An app error occurred. Try re-running the app, or see the error detail and contact the app owner for help. Details

```
# Importing necessary libararies
import numpy as np # For numercial operations
import pandas as pd # For data manipulation
import seaborn as sns # For data visualization
import matplotlib.pyplot as plt  # For plotting
```

```
# the the dataset file is named "superstore_data.csv"
file_path = "/work/dataset_Superstore.csv"
# Read the dataset into a pandas DataFrame
data = pd.read_csv(file_path)
# Display the first few rows of the DataFrame
data
```

	Row ID int64 1 - 9994	Order ID object CA-2017-1... 0.1% CA-2017-1... 0.1% 5007 others 99.7%	Order Date object 05/09/2016 0.4% 02/09/2017 0.4% 1235 others 99.3%	Ship Date object 16/12/2015 0.4% 26/09/2017 0.3% 1332 others 99.3%	Ship Mode object Standard ... 59.7% Second Cl... 19.5% 2 others 20.8%	Customer ID obje... WB-21850 0.4% PP-18955 0.3% 791 others 99.3%	Customer Name o.. William Bro... 0.4% Paul Prost 0.3% 791 others 99.3%
0	3783	CA-2017-165204	13/11/2017	16/11/2017	Second Class	MN-17935	Michael Nguyen
1	7322	CA-2017-167626	03/09/2017	07/09/2017	Standard Class	MY-18295	Muhammed Yedwa
2	1709	CA-2017-123491	30/10/2017	05/11/2017	Standard Class	JK-15205	Jamie Kunitz
3	2586	CA-2015-121041	03/11/2015	10/11/2015	Standard Class	CS-12250	Chris Selesnick
4	356	CA-2016-138520	08/04/2016	13/04/2016	Standard Class	JL-15505	Jeremy Lonsdale
5	3742	CA-2016-137848	15/01/2016	21/01/2016	Standard Class	WB-21850	William Brown
6	7828	CA-2017-117114	31/10/2017	05/11/2017	Standard Class	CY-12745	Craig Yedwab
7	3696	CA-2016-142097	15/10/2016	20/10/2016	Standard Class	QJ-19255	Quincy Jones
8	4641	CA-2016-102162	11/09/2016	16/09/2016	Standard Class	JF-15565	Jill Fjeld
9	3673	CA-2016-104969	08/04/2016	14/04/2016	Standard Class	EH-14125	Eugene Hildebrand

9994 rows, 29 cols 10 / page Page 1 of 1000

```
# 1. Identifying and handling missing values
# Check for missing values in the dataset
missing_values = data.isnull().sum()
print("Missing values:\n", missing_values)
```

```
Missing values:
 Row ID            0
Order ID           0
Order Date         0
Ship Date          0
Ship Mode          0
Customer ID        0
Customer Name      0
Customer_no        0
Segment            0
Segment_no         0
Country            0
City               0
State              0
State_no           0
Postal Code        0
Region             0
Region_no          0
Product ID         0
Category           0
Category_no        0
Sub-Category       0
Sub-Category_no    0
Product Name       0
Product Name_no    0
Sales              0
Quantity           0
Discount           0
Profit             0
Returned           0
```

```
# 2. Checking and handling data entry errors
# check data types of columns
print("Data types:\n", data.dtypes)
```

```
Data types:
 Row ID              int64
Order ID           object
Order Date         object
Ship Date          object
Ship Mode          object
Customer ID        object
Customer Name      object
Customer_no         int64
Segment            object
Segment_no          int64
Country            object
City               object
State              object
State_no            int64
Postal Code         int64
Region             object
Region_no           int64
Product ID         object
Category           object
Category_no         int64
Sub-Category       object
Sub-Category_no     int64
Product Name       object
Product Name_no     int64
Sales             float64
Quantity            int64
Discount          float64
Profit            float64
Returned             bool
```

```
# If there are any incorrect data types or Inconsistent formatting, convert them to the correct format
# For example, convert date columns to datetime format
# Convert 'Order Date' and 'Ship Date' to Datetime
data[ 'Order Date'] = pd.to_datetime(data['Order Date'])
data['Ship Date'] = pd.to_datetime(data['Ship Date'])
```

```
/tmp/ipykernel_135/1663158704.py:4: UserWarning: Parsing dates in %d/%m/%Y format when dayfirst=False (the default) was specified
  data[ 'Order Date'] = pd.to_datetime(data['Order Date'])
/tmp/ipykernel_135/1663158704.py:5: UserWarning: Parsing dates in %d/%m/%Y format when dayfirst=False (the default) was specified
  data['Ship Date'] = pd.to_datetime(data['Ship Date'])
```

```
columns_to_drop=["Customer_no","Segment_no","State_no","Region_no","Category_no","Sub-Category_no","Product Name_no"]
#created a variable of all the columns that are not required
```

```
data_clean=data.drop(columns=columns_to_drop)
#deleting all the unrequired columns
```

```
data_clean[['Sales','Quantity','Discount','Profit']].describe()
```

	Sales float64	**Quantity** float64	**Discount** float64	**Profit** float64
coun	9994	9994	9994	9994
mea	229.8580008	6.789673804	0.1564028417	28.65689631
std	623.2451005	173.1883741	0.2070267139	234.2601077
min	0.444	1	0	-6599.978
25%	17.28	2	0	1.72875
50%	54.49	3	0.2	8.6665
75%	209.94	5	0.2	29.364
max	22638.48	10000	1.3	8399.976

8 rows, 4 cols 10 / page « ‹ Page 1 of 1 › »

```
data_clean[["Sales","Profit"]].skew()
```

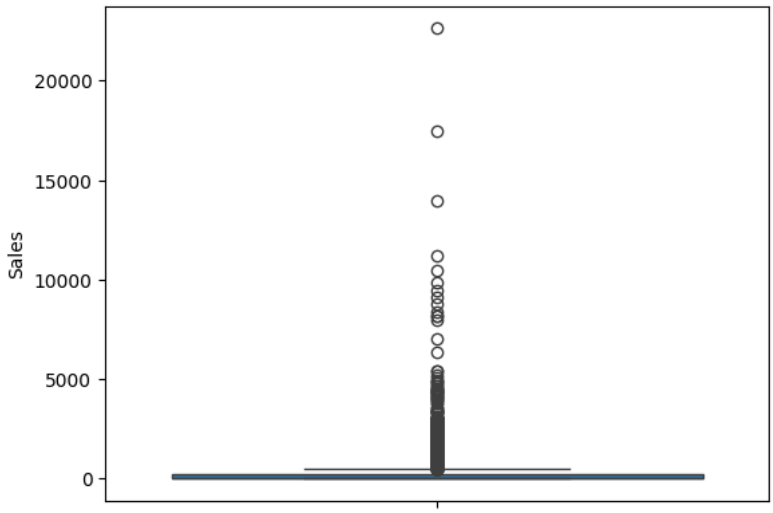
```
Sales     12.972752
Profit     7.561432
dtype: float64
```

```
data_clean[["Sales","Profit"]].kurtosis()
```

