Advanced Sales Data Analysis

November 2, 2024

0.1 Name Lakshman Chaudhary

0.2 Project Title: Advanced Sales Data Analysis

0.2.1 Import Libraries

[3]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

0.2.2 Load the Dataset

[4]:
```
# Load the data
file_path = 'ECOMM DATA.xlsx'
data = pd.read_excel(file_path)
```

[5]:
```
# Display the first few rows of the dataframe
print(orders_df.head())
```

```
   Row ID         Order ID Order Date  Ship Date     Ship Mode Customer ID  \
0   32298   CA-2012-124891 2012-07-31 2012-07-31      Same Day    RH-19495
1   26341    IN-2013-77878 2013-02-05 2013-02-07  Second Class    JR-16210
2   25330    IN-2013-71249 2013-10-17 2013-10-18   First Class    CR-12730
3   13524  ES-2013-1579342 2013-01-28 2013-01-30   First Class    KM-16375
4   47221     SG-2013-4320 2013-11-05 2013-11-06      Same Day     RH-9495

      Customer Name      Segment           City            State  …  \
0       Rick Hansen     Consumer  New York City         New York  …
1     Justin Ritter    Corporate     Wollongong  New South Wales  …
2      Craig Reiter     Consumer       Brisbane       Queensland  …
3  Katherine Murray  Home Office         Berlin           Berlin  …
4       Rick Hansen     Consumer          Dakar            Dakar  …

         Product ID    Category Sub-Category  \
0   TEC-AC-10003033  Technology  Accessories
1   FUR-CH-10003950   Furniture       Chairs
2   TEC-PH-10004664  Technology       Phones
3   TEC-PH-10004583  Technology       Phones
4  TEC-SHA-10000501  Technology      Copiers
```

```
                                        Product Name     Sales Quantity  \
0  Plantronics CS510 - Over-the-Head monaural Wir…  2309.650        7
1          Novimex Executive Leather Armchair, Black  3709.395        9
2                  Nokia Smart Phone, with Caller ID  5175.171        9
3                     Motorola Smart Phone, Cordless  2892.510        5
4                     Sharp Wireless Fax, High-Speed  2832.960        8

  Discount    Profit  Shipping Cost  Order Priority
0      0.0  762.1845         933.57        Critical
1      0.1 -288.7650         923.63        Critical
2      0.1  919.9710         915.49          Medium
3      0.1  -96.5400         910.16          Medium
4      0.0  311.5200         903.04        Critical

[5 rows x 24 columns]
```

0.2.3 Inspect the Data

[6]:
```
# Display the first few rows
data.head()
```

[6]:
```
   Row ID         Order ID Order Date  Ship Date     Ship Mode Customer ID  \
0   32298   CA-2012-124891 2012-07-31 2012-07-31      Same Day    RH-19495
1   26341    IN-2013-77878 2013-02-05 2013-02-07  Second Class    JR-16210
2   25330    IN-2013-71249 2013-10-17 2013-10-18   First Class    CR-12730
3   13524  ES-2013-1579342 2013-01-28 2013-01-30   First Class    KM-16375
4   47221     SG-2013-4320 2013-11-05 2013-11-06      Same Day     RH-9495

      Customer Name      Segment           City            State  …  \
0       Rick Hansen     Consumer  New York City         New York  …
1     Justin Ritter    Corporate     Wollongong  New South Wales  …
2      Craig Reiter     Consumer       Brisbane       Queensland  …
3  Katherine Murray  Home Office         Berlin           Berlin  …
4       Rick Hansen     Consumer          Dakar            Dakar  …

         Product ID    Category Sub-Category  \
0   TEC-AC-10003033  Technology  Accessories
1   FUR-CH-10003950   Furniture       Chairs
2   TEC-PH-10004664  Technology       Phones
3   TEC-PH-10004583  Technology       Phones
4  TEC-SHA-10000501  Technology      Copiers

                                        Product Name     Sales Quantity  \
0  Plantronics CS510 - Over-the-Head monaural Wir…  2309.650        7
1          Novimex Executive Leather Armchair, Black  3709.395        9
2                  Nokia Smart Phone, with Caller ID  5175.171        9
```

```
3                    Motorola Smart Phone, Cordless  2892.510        5
4                    Sharp Wireless Fax, High-Speed  2832.960        8

  Discount    Profit  Shipping Cost  Order Priority
0      0.0  762.1845         933.57        Critical
1      0.1 -288.7650         923.63        Critical
2      0.1  919.9710         915.49          Medium
3      0.1  -96.5400         910.16          Medium
4      0.0  311.5200         903.04        Critical

[5 rows x 24 columns]
```

[7]:
```
# Get data summary
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51290 entries, 0 to 51289
Data columns (total 24 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Row ID          51290 non-null  int64
 1   Order ID        51290 non-null  object
 2   Order Date      51290 non-null  datetime64[ns]
 3   Ship Date       51290 non-null  datetime64[ns]
 4   Ship Mode       51290 non-null  object
 5   Customer ID     51290 non-null  object
 6   Customer Name   51290 non-null  object
 7   Segment         51290 non-null  object
 8   City            51290 non-null  object
 9   State           51290 non-null  object
 10  Country         51290 non-null  object
 11  Postal Code     9994 non-null   float64
 12  Market          51290 non-null  object
 13  Region          51290 non-null  object
 14  Product ID      51290 non-null  object
 15  Category        51290 non-null  object
 16  Sub-Category    51290 non-null  object
 17  Product Name    51290 non-null  object
 18  Sales           51290 non-null  float64
 19  Quantity        51290 non-null  int64
 20  Discount        51290 non-null  float64
 21  Profit          51290 non-null  float64
 22  Shipping Cost   51290 non-null  float64
 23  Order Priority  51290 non-null  object
dtypes: datetime64[ns](2), float64(5), int64(2), object(15)
memory usage: 9.4+ MB
```

[8]:
```
# Check for any missing values
data.isnull().sum()
```

[8]:
```
Row ID               0
Order ID             0
Order Date           0
Ship Date            0
Ship Mode            0
Customer ID          0
Customer Name        0
Segment              0
City                 0
State                0
Country              0
Postal Code      41296
Market               0
Region               0
Product ID           0
Category             0
Sub-Category         0
Product Name         0
Sales                0
Quantity             0
Discount             0
Profit               0
Shipping Cost        0
Order Priority       0
dtype: int64
```

[2]:
```
# Load the Excel file
file_path = 'ECOMM DATA.xlsx'
orders_df = pd.read_excel(file_path, sheet_name='Orders')

# Display the first few rows of the dataframe
print(orders_df.head())
```

```
   Row ID         Order ID Order Date  Ship Date     Ship Mode Customer ID  \
0   32298   CA-2012-124891 2012-07-31 2012-07-31      Same Day    RH-19495
1   26341    IN-2013-77878 2013-02-05 2013-02-07  Second Class    JR-16210
2   25330    IN-2013-71249 2013-10-17 2013-10-18   First Class    CR-12730
3   13524  ES-2013-1579342 2013-01-28 2013-01-30   First Class    KM-16375
4   47221     SG-2013-4320 2013-11-05 2013-11-06      Same Day     RH-9495

      Customer Name      Segment           City            State  … \
0       Rick Hansen     Consumer  New York City         New York  …
1     Justin Ritter    Corporate     Wollongong  New South Wales  …
2      Craig Reiter     Consumer       Brisