jupyter Ecommerce Data Analysis Last Checkpoint: 9 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

Ecommerce Data Analysis Project Using Python

In [1]:
```
# Define the heading text
heading = "The E-commerce Data Analysis Project aims to analyze and derive insights from a dataset containing i

# Print the heading with red color
print("\033[91m"+ "\033[1m" + heading + "\033[0m")
```

The E-commerce Data Analysis Project aims to analyze and derive insights from a dataset containing information about customer behavior, product sales, and other relevant metrics in the E-commerce domain. Python will be used to perform various data analysis tasks, including data cleaning, exploratory data analysis (EDA), visualization, and statistical analysis.

EDA Part - Exploratory Data Analysis

In [2]:
```
# Importing pandas library for data manipulation and analysis
import pandas as pd
```

In [3]:
```
Ecom=pd.read_excel("ECOMM DATA.xlsx")
```

In [4]:
```
# Preview the data
Ecom.head()
```

Out[4]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	City	State	...	Product ID	Catego
0	32298	CA-2012-124891	2012-07-31	2012-07-31	Same Day	RH-19495	Rick Hansen	Consumer	New York City	New York	...	TEC-AC-10003033	Technolo
1	26341	IN-2013-77878	2013-02-05	2013-02-07	Second Class	JR-16210	Justin Ritter	Corporate	Wollongong	New South Wales	...	FUR-CH-10003950	Furnitu

In [5]:
```
#check how much rows and columns
Ecom.shape
```

Out[5]: (51290, 24)

In [6]:
```
Ecom.info()
```

In [6]:
```
Ecom.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51290 entries, 0 to 51289
Data columns (total 24 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Row ID          51290 non-null  int64
 1   Order ID        51290 non-null  object
 2   Order Date      51290 non-null  datetime64[ns]
 3   Ship Date       51290 non-null  datetime64[ns]
 4   Ship Mode       51290 non-null  object
 5   Customer ID     51290 non-null  object
 6   Customer Name   51290 non-null  object
 7   Segment         51290 non-null  object
 8   City            51290 non-null  object
 9   State           51290 non-null  object
 10  Country         51290 non-null  object
 11  Postal Code     9994 non-null   float64
 12  Market          51290 non-null  object
 13  Region          51290 non-null  object
 14  Product ID      51290 non-null  object
 15  Category        51290 non-null  object
 16  Sub-Category    51290 non-null  object
 17  Product Name    51290 non-null  object
 18  Sales           51290 non-null  float64
 19  Quantity        51290 non-null  int64
 20  Discount        51290 non-null  float64
 21  Profit          51290 non-null  float64
 22  Shipping Cost   51290 non-null  float64
```

Activate Windows
Go to Settings to activa

In [7]:
```
# count observations of each column to check for missing values
Ecom.count()
```

Out[7]:
```
Row ID            51290
Order ID          51290
Order Date        51290
Ship Date         51290
Ship Mode         51290
Customer ID       51290
Customer Name     51290
Segment           51290
City              51290
State             51290
Country           51290
Postal Code        9994
Market            51290
Region            51290
Product ID        51290
Category          51290
Sub-Category      51290
Product Name      51290
Sales             51290
Quantity          51290
Discount          51290
Profit            51290
Shipping Cost     51290
Order Priority    51290
dtype: int64
```


In [8]:
```
Ecom.isnull().sum()
```

Out[8]:
```
Row ID                0
Order ID              0
Order Date            0
Ship Date             0
Ship Mode             0
Customer ID           0
Customer Name         0
Segment               0
City                  0
State                 0
Country               0
Postal Code       41296
Market                0
Region                0
Product ID            0
Category              0
Sub-Category          0
Product Name          0
Sales                 0
Quantity              0
Discount              0
Profit                0
Shipping Cost         0
Order Priority        0
dtype: int64
```


In [9]:
```
# count the number of missing values from all columns and rows
Ecom.isnull().sum().sum()
```

