Sales Data Analysis - Kavan Prajapati

Importing Libraries

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

Loading Data

```
# Load the dataset
sales_df = pd.read_csv("sales.csv")
```

Display basic information

```
# Display basic information
print(sales_df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 17 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   Invoice ID               1000 non-null   object 
 1   Branch                   1000 non-null   object 
 2   City                     1000 non-null   object 
 3   Customer type            1000 non-null   object 
 4   Gender                   1000 non-null   object 
 5   Product line             1000 non-null   object 
 6   Unit price               1000 non-null   float64
 7   Quantity                 1000 non-null   int64  
 8   Tax 5%                   1000 non-null   float64
 9   Total                    1000 non-null   float64
 10  Date                     1000 non-null   object 
 11  Time                     1000 non-null   object 
 12  Payment                  1000 non-null   object 
 13  cogs                     1000 non-null   float64
 14  gross margin percentage  1000 non-null   float64
 15  gross income             1000 non-null   float64
 16  Rating                   1000 non-null   float64
dtypes: float64(7), int64(1), object(9)
memory usage: 132.9+ KB
None
```

Printing rows of the Data

```
#displaying first five rows
display(sales_df.head())
```

```
    Invoice ID Branch       City Customer type  Gender  \
0  750-67-8428      A     Yangon        Member  Female
1  226-31-3081      C  Naypyitaw        Normal  Female
2  631-41-3108      A     Yangon        Normal    Male
3  123-19-1176      A     Yangon        Member    Male
4  373-73-7910      A     Yangon        Normal    Male

             Product line  Unit price  Quantity   Tax 5%     Total  \
0       Health and beauty       74.69         7  26.1415  548.9715
1  Electronic accessories       15.28         5   3.8200   80.2200
2      Home and lifestyle       46.33         7  16.2155  340.5255
3       Health and beauty       58.22         8  23.2880  489.0480
4       Sports and travel       86.31         7  30.2085  634.3785

         Date   Time      Payment    cogs  gross margin percentage  \
0  01-05-2019  13:08      Ewallet  522.83                 4.761905
1  03-08-2019  10:29         Cash   76.40                 4.761905
2  03-03-2019  13:23  Credit card  324.31                 4.761905
3   1/27/2019  20:33      Ewallet  465.76                 4.761905
4  02-08-2019  10:37      Ewallet  604.17                 4.761905

   gross income  Rating
0       26.1415     9.1
1        3.8200     9.6
2       16.2155     7.4
3       23.2880     8.4
4       30.2085     5.3

#displaying last five rows
display(sales_df.tail())

      Invoice ID Branch       City Customer type  Gender
Product line  \
995  233-67-5758      C  Naypyitaw        Normal    Male    Health and
beauty
996  303-96-2227      B   Mandalay        Normal  Female   Home and
lifestyle
997  727-02-1313      A     Yangon        Member    Male   Food and
beverages
998  347-56-2442      A     Yangon        Normal    Male   Home and
lifestyle
999  849-09-3807      A     Yangon        Member  Female  Fashion
accessories

     Unit price  Quantity   Tax 5%      Total        Date   Time
Payment  \
995       40.35         1   2.0175    42.3675   1/29/2019  13:46
Ewallet
996       97.38        10  48.6900  1022.4900  03-02-2019  17:16
Ewallet
```

```
997       31.84          1   1.5920    33.4320  02-09-2019  13:22
Cash
998       65.82          1   3.2910    69.1110   2/22/2019  15:33
Cash
999       88.34          7  30.9190   649.2990   2/18/2019  13:28
Cash

       cogs  gross margin percentage  gross income  Rating
995   40.35                 4.761905        2.0175     6.2
996  973.80                 4.761905       48.6900     4.4
997   31.84                 4.761905        1.5920     7.7
998   65.82                 4.761905        3.2910     4.1
999  618.38                 4.761905       30.9190     6.6
```

Printing the column names of the DataFrame

```
# print all the column name of the dataframe
print(list(sales_df.columns))

['Invoice ID', 'Branch', 'City', 'Customer type', 'Gender', 'Product
line', 'Unit price', 'Quantity', 'Tax 5%', 'Total', 'Date', 'Time',
'Payment', 'cogs', 'gross margin percentage', 'gross income',
'Rating']
```

