Suraj Shete
9326183641

WALMART RETAIL AND SALES ANALYSIS

Dataset Description: -

- The dataset has been taken from the website: - Kaggle.
- It contains 16 columns and 9985 rows.
- It contains sales of Walmart store from the different states of USA from year 2011 to 2014.
- It tells us about the Sales, Quantity, Discount and Profit of different product from different States, City and Regions of USA based on Category and Sub-Category.
- Revenue and Unit Price are the two columns that have been added as require.
- It also contains null values, unique values, categorical and numerical values.
- The Country column represents both USA as well as United States.
- It may also contain falsy values or null rows and columns.
- Here the target attributes are Sales, Profit, Revenue and Unit Price

Dataset link: -

https://www.kaggle.com/datasets/surajjjjjjjjj/walmart-sales

Analysis Tasks: -

- Using the above-mentioned dataset, we will perform all the phases of Data Science life cycle i.e., Data Understanding, Data Preparation, Data Visualization, Data Modelling and Model Evaluation

- The below table specifies the name of the columns, their data types, the feature is categorical or numerical and their description.

Suraj Shete
9326183641

Column Name	Datatype	Categorical/ Numerical/ Unique	Description
Order ID	object	Unique	It contains unique id for each product
Order Date	object	Time stamp	The date on which order placed
Ship Date	object	Time Stamp	The date on which order shipped
Customer Name	object	Categorical	Name of the customer
Country	object	Categorical	Name of the country
City	object	Categorical	Name of city from which order placed
State	object	Categorical	Name of state from which order placed
Postal Code	float64	Numerical	Pin-code
Region	object	Categorical	Order from which region
Category	object	Categorical	Product category
Sub-Category	object	Categorical	Product sub-category
Product Name	object	Categorical	Name of the product
Sales	float64	Numerical	Sale of a product
Quantity	float64	Numerical	Quantity of product customer buy
Discount	float64	Numerical	Discount applied on product

Profit	float64	Numerical	Profit gain form the product

- The following two columns are added as required
    1. Revenue
    2. Unit Price

Revenue	**Float64**	**Numerical**	**Revenue generated**
Unit Price	Float64	Numerical	Price per product

Suraj Shete
9326183641

Project

Code and Output with their Explanation:

Importing the required libraries and reading the dataset:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
%matplotlib inline
```

```
path="/content/drive/MyDrive/Colab Notebooks/Walmart.csv"
walmart=pd.read_csv(path)
walmart.head()
```

	Order ID	Order Date	Ship Date	Customer Name	Country	City	State	Postal Code	Region	Category	Sub-Category	Product Name	Sales	Quantity	Discount	Profit
0	CA-2013-152156	09-11-2013	12-11-2013	Claire Gute	United States	Henderson	Kentucky	42420.0	South	Furniture	Bookcases	Bush Somerset Collection Bookcase	261.9600	2.0	0.00	41.9136
1	CA-2013-152156	09-11-2013	12-11-2013	Claire Gute	United States	Henderson	Kentucky	42420.0	South	Furniture	Chairs	Hon Deluxe Fabric Upholstered Stacking Chairs,...	731.9400	3.0	0.00	219.5820
2	CA-2013-138688	13-06-2013	17-06-2013	Darrin Van Huff	United States	Los Angeles	California	90036.0	West	Office Supplies	Labels	Self-Adhesive Address Labels for Typewriters b...	14.6200	2.0	0.00	6.8714
3	US-2012-108966	11-10-2012	18-10-2012	Sean O'Donnell	United States	Fort Lauderdale	Florida	33311.0	South	Furniture	Tables	Bretford CR4500 Series Slim Rectangular Table	957.5775	5.0	0.45	-383.0310
4	US-2012-108966	11-10-2012	18-10-2012	Sean O'Donnell	United States	Fort Lauderdale	Florida	33311.0	South	Office Supplies	Storage	Eldon Fold 'N Roll Cart System	22.3680	2.0	0.20	2.5164

```
walmart.info
```

```
<bound method DataFrame.info of            Order ID  Order Date   Ship Date   (
0     CA-2013-152156  09-11-2013  12-11-2013      Claire Gute  United States
1     CA-2013-152156  09-11-2013  12-11-2013      Claire Gute  United States
2     CA-2013-138688  13-06-2013  17-06-2013  Darrin Van Huff  United States
3     US-2012-108966  11-10-2012  18-10-2012   Sean O'Donnell  United States
4     US-2012-108966  11-10-2012  18-10-2012   Sean O'Donnell  United States
...              ...         ...         ...              ...            ...
9992  CA-2014-121258  27-02-2014  04-03-2014      Dave Brooks  United States
9993  CA-2014-121258  27-02-2014  04-03-2014      Dave Brooks  United States
9994             NaN         NaN         NaN              NaN            NaN
```

Suraj Shete
9326183641

Printing number of rows, columns and checking data types for each column:

