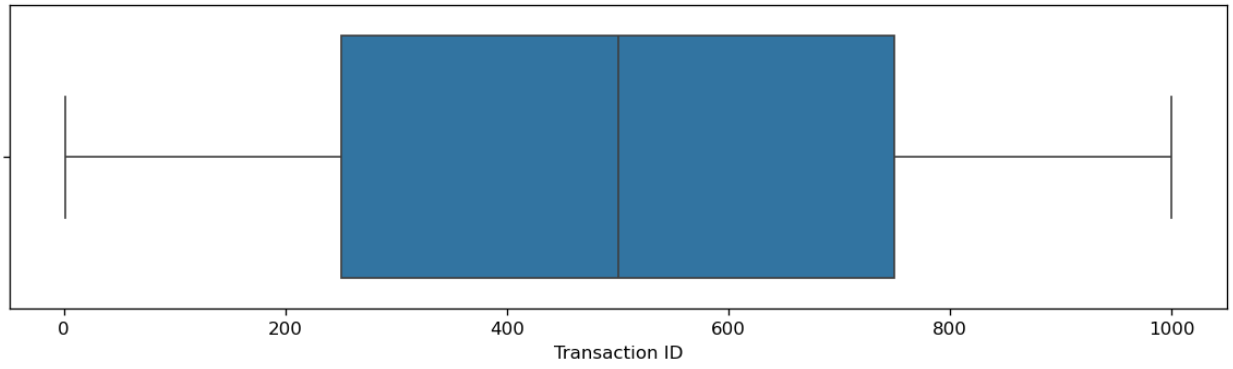


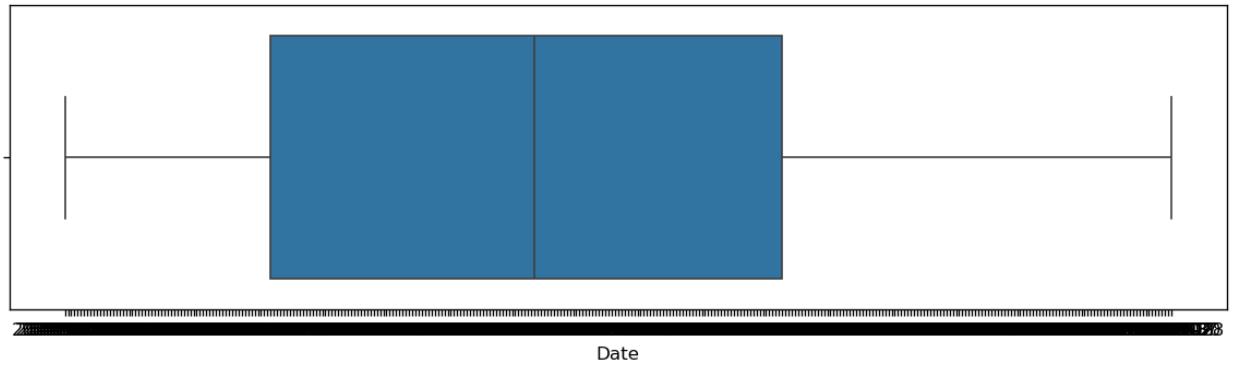

```
df.corr(numeric_only=True)
```

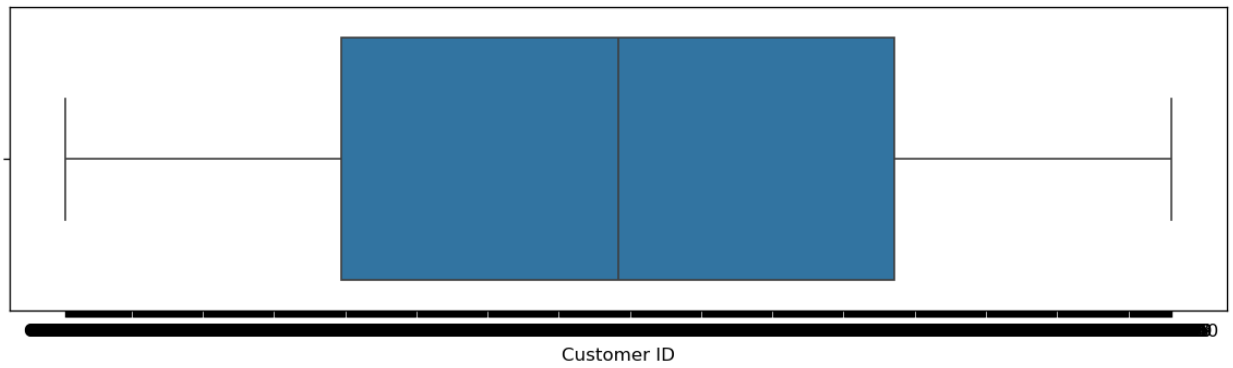
```
                Transaction ID       Age  Quantity  Price per Unit  \
Transaction ID        1.000000  0.065191 -0.026623       -0.060837   
Age                   0.065191  1.000000 -0.023737       -0.038423   
Quantity             -0.026623 -0.023737  1.000000        0.017501   
Price per Unit       -0.060837 -0.038423  0.017501        1.000000   
Total Amount         -0.075034 -0.060568  0.373707        0.851925   

                Total Amount  
Transaction ID     -0.075034  
Age                -0.060568  
Quantity            0.373707  
Price per Unit      0.851925  
Total Amount        1.000000  
```

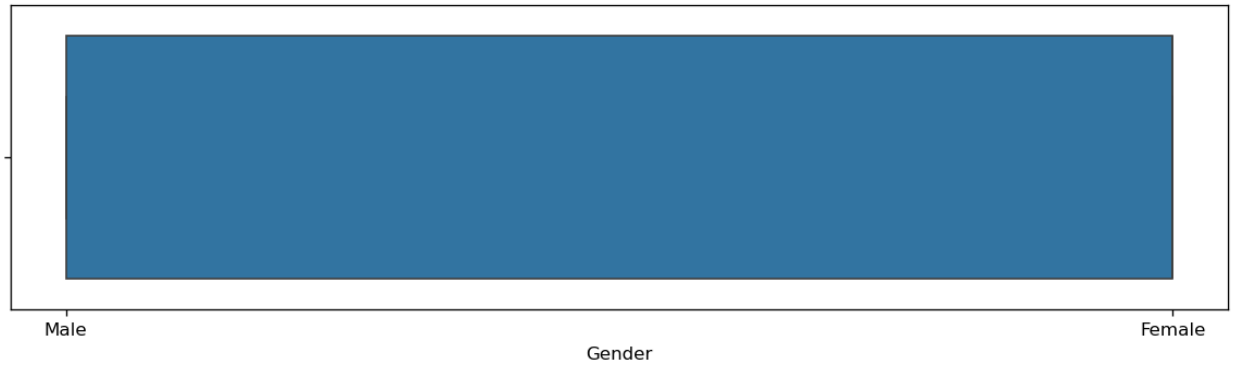
```
i = 0
while i < len(df.columns):
    try:
        fig = plt.figure(figsize=(12,3),dpi=120)
```

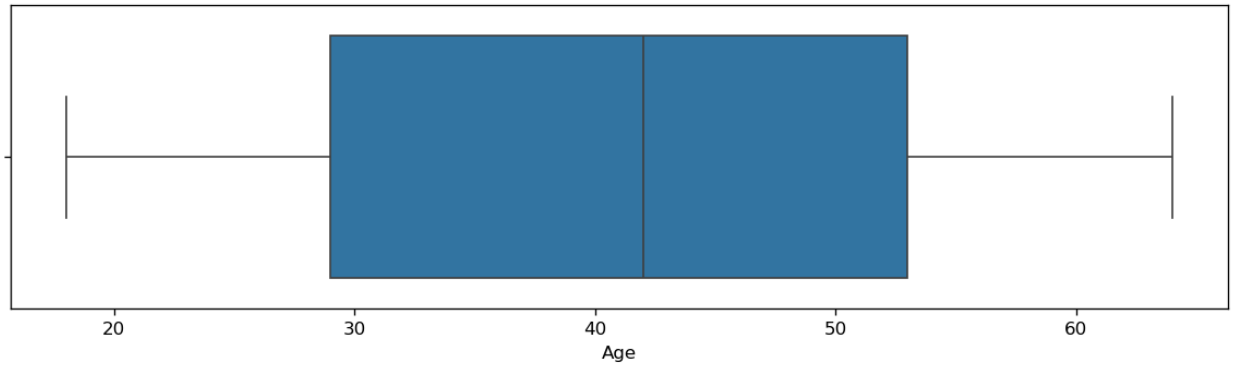
```
        sns.boxplot(x = df.columns[i], data=df)
        i += 1
    except:
        continue

plt.show()
```

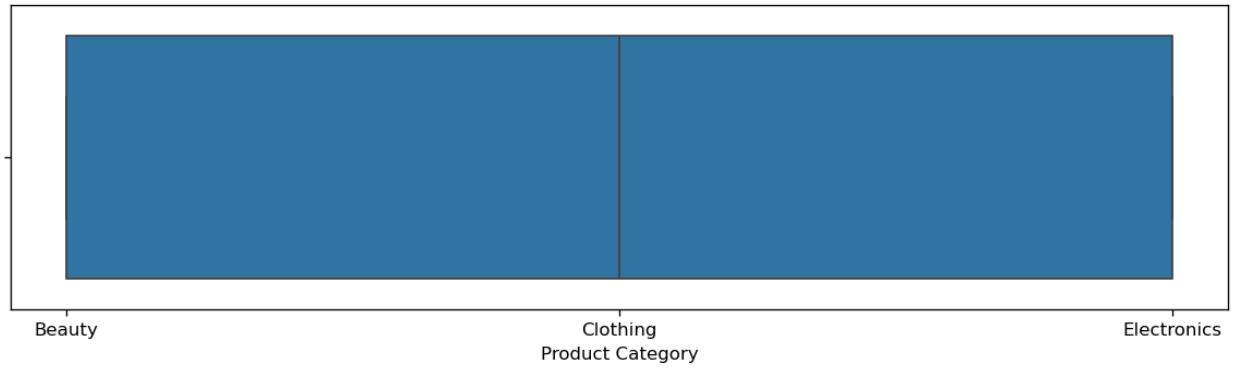


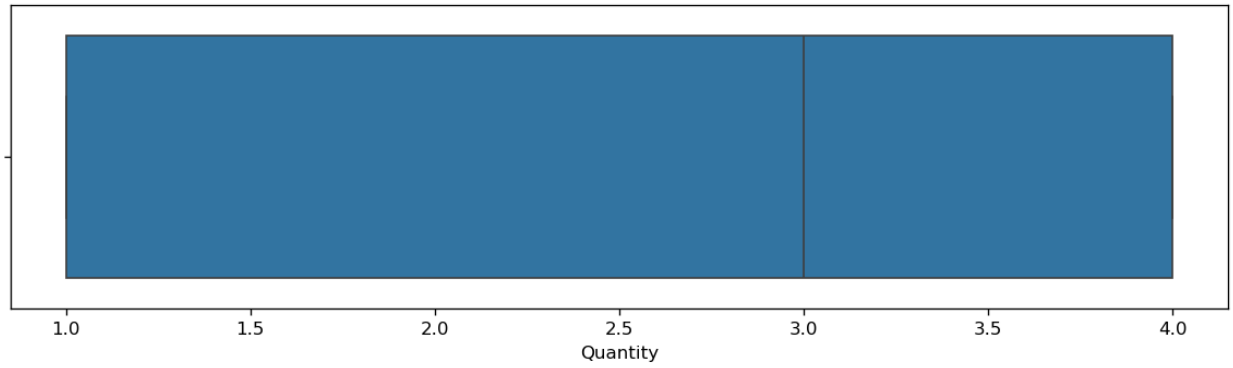


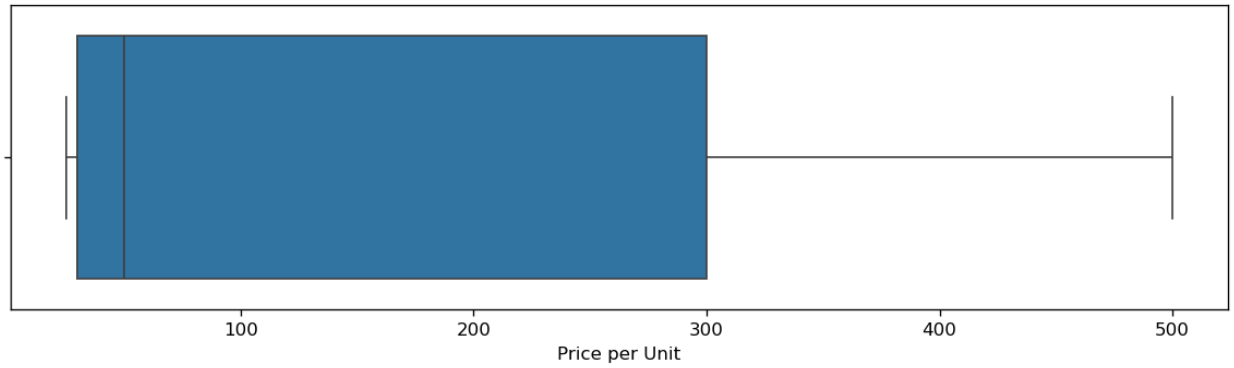


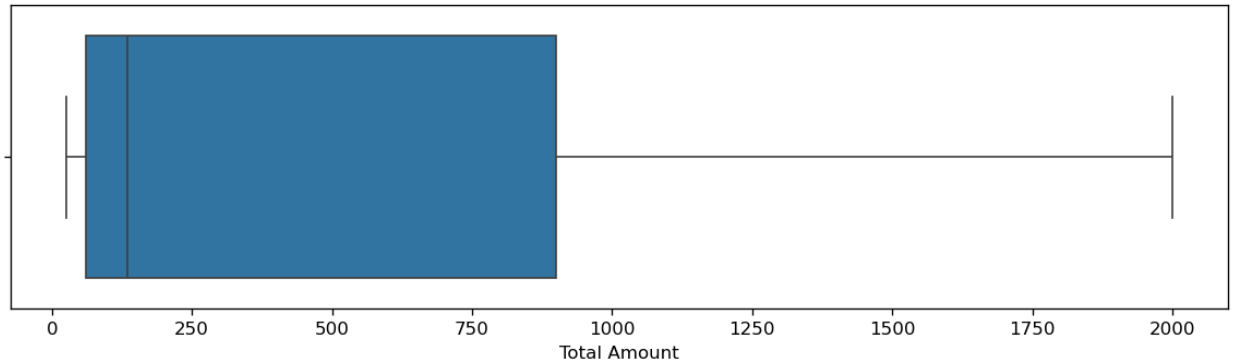

Male
Female
Gender


20
30
40
50
60
Age

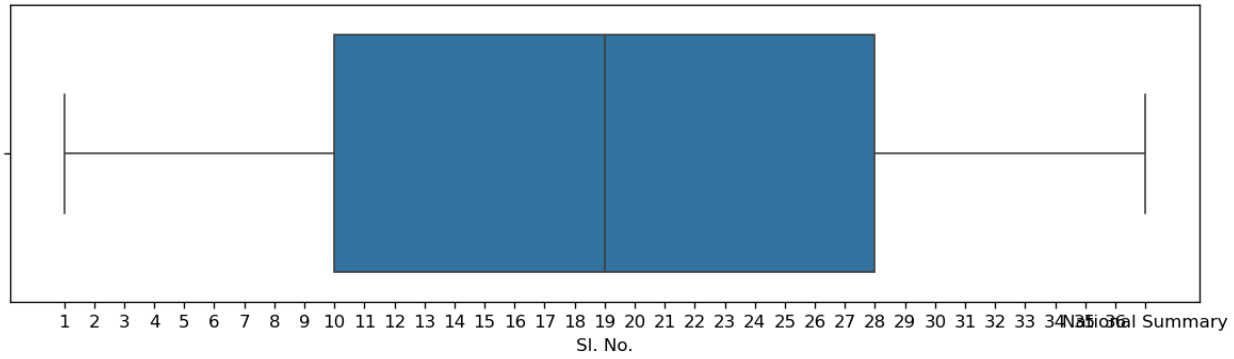

Beauty
Clothing
Electronics
Product Category

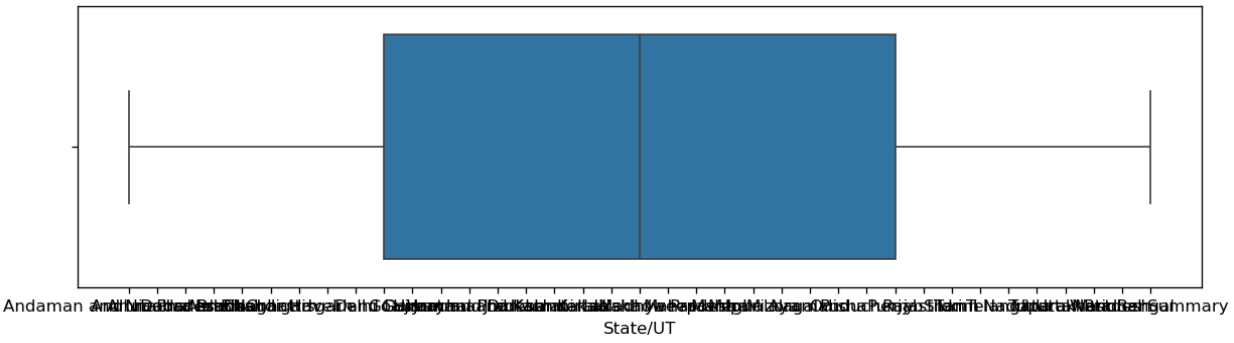

1.0
1.5
2.0
2.5
3.0
3.5
4.0
Quantity

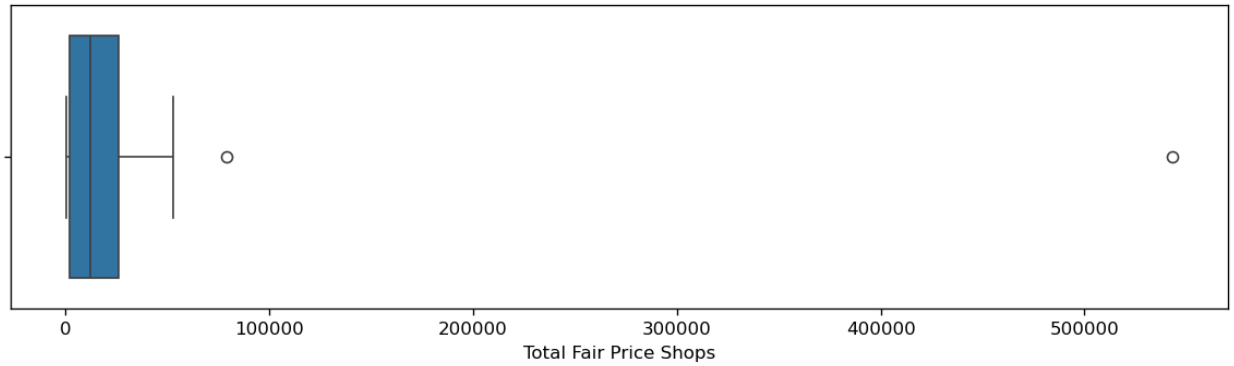


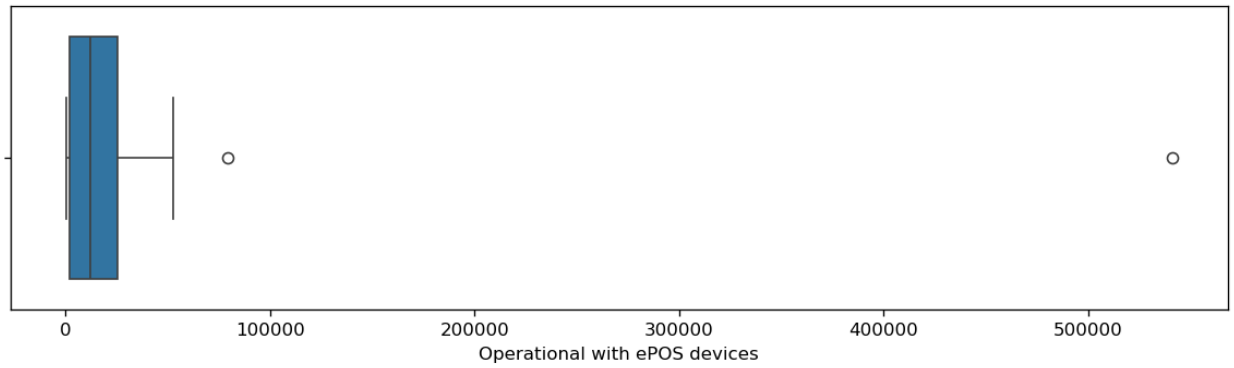


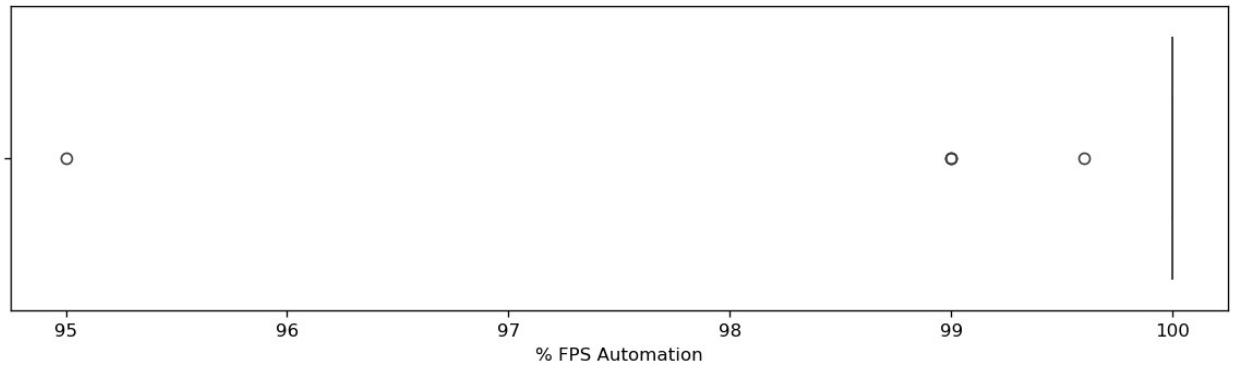

```
i = 0
while i < len(df1.columns):
    try:
        fig = plt.figure(figsize=(12,3),dpi=120)
        sns.boxplot(x = df1.columns[i], data=df1)
        i += 1
    except:
        continue

plt.show()
```



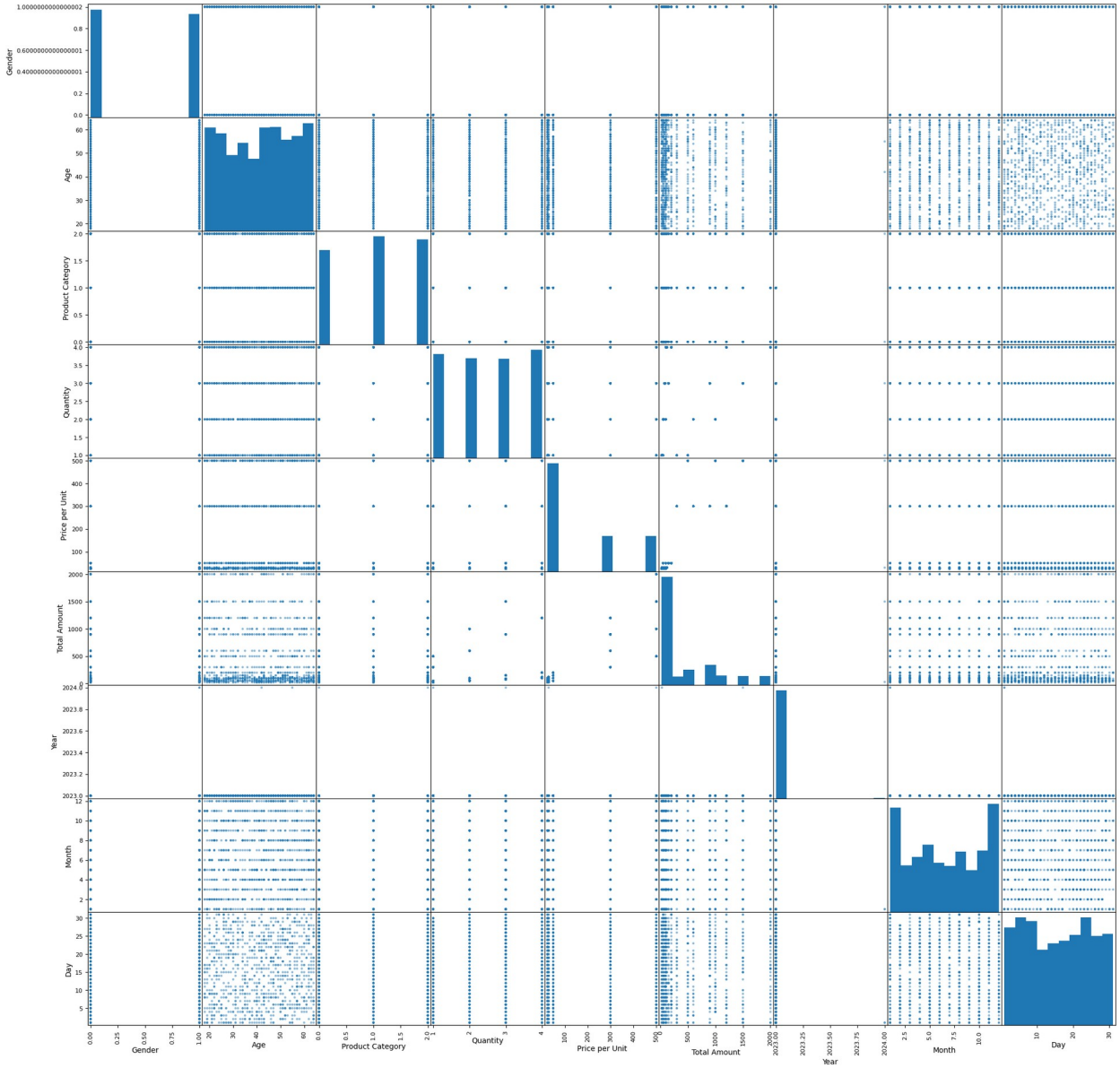




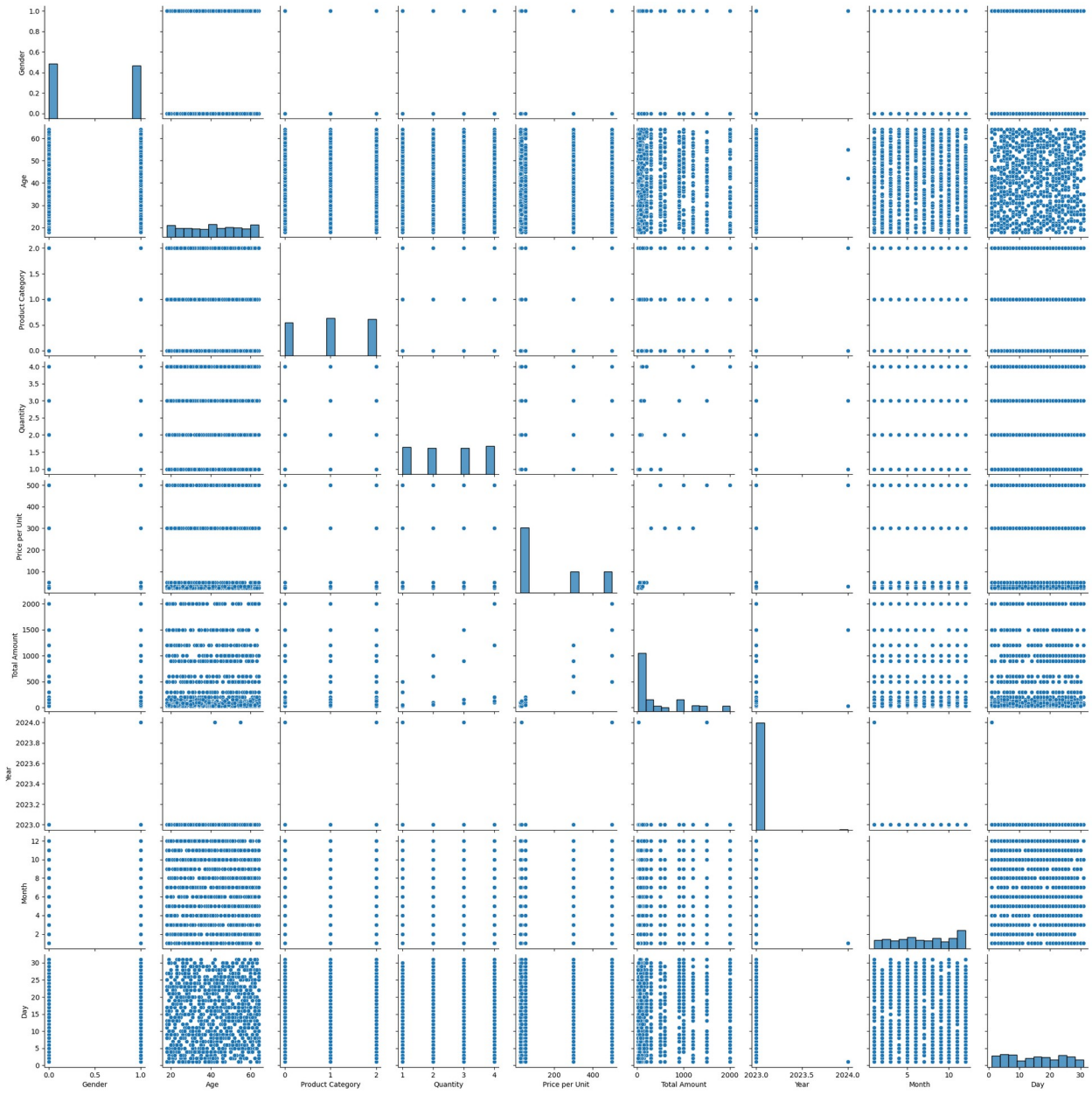





```
from pandas.plotting import scatter_matrix
p = scatter_matrix(df,figsize=(25,25))
```




```
p = sns.pairplot(df)
```




```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, OneHotEncoder,
StandardScaler

# Preprocessing steps
# Extract features from the 'Date' column
df['Date'] = pd.to_datetime(df['Date'])
df['Year'] = df['Date'].dt.year
df['Month'] = df['Date'].dt.month
df['Day'] = df['Date'].dt.day
df = df.drop(columns=['Date', 'Transaction ID', 'Customer ID'])
```

```
# Encode categorical features
label_encoders = {}
for col in ['Gender', 'Product Category']:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le

# Separate features and targets for regression and classification
X = df.drop(columns=['Total Amount', 'Price per Unit', 'Gender',
'Product Category'])
y_class_gender = df['Gender']
y_class_category = df['Product Category']
y_reg_amount = df['Total Amount']
y_reg_price = df['Price per Unit']

# Scale numerical features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split the data into train and test sets
X_train_class, X_test_class, y_train_class_gender, y_test_class_gender
= train_test_split(
    X_scaled, y_class_gender, test_size=0.2, random_state=42)

X_train_reg, X_test_reg, y_train_reg_amount, y_test_reg_amount =
train_test_split(
    X_scaled, y_reg_amount, test_size=0.2, random_state=42)

X_train_cat, X_test_cat, y_train_cat, y_test_cat = train_test_split(
    X_scaled, y_class_category, test_size=0.2, random_state=42)