Missing Data Handing

```
# Find missing values in the dataset
sales_df.isnull( )

     Invoice ID  Branch   City  Customer type  Gender  Product line  \
0         False   False  False          False   False         False
1         False   False  False          False   False         False
2         False   False  False          False   False         False
3         False   False  False          False   False         False
4         False   False  False          False   False         False
..          ...     ...    ...            ...     ...           ...
995       False   False  False          False   False         False
996       False   False  False          False   False         False
997       False   False  False          False   False         False
998       False   False  False          False   False         False
999       False   False  False          False   False         False

     Unit price  Quantity  Tax 5%  Total   Date   Time  Payment   cogs
\
0         False     False   False  False  False  False    False  False

1         False     False   False  False  False  False    False  False

2         False     False   False  False  False  False    False  False

3         False     False   False  False  False  False    False  False
```

```
4         False     False   False  False  False  False    False  False

..          ...       ...     ...    ...    ...    ...      ...    ...

995       False     False   False  False  False  False    False  False

996       False     False   False  False  False  False    False  False

997       False     False   False  False  False  False    False  False

998       False     False   False  False  False  False    False  False

999       False     False   False  False  False  False    False  False


     gross margin percentage  gross income  Rating
0                      False         False   False
1                      False         False   False
2                      False         False   False
3                      False         False   False
4                      False         False   False
..                       ...           ...     ...
995                    False         False   False
996                    False         False   False
997                    False         False   False
998                    False         False   False
999                    False         False   False

[1000 rows x 17 columns]
```

```
# Find the number of missing values in the dataset
sales_df.isnull().sum()
```

```
Invoice ID                 0
Branch                     0
City                       0
Customer type              0
Gender                     0
Product line               0
Unit price                 0
Quantity                   0
Tax 5%                     0
Total                      0
Date                       0
Time                       0
Payment                    0
cogs                       0
gross margin percentage    0
gross income               0
```

```
Rating                        0
dtype: int64
```

Descriptive Statistical Measures of a DataFrame

```
# Summary statistics of numerical columns
sales_df.describe()
```

```
        Unit price     Quantity       Tax 5%        Total         cogs
\
count  1000.000000  1000.000000  1000.000000  1000.000000  1000.00000

mean     55.672130     5.510000    15.379369   322.966749   307.58738

std      26.494628     2.923431    11.708825   245.885335   234.17651

min      10.080000     1.000000     0.508500    10.678500    10.17000

25%      32.875000     3.000000     5.924875   124.422375   118.49750

50%      55.230000     5.000000    12.088000   253.848000   241.76000

75%      77.935000     8.000000    22.445250   471.350250   448.90500

max      99.960000    10.000000    49.650000  1042.650000   993.00000


       gross margin percentage  gross income      Rating
count             1.000000e+03   1000.000000  1000.00000
mean              4.761905e+00     15.379369     6.97270
std               6.131498e-14     11.708825     1.71858
min               4.761905e+00      0.508500     4.00000
25%               4.761905e+00      5.924875     5.50000
50%               4.761905e+00     12.088000     7.00000
75%               4.761905e+00     22.445250     8.50000
max               4.761905e+00     49.650000    10.00000
```

Check Data Types

```
print(sales_df.dtypes)
```

```
Invoice ID                 object
Branch                     object
City                       object
Customer type              object
Gender                     object
Product line               object
Unit price                float64
Quantity                    int64
Tax 5%                    float64
Total                     float64
```

