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

```
df=pd.read_csv("/content/retail_sales_dataset.csv")
df
```

	Transaction ID	Date	Customer ID	Gender	Age	Product Category	Quantity	Price per Unit	Total Amount
0	1	2023-11-24	CUST001	Male	34	Beauty	3	50	150
1	2	2023-02-27	CUST002	Female	26	Clothing	2	500	1000
2	3	2023-01-13	CUST003	Male	50	Electronics	1	30	30
3	4	2023-05-21	CUST004	Male	37	Clothing	1	500	500
4	5	2023-05-06	CUST005	Male	30	Beauty	2	50	100
...	...	...	...	...	...	...	...	...	...
995	996	2023-05-16	CUST996	Male	62	Clothing	1	50	50
996	997	2023-	CUST997	Male	52	Beauty	3	30	90

Display basic information about the dataset

```
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.head()
```

	Transaction ID	Date	Customer ID	Gender	Age	Product Category	Quantity	Price per Unit	Total Amount
0	1	2023-11-24	CUST001	Male	34	Beauty	3	50	150
1	2	2023-02-27	CUST002	Female	26	Clothing	2	500	1000
2	3	2023-01-13	CUST003	Male	50	Electronics	1	30	30

```
df.tail()
```

	Transaction ID	Date	Customer ID	Gender	Age	Product Category	Quantity	Price per Unit	Total Amount
995	996	2023-05-16	CUST996	Male	62	Clothing	1	50	50
996	997	2023-11-17	CUST997	Male	52	Beauty	3	30	90
997	998	2023-10-29	CUST998	Female	23	Beauty	4	25	100

Display the shape of the dataset

```
row,columns=df.shape
print("The number of rows are",row)
print("The number of columns are",columns)
```

```
The number of rows are 1000
The number of columns are 9
```

```
df.size
```

```
9000
```

```
df.columns
```

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

```
df.dtypes
```

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

```
df.isna().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
```

In this dataframe, the date, gender, and product_category columns are incorrectly assigned data types; now, they need to be changed to the correct data types.

```
# change date data type
df['Date'] = pd.to_datetime(df['Date'])
```

```
# change gender data type
df['Gender'] = df['Gender'].astype('category')
```

```
# change product_category data type
df['Product Category'] = df['Product Category'].astype('category')
```

```
# change Customer ID data type
df['Customer ID'] = df['Customer ID'].astype('category')
```

```
# remove Transaction ID which is not useful our analysis
df.drop('Transaction ID', axis=1, inplace=True)
```

```
df.dtypes
```

```
Date               datetime64[ns]
Customer ID              category
Gender                   category
Age                         int64
Product Category         category
Quantity                    int64
Price per Unit              int64
Total Amount                int64
dtype: object
```

```
df['Date'].describe()
```

```
<ipython-input-22-a430526189f8>:1: FutureWarning: Treating datetime data as categorical rather than numeric in `.describe` is deprecated and w
  df['Date'].describe()
```

```
count                    1000
unique                    345
top       2023-05-16 00:00:00
freq                       11
first     2023-01-01 00:00:00
last      2024-01-01 00:00:00
Name: Date, dtype: object
```

Calculate basic statistics

```
mean_df = df.mean()
median_df = df.median()
mode_df = df.mode().iloc[0]
std_dev_df = df.std()
```

```
<ipython-input-37-5f50c0298cbf>:1: FutureWarning: DataFrame.mean and DataFrame.median with numeric_only=None will include datetime64 and datet
  mean_df = df.mean()
<ipython-input-37-5f50c0298cbf>:1: FutureWarning: The default value of numeric_only in DataFrame.mean is deprecated. In a future version, it w
  mean_df = df.mean()
<ipython-input-37-5f50c0298cbf>:2: FutureWarning: DataFrame.mean and DataFrame.median with numeric_only=None will include datetime64 and datet
  median_df = df.median()
<ipython-input-37-5f50c0298cbf>:2: FutureWarning: The default value of numeric_only in DataFrame.median is deprecated. In a future version, it
  median_df = df.median()
<ipython-input-37-5f50c0298cbf>:4: FutureWarning: The default value of numeric_only in DataFrame.std is deprecated. In a future version, it wi
  std_dev_df = df.std()
```

```
print("Dataset 1:")
print("Mean:")
print(mean_df)
print("\nMedian:")
print(median_df)
print("\nMode:")
print(mode_df)
print("\nStandard Deviation:")
print(std_dev_df)
```

```
Dataset 1:
Mean:
Age               41.392
Quantity           2.514
Price per Unit   179.890
Total Amount     456.000
dtype: float64

Median:
Age               42.0
Quantity           3.0
Price per Unit    50.0
Total Amount     135.0
dtype: float64

Mode:
Date                2023-05-16 00:00:00
Customer ID                     CUST001
Gender                           Female
Age                                43.0
Product Category               Clothing
Quantity                            4.0
Price per Unit                     50.0
Total Amount                       50.0
Name: 0, dtype: object

Standard Deviation:
Date              105 days 06:38:33.163228086
Age                                 13.68143
Quantity                            1.132734
Price per Unit                    189.681356
Total Amount                      559.997632
dtype: object
```

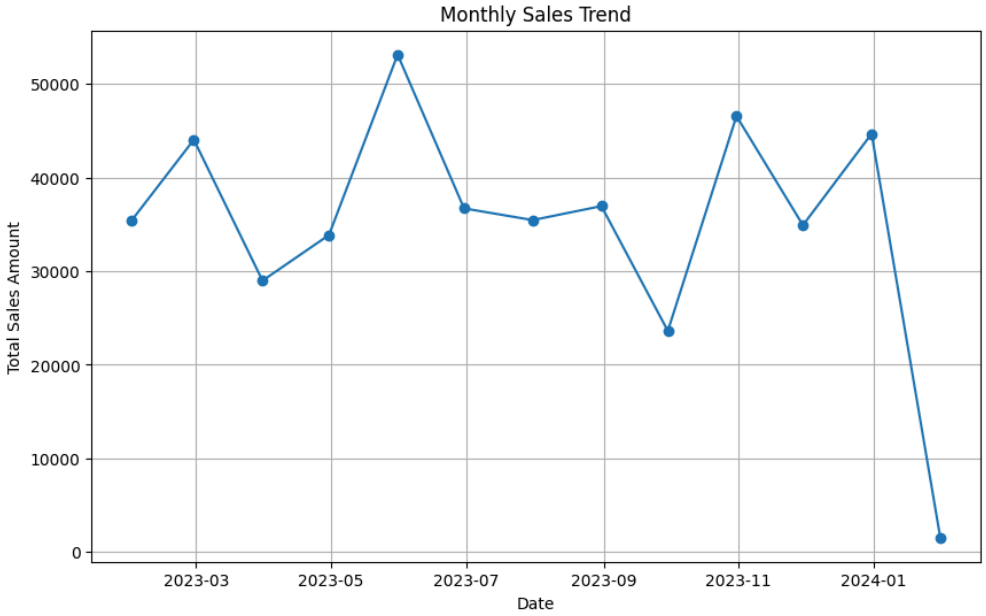
Time Series Analysis:

```
# Convert 'date' column to datetime
df['Date'] = pd.to_datetime(df['Date'])
```

```
# Set 'date' column as index
df.set_index('Date', inplace=True)
```

```
# Resample data to monthly frequency and sum the sales
monthly_sales = df['Total Amount'].resample('M').sum()
```

```
# Plot sales trend over time
plt.figure(figsize=(10, 6))
plt.plot(monthly_sales, marker='o', linestyle='-')
plt.title('Monthly Sales Trend')
plt.xlabel('Date')
plt.ylabel('Total Sales Amount')
plt.grid(True)
plt.show()
```



Customer and Product Analysis:

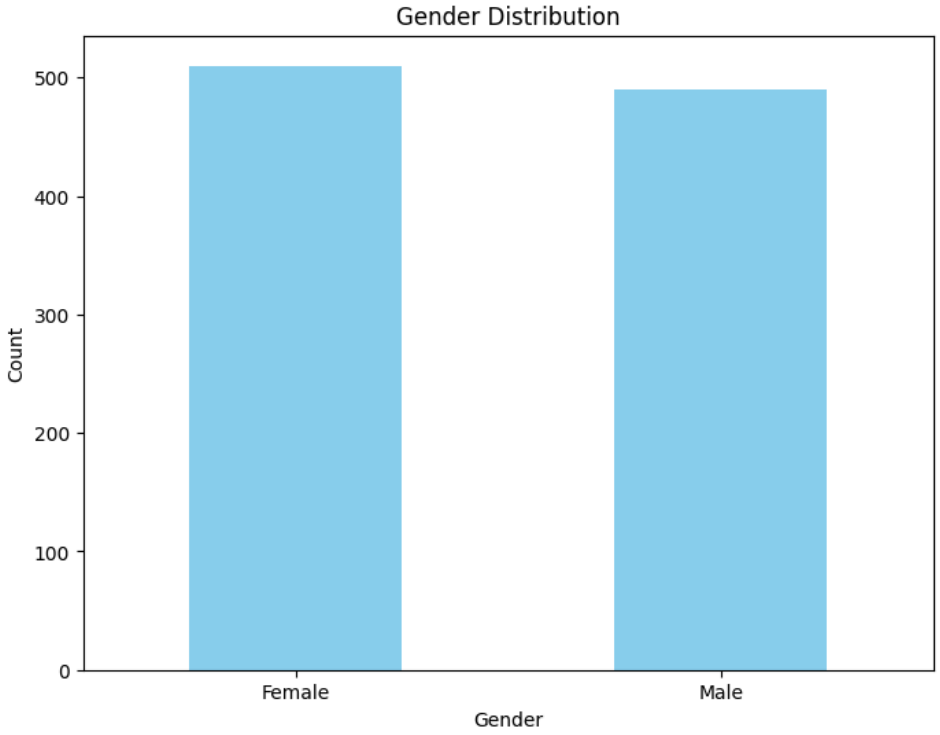
```
# Gender distribution
gender_distribution = df['Gender'].value_counts()
gender_distribution
```

```
Female    510
Male      490
Name: Gender, dtype: int64
```

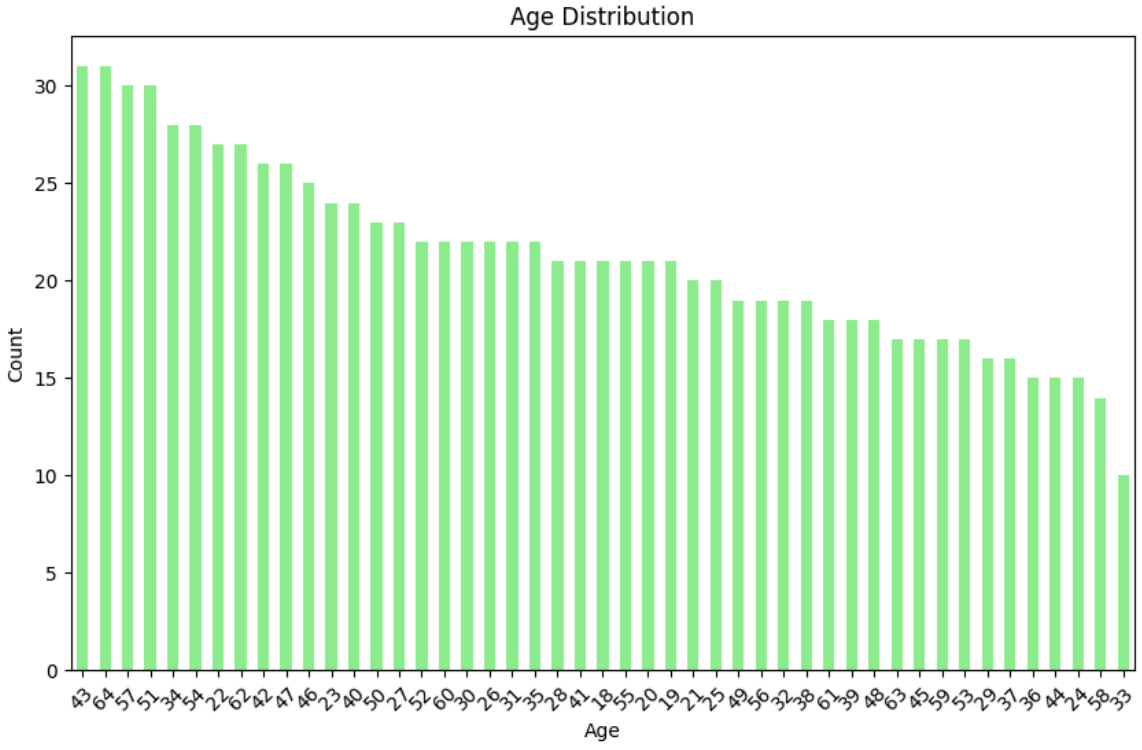
```
# Age distribution
age_distribution = df['Age'].value_counts()
```

```
# Product Preferences Analysis
# Total sales by product category
product_sales = df.groupby('Product Category')['Total Amount'].sum()
```

