```
#Import Libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

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

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
import pandas as pd
# Step 3: Read Your sales_data.csv
df= pd.read_csv('sales_data.csv',encoding='latin1')
df.head()
```

| | Row_ID | Order_ID | Order_Date | Ship_ Date | Ship_Mode | Customer_ID | Customer_Name | Segment | Country | City | ... | Postal_Code | Regio |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | CA-2016-152156 | 08-11-2016 | 11-11-2016 | Second Class | CG-12520 | Claire Gute | Consumer | United States | Henderson | ... | 42420 | Soutl |
| 1 | 3 | CA-2016-138688 | 12-06-2016 | 16-06-2016 | Second Class | DV-13045 | Darrin Van Huff | Corporate | United States | Los Angeles | ... | 90036 | Wes |
| 2 | 4 | US-2015-108966 | 11-10-2015 | 18-10-2015 | Standard Class | SO-20335 | Sean O'Donnell | Consumer | United States | Fort Lauderdale | ... | 33311 | Soutl |
| 3 | 6 | CA-2014-115812 | 09-06-2014 | 14-06-2014 | Standard Class | BH-11710 | Brosina Hoffman | Consumer | United States | Los Angeles | ... | 90032 | Wes |
| 4 | 13 | CA-2017-114412 | 15-04-2017 | 20-04-2017 | Standard Class | AA-10480 | Andrew Allen | Consumer | United States | Concord | ... | 28027 | Soutl |

5 rows × 21 columns

```
# Basic info about dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5009 entries, 0 to 5008
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Row_ID         5009 non-null   int64
 1   Order_ID       5009 non-null   object
 2   Order_Date     5009 non-null   object
 3   Ship_ Date     5009 non-null   object
 4   Ship_Mode      5009 non-null   object
 5   Customer_ID    5009 non-null   object
 6   Customer_Name  5009 non-null   object
 7   Segment        5009 non-null   object
 8   Country        5009 non-null   object
 9   City           5009 non-null   object
 10  State          5009 non-null   object
 11  Postal_Code    5009 non-null   int64
 12  Region         5009 non-null   object
 13  Product_ID     5009 non-null   object
 14  Category       5009 non-null   object
 15  Sub_Category   5009 non-null   object
 16  Product_Name   5009 non-null   object
 17  Sales          5009 non-null   float64
 18  Quantity       5009 non-null   int64
 19  Discount       5009 non-null   float64
 20  Profit         5009 non-null   float64
dtypes: float64(3), int64(3), object(15)
memory usage: 821.9+ KB
```

```
# Check for missing values
df.isnull().sum()
```

| | 0 |
|---|---|
| **Row_ID** | 0 |
| **Order_ID** | 0 |
| **Order_Date** | 0 |
| **Ship_ Date** | 0 |
| **Ship_Mode** | 0 |
| **Customer_ID** | 0 |
| **Customer_Name** | 0 |
| **Segment** | 0 |
| **Country** | 0 |
| **City** | 0 |
| **State** | 0 |
| **Postal_Code** | 0 |
| **Region** | 0 |
| **Product_ID** | 0 |
| **Category** | 0 |
| **Sub_Category** | 0 |
| **Product_Name** | 0 |
| **Sales** | 0 |
| **Quantity** | 0 |
| **Discount** | 0 |
| **Profit** | 0 |

**dtype:** int64

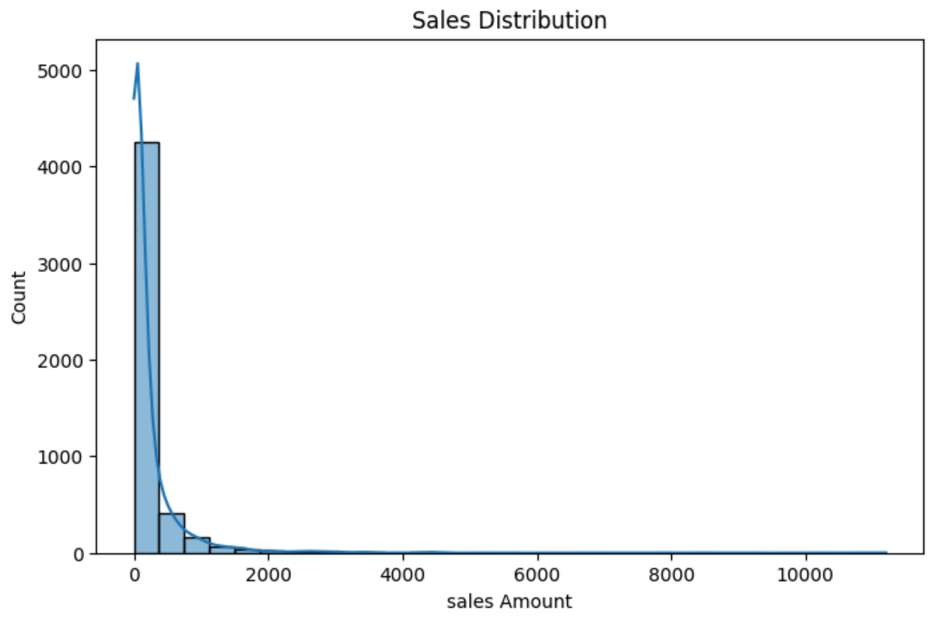
```
# See basic statistics
df.describe()
```

