A capstone project involves applying your knowledge to analyze a given dataset. You will conduct extensive research, use critical thinking, and apply practical skills to derive meaningful insights and solutions. This project will demonstrate your expertise in data analysis and your ability to tackle real-world problems.

Import library

+ Code + Text

```
import pandas as pd
```

Task 1: Load the dataset

```
df = pd.read_csv('/content/E_Commerce (1).csv')
```

1. Data Exploration and Cleaning

Task 2: Display basic statistics

```
print("Summary Statistics:\n")
print(df.describe(include='all').transpose())
```

```
Summary Statistics:

                      count unique   top  freq         mean          std  \
ID                  10999.0    NaN   NaN   NaN       5500.0   3175.28214
Warehouse_block       10999      5     F  3666          NaN          NaN
Mode_of_Shipment      10999      3  Ship  7462          NaN          NaN
Customer_care_calls 10999.0    NaN   NaN   NaN     4.054459      1.14149
Customer_rating     10999.0    NaN   NaN   NaN     2.990545     1.413603
Cost_of_the_Product 10999.0    NaN   NaN   NaN   210.196836    48.063272
Prior_purchases     10999.0    NaN   NaN   NaN     3.567597      1.52286
Product_importance    10999      3   low  5297          NaN          NaN
Gender                10999      2     F  5545          NaN          NaN
Discount_offered    10999.0    NaN   NaN   NaN    13.373216    16.205527
Weight_in_gms       10999.0    NaN   NaN   NaN  3634.016729  1635.377251
Reached.on.Time_Y.N 10999.0    NaN   NaN   NaN     0.596691     0.490584

                       min     25%     50%     75%      max
ID                     1.0  2750.5  5500.0  8249.5  10999.0
Warehouse_block        NaN     NaN     NaN     NaN      NaN
Mode_of_Shipment       NaN     NaN     NaN     NaN      NaN
Customer_care_calls    2.0     3.0     4.0     5.0      7.0
Customer_rating        1.0     2.0     3.0     4.0      5.0
Cost_of_the_Product   96.0   169.0   214.0   251.0    310.0
Prior_purchases        2.0     3.0     3.0     4.0     10.0
Product_importance     NaN     NaN     NaN     NaN      NaN
Gender                 NaN     NaN     NaN     NaN      NaN
Discount_offered       1.0     4.0     7.0    10.0     65.0
Weight_in_gms       1001.0  1839.5  4149.0  5050.0   7846.0
Reached.on.Time_Y.N    0.0     0.0     1.0     1.0      1.0
```

Compute mode separately (optional but useful)

```
print("\nMode of each column:\n")
print(df.mode().iloc[0])
```

Task 3: Handling missing values

```
print("\nMissing Values:\n")
print(df.isnull().sum())
```

```
Missing Values:

ID                     0
```

```
Warehouse_block        0
Mode_of_Shipment       0
Customer_care_calls    0
Customer_rating        0
Cost_of_the_Product    0
Prior_purchases        0
Product_importance     0
Gender                 0
Discount_offered       0
Weight_in_gms          0
Reached.on.Time_Y.N    0
dtype: int64
```

Example: Filling missing values

You can choose mean, median, or mode based on the column type

```
df.fillna({
    'Customer_care_calls': df['Customer_care_calls'].mode()[0],
    'Customer_rating': df['Customer_rating'].mode()[0],
    # Add more if necessary
}, inplace=True)
```

Task 4: Remove duplicate rows

```
duplicates = df.duplicated().sum()
print(f"\nNumber of duplicate rows: {duplicates}")
df.drop_duplicates(inplace=True)
```

Label Encoding for 'Gender' (binary category)

```
df['Gender'] = df['Gender'].map({'M': 0, 'F': 1})
```

One-Hot Encoding for other categorical columns

```
df = pd.get_dummies(df, columns=['Warehouse_block', 'Mode_of_Shipment', 'Product_importance'], drop_first=True)
```

2. Data Visualization

```
pip install matplotlib seaborn
```

```
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages (3.10.0)
Requirement already satisfied: seaborn in /usr/local/lib/python3.11/dist-packages (0.13.2)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.3.2)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (4.58.1)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.4.8)
Requirement already satisfied: numpy>=1.23 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (2.0.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (24.2)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (11.2.1)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (3.2.3)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (2.9.0.post0)
Requirement already satisfied: pandas>=1.2 in /usr/local/lib/python3.11/dist-packages (from seaborn) (2.2.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.2->seaborn) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.2->seaborn) (2025.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.7->matplotlib) (1.17.0)
```

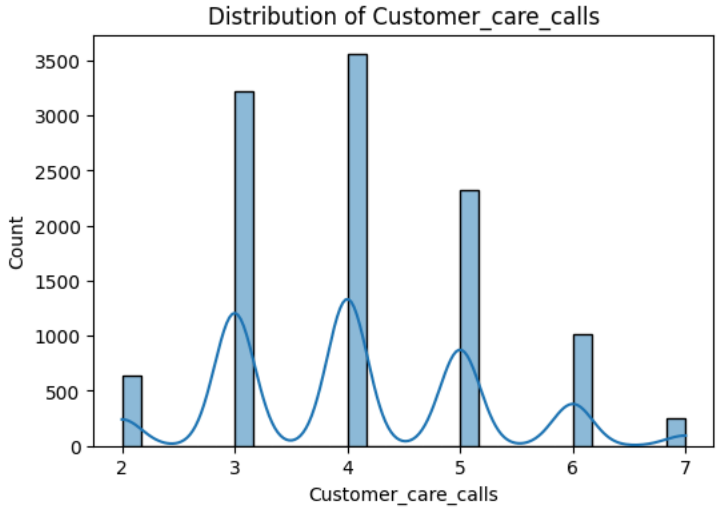
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

