In [2]:
```
import pandas as pd

# Load dataset
df = pd.read_csv("C:\\datasets\\ecommerce_returns_synthetic_data.csv")

# Inspect first few rows
print(df.head())

# Check data types and missing values
print(df.info())
print(df.isnull().sum())
```

```
     Order_ID    Product_ID       User_ID  Order_Date Return_Date  \
0  ORD00000000  PROD00000000  USER00000000  2023-08-05  2024-08-26
1  ORD00000001  PROD00000001  USER00000001  2023-10-09  2023-11-09
2  ORD00000002  PROD00000002  USER00000002  2023-05-06         NaN
3  ORD00000003  PROD00000003  USER00000003  2024-08-29         NaN
4  ORD00000004  PROD00000004  USER00000004  2023-01-16         NaN

  Product_Category  Product_Price  Order_Quantity Return_Reason Return_Status  \
0         Clothing         411.59               3  Changed mind      Returned
1            Books         288.88               3    Wrong item      Returned
2             Toys         390.03               5           NaN  Not Returned
3             Toys         401.09               3           NaN  Not Returned
4            Books         110.09               4           NaN  Not Returned

   Days_to_Return  User_Age User_Gender User_Location Payment_Method  \
0           387.0        58        Male        City54     Debit Card
1            31.0        68      Female        City85    Credit Card
2             NaN        22      Female        City30     Debit Card
3             NaN        40        Male        City95         PayPal
4             NaN        34      Female        City80      Gift Card

  Shipping_Method  Discount_Applied
0        Next-Day             45.27
1         Express             47.79
2        Next-Day             26.64
3        Next-Day             15.37
4        Standard             16.37
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 17 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Order_ID          10000 non-null  object
 1   Product_ID        10000 non-null  object
 2   User_ID           10000 non-null  object
 3   Order_Date        10000 non-null  object
 4   Return_Date       5052 non-null   object
 5   Product_Category  10000 non-null  object
 6   Product_Price     10000 non-null  float64
 7   Order_Quantity    10000 non-null  int64
 8   Return_Reason     5052 non-null   object
 9   Return_Status     10000 non-null  object
 10  Days_to_Return    5052 non-null   float64
 11  User_Age          10000 non-null  int64
 12  User_Gender       10000 non-null  object
 13  User_Location     10000 non-null  object
 14  Payment_Method    10000 non-null  object
 15  Shipping_Method   10000 non-null  object
 16  Discount_Applied  10000 non-null  float64
dtypes: float64(3), int64(2), object(12)
memory usage: 1.3+ MB
None
Order_ID               0
Product_ID             0
User_ID                0
Order_Date             0
Return_Date         4948
Product_Category       0
Product_Price          0
Order_Quantity         0
Return_Reason       4948
Return_Status          0
Days_to_Return      4948
User_Age               0
User_Gender            0
User_Location          0
Payment_Method         0
Shipping_Method        0
Discount_Applied       0
dtype: int64
```

In [4]:
```
# Convert date columns safely
df['Order_Date'] = pd.to_datetime(df['Order_Date'], dayfirst=True, errors='coerce')
df['Return_Date'] = pd.to_datetime(df['Return_Date'], dayfirst=True, errors='coerce')

# Check which rows failed to parse
invalid_order_dates = df[df['Order_Date'].isna()]
invalid_return_dates = df[df['Return_Date'].isna()]

print("Invalid Order_Date rows:")
print(invalid_order_dates)

print("Invalid Return_Date rows:")
print(invalid_return_dates)
```

```
Invalid Order_Date rows:
        Order_ID    Product_ID       User_ID Order_Date Return_Date  \
3     ORD00000003  PROD00000003  USER00000003        NaT         NaT
4     ORD00000004  PROD00000004  USER00000004        NaT         NaT
5     ORD00000005  PROD00000005  USER00000005        NaT  2024-09-22
6     ORD00000006  PROD00000006  USER00000006        NaT  2023-08-03
8     ORD00000008  PROD00000008  USER00000008        NaT  2024-09-25
...           ...           ...           ...        ...         ...
9992  ORD00009992  PROD00009992  USER00009992        NaT  2024-03-14
9993  ORD00009993  PROD00009993  USER00009993        NaT         NaT
9995  ORD00009995  PROD00009995  USER00009995        NaT         NaT
9996  ORD00009996  PROD00009996  USER00009996        NaT         NaT
9998  ORD00009998  PROD00009998  USER00009998        NaT         NaT

     Product_Category  Product_Price  Order_Quantity Return_Reason  \
3                Toys         401.09               3           NaN
4               Books         110.09               4           NaN
5         Electronics         252.12               1     Defective
6            Clothing         382.89               2    Wrong item
8                Home         302.40               5     Defective
...               ...            ...             ...           ...
9992             Home         482.84               2  Changed mind
9993             Home         210.20               5           NaN
9995             Home         142.50               4           NaN
9996      Electronics         484.63               3           NaN
9998             Toys         129.22               1           NaN

     Return_Status  Days_to_Return  User_Age User_Gender User_Location  \
3     Not Returned             NaN        40        Male        City95
4     Not Returned             NaN        34      Female        City80
5         Returned           221.0        46      Female        City47
6         Returned            66.0        25        Male        City50
8         Returned           -91.0        54        Male        City81
...            ...             ...       ...         ...           ...
9992      Returned          -227.0        56      Female        City83
9993  Not Returned             NaN        61      Female        City31
9995  Not Returned             NaN        37        Male        City40
9996  Not Returned             NaN        69        Male        City62
9998  Not Returned             NaN        34      Female        City34

     Payment_Method Shipping_Method  Discount_Applied
3            PayPal        Next-Day             15.37
4         Gift Card        Standard             16.37
5       Debit Card        Next-Day             47.61
6      Credit Card         Express             28.49
8       Debit Card        Standard             18.50
...            ...             ...               ...
9992         PayPal        Next-Day             18.53
9993         PayPal         Express             25.60
9995         PayPal        Standard             34.27
9996     Debit Card         Express             25.44
9998      Gift Card         Express             49.97

[6030 rows x 17 columns]
Invalid Return_Date rows:
        Order_ID    Product_ID       User_ID Order_Date Return_Date  \
2     ORD00000002  PROD00000002  USER00000002 2023-06-05         NaT
3     ORD00000003  PROD00000003  USER00000003        NaT         NaT
4     ORD00000004  PROD00000004  USER00000004        NaT         NaT
10    ORD00000010  PROD00000010  USER00000010        NaT         NaT
11    ORD00000011  PROD00000011  USER00000011 2024-08-02         NaT
...           ...           ...           ...        ...         ...
9993  ORD00009993  PROD00009993  USER00009993        NaT         NaT
9995  ORD00009995  PROD00009995  USER00009995        NaT         NaT
9996  ORD00009996  PROD00009996  USER00009996        NaT         NaT
9997  ORD00009997  PROD00009997  USER00009997 2024-10-05         NaT
9998  ORD00009998  PROD00009998  USER00009998        NaT         NaT

     Product_Category  Product_Price  Order_Quantity Return_Reason  \
2                Toys         390.03               5           NaN
3                Toys         401.09               3           NaN
4               Books         110.09               4           NaN
10               Toys         119.00               2           NaN
11               Home         480.48               4           NaN
...               ...            ...             ...           ...
9993             Home         210.20               5           NaN
9995             Home         142.50               4           NaN
9996      Electronics         484.63               3           NaN
9997             Toys         386.57               5           NaN
9998             Toys         129.22               1           NaN

     Return_Status  Days_to_Return  User_Age User_Gender User_Location  \
2     Not Returned             NaN        22      Female        City30
3     Not Returned             NaN        40        Male        City95
4     Not Returned             NaN        34      Female        City80
10    Not Returned             NaN        70      Female        City41
11    Not Returned             NaN        54        Male        City22
...            ...             ...       ...         ...           ...
9993  Not Returned             NaN        61      Female        City31
9995  Not Returned             NaN        37        Male        City40
9996  Not Returned             NaN        69        Male        City62
9997  Not Returned             NaN        46        Male        City74
9998  Not Returned             NaN        34      Female        City34

     Payment_Method Shipping_Method  Discount_Applied
2        Debit Card        Next-Day             26.64
3            PayPal        Next-Day             15.37
4         Gift Card        Standard             16.37
10     Credit Card        Next-Day              8.72
11        Gift Card        Next-Day              0.27
...             ...             ...               ...
9993         PayPal         Express             25.60
9995         PayPal        Standard             34.27
9996     Debit Card         Express             25.44
9997    Credit Card        Next-Day             12.67
9998      Gift Card         Express             49.97

[4948 rows x 17 columns]
C:\Users\harsh\AppData\Local\Temp\ipykernel_16044\4251629337.py:3: UserWarning: Parsing dates in %Y-%m-%d format when dayfirst=True was specified. Pass `dayfirst=False` or specify a format to silence this warning.
  df['Return_Date'] = pd.to_datetime(df['Return_Date'], dayfirst=True, errors='coerce')
```