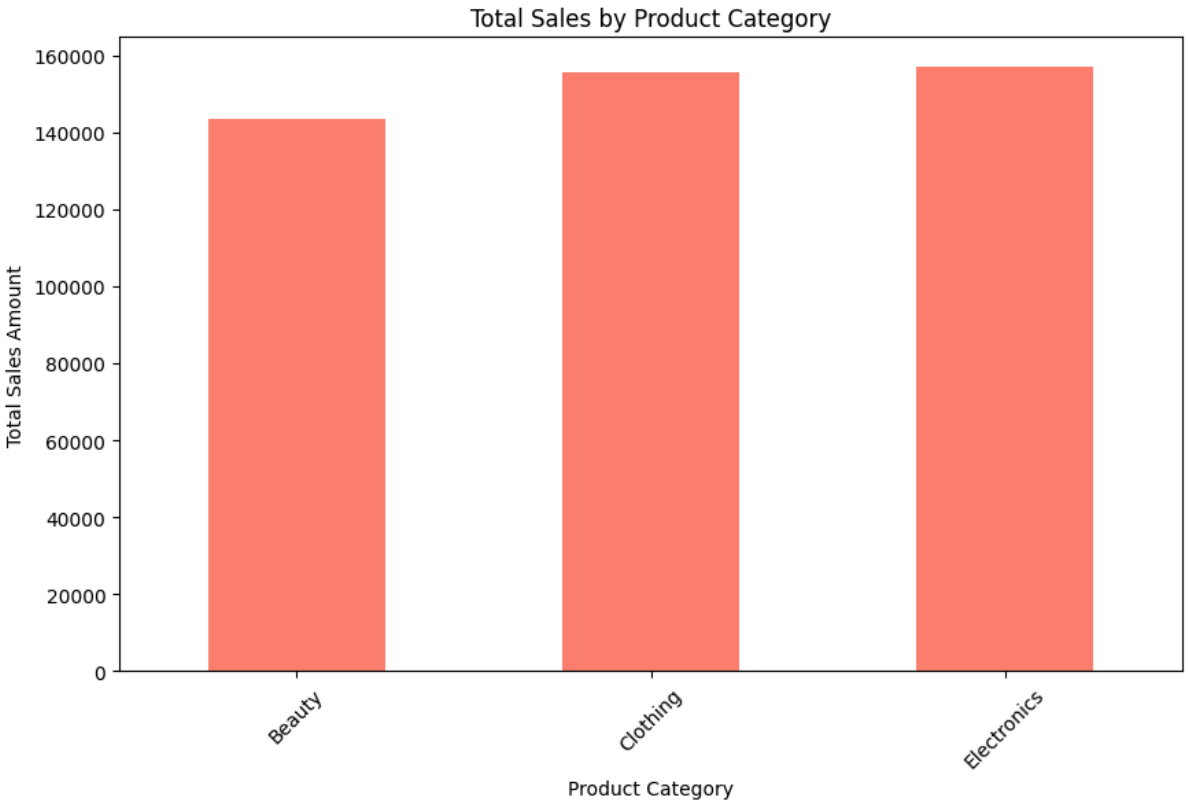
```
# Visualization
# Gender distribution
plt.figure(figsize=(8, 6))
gender_distribution.plot(kind='bar', color='skyblue')
plt.title('Gender Distribution')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.xticks(rotation=0)
plt.show()
```




```
# Age distribution
plt.figure(figsize=(10, 6))
age_distribution.plot(kind='bar', color='lightgreen')
plt.title('Age Distribution')
plt.xlabel('Age')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```

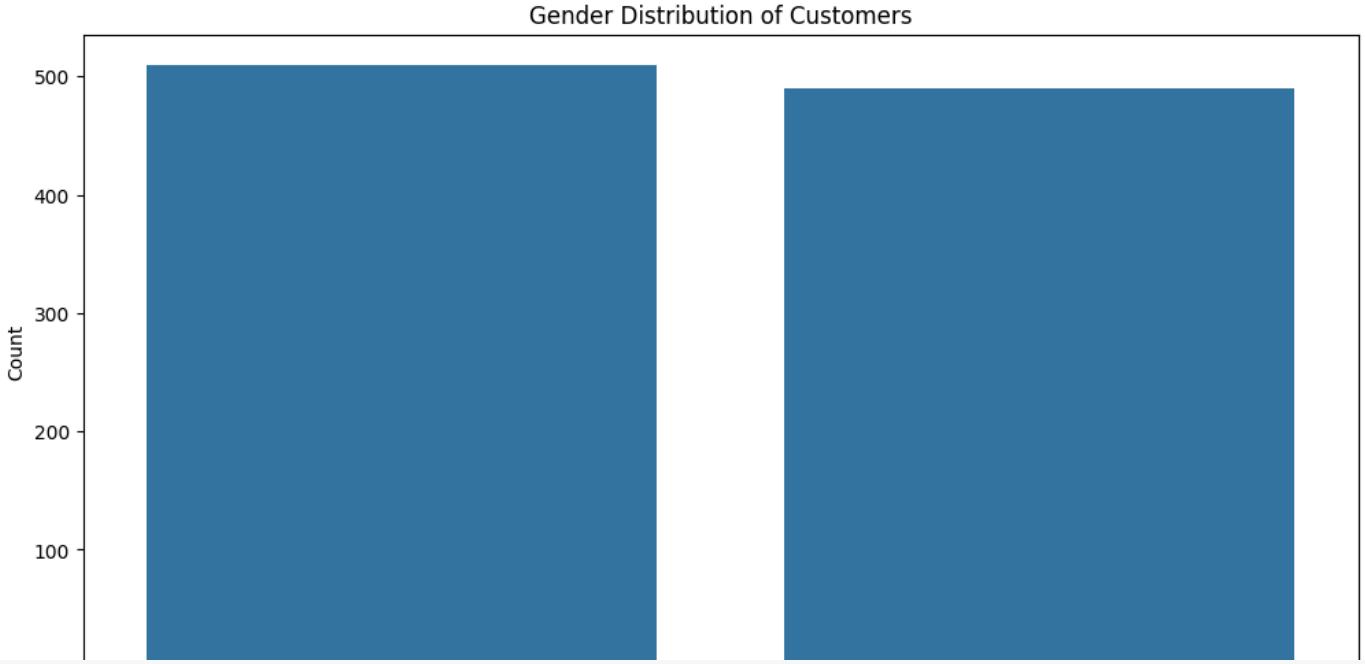