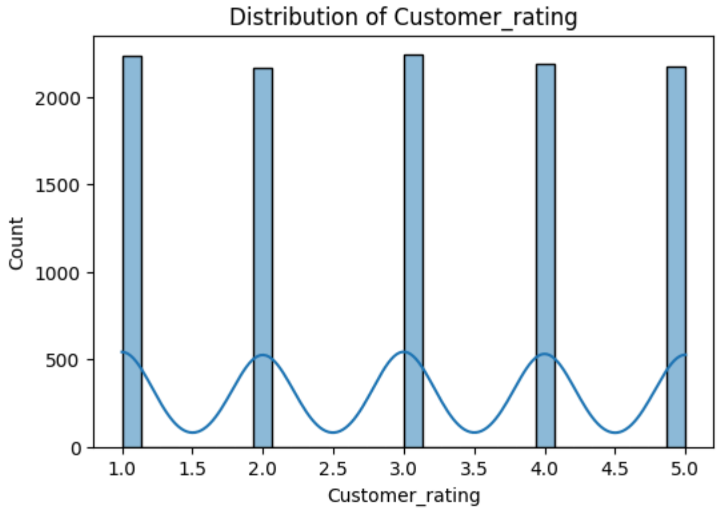
Task 6: Numerical Feature Distributions

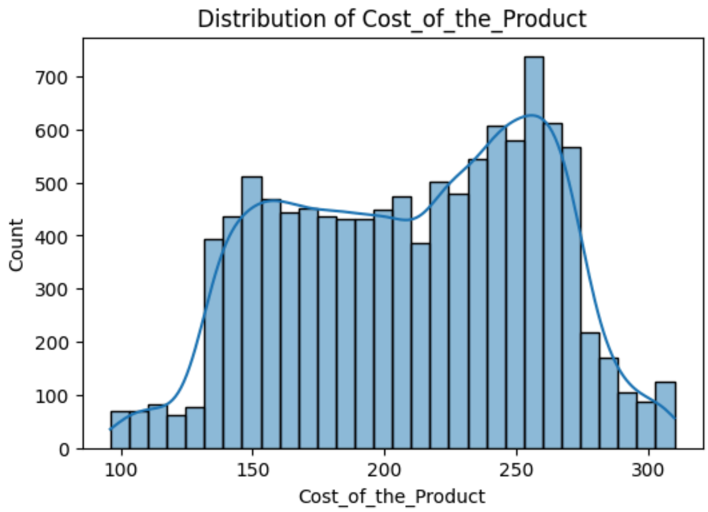
```
numerical_cols = ['Customer_care_calls', 'Customer_rating', 'Cost_of_the_Product',
                  'Prior_purchases', 'Discount_offered', 'Weight_in_gms']
```

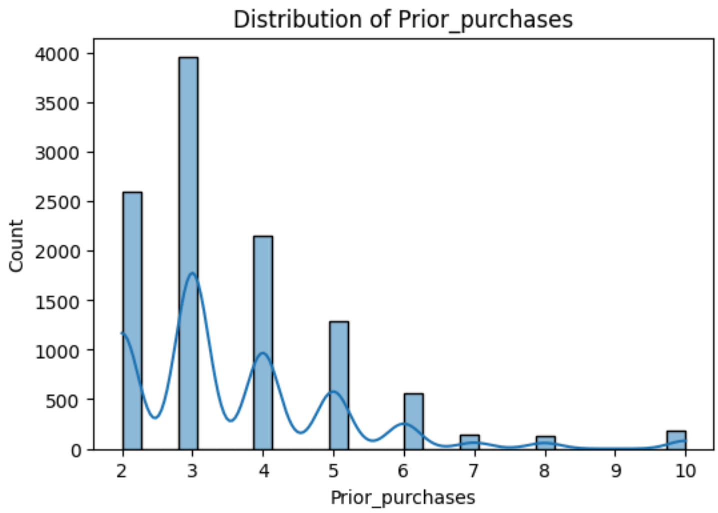
Histograms

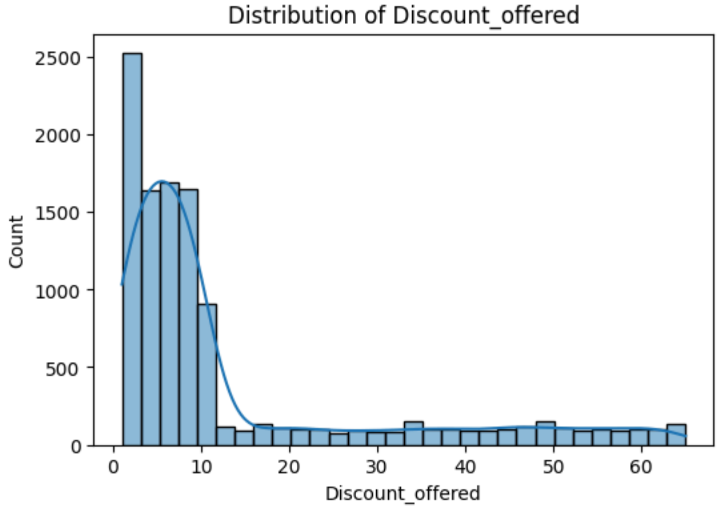
```
for col in numerical_cols:
    plt.figure(figsize=(6, 4))
    sns.histplot(df[col], kde=True, bins=30)
    plt.title(f'Distribution of {col}')
for col in numerical_cols:
    plt.figure(figsize=(6, 4))
    sns.histplot(df[col], kde=True, bins=30)
    plt.title(f'Distribution of {col}')
    plt.xlabel(col)
    plt.ylabel('Frequency')
    plt.tight_layout()
    plt.show()
```