| | Row_ID | Postal_Code | Sales | Quantity | Discount | Profit |
|---|---|---|---|---|---|---|
| **count** | 5009.000000 | 5009.000000 | 5009.000000 | 5009.000000 | 5009.000000 | 5009.000000 |
| **mean** | 5044.337792 | 55410.136754 | 219.577213 | 3.801956 | 0.157007 | 26.455628 |
| **std** | 2884.456224 | 31989.416278 | 553.491762 | 2.223151 | 0.207802 | 207.275558 |
| **min** | 1.000000 | 1040.000000 | 0.560000 | 1.000000 | 0.000000 | -3839.990000 |
| **25%** | 2552.000000 | 23464.000000 | 16.270000 | 2.000000 | 0.000000 | 1.800000 |
| **50%** | 5060.000000 | 60068.000000 | 51.710000 | 3.000000 | 0.200000 | 8.470000 |
| **75%** | 7577.000000 | 90008.000000 | 199.990000 | 5.000000 | 0.200000 | 27.990000 |
| **max** | 9994.000000 | 99301.000000 | 11199.970000 | 14.000000 | 0.800000 | 5039.990000 |

```
# Check column names
df.columns
```

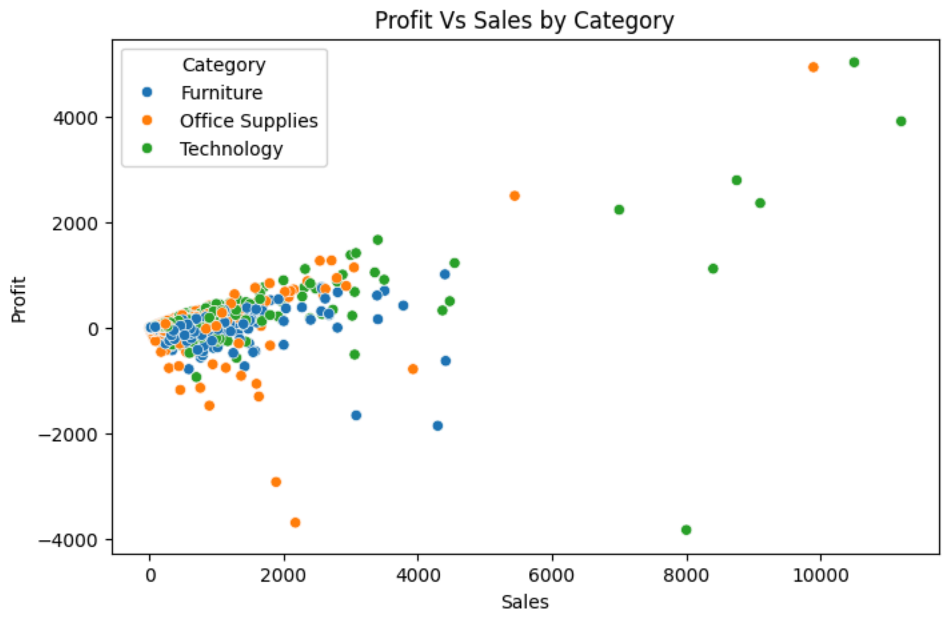
```
Index(['Row_ID', 'Order_ID', 'Order_Date', 'Ship_ Date', 'Ship_Mode',
       'Customer_ID', 'Customer_Name', 'Segment', 'Country', 'City', 'State',
       'Postal_Code', 'Region', 'Product_ID', 'Category', 'Sub_Category',
       'Product_Name', 'Sales', 'Quantity', 'Discount', 'Profit'],
      dtype='object')
```

```
#Sales Distribution – Histogram

plt.figure(figsize=(8,5))
sns.histplot(df['Sales'],bins=30,kde=True)
plt.title('Sales Distribution')
plt.xlabel('sales Amount')
plt.ylabel('Count')
plt.show()
```

