Slash Data Analysis task

Author: Shrouk Mohamed Alaa

Slash Data Analysis Task

Objective:

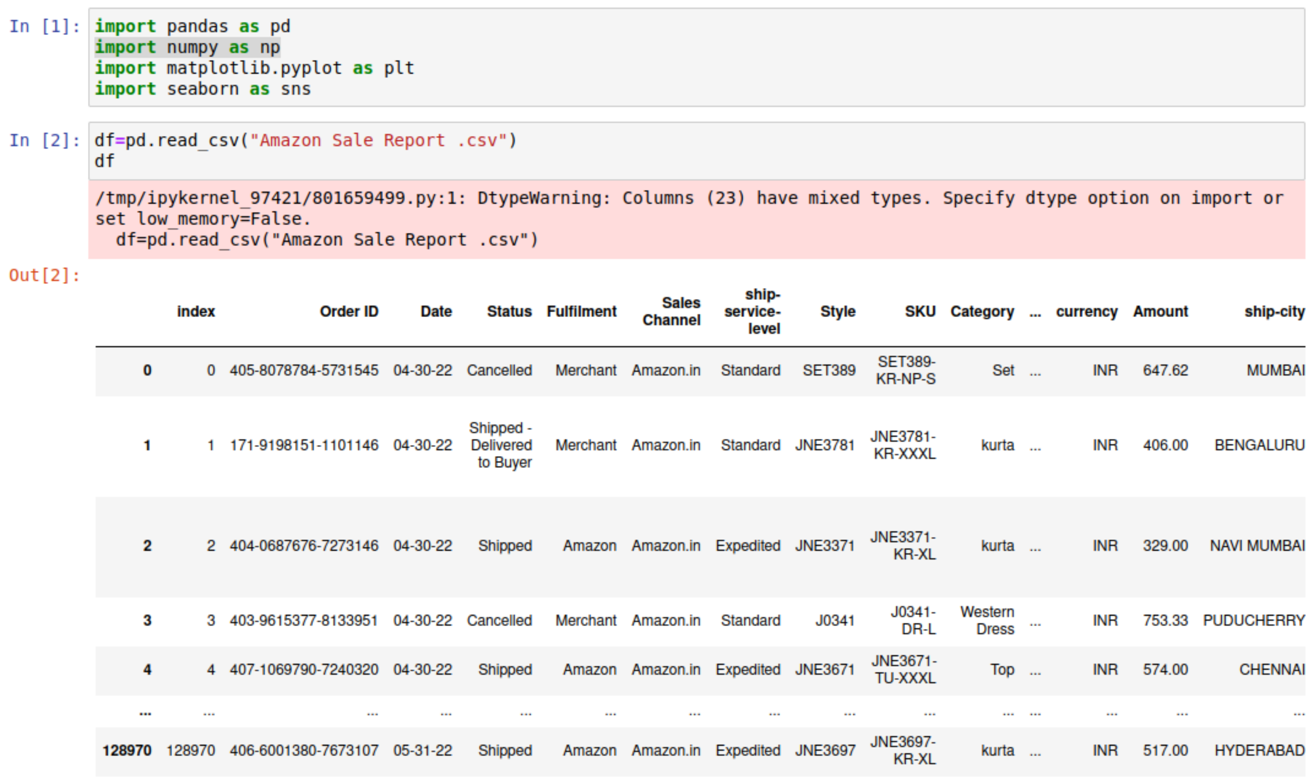
Analyzed the Amazon sales dataset to extract meaningful insights,

preprocess the data, create visualizations using Python libraries (matplotlib and seaborn),

built predictive models, and developed a dashboard for comprehensive data presentation.

1-Exploratory Data Analysis (EDA)

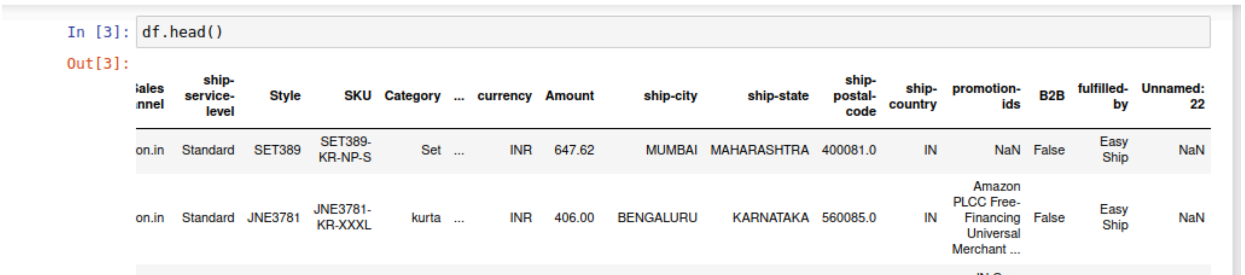
- Importing python libraries for the dataset ,reading the dataset.



```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns

In [2]: df=pd.read_csv("Amazon Sale Report .csv")
        df

/tmp/ipykernel_97421/801659499.py:1: DtypeWarning: Columns (23) have mixed types. Specify dtype option on import or
set low_memory=False.
  df=pd.read_csv("Amazon Sale Report .csv")

Out[2]:
```

	index	Order ID	Date	Status	Fulfilment	Sales Channel	ship-service-level	Style	SKU	Category	...	currency	Amount	ship-city
0	0	405-8078784-5731545	04-30-22	Cancelled	Merchant	Amazon.in	Standard	SET389	SET389-KR-NP-S	Set	...	INR	647.62	MUMBAI
1	1	171-9198151-1101146	04-30-22	Shipped - Delivered to Buyer	Merchant	Amazon.in	Standard	JNE3781	JNE3781-KR-XXXL	kurta	...	INR	406.00	BENGALURU
2	2	404-0687676-7273146	04-30-22	Shipped	Amazon	Amazon.in	Expedited	JNE3371	JNE3371-KR-XL	kurta	...	INR	329.00	NAVI MUMBAI
3	3	403-9615377-8133951	04-30-22	Cancelled	Merchant	Amazon.in	Standard	J0341	J0341-DR-L	Western Dress	...	INR	753.33	PUDUCHERRY
4	4	407-1069790-7240320	04-30-22	Shipped	Amazon	Amazon.in	Expedited	JNE3671	JNE3671-TU-XXXL	Top	...	INR	574.00	CHENNAI
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
128970	128970	406-6001380-7673107	05-31-22	Shipped	Amazon	Amazon.in	Expedited	JNE3697	JNE3697-KR-XL	kurta	...	INR	517.00	HYDERABAD

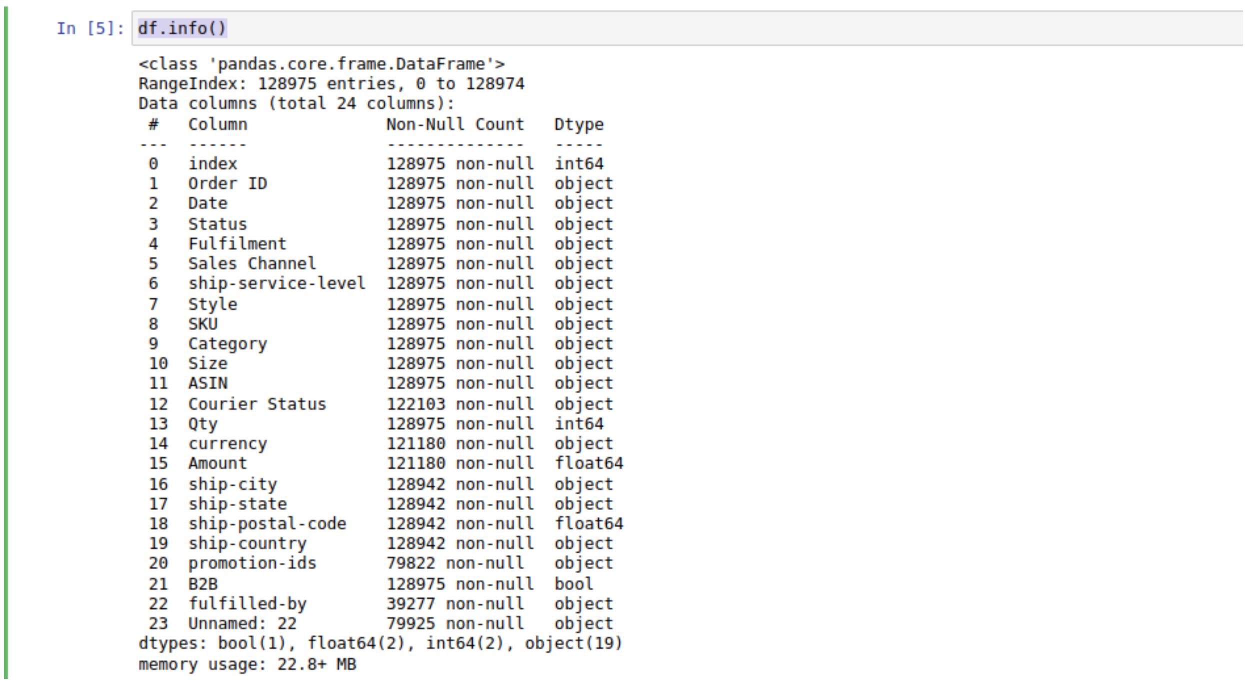
- Display the first 5 Rows & defining the data types of the variables



```
In [3]: df.head()

Out[3]:
```

