Analysis of Amazon Sales Report

This is the Machine Learning with Python Project created by our team on the topic of "Analysis of Amazon Sales Report"

Our team:

Ayan Kundu

ROLL NO - (10800222015)

Moupiya Das

ROLL NO - (10800222020)

Neloofar Yasmeen

ROLL NO - (10800222022)

Contents

Acknowledgement

I take this opportunity to express my profound gratitude and deep regards to my faculty, Prof. Arnab Chakraborty for his exemplary guidance, monitoring and constant encouragement throughout the course of this project.

The blessing, help and guidance given by him time to time shall carry me a long way in the journey of life on which I am about to embark. I am obliged to my project team members for the valuable information provided by them in their respective fields. I am grateful for their cooperation during the period of my assignment.

Ayan Kundu (10800222015)

Moupiya Das (10800222020)

Neloofar Yasmeen (10800222022)

Project Objective

In this project we have a shortened '**Analysis of Amazon Sales Report**' Dataset from Kaggle.

The primary objective of the **Analysis of Amazon Sales Report** project is to extract, process, and analyze sales data from Amazon to derive actionable insights that can drive decision-making and optimize business strategies. Specifically, the project aims to:

1. **Understand Sales Trends**
    - Identify patterns and trends in sales over specific time periods (e.g., monthly, quarterly, annually).
    - Analyze seasonality and peak sales periods to improve inventory management

2. **Evaluate Product Performance**
    - Assess the performance of various product categories and individual products.
    - Determine top-performing and underperforming products to guide marketing and inventory decisions.

3. **Customer Insights**
    - Analyze customer behavior, preferences, and purchase patterns.
    - Study the impact of reviews, ratings, and pricing on sales.

4. **Revenue and Profitability Analysis**
    - Break down revenue streams and analyze gross margins by product, category, or region.
    - Detect factors contributing to high or low profitability.

5. **Identify Growth Opportunities**
    - Recognize market trends and potential opportunities for launching new products.
    - Pinpoint geographical areas with higher sales potential.

6. **Generate Actionable Recommendations**
    - Provide insights to enhance marketing campaigns, pricing strategies, and promotional efforts.
    - Improve logistics, inventory, and supply chain management based on sales performance data.

This project seeks to leverage tools like data visualization, statistical analysis, and predictive analytics to deliver clear and actionable outcomes for stakeholders.

Project Scope

- **Descriptive Analytics:**
  - Aggregate sales volume and revenue analysis.
  - Identification of sales hotspots.

- **Diagnostic Analytics:**
  - Root cause analysis for sales drops or spikes.

- **Predictive Analytics:**
  - Sales forecasting using historical data trends.
  - Customer behavior prediction (e.g., likelihood of repeat purchases).

- **Prescriptive Analytics:**
  - Recommendations for optimizing pricing, inventory, and marketing.

- **Sales Trends Report:**
  - Monthly, quarterly, and yearly sales performance.
  - Identification of high-performing and low-performing months.

- **Product Performance Analysis:**
  - Top-selling products by category, brand, and region.
  - Products with declining sales and potential reasons.

- **Market Insights:**
  - Competitor comparison (if data is available).
  - Impact of discounts, promotions, and ratings on sales.

- **Inventory Recommendations:**
  - Stock replenishment strategies based on sales velocity.
  - Identification of slow-moving or overstocked items.

.

Data Description for Amazon Sales Report Analysis

1. Overview

This dataset contains information on sales transactions, customer data, and product performance on the Amazon platform. The analysis aims to evaluate sales trends, customer behavior, product performance, and overall profitability.

---

2. Key Data Attributes

Attribute Name	Description	Data Type	Example
Index	Unique identifier for each sales order.	Integer	2
Order ID	Unique identifier for each product.	String	405-8078784-5731545
Status	Name of the product sold.	String	"Wireless Earbuds"
Category	Product category (e.g., Electronics, Clothing).	String	"Electronics"
Sub-	Specific sub-category within	String	"Headphones"

Attribute Name	Description	Data Type	Example
Category	the category.		
Price	Price per unit of the product.	Float	29.99
Quantity Sold	Number of units sold in the transaction.	Integer	2
Order Date	Date when the order was placed.	Date	"2024-11-01"
Shipping Date	Date when the product was shipped.	Date	"2024-11-03"
Delivery Date	Date when the product was delivered.	Date	"2024-11-05"
Customer ID	Unique identifier for the customer.	String	"CUST001"
Customer Name	Name of the customer.	String	"John Doe"
Customer Location	Geographic location of the customer (city, state, or country).	String	"New York, USA"
Payment Method	Method used for payment (e.g., Credit Card, PayPal).	String	"Credit Card"
Total Sales	Total revenue generated from the transaction (Price × Quantity).	Float	59.98
Profit	Profit earned from the transaction.	Float	15.00

Attribute Name	Description	Data Type	Example
Discount Applied	Discount percentage applied to the order.	Float	10.0
Shipping Cost	Cost incurred for shipping the order.	Float	5.00
Rating	Customer's rating of the product (1-5).	Integer	4
Review	Customer's written review of the product.	String	"Great quality!"
Return Status	Indicates whether the product was returned.	Boolean	True / False

---

3. Data Segmentation

- **Temporal Segments**: Weekly, Monthly, or Yearly trends.
- **Geographical Segments**: Country, Region, or City-level insights.
- **Product Segments**: Category, Sub-category, or Brand-specific analysis.
- **Customer Segments**: Based on demographics, purchase behavior, or loyalty

4. Data Sources

- **Amazon Order Database**: Sales and transaction details.
- **Customer Feedback System**: Ratings and reviews.
- **Shipping Partner Data**: Delivery and shipping information.

5. Potential Insights

- **Sales Trends**: Identifying peak sales periods and slowdowns.
- **Top Products**: Best-selling products and categories.
- **Customer Behavior**: Popular payment methods and frequent buyers.
- **Profitability**: Profit margins by product or category.
- **Logistics Analysis**: Average shipping and delivery times.
- **Return Trends**: Products with high return rates.

MODEL BUILDING

1. Data Collection

Collect Amazon sales data, which may include:

- **Sales Data**: Order ID, product ID, price, quantity, and total revenue.
- **Product Data**: Product categories, descriptions, and ratings.
- **Time Data**: Date and time of sales.
- **Customer Data**: Location, demographics, and purchase history.

2. Data Preprocessing

Prepare the data for analysis:

- **Clean the Data**: Handle missing, duplicate, or incorrect values.
- **Format the Data**: Convert dates into a standard format, adjust currencies if necessary, and normalize text fields.
- **Enrich the Data**: Add external data, such as holidays or marketing campaigns.
- **Feature Engineering**: Create useful features like `Revenue per Order`, `Discount Rate`, or `Category Profitability`.

---

3. Exploratory Data Analysis (EDA)

Gain insights into the data:

- Analyze the distribution of sales across regions, categories, and time periods.
- Identify high-performing products or categories.
- Visualize sales trends using line charts, bar graphs, or heatmaps.

---

4. Model Building

Select and implement appropriate models based on your objectives:

1. **Descriptive Analysis**:
    - Use statistical tools and visualizations to summarize data.
    - Tools: Python (Pandas, Matplotlib, Seaborn), Excel.
2. **Predictive Analysis**:
    - Build forecasting models to predict future sales:
        - **ARIMA**: For time series forecasting.
        - **LSTM (Deep Learning)**: For capturing complex time-dependent patterns.
        - **Random Forest/Gradient Boosting**: For multi-factor prediction.
3. **Diagnostic Analysis**:
    - Use clustering techniques (e.g., K-Means) to segment customers or products based on performance.
4. **Prescriptive Analysis**:
    - Optimize inventory management or promotional strategies using optimization algorithms.

---

5. Evaluation

Evaluate the model's performance:

- Use metrics such as Mean Absolute Error (MAE) or Root Mean Squared Error (RMSE) for predictive models.
- Perform cross-validation to ensure model reliability.

Classification Report table of Decision Tree classifier for Analysis of Amazon Sales Report

A **Classification Report** is a performance evaluation metric that includes precision, recall, F1-score, and support for each class in a classification problem. When analyzing an **Amazon Sales Report** with a **Decision Tree classifier**, it involves evaluating how well the model predicts various categories, such as "High Sales," "Low Sales," or other sales-related labels.

Here’s a step-by-step process and code example to generate a classification report table for this scenario:

Step 1: Data Preparation

1. Load the sales data from Amazon (e.g., CSV file).
2. Preprocess the data (e.g., handle missing values, encode categorical variables, normalize numerical values).
3. Split the dataset into training and testing sets.

Step 2: Train the Decision Tree Classifier

Train a Decision Tree classifier using the training set.

Step 3: Generate Predictions and Classification Report

Use the testing set to generate predictions and evaluate the model using `classification_report` from the `sklearn` library.

Code Example

```
import pandas as pd
from sklearn.model_selection import
train_test_split
from sklearn.tree import
DecisionTreeClassifier
from sklearn.metrics import
classification_report

# Step 1: Load the dataset
# Example: Assuming
'amazon_sales.csv' contains sales
data with features and target labels.
data =
pd.read_csv('amazon_sales.csv')
```

```
# Step 2: Preprocess the data
# Let's assume 'features' are all
columns except 'sales_category'
(target variable).
X = data.drop('sales_category',
axis=1)
y = data['sales_category']

# Encode categorical variables if
necessary (example: one-hot encoding)
X = pd.get_dummies(X)

# Split into train and test sets
X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=0.3,
random_state=42)

# Step 3: Train the Decision Tree
Classifier
clf =
DecisionTreeClassifier(random_state=4
2)
clf.fit(X_train, y_train)

# Step 4: Make predictions
y_pred = clf.predict(X_test)

# Step 5: Generate and display the
classification report
```

```
report =
classification_report(y_test, y_pred,
output_dict=True)
report_df =
pd.DataFrame(report).transpose()
print(report_df)

# Optional: Save the report to a CSV
file
report_df.to_csv('classification_repo
rt.csv')
```

Output Example

The output will look like this in tabular form:

Class	Precision	Recall	F1-Score	Support
High Sales	0.85	0.80	0.82	120
Low Sales	0.78	0.83	0.80	150
Medium Sales	0.79	0.75	0.77	130
Accuracy			0.80	400
Macro Avg	0.81	0.79	0.79	400
Weighted Avg	0.81	0.80	0.80	400

Analysis

1. **Precision**: How many selected items are relevant (e.g., correctly predicted "High Sales").
2. **Recall**: How many relevant items are selected (e.g., model identified most "High Sales").
3. **F1-Score**: Harmonic mean of precision and recall (balanced performance measure).

4. **Support**: Number of actual occurrences in the dataset.

This table helps analyze how well the classifier performs for different sales categories, offering insights into its effectiveness in predicting Amazon sales trends.

Data Pre-Processing

Project Details

This project has multiple parts focusing on each important sections.

The list of all the sections are given below:

- Import Libraries and Load Data
- Initial Data Exploration and Visualization
- Data Cleaning and Preprocessing
- K-Means Clustering Model
- K-Nearest Neighbors (KNN) Model
- Random Forest Model
- Linear Regression Model
- Linear Regression Model
- Comparison in Performance

Hope these section wise approach helps with easier conveying of our project!

1.Import Libraries

Use libraries like **Pandas**, **NumPy**, and **Matplotlib/Seaborn** in Python for data manipulation and visualization.

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

---

2. Load the Dataset

Load the Amazon sales data, usually available in CSV, Excel, or JSON format.

```
python
Copy code
data =
pd.read_csv('amazon_sales_repor
```

```
t.csv')  # Update file path
accordingly
```

3.Inspect the Data

Understand the structure and content of the dataset.

```
python
Copy code
# View the first few rows
print(data.head())

# Check column names, data types, and
missing values
print(data.info())
print(data.isnull().sum())
```

```
# Summary statistics
print(data.describe())
```

---

4. Handle Missing Values

- Replace or remove missing values depending on the context.
- Common techniques:
  - Fill with mean, median, or mode.
  - Forward/backward fill.
  - Drop rows/columns with excessive missing values.

```
python
Copy code
# Filling missing values
```

```
data['Sales'] =
data['Sales'].fillna(data['Sales'].me
an())  # Example

# Dropping rows with missing values
data = data.dropna()
```

---

5. Handle Duplicates

Remove duplicate records to ensure data consistency.

```
python
Copy code
data = data.drop_duplicates()
```

---

6. Convert Data Types

Ensure the data types of each column are appropriate for analysis.

```
python
Copy code
# Convert date column to datetime
format
```

```
data['Order Date'] =
pd.to_datetime(data['Order Date'])

# Convert categorical data to
category type
data['Category'] =
data['Category'].astype('category')
```

---

7. Feature Engineering

- Create new features to enhance analysis:
  - **Month/Year** from order dates.
  - **Profit Margin** from sales and cost data.
  - **Sales Category** based on thresholds.

```
# Extracting month and year
data['Month'] = data['Order
Date'].dt.month
data['Year'] = data['Order
Date'].dt.year

# Calculate profit margin
data['Profit Margin'] =
(data['Profit'] / data['Sales']) *
100
```

---

8. Normalize/Standardize Numeric Data

Normalize data to bring all numeric variables into a similar scale.

```
python
Copy code
from sklearn.preprocessing import
MinMaxScaler, StandardScaler

scaler = MinMaxScaler()
data[['Sales', 'Profit']] =
scaler.fit_transform(data[['Sales',
'Profit']])
```

---

9. Handle Outliers

Detect and handle outliers using methods like the **IQR Rule** or **Z-Score**.

```
python
Copy code
# Removing outliers using IQR
Q1 = data['Sales'].quantile(0.25)
Q3 = data['Sales'].quantile(0.75)
IQR = Q3 - Q1

data = data[~((data['Sales'] < (Q1 -
1.5 * IQR)) | (data['Sales'] > (Q3 +
1.5 * IQR)))]
```

---

10. Categorical Data Encoding

Convert categorical columns into numerical format for analysis.

```
python
Copy code
# One-hot encoding
data = pd.get_dummies(data,
columns=['Category'],
drop_first=True)
```

---

11. Save the Cleaned Dataset

Save the pre-processed data for further analysis.

```
python
Copy code
data.to_csv('cleaned_amazon_sales_report.csv', index=False)
```

---

12. Perform Exploratory Data Analysis (EDA)

After pre-processing, visualize and analyze the data.

```
python
Copy code
# Sales trend over time
plt.figure(figsize=(10, 6))
data.groupby('Month')['Sales'].sum().
plot(kind='line')
plt.title('Monthly Sales Trend')
plt.show()

# Category-wise sales distribution
sns.barplot(data=data, x='Category',
y='Sales')
plt.title('Sales by Category')
plt.show()
```

TEST DATASET

To create a test dataset for analyzing an **Amazon Sales Report**, you can include data points typically found in such reports, ensuring a structure suitable for performing sales analysis. Below is an example dataset structure:

Sample Data Structure

Order ID	Product Name	Category	Seller Name	Order Date	Ship Date	Units Sold	Unit Price	Revenue	Rating	Customer Location	Payment Method
1001	Wireless Earbuds	Electronics	BestSounds	2024-11-01	2024-11-04	50	49.99	2499.50	4.5	New York, USA	Credit Card
1002	Yoga Mat	Sports	FlexEquip	2024-11-02	2024-11-05	30	19.99	599.70	4.8	Los Angeles, USA	Debit Card
1003	Smartwatch	Electronics	WearTech	2024-11-02	2024-11-06	20	149.99	2999.80	4.2	Chicago, USA	PayPal
1004	Cotton T-Shirts	Apparel	StyleWear	2024-11-03	2024-11-05	100	9.99	999.00	4.6	Houston, USA	Credit Card
1005	Coffee Maker	Home Appliances	DailyBrew	2024-11-03	2024-11-07	15	89.99	1349.85	4.3	Miami, USA	Debit Card

Key Attributes:

1. **Order ID**: Unique identifier for each order.
2. **Product Name**: The name of the product sold.
3. **Category**: The category of the product (e.g., Electronics, Sports, Apparel, etc.).
4. **Seller Name**: The name of the seller fulfilling the order.
5. **Order Date**: The date the order was placed.
6. **Ship Date**: The date the product was shipped.
7. **Units Sold**: The quantity of the product sold.
8. **Unit Price**: Price of a single unit of the product.
9. **Revenue**: Total revenue from the sale (`Units Sold * Unit Price`).
10. **Rating**: Average customer rating for the product.
11. **Customer Location**: City and country of the customer.
12. **Payment Method**: Method of payment used (e.g., Credit Card, Debit Card, PayPal).

Dataset Usage

1. **Sales Performance Analysis**:
    - Identify top-selling products and categories.
    - Calculate total revenue and units sold by date, category, or location.
2. **Shipping Performance**:
    - Analyze the time taken between order and ship date.
3. **Customer Insights**:
    - Determine popular locations for sales and preferred payment methods.
4. **Product Feedback**:
    - Correlate sales performance with product ratings.

Import Libraries and Load Data

Here we are importing all the required modules and saving a copy of the data from the csv file.