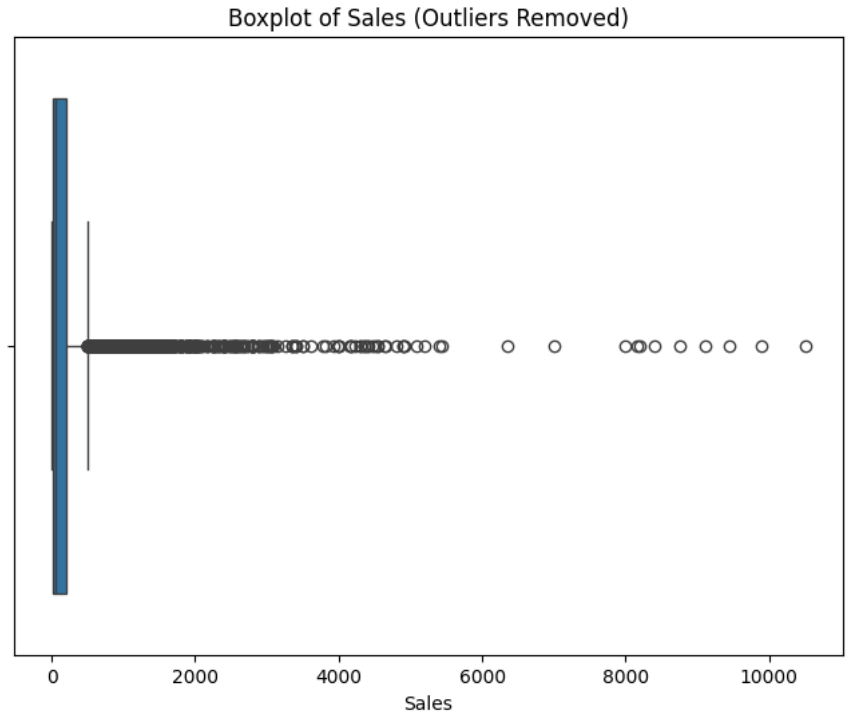
```
Sales     305.311753
Profit    397.188515
dtype: float64
```

```
# Plot a boxplot to visualize the distribution of sales
sns.boxplot(data["Sales"])
```

```
<AxesSubplot: ylabel='Sales'>
```




```
# Remove outliers from the 'Sales' column where sales are greater than 11000
data = data[data['Sales'] <= 11000]

# Create a boxplot to visualise the distribution of sales after removing outliers
plt.figure(figsize=(8, 6))
sns.boxplot(x=data['Sales'])
plt.title('Boxplot of Sales (Outliers Removed)')
plt.xlabel('Sales')
plt.show()
```

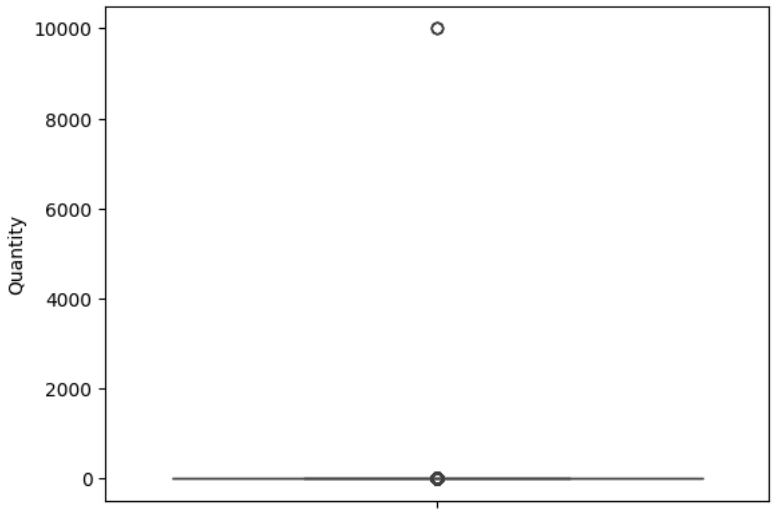


```
data.shape #check how many rows are removed
```

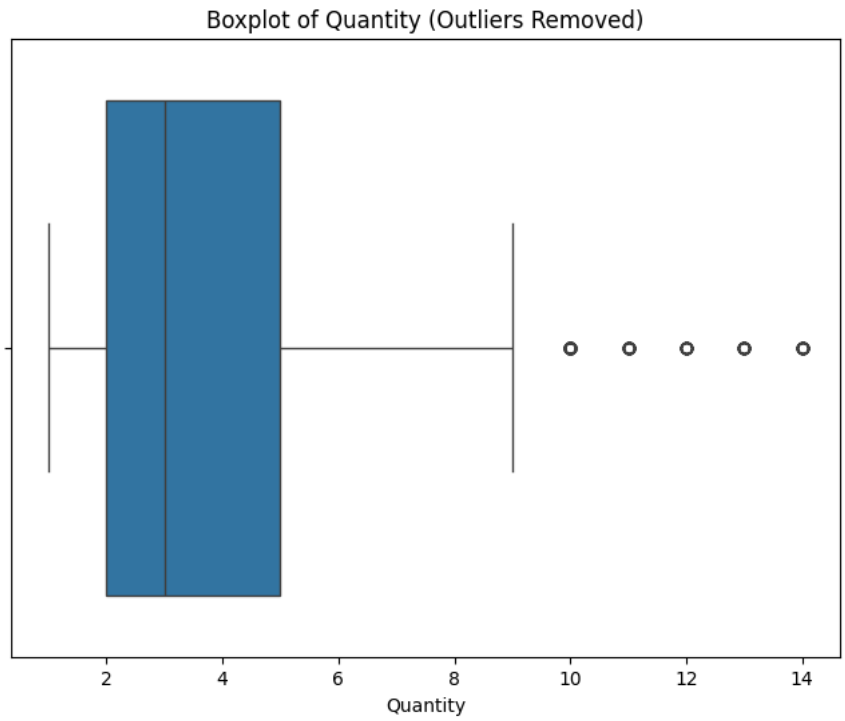
```
(9990, 29)
```

```
# Create a boxplot to visualize the distribution of Quantity
sns.boxplot(data["Quantity"])
```