```
# Product preferences
plt.figure(figsize=(10, 6))
product_sales.plot(kind='bar', color='salmon')
plt.title('Total Sales by Product Category')
plt.xlabel('Product Category')
plt.ylabel('Total Sales Amount')
plt.xticks(rotation=45)
plt.show()
```

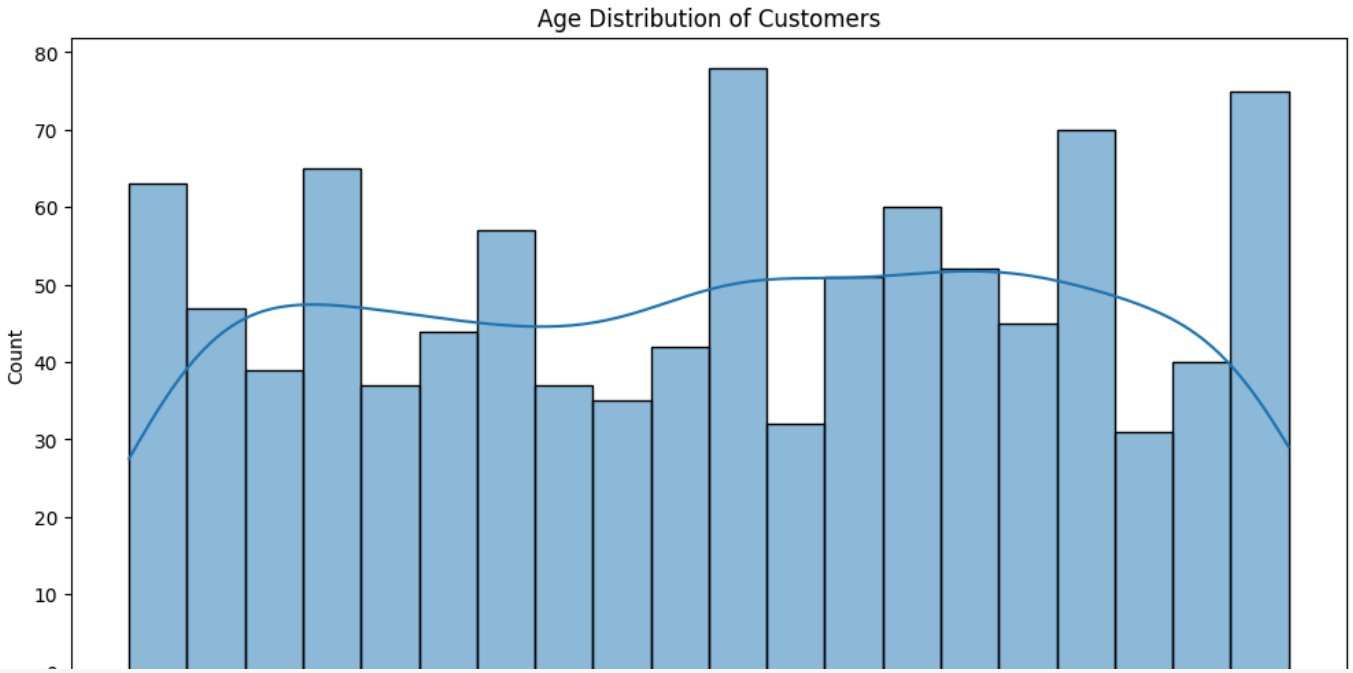


Visualization:

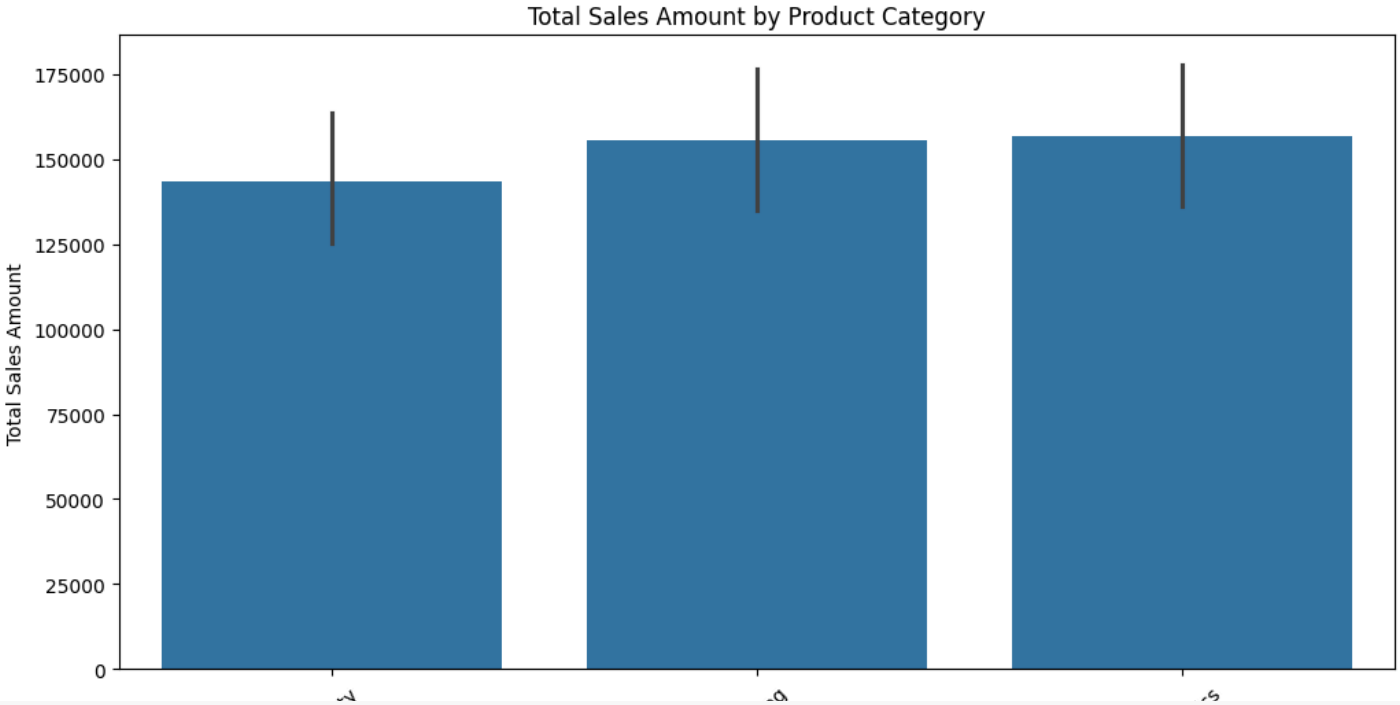
```
# Customer demographics analysis
plt.figure(figsize=(12, 6))
sns.countplot(data=df, x='Gender')
plt.title('Gender Distribution of Customers')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.show()
```



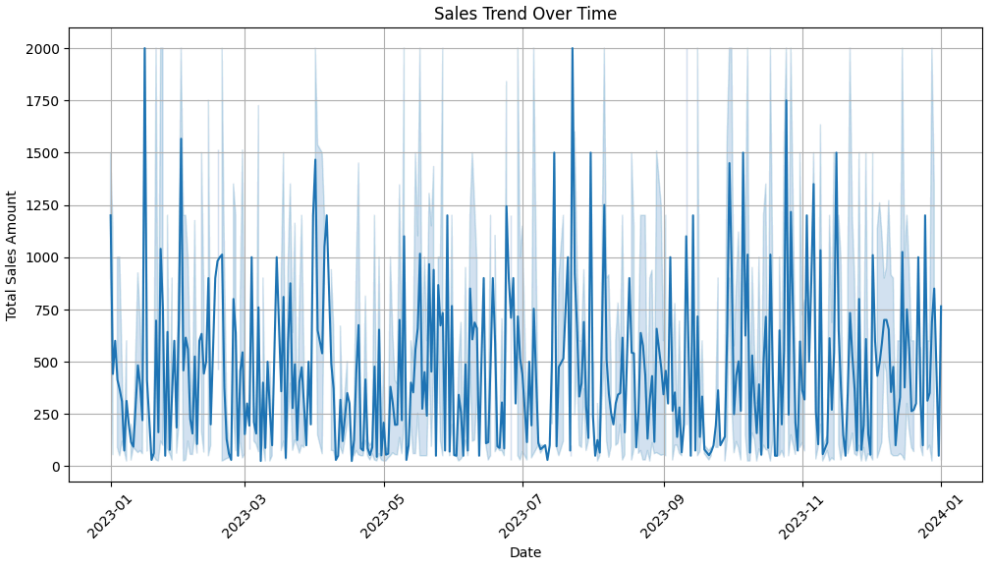