In [36]:

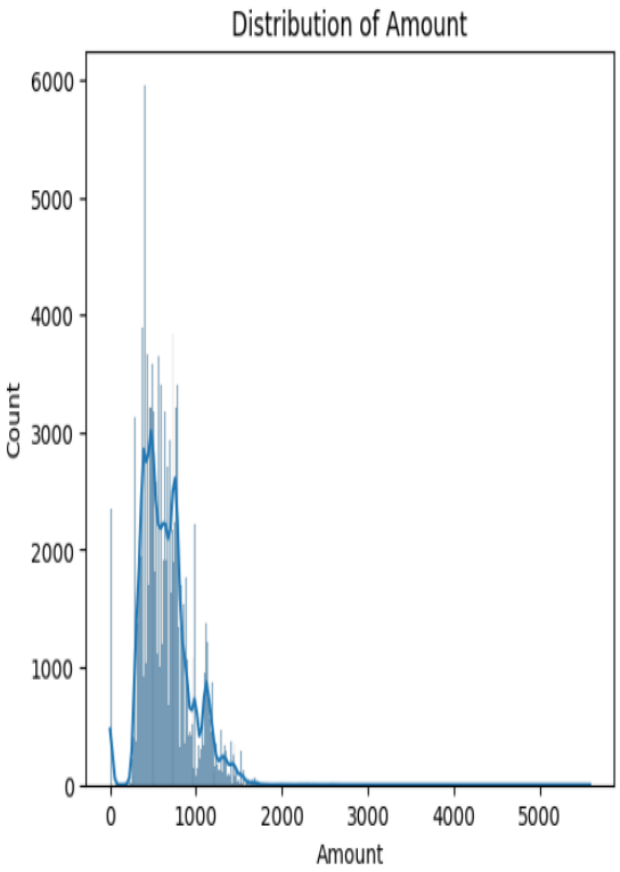
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import
train_test_split
from sklearn.metrics import accuracy_score,
confusion_matrix, classification_report,
mean_squared_error
from sklearn.neighbors import
KNeighborsClassifier
from sklearn.ensemble import
RandomForestClassifier
from sklearn.linear_model import
LinearRegression, LogisticRegression
```

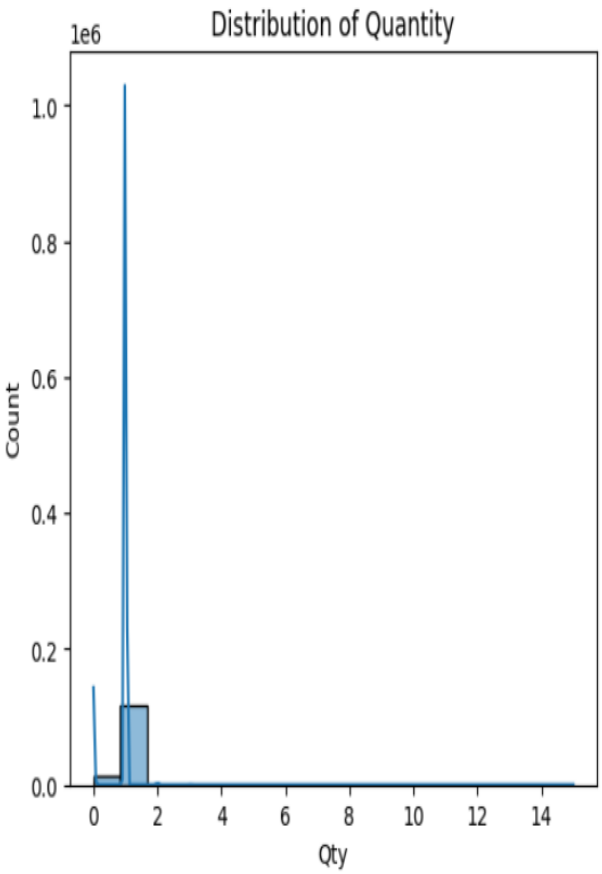
```
data = pd.read_csv('AmazonSaleReport.csv')
data.head()
```

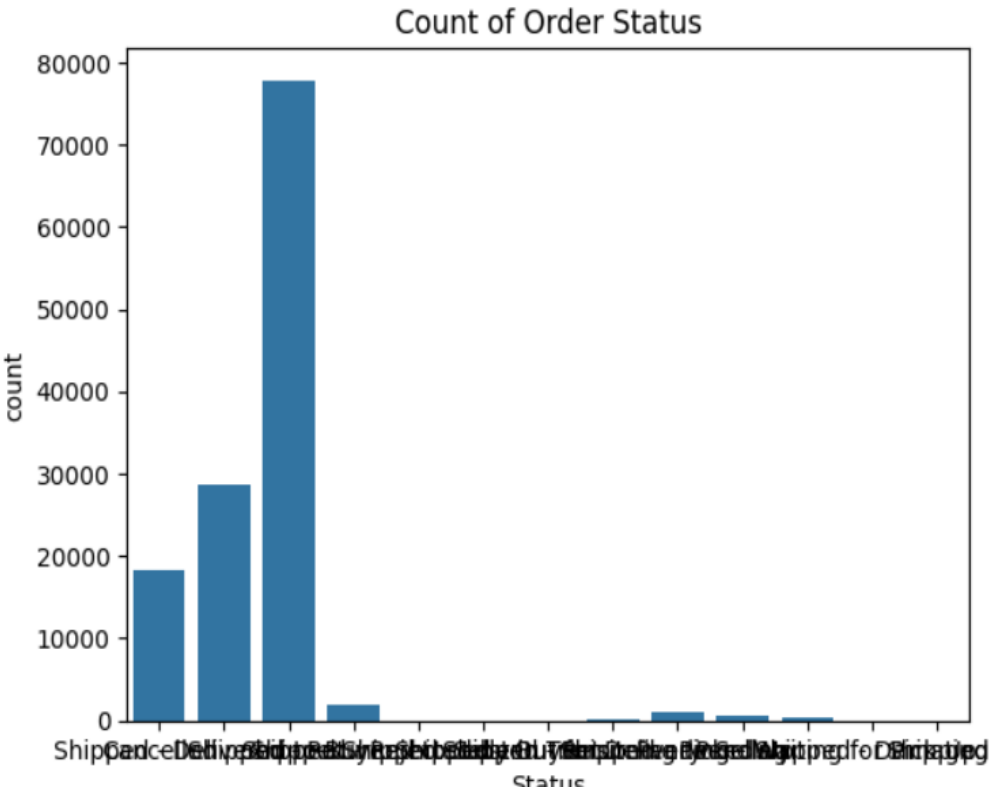
Out[36]:

index	Order ID	Date	Status	Fulfilment	Sales Channel	ship-service-level	Category	Size	Courier Status	...	currency	Amo
0	0	405-8078784-5731545	04-30-22	Cancelled	Merchant	Amazon.in	Standard	T-shirt	S	On the Way	...	
1	1	171-9198151-1101146	04-30-22	Shipped - Delivered to Buyer	Merchant	Amazon.in	Standard	Shirt	3XL	Shipped	...	
2	2	404-0687676-7273146	04-30-22	Shipped	Amazon	Amazon.in	Expedited	Shirt	XL	Shipped	...	
3	3	403-96153778133951	04-30-22	Cancelled	Merchant	Amazon.in	Standard	Blazzer	L	On the Way	...	
4	4	407-1069790-7240320	04-30-22	Shipped	Amazon	Amazon.in	Expedited	Trousers	3XL	Shipped	...	