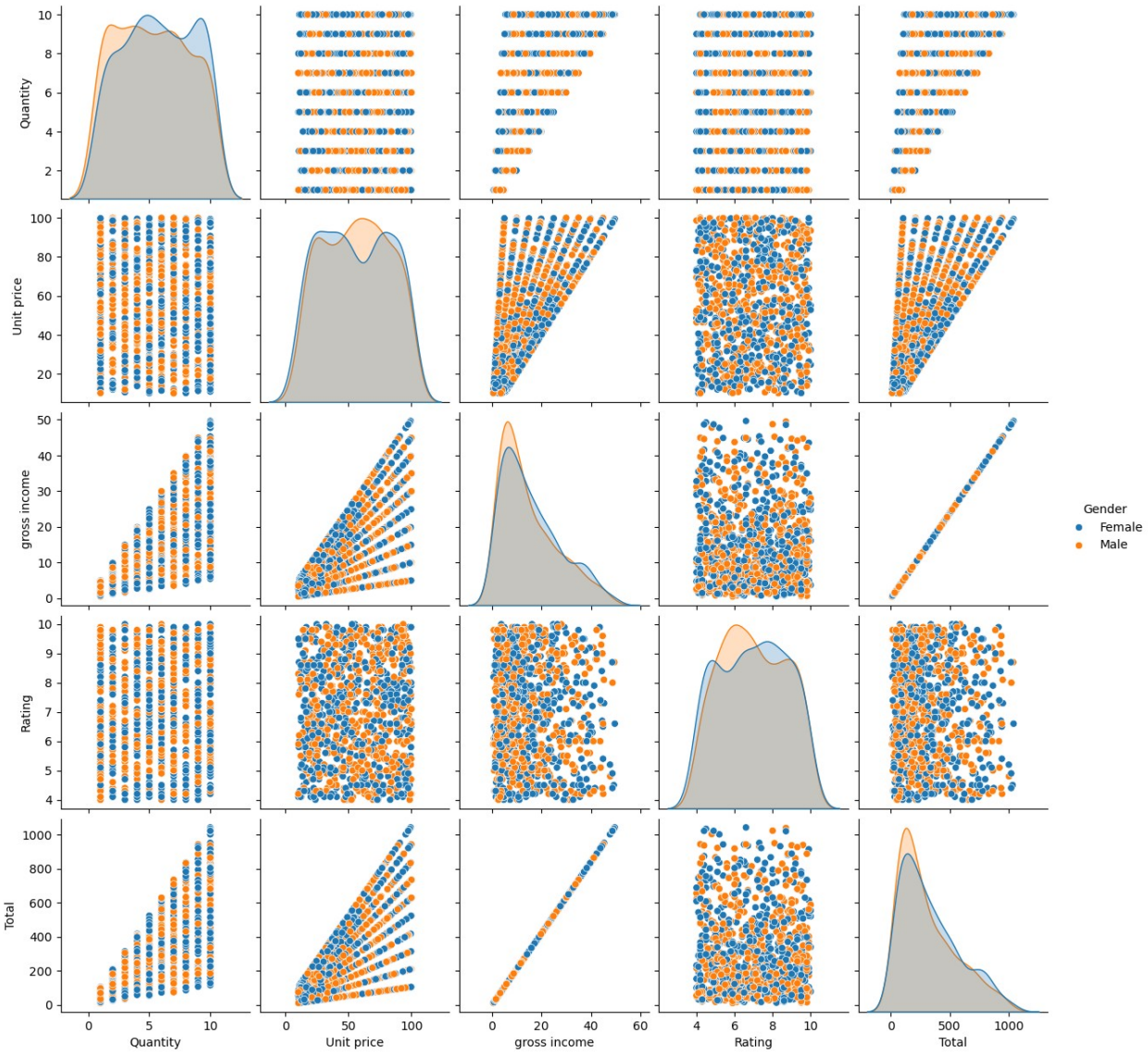
```
Date                          object
Time                          object
Payment                       object
cogs                         float64
gross margin percentage      float64
gross income                 float64
Rating                       float64
dtype: object
```

Correlation Heatmap

```
# Compute correlation matrix for numerical features
correlation_matrix = sales_df.select_dtypes(include=['float64',
'int64']).corr()

# Generate a heatmap of the correlation matrix
plt.figure(figsize=(10, 5))
sns.heatmap(correlation_matrix, annot=True, cmap="RdYlBu",
linewidths=0.5)
plt.title("Correlation Heatmap of Sales Data")
plt.show()
```