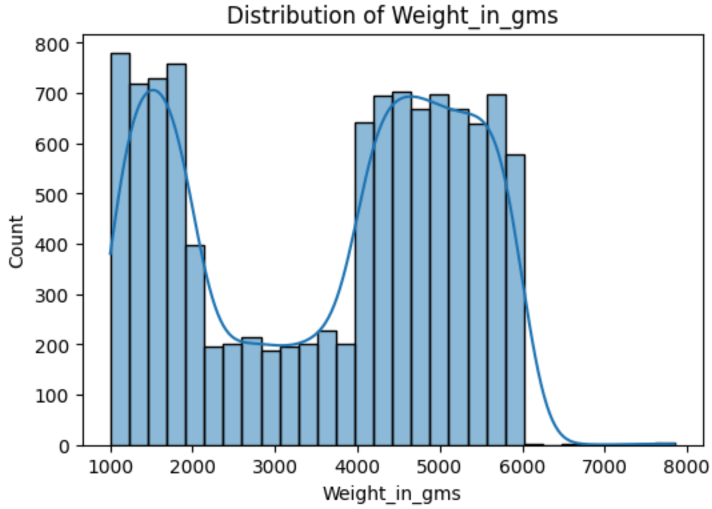

Distribution of Customer_care_calls
Count
Customer_care_calls

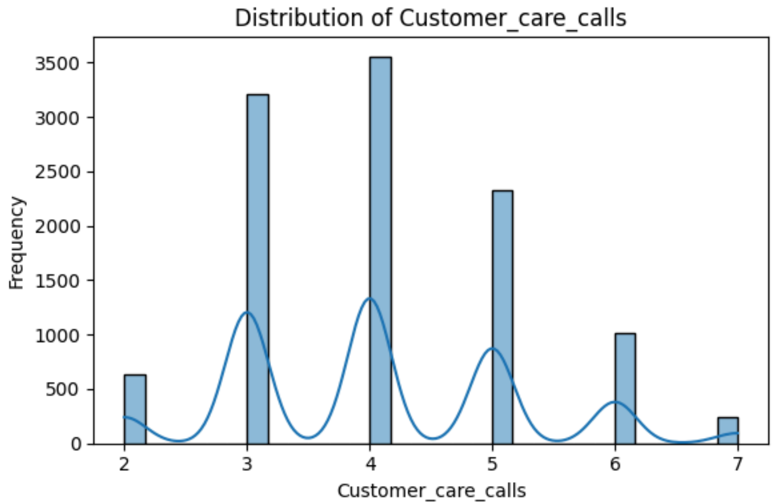

Distribution of Customer_rating
Count
Customer_rating

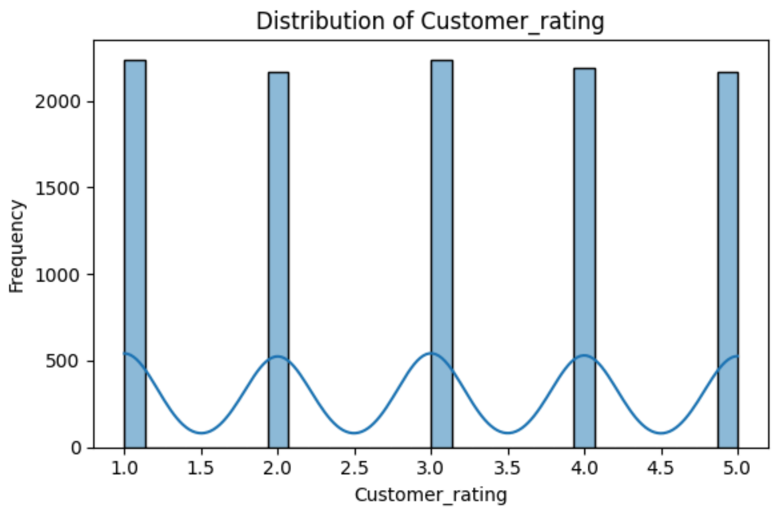

Distribution of Cost_of_the_Product
Count
Cost_of_the_Product

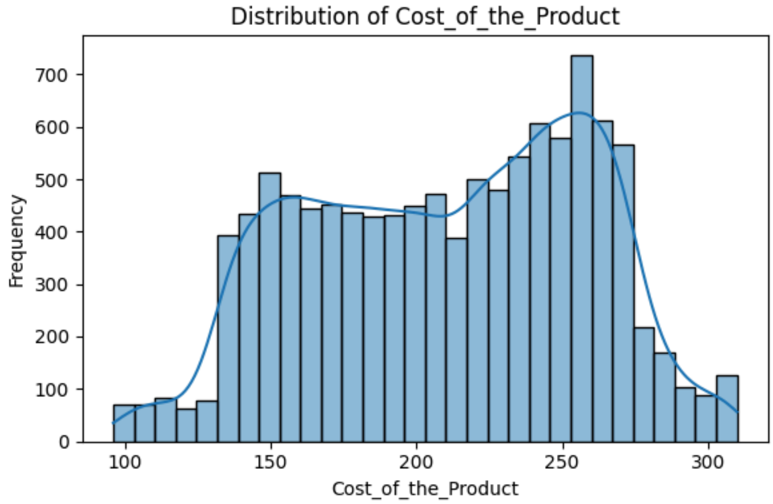

Distribution of Prior_purchases
Count
Prior_purchases

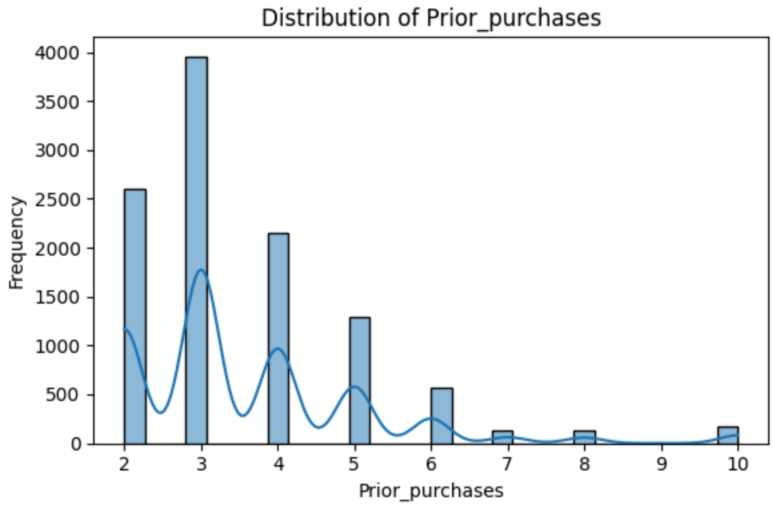

Distribution of Discount_offered
Count
Discount_offered
2500
2000
1500
1000
500
0
0
10
20
30
40
50
60

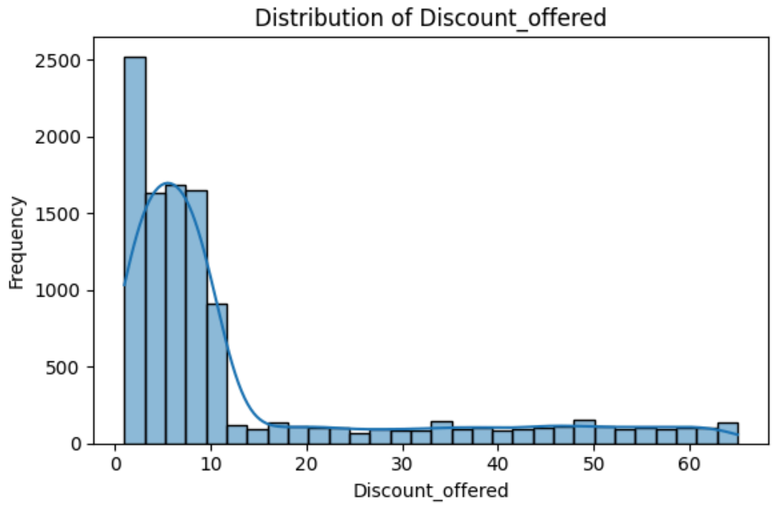

Distribution of Weight_in_gms
Count
Weight_in_gms
800
700
600
500
400
300
200
100
0
1000
2000
3000
4000
5000
6000
7000
8000