Out[9]:
```
41296
```

In [10]:
```
# check duplicates values
Ecom.duplicated().sum()
```

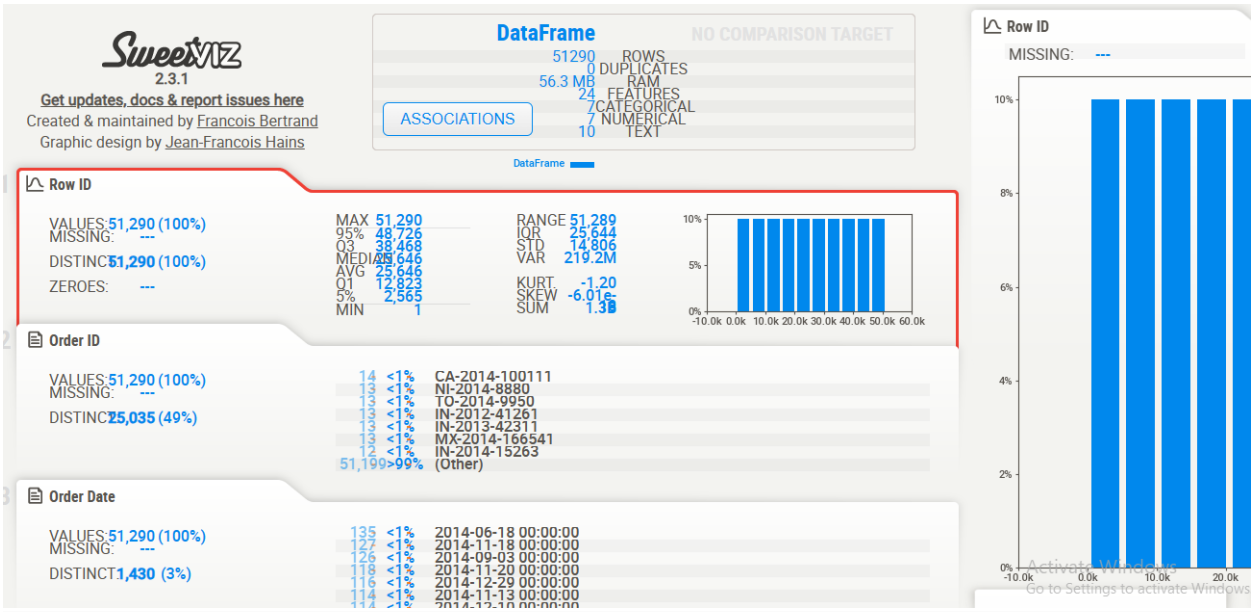
Out[10]:
```
0
```

In [11]:
```
Ecom.head(1)
```

Out[11]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	City	State	...	Product ID	Category	Sub-Category	P
0	32298	CA-2012-124891	2012-07-31	2012-07-31	Same Day	RH-19495	Rick Hansen	Consumer	New York City	New York	...	TEC-AC-10003033	Technology	Accessories	

1 rows × 24 columns

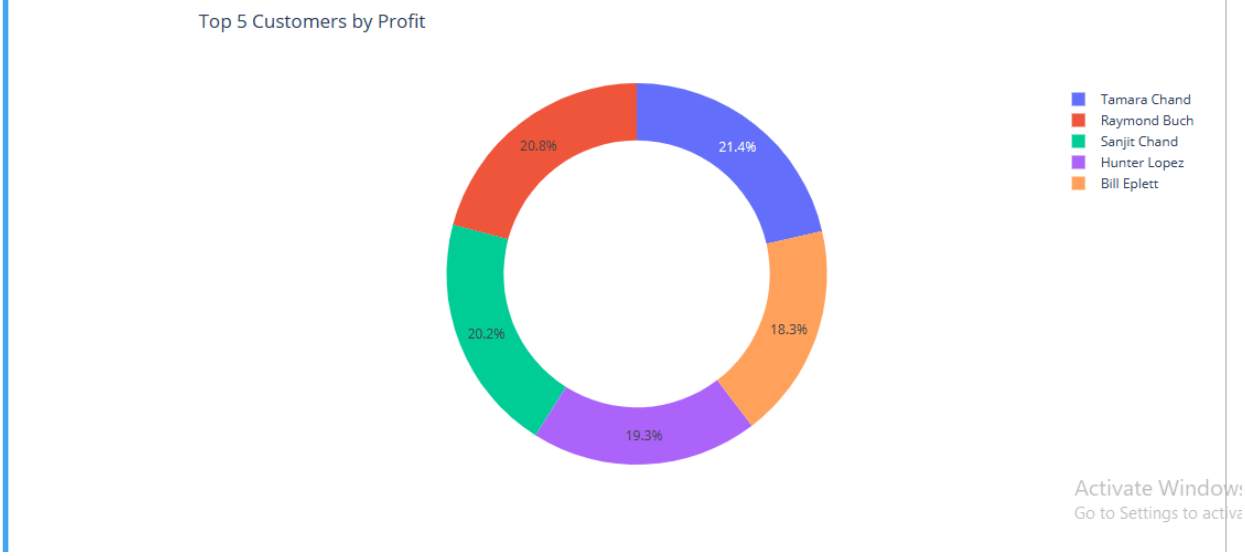


In [49]:
```
import sweetviz as sv

report=sv.analyze(Ecom)
report.show_html("Data.html")
```

```
Done! Use 'show' commands to display/save. [100%] 00:02 -> (00:00 left)

Report Data.html was generated! NOTEBOOK/COLAB USERS: the web browser MAY not pop up, regardless, the report I
S saved in your notebook/colab files.
```

In [12]:
```
customer_profit = Ecom.groupby('Customer Name')['Profit'].sum()
top_5_customers = customer_profit.sort_values(ascending=False).head(5)

# Print the top 5 customers by profit
print("Top 5 Customers by Profit:")
print(top_5_customers)
```

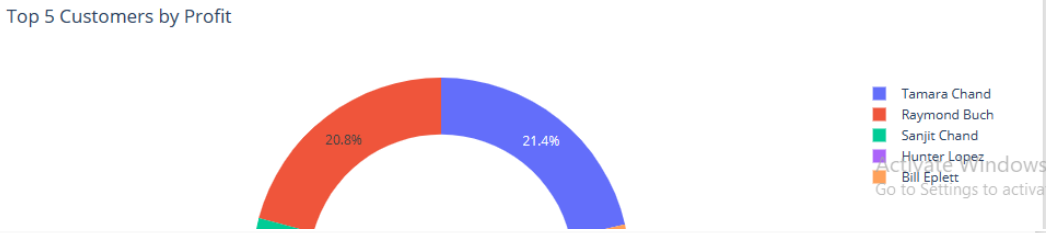
```
Top 5 Customers by Profit:
Customer Name
Tamara Chand    8672.89890
Raymond Buch    8453.04950
Sanjit Chand    8205.37990
Hunter Lopez    7816.56778
Bill Eplett     7410.00530
Name: Profit, dtype: float64
```