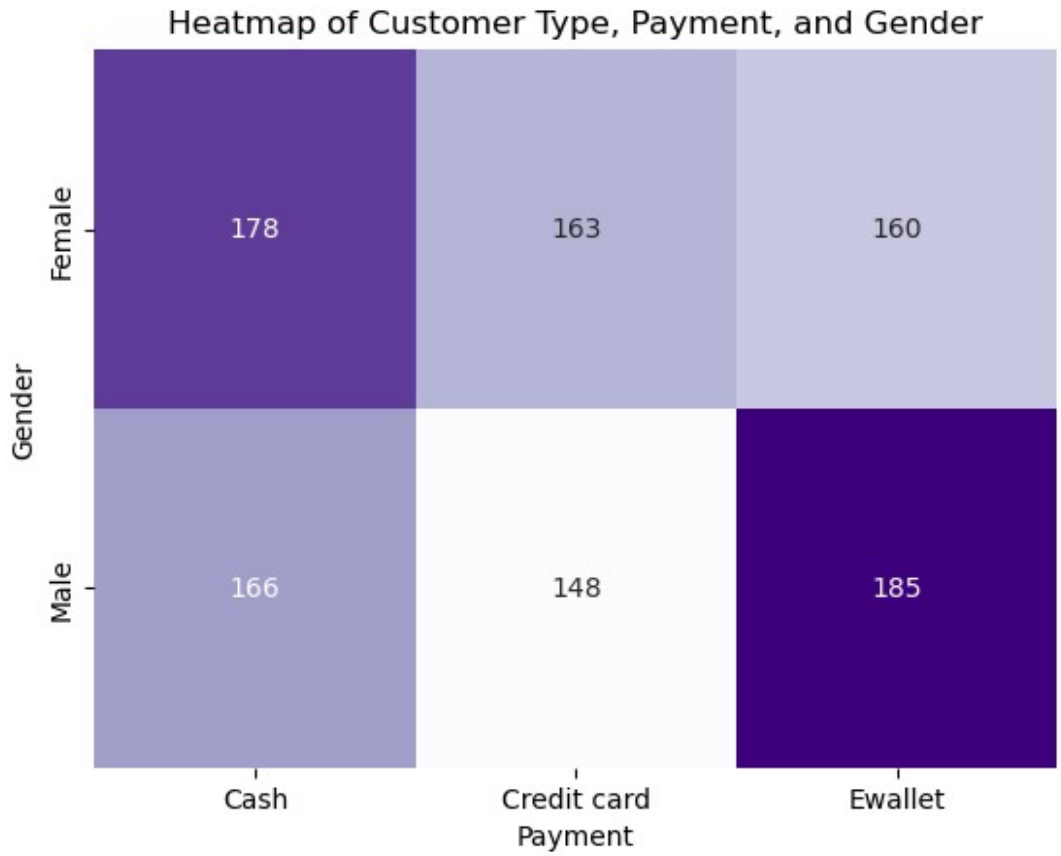
Pairplot for Important Numerical Features

```
# Pairplot to analyze relationships
sns.pairplot(sales_df, vars=['Quantity', 'Unit price', 'gross income',
'Rating', 'Total'], hue="Gender")
plt.show()
```