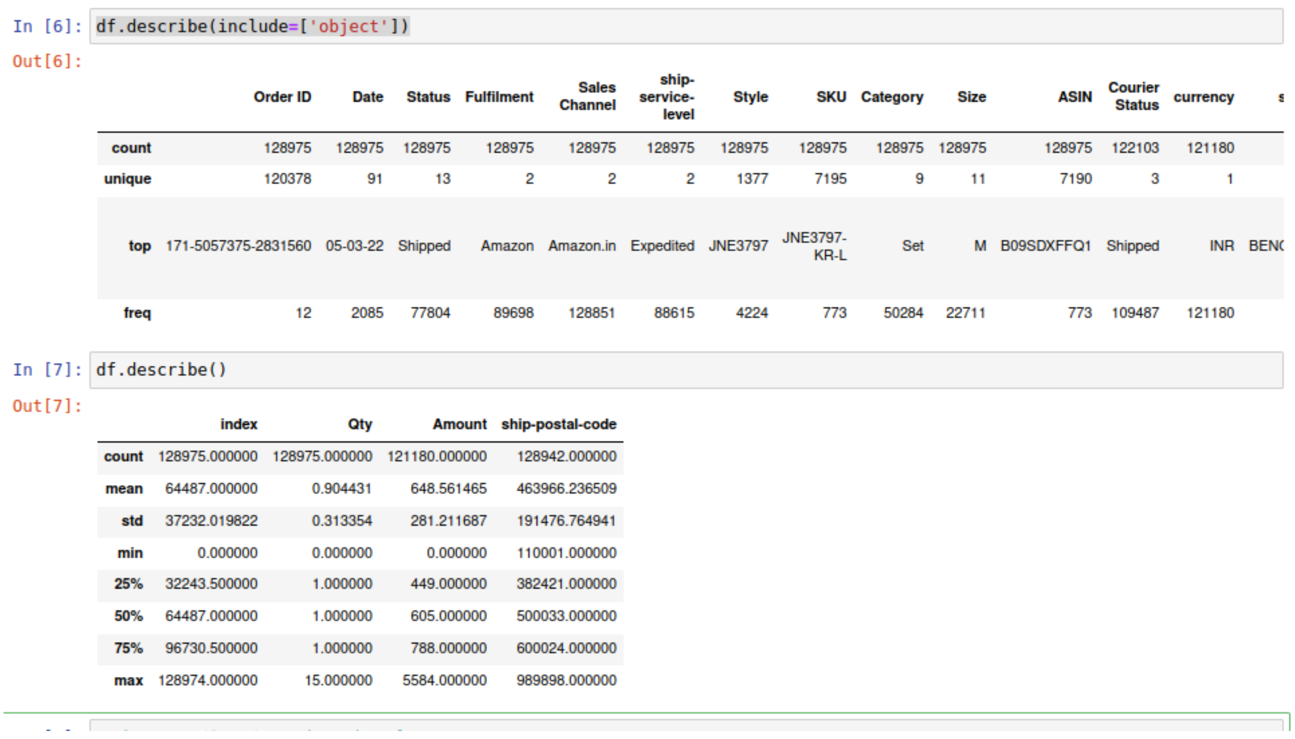
ales nnel	ship-service-level	Style	SKU	Category	...	currency	Amount	ship-city	ship-state	ship-postal-code	ship-country	promotion-ids	B2B	fulfilled-by	Unnamed: 22
on.in	Standard	SET389	SET389-KR-NP-S	Set	...	INR	647.62	MUMBAI	MAHARASHTRA	400081.0	IN	NaN	False	Easy Ship	NaN
on.in	Standard	JNE3781	JNE3781-KR-XXXL	kurta	...	INR	406.00	BENGALURU	KARNATAKA	560085.0	IN	Amazon PLCC Free-Financing Universal Merchant ...	False	Easy Ship	NaN

- display a concise summary of a The DataFrame



```
In [5]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 128975 entries, 0 to 128974
Data columns (total 24 columns):
 #   Column              Non-Null Count   Dtype  
---  ------              --------------   -----  
 0   index               128975 non-null  int64  
 1   Order ID            128975 non-null  object 
 2   Date                128975 non-null  object 
 3   Status              128975 non-null  object 
 4   Fulfilment          128975 non-null  object 
 5   Sales Channel       128975 non-null  object 
 6   ship-service-level  128975 non-null  object 
 7   Style               128975 non-null  object 
 8   SKU                 128975 non-null  object 
 9   Category            128975 non-null  object 
 10  Size                128975 non-null  object 
 11  ASIN                128975 non-null  object 
 12  Courier Status      122103 non-null  object 
 13  Qty                 128975 non-null  int64  
 14  currency            121180 non-null  object 
 15  Amount              121180 non-null  float64
 16  ship-city           128942 non-null  object 
 17  ship-state          128942 non-null  object 
 18  ship-postal-code    128942 non-null  float64
 19  ship-country        128942 non-null  object 
 20  promotion-ids       79822 non-null   object 
 21  B2B                 128975 non-null  bool   
 22  fulfilled-by        39277 non-null   object 
 23  Unnamed: 22         79925 non-null   object 
dtypes: bool(1), float64(2), int64(2), object(19)
memory usage: 22.8+ MB
```

- Generate summary statistics for numerical and categorical variables



```
In [6]: df.describe(include=['object'])
Out[6]:
```

	Order ID	Date	Status	Fulfilment	Sales Channel	ship-service-level	Style	SKU	Category	Size	ASIN	Courier Status	currency	s
count	128975	128975	128975	128975	128975	128975	128975	128975	128975	128975	128975	122103	121180	
unique	120378	91	13	2	2	2	1377	7195	9	11	7190	3	1	
top	171-5057375-2831560	05-03-22	Shipped	Amazon	Amazon.in	Expedited	JNE3797	JNE3797-KR-L	Set	M	B09SDXFFQ1	Shipped	INR	BEN
freq	12	2085	77804	89698	128851	88615	4224	773	50284	22711	773	109487	121180	

```
In [7]: df.describe()
Out[7]:
```

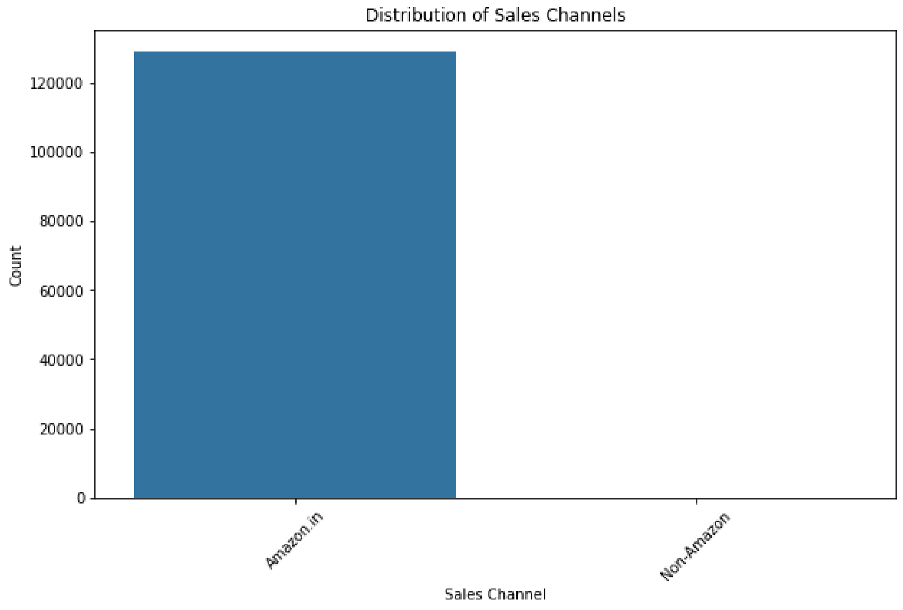
	index	Qty	Amount	ship-postal-code
count	128975.000000	128975.000000	121180.000000	128942.000000
mean	64487.000000	0.904431	648.561465	463966.236509
std	37232.019822	0.313354	281.211687	191476.764941
min	0.000000	0.000000	0.000000	110001.000000
25%	32243.500000	1.000000	449.000000	382421.000000
50%	64487.000000	1.000000	605.000000	500033.000000
75%	96730.500000	1.000000	788.000000	600024.000000
max	128974.000000	15.000000	5584.000000	989898.000000

- Visualize the distribution of key features to identify trends and patterns.