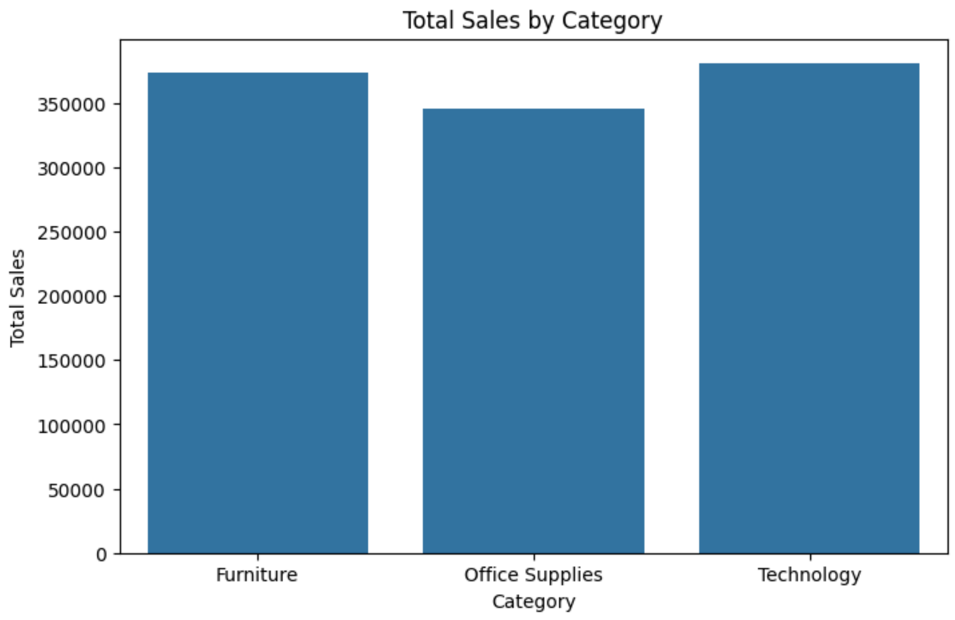


```
#Profit vs Sales - Scatter Plot

plt.figure(figsize=(8,5))
sns.scatterplot(data=df, x='Sales', y='Profit', hue='Category')
plt.title('Profit Vs Sales by Category')
plt.xlabel('Sales')
plt.ylabel('Profit')
plt.show
```

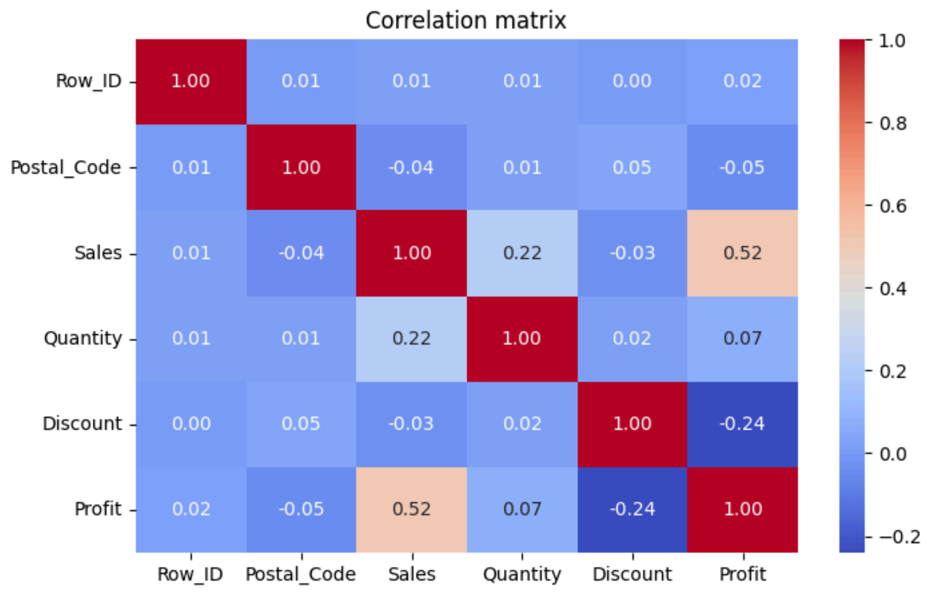
```
matplotlib.pyplot.show
def show(*args, **kwargs) -> None

Display all open figures.

Parameters
----------
block : bool, optional
    Whether to wait for all figures to be closed before returning.
```



```
# Total Sales by Category - Bar Chart
category_sales = df.groupby('Category')['Sales'].sum().reset_index()

plt.figure(figsize=(8,5))
sns.barplot(data=category_sales,x='Category',y='Sales')
plt.title('Total Sales by Category')
plt.ylabel('Total Sales')
plt.show()
```