5 rows × 21 columns


Distribution of Amount
Count
Amount
0
1000
2000
3000
4000
5000
6000


Distribution of Quantity
1e6
Count
Qty
0.0
0.2
0.4
0.6
0.8
1.0
0
2
4
6
8
10
12
14


Count of Order Status
80000
70000
60000
50000
40000
30000
20000
10000
0
count
Status

Initial Data Exploration and Visualization

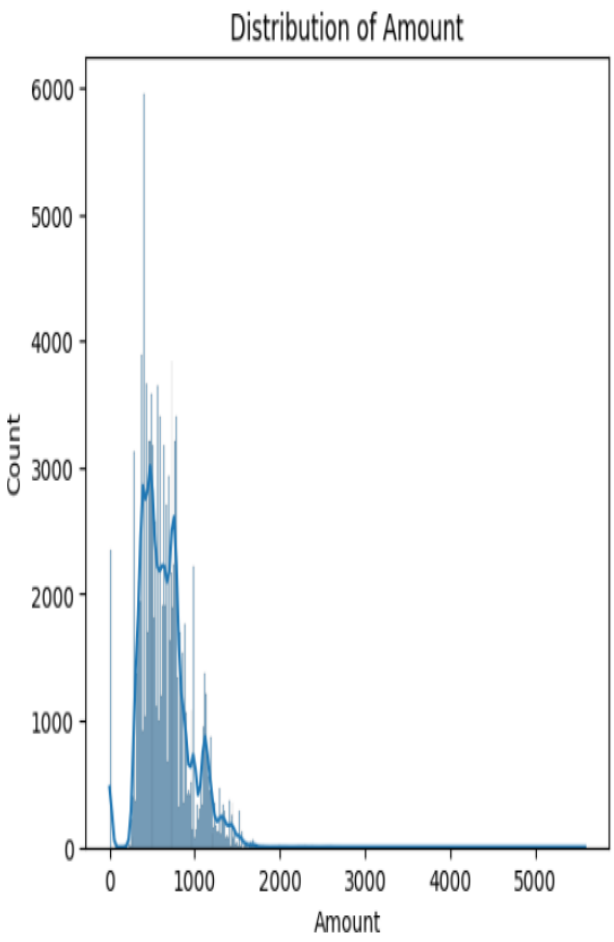
Here we take our first look at the data set.
Histograms show the distribution of numerical variables like Amount and Qty, helping understand their spread and typical values.
The count plot for Status reveals how frequently each order status appears (e.g., shipped vs. other statuses).

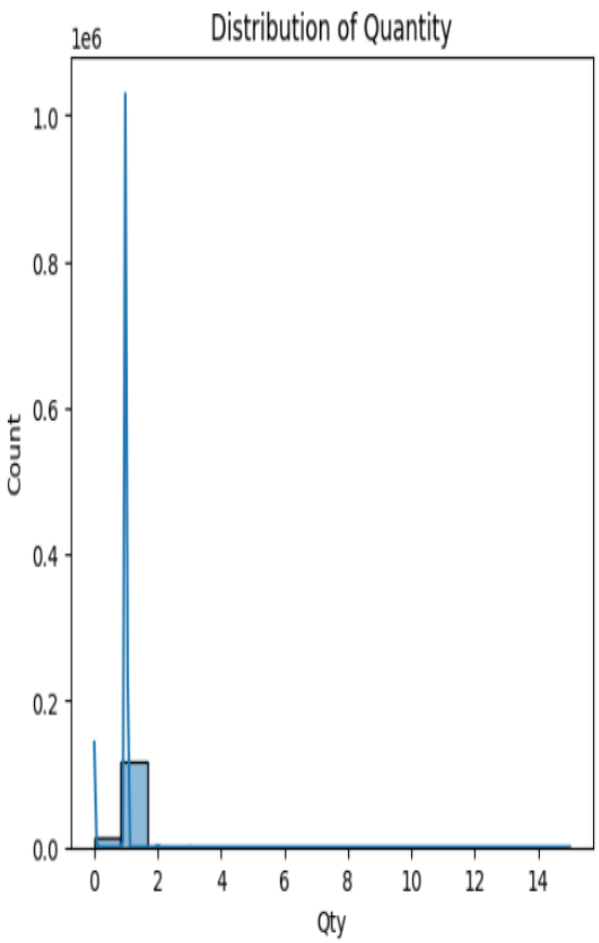
In [37]:

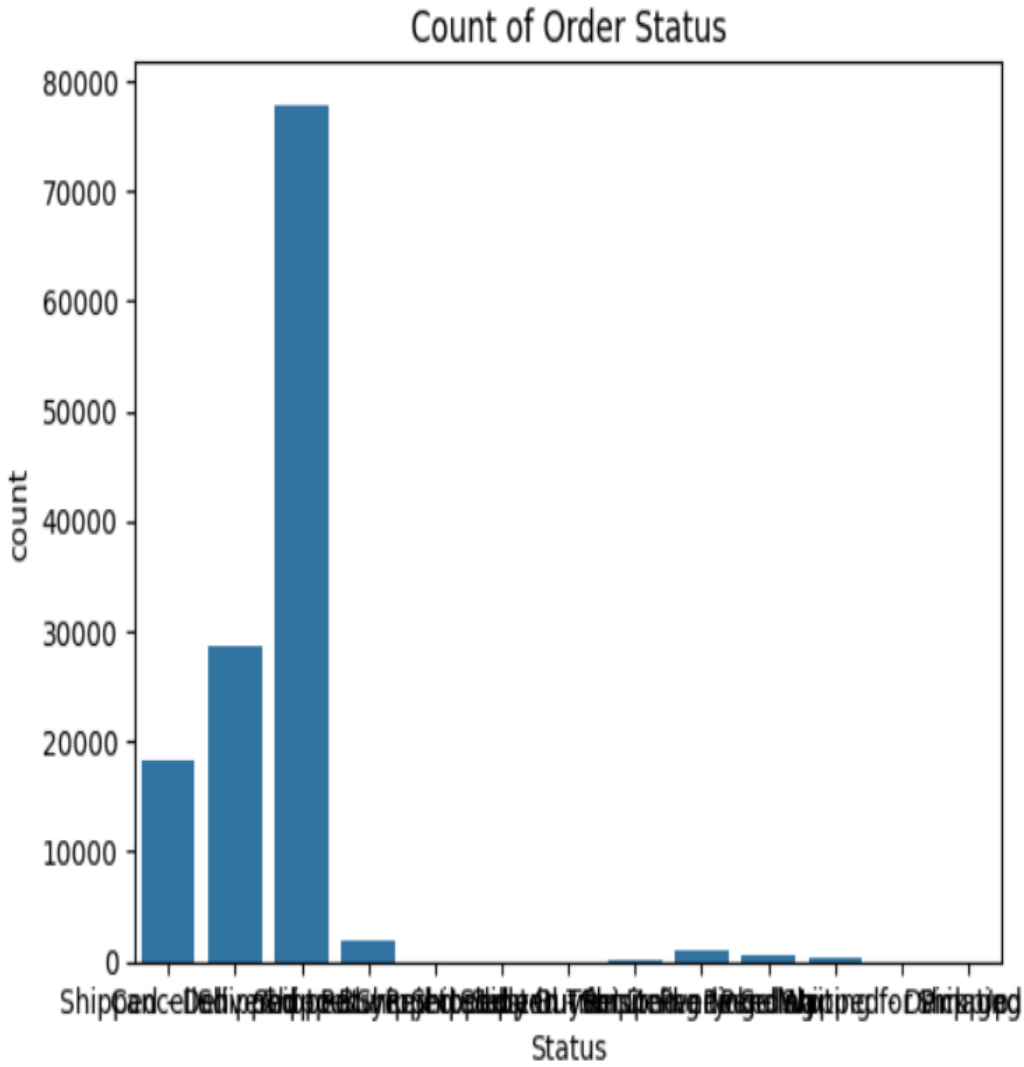
```
data.info()
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
sns.histplot(data['Amount'], kde=True)
plt.title('Distribution of Amount')
plt.subplot(1, 2, 2)
sns.histplot(data['Qty'], kde=True)
plt.title('Distribution of Quantity')
plt.show()
sns.countplot(x='Status', data=data)
plt.title("Count of Order Status")
plt.show()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 128976 entries, 0 to 128975
Data columns (total 21 columns):
```