Customer Type vs Payment Heatmap

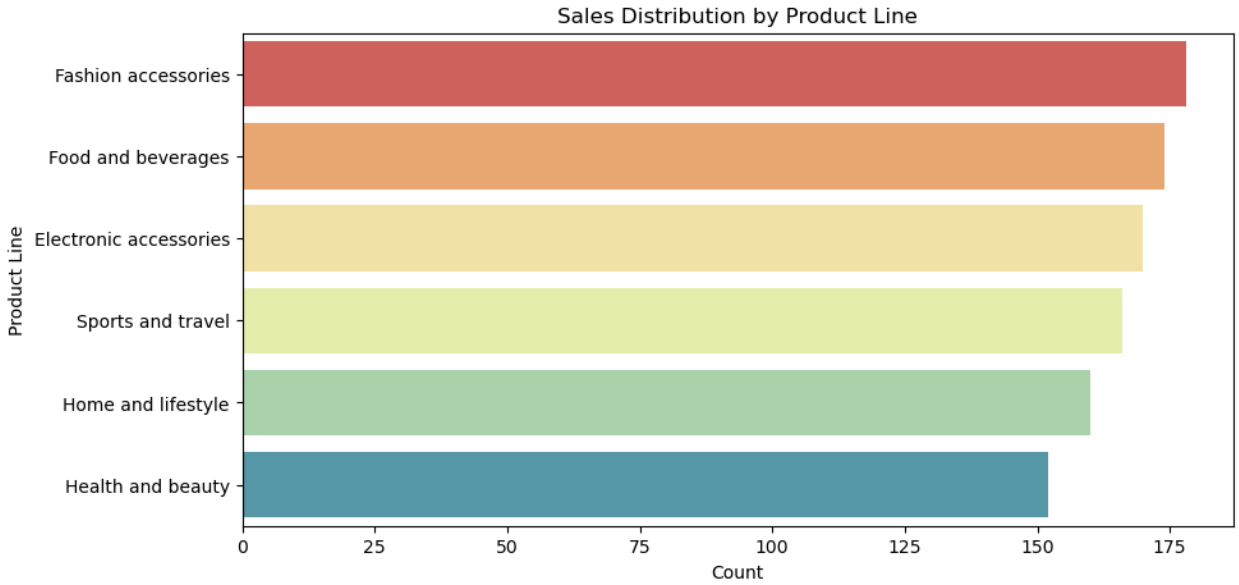
```
pivot_table = sales_df.pivot_table(index='Gender', columns='Payment',
values='Customer type', aggfunc='count', fill_value=0)

sns.heatmap(pivot_table, annot=True, fmt='d', cmap='Purples',
cbar=False)
plt.title("Heatmap of Customer Type, Payment, and Gender")
plt.show()
```



Sales Distribution by Product Line

```
plt.figure(figsize=(10, 5))
sns.countplot(y="Product line", data=sales_df, order=sales_df["Product
line"].value_counts().index, palette="Spectral")
plt.title("Sales Distribution by Product Line")
plt.xlabel("Count")
plt.ylabel("Product Line")
plt.show()
```

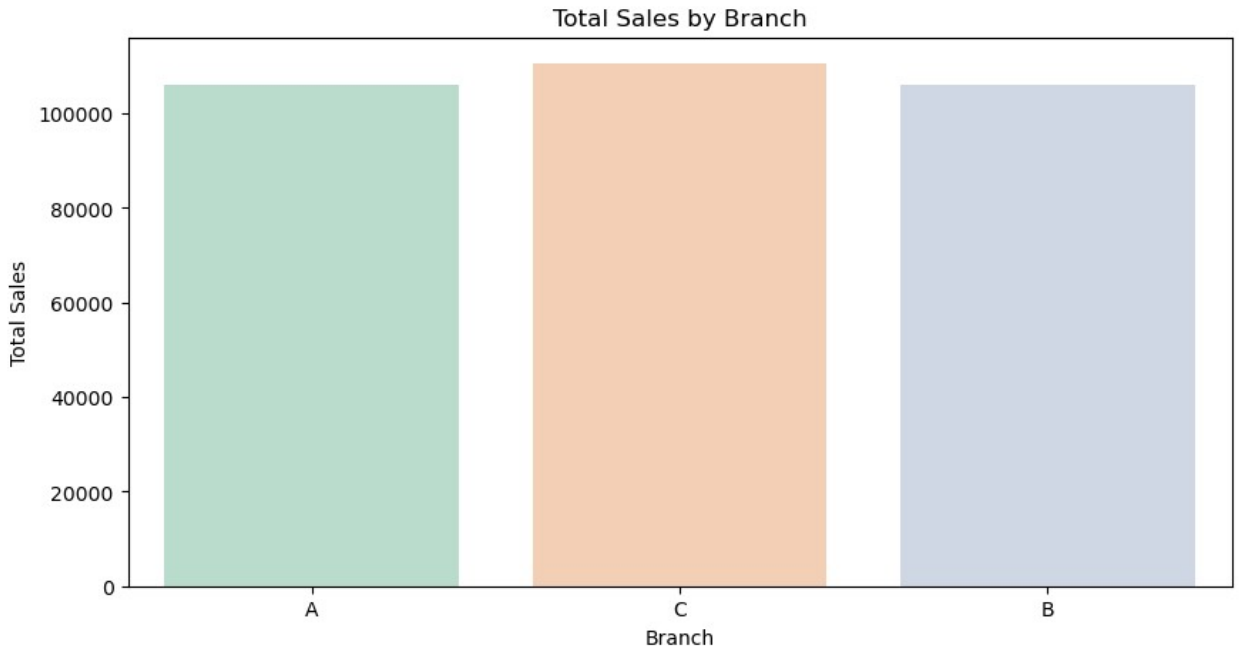
```
C:\Users\Asus\AppData\Local\Temp\ipykernel_7480\699676609.py:2:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `y` variable to `hue` and set
`legend=False` for the same effect.

  sns.countplot(y="Product line", data=sales_df,
order=sales_df["Product line"].value_counts().index,
palette="Spectral")
```