# Preprocessing summary
df.head(), X.shape
```

```
(   Gender  Age  Product Category  Quantity  Price per Unit  Total
Amount  \
 0       1   34                 0         3              50
150
 1       0   26                 1         2             500
1000
 2       1   50                 2         1              30
30
 3       1   37                 1         1             500
500
 4       1   30                 0         2              50
100

    Year  Month  Day
 0  2023     11   24
 1  2023      2   27
```

```
 2  2023      1   13
 3  2023      5   21
 4  2023      5    6  ,
 (1000, 5))
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report

# Logistic Regression
log_reg = LogisticRegression(random_state=42)
log_reg.fit(X_train_class, y_train_class_gender)
y_pred_log_reg = log_reg.predict(X_test_class)
log_reg_acc = accuracy_score(y_test_class_gender, y_pred_log_reg)

# Random Forest Classifier
rf_clf = RandomForestClassifier(random_state=42)
rf_clf.fit(X_train_class, y_train_class_gender)
y_pred_rf_clf = rf_clf.predict(X_test_class)
rf_clf_acc = accuracy_score(y_test_class_gender, y_pred_rf_clf)

# Support Vector Machine
svc_clf = SVC(random_state=42)
svc_clf.fit(X_train_class, y_train_class_gender)
y_pred_svc_clf = svc_clf.predict(X_test_class)
svc_clf_acc = accuracy_score(y_test_class_gender, y_pred_svc_clf)

# Results
{
    "Logistic Regression Accuracy": log_reg_acc,
    "Random Forest Classifier Accuracy": rf_clf_acc,
    "SVC Accuracy": svc_clf_acc
}
```

```
{'Logistic Regression Accuracy': 0.555,