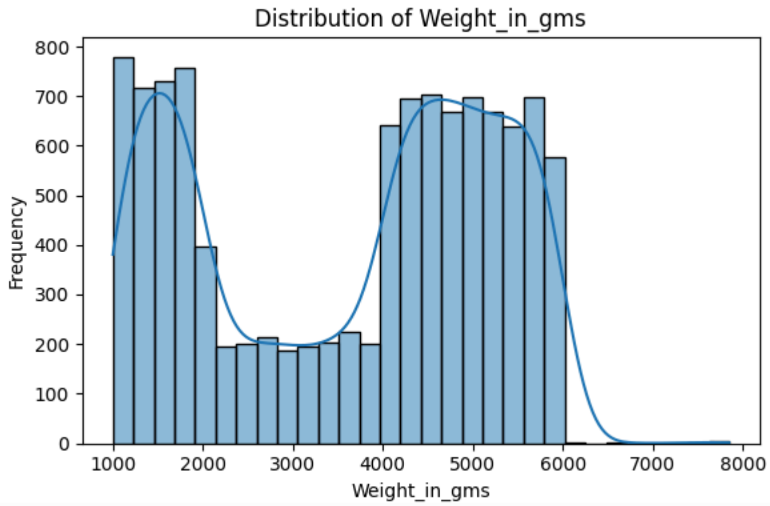

Distribution of Customer_care_calls
Frequency
Customer_care_calls
3500
3000
2500
2000
1500
1000
500
0
2
3
4
5
6
7


Distribution of Customer_rating
Frequency
Customer_rating
2000
1500
1000
500
0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0


Distribution of Cost_of_the_Product
Frequency
0
100
200
300
400
500
600
700
100
150
200
250
300
Cost_of_the_Product

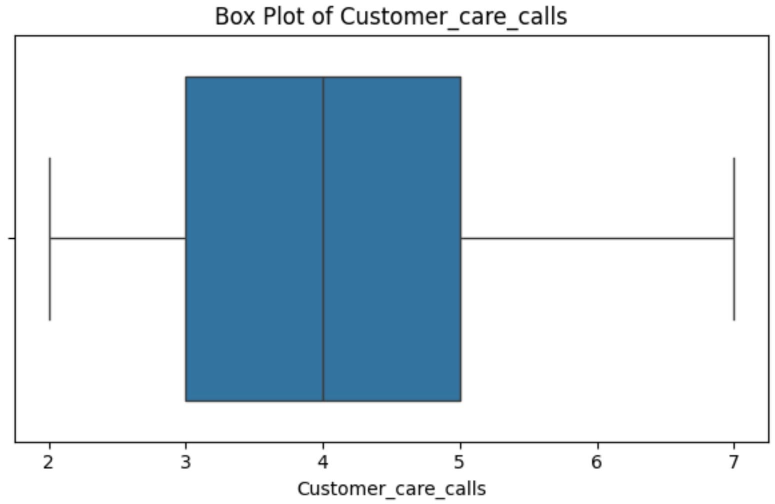

Distribution of Prior_purchases
Frequency
0
500
1000
1500
2000
2500
3000
3500
4000
2
3
4
5
6
7
8
9
10
Prior_purchases

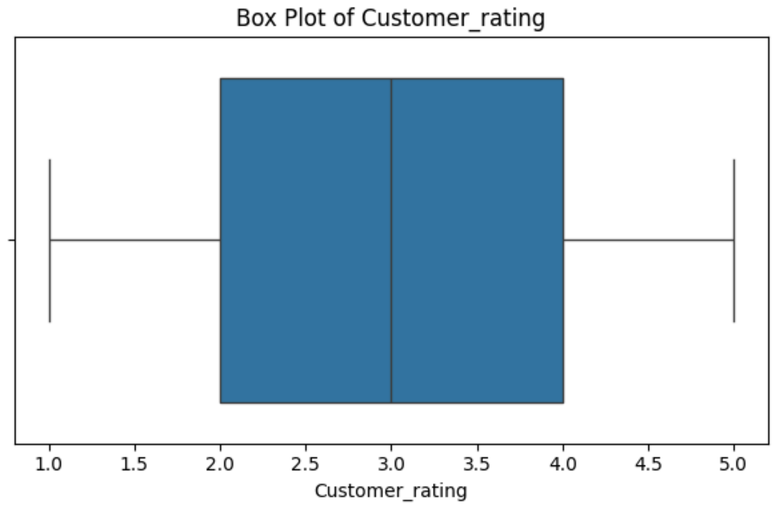

Distribution of Discount_offered
Frequency
0
500
1000
1500
2000
2500
0
10
20
30
40
50
60
Discount_offered

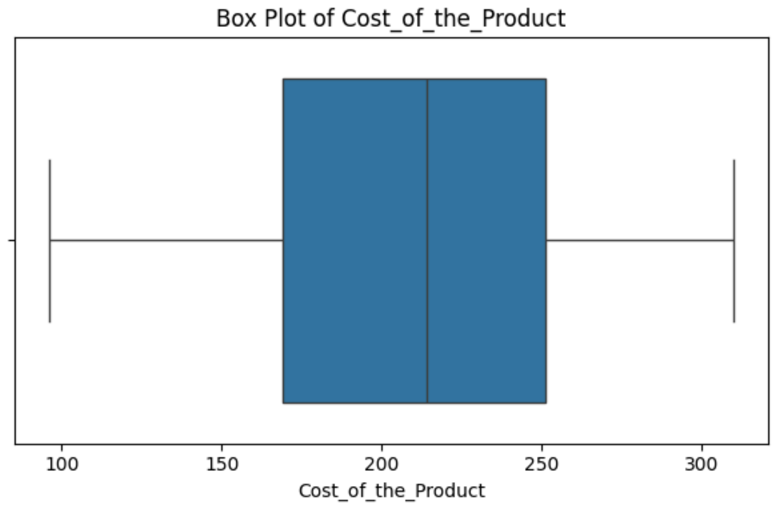

Distribution of Weight_in_gms
Frequency
0
100
200
300
400
500
600
700
800
1000
2000
3000
4000
5000
6000
7000
8000
Weight_in_gms