In [5]:
```
# Fill missing Return_Status with 'Not Returned'
df['Return_Status'] = df['Return_Status'].fillna('Not Returned')
```

```
# Fill missing Return_Reason with 'No Return'
df['Return_Reason'] = df['Return_Reason'].fillna('No Return')

# Optional: Ensure numeric columns are numeric
df['Product_Price'] = pd.to_numeric(df['Product_Price'], errors='coerce')
df['Order_Quantity'] = pd.to_numeric(df['Order_Quantity'], errors='coerce')
df['Days_to_Return'] = pd.to_numeric(df['Days_to_Return'], errors='coerce')
df['Discount_Applied'] = pd.to_numeric(df['Discount_Applied'], errors='coerce')
```

In [6]:
```
# Return rate per category
category_return = df.groupby('Product_Category')['Return_Status'].apply(lambda x: (x=='Returned').mean()*100)
print(category_return)

# Return rate by location
location_return = df.groupby('User_Location')['Return_Status'].apply(lambda x: (x=='Returned').mean()*100)
print(location_return)

# Return rate by payment method
payment_return = df.groupby('Payment_Method')['Return_Status'].apply(lambda x: (x=='Returned').mean()*100)
print(payment_return)

# Return rate by shipping method
shipping_return = df.groupby('Shipping_Method')['Return_Status'].apply(lambda x: (x=='Returned').mean()*100)
print(shipping_return)
```

```
Product_Category
Books          50.661440
Clothing       52.450000
Electronics    50.931990
Home           49.014778
Toys           49.537037
Name: Return_Status, dtype: float64
User_Location
City1      42.982456
City10     59.803922
City100    48.113208
City11     49.494949
City12     54.081633
             ...
City95     55.421687
City96     48.387097
City97     50.000000
City98     47.663551
City99     49.494949
Name: Return_Status, Length: 100, dtype: float64
Payment_Method
Credit Card    50.505459
Debit Card     51.115538
Gift Card      51.596374
PayPal         48.830645
Name: Return_Status, dtype: float64
Shipping_Method
Express     49.939504
Next-Day    51.090159
Standard    50.515152
Name: Return_Status, dtype: float64
```

In [7]:
```
return_reasons = df[df['Return_Status']=='Returned'].groupby('Return_Reason').size().sort_values(ascending=False)
print(return_reasons)
```

```
Return_Reason
Defective           1327
Wrong item          1258
Changed mind        1255
Not as described    1212
dtype: int64
```

In [8]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix

# Target variable
df['Return_Flag'] = df['Return_Status'].apply(lambda x: 1 if x=='Returned' else 0)

# Features
X = df[['Product_Category','Product_Price','Order_Quantity','User_Age','User_Gender','User_Location','Payment_Method','Shipping_Method','Discount_Applied']]
y = df['Return_Flag']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Preprocessing for categorical and numeric features
numeric_features = ['Product_Price','Order_Quantity','User_Age','Discount_Applied']
categorical_features = ['Product_Category','User_Gender','User_Location','Payment_Method','Shipping_Method']

numeric_transformer = StandardScaler()
categorical_transformer = OneHotEncoder(handle_unknown='ignore', sparse_output=False)

preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
        ('cat', categorical_transformer, categorical_features)
    ])

# Logistic Regression pipeline
clf = Pipeline(steps=[('preprocessor', preprocessor),
                      ('classifier', LogisticRegression(max_iter=1000))])

# Train model
clf.fit(X_train, y_train)

# Predictions
y_pred = clf.predict(X_test)

# Evaluation
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.50      0.47      0.48      1009
           1       0.50      0.53      0.51       991

    accuracy                           0.50      2000
   macro avg       0.50      0.50      0.50      2000
weighted avg       0.50      0.50      0.50      2000

[[470 539]
 [461 530]]
```

In [9]:
```
# Predict probability of return
df['Return_Prob'] = clf.predict_proba(X)[:,1]

# Filter high-risk products
high_risk_products = df[df['Return_Prob'] > 0.5]  # Threshold 0.5
high_risk_products[['Product_ID','Product_Category','Return_Prob']].to_csv('high_risk_products.csv', index=False)
```

In [ ]:
```

```