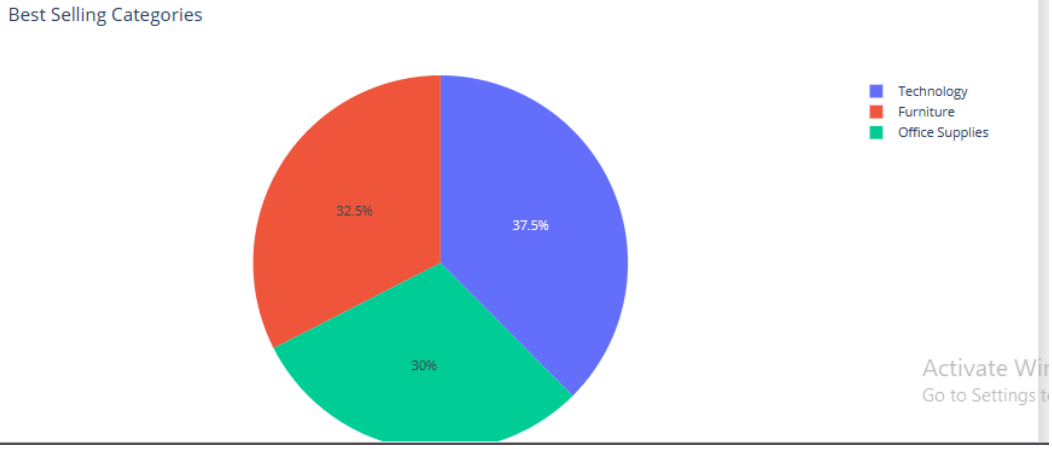
Visualization

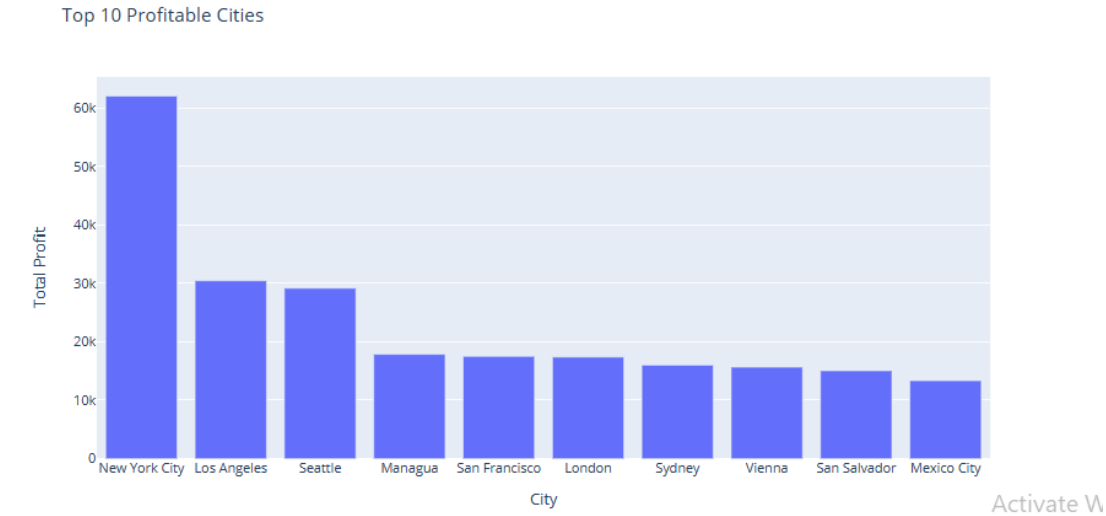
In [13]:
```
# Importing matplotlib library for data visualization
import plotly.graph_objects as go
```

In [14]:
```
# Group the DataFrame by 'Customer Name' and calculate the total profit for each customer

customer_profit = Ecom.groupby('Customer Name')['Profit'].sum()
top_5_customers = customer_profit.sort_values(ascending=False).head(5)
fig = go.Figure(data=[go.Pie(labels=top_5_customers.index, values=top_5_customers, hole=0.7)])
fig.update_layout(title='Top 5 Customers by Profit')
fig.show()
```

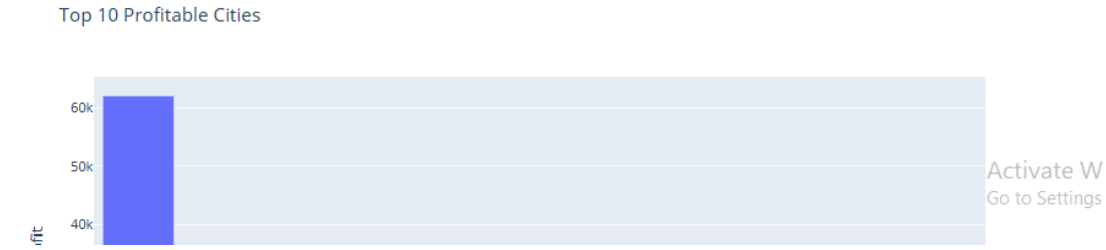


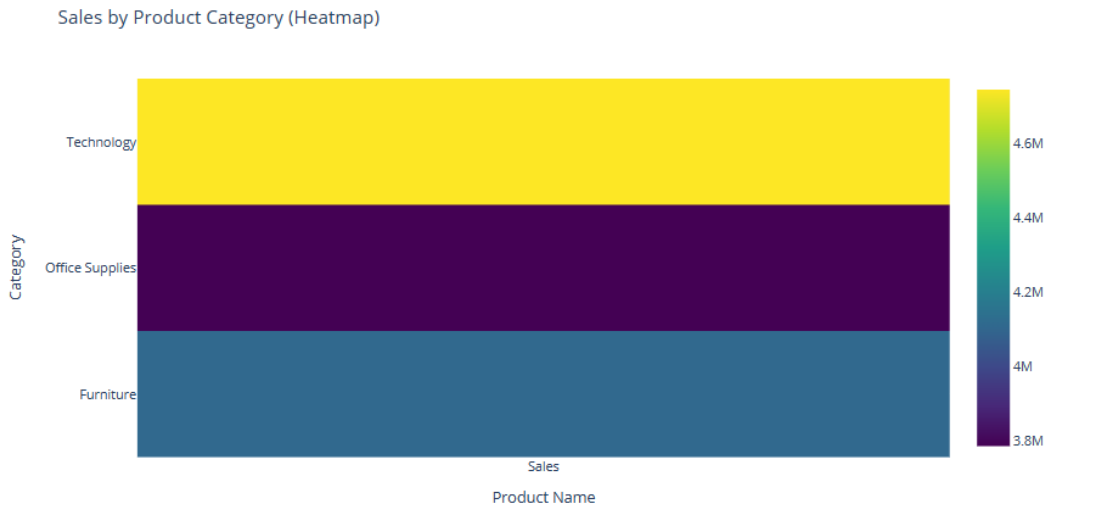
In [15]:
```
# Group the DataFrame by 'Category' and calculate the total sales for each category

category_sales = Ecom.groupby('Category')['Sales'].sum()
top_categories = category_sales.sort_values(ascending=False)
fig = go.Figure(data=[go.Pie(labels=top_categories.index, values=top_categories)])
fig.update_layout(title='Best Selling Categories')
fig.show()
```



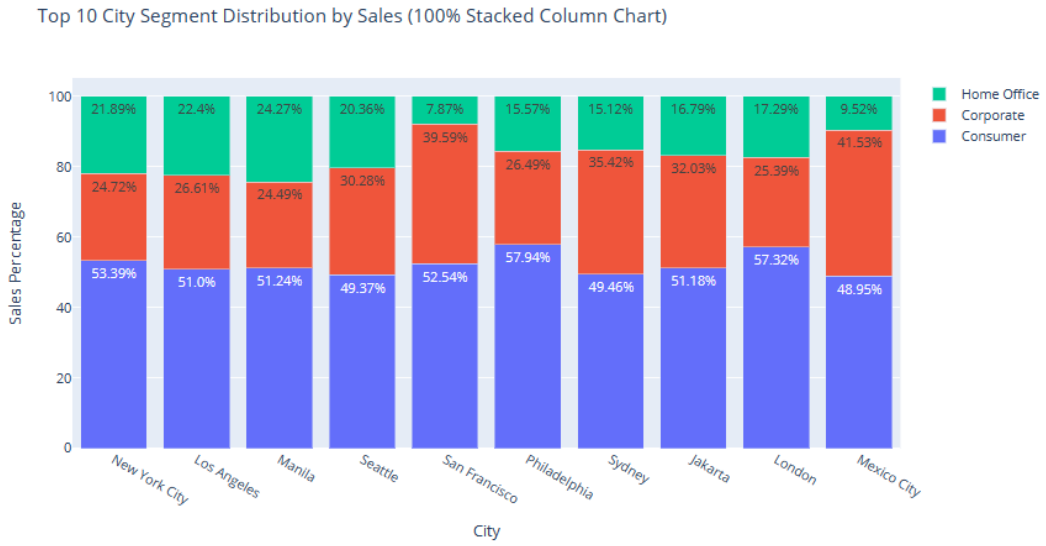


In [16]:
```
city_profit = Ecom.groupby('City')['Profit'].sum()

# Sort the city_profit Series in descending order to get the top 10 profitable cities
top_10_cities = city_profit.sort_values(ascending=False).head(10)

# Extract the top 10 cities and their profits
top_10_city_names = top_10_cities.index
top_10_city_profits = top_10_cities.values

# Create a stacked bar chart using Plotly
fig = go.Figure(data=[go.Bar(name='Profit', x=top_10_city_names, y=top_10_city_profits)])

# Update Layout
fig.update_layout(barmode='stack', title='Top 10 Profitable Cities', xaxis_title='City', yaxis_title='Total Pro

# Show the plot
fig.show()
```





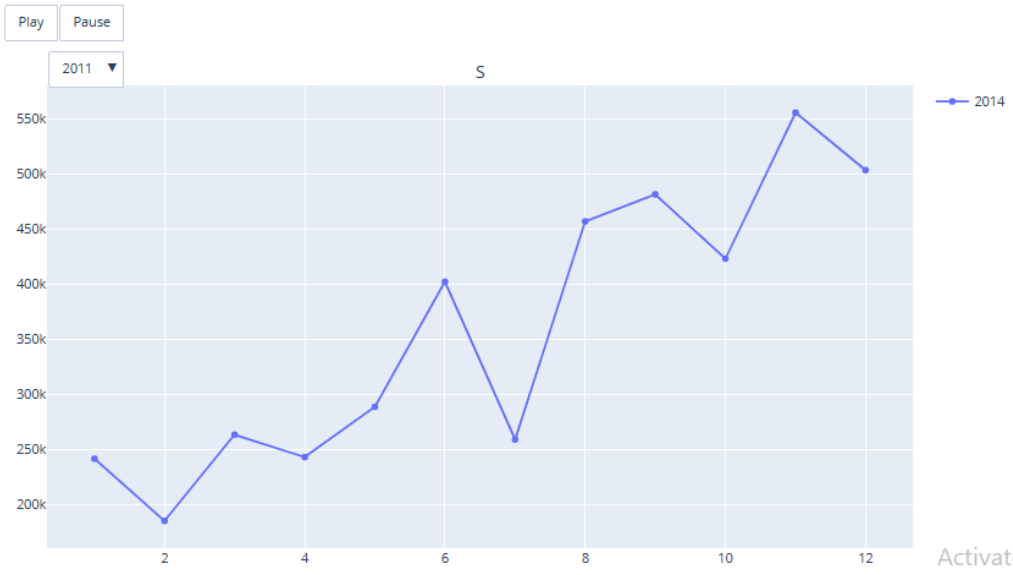
In [17]:
```
import plotly.graph_objects as go

# Pivot the DataFrame to get sales by product category
sales_by_category = Ecom.pivot_table(index='Category', values='Sales', aggfunc='sum')

# Create a heatmap using Plotly
fig = go.Figure(data=go.Heatmap(
    z=sales_by_category.values,
    x=sales_by_category.columns,
    y=sales_by_category.index,
    colorscale='Viridis'))

# Update layout
fig.update_layout(title='Sales by Product Category (Heatmap)', xaxis_title='Product Name', yaxis_title='Categor

# Show the plot
fig.show()
```