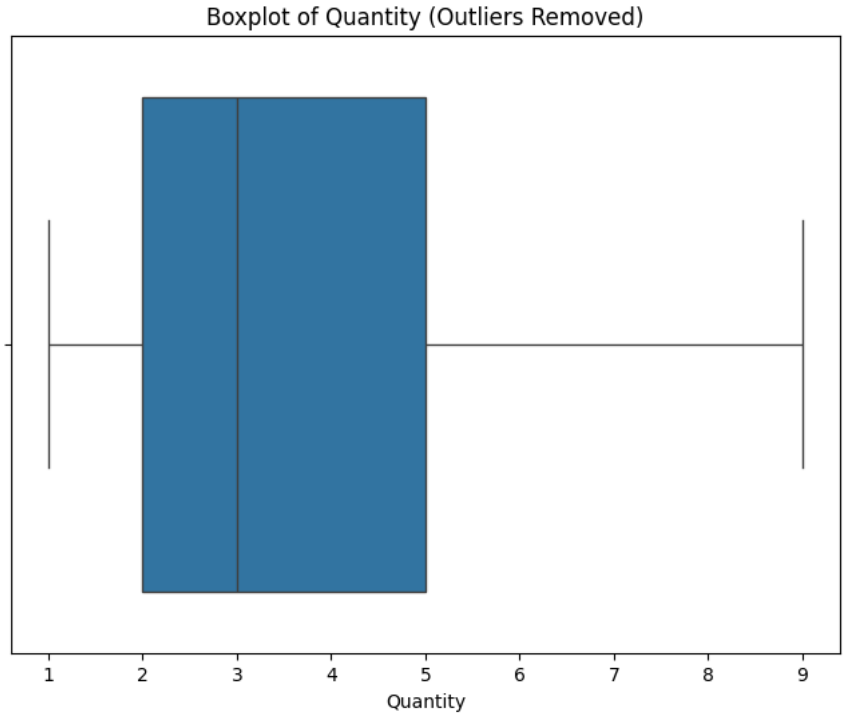
```
<AxesSubplot: ylabel='Quantity'>
```



```
# Filter the dataset to remove outliers in the 'Quantity' column
data= data[data['Quantity'] <= 1000]

# Create a boxplot to visualize the distribution of Quantity after removing outliers
plt.figure(figsize=(8, 6))
sns.boxplot(x=data['Quantity'])
plt.title('Boxplot of Quantity (Outliers Removed)')
plt.xlabel('Quantity')
plt.show()
```

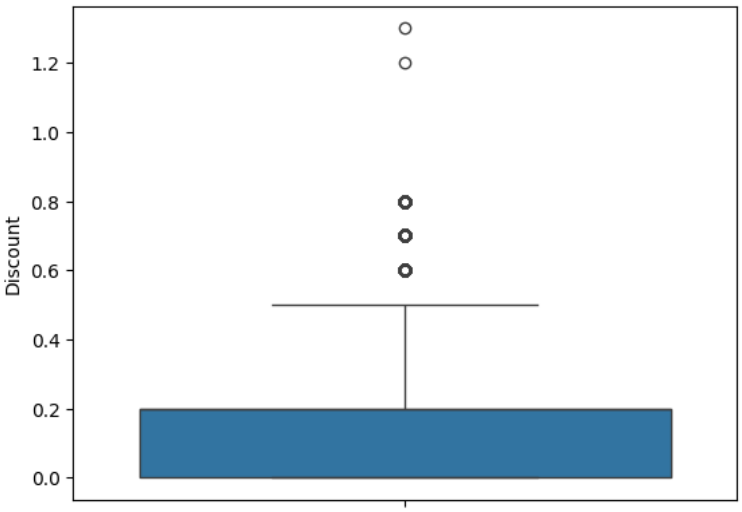


```
# Filter the data to remove outliers above 10 in the 'Quantity' column
data = data[data['Quantity'] <= 9]

# Create the boxplot
plt.figure(figsize=(8, 6))
sns.boxplot(x=data['Quantity'])
plt.title('Boxplot of Quantity (Outliers Removed)')
plt.xlabel('Quantity')
plt.show()
```

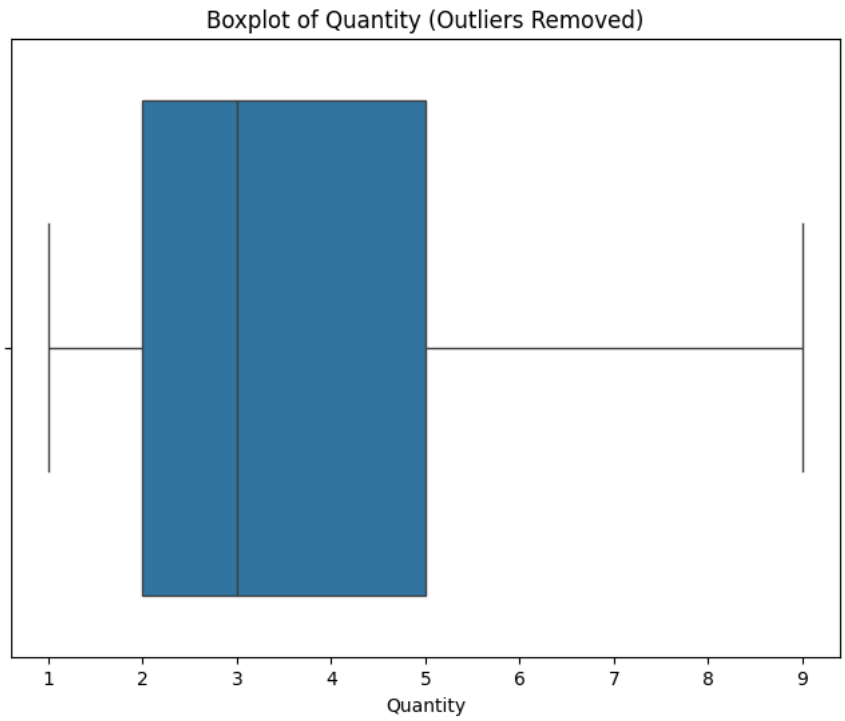



```
sns.boxplot(data["Discount"]) # Create a boxplot to visalize the distribution of Discount
```

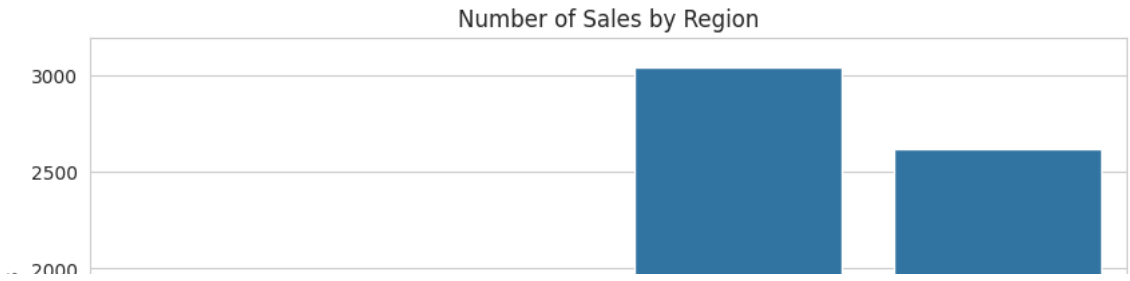
```
<AxesSubplot: ylabel='Discount'>
```