```
print("Rows", walmart.shape[0])
print("Columns", walmart.shape[1])
```

```
Rows 9997
Columns 16
```

```
walmart.dtypes
```

```
Order ID          object
Order Date        object
Ship Date         object
Customer Name     object
Country           object
City              object
State             object
Postal Code      float64
Region            object
Category          object
Sub-Category      object
Product Name      object
Sales            float64
Quantity         float64
Discount         float64
Profit           float64
dtype: object
```

Checking for null values and their count using .sum() function:

Suraj Shete
9326183641

```
walmart.isna().sum()
```

```
Order ID         13
Order Date       13
Ship Date        13
Customer Name    13
Country          13
City             13
State            13
Postal Code      12
Region           13
Category         13
Sub-Category     13
Product Name     13
Sales            14
Quantity         13
Discount         13
Profit           13
dtype: int64
```

Removing the rows with null values:

```
# Remove rows with any null values
walmart = walmart.dropna()
print(walmart)
```

```
            Order ID  Order Date   Ship Date    Customer Name
0     CA-2013-152156  09-11-2013  12-11-2013      Claire Gute
1     CA-2013-152156  09-11-2013  12-11-2013      Claire Gute
2     CA-2013-138688  13-06-2013  17-06-2013  Darrin Van Huff
3     US-2012-108966  11-10-2012  18-10-2012   Sean O'Donnell
4     US-2012-108966  11-10-2012  18-10-2012   Sean O'Donnell
...              ...         ...         ...              ...
9990  CA-2014-121258  27-02-2014  04-03-2014      Dave Brooks
9991  CA-2014-121258  27-02-2014  04-03-2014      Dave Brooks
9992  CA-2014-121258  27-02-2014  04-03-2014      Dave Brooks
```

Rechecking the count of null values:

Suraj Shete
9326183641

```
walmart.isna().sum()

Order ID         0
Order Date       0
Ship Date        0
Customer Name    0
Country          0
City             0
State            0
Postal Code      0
Region           0
Category         0
Sub-Category     0
Product Name     0
Sales            0
Quantity         0
Discount         0
Profit           0
dtype: int64
```

The data is collected from country USA but the country column contains two unique values by using .unique() function:

```
walmart['Country'].unique()

array(['United States', 'USA'], dtype=object)
```

Replacing ‘USA’ with ‘United States’ in country column

```
#replacing "USA" with "United states" in Country column
walmart = pd.DataFrame(walmart)
walmart['Country'] = walmart['Country'].replace(['USA', 'United States'])
```

Checking unique values from ‘Category’ and their value counts using function value_counts()

Suraj Shete
9326183641

```
walmart['Category'].unique()
```

```
array(['Furniture', 'Office Supplies', 'Technology'], dtype=object)
```

```
walmart['Category'].value_counts()
```

```
Office Supplies    6020
Furniture          2119
Technology         1844
Name: Category, dtype: int64
```

Similarly, for ‘Sub-Category’ column:

```
walmart['Sub-Category'].unique()
```

```
array(['Bookcases', 'Chairs', 'Labels', 'Tables', 'Storage',
       'Furnishings', 'Art', 'Phones', 'Binders', 'Appliances', 'Paper',
       'Accessories', 'Envelopes', 'Fasteners', 'Supplies', 'Machines',
       'Copiers'], dtype=object)
```

```
walmart['Sub-Category'].value_counts()
```

```
Binders        1520
Paper          1368
Furnishings     956
Phones          888
Storage         846
Art             796
Accessories     774
Chairs          617
Appliances      465
Labels          364
Tables          318
Envelopes       254
Bookcases       228
Fasteners       217
Supplies        190
Machines        114
Copiers          68
Name: Sub-Category, dtype: int64
```

Understanding the central tendencies:

Checking null values for ‘Sales’ column and filling those with mean but we don’t get any null values:

Suraj Shete
9326183641

```
#checking null values in Sales column
walmart= pd.DataFrame(walmart)
null=walmart["Sales"].isna().value_counts()
print(null)
```

```
False    9983
Name: Sales, dtype: int64
```

```
mean_sales=walmart['Sales'].mean()
walmart['Sales'].fillna(mean_sales, inplace=True)
```

```
walmart["Sales"].isna().value_counts()
```

```
False    9983
Name: Sales, dtype: int64
```

Printing the maximum and minimum for columns ‘Sales’ and ‘Profit’

```
print("Maximun Sales=",walmart["Sales"].max())
print("Minimun Sales=",walmart["Sales"].min())
```

```
Maximun Sales= 22638.48
Minimun Sales= 0.444
```

```
print("Maximun Profit=",walmart["Profit"].max())
print("Minimun Profit=",walmart["Profit"].min())
```