```
plt.figure(figsize=(12, 6))
sns.histplot(data=df, x='Age', bins=20, kde=True)
plt.title('Age Distribution of Customers')
plt.xlabel('Age')
plt.ylabel('Count')
plt.show()
```




```
# Purchasing behavior analysis
plt.figure(figsize=(12, 6))
sns.barplot(data=df, x='Product Category', y='Total Amount', estimator=sum)
plt.title('Total Sales Amount by Product Category')
plt.xlabel('Product Category')
plt.ylabel('Total Sales Amount')
plt.xticks(rotation=45)
plt.show()
```

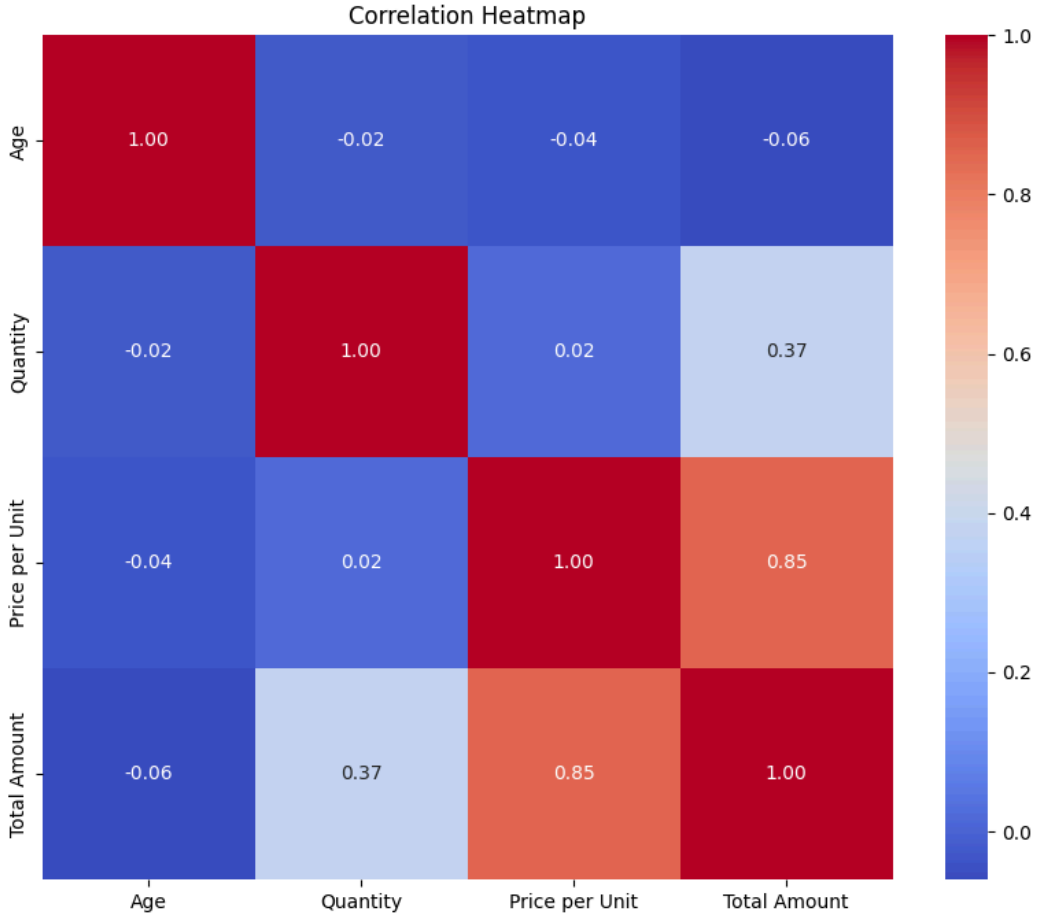



```
plt.figure(figsize=(12, 6))
sns.lineplot(data=df, x='Date', y='Total Amount')
plt.title('Sales Trend Over Time')
plt.xlabel('Date')
plt.ylabel('Total Sales Amount')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```




```
# Heatmap for correlation analysis
plt.figure(figsize=(10, 8))
correlation_matrix = df.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()
```

```
<ipython-input-57-de6cbd938a8a>:3: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it w
  correlation_matrix = df.corr()
```

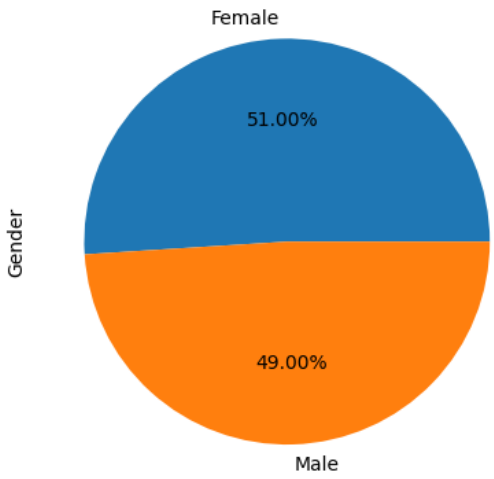



```
# gender column frequency
df['Gender'].value_counts()
```

```
Female    510
Male      490
Name: Gender, dtype: int64
```

```
df['Gender'].value_counts().plot(kind='pie', autopct='%.2f%%')
```

```
<Axes: ylabel='Gender'>
```



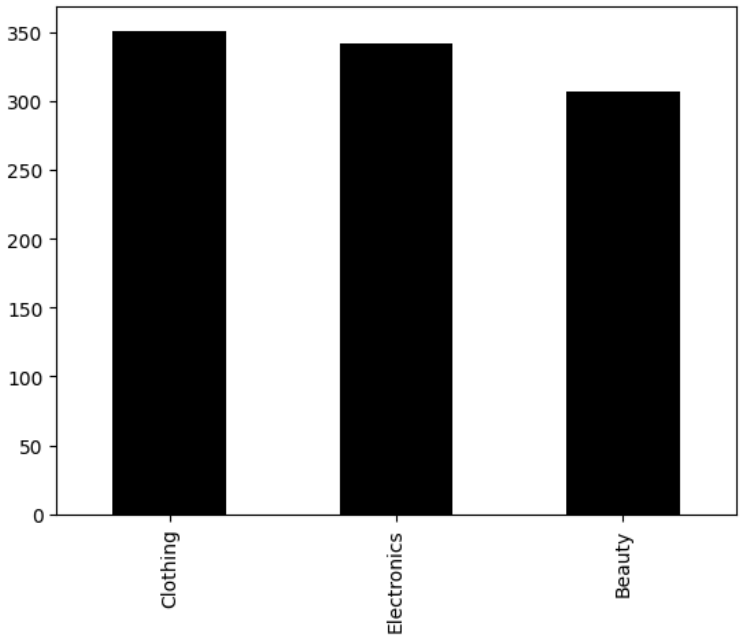
The retail sales data shows a slight majority of female customers (51%) compared to males (49%), indicating a relatively balanced gender distribution in the customer base.

```
# frequency of product_category
df['Product Category'].value_counts()
```

```
Clothing       351
Electronics    342
Beauty         307
Name: Product Category, dtype: int64
```

```
df['Product Category'].value_counts().plot(kind='bar', color='black')
```

```
<Axes: >
```



Product Category conclusions:

The sales data reveals that clothing, electronics, and beauty products are the top three categories, with clothing leading slightly (351), followed closely by electronics (342), and beauty products (307).

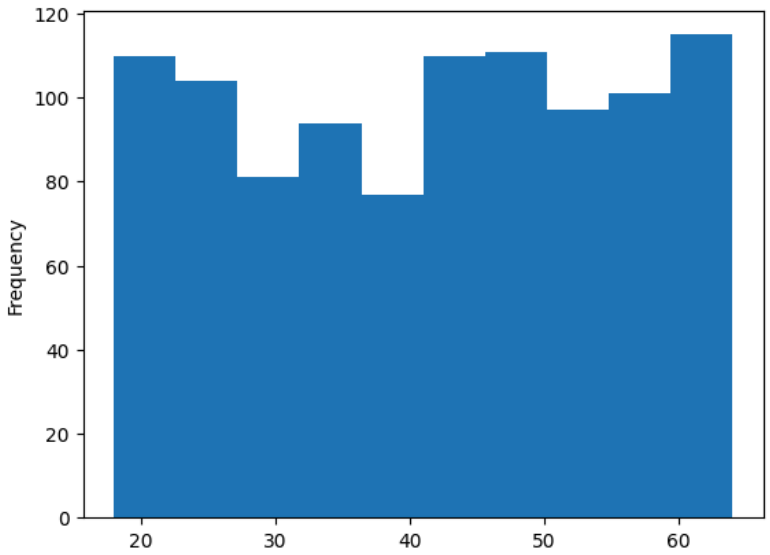
```
df['Age'].describe()
```

```
count    1000.00000
mean       41.39200
std        13.68143
min        18.00000
25%        29.00000
50%        42.00000
75%        53.00000
```

```
max        64.00000
Name: Age, dtype: float64
```

```
df['Age'].plot(kind='hist')
```

```
<Axes: ylabel='Frequency'>
```




```
df['Age'].skew()
```

```
-0.04881245380328967
```

Age conclusions:

The skewness score of -0.0488 suggests a near-symmetrical distribution, which aligns with the appearance of a uniform distribution. The age distribution of customers exhibits a very slight negative skew (-0.0488), indicating a slight tendency towards younger ages, though overall it remains relatively symmetrical. With a mean age of 41.39 years and a standard deviation of 13.68, customers range from 18 to 64 years old, with median and quartile values providing insight into the central tendency and dispersion of the data.

Analyze sales performance by product category

```
product_sales = df.groupby('Product Category')['Total Amount'].sum().sort_values(ascending=False)
plt.figure(figsize=(10, 6))
sns.barplot(x=product_sales.values, y=product_sales.index, palette='viridis')
plt.title('Sales Performance by Product Category')
plt.xlabel('Total Sales Amount')
plt.ylabel('Product Category')
plt.show()
```

```
<ipython-input-58-d299194f3098>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign

  sns.barplot(x=product_sales.values, y=product_sales.index, palette='viridis')
```