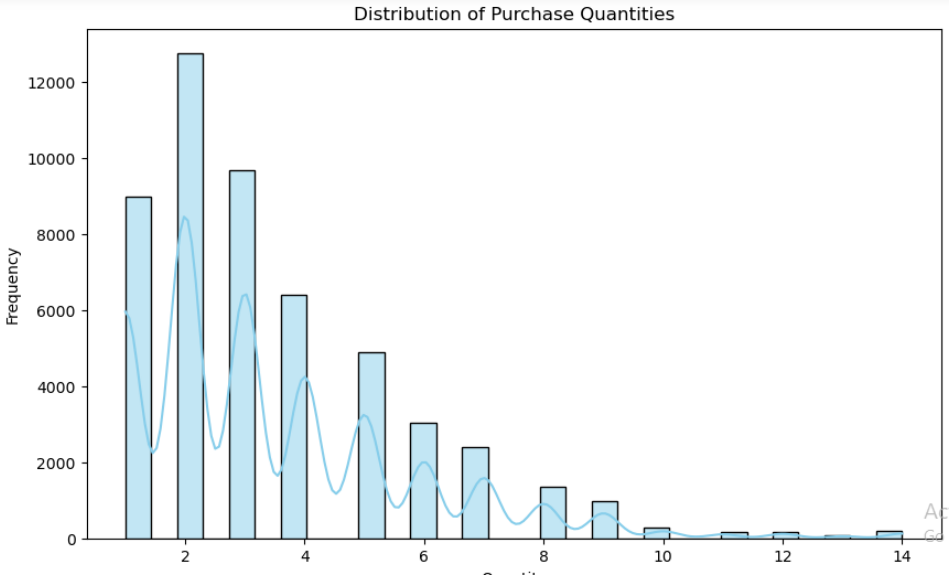


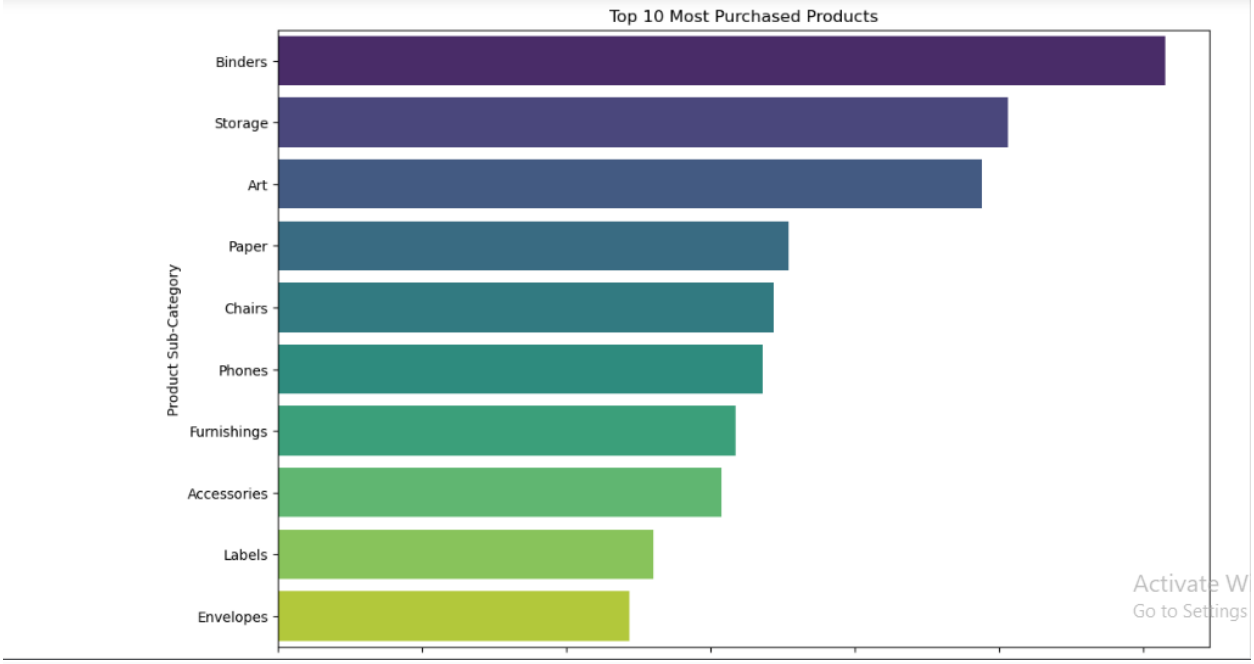
Run Code

In [18]:

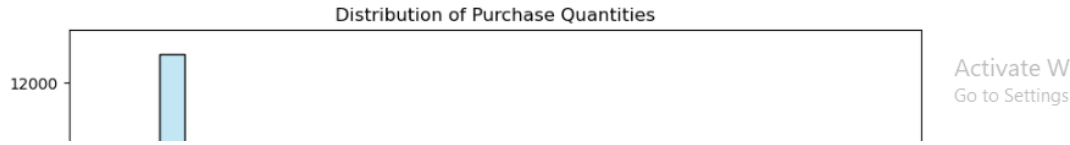
```
import plotly.graph_objects as go

# Group the DataFrame by 'City' and 'Segment', and calculate the total sales for each segment in each city
city_segment_sales = Ecom.groupby(['City', 'Segment'])['Sales'].sum().unstack().fillna(0)

# Select the top 10 cities by total sales
top_10_cities = city_segment_sales.sum(axis=1).nlargest(10).index

# Filter the DataFrame to include only the top 10 cities
top_10_city_segment_sales = city_segment_sales.loc[top_10_cities]

# Normalize the data to get percentages for each segment in each city
normalized_top_10_city_segment_sales = top_10_city_segment_sales.div(top_10_city_segment_sales.sum(axis=1), axi

# Create a 100% stacked column chart using Plotly
fig = go.Figure()

for segment in normalized_top_10_city_segment_sales.columns:
    fig.add_trace(go.Bar(
        x=normalized_top_10_city_segment_sales.index,
        y=normalized_top_10_city_segment_sales[segment],
        name=segment,
        text=normalized_top_10_city_segment_sales[segment].round(2).astype(str) + '%', # Display percentage val
        textposition='auto'
    ))

# Update layout
fig.update_layout(
    barmode='stack',
    title='Top 10 City Segment Distribution by Sales (100% Stacked Column Chart)',
    xaxis_title='City',
    yaxis_title='Sales Percentage'
)

# Show the plot
fig.show()
```



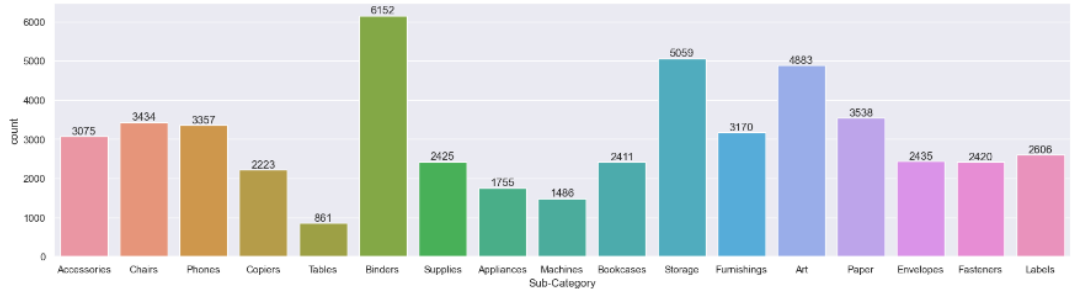




In [17]:
```
# Distribution of purchase quantities
plt.figure(figsize=(10, 6))
sns.histplot(data=Ecom, x='Quantity', bins=30,  kde=True,color='skyblue')
plt.title('Distribution of Purchase Quantities')
plt.xlabel('Quantity')
plt.ylabel('Frequency')
plt.show()

# Distribution of purchase amounts
plt.figure(figsize=(10, 6))
sns.histplot(data=Ecom, x='Sales', bins=30, kde=True, color='salmon')
plt.title('Distribution of Purchase Amounts')
plt.xlabel('Sales')
plt.ylabel('Frequency')
plt.show()

# Most purchased products
plt.figure(figsize=(12, 8))
top_products = Ecom['Sub-Category'].value_counts().nlargest(10)
sns.barplot(x=top_products.values, y=top_products.index, palette='viridis')
plt.title('Top 10 Most Purchased Products')
plt.xlabel('Number of Purchases')
plt.ylabel('Product Sub-Category')
plt.show()
```



In [24]:
```
import seaborn as sns
sns.set(rc={'figure.figsize':(20,5)})
br = sns.countplot(data = Ecom, x = 'Sub-Category')

for bars in br.containers:
    br.bar_label(bars)
```



In [22]:
```
sns.pairplot(Ecom)
```

```
C:\Users\hp\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

Out[22]: <seaborn.axisgrid.PairGrid at 0x25913f534d0>

Out[35]:
```
0        762.1845
1       -288.7650
2        919.9710
3        -96.5400
4        311.5200
           ...
51285      4.5000
51286     -1.1100
51287     11.2308
51288      2.4000
51289      1.8000
Name: Profit, Length: 51290, dtype: float64
```

In [37]:
```
from sklearn.model_selection import train_test_split
```

In [38]:
```
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.8, random_state=42)
```

In [39]:
```
X_test
```

Out[39]:

	Sales	Quantity	Discount
49728	5.868	3	0.4
45547	10.368	2	0.2
15664	269.220	2	0.0
40561	43.176	3	0.2
49426	5.712	1	0.4
...	...	...	...
13528	190.620	2	0.0

In [40]:
```
from sklearn.linear_model import LinearRegression# Importing LinearRegression class from scikit-learn
```

In [41]:
```
Model = LinearRegression()
Model.fit(X_train, y_train)
```

Out[41]:
```
LinearRegression
LinearRegression()
```

In [42]:
```
Predictions = Model.predict(X_test)
```

In [43]:
```
# Calculate the accuracy of the model on the testing data