```
Maximun Profit= 8399.976
Minimun Profit= -6599.978
```

Getting the mean and standard deviation for ‘Sales’

```
#mean value of sales column
print("Average Sales", walmart["Sales"].mean())
```

```
Average Sales 229.7878046979866
```

```
#Standard Deviation of sales column
print("Standard Deviation", walmart["Sales"].std())
#here the spread of the data is large due to SD>Mean
```

```
Standard Deviation 623.4191215790974
```

Maximum Sales are from Technology Category and Minimum are from Office Supplies:

Suraj Shete
9326183641

```
walmart.groupby("Category")["Sales"].max().sort_values(ascending=False)
#maximun Sales from Category

Category
Technology        22638.480
Office Supplies    9892.740
Furniture          4416.174
Name: Sales, dtype: float64

walmart.groupby("Category")["Sales"].min().sort_values(ascending=False)
#minimun Sales from Category

Category
Furniture         1.892
Technology        0.990
Office Supplies   0.444
Name: Sales, dtype: float64
```

Maximum Profit generated through Technology Category:

```
walmart.groupby("Category")["Profit"].max().sort_values(ascending=False)
#maximun Profit from category

Category
Technology        8399.976
Office Supplies   4946.370
Furniture         1013.127
Name: Profit, dtype: float64

walmart.groupby("Category")["Profit"].min().sort_values(ascending=False)
#minimun Profit from category

Category
Furniture        -1862.3124
Office Supplies  -3701.8928
Technology       -6599.9780
Name: Profit, dtype: float64
```

Names of the top 5 Customers from which maximum 'Sales' and 'Profit' is been generated:

```
[ ] walmart.groupby("Customer Name")["Sales"].max().sort_values(ascending=False).head()
    #top 5 Customer with max sales

    Customer Name
    Sean Miller      22638.480
    Tamara Chand     17499.950
    Raymond Buch     13999.960
    Tom Ashbrook     11199.968
    Hunter Lopez     10499.970
    Name: Sales, dtype: float64

[ ] walmart.groupby("Customer Name")["Profit"].max().sort_values(ascending=False).head()
    #Top 5 Customer with max Profit

    Customer Name
    Tamara Chand      8399.9760
    Raymond Buch      6719.9808
    Hunter Lopez      5039.9856
    Adrian Barton     4946.3700
    Sanjit Chand      4630.4755
    Name: Profit, dtype: float64
```

Suraj Shete
9326183641

Top 5 Cities that had generated highest Profit:

```
print("Top 5 City with highest Profit",walmart.groupby("City")["Profit"].max().sort_values(ascending=False).head())

Top 5 City with highest Profit City
Lafayette      8399.9760
Seattle        6719.9808
Newark         5039.9856
Detroit        4946.3700
Minneapolis    4630.4755
Name: Profit, dtype: float64
```

City with the highest Profit with all the details: Tamara Chand from the city Lafayette have been the customer from which the profit of 83999.976 is been generated**.**

```
# Find the city with the highest total profit
highest_profit_city = walmart[walmart["Profit"] == walmart["Profit"].max()]

# Print the city with the highest profit
print("City with the highest profit with all the details:")
print(highest_profit_city)

City with the highest profit with all the details:
            Order ID  Order Date   Ship Date Customer Name        Country  \
6826  CA-2013-118689  03-10-2013  10-10-2013  Tamara Chand  United States

           City    State  Postal Code   Region    Category Sub-Category  \
6826  Lafayette  Indiana      47905.0  Central  Technology      Copiers

                        Product Name     Sales  Quantity  Discount  \
6826  Canon imageCLASS 2200 Advanced Copier  17499.95       5.0       0.0

        Profit
6826  8399.976
```

Top 5 Profitable and Non-Profitable product: Canon imageCLASS 2200 Advance Copier is the most profitable product

```
#Top 5 profitable and non-profitable products
product_profit = print("Most Profitable",walmart.groupby('Product Name')['Profit'].sum().sort_values(ascending=False).head())
print(product_profit)
least_profit = print("Least Profitable",walmart.groupby('Product Name')['Profit'].sum().sort_values(ascending=True).head())
print(least_profit)

Most Profitable Product Name
Canon imageCLASS 2200 Advanced Copier                                          25199.9280
Fellowes PB500 Electric Punch Plastic Comb Binding Machine with Manual Bind     7753.0390
Hewlett Packard LaserJet 3310 Copier                                            6983.8836
Canon PC1060 Personal Laser Copier                                              4570.9347
HP Designjet T520 Inkjet Large Format Printer - 24" Color                       4094.9766
Name: Profit, dtype: float64
None
Least Profitable Product Name
Cubify CubeX 3D Printer Double Head Print                -8879.9704
Lexmark MX611dhe Monochrome Laser Printer                -4589.9730
Cubify CubeX 3D Printer Triple Head Print                -3839.9904
Chromcraft Bull-Nose Wood Oval Conference Tables & Bases  -2876.1156
Bush Advantage Collection Racetrack Conference Table      -1934.3976
Name: Profit, dtype: float64
None
```

Statistical Summary using aggregate function:

Suraj Shete
9326183641