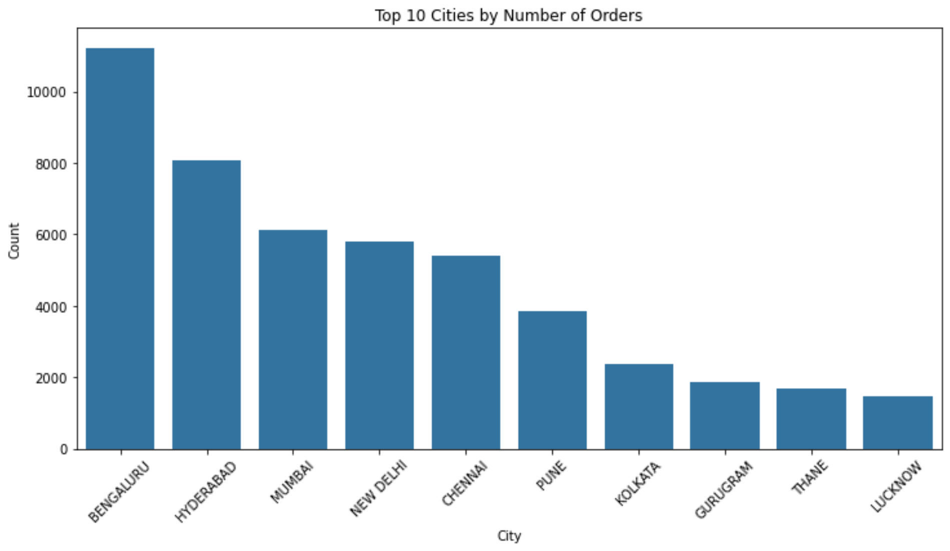
In [86]:
```python
# Sales by Date
plt.figure(figsize=(12, 6))
df.groupby(df['Date'].dt.to_period('M')).size().plot(kind='bar')
plt.title('Number of Orders by Month')
plt.xlabel('Month')
plt.ylabel('Number of Orders')
plt.xticks(rotation=45)
plt.show()
```

Number of Orders by Month

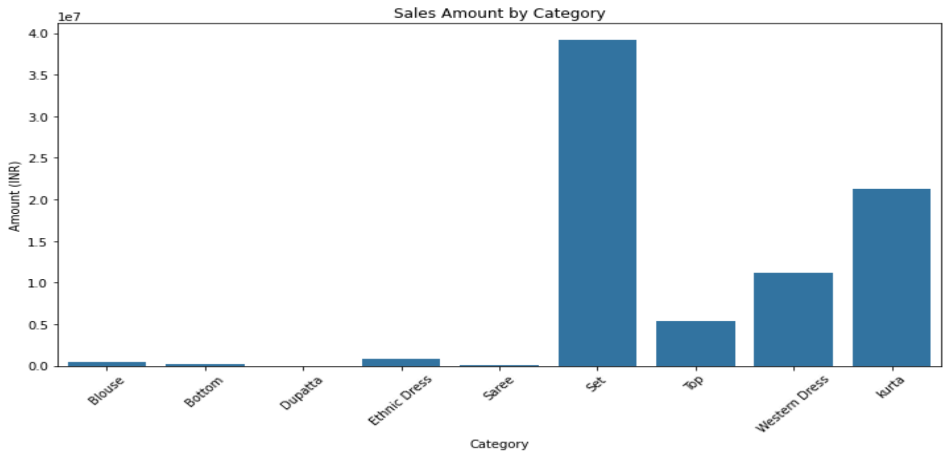
In [87]:
```python
# Sales Channel Distribution
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='Sales Channel')
plt.title('Distribution of Sales Channels')
plt.xlabel('Sales Channel')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```



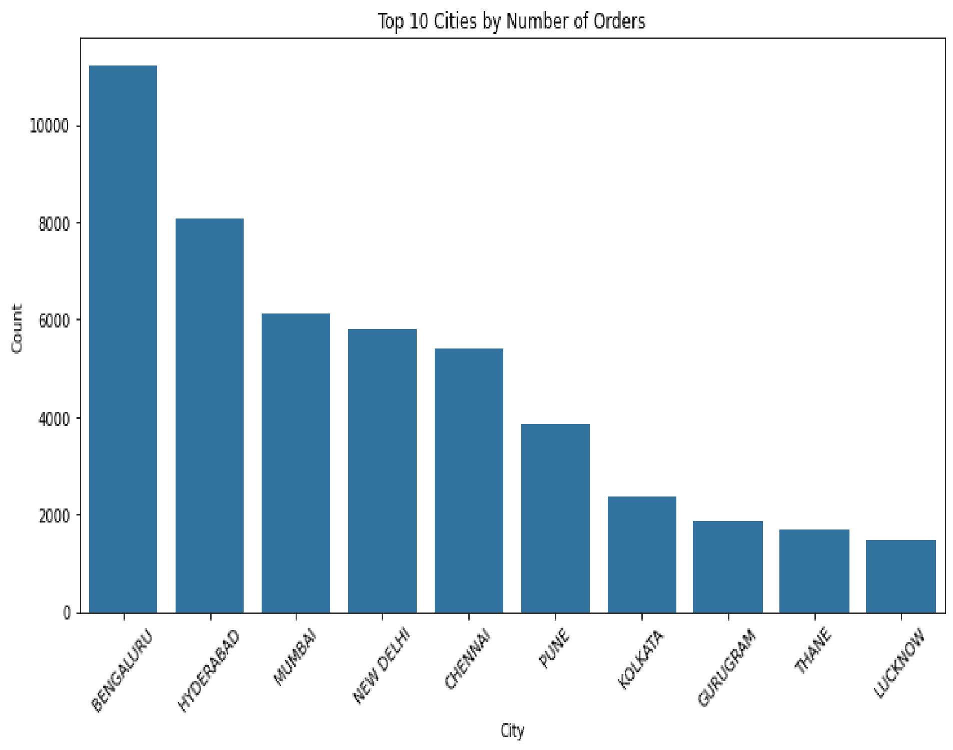
In [91]:
```python
# Sales by Region (Ship City)
plt.figure(figsize=(12, 6))
top_cities = df['ship-city'].value_counts().head(10).index
sns.countplot(data=df[df['ship-city'].isin(top_cities)], x='ship-city', order=top_cities)
plt.title('Top 10 Cities by Number of Orders')
plt.xlabel('City')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```



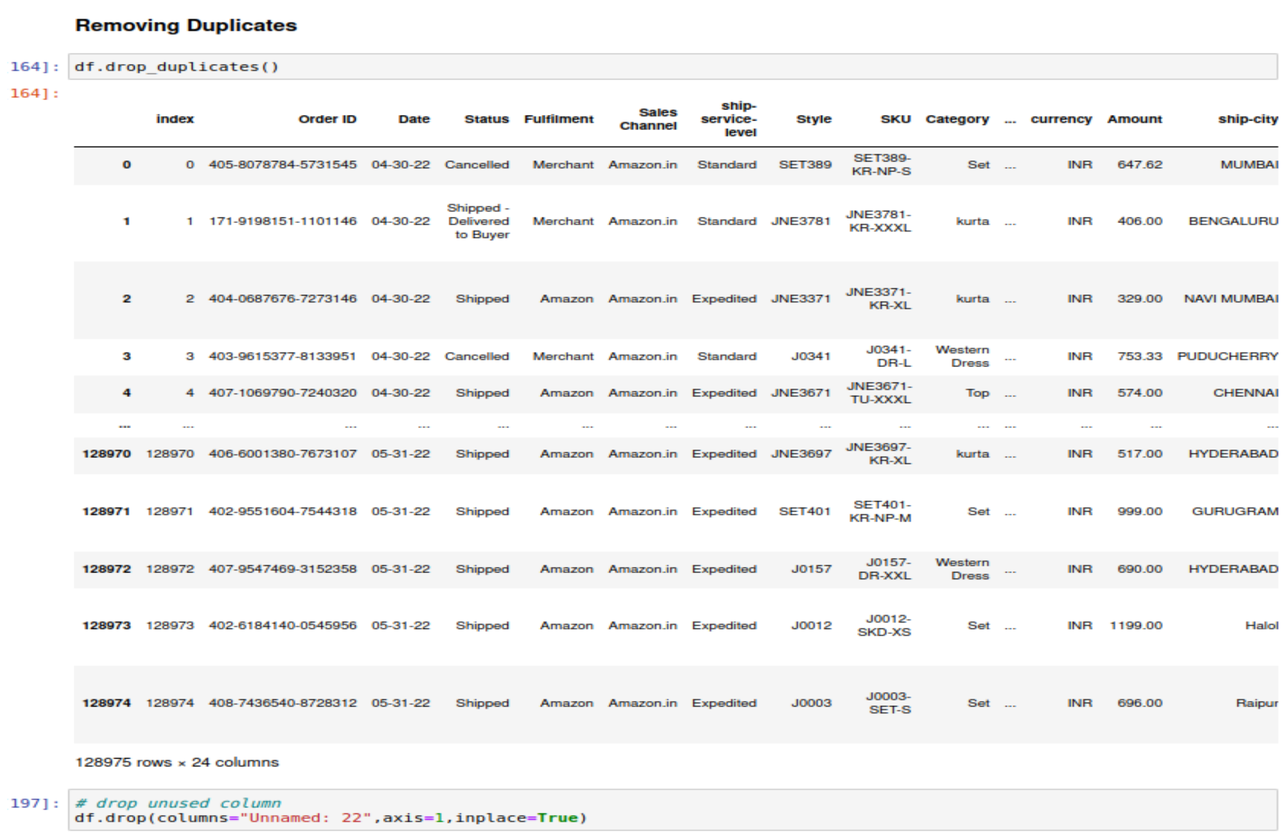
In [93]:
```python
# Aggregating the data to get the sum of 'Amount' by 'Category'
category_sales = df.groupby('Category')['Amount'].sum().reset_index()

# Plotting the bar chart
plt.figure(figsize=(12, 6))
sns.barplot(data=category_sales, x='Category', y='Amount')
plt.title('Sales Amount by Category')
plt.xlabel('Category')
plt.ylabel('Amount (INR)')
plt.xticks(rotation=45)
plt.show()
```



In [91]:
```python
# Sales by Region (Ship City)
plt.figure(figsize=(12, 6))
top_cities = df['ship-city'].value_counts().head(10).index
sns.countplot(data=df[df['ship-city'].isin(top_cities)], x='ship-city', order=top_cities)
plt.title('Top 10 Cities by Number of Orders')
plt.xlabel('City')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```



2-Data Preprocessing



Removing Duplicates