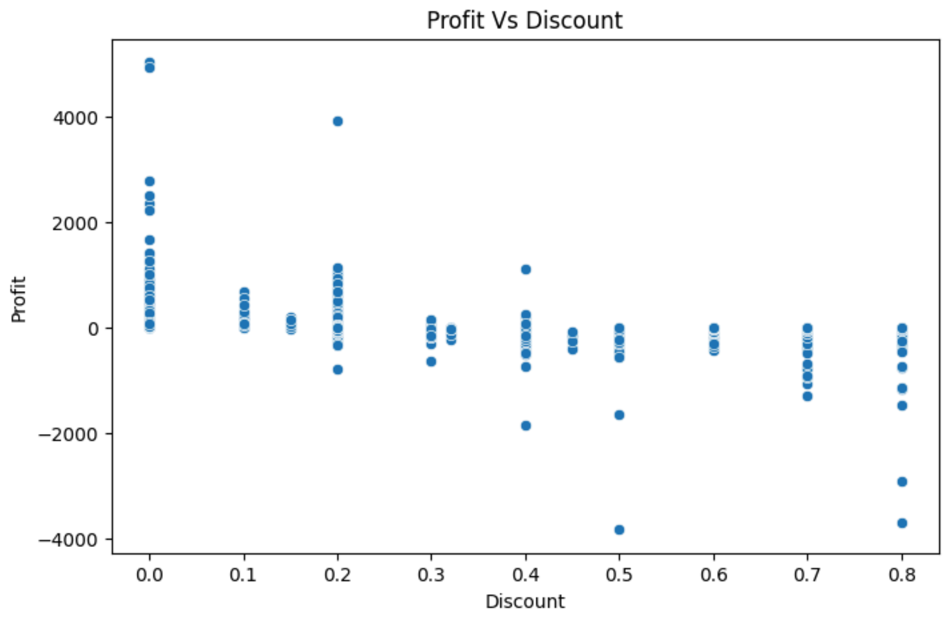


```
# Correlation matrix
corr = df.corr(numeric_only=True)

# Plot heatmap
plt.figure(figsize=(8,5))
sns.heatmap(corr,annot=True,cmap='coolwarm',fmt=".2f")
plt.title("Correlation matrix")
plt.show()
```