```
#Statistical Summary through table using aggrigate function
walmart.agg(
    {
    "Sales":["min", "max","mean"],
    "Profit":["min", "max","mean"]
    }
)
```

	Sales	Profit
min	0.444000	-6599.978000
max	22638.480000	8399.976000
mean	229.787805	28.682607

Finding Correlation:

‘Sales’ and ‘Quantity’ are positively correlated.

‘Quantity’ and ‘Profit’ are positively correlated.

‘Discount’ and ‘Profit’ are negatively correlated.

```
walmart['Sales'].corr(walmart['Quantity'])
```

```
0.201123100024671
```

```
walmart['Quantity'].corr(walmart['Profit'])
```

```
0.0663468210221917
```

```
walmart['Discount'].corr(walmart['Profit'])
```

```
-0.21935917522442527
```

```
walmart.corr() #using corr() function
```

```
<ipython-input-35-77a9cb43bf62>:1: FutureWarning: The default value of nume
  walmart.corr() #using corr() function
```

	Postal Code	Sales	Quantity	Discount	Profit
Postal Code	1.000000	-0.023650	0.012497	0.058096	-0.029873
Sales	-0.023650	1.000000	0.201123	-0.028021	0.479143
Quantity	0.012497	0.201123	1.000000	0.007804	0.066347
Discount	0.058096	-0.028021	0.007804	1.000000	-0.219359
Profit	-0.029873	0.479143	0.066347	-0.219359	1.000000

Kurtosis:

Postal Code has distribution of fewer extreme values, it is Platykurtic.

Sales, Quantity, Discount, Profit suggest that distribution may have many extreme values, both very high and very low.

```
#Kurtosis
walmart.kurt()
```

```
<ipython-input-36-a7139c28e432>:2:
  walmart.kurt()
Postal Code     -1.493248
Sales          305.311764
Quantity         1.998331
Discount         2.418906
Profit         396.910218
dtype: float64
```

Skewness:

Suraj Shete

9326183641

Postal Code is Negatively Skewed whereas

Sales, Quantity, Discount and Profit are Positively Skewed

Changing the datatype of “Profit” and “Quantity” column to int64

```
#Skewness
walmart.skew()
```

```
<ipython-input-41-f9b2e7b06786>:2: Futur
  walmart.skew()
Postal Code    -0.128879
Sales          12.975758
Quantity        1.279952
Discount        1.685860
Profit          7.559241
dtype: float64
```

```
walmart["Profit"]=walmart["Profit"].astype('int64') #cha
walmart["Quantity"]=walmart["Quantity"].astype('int64')
walmart.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9983 entries, 0 to 9996
Data columns (total 16 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Order ID       9983 non-null   object 
 1   Order Date     9983 non-null   object 
 2   Ship Date      9983 non-null   object 
 3   Customer Name  9983 non-null   object 
 4   Country        9983 non-null   object 
 5   City           9983 non-null   object 
 6   State          9983 non-null   object 
 7   Postal Code    9983 non-null   float64
 8   Region         9983 non-null   object 
 9   Category       9983 non-null   object 
 10  Sub-Category   9983 non-null   object 
 11  Product Name   9983 non-null   object 
 12  Sales          9983 non-null   float64
 13  Quantity       9983 non-null   int64  
 14  Discount       9983 non-null   float64
 15  Profit         9983 non-null   int64  
dtypes: float64(3), int64(2), object(11)
memory usage: 1.3+ MB
```

Updating the dataset with ‘Revenue’ attribute and filling data with total revenue generated by each product:

```
#Updating the data with "Revenue" column and finding out total revenue generated by each product
walmart["Revenue"]=walmart["Sales"] + walmart["Profit"]
walmart.head()
```

	Order ID	Order Date	Ship Date	Customer Name	Country	City	State	Postal Code	Region	Category	Sub-Category	Product Name	Sales	Quantity	Discount	Profit	Revenue
0	CA-2013-152156	09-11-2013	12-11-2013	Claire Gute	United States	Henderson	Kentucky	42420.0	South	Furniture	Bookcases	Bush Somerset Collection Bookcase	261.9600	2	0.00	41	302.9600

Updating the dataset with ‘Unit Price’ attribute and filling column with price per product:

```
walmart["Unit Price"]= (walmart["Revenue"] - walmart["Profit"]) /walmart["Quantity"]
walmart.head()
```

	Order ID	Order Date	Ship Date	Customer Name	Country	City	State	Postal Code	Region	Category	Sub-Category	Product Name	Sales	Quantity	Discount	Profit	Revenue	Unit Price
0	CA-2013-152156	09-11-2013	12-11-2013	Claire Gute	United States	Henderson	Kentucky	42420.0	South	Furniture	Bookcases	Bush Somerset Collection Bookcase	261.9600	2	0.00	41	302.9600	130.9800

Data Visualization

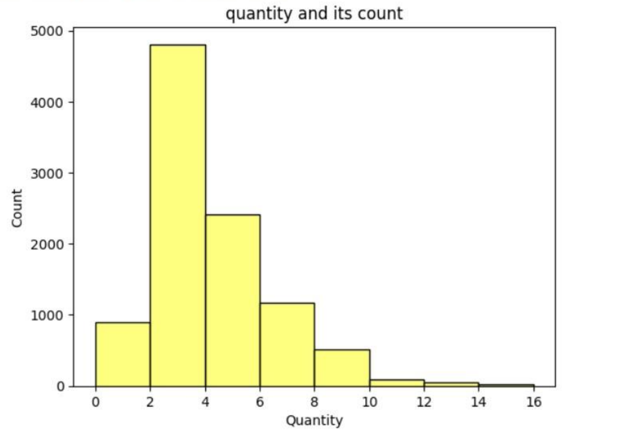
Suraj Shete
9326183641

Histogram Plot:

The Count for ‘Quantity’ is highest between 2-4

```
bin=[0,2,4,6,8,10,12,14,16]
sns.histplot(x='Quantity', bins=bin, data=walmart, color="yellow", alpha=0.5)
plt.title("quantity and its count")
#this histogram show the quantity and its count
```

```
Text(0.5, 1.0, 'quantity and its count')
```

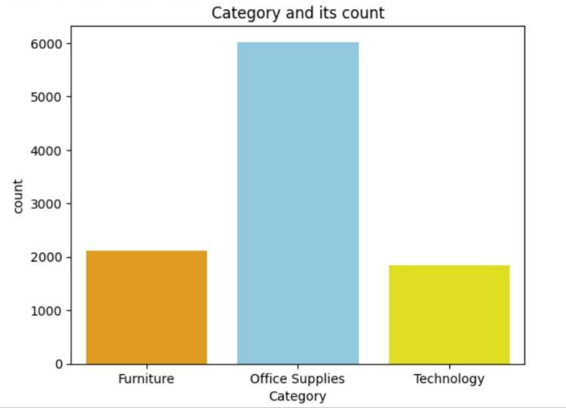


Count Plot:

‘Office Supplies’ has the highest count

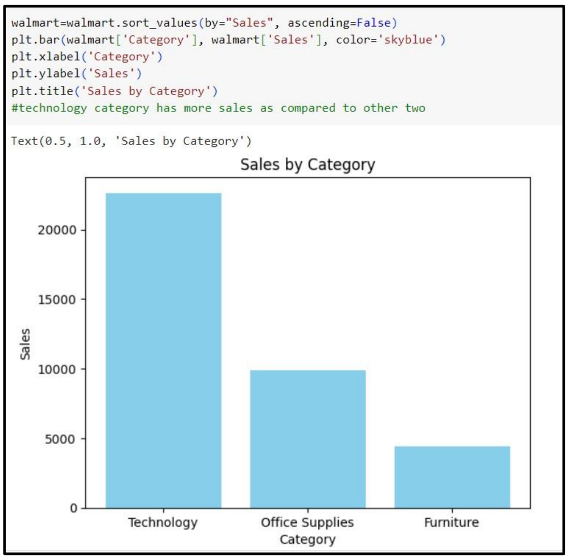
```
sns.countplot(x='Category', data=walmart, palette=["orange","skyblue","yellow"])
plt.title("Category and its count")
#here the office supplies have the highest count
```

