Capstone Project

Problem Statement

The goal of this project is to analyze a dataset from an international e-commerce company to identify patterns and causes behind late deliveries. The dataset includes shipment mode, customer service details, product importance, and delivery times. By investigating this data, we aim to reduce delivery delays and improve the supply chain process.

Objective

- Clean and preprocess the dataset for analysis and visualization
- Create meaningful visualizations using Tableau
- Identify major reasons behind late deliveries
- Train classification models to predict delivery success
- Recommend operational changes based on findings

Data Description

The dataset consists of 10,999 entries, each representing an order with attributes such as:

- ID: Order ID

- Gender: Customer gender
- Warehouse_block: Warehouse segment
- Mode_of_Shipment: Shipment method (Flight, Ship, Road)
- Product_importance: Importance level of the product
- Customer_rating: Rating given by customer (1–5)
- Cost_of_the_Product: Price of the product
- Discount_offered: Discounts applied
- Weight_in_gms: Product weight
- Reached.on.Time_Y.N: Target variable (0=On time, 1=Delayed)
- Additional engineered fields: Final_price, Discount_to_Weight

1. Data Exploration and Cleaning

A. Load the dataset into a pandas DataFrame and display the first few rows.

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

df = pd.read_csv('/Users/vishal/Desktop/Bootcamp 2025/E_Commerce.csv')

df.head()
```

```
   ID Warehouse_block Mode_of_Shipment  Customer_care_calls
Customer_rating  \
0   1               D           Flight                    4
2
1   2               F           Flight                    4
5
2   3               A           Flight                    2
2
3   4               B           Flight                    3
3
4   5               C           Flight                    2
2

   Cost_of_the_Product  Prior_purchases Product_importance Gender  \
0                  177                3                low      F
1                  216                2                low      M
2                  183                4                low      M
3                  176                4             medium      M
4                  184                3             medium      F

   Discount_offered  Weight_in_gms  Reached.on.Time_Y.N
0                44           1233                    1
1                59           3088                    1
2                48           3374                    1
```

```
3                10           1177                     1
4                46           2484                     1
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10999 entries, 0 to 10998
Data columns (total 12 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   ID                   10999 non-null  int64
 1   Warehouse_block      10999 non-null  object
 2   Mode_of_Shipment     10999 non-null  object
 3   Customer_care_calls  10999 non-null  int64
 4   Customer_rating      10999 non-null  int64
 5   Cost_of_the_Product  10999 non-null  int64
 6   Prior_purchases      10999 non-null  int64
 7   Product_importance   10999 non-null  object
 8   Gender               10999 non-null  object
 9   Discount_offered     10999 non-null  int64
 10  Weight_in_gms        10999 non-null  int64
 11  Reached.on.Time_Y.N  10999 non-null  int64
dtypes: int64(8), object(4)
memory usage: 1.0+ MB
```

B.Summarize the dataset by providing basic statistics (mean, median, mode, standard deviation, etc.).

```
df.describe()
```

```
                ID  Customer_care_calls  Customer_rating
Cost_of_the_Product  \
count  10999.00000         10999.000000     10999.000000
10999.000000
mean    5500.00000             4.054459         2.990545
210.196836
std     3175.28214             1.141490         1.413603
48.063272
min        1.00000             2.000000         1.000000
96.000000
25%     2750.50000             3.000000         2.000000
169.000000
50%     5500.00000             4.000000         3.000000
214.000000
75%     8249.50000             5.000000         4.000000
251.000000
max    10999.00000             7.000000         5.000000
310.000000
```

```
       Prior_purchases  Discount_offered  Weight_in_gms
Reached.on.Time_Y.N
count     10999.000000      10999.000000   10999.000000
10999.000000
mean          3.567597         13.373216    3634.016729
0.596691
std           1.522860         16.205527    1635.377251
0.490584
min           2.000000          1.000000    1001.000000
0.000000
25%           3.000000          4.000000    1839.500000
0.000000
50%           3.000000          7.000000    4149.000000
1.000000
75%           4.000000         10.000000    5050.000000
1.000000
max          10.000000         65.000000    7846.000000
1.000000
```

```python
# Median
df.median(numeric_only=True)
```

```
ID                     5500.0
Customer_care_calls       4.0
Customer_rating           3.0
Cost_of_the_Product     214.0
Prior_purchases           3.0
Discount_offered          7.0
Weight_in_gms          4149.0
Reached.on.Time_Y.N       1.0
dtype: float64
```

```python
# Mode (can return multiple values if tie)
print("Mode of the numerical Value")
df.mode(numeric_only=True).iloc[0]
```

```
Mode of the numerical Value

ID                        1.0
Customer_care_calls       4.0
Customer_rating           3.0
Cost_of_the_Product     245.0
Prior_purchases           3.0
Discount_offered         10.0
Weight_in_gms          4883.0
Reached.on.Time_Y.N       1.0
Name: 0, dtype: float64
```

C. Identify and handle missing values. Explain the chosen method for handling them.

```
df.isnull().sum()
```

```
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
```

Observation:

This dataset has no missing values.

D. Identify and handle duplicate rows if any.