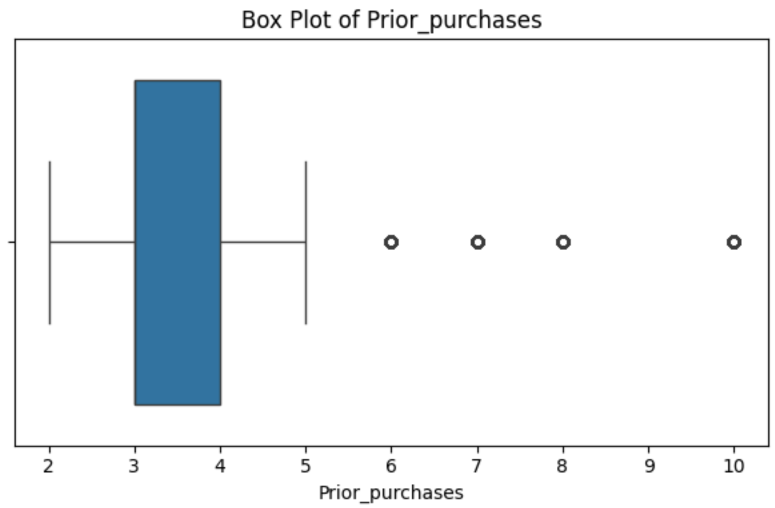
Box Plots

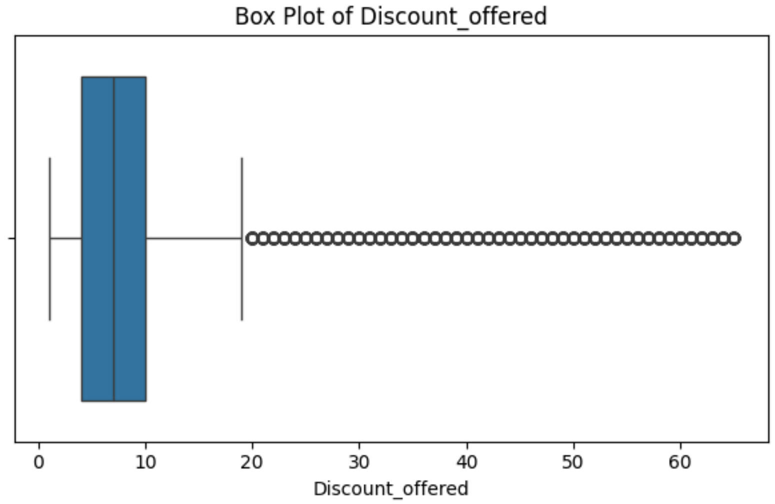
```
for col in numerical_cols:
    plt.figure(figsize=(6, 4))
    sns.boxplot(x=df[col])
    plt.title(f'Box Plot of {col}')
    plt.tight_layout()
    plt.show()
```

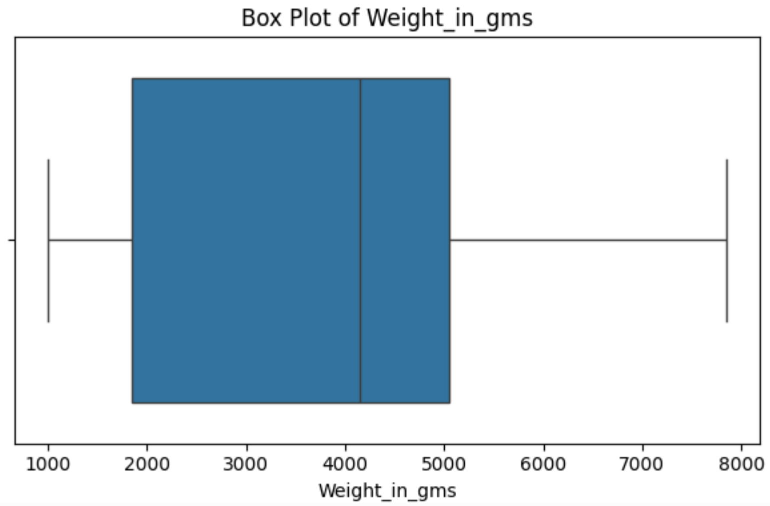

Box Plot of Customer_care_calls
2
3
4
5
6
7
Customer_care_calls


Box Plot of Customer_rating
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Customer_rating