```
Text(0.5, 1.0, 'Category and its count')
```

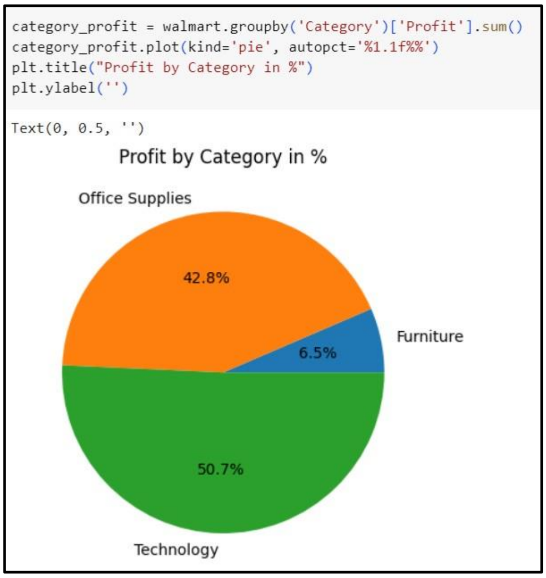


Technology has the highest Sales from the Category

Suraj Shete
9326183641



From the below pie chart, we can conclude that the profit generated by the Technology Category is the highest i.e., 50.7%

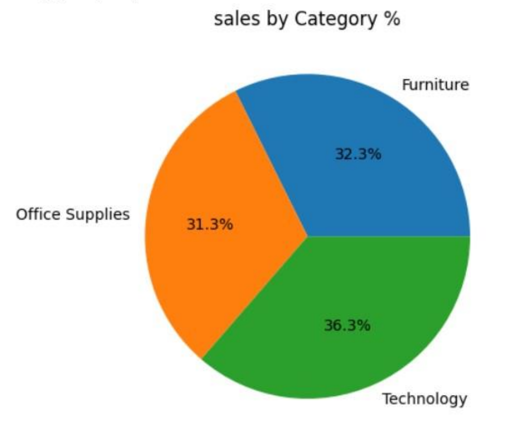


36.3% of the Sales is done from Technology Category:

Suraj Shete
9326183641

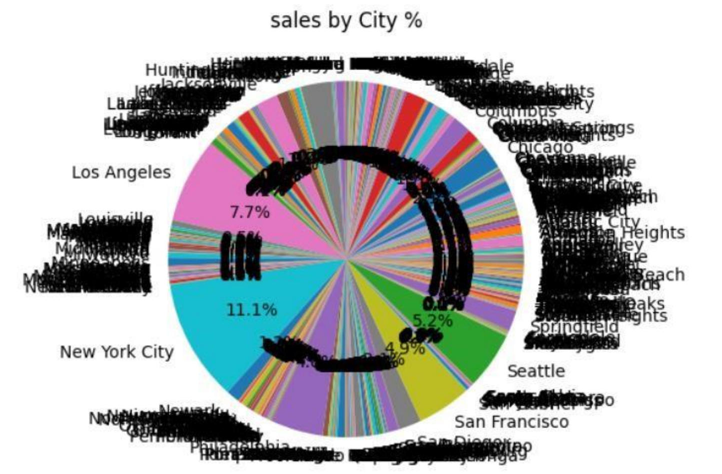
```
category_profit = walmart.groupby('Category')['Sales'].sum()
category_profit.plot(kind='pie', autopct='%1.1f%%')
plt.title("sales by Category %")
plt.ylabel('')
```

```
Text(0, 0.5, '')
```



The New-York City has the highest number of Sales i.e., 11.1% rather than other cities

```
Text(0, 0.5, '')
```



Correlation Coefficient:

There is a positive correlation between Sales and Profit= 0.48, which is greater than 0

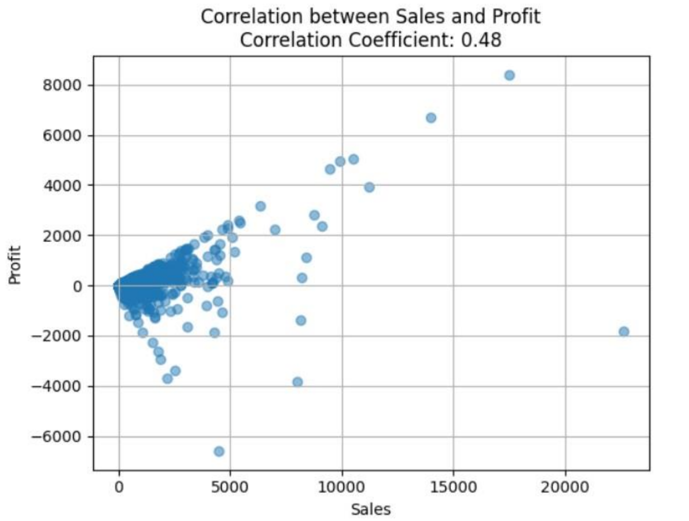
```
# Calculate the correlation coefficient between Sales and Profit
correlation_coefficient = walmart['Sales'].corr(walmart['Profit'])

# Create a scatter plot to visualize the relationship
plt.scatter(walmart['Sales'], walmart['Profit'], alpha=0.5)
plt.title(f"Correlation between Sales and Profit\nCorrelation Coefficient: {correlation_coefficient:.2f}")
plt.xlabel("Sales")
plt.ylabel("Profit")
plt.grid(True)