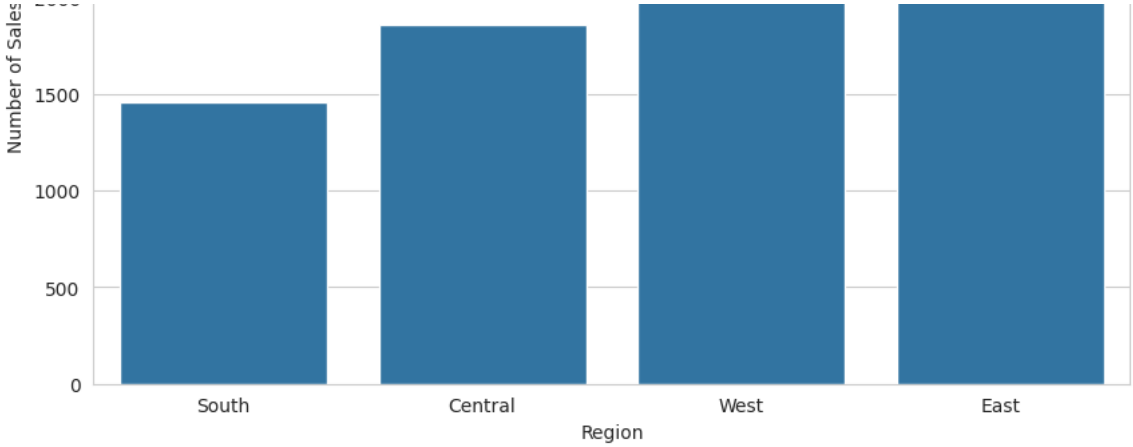


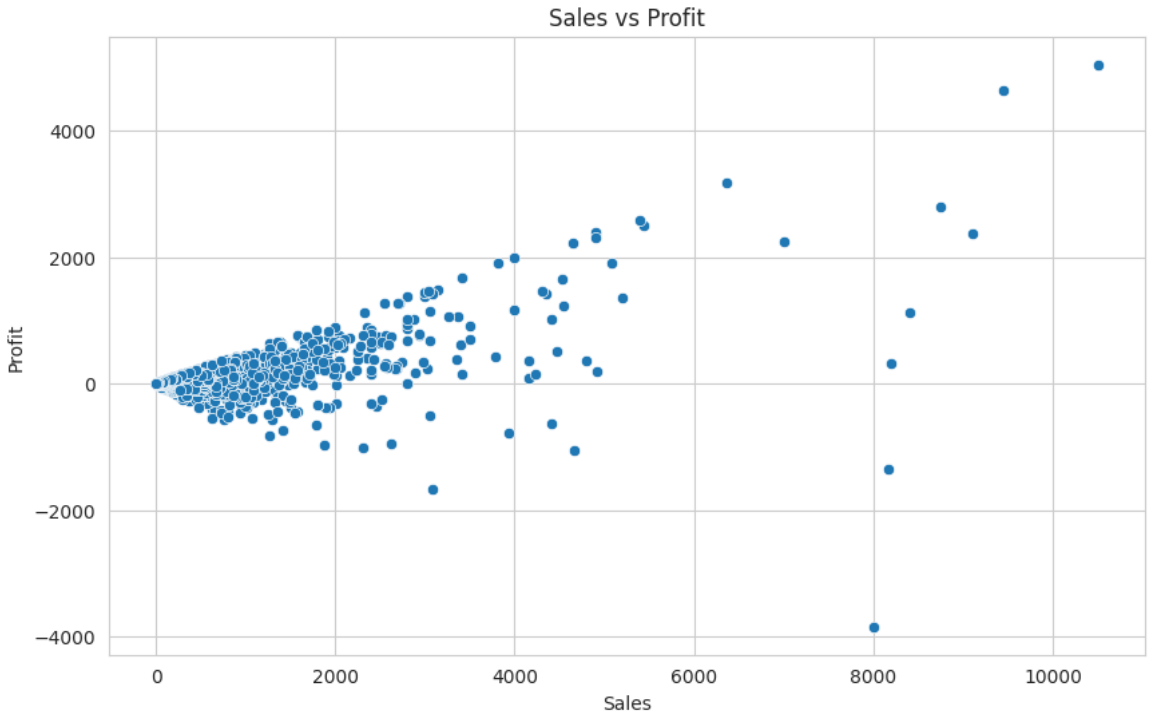
```
# Remove outliers from the 'Discount' column where the discount value is greater than 0.5
data = data[data['Discount'] <= 0.5]

# Create a boxplot to visualize the distribution of Quantity after removing outliers
plt.figure(figsize=(8, 6))
sns.boxplot(x=data['Quantity'])
plt.title('Boxplot of Quantity (Outliers Removed)')
plt.xlabel('Quantity')
plt.show()
```



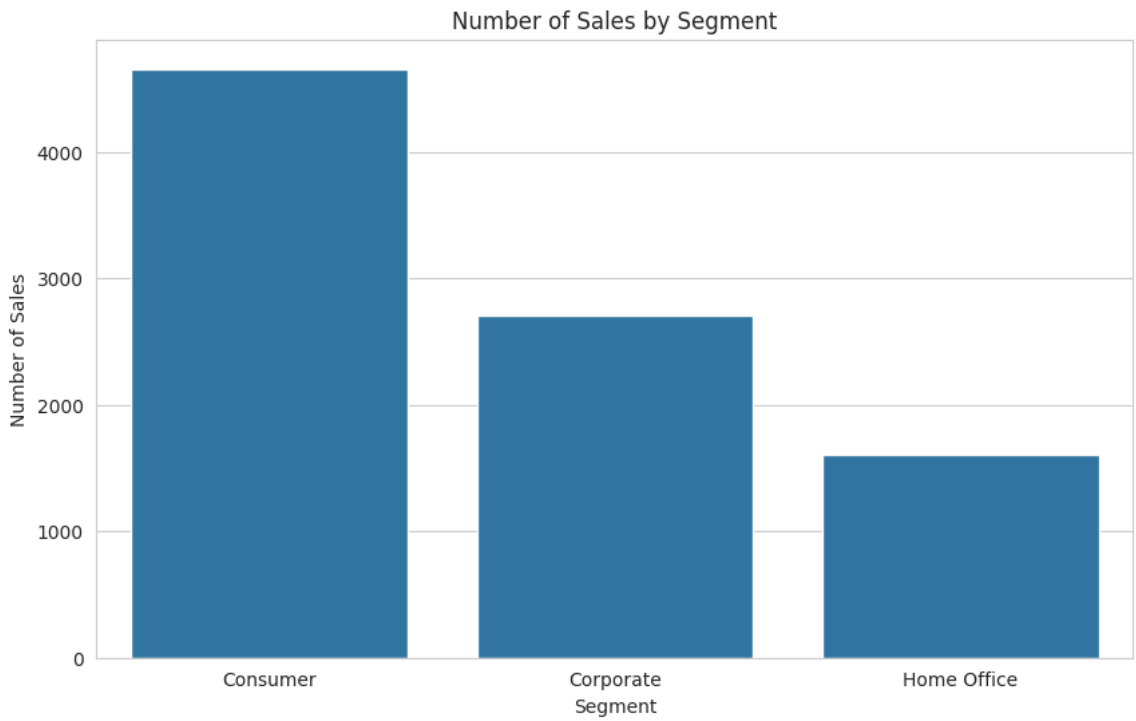

```
import seaborn as sns
import matplotlib.pyplot as plt

# Set the style of the plots
sns.set_style("whitegrid")

#Categorial (Bar chart) plot for the 'Region' column
plt.figure(figsize=(10, 6))
sns.countplot(x='Region', data=data)
plt.title('Number of Sales by Region')
plt.xlabel('Region')
plt.ylabel('Number of Sales')
plt.show()

# Continous (Scatter plot) for the 'Sales' and 'Profit' columns
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Sales', y='Profit', data=data)
plt.title('Sales vs Profit')
plt.xlabel('Sales')
plt.ylabel('Profit')
plt.show()
```