Box Plot of Cost_of_the_Product
100
150
200
250
300
Cost_of_the_Product

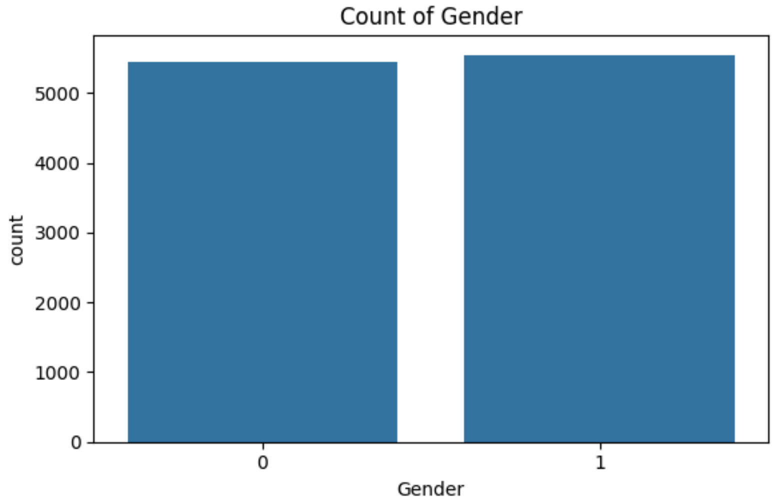

Box Plot of Prior_purchases
2
3
4
5
6
7
8
9
10
Prior_purchases

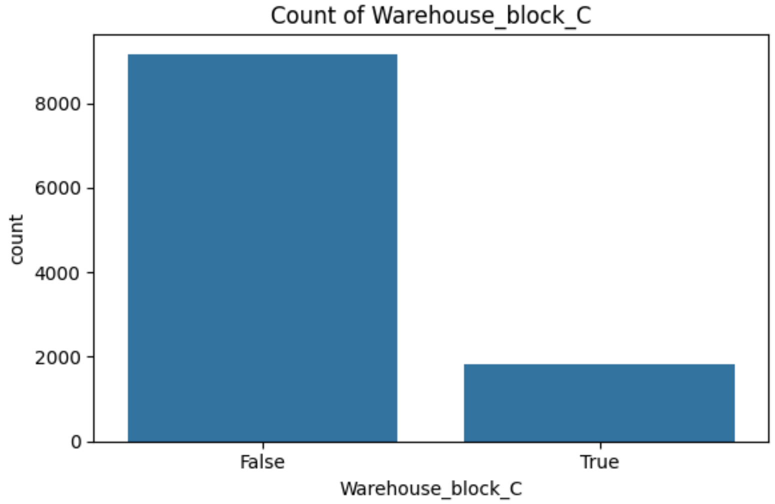

Box Plot of Discount_offered
0
10
20
30
40
50
60
Discount_offered

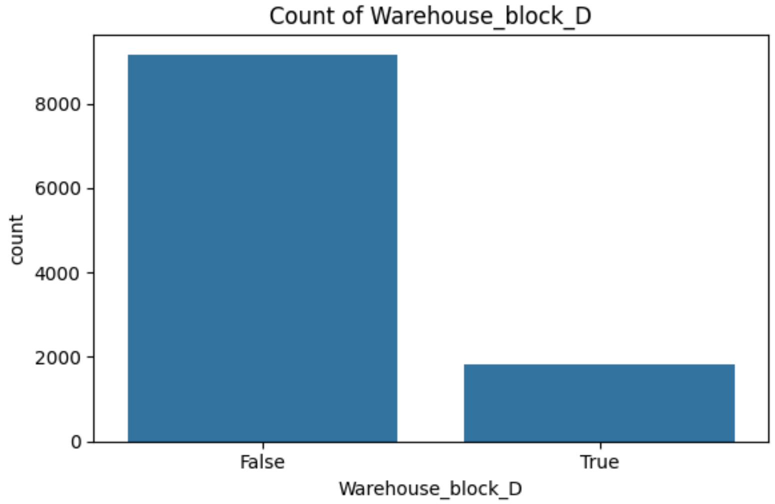

Box Plot of Weight_in_gms
1000
2000
3000
4000
5000
6000
7000
8000
Weight_in_gms

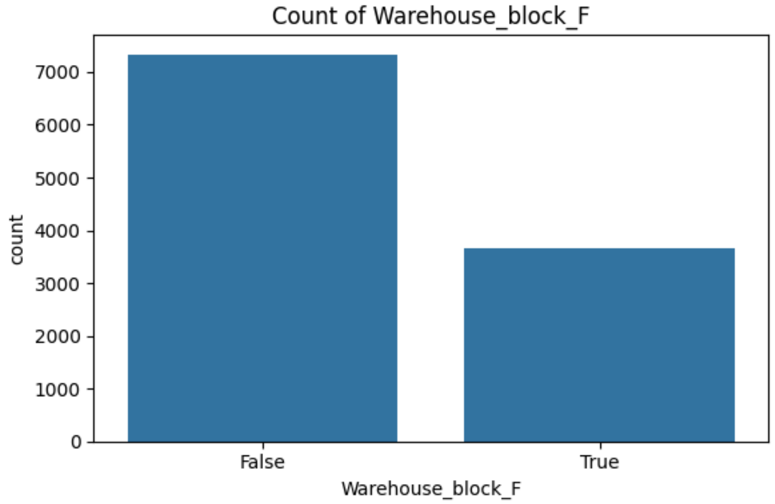
Bar Charts

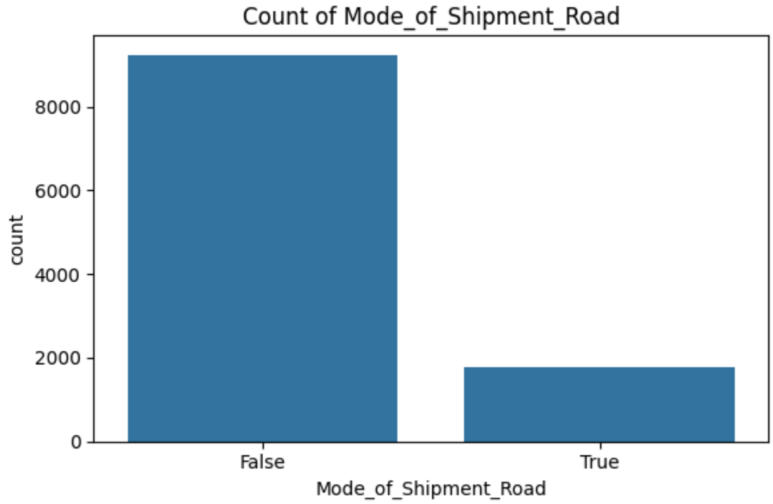
```
# %% [markdown]
# # Bar Charts
# %%

categorical_cols = ['Warehouse_block', 'Mode_of_Shipment', 'Product_importance', 'Gender']
# You need to define original_df if you intend to use it.
# If you want to use the modified df, change original_df to df.
# Assuming you want to use the modified df after one-hot encoding:
for col in ['Gender', 'Warehouse_block_C', 'Warehouse_block_D', 'Warehouse_block_F', 'Mode_of_Shipment_Road', 'Mode_of_Shipment_Ship',
    if col in df.columns: # Check if the column exists after one-hot encoding
        plt.figure(figsize=(6, 4))
        # Use the processed df for plotting the new categorical columns
        sns.countplot(data=df, x=col)
        plt.title(f'Count of {col}')
        plt.tight_layout()
        plt.show()
```