Total Sales by Branch

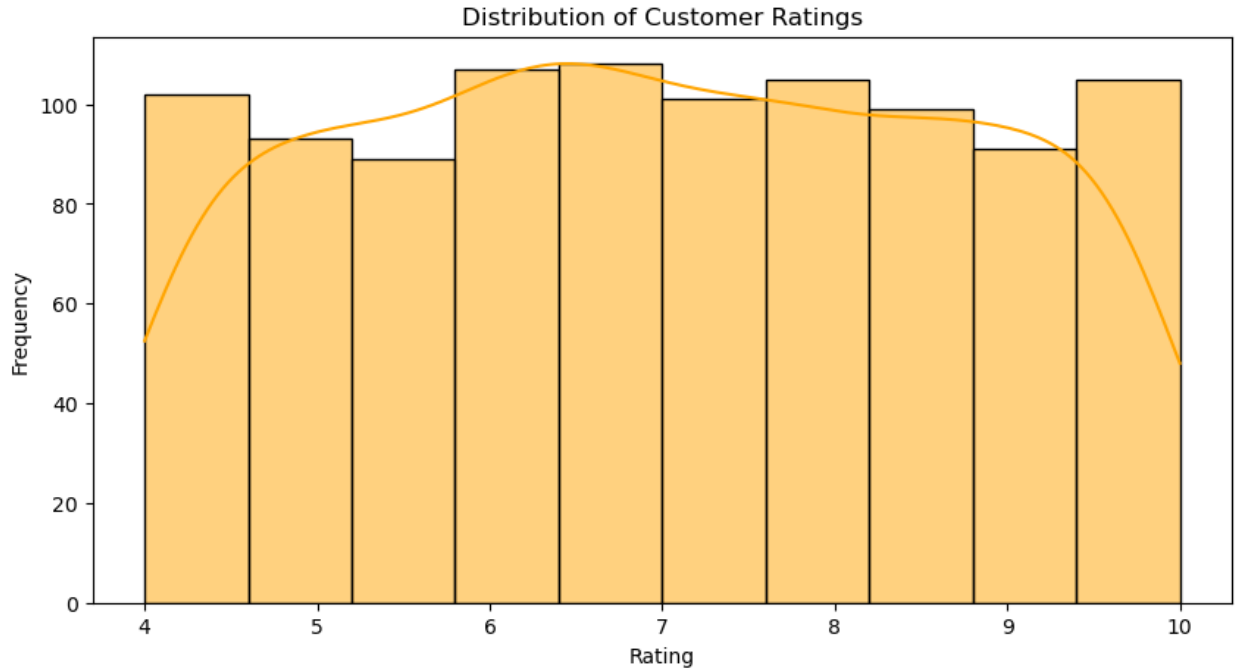
```
plt.figure(figsize=(10, 5))
sns.barplot(x="Branch", y="Total", data=sales_df, estimator=sum,
ci=None, palette="Pastel2")
plt.title("Total Sales by Branch")
plt.xlabel("Branch")
plt.ylabel("Total Sales")
plt.show()
```

```
C:\Users\Asus\AppData\Local\Temp\ipykernel_7480\693187590.py:2:
FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same
effect.

  sns.barplot(x="Branch", y="Total", data=sales_df, estimator=sum,
ci=None, palette="Pastel2")
C:\Users\Asus\AppData\Local\Temp\ipykernel_7480\693187590.py:2:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x="Branch", y="Total", data=sales_df, estimator=sum,
ci=None, palette="Pastel2")
```