 'Random Forest Classifier Accuracy': 0.465,
 'SVC Accuracy': 0.515}
```

```
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error, r2_score

# Linear Regression
lin_reg = LinearRegression()
lin_reg.fit(X_train_reg, y_train_reg_amount)
y_pred_lin_reg = lin_reg.predict(X_test_reg)
lin_reg_rmse = mean_squared_error(y_test_reg_amount, y_pred_lin_reg,
squared=False)
```

```
lin_reg_r2 = r2_score(y_test_reg_amount, y_pred_lin_reg)

# Random Forest Regressor
rf_reg = RandomForestRegressor(random_state=42)
rf_reg.fit(X_train_reg, y_train_reg_amount)
y_pred_rf_reg = rf_reg.predict(X_test_reg)
rf_reg_rmse = mean_squared_error(y_test_reg_amount, y_pred_rf_reg,
squared=False)
rf_reg_r2 = r2_score(y_test_reg_amount, y_pred_rf_reg)

# Support Vector Regressor
svr_reg = SVR()
svr_reg.fit(X_train_reg, y_train_reg_amount)
y_pred_svr_reg = svr_reg.predict(X_test_reg)
svr_reg_rmse = mean_squared_error(y_test_reg_amount, y_pred_svr_reg,
squared=False)
svr_reg_r2 = r2_score(y_test_reg_amount, y_pred_svr_reg)

# Results
{
    "Linear Regression RMSE": lin_reg_rmse,
    "Linear Regression R2": lin_reg_r2,
    "Random Forest Regressor RMSE": rf_reg_rmse,
    "Random Forest Regressor R2": rf_reg_r2,
    "SVR RMSE": svr_reg_rmse,
    "SVR R2": svr_reg_r2
}
```

```
{'Linear Regression RMSE': 634.6016014411379,
 'Linear Regression R2': -0.37573979551204406,
 'Random Forest Regressor RMSE': 546.684262105346,
 'Random Forest Regressor R2': -0.020956212074570324,
 'SVR RMSE': 641.455065502614,
 'SVR R2': -0.4056152192167819}
```