```
164]: df.drop_duplicates()
```

164]:

	index	Order ID	Date	Status	Fulfilment	Sales Channel	ship-service-level	Style	SKU	Category	...	currency	Amount	ship-city
0	0	405-8078784-5731545	04-30-22	Cancelled	Merchant	Amazon.in	Standard	SET389	SET389-KR-NP-S	Set	...	INR	647.62	MUMBAI
1	1	171-9198151-1101146	04-30-22	Shipped - Delivered to Buyer	Merchant	Amazon.in	Standard	JNE3781	JNE3781-KR-XXXL	kurta	...	INR	406.00	BENGALURU
2	2	404-0687676-7273146	04-30-22	Shipped	Amazon	Amazon.in	Expedited	JNE3371	JNE3371-KR-XL	kurta	...	INR	329.00	NAVI MUMBAI
3	3	403-9615377-8133951	04-30-22	Cancelled	Merchant	Amazon.in	Standard	J0341	J0341-DR-L	Western Dress	...	INR	753.33	PUDUCHERRY
4	4	407-1069790-7240320	04-30-22	Shipped	Amazon	Amazon.in	Expedited	JNE3671	JNE3671-TU-XXXL	Top	...	INR	574.00	CHENNAI
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
128970	128970	406-6001380-7673107	05-31-22	Shipped	Amazon	Amazon.in	Expedited	JNE3697	JNE3697-KR-XL	kurta	...	INR	517.00	HYDERABAD
128971	128971	402-9551604-7544318	05-31-22	Shipped	Amazon	Amazon.in	Expedited	SET401	SET401-KR-NP-M	Set	...	INR	999.00	GURUGRAM
128972	128972	407-9547469-3152358	05-31-22	Shipped	Amazon	Amazon.in	Expedited	J0157	J0157-DR-XXL	Western Dress	...	INR	690.00	HYDERABAD
128973	128973	402-6184140-0545956	05-31-22	Shipped	Amazon	Amazon.in	Expedited	J0012	J0012-SKD-XS	Set	...	INR	1199.00	Halol
128974	128974	408-7436540-8728312	05-31-22	Shipped	Amazon	Amazon.in	Expedited	J0003	J0003-SET-S	Set	...	INR	696.00	Raipur

128975 rows × 24 columns

```
197]: # drop unused column
      df.drop(columns="Unnamed: 22",axis=1,inplace=True)
```

Handling Missing Values

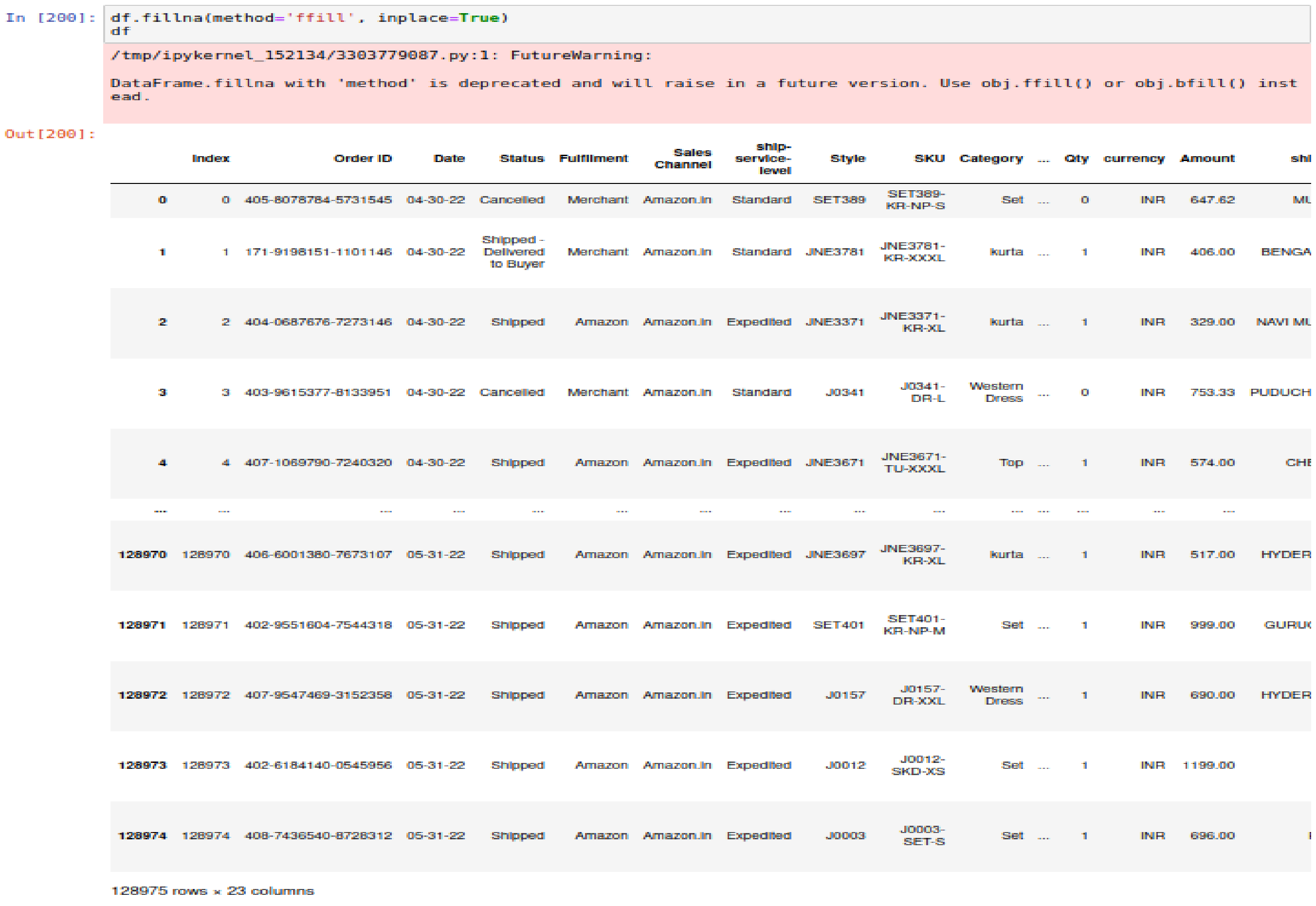
```
In [198]: df.isnull().sum()
```

```
Out[198]: index                     0
          Order ID                  0
          Date                      0
          Status                    0
          Fulfilment                0
          Sales Channel             0
          ship-service-level        0
          Style                     0
          SKU                       0
          Category                  0
          Size                      0
          ASIN                      0
          Courier Status         6872
          Qty                       0
          currency               7795
          Amount                 7795
          ship-city                33
          ship-state               33
          ship-postal-code         33
          ship-country             33
          promotion-ids         49153
          B2B                       0
          fulfilled-by          89698
          dtype: int64
```

```
In [199]: df.isnull().sum().sum()
```

```
Out[199]: 161445
```

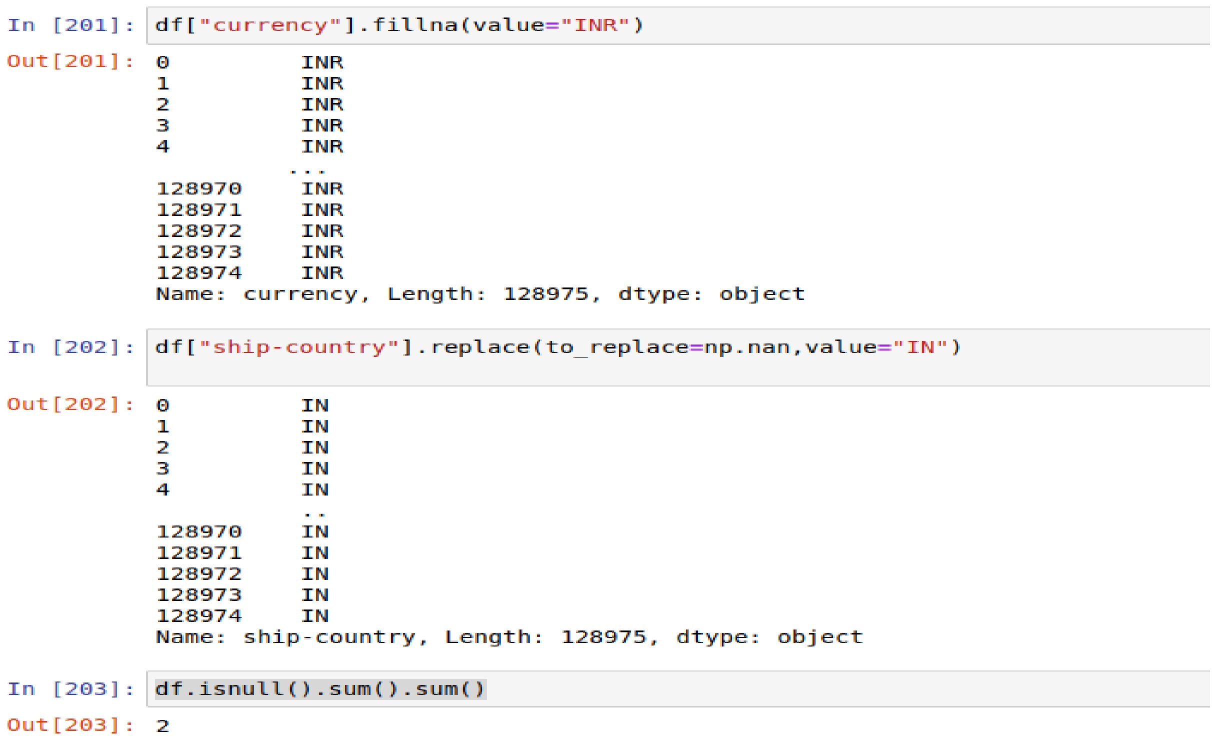
- Filling the cells of missing values