# Determine if there's a positive correlation
if correlation_coefficient > 0:
    print("There is a positive correlation between Sales and Profit.")
elif correlation_coefficient < 0:
    print("There is a negative correlation between Sales and Profit.")
else:
    print("There is no significant correlation between Sales and Profit.")
```

Suraj Shete
9326183641

There is a positive correlation between Sales and Profit.



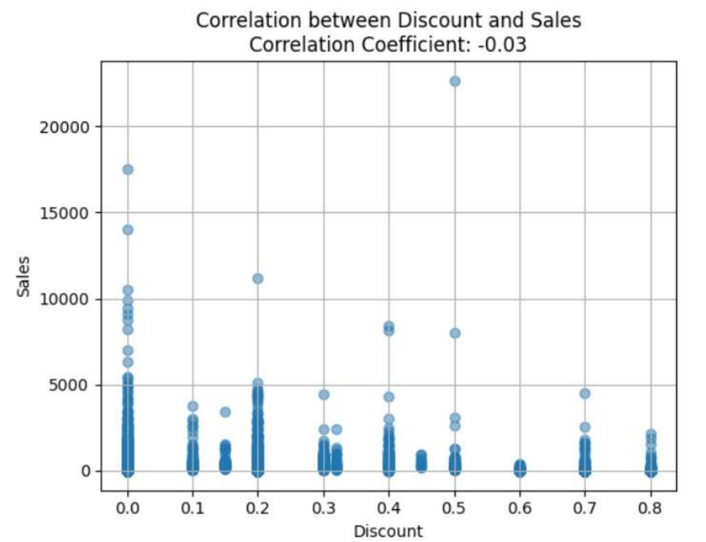
There is negative correlation between Sales and Discount= -0.03, which is less than 0

```python
# Calculate the correlation coefficient between Sales and Discount
correlation_coefficient = walmart['Discount'].corr(walmart['Sales'])

# Create a scatter plot to visualize the relationship
plt.scatter(walmart['Discount'], walmart['Sales'], alpha=0.5)
plt.title(f"Correlation between Discount and Sales\nCorrelation Coefficient: {correlation_coefficient:.2f}")
plt.xlabel("Discount")
plt.ylabel("Sales")
plt.grid(True)

# Determine if there's a positive correlation
if correlation_coefficient > 0:
    print("There is a positive correlation between Sales and Discount.")
elif correlation_coefficient < 0:
    print("There is a negative correlation between Sales and Discount.")
else:
    print("There is no significant correlation between Sales and Discount.")
```

There is a negative correlation between Sales and Discount.



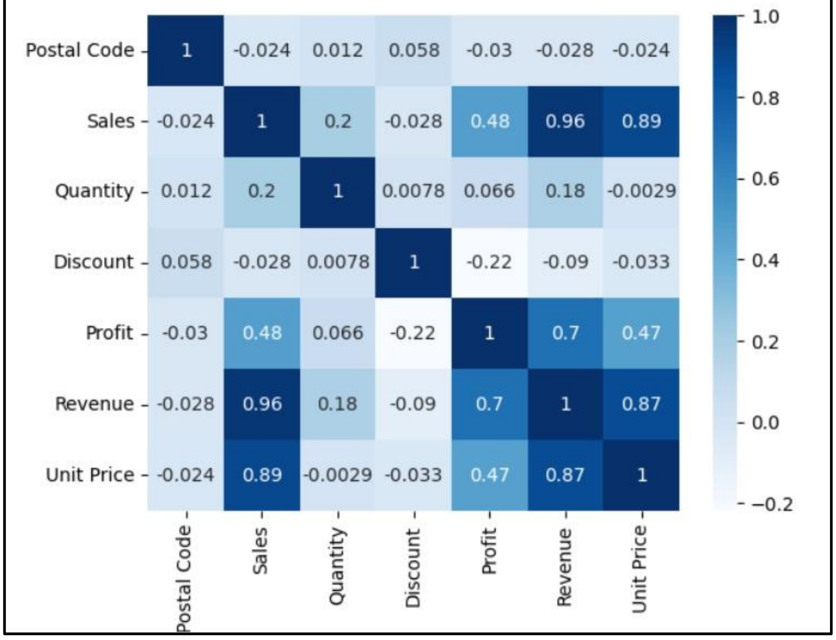
Suraj Shete
9326183641

Correlation using Heatmap:

We can observe that Sales and Revenue are highly correlated with each other= 0.96