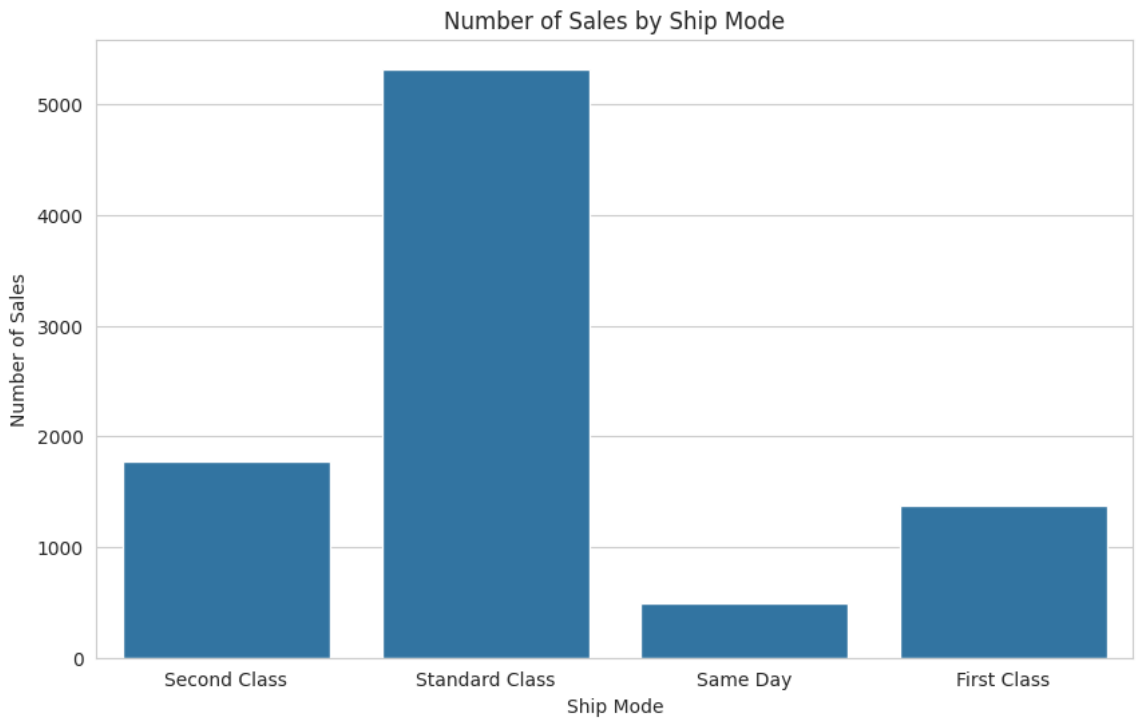

Number of Sales
1500
1000
500
0
South
Central
West
East
Region


Sales vs Profit
Profit
4000
2000
0
−2000
−4000
0
2000
4000
6000
8000
10000
Sales

```
# Categorical (Bar chart) plot for the 'Segment' column
plt.figure(figsize=(10, 6))
sns.countplot(x='Segment', data=data)
plt.title('Number of Sales by Segment')
plt.xlabel('Segment')
plt.ylabel('Number of Sales')
plt.show()
```



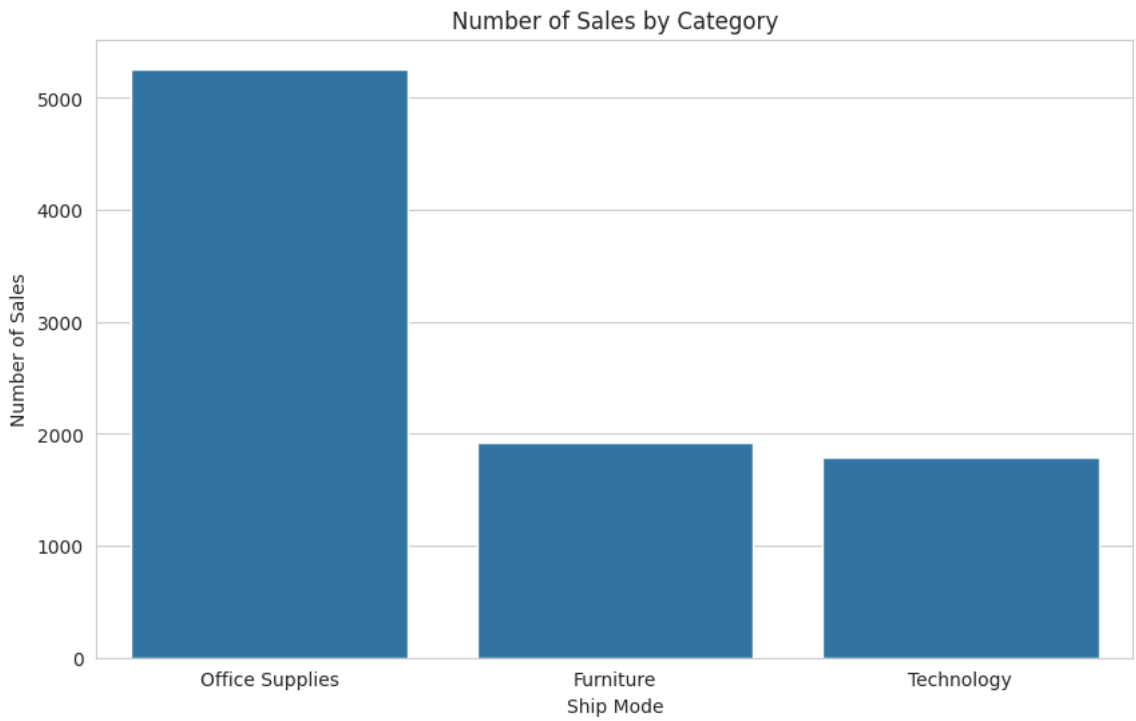
```
# Categorial (Bar chart) plot for the 'Ship Mode' column
plt.figure(figsize=(10, 6))
sns.countplot(x='Ship Mode', data=data)
plt.title('Number of Sales by Ship Mode')
plt.xlabel('Ship Mode')
plt.ylabel('Number of Sales')
plt.show()
```