```
In [200]: df.fillna(method='ffill', inplace=True)
          df
          /tmp/ipykernel_152134/3303779087.py:1: FutureWarning:

          DataFrame.fillna with 'method' is deprecated and will raise in a future version. Use obj.ffill() or obj.bfill() inst
          ead.
Out[200]:
```

	Index	Order ID	Date	Status	Fulfilment	Sales Channel	ship-service-level	Style	SKU	Category	...	Qty	currency	Amount	shi
0	0	405-8078784-5731545	04-30-22	Cancelled	Merchant	Amazon.in	Standard	SET389	SET389-KR-NP-S	Set	...	0	INR	647.62	MU
1	1	171-9198151-1101146	04-30-22	Shipped - Delivered to Buyer	Merchant	Amazon.in	Standard	JNE3781	JNE3781-KR-XXXL	kurta	...	1	INR	406.00	BENGA
2	2	404-0687676-7273146	04-30-22	Shipped	Amazon	Amazon.in	Expedited	JNE3371	JNE3371-KR-XL	kurta	...	1	INR	329.00	NAVI MU
3	3	403-9615377-8133951	04-30-22	Cancelled	Merchant	Amazon.in	Standard	J0341	J0341-DR-L	Western Dress	...	0	INR	753.33	PUDUCH
4	4	407-1069790-7240320	04-30-22	Shipped	Amazon	Amazon.in	Expedited	JNE3671	JNE3671-TU-XXXL	Top	...	1	INR	574.00	CHE
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
128970	128970	406-6001380-7673107	05-31-22	Shipped	Amazon	Amazon.in	Expedited	JNE3697	JNE3697-KR-XL	kurta	...	1	INR	517.00	HYDER
128971	128971	402-9551604-7544318	05-31-22	Shipped	Amazon	Amazon.in	Expedited	SET401	SET401-KR-NP-M	Set	...	1	INR	999.00	GURU
128972	128972	407-9547469-3152358	05-31-22	Shipped	Amazon	Amazon.in	Expedited	J0157	J0157-DR-XXL	Western Dress	...	1	INR	690.00	HYDER
128973	128973	402-6184140-0545956	05-31-22	Shipped	Amazon	Amazon.in	Expedited	J0012	J0012-SKD-XS	Set	...	1	INR	1199.00	
128974	128974	408-7436540-8728312	05-31-22	Shipped	Amazon	Amazon.in	Expedited	J0003	J0003-SET-S	Set	...	1	INR	696.00	I

128975 rows × 23 columns



```
In [201]: df["currency"].fillna(value="INR")
Out[201]: 0         INR
          1         INR
          2         INR
          3         INR
          4         INR
                   ...
          128970    INR
          128971    INR
          128972    INR
          128973    INR
          128974    INR
          Name: currency, Length: 128975, dtype: object

In [202]: df["ship-country"].replace(to_replace=np.nan,value="IN")

Out[202]: 0         IN
          1         IN
          2         IN
          3         IN
          4         IN
                    ..
          128970    IN
          128971    IN
          128972    IN
          128973    IN
          128974    IN
          Name: ship-country, Length: 128975, dtype: object

In [203]: df.isnull().sum().sum()
Out[203]: 2
```

```
In [205]: df.isnull().sum()
```

```
Out[205]: index                  0
          Order ID               0
          Date                   0
          Status                 0
          Fulfilment             0
          Sales Channel          0
          ship-service-level     0
          Style                  0
          SKU                    0
          Category               0
          Size                   0
          ASIN                   0
          Courier Status         1
          Qty                    0
          currency               0
          Amount                 0
          ship-city              0
          ship-state             0
          ship-postal-code       0
          ship-country           0
          promotion-ids          1
          B2B                    0
          fulfilled-by           0
          dtype: int64
```

```
In [208]:  #Drop rows with NaN values that remain
          df.dropna(inplace=True)
```

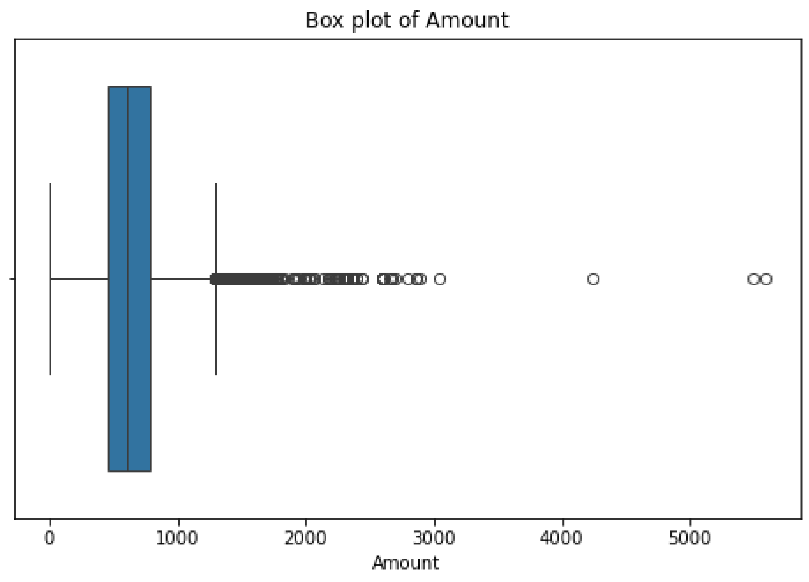
```
In [209]: df.isnull().sum().sum()
```

```
Out[209]: 0
```

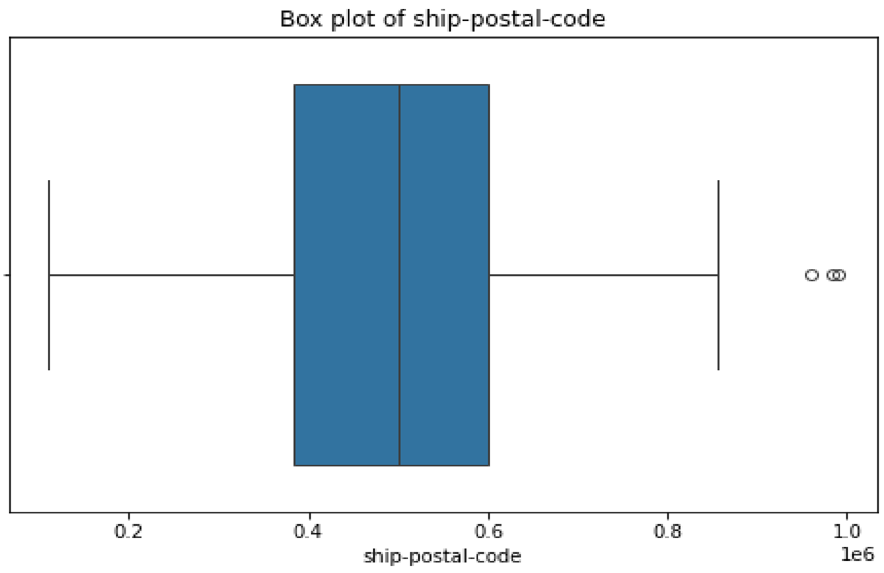
```
In [204]: # Convert 'Date' to datetime format
          df['Date'] = pd.to_datetime(df['Date'], format='%m-%d-%y', errors='coerce')
```

Determine outliers using Box Plot

```
In [65]: plt.figure(figsize=(8, 5))
         sns.boxplot(x=df['Amount'])
         plt.title('Box plot of Amount')
         plt.show()
```



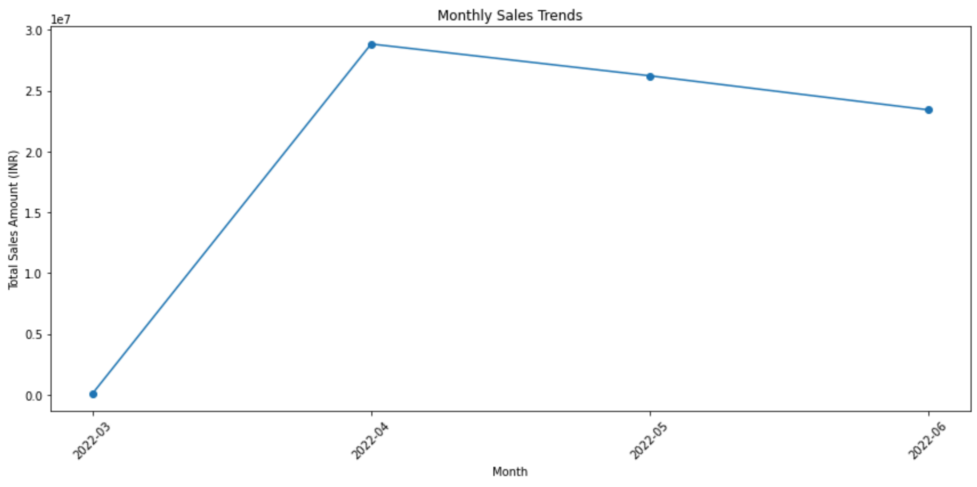
```
In [67]: plt.figure(figsize=(8, 5))
         sns.boxplot(x=df['ship-postal-code'])
         plt.title('Box plot of ship-postal-code')
         plt.show()
```



3- Data Visualization

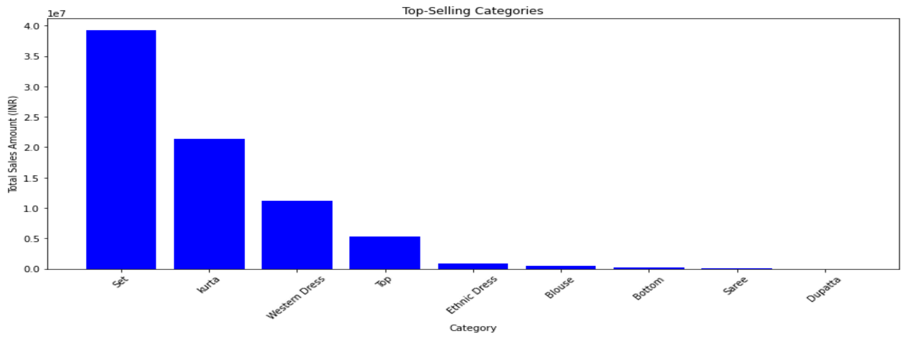
- Sales Monthly Trends