```
sns.heatmap(walmart.corr(), annot=True, cmap="Blues")
```

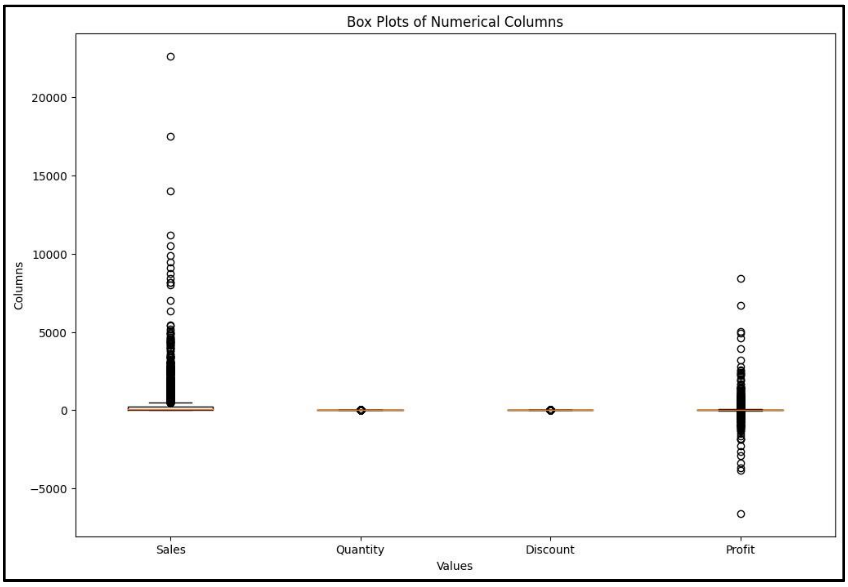
```
<ipython-input-67-0966444d4b40>:1: FutureWarning: The default value of numeric_
  sns.heatmap(walmart.corr(), annot=True, cmap="Blues")
<Axes: >
```



	Postal Code	Sales	Quantity	Discount	Profit	Revenue	Unit Price
Postal Code	1	-0.024	0.012	0.058	-0.03	-0.028	-0.024
Sales	-0.024	1	0.2	-0.028	0.48	0.96	0.89
Quantity	0.012	0.2	1	0.0078	0.066	0.18	-0.0029
Discount	0.058	-0.028	0.0078	1	-0.22	-0.09	-0.033
Profit	-0.03	0.48	0.066	-0.22	1	0.7	0.47
Revenue	-0.028	0.96	0.18	-0.09	0.7	1	0.87
Unit Price	-0.024	0.89	-0.0029	-0.033	0.47	0.87	1

Outliers Detection for NumericalData

```
numerical_columns = ['Sales', 'Quantity', 'Discount', 'Profit']
plt.figure(figsize=(12, 8))
plt.boxplot([walmart[column] for column in numerical_columns], labels=numerical_columns, vert=True)
plt.title("Box Plots of Numerical Columns")
plt.xlabel("Values")
plt.ylabel("Columns")
```



Suraj Shete
9326183641

Model Building and Evaluation:

Multi-Linear Regression model:

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

# Select the independent variables (features) and the dependent variable (target)
X = walmart[['Sales', 'Quantity', 'Discount', 'Profit','Unit Price']]
y = walmart['Revenue']

# 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)

# Initialize and train the linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

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

mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error:", mse)
print("Root Mean Squared Error:", rmse)
print("R-squared:", r2)

# Coefficients and intercept
print("Coefficients:", model.coef_)
print("Intercept:", model.intercept_)
```

```
Mean Squared Error: 153762.58659207032
Root Mean Squared Error: 392.1257280414922
R-squared: 0.9594618571098177
Coefficients: [   1.23555439   -43.04493259 -108.79673922    2.23239797   -0.28656498]
Intercept: 141.55402537035707
```

Interpretation for Multi-linear Regression model:

Mean Squared Error	153762.58659207032
Root Mean Squared Error	392.1257280414922
R-Squared Error	0.9594618571098177
Coefficients	1.23555439, -43.04493259, -108.79673922, 2.23239797, -0.28656498
Intercept	141.55402537035707

Suraj Shete
9326183641

1. **Sales:** The coefficient for 'Sales' is approximately 1.236. This means that, while holding all other variables constant, a one-unit increase in 'Sales' is associated with an increase of approximately 1.236 units in 'Revenue'. In other words, for each additional unit of sales, you can expect the revenue to increase by roughly 1.236 units.
2. **Quantity:** The coefficient for 'Quantity' is approximately -43.045. This suggests that, while holding all other variables constant, a one-unit increase in 'Quantity' is associated with a decrease of approximately 43.045 units in 'Revenue'. In other words, higher quantities sold are associated with lower revenue.
3. **Discount:** The coefficient for 'Discount' is approximately -108.797. This implies that, while holding all other variables constant, a one-unit increase in 'Discount' is associated with a decrease of approximately 108.797 units in 'Revenue'. This suggests that offering discounts has a significant negative impact on revenue.
4. **Profit:** The coefficient for 'Profit' is approximately 2.232. This means that, while holding all other variables constant, a one-unit increase in 'Profit' is associated with an increase of approximately 2.232 units in 'Revenue'. In other words, higher profits are associated with higher revenue.
5. **Unit Price:** The coefficient for 'Unit Price' is approximately -0.287. This indicates that, while holding all other variables constant, a one-unit increase in 'Unit Price' is associated with a decrease of approximately 0.287 units in 'Revenue'. This suggests that higher unit prices may lead to lower revenue.

These interpretations provide insights into how each independent variable influences the dependent variable 'Revenue' in our linear regression model. The interpretations assume a linear relationship between the variables, and the coefficients represent the change in 'Revenue' for a one-unit change in each independent variable while keeping the other variables constant.