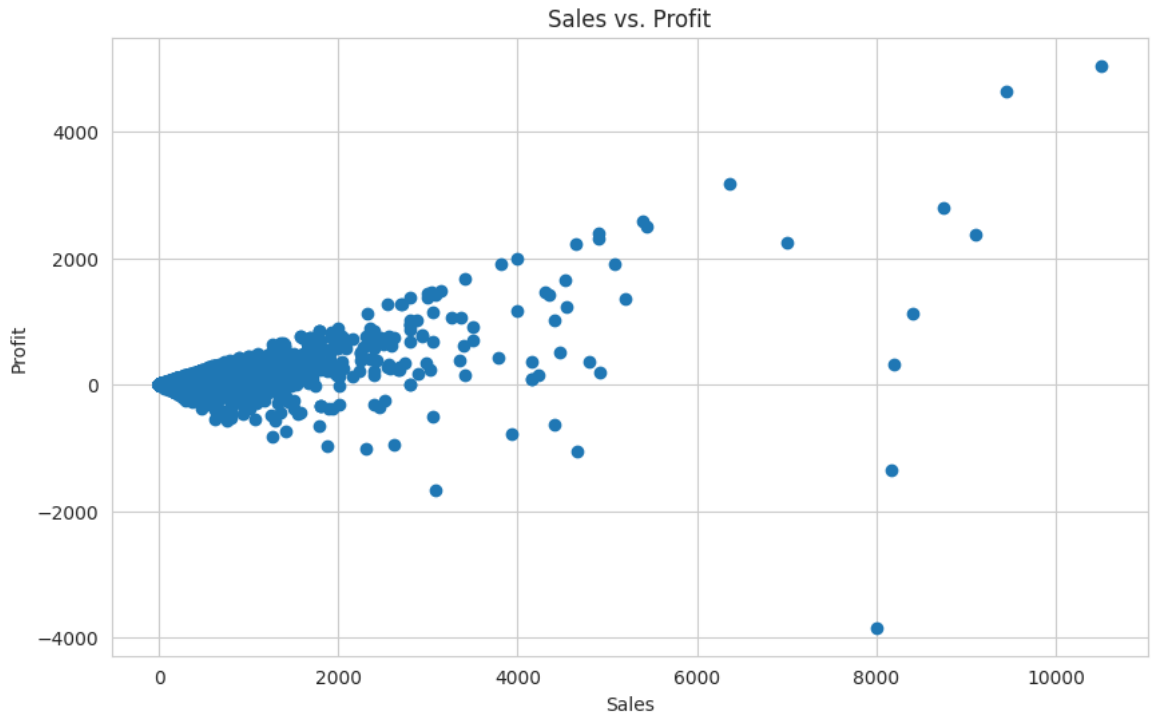

```
data["Category"].value_counts()
```

```
Category
Office Supplies    5260
Furniture          1927
Technology         1784
Name: count, dtype: int64
```

```
# Categorial (Bar chart) plot for the 'Category' column
plt.figure(figsize=(10, 6))
sns.countplot(x='Category', data=data)
plt.title('Number of Sales by Category')
plt.xlabel('Ship Mode')
plt.ylabel('Number of Sales')
plt.show()
```



```
# Continous (Scatter plot) for Sales vs. Profit
plt.figure(figsize=(10, 6))
plt.scatter(data['Sales'], data['Profit'])
plt.title('Sales vs. Profit')
plt.xlabel('Sales')
plt.ylabel('Profit')
plt.show()
```




```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 8971 entries, 0 to 9993
Data columns (total 29 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Row ID           8971 non-null   int64
 1   Order ID         8971 non-null   object
 2   Order Date       8971 non-null   datetime64[ns]
 3   Ship Date        8971 non-null   datetime64[ns]
 4   Ship Mode        8971 non-null   object
 5   Customer ID      8971 non-null   object
 6   Customer Name    8971 non-null   object
 7   Customer_no      8971 non-null   int64
 8   Segment          8971 non-null   object
 9   Segment_no       8971 non-null   int64
 10  Country          8971 non-null   object
 11  City             8971 non-null   object
 12  State            8971 non-null   object
 13  State_no         8971 non-null   int64
 14  Postal Code      8971 non-null   int64
 15  Region           8971 non-null   object
 16  Region_no        8971 non-null   int64
 17  Product ID       8971 non-null   object
 18  Category         8971 non-null   object
 19  Category_no      8971 non-null   int64
 20  Sub-Category     8971 non-null   object
 21  Sub-Category_no  8971 non-null   int64
 22  Product Name     8971 non-null   object
 23  Product Name_no  8971 non-null   int64
 24  Sales            8971 non-null   float64
```

```
# Get the value counts for the 'Ship Mode' column
data["Ship Mode"].value_counts()
```

```
Ship Mode
Standard Class    5322
Second Class      1780
First Class       1375
Same Day           494
Name: count, dtype: int64
```

```
# Get the value counts for the 'Segment' column
data["Segment"].value_counts()
```

```
Segment
Consumer       4658
Corporate      2704
Home Office    1609
Name: count, dtype: int64
```

```
# Get the value counts for the 'Country' column
data["Country"].value_counts()
```

```
Country
United States    8971
Name: count, dtype: int64
```

```
# Get the value counts for the 'City' column
data["City"].value_counts()
```

```
City
New York City     896
Los Angeles       736
San Francisco     498
Philadelphia      433
Seattle           418
                 ...
Palatine            1
Altoona             1
Cuyahoga Falls      1
Jefferson City      1
Antioch             1
Name: count, Length: 519, dtype: int64
```

```
# Get the value counts for the 'State' column
data["State"].value_counts()
```

```
State
California              1963
New York                1103
Texas                    694
Washington               494
Pennsylvania             472
Ohio                     384
Illinois                 331
Florida                  309
Michigan                 252
Virginia                 218
North Carolina           207
Georgia                  182
Arizona                  180
Tennessee                153
Indiana                  145
Colorado                 141
Kentucky                 135
Massachusetts            132
New Jersey               130
Wisconsin                108
Maryland                 104
Oregon                   101
Delaware                  96
Minnesota                 87
Connecticut               80
Missouri                  65
Oklahoma                  65
Alabama                   60
Arkansas                  60
Rhode Island              56
Mississippi               52
```

```
data["Category"].value_counts()
```

```
Category
Office Supplies    5260
Furniture          1927
Technology         1784
Name: count, dtype: int64
```

```
data["Sub-Category"].value_counts()
```

```
Sub-Category
Paper          1349
Binders         890
Phones          874
Storage         832
Furnishings     803
Art             787
Accessories     757
Chairs          605
Appliances      393
Labels          356
Tables          311
Envelopes       253
Fasteners       211
Bookcases       208
Supplies        189
Machines         88
Copiers          65
Name: count, dtype: int64
```

```
data["Sub-Category_no"].value_counts()
```

```
Sub-Category_no
1     1349
7      890
9      874
6      832
5      803
4      787
13     757
16     605
3      393
12     356
14     311
2      253
10     211
15     208
11     189
8       88
17      65
Name: count, dtype: int64
```

```
data["Product Name"].value_counts()
```

```
Product Name
Staple envelope                                                     48
Easy-staple paper                                                   46
Staples                                                             43
KI Adjustable-Height Table                                          18
Staple remover                                                      18
                                                                    ..
Snap-A-Way Black Print Carbonless Speed Message, No Reply Area, Duplicate     1
Socket Bluetooth Cordless Hand Scanner (CHS)                        1
GBC VeloBinder Strips                                               1
Sony 8GB Class 10 Micro SDHC R40 Memory Card                        1
Belkin 8 Outlet SurgeMaster II Gold Surge Protector with Phone Protection     1
Name: count, Length: 1833, dtype: int64
```

```
data["Product Name"].value_counts()
```

```
Product Name
Staple envelope                                                     48
Easy-staple paper                                                   46
Staples                                                             43
KI Adjustable-Height Table                                          18
Staple remover                                                      18
                                                                    ..
Snap-A-Way Black Print Carbonless Speed Message, No Reply Area, Duplicate     1
Socket Bluetooth Cordless Hand Scanner (CHS)                        1
GBC VeloBinder Strips                                               1
Sony 8GB Class 10 Micro SDHC R40 Memory Card                        1
Belkin 8 Outlet SurgeMaster II Gold Surge Protector with Phone Protection     1
Name: count, Length: 1833, dtype: int64
```

```
data.shape
```

```
(8971, 29)
```

```
# Replace the categorical values in the 'Ship Mode' column with numercial values
data=data.replace(["Standard Class","Second Class","First Class","Same Day"],[0,1,2,3])
```

```
data=data.replace(["Consumer","Corporate","Home Office"],[0,1,2])
```

```
data=data.replace(["Louisiana" , "California","New York","Texas","Washington","Pennsylvania","Ohio","Illinois","Florida",
```

```
data=data.replace(["Office Supplies","Furniture","Technology"],[0,1,2])
```

```
data=data.replace(["Paper","Binders","Phones","Storage","Furnishings","Art","Accessories","Chairs","Appliances","Labels",
```

```
data=data.replace(["West","East","Central","South"],[0,1,2,3])
```

```
data  #view ultered data
```