```
In [56]: df['Date'] = pd.to_datetime(df['Date'])

         df['MonthYear'] = df['Date'].dt.to_period('M')
         monthly_sales = df.groupby('MonthYear')['Amount'].sum().reset_index()
         monthly_sales['MonthYear'] = monthly_sales['MonthYear'].astype(str)


         # Plotting the monthly sales trends
         plt.figure(figsize=(12, 6))
         plt.plot(monthly_sales['MonthYear'].values , monthly_sales['Amount'].values,  marker='o',linestyle='-')
         plt.title('Monthly Sales Trends')
         plt.xlabel('Month')
         plt.ylabel('Total Sales Amount (INR)')
         plt.xticks(rotation=45)
         plt.tight_layout()
         plt.show()
```



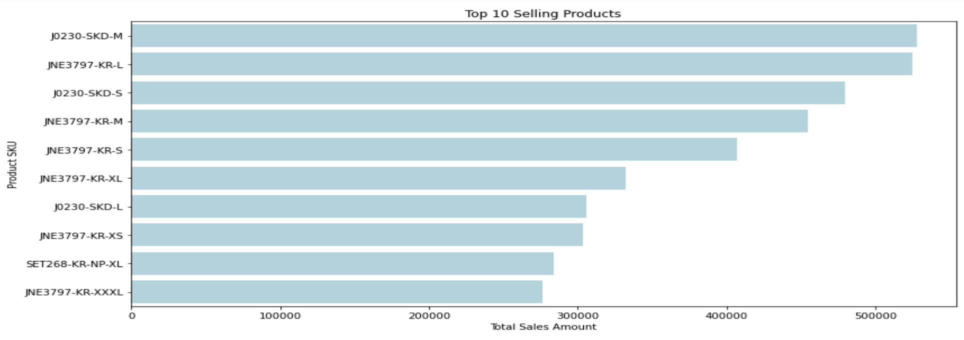
- Top Selling Categories

```
In [53]: # Top-Selling categories
         # Aggregate sales data by category
         top_categories = df.groupby('Category')['Amount'].sum().sort_values(ascending=False).reset_index()

         # Plotting the top-selling categories
         plt.figure(figsize=(12, 6))
         plt.bar(top_categories['Category'], top_categories['Amount'], color='blue')
         plt.title('Top-Selling Categories')
         plt.xlabel('Category')
         plt.ylabel('Total Sales Amount (INR)')
         plt.xticks(rotation=45)
         plt.tight_layout()
         plt.show()
```



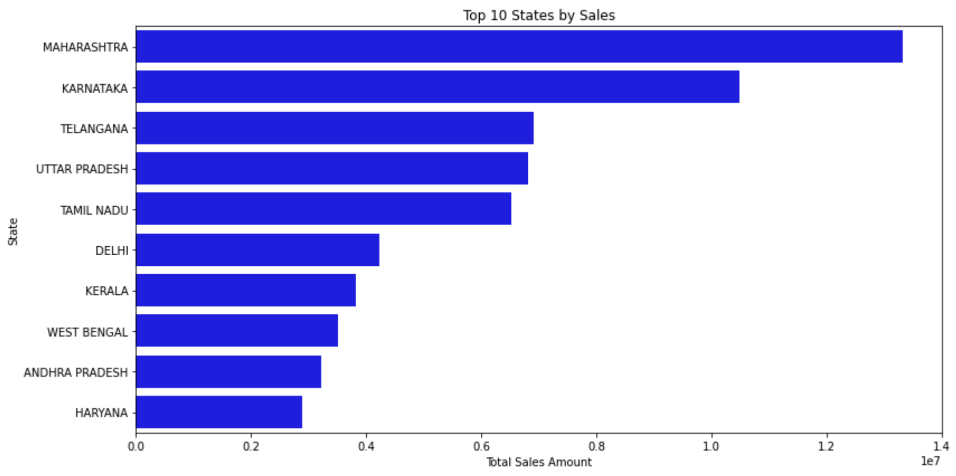
- Top Selling Products

```
In [57]: # Top-Selling Products
         top_products = df.groupby('SKU')['Amount'].sum().reset_index().sort_values(by='Amount', ascending=False).head(10)

         plt.figure(figsize=(12, 6))
         sns.barplot(data=top_products, x='Amount', y='SKU',color='lightblue' )
         plt.title('Top 10 Selling Products')
         plt.xlabel('Total Sales Amount')
         plt.ylabel('Product SKU')
         plt.tight_layout()
         plt.show()
```



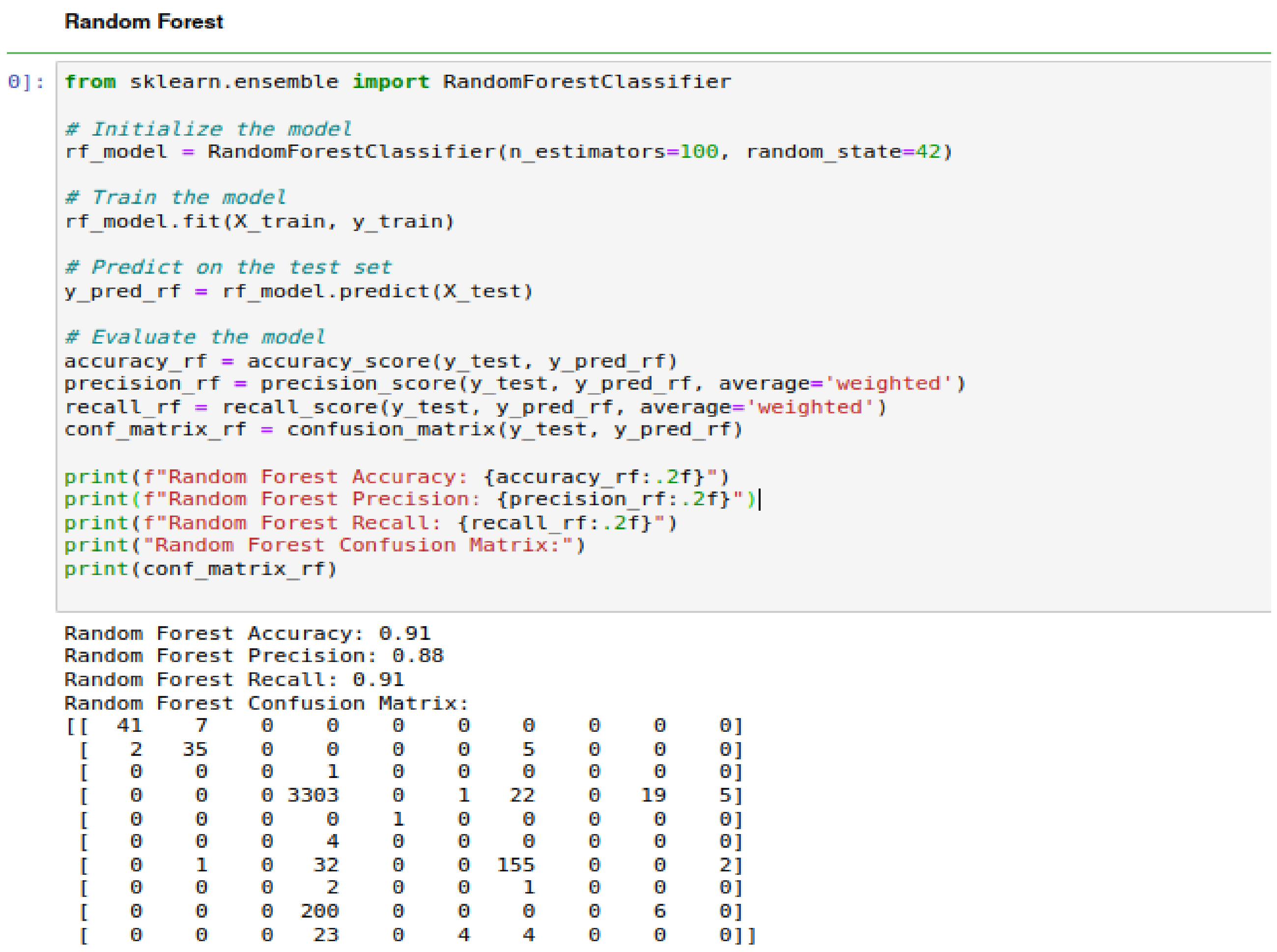
- Top 10 States by Sales