Accuracy = Model.score(X_test, y_test)
print("Model Accuracy:", Accuracy)
```

```
Model Accuracy: 0.31589766128966423
```

In [44]:
```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score  # Importing evaluation metrics
```

In [45]:
```
mse = mean_squared_error(y_test, Predictions)  # Calculating Mean Squared Error by predicted and actaul value
rmse = np.sqrt(mse)  # Calculating Root Mean Squared Error by Mean Squared Error and Predicted
mae = mean_absolute_error(y_test, Predictions)  # Calculating Mean Absolute Error
r_squared = r2_score(y_test, Predictions)
```

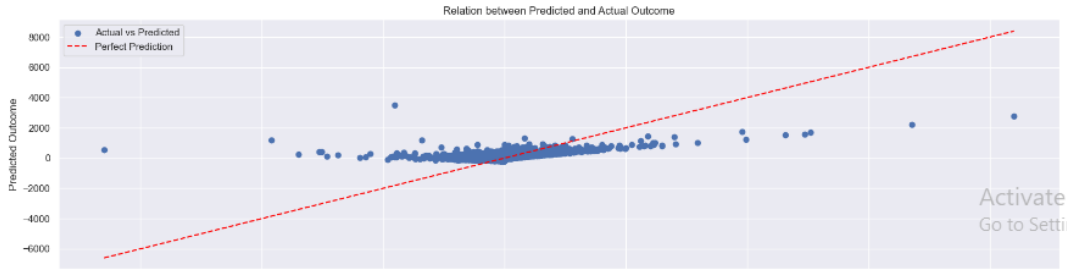
In [46]:
```
print('Mean Squared Error (MSE):', mse)
print('Root Mean Squared Error (RMSE):', rmse)
print('Mean Absolute Error (MAE):', mae)
print('R-squared (R²) Score:', r_squared)
```



```
Mean Squared Error (MSE): 21440.32003740198
```

```
Mean Squared Error (MSE): 21440.32003740198
Root Mean Squared Error (RMSE): 146.42513458215424
Mean Absolute Error (MAE): 59.12111580656918
R-squared (R²) Score: 0.31589766128966423
```

In [44]:
```
y_test_df = pd.DataFrame(y_test)
y_test_df.reset_index(drop=True, inplace=True)

sorted_indices = y_test.argsort()
sorted_pred = Predictions[sorted_indices]
sorted_y_test = y_test_df.iloc[sorted_indices]