```
df.duplicated().sum()
```

```
0
```

Observation : There is no any duplicate values

E. Convert categorical variables to numerical values using appropriate encoding techniques (e.g., one-hot encoding, label encoding).

```
from sklearn.preprocessing import LabelEncoder

df_encoded = df.copy()
le = LabelEncoder()
df_encoded['Warehouse_block'] =
le.fit_transform(df_encoded['Warehouse_block'])
df_encoded['Mode_of_Shipment'] =
le.fit_transform(df_encoded['Mode_of_Shipment'])
df_encoded['Product_importance'] =
le.fit_transform(df_encoded['Product_importance'])
df_encoded['Gender'] = le.fit_transform(df_encoded['Gender'])
```

Obersvation from Data Exploration and Cleaning

1. No missing values or duplicate records were found.

2. Key numerical columns such as Customer_rating, Discount_offered, and Weight_in_gms have wide variation.

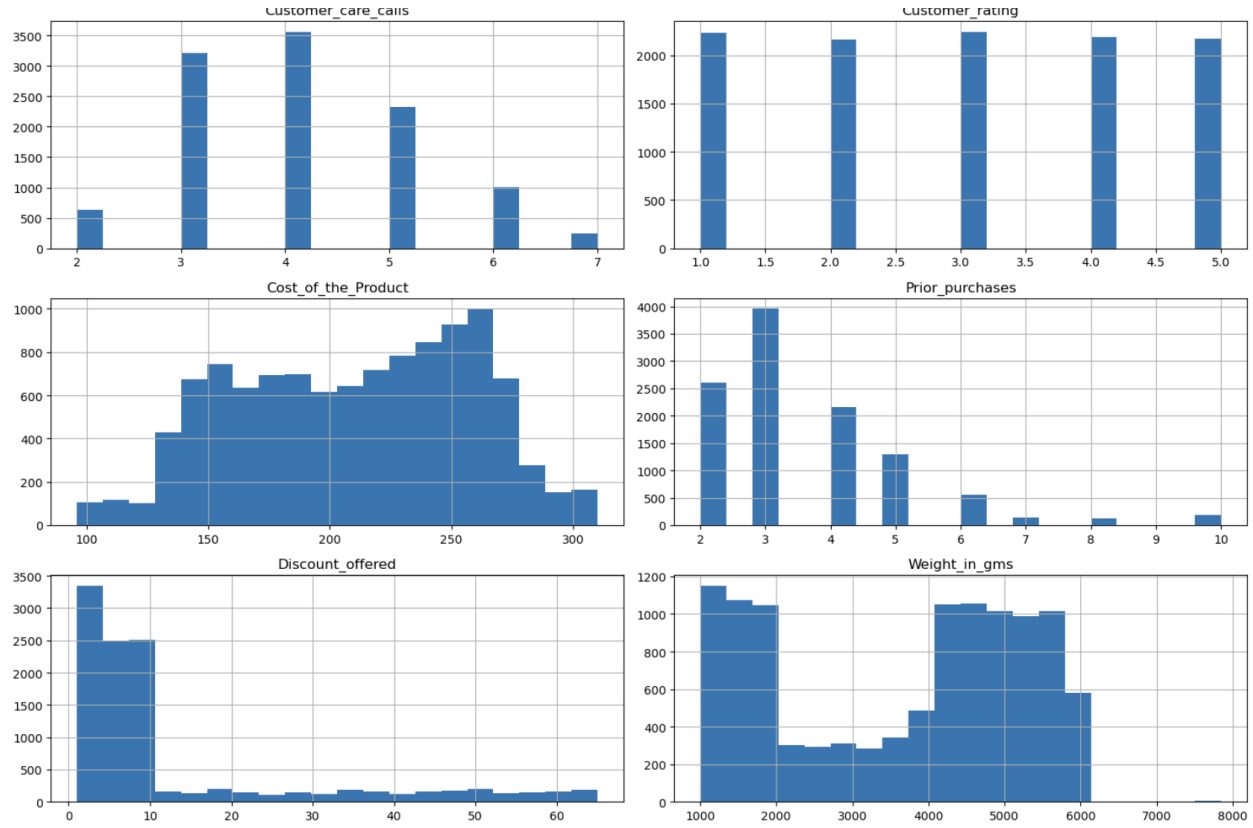
3. Mode of Customer_rating = 3, indicating most users gave average feedback.

4. Most products are low-cost, with mean around 210.

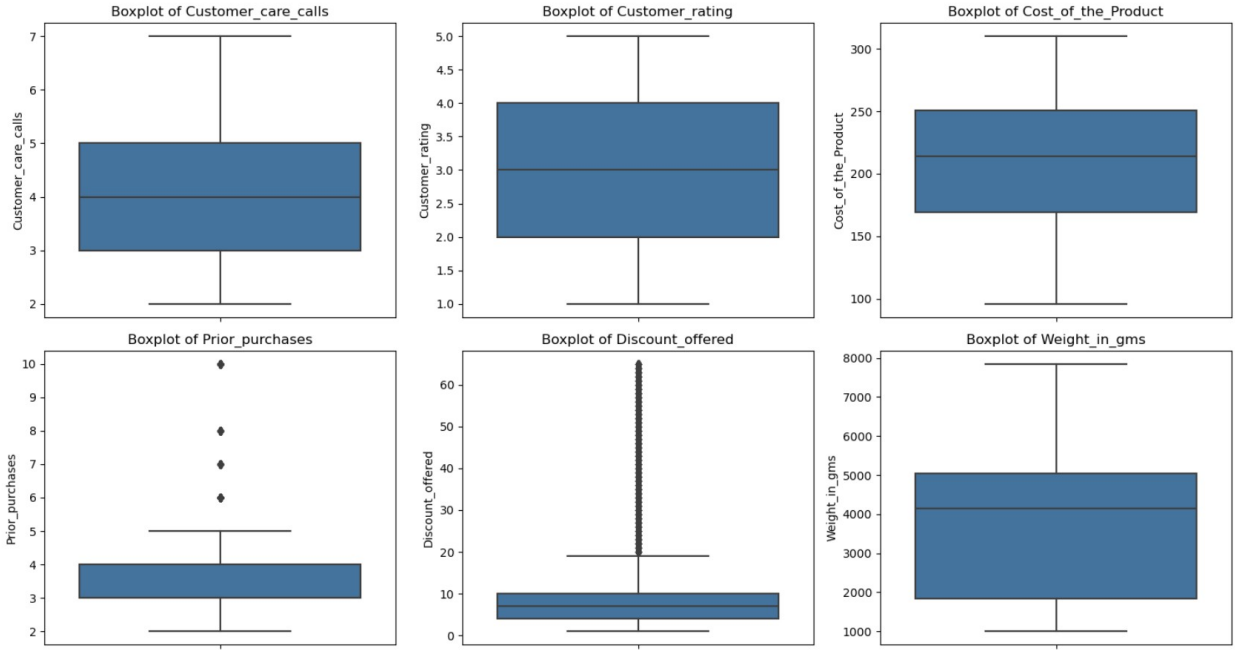
Initial exploration reveals a balanced dataset in terms of structure. Class imbalance is present in the target (Reached.on.Time_Y.N) with more delayed deliveries.

2. Data Visualization

A.Create visualizations to understand the distribution of numerical features (e.g.,histograms, box plots).



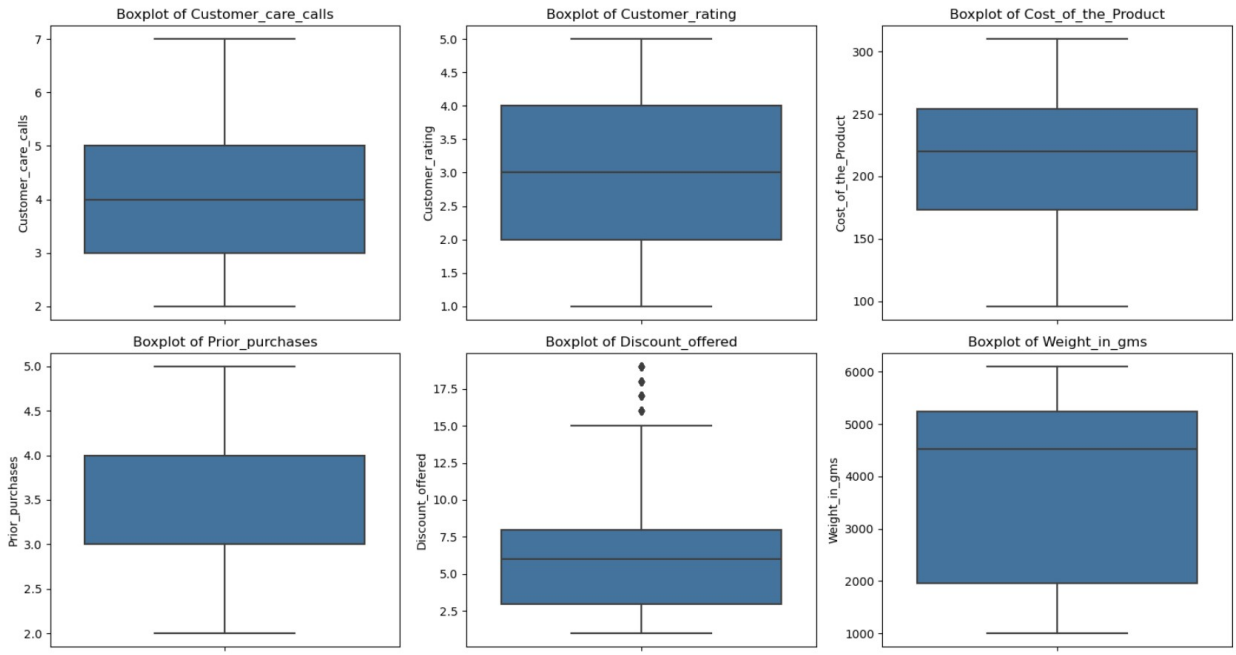
Observation: Histogram: Understand distribution (normal, skewed, etc.).



Observation: Boxplot: Detect outliers and data spread.

Removing Detected Outliers

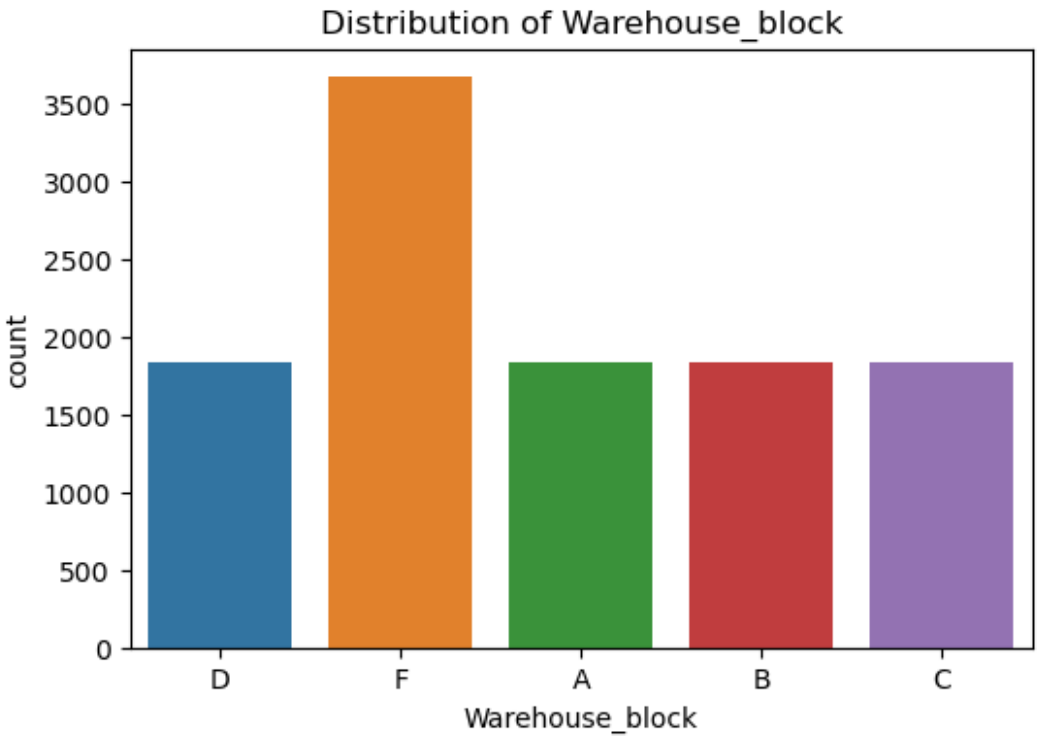
Replot the boxplots to verify outliers are removed

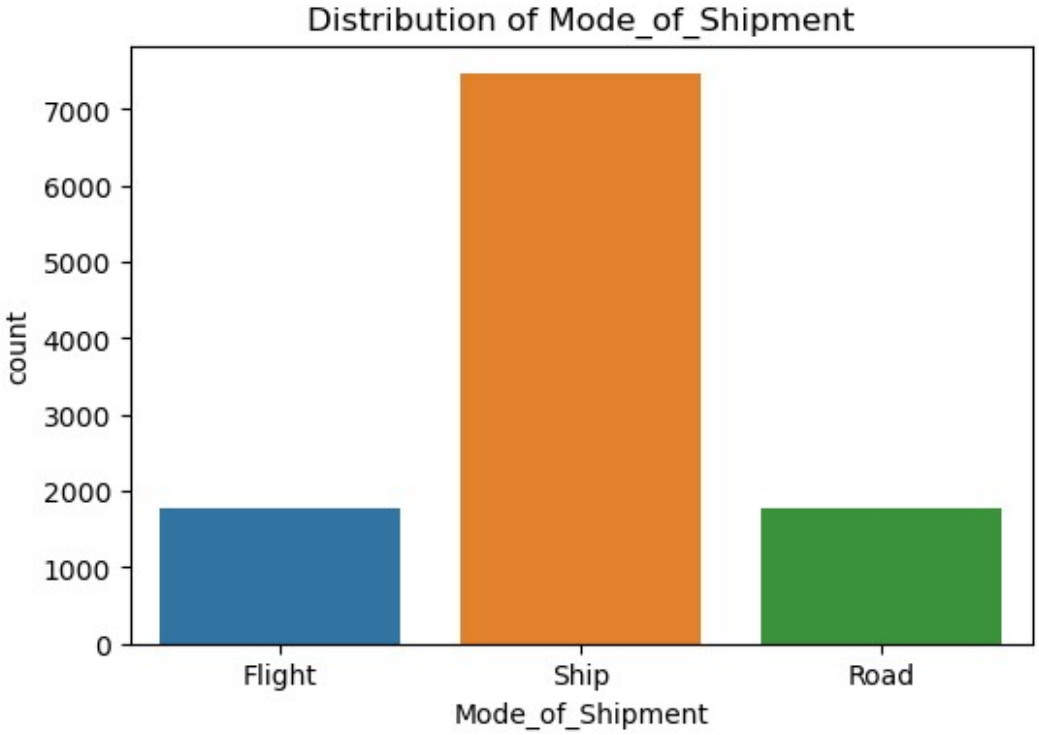


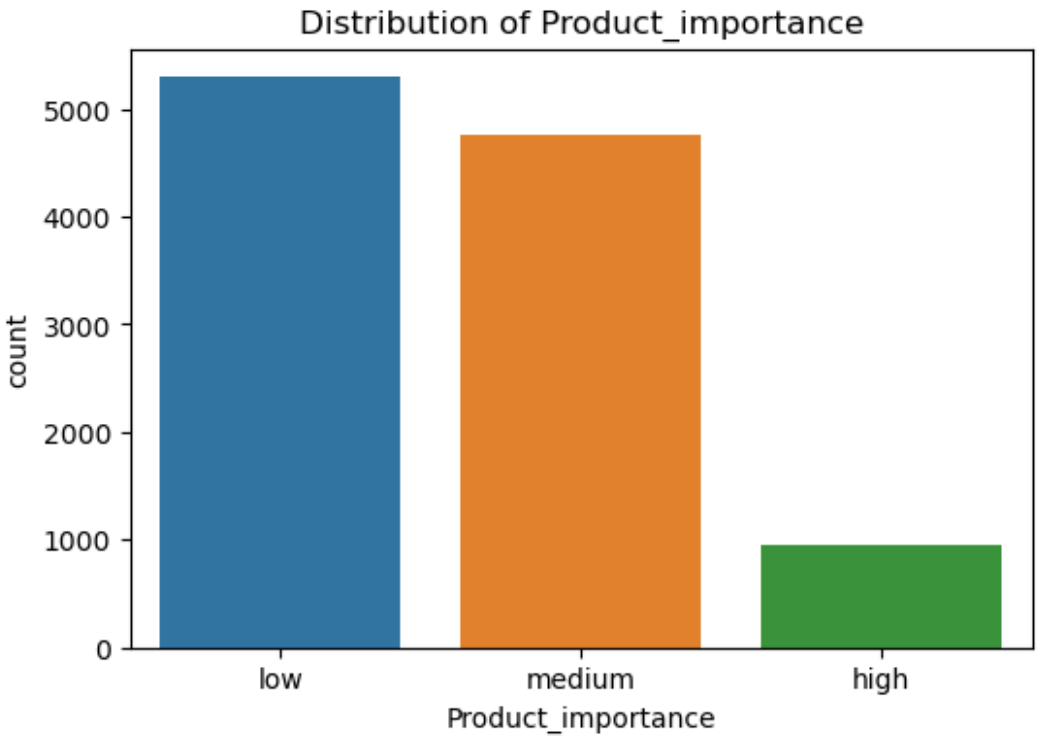
B. Create visualizations for categorical features (e.g., bar charts, pie charts).

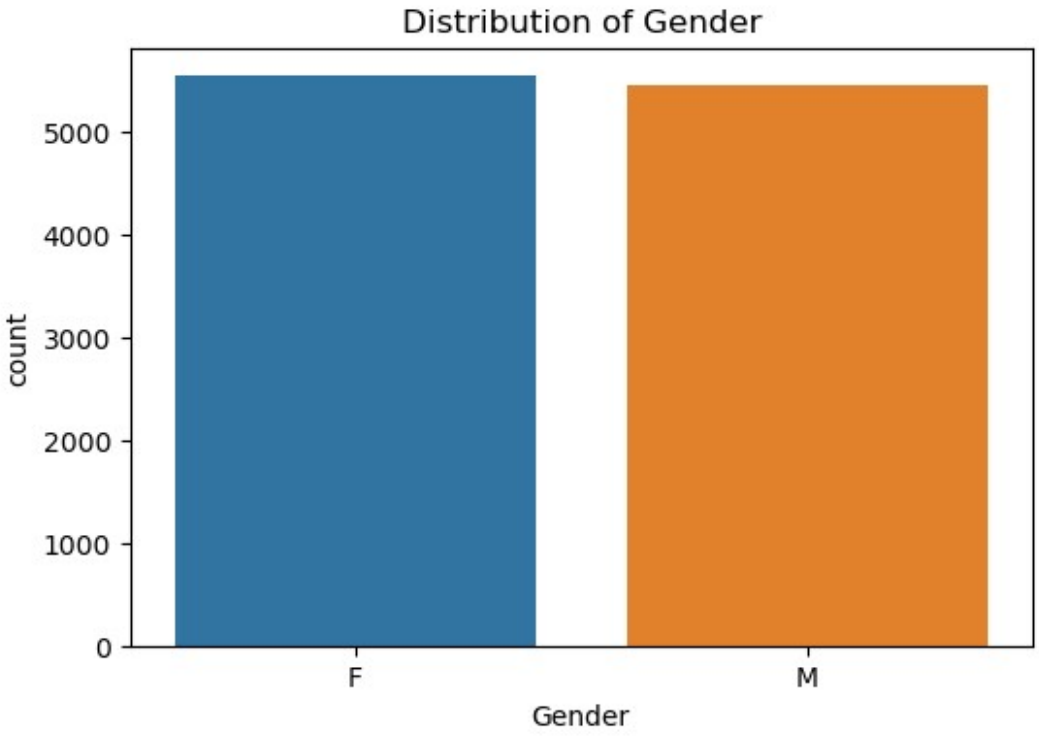
```
categorical_cols = ['Warehouse_block', 'Mode_of_Shipment',
'Product_importance', 'Gender']

for col in categorical_cols:
    plt.figure(figsize=(6,4))
    sns.countplot(data=df, x=col)
    plt.title(f'Distribution of {col}')
    plt.xticks(rotation=0)
    plt.show()
```


Distribution of Warehouse_block
count
3500
3000
2500
2000
1500
1000
500
0
D
F
A
B
C
Warehouse_block


Distribution of Mode_of_Shipment
count
7000
6000
5000
4000
3000
2000
1000
0
Flight
Ship
Road
Mode_of_Shipment


Distribution of Product_importance
count
5000
4000
3000
2000
1000
0
low
medium
high
Product_importance


Distribution of Gender
count
5000
4000
3000
2000
1000
0
F
M
Gender

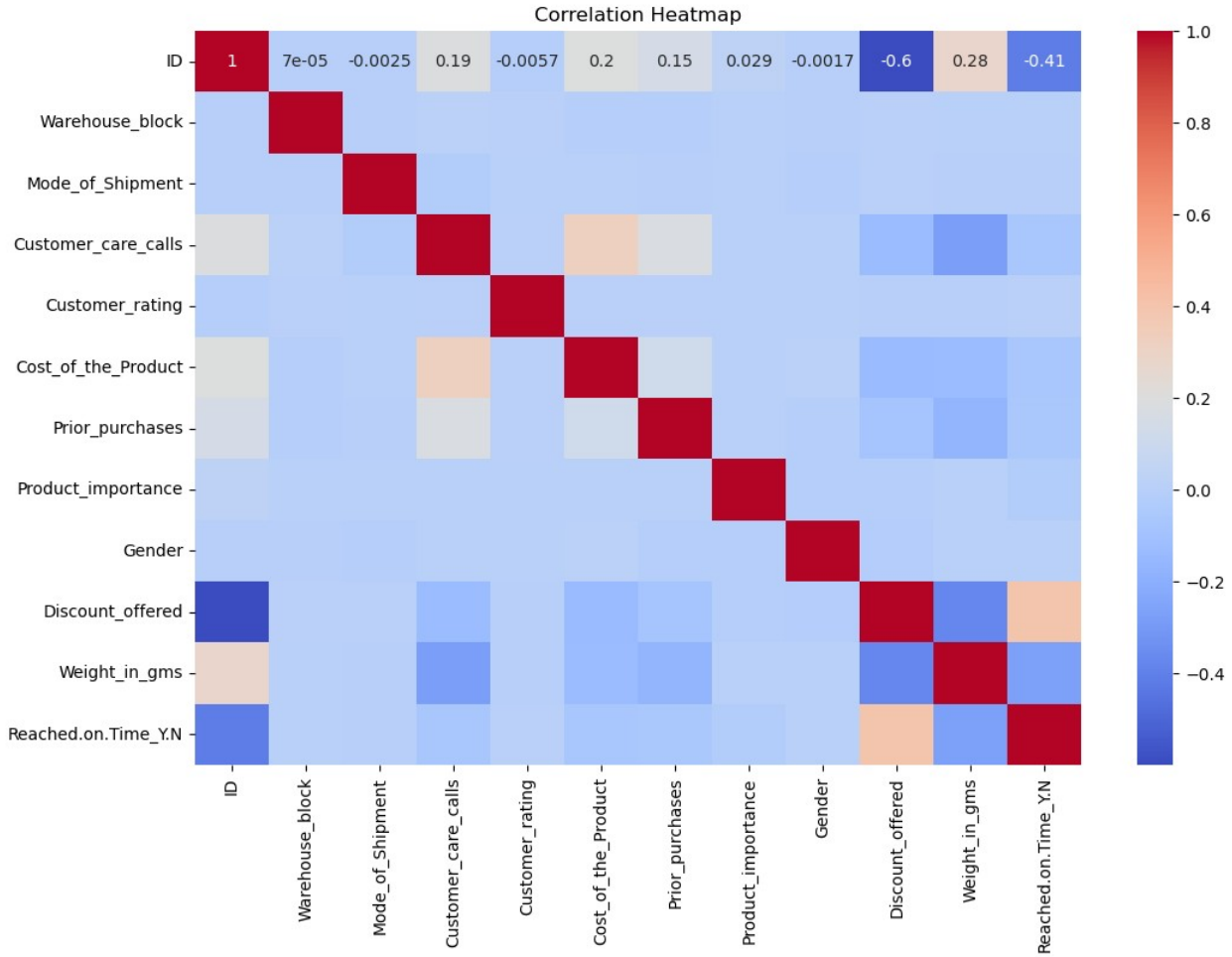
Observation:

Some blocks or shipping modes are more frequent.

Might indicate operational preferences.

C. Generate correlation heatmaps to identify relationships between numerical features.