```
In [58]: top_states = df.groupby('ship-state')['Amount'].sum().reset_index().sort_values(by='Amount', ascending=False).head(10

         plt.figure(figsize=(12, 6))
         sns.barplot(data=top_states, x='Amount', y='ship-state', color='blue')
         plt.title('Top 10 States by Sales')
         plt.xlabel('Total Sales Amount')
         plt.ylabel('State')
         plt.tight_layout()
         plt.show()
```



4- Building Predictive Model

- The Accuracy of the Random Forest (91%)

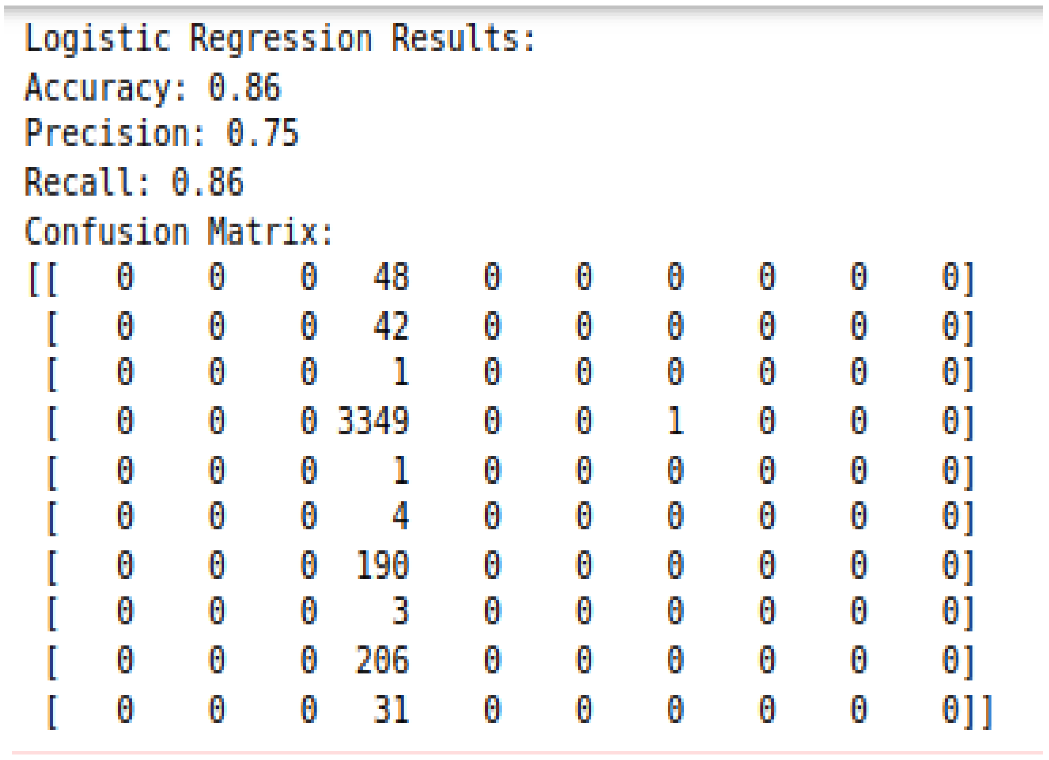


Random Forest

0]:

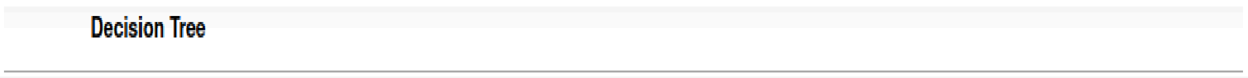
```
from sklearn.ensemble import RandomForestClassifier

# Initialize the model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the model
rf_model.fit(X_train, y_train)

# Predict on the test set
y_pred_rf = rf_model.predict(X_test)

# Evaluate the model
accuracy_rf = accuracy_score(y_test, y_pred_rf)
precision_rf = precision_score(y_test, y_pred_rf, average='weighted')
recall_rf = recall_score(y_test, y_pred_rf, average='weighted')
conf_matrix_rf = confusion_matrix(y_test, y_pred_rf)

print(f"Random Forest Accuracy: {accuracy_rf:.2f}")
print(f"Random Forest Precision: {precision_rf:.2f}")
print(f"Random Forest Recall: {recall_rf:.2f}")
print("Random Forest Confusion Matrix:")
print(conf_matrix_rf)
```

```
Random Forest Accuracy: 0.91
Random Forest Precision: 0.88
Random Forest Recall: 0.91
Random Forest Confusion Matrix:
[[  41    7    0    0    0    0    0    0    0    0]
 [   2   35    0    0    0    0    5    0    0    0]
 [   0    0    0    1    0    0    0    0    0    0]
 [   0    0    0 3303    0    1   22    0   19    5]
 [   0    0    0    0    1    0    0    0    0    0]
 [   0    0    0    4    0    0    0    0    0    0]
 [   0    1    0   32    0    0  155    0    0    2]
 [   0    0    0    2    0    0    1    0    0    0]
 [   0    0    0  200    0    0    0    0    6    0]
 [   0    0    0   23    0    4    4    0    0    0]]
```

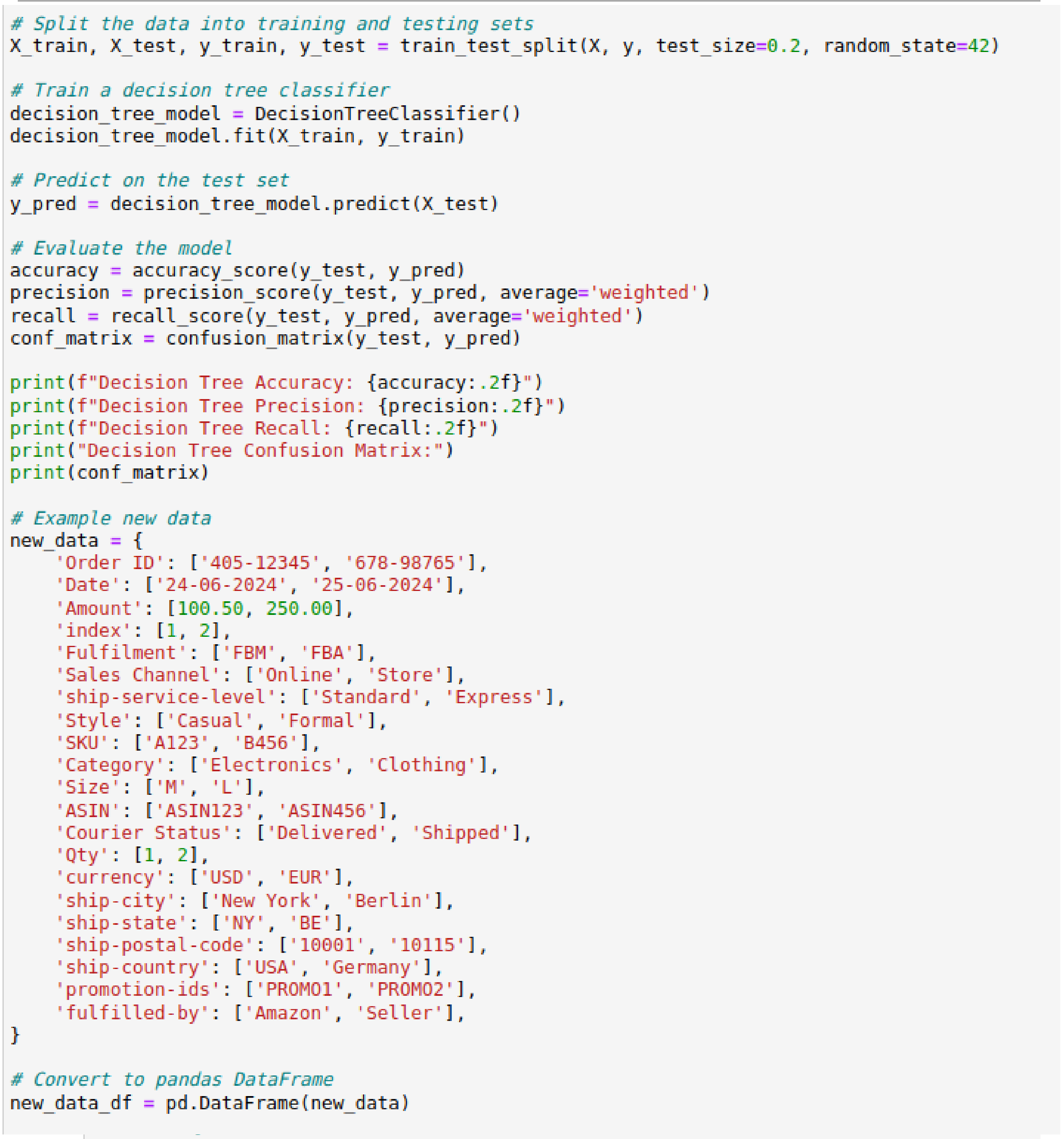


```
Logistic Regression Results:
Accuracy: 0.86
Precision: 0.75
Recall: 0.86
Confusion Matrix:
[[   0    0    0   48    0    0    0    0    0    0]
 [   0    0    0   42    0    0    0    0    0    0]
 [   0    0    0    1    0    0    0    0    0    0]
 [   0    0    0 3349    0    0    1    0    0    0]
 [   0    0    0    1    0    0    0    0    0    0]
 [   0    0    0    4    0    0    0    0    0    0]
 [   0    0    0  190    0    0    0    0    0    0]
 [   0    0    0    3    0    0    0    0    0    0]
 [   0    0    0  206    0    0    0    0    0    0]
 [   0    0    0   31    0    0    0    0    0    0]]
```

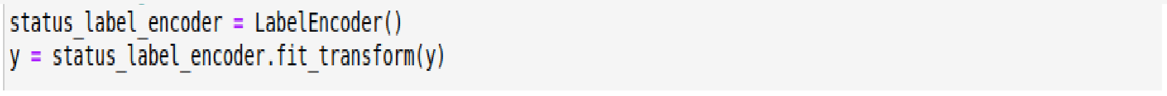
- The Accuracy of the Decision Tree is 96% is the highest accuracy
- Predict on new data



Decision Tree