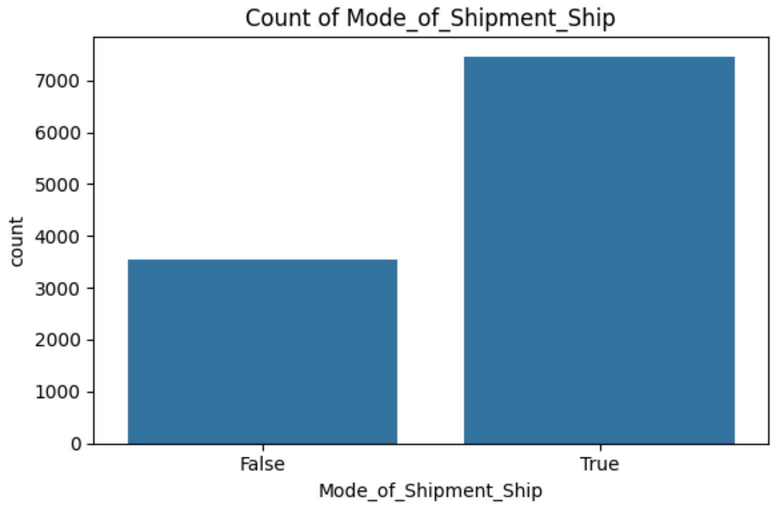

Count of Gender
count
5000
4000
3000
2000
1000
0
0
1
Gender

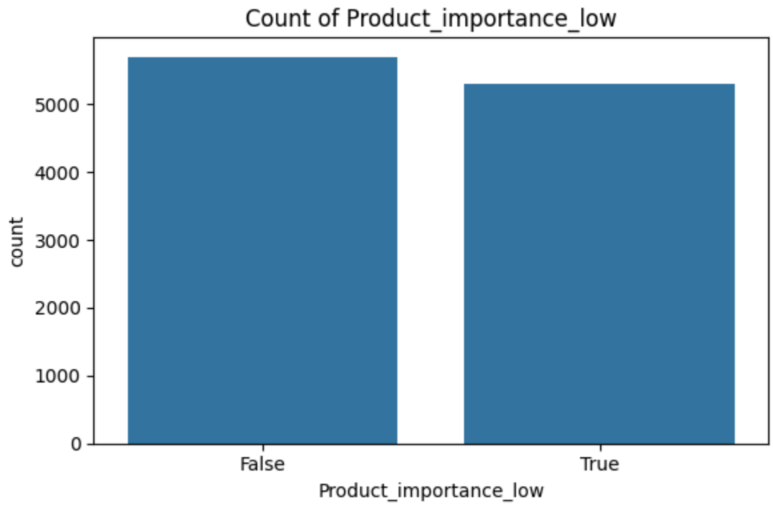

Count of Warehouse_block_C
count
8000
6000
4000
2000
0
False
True
Warehouse_block_C

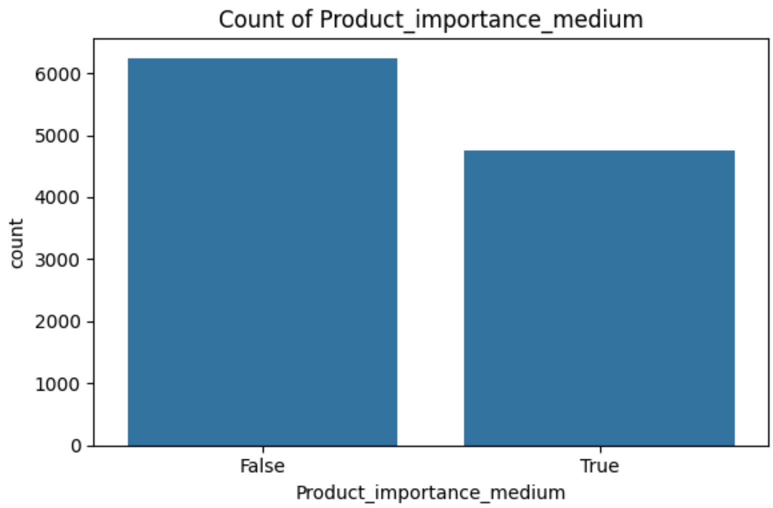

Count of Warehouse_block_D
count
8000
6000
4000
2000
0
False
True
Warehouse_block_D


Count of Warehouse_block_F
count
7000
6000
5000
4000
3000
2000
1000
0
False
True
Warehouse_block_F


Count of Mode_of_Shipment_Road
count
0
2000
4000
6000
8000
False
True
Mode_of_Shipment_Road

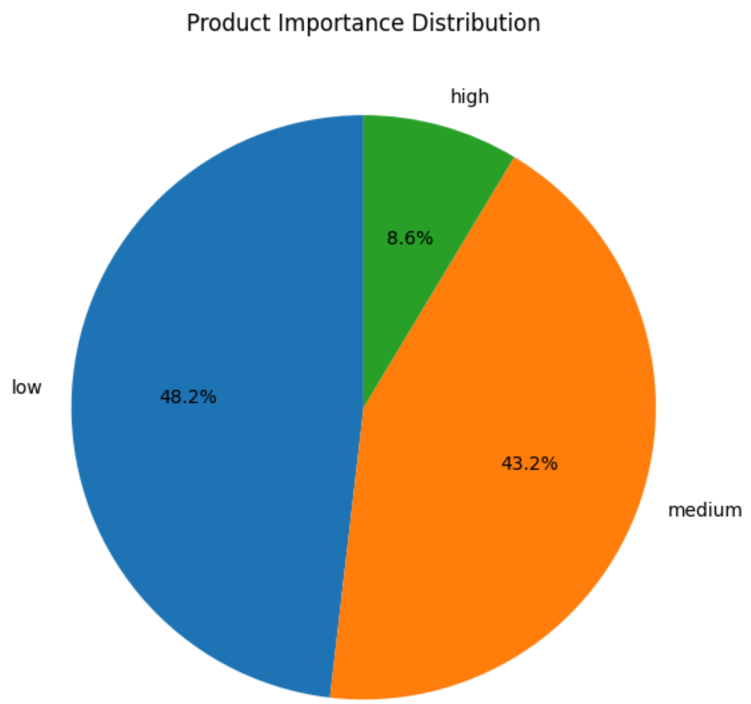

Count of Mode_of_Shipment_Ship
count
0
1000
2000
3000
4000
5000
6000
7000
False
True
Mode_of_Shipment_Ship


Count of Product_importance_low
count
0
1000
2000
3000
4000
5000
False
True
Product_importance_low


Count of Product_importance_medium
count
0
1000
2000
3000
4000
5000
6000
False
True
Product_importance_medium