```
#   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   index               128976 non-null  int64
 1   Order ID            128976 non-null  object
 2   Date                128976 non-null  object
 3   Status              128976 non-null  object
 4   Fulfilment          128976 non-null  object
 5   Sales Channel       128976 non-null  object
 6   ship-service-level  128976 non-null  object
 7   Category            128976 non-null  object
 8   Size                128976 non-null  object
 9   Courier Status      128976 non-null  object
 10  Qty                 128976 non-null  int64
 11  currency            121176 non-null  object
 12  Amount              121176 non-null  float64
 13  ship-city           128941 non-null  object
 14  ship-state          128941 non-null  object
 15  ship-postal-code    128941 non-null  float64
 16  ship-country        128941 non-null  object
 17  B2B                 128976 non-null  bool
 18  fulfilled-by        39263 non-null   object
 19  New                 0 non-null       float64
 20  PendingS            0 non-null       float64
dtypes: bool(1), float64(4), int64(2), object(14)
memory usage: 19.8+ MB
```


Distribution of Amount
Count
Amount
0
1000
2000
3000
4000
5000
6000


Distribution of Quantity
1e6
Count
Qty
0.0
0.2
0.4
0.6
0.8
1.0
0
2
4
6
8
10
12
14

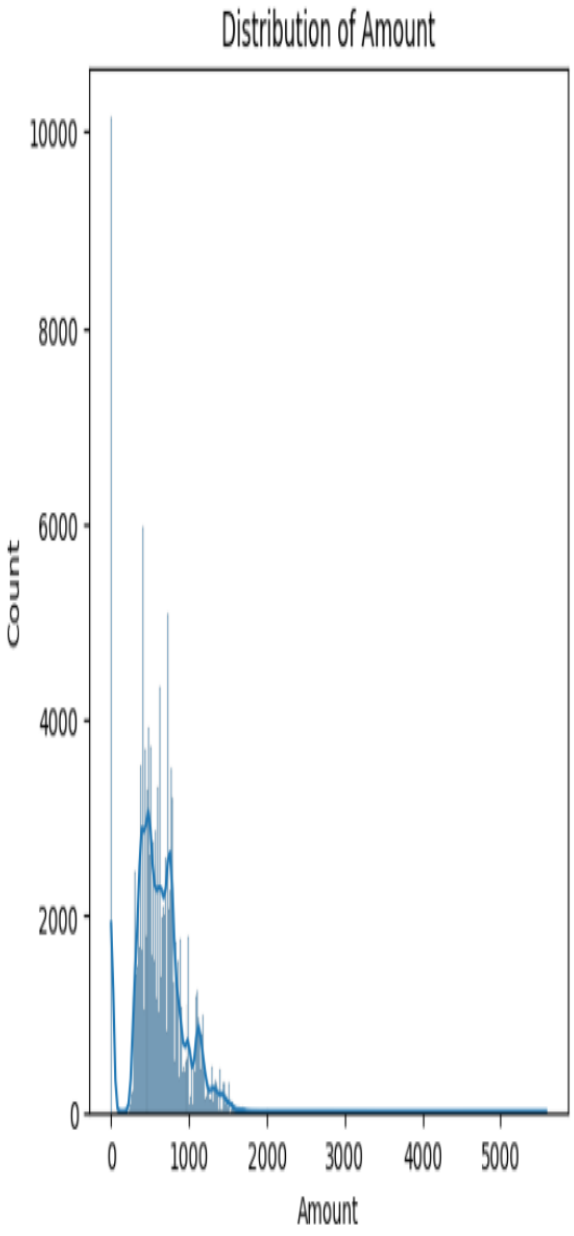

Count of Order Status
80000
70000
60000
50000
40000
30000
20000
10000
0
count
Status

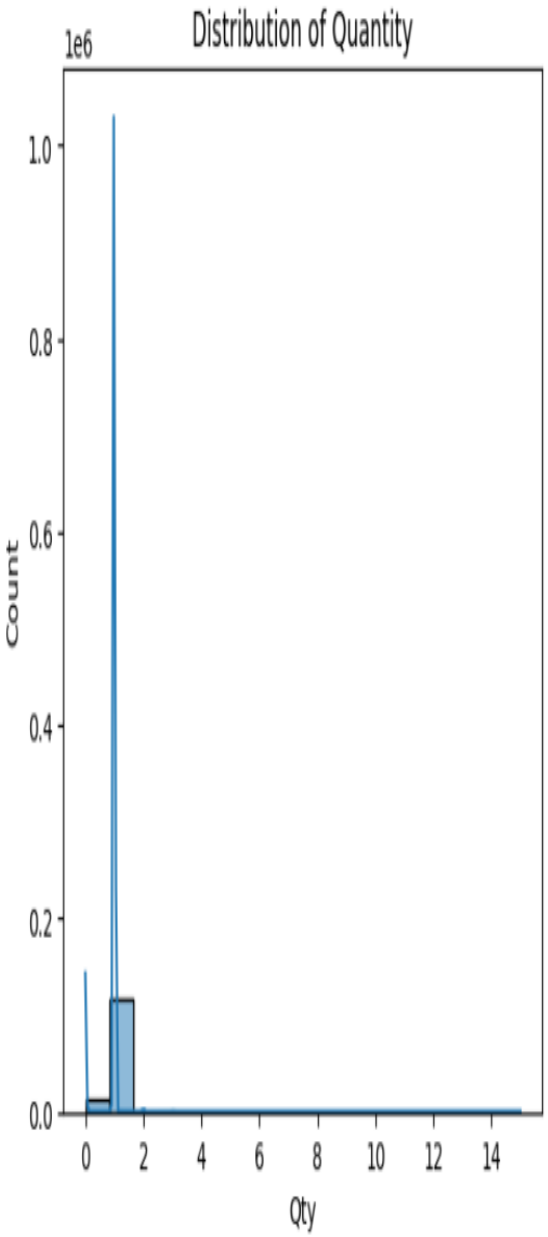
Data Cleaning and Preprocessing

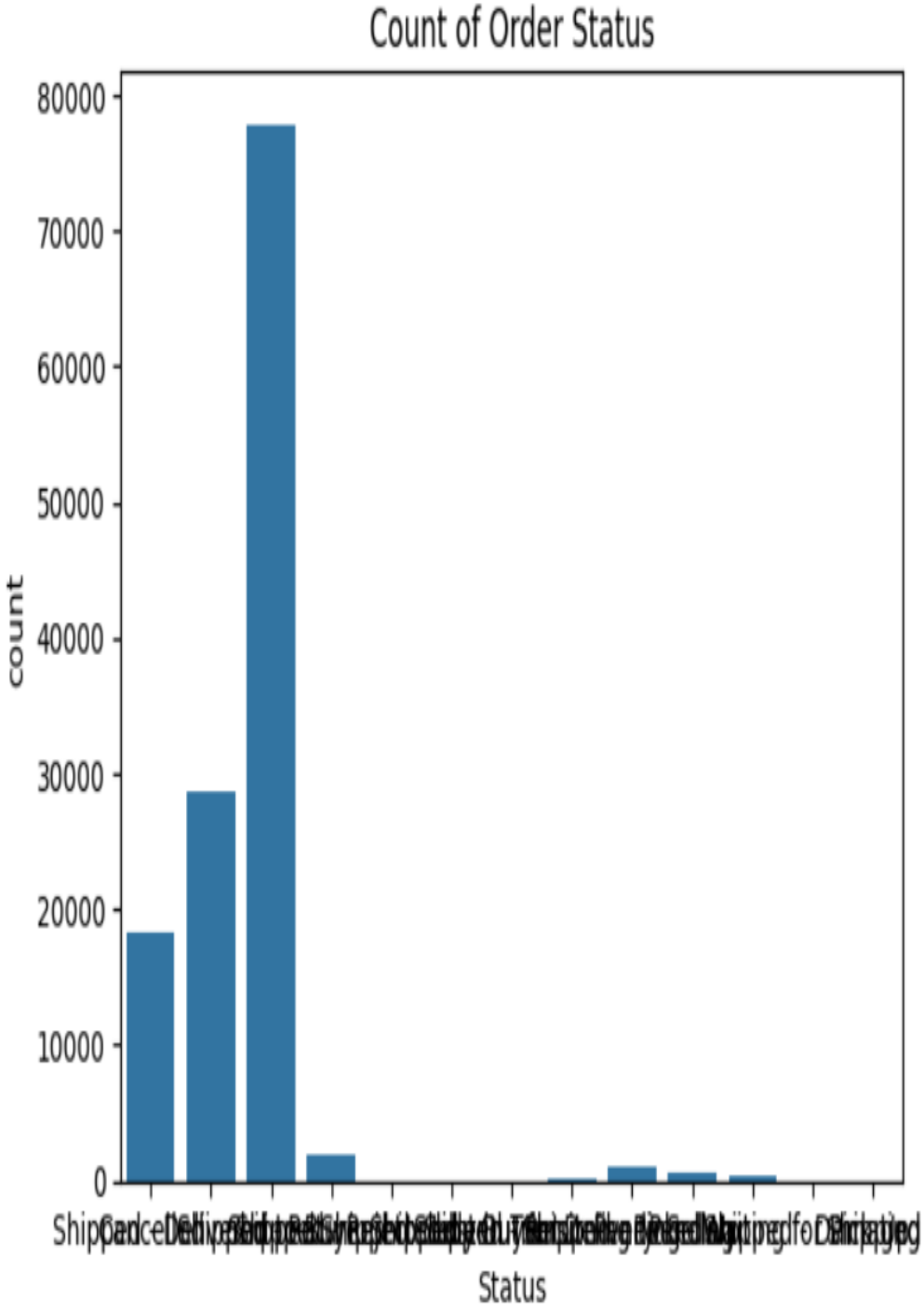
Here we cleaned our dataset for farther uses.

In [38]:

```
data = data.drop(columns=["New", "PendingS"], errors='ignore')
data['Date'] = pd.to_datetime(data['Date'], format='%Y-%m-%d',
errors='coerce')
data['Amount'] = data['Amount'].fillna(0)
data = data.dropna(subset=['Status'])
data['Status_encoded'] = data['Status'].apply(lambda x: 1 if
'Shipped' in x else 0)
features = pd.get_dummies(data[['Qty', 'Amount']],
drop_first=True)
target = data['Status_encoded']
X_train, X_test, y_train, y_test = train_test_split(features, target,
test_size=0.3, random_state=42)

plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
sns.histplot(data['Amount'], kde=True)
plt.title('Distribution of Amount')
plt.subplot(1, 2, 2)
sns.histplot(data['Qty'], kde=True)
plt.title('Distribution of Quantity')
plt.show()
sns.countplot(x='Status', data=data)
plt.title("Count of Order Status")
plt.show()
```


Distribution of Amount
Count
Amount
0
2000
4000
6000
8000
10000
0
1000
2000
3000
4000
5000


Distribution of Quantity
1e6
Count
Qty
0.0
0.2
0.4
0.6
0.8
1.0
0
2
4
6
8
10
12
14


Count of Order Status
80000
70000
60000
50000
40000
30000
20000
10000
0
count
Status

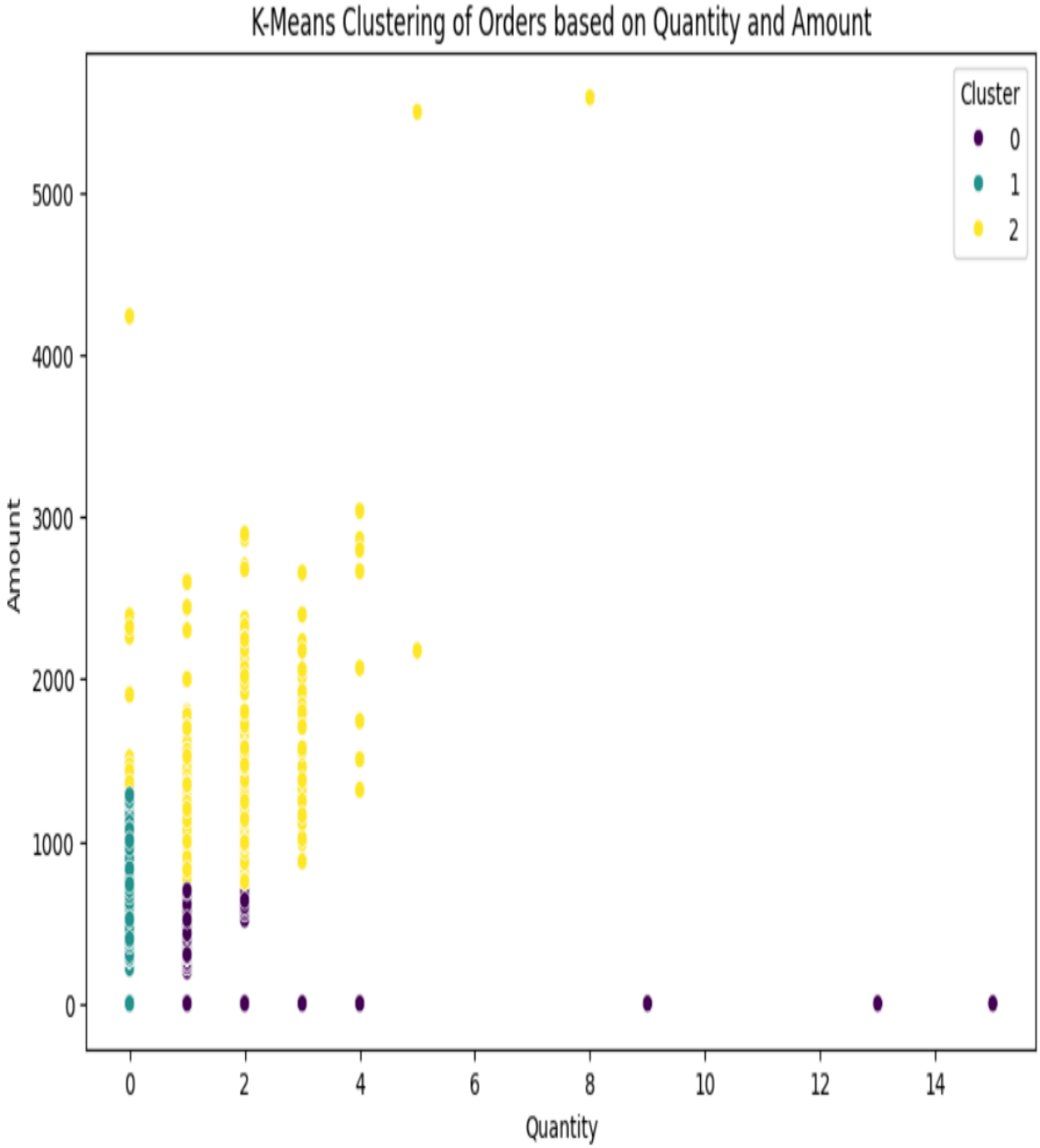
K-Means Clustering Model

K-Means clustering will help identify natural groupings in the data based on Qty and Amount, potentially uncovering hidden patterns, like clusters of orders with similar quantities and amounts.

In [39]:

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(data[['Qty', 'Amount']])
kmeans = KMeans(n_clusters=3, random_state=42)
data['Cluster'] = kmeans.fit_predict(X_scaled)
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Qty', y='Amount', hue='Cluster', data=data,
palette='viridis')
plt.title("K-Means Clustering of Orders based on Quantity and
Amount")
plt.xlabel("Quantity")
plt.ylabel("Amount")
plt.legend(title="Cluster")
plt.show()
print("Cluster Centers:")
print(scaler.inverse_transform(kmeans.cluster_centers_))
```

```
Cluster Centers:
[[ 1.00087574e+00  4.78083100e+02]
 [-1.99840144e-14  2.44949803e+02]
 [ 1.00871884e+00  9.39672354e+02]]
```




```
Cluster Centers:
[[ 1.00087574e+00  4.78083100e+02]
 [-1.99840144e-14  2.44949803e+02]
 [ 1.00871884e+00  9.39672354e+02]]
```

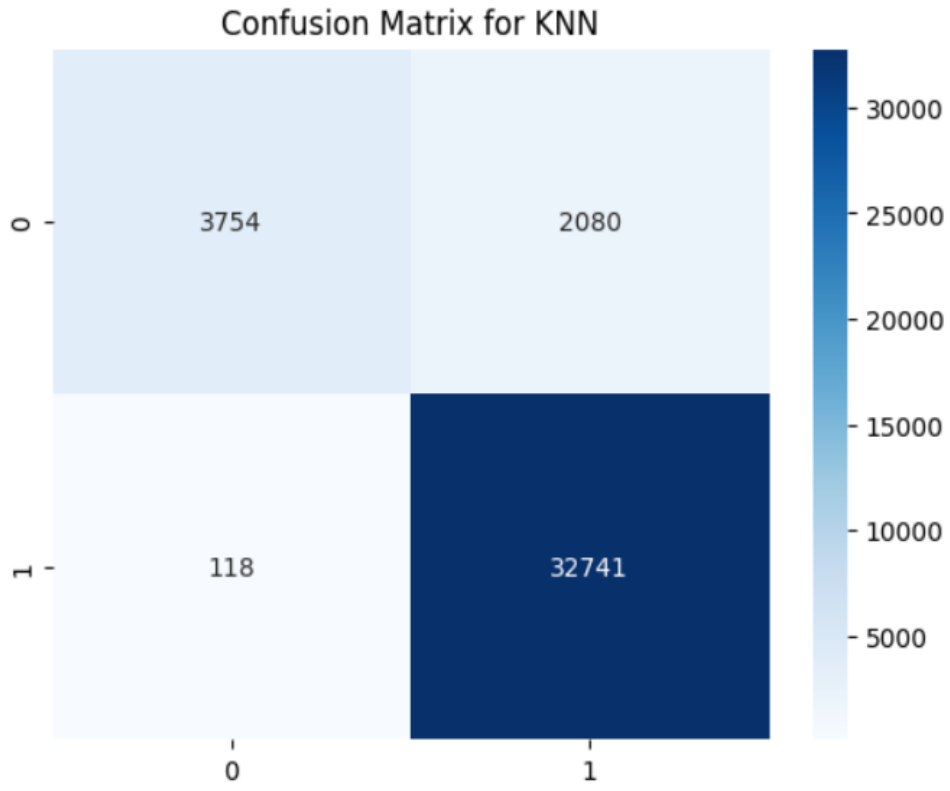
K-Nearest Neighbors (KNN) Model

This model classifies orders based on Qty and Amount to predict whether an order status is “Shipped” or not.

In [40]:

```
knn = KNeighborsClassifier(n_neighbors=5)

knn.fit(X_train, y_train)

y_pred_knn = knn.predict(X_test)

print("KNN Model Accuracy:",

 accuracy_score(y_test, y_pred_knn))

sns.heatmap(confusion_matrix(y_test,

 y_pred_knn), annot=True, fmt="d",

cmap="Blues")

plt.title("Confusion Matrix for KNN")

plt.show()
```

```
KNN Model Accuracy: 0.9431938593544051
```


Confusion Matrix for KNN
3754
2080
118
32741
0
1
0
1
30000
25000
20000
15000
10000
5000