```
# 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)

# Train a decision tree classifier
decision_tree_model = DecisionTreeClassifier()
decision_tree_model.fit(X_train, y_train)

# Predict on the test set
y_pred = decision_tree_model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
conf_matrix = confusion_matrix(y_test, y_pred)

print(f"Decision Tree Accuracy: {accuracy:.2f}")
print(f"Decision Tree Precision: {precision:.2f}")
print(f"Decision Tree Recall: {recall:.2f}")
print("Decision Tree Confusion Matrix:")
print(conf_matrix)

# Example new data
new_data = {
    'Order ID': ['405-12345', '678-98765'],
    'Date': ['24-06-2024', '25-06-2024'],
    'Amount': [100.50, 250.00],
    'index': [1, 2],
    'Fulfilment': ['FBM', 'FBA'],
    'Sales Channel': ['Online', 'Store'],
    'ship-service-level': ['Standard', 'Express'],
    'Style': ['Casual', 'Formal'],
    'SKU': ['A123', 'B456'],
    'Category': ['Electronics', 'Clothing'],
    'Size': ['M', 'L'],
    'ASIN': ['ASIN123', 'ASIN456'],
    'Courier Status': ['Delivered', 'Shipped'],
    'Qty': [1, 2],
    'currency': ['USD', 'EUR'],
    'ship-city': ['New York', 'Berlin'],
    'ship-state': ['NY', 'BE'],
    'ship-postal-code': ['10001', '10115'],
    'ship-country': ['USA', 'Germany'],
    'promotion-ids': ['PROMO1', 'PROMO2'],
    'fulfilled-by': ['Amazon', 'Seller'],
}

# Convert to pandas DataFrame
new_data_df = pd.DataFrame(new_data)
```



```
status_label_encoder = LabelEncoder()
y = status_label_encoder.fit_transform(y)
```

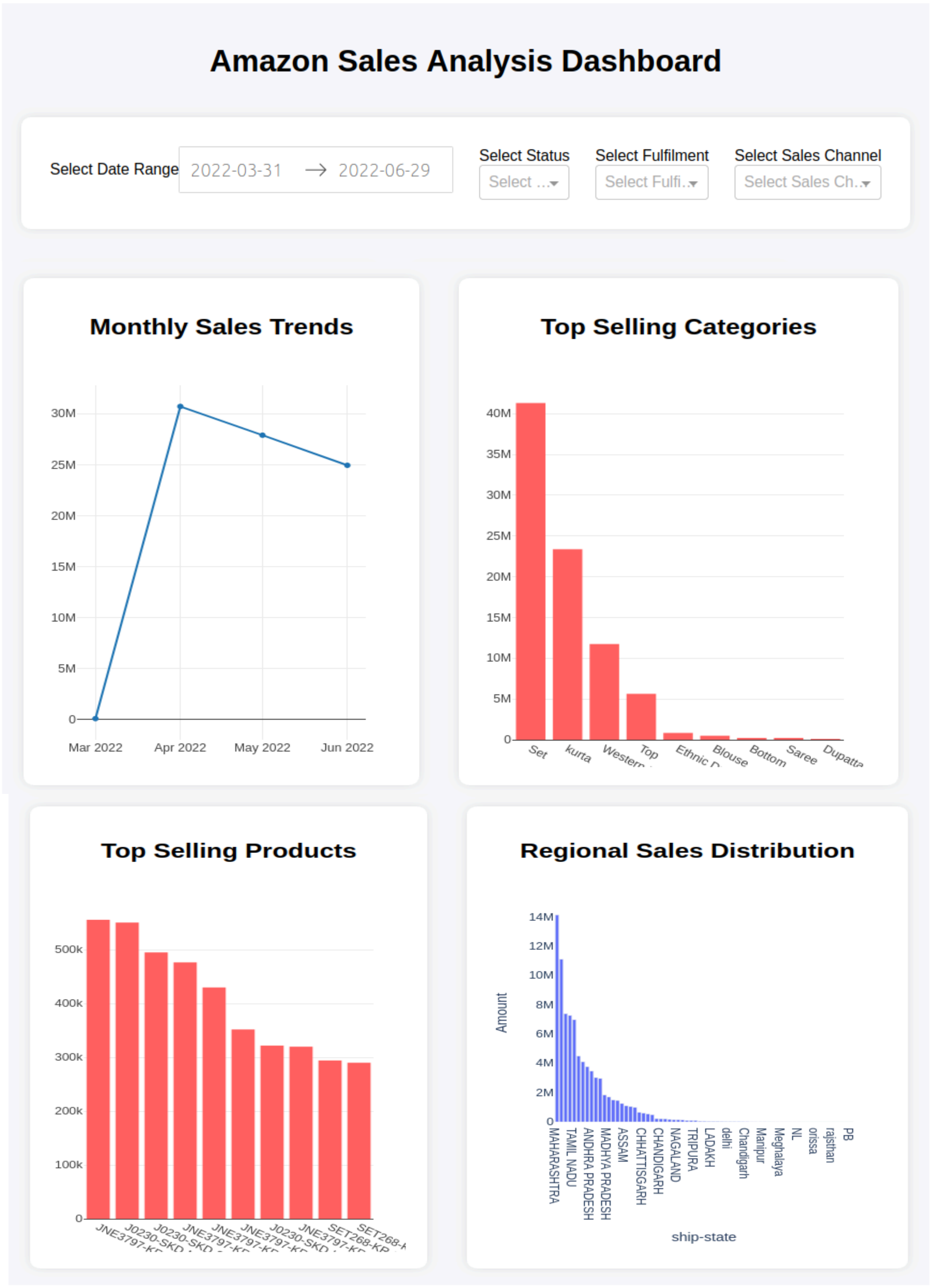
```python
# Convert to pandas DataFrame
new_data_df = pd.DataFrame(new_data)

# Preprocess new data
new_data_df['Date'] = pd.to_datetime(new_data_df['Date'], format='%d-%m-%Y')  # Adjust format
new_data_df['Year'] = new_data_df['Date'].dt.year
new_data_df['Month'] = new_data_df['Date'].dt.month
new_data_df['Day'] = new_data_df['Date'].dt.day
new_data_df['Amount'] = pd.to_numeric(new_data_df['Amount'], errors='coerce')

# Drop the 'Date' column
new_data_df.drop(columns=['Date'], inplace=True)

# Create a template DataFrame with the same columns as 'features'
template_df = pd.DataFrame(columns=features.columns)

# Append the new data to the template DataFrame
new_data_encoded = pd.concat([template_df, new_data_df], ignore_index=True)

# Fill any missing columns with zeros or appropriate default values
new_data_encoded = new_data_encoded.fillna(0)

# Encode new data with the same encoders used for training
def encode_new_data(new_data, label_encoders):
    for column in new_data.select_dtypes(include=['object']).columns:
        if column in label_encoders:
            le = label_encoders[column]
            # Handle unseen labels
            new_data[column] = new_data[column].apply(lambda x: le.transform([x])[0] if x in le.classes_ else -1)
    return new_data

new_data_encoded = encode_new_data(new_data_encoded, label_encoders)

# Ensure the new data has the same feature columns as the training data
new_data_encoded = new_data_encoded[X.columns]

# Make predictions
predictions = decision_tree_model.predict(new_data_encoded)

# Convert numerical predictions back to categorical labels
categorical_predictions = status_label_encoder.inverse_transform(predictions)

print(categorical_predictions)
```

```
Decision Tree Accuracy: 0.96
Decision Tree Precision: 0.97
Decision Tree Recall: 0.96
Decision Tree Confusion Matrix:
[[ 3579    15     0    14     0     1     0     0     0     0     2     0]
 [   17   111     3     2     0     0     0     0     0     0     0     0]
 [    0     0    58     0     0     0     0     0     0     0     0     0]
 [   14     2     0 15625     0     0     0     0     0     0     0     0]
 [    0     0     0     0     0     0     0     0     0     0     0     1]
 [    2     0     0     0     0  5261     0     7    14     0   434    22]
 [    0     0     0     0     0     2     0     0     0     0     0     0]
 [    0     0     0     0     0     4     0     0     1     0     0     0]
 [    0     0     0     0     0    13     0     0   163     1     1     5]
 [    0     0     0     0     0     2     0     0     0     0     0     0]
 [    0     0     0     0     0   338     0     1     1     0    47     1]
 [    0     0     0     0     0    23     0     1     5     0     2     0]]
```

```
/tmp/ipykernel_152134/1923130460.py:109: FutureWarning:

The behavior of DataFrame concatenation with empty or all-NA entries is depre
o longer exclude empty or all-NA columns when determining the result dtypes.
relevant entries before the concat operation.

/tmp/ipykernel_152134/1923130460.py:112: FutureWarning:

Downcasting object dtype arrays on .fillna, .ffill, .bfill is deprecated and
esult.infer_objects(copy=False) instead. To opt-in to the future behavior, se
casting', True)`
```

```
['Cancelled' 'Pending']
```

5-Dashboard



Amazon Sales Analysis Dashboard

Select Date Range 2022-03-31 → 2022-06-29

Select Status: Select ...

Select Fulfilment: Select Fulfi..

Select Sales Channel: Select Sales Ch..