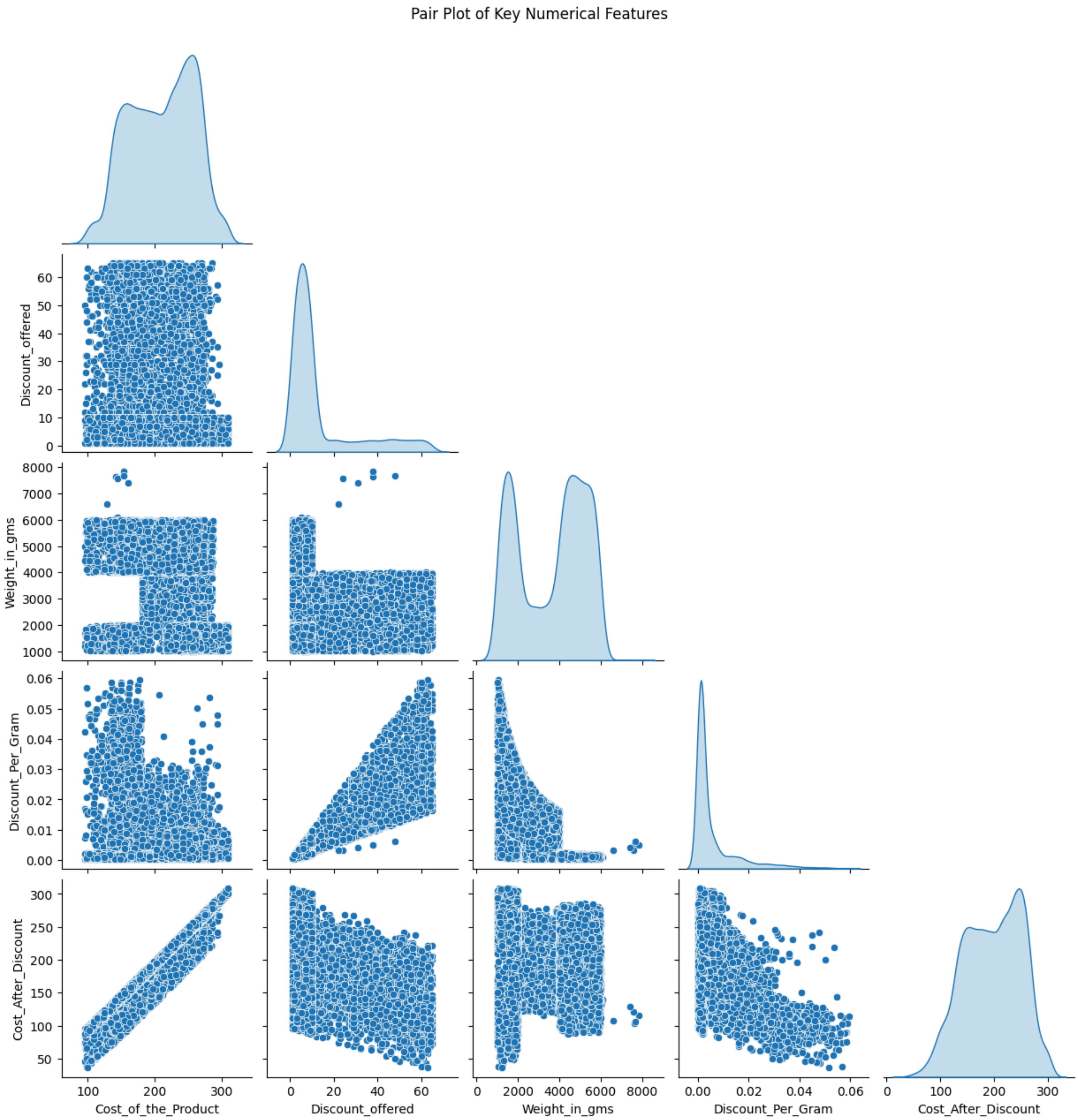
Pie Chart for Product Importance

```
plt.figure(figsize=(6, 6))
original_df['Product_importance'].value_counts().plot.pie(autopct='%1.1f%%', startangle=90)
plt.title('Product Importance Distribution')
plt.ylabel('')
plt.tight_layout()
plt.show()
```



Select numerical features to visualize

```
import seaborn as sns
import matplotlib.pyplot as plt
pairplot_cols = ['Cost_of_the_Product', 'Discount_offered', 'Weight_in_gms',
                 'Discount_Per_Gram', 'Cost_After_Discount']

# Create pair plot
# Corrected variable name from ecom_df to df
sns.pairplot(df[pairplot_cols], diag_kind='kde', corner=True)
plt.suptitle('Pair Plot of Key Numerical Features', y=1.02)
plt.show()
```



3. Feature Engineering

Task 10: Feature Engineering

Task 11: Standardization

```
# Import StandardScaler
from sklearn.preprocessing import StandardScaler
```

```
# Task 10: Feature Engineering
# Changed ecom_df to df
df['Discount_Per_Gram'] = df['Discount_offered'] / df['Weight_in_gms']
df['Cost_After_Discount'] = df['Cost_of_the_Product'] - df['Discount_offered']
df['Is_High_Value'] = (df['Cost_of_the_Product'] > 200).astype(int)

# Task 11: Standardization
scaler = StandardScaler()
cols_to_scale = ['Cost_of_the_Product', 'Discount_offered', 'Weight_in_gms']
# Changed ecom_df to df
df[[col + '_scaled' for col in cols_to_scale]] = scaler.fit_transform(df[cols_to_scale]

# Preview the new DataFrame
# Changed ecom_df to df
df[['Cost_of_the_Product', 'Discount_offered', 'Weight_in_gms',
         'Discount_Per_Gram', 'Cost_After_Discount', 'Is_High_Value'] +
```

	Cost_of_the_Product	Discount_offered	Weight_in_gms	Discount_Per_Gram	Cost_A ter_Discount	Is_High_Value	Cost_of_the_Product_
0	177	44	1233	0.035685	133	0	-0
1	216	59	3088	0.019106	157	1	0
2	183	48	3374	0.014226	135	0	-0
3	176	10	1177	0.008496	166	0	-0
4	184	46	2484	0.018519	138	0	-0

```
import pandas as pd
from sklearn.preprocessing import StandardScaler

# Load data (only needed if not already loaded)
# ecom_df = pd.read_csv("E_Commerce.csv")
```

4. Model Building

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, mean_squared_error
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier


# Define features (X) and target (y)
# The target variable is likely 'Reached.on.Time_Y.N'. Let's assume it is.
# We need to drop the original categorical columns and the target variable itself from features.
X = df.drop(columns=['ID', 'Reached.on.Time_Y.N', 'Warehouse_block', 'Mode_of_Shipment', 'Product_importance'])
y = df['Reached.on.Time_Y.N']


# Task 12: Split the dataset into training and testing sets
```