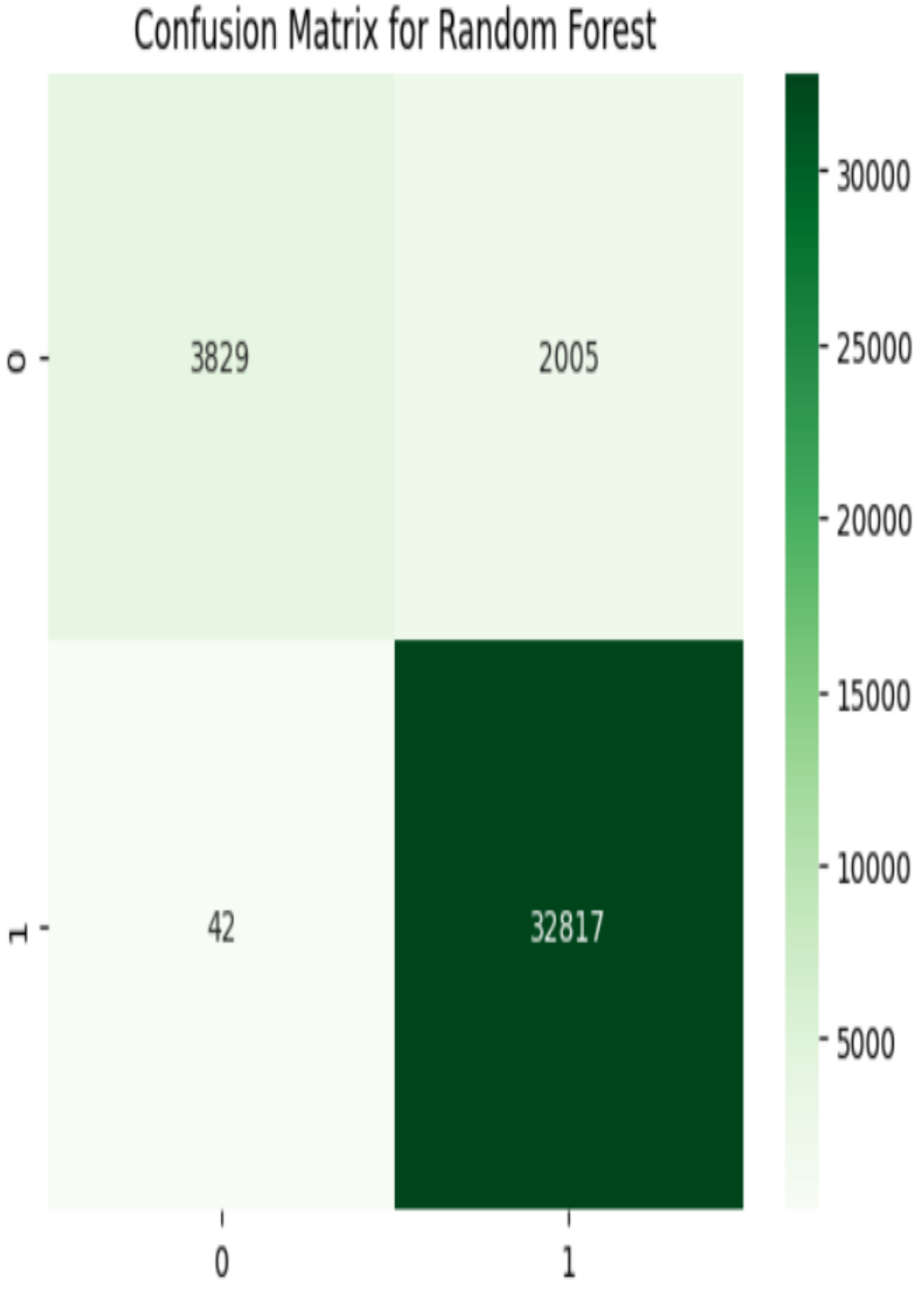
Random Forest Model

Like KNN, Random Forest predicts whether an order status is “Shipped” or not. It uses multiple decision trees to improve accuracy and capture complex patterns in the data.

In [41]:

```
rf = RandomForestClassifier(n_estimators=100,

random_state=42)

rf.fit(X_train, y_train)

y_pred_rf = rf.predict(X_test)

print("Random Forest Accuracy:",

accuracy_score(y_test, y_pred_rf))

sns.heatmap(confusion_matrix(y_test,

y_pred_rf), annot=True, fmt="d", cmap="Greens")

plt.title("Confusion Matrix for Random Forest")

plt.show()
```

```
Random Forest Accuracy: 0.9470963740211408
```


Confusion Matrix for Random Forest
3829
2005
42
32817
0
1
0
1
30000
25000
20000
15000
10000
5000

Linear Regression Model

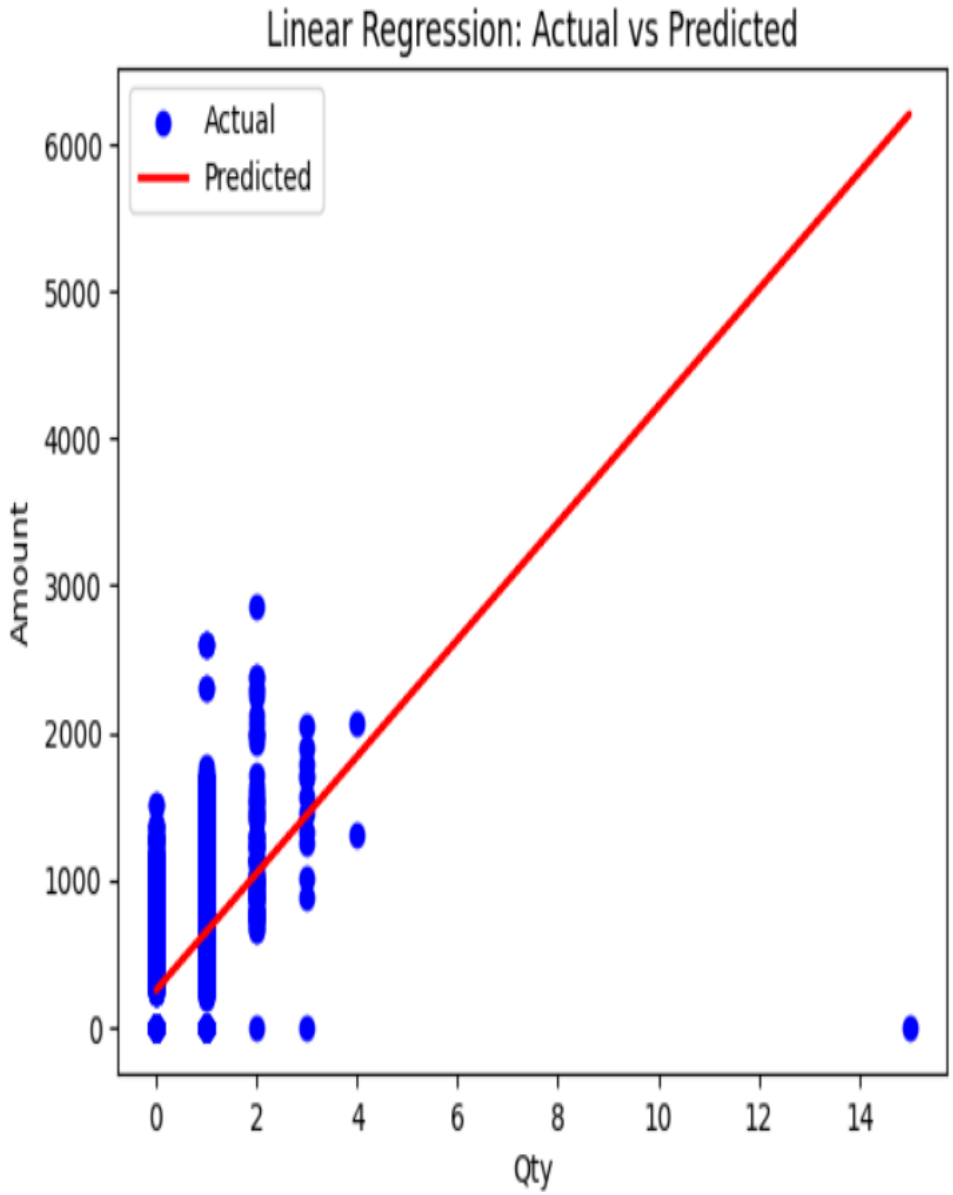
This model uses Qty to predict the Amount, treating Amount as a continuous variable. This can help understand how quantity affects sales amount in general.

In [42]:

```
X_reg = data[['Qty']]
y_reg = data['Amount']
X_train_reg, X_test_reg, y_train_reg,
y_test_reg = train_test_split(X_reg,
y_reg, test_size=0.3, random_state=42)
linear_model = LinearRegression()
linear_model.fit(X_train_reg,
y_train_reg)
y_pred_lr =
linear_model.predict(X_test_reg)
print("Linear Regression MSE:",
mean_squared_error(y_test_reg,
y_pred_lr))
plt.scatter(X_test_reg, y_test_reg,
color="blue", label="Actual")
plt.plot(X_test_reg, y_pred_lr,
color="red", linewidth=2,
label="Predicted")
plt.title("Linear Regression: Actual vs
Predicted")
plt.xlabel("Qty")
plt.ylabel("Amount")
plt.legend()
plt.show()
```

```
Linear Regression MSE: 83885.2308043057
```

```
Linear Regression MSE: 83885.2308043057
```