```
plt.figure(figsize=(12, 8))
sns.heatmap(df_encoded.corr(), annot=True, cmap='coolwarm')
plt.title("Correlation Heatmap")
plt.show()
```



Observation:

1. No feature is strongly correlated with the target (Reached.on.Time_Y.N).

2. Highest negative correlation is with Discount_offered (−0.43) — suggesting higher discounts are linked with late deliveries.

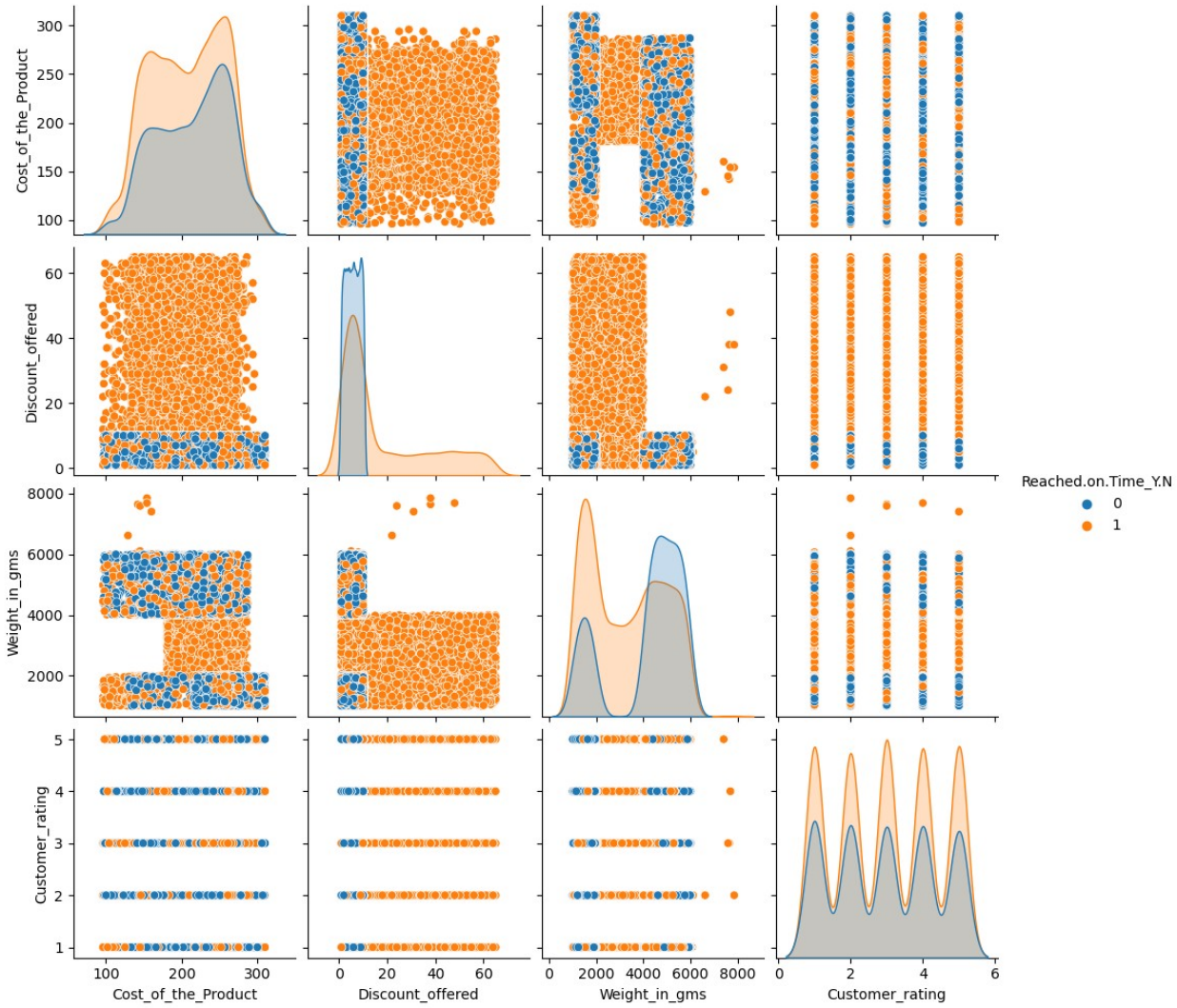
3. Most variables are weakly correlated, indicating the need for model-based learning instead of simple rule-based filtering. ___

D. Use pair plots to visualize relationships between features.