	Row ID int64 1 - 9994	Order ID object CA-2017-1... 0.1% CA-2017-1... 0.1% 4707 others 99.7%	Order Date dateti... 2014-01-03 00:0...	Ship Date dateti... 2014-01-07 00:0...	Ship Mode int64 0 - 3	Customer ID obje... JL-15835 0.4% WB-21850 0.4% 790 others 99.3%	Customer Name o.. John Lee 0.4% William Bro... 0.4% 790 others 99.3%	C 1
0	3783	CA-2017-165204	2017-11-13 00:00:0	2017-11-16 00:00:0	1	MN-17935	Michael Nguyen	
1	7322	CA-2017-167626	2017-09-03 00:00:	2017-09-07 00:00:	0	MY-18295	Muhammed Yedwa	
2	1709	CA-2017-123491	2017-10-30 00:00:	2017-11-05 00:00:	0	JK-15205	Jamie Kunitz	
3	2586	CA-2015-121041	2015-11-03 00:00:	2015-11-10 00:00:0	0	CS-12250	Chris Selesnick	
4	356	CA-2016-138520	2016-04-08 00:00:	2016-04-13 00:00:	0	JL-15505	Jeremy Lonsdale	
5	3742	CA-2016-137848	2016-01-15 00:00:	2016-01-21 00:00:	0	WB-21850	William Brown	
6	7828	CA-2017-117114	2017-10-31 00:00:	2017-11-05 00:00:	0	CY-12745	Craig Yedwab	
7	3696	CA-2016-142097	2016-10-15 00:00:	2016-10-20 00:00:	0	QJ-19255	Quincy Jones	
8	4641	CA-2016-102162	2016-09-11 00:00:	2016-09-16 00:00:	0	JF-15565	Jill Fjeld	
9	3673	CA-2016-104969	2016-04-08 00:00:	2016-04-14 00:00:	0	EH-14125	Eugene Hildebrand	

8971 rows, 29 cols 10 / page « ‹ Page 1 of 898 › »

```
features = ['Ship Mode','Segment','State',"Region","Category","Sales","Quantity","Discount","Profit"]
```

```
# Importing necessary libraries for model training and evaluation
from sklearn.model_selection import train_test_split # Importing train_test_split function for splitting the dataset
from sklearn.linear_model import LinearRegression # Importing LinearRegression class for creating a linear regression mod
from sklearn.metrics import mean_squared_error, r2_score # Importing evaluation metrics for model performance

# The code above imports the required modules from scikit-learn to perform linear regression analysis.
# These modules will be used to split the dataset into training and testing sets, create and train the linear regression

# Further code implementation will involve loading the dataset, preprocessing (if necessary), splitting the dataset into
# Following that, a LinearRegression object will be instantiated, trained using the training data, and evaluated on the t
# using mean squared error and R-squared metrics.
```

```
# Prepare the data
# Select relevant features for modeling (excluding non-numeric or irrelevant columns)

X = data[features]
Y = data['Profit']
```

```
X # data feature stored in X
```

	Ship Mode int64 0 - 3	Segment int64 0 - 2	State int64 0 - 48	Region int64 0 - 3	Category int64 0 - 2	Sales float64 0.99 - 10499.97	Quantity int64 1 - 9	D 0
0	1	0	14	3	0	8.904	3	
1	0	1	7	2	0	8.904	3	
2	0	0	1	0	0	7.42	2	
3	0	1	3	2	0	6.608	2	
4	0	0	2	1	0	8.26	2	
5	0	0	2	1	0	16.52	4	
6	0	1	7	2	0	9.912	3	
7	0	1	10	3	0	66.54	6	
8	0	0	10	3	0	11.09	1	
9	0	2	6	1	0	8.872	1	