plt.scatter(sorted_y_test, sorted_pred, label='Actual vs Predicted')
plt.plot(sorted_y_test, sorted_y_test, color='red', linestyle='--', label='Perfect Prediction')
plt.title('Relation between Predicted and Actual Outcome')
plt.xlabel('Actual Outcome')
plt.ylabel('Predicted Outcome')
plt.legend()
plt.show()
```



Statistical Analysis

Linear Regression Model

In [25]:
```
# Define the heading text
heading = "Linear regression is a statistical method used to model the relationship between a dependent variabl
print("\033[96m"+ "\033[1m" + heading + "\033[0m")
```

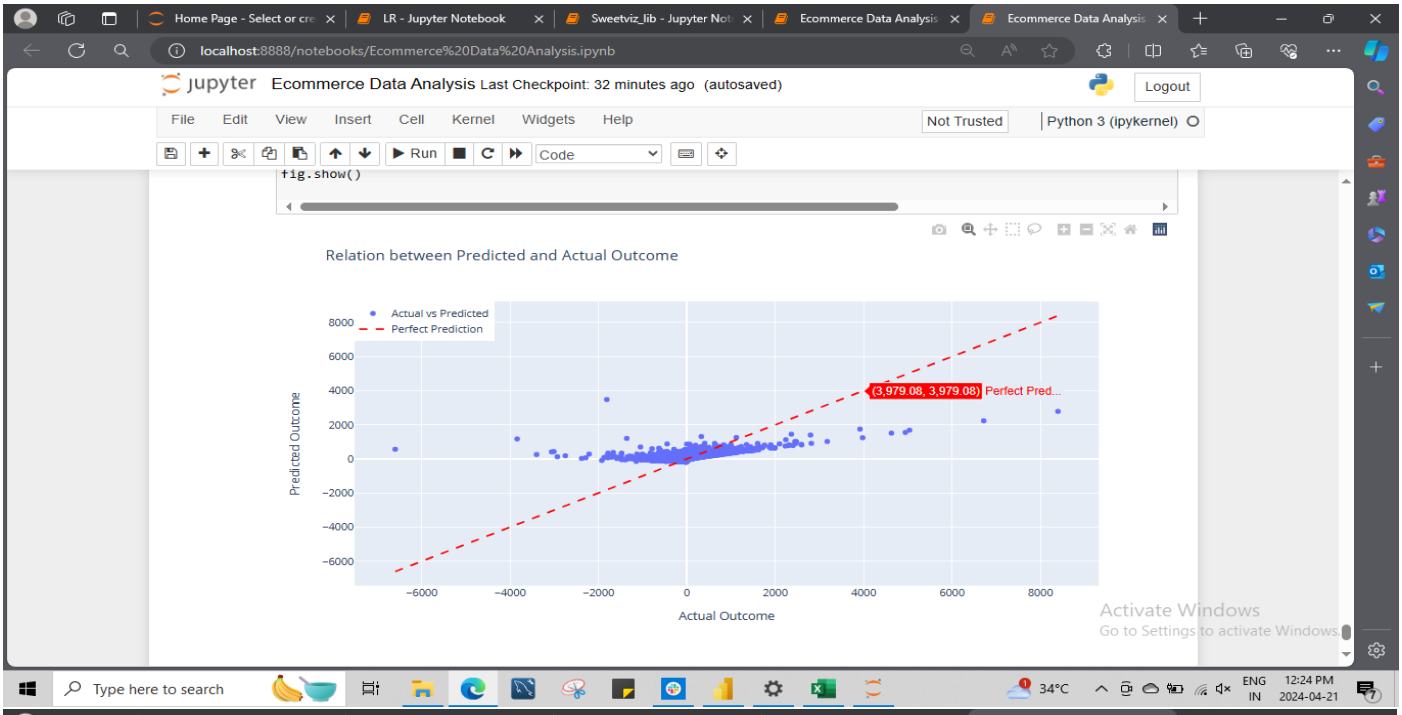
```
Linear regression is a statistical method used to model the relationship between a dependent variable and one
or more independent variables by fitting a linear equation 'Y = mx + c'
```

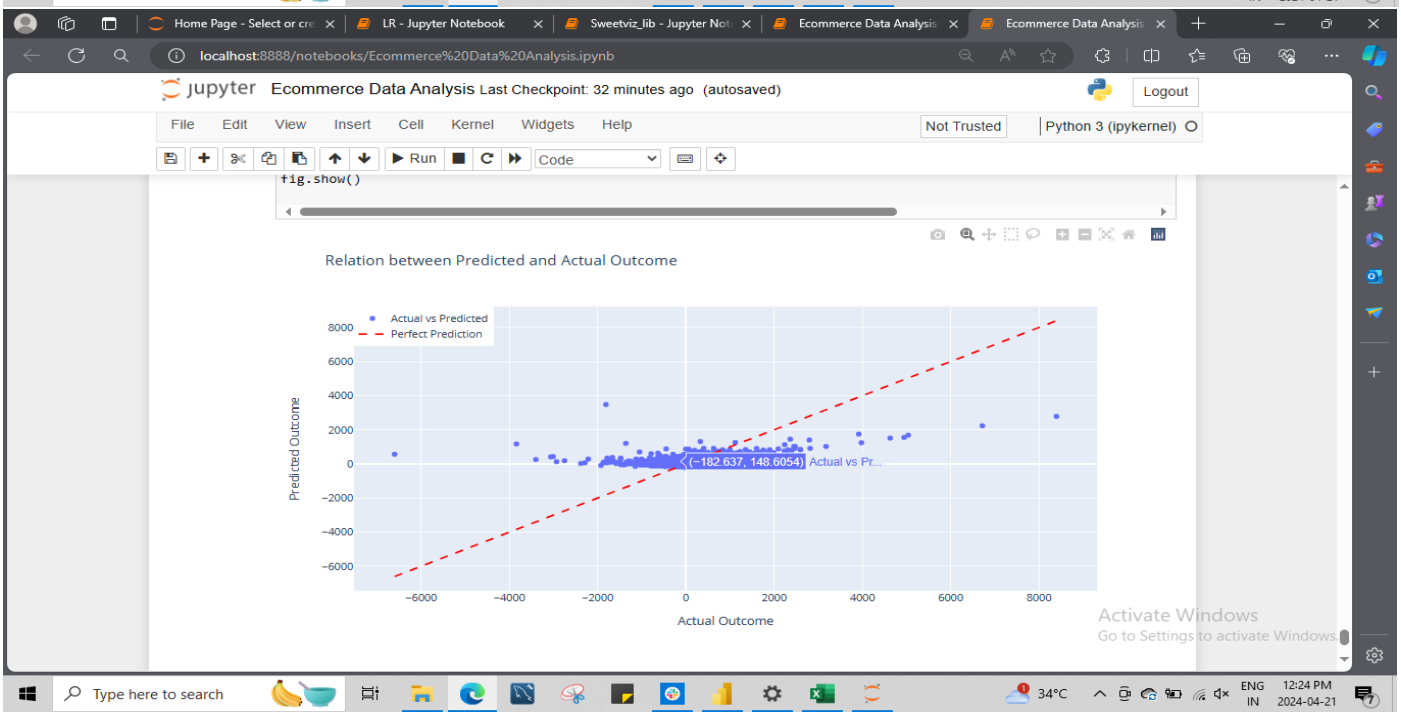
In [33]:
```
X=Ecom[['Sales','Quantity','Discount']]
Y=Ecom['Profit']
```

In [34]:
```
X
```

Out[34]:

	Sales	Quantity	Discount
0	2309.650	7	0.0
1	3709.395	9	0.1
2	5175.171	9	0.1
3	2892.510	5	0.1
4	2832.960	8	0.0
...	...	...	...
51285	65.100	5	0.0


Home Page - Select or cre
LR - Jupyter Notebook
Sweetviz_lib - Jupyter Not
Ecommerce Data Analysis
Ecommerce Data Analysis
localhost:8888/notebooks/Ecommerce%20Data%20Analysis.ipynb
jupyter Ecommerce Data Analysis Last Checkpoint: 32 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
fig.show()
Relation between Predicted and Actual Outcome
Actual vs Predicted
Perfect Prediction
(3,979.08, 3,979.08) Perfect Pred...
Predicted Outcome
Actual Outcome
8000
6000
4000
2000
0
−2000
−4000
−6000
Activate Windows
Go to Settings to activate Windows.
Type here to search
34°C
ENG
IN
12:24 PM
2024-04-21


Home Page - Select or cre
LR - Jupyter Notebook
Sweetviz_lib - Jupyter Not
Ecommerce Data Analysis
Ecommerce Data Analysis
localhost:8888/notebooks/Ecommerce%20Data%20Analysis.ipynb
jupyter Ecommerce Data Analysis Last Checkpoint: 32 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
fig.show()
Relation between Predicted and Actual Outcome
Actual vs Predicted
Perfect Prediction
(−182.637, 148.6054) Actual vs Pr...
Predicted Outcome
Actual Outcome
8000
6000
4000
2000
0
−2000
−4000
−6000
Activate Windows
Go to Settings to activate Windows.
Type here to search
34°C
ENG
IN
12:24 PM
2024-04-21