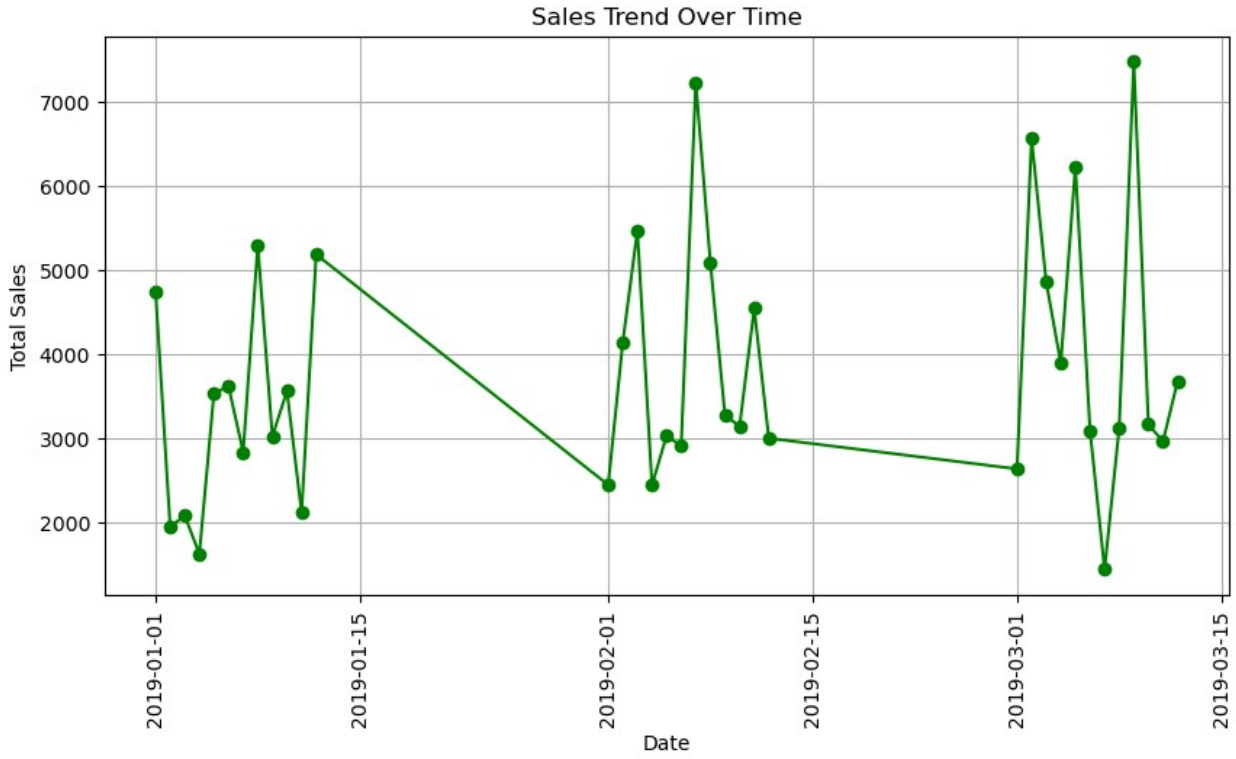
Distribution of Ratings

```
plt.figure(figsize=(10, 5))
sns.histplot(sales_df["Rating"], bins=10, kde=True, color="Orange")
plt.title("Distribution of Customer Ratings")
plt.xlabel("Rating")
plt.ylabel("Frequency")
plt.show()
```