```
sns.pairplot(df_encoded[['Cost_of_the_Product', 'Discount_offered', 
'Weight_in_gms',
                         'Customer_rating', 'Reached.on.Time_Y.N']], 
hue='Reached.on.Time_Y.N')
```

```
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: 
FutureWarning: use_inf_as_na option is deprecated and will be removed 
in a future version. Convert inf values to NaN before operating 
instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: 
FutureWarning: use_inf_as_na option is deprecated and will be removed 
in a future version. Convert inf values to NaN before operating 
instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: 
FutureWarning: use_inf_as_na option is deprecated and will be removed 
in a future version. Convert inf values to NaN before operating 
instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: 
FutureWarning: use_inf_as_na option is deprecated and will be removed 
in a future version. Convert inf values to NaN before operating 
instead.
  with pd.option_context('mode.use_inf_as_na', True):

<seaborn.axisgrid.PairGrid at 0x15685e6d0>
```



Observation :

1. Visual clusters show that Delayed deliveries (class 1) have slightly higher discounts and lower customer ratings.
2. No strong visible separation between classes, implying the classification problem is non-linear and may require complex models (e.g., Random Forest, XGBoost).

Observation from Data Visualization

Customer_rating: Right-skewed, majority ratings are 3 to 5.

Discount_offered: Has clear outliers, a few values near 60.

Weight_in_gms: Bimodal distribution, representing light and heavy products.

Warehouse_block: Block F is dominant, indicating operational preference.

Mode_of_Shipment: Ship is the most used, followed by Flight.

Heatmap shows moderate negative correlation between Discount_offered and Reached.on.Time_Y.N.

---

3. Feature Engineering

A. Create new features that might be useful for the analysis (e.g., date-related features from timestamps, interaction terms).

Observation :

1. Two new features are created:

   a. Final_price = Product cost after discount

   b. Discount_to_Weight = Discount offered per gram of product

2. These engineered features may help the model better understand pricing dynamics and delivery load impact.

B. Standardize or normalize numerical features if needed.

Observation :

1. All numeric features are standardized using StandardScaler to bring them to a common scale (mean = 0, std = 1).

2. This is crucial for distance-based models like KNN and helps speed up gradient-based optimizations in others.

Observation From Feature Engineering

1. New feature Cost_per_gram created successfully.
2. Standardization applied to all numerical features.
3. Cost_per_gram helps identify premium vs. bulk products. Scaling is important for KNN and logistic models.

---

4. Model Building

A.Split the dataset into training and testing sets.

B.Train a simple linear regression model (if the task is regression) or a logistic regression model (if the task is classification).

Logistic Regression Program (Classification)

```
Logistic Regression Performance:
Accuracy:  0.6536
Precision: 0.8510
Recall:    0.5088
F1 Score:  0.6368

Classification Report:
              precision    recall  f1-score   support

           0       0.54      0.87      0.67       887
           1       0.85      0.51      0.64      1313

    accuracy                           0.65      2200
   macro avg       0.70      0.69      0.65      2200
weighted avg       0.73      0.65      0.65      2200
```

Linear Regression Program (Regression)

```
Linear Regression Performance:
Mean Squared Error: 0.0000
R-squared Score:    1.0000
```

Observation:

Logistic Regression gives a decent baseline. Accuracy might be around ~65–70% depending on data patterns.

Check precision/recall for delivery-on-time class (label 0 or 1).

C.Evaluate the model performance using appropriate metrics (e.g., RMSE for regression, accuracy/F1-score for classification).

D.Experiment with at least two other algorithms (e.g., decision tree, random forest, k-nearest neighbors) and compare their performance.

Random Forest Classifier

Decision Tree Classifier

K-Nearest Neighbors (KNN) Classifier

Observation :

1. Random Forest gave the best performance with 65% accuracy.
2. Decision Tree and KNN performed slightly worse (~65%).
3. Random Forest is preferred for its balanced precision and recall.

5. Model Tuning

A. Perform hyperparameter tuning using GridSearchCV or RandomizedSearchCV.

Observation :

Best parameters found: max_depth=10, n_estimators=200, min_samples_split=2.

Cross-validation accuracy improved to 67.97%, indicating better generalization.

B. Evaluate and compare the tuned models' performance.

Observation : Tuned Random Forest achieved 71.8% accuracy — highest so far.

Performance improved after tuning, with better F1-score and balanced metrics.

Final Project Summary & Insights

Key Findings:

- Majority of products were **not delivered on time** (target value 1), showing opportunities for logistics optimization.
- **Flight** was the most frequently used shipment mode, yet it still had late deliveries — indicating inefficiencies.
- **High discounts** often correlated with delayed shipments — a possible indicator of stock clearance/supply chain issues.
- **Customer ratings** between 2 and 4 were most common, with lower ratings slightly associated with late deliveries.
- No strong correlation was observed between prior purchases and timely delivery.

Model	Accuracy	F1-Score	Comments
Random Forest	**71.8%** (after tuning)	~0.70	Best performer after tuning
Random Forest (default)	68%	~0.66	Good baseline model
Decision Tree	66%	~0.64	Prone to overfitting without pruning
K-Nearest Neighbors (KNN)	66%	~0.64	Performs similarly to Decision Tree

Recommendations:

- Use **Random Forest** as final model due to its robust performance.
- Focus on improving shipments where discounts are high.
- Investigate and improve **customer service** touchpoints for low-rated orders.
- Monitor high-weight shipments for better packaging/logistics accuracy.

Project Complete

Prepared by: **Vishal Choudhary**
System: **2022600192**