8971 rows, 9 cols 10 / page « ‹ Page 1 of 898 › »

```
Y #profit column is assign to y
```

```
0       3.3390
1       3.3390
2       3.7100
3       2.1476
4       3.7996
         ...
9983    0.9048
9984    7.6440
9988    3.0576
9992    3.6192
9993    3.0576
Name: Profit, Length: 8971, dtype: float64
```

```
# Split the data into training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
```

```
# Train the linear regression model
model = LinearRegression()  # Create an instance of the LinearRegression model
model.fit(X_train, Y_train)  # Train the model on the training data (X_train: features, Y_train: target variable)

# make predictions on the testing set
Y_pred = model.predict(X_test) # Use the trained model to make predictions on the testing data (X_test: features)

# Evaluate the model
mse = mean_squared_error(Y_test, Y_pred) # Calculate the mean squared error between the actual target values (Y_test) and
r2 = r2_score(Y_test, Y_pred) # Calculate the R-squared value to measure the goodness of fit of the model
print("Mean Squared Error:",mse) # Print the mean squared error
print("R-Squared:", r2) # Print the R-squared value
```

```
Mean Squared Error: 7.801326149133927e-27
R-Squared: 1.0
```

```
# Get the coefficients of the linear regression model
coefficients = pd.DataFrame({'Feature': features, 'Coefficient': model.coef_})
print(coefficients)
```

```
     Feature   Coefficient
0  Ship Mode -2.029883e-16
1    Segment -2.983724e-15
2      State -1.666202e-15
3     Region  2.855111e-16
4   Category -1.417808e-15
5      Sales -2.775558e-17
6   Quantity -2.161899e-16
7   Discount -5.317598e-14
8     Profit  1.000000e+00
```

```
data=data.drop(columns=["Row ID","Order ID","Order Date","Ship Date","Customer ID","Customer Name","Country","City","Post
```

```
data
```

	Ship Mode int64 0 - 3	Customer_no int... 1 - 793	Segment int64 0 - 2	Segment_no int64 1 - 3	State int64 0 - 48	State_no int64 1 - 49	Region int64 0 - 3	R 1
0	1	1	0	1	14	41	3	
1	0	2	1	2	7	12	2	
2	0	3	0	1	1	4	0	
3	0	4	1	2	3	42	2	
4	0	5	0	1	2	31	1	
5	0	6	0	1	2	31	1	
6	0	7	1	2	7	12	2	
7	0	8	1	2	10	45	3	
8	0	9	0	1	10	45	3	
9	0	10	2	3	6	34	1	

8971 rows, 20 cols 10 / page Page 1 of 898

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

X = data.iloc[:,:-1]
```

```
features=["Ship Mode","Customer_no","Segment","Segment_no","State","State_no","Region","Region_no","Category","Category_n
```

```
y = data['Profit']
```

```
# Split the data into training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)

# Train the linear regression model
model = LinearRegression()
model.fit(X_train, Y_train)

# Make predictions on the testing set
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(Y_test, Y_pred)
r2 = r2_score(Y_test, Y_pred)
print("Mean Squared Error:",mse)
print("R-Squared:",r2)

# Get the coefficients of the linear regression model
coefficients = pd.DataFrame({'Feature': features, 'Coefficient': model.coef_})
print(coefficients)
```

```
import pandas as pd
from sklearn.ensemble import GradientBoostingRegressor

# Assuming 'features' is a list containing the names of the features
features = X.columns.tolist()

# Train the Gradient Boosting Regressor model
model = GradientBoostingRegressor(n_estimators=500)
model.fit(X_train, Y_train)

# Get feature importance
feature_importance = pd.DataFrame({'Feature': features, 'Importance': model.feature_importances_})

# Sort feature importance in descending order
feature_importance = feature_importance.sort_values(by='Importance', ascending=False)

# Print feature importance
print(feature_importance)
```

→

Introduction: Businesses can make data-driven decisions by using data analysis to glean valuable insights from unprocessed data (Provost & Fawcett, 2013). Large dataset analysis necessitates methodical processing, such as data transformation, cleansing, and visualization. This report uses Python modules like Pandas, NumPy and Seaborn, to analyse a retail dataset called superstore_data.csv in an organized manner. This report's main objectives are to address missing numbers, fix data entry mistakes, and make sure the dataset is in an appropriate format for additional research.

Methodology :Bringing in Required Libraries -Numerous libraries are available in Python for effective data analysis. Seaborn is used for visualization, Pandas is used for data processing, and NumPy is used for numerical calculations (McKinney, 2017).

The code sample that follows shows how to import these libraries:1. Import pandas as pd import NumPy as np.2. Loading the Dataset import matplotlib.pyplot `read_csv()` function is used to import the dataset, guaranteeing that it is tabularly structured (Wickham & Grolemund, 2017). The path to the dataset file is giv pd.read_csv data.head()3. Finding and Dealing with Missing Values Data analysis accuracy may be impacted by missing values.
The `isnull().sum()` function is used to check the dataset for missing values: missing_values = data.isnull()."Missing values:," missing_values, total() print()4. Exa Inaccurate data types or uneven formatting might lead to data entry mistakes. Pandas' `dtypes` function enables column data type inspection:
Data types:, data.dtypes, print()5. Analysis of Exploratory Data (EDA) To comprehend data distribution and spot trends, exploratory data analysis is crucial (Tuk A variety of visualization methods are employed, including box plots and histograms: data['Sales'], bins=30, kde=True; sns.histplot plt.show()
Additionally, a correlation heatmap is created to examine the connections between numerical variables: plt.figure(figsize=(10,6))

Results and Discussion –Missing Values :Some columns in the dataset have missing values that call for suitable handling techniques.
Certain columns, especially those pertaining to dates, need conversion to datetime format in order to maintain data type consistency. –

Trends and Insights :Exploratory Data Analysis also known as EDA identified trend in Sales Data, showing outliers in certain variables and a normal distribution in others.

Points Regarding The Features Used :· Sub-category – It refers to the products which are more involved to determine profit.· Customer_no: It is more used as a customer identifier as it helps in getting to know about the customers who are more involved in the profitability.· Quantity: It refers to the items being sold in every transaction as this also helps in determining the profit.· Category : It refers to the vast category of the products which also helps in determining the profit by telling which product has been the most influential in achieving profit.

Conclusion This analysis emphasizes how crucial preparation and data cleaning are to guaranteeing dataset quality.Resolving formatting errors and missing nur of conclusions derived from the data.The results of Linear Equation Model were also very different as the MSE is very less indicating a very small error between predicted values which indicates that it refers to the data directly. The R-Squared value also indicates that it is a perfect fit of the model for the data. Thirdly, th close to 0 except for the profit indicating that the model is fully dependent on the profit to determine the profit.

ReferencesLittle, R.J. & Rubin, D.B. (2019). Statistical Analysis with Missing Data. 3rd ed. Wiley.McKinney, W. (2017). Python for Data Analysis. O'Reilly Media.Provost, F. & Fawcett, T. (2013). Data Science for Business. O'Reilly Media.Tukey, J.W. (1977). Exploratory Data Analysis. Addison-Wesley.Wickham, H. & Grolemund, G. (2017). R for Data Science. O'Reilly Media.