Sales Trends Over Time

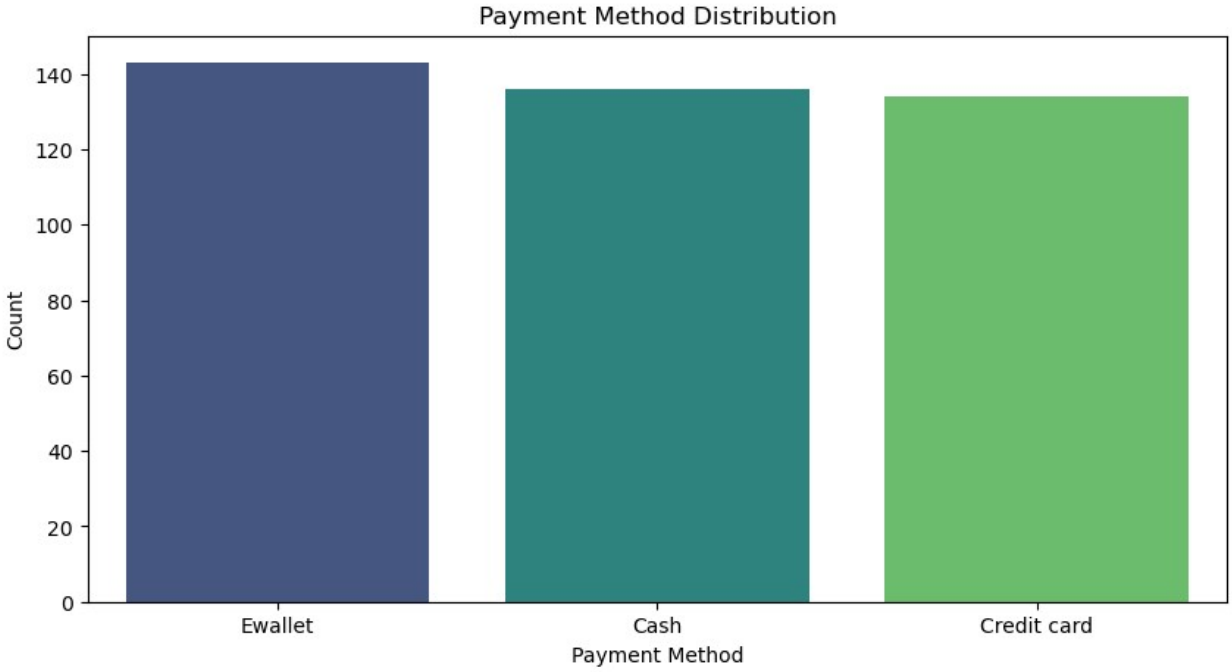
```
# Convert 'Date' column to datetime format, handling errors
sales_df["Date"] = pd.to_datetime(sales_df["Date"], errors="coerce")

# Drop rows where 'Date' conversion failed
sales_df = sales_df.dropna(subset=["Date"])

# Ensure 'Total' column is numeric
sales_df["Total"] = pd.to_numeric(sales_df["Total"], errors="coerce")

# Group by Date and sum the total sales
daily_sales = sales_df.groupby("Date")["Total"].sum()

# Plot the sales trend
plt.figure(figsize=(10, 5))
plt.plot(daily_sales.index, daily_sales.values, marker="o",
linestyle="-", color="green")
plt.title("Sales Trend Over Time")
plt.xlabel("Date")
plt.ylabel("Total Sales")
plt.xticks(rotation=90)
plt.grid(True)
plt.show()
```



Payment Method Distribution

```
plt.figure(figsize=(10, 5))
sns.countplot(x="Payment", data=sales_df, palette="viridis")
plt.title("Payment Method Distribution")
plt.xlabel("Payment Method")
plt.ylabel("Count")
plt.show()
```

```
C:\Users\Asus\AppData\Local\Temp\ipykernel_7480\2000526270.py:2:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.countplot(x="Payment", data=sales_df, palette="viridis")
```



Boxplot of Unit Price by Product Line

```
plt.figure(figsize=(10, 5))
sns.boxplot(x="Product line", y="Unit price", data=sales_df,
palette="viridis")
plt.title("Boxplot of Unit Price by Product Line")
plt.xticks(rotation=90)
plt.show()
```

```
C:\Users\Asus\AppData\Local\Temp\ipykernel_7480\682750.py:2:
FutureWarning:
```

```
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x="Product line", y="Unit price", data=sales_df, palette="viridis")
```