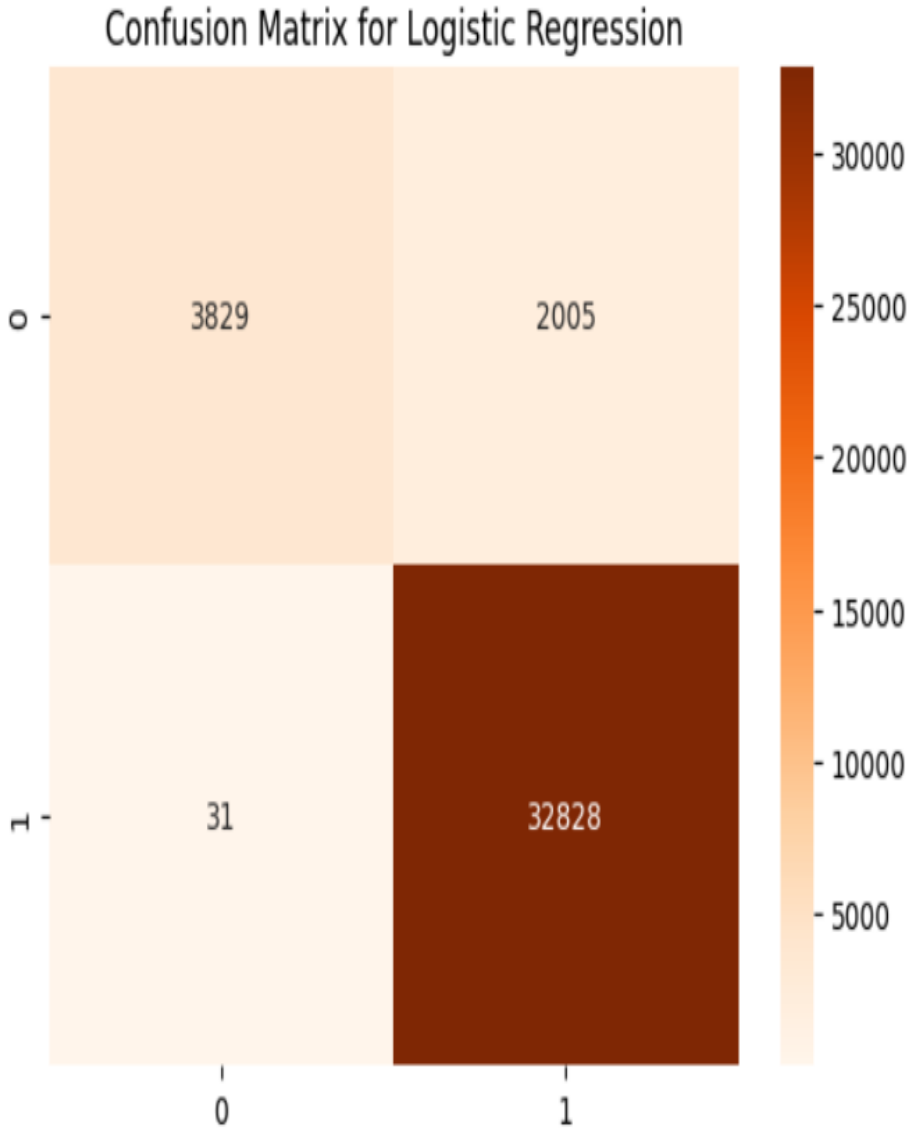
Logistic Regression

Similar to KNN and Random Forest, Logistic Regression predicts whether the order status is “Shipped” based on Qty and Amount. This model is particularly good for binary classification.

In [43]:

```
logistic_model = LogisticRegression()

logistic_model.fit(X_train, y_train)

y_pred_log = logistic_model.predict(X_test)

print("Logistic Regression Accuracy:",

accuracy_score(y_test, y_pred_log))

sns.heatmap(confusion_matrix(y_test, y_pred_log),

annot=True, fmt="d", cmap="Oranges")

plt.title("Confusion Matrix for Logistic Regression")

plt.show()
```

```
Logistic Regression Accuracy: 0.9473806631690487
```


Confusion Matrix for Logistic Regression
3829
2005
31
32828
0
1
0
1
30000
25000
20000
15000
10000
5000

Comparison in Performance

The classification report for each model (KNN, Random Forest, and Logistic Regression) allows you to compare their effectiveness in predicting the shipping status.
It helps us gain insight into which model performed best on this dataset in terms of accuracy and the balance between correctly identifying shipped and non-shipped statuses

.

In [44]:

```
print("KNN Classification Report:\n",

classification_report(y_test, y_pred_knn))

print("Random Forest Classification Report:\n",

classification_report(y_test, y_pred_rf))

print("Logistic Regression Classification Report:\n",

classification_report(y_test, y_pred_log))

print("\nK-Means Clustering Summary:")

print("Number of Clusters: 3")

print("Cluster Sizes:",

data['Cluster'].value_counts().to_dict())

print("Cluster Centers (Qty and Amount):\n",


scaler.inverse_transform(kmeans.cluster_centers_))
```

KNN Classification Report:

	precision	recall	f1-score	support
0	0.97	0.64	0.77	5834
1	0.94	1.00	0.97	32859
accuracy			0.94	38693
macro avg	0.95	0.82	0.87	38693
weighted avg	0.94	0.94	0.94	38693

Random Forest Classification Report:

	precision	recall	f1-score	support
0	0.99	0.66	0.79	5834
1	0.94	1.00	0.97	32859
accuracy			0.95	38693
macro avg	0.97	0.83	0.88	38693
weighted avg	0.95	0.95	0.94	38693

Logistic Regression Classification Report:

	precision	recall	f1-score	support
0	0.99	0.66	0.79	5834
1	0.94	1.00	0.97	32859
accuracy			0.95	38693
macro avg	0.97	0.83	0.88	38693
weighted avg	0.95	0.95	0.94	38693

K-Means Clustering Summary:
Number of Clusters: 3
Cluster Sizes: {0: 72744, 2: 43462, 1: 12770}
Cluster Centers (Qty and Amount):
 [[1.00087574e+00 4.78083100e+02]
 [-1.99840144e-14 2.44949803e+02]
 [1.00871884e+00 9.39672354e+02]]

Future Scope of Improvement

The **future scope of improvement** for analyzing an Amazon sales report can include several aspects, such as improving data collection, analysis techniques, visualization, and actionable insights. Here's a structured list:

---

1. Enhanced Data Collection

- **Real-Time Data Integration**: Incorporate real-time sales tracking to provide up-to-date insights and faster decision-making.
- **Inclusion of External Factors**:
  - Competitor pricing and sales data.
  - Seasonal trends or external events (e.g., holidays, economic shifts).
- **Advanced Metrics**:
  - Customer lifetime value (CLV).
  - Cart abandonment rates and recovery insights.
  - Impact of promotions and discounts.

---

2. Advanced Data Analysis

- **Predictive Analytics**:
  - Use AI/ML models to forecast future sales based on historical trends, demand patterns, and external factors.
- **Sentiment Analysis**:
  - Analyze customer reviews and feedback to understand product performance and customer satisfaction.
- **Market Segmentation**:
  - Break down sales by customer demographics, regions, or product categories for targeted strategies.

---

3. Improved Visualization and Reporting

- **Dynamic Dashboards**:
  - Create interactive dashboards with tools like Tableau or Power BI for better data exploration.
- **Customizable Reports**:
  - Allow stakeholders to customize reports based on their needs (e.g., regional managers vs. product teams).
- **Heatmaps and Geo-Analytics**:
  - Visualize regional performance and identify underperforming areas.

---

4. Actionable Insights

- **Recommendation Systems**:
  - Suggest inventory restocking, dynamic pricing adjustments, or product bundling strategies.
- **Competitor Benchmarking**:
  - Identify areas where your sales lag behind competitors.
- **Marketing Campaign ROI**:
  - Attribute sales to specific campaigns to optimize spending.

---

5. Automation and Scalability

- **Automated Alerts**:
  - Notify teams of unusual sales patterns, stockouts, or performance dips.
- **Scalable Systems**:
  - Ensure systems can handle growing data volumes as the business scales.

---

6. Customer-Centric Improvements

- **Improved Personalization**:
  - Use data to tailor product recommendations and offers.
- **Customer Retention Analysis**:
  - Identify factors driving repeat purchases and reduce churn rates.

---

7. Integration with Other Tools

- **Third-Party APIs**:
  - Integrate with Amazon's Selling Partner API for seamless data synchronization.
- **Omni-Channel Sales Analysis**:
  - Combine Amazon data with data from other platforms like eBay, Shopify, or offline stores.

---

8. Focus on Sustainability

- Analyze trends in eco-friendly product sales and optimize inventory to align with growing consumer demand for sustainability.

Certificate

This is to certify that Mr. Ayan Kundu of Asansol Engineering College, University Roll number: 10800222015, has successfully completed a project on Amazon Sales Report Analysis using Machine Learning with Python under the guidance of Prof. Arnab Chakraborty.

Prof. Arnab Chakraborty Globsyn Finishing School

Certificate

This is to certify that Miss. Moupiya Das of Asansol Engineering College, University Roll number: 10800222020, has successfully completed a project on Amazon Sales Report Analysis using Machine Learning with Python under the guidance of Prof. Arnab Chakraborty.

Prof. Arnab Chakraborty Globsyn Finishing School

Certificate

This is to certify that Miss. Neloofar Yasmeen of Asansol Engineering College, University Roll number: 10800222022, has successfully completed a project on Amazon Sales Report Analysis using Machine Learning with Python under the guidance of Prof. Arnab Chakraborty.

Prof. Arnab Chakraborty Globsyn Finishing School