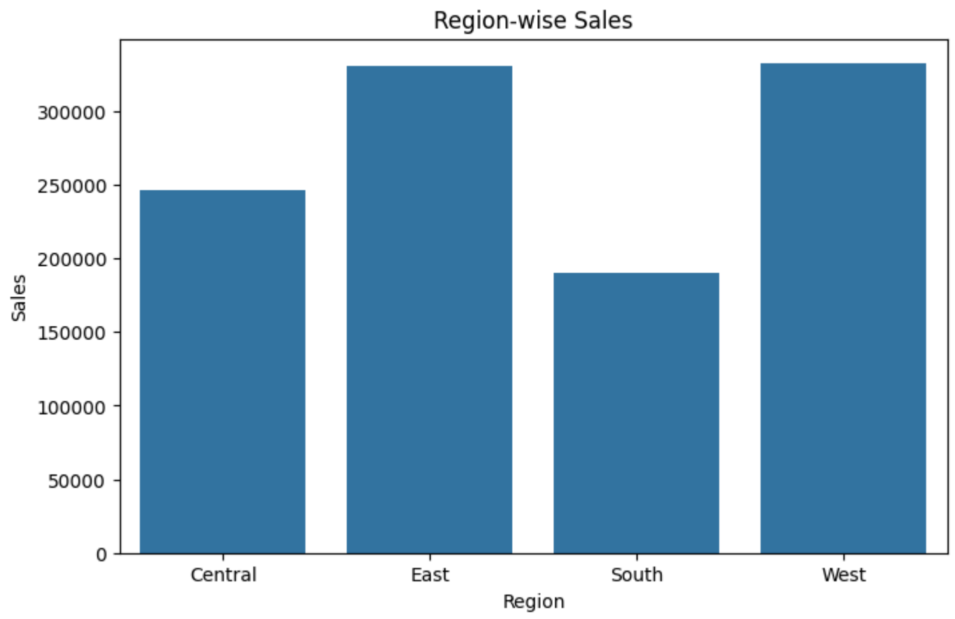


| | Row_ID | Postal_Code | Sales | Quantity | Discount | Profit |
|---|---|---|---|---|---|---|
| Row_ID | 1.00 | 0.01 | 0.01 | 0.01 | 0.00 | 0.02 |
| Postal_Code | 0.01 | 1.00 | -0.04 | 0.01 | 0.05 | -0.05 |
| Sales | 0.01 | -0.04 | 1.00 | 0.22 | -0.03 | 0.52 |
| Quantity | 0.01 | 0.01 | 0.22 | 1.00 | 0.02 | 0.07 |
| Discount | 0.00 | 0.05 | -0.03 | 0.02 | 1.00 | -0.24 |
| Profit | 0.02 | -0.05 | 0.52 | 0.07 | -0.24 | 1.00 |

```
# Profit vs Discount
plt.figure(figsize=(8,5))
sns.scatterplot(data=df,x='Discount',y='Profit')
plt.title("Profit Vs Discount")
plt.show()
```



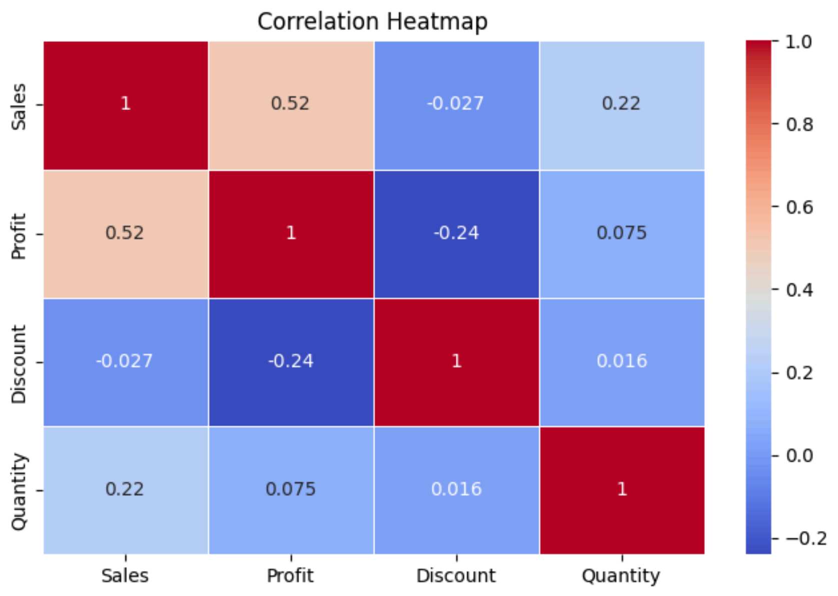
```
# Regionalwise sales
region_sales=df.groupby('Region')['Sales'].sum().reset_index()
plt.figure(figsize=(8,5))

sns.barplot(data=region_sales,x='Region',y='Sales')
plt.title('Region-wise Sales')
plt.ylabel='Total Sales'
plt.show()
```



```
# Quantity Vs Profit
plt.figure(figsize=(8,5))
sns.scatterplot(data=df,x='Profit',y='Quantity')
plt.show()
```

```
# Correlation Heatmap
# Let’s calculate and visualize the correlation between key numerical features:

# Sales

# Profit

# Discount

# Quantity
numeric_data=df[['Sales','Profit','Discount','Quantity']]

plt.figure(figsize=(8,5))
corr2=numeric_data.corr()
sns.heatmap(corr2,annot=True,cmap='coolwarm',linewidths=0.5)
plt.title('Correlation Heatmap')
plt.show
```

```
matplotlib.pyplot.show
def show(*args, **kwargs) -> None

Display all open figures.

Parameters
----------
block : bool, optional
    Whether to wait for all figures to be closed before returning.
```



|  | Sales | Profit | Discount | Quantity |
|---|---|---|---|---|
| Sales | 1 | 0.52 | -0.027 | 0.22 |
| Profit | 0.52 | 1 | -0.24 | 0.075 |
| Discount | -0.027 | -0.24 | 1 | 0.016 |
| Quantity | 0.22 | 0.075 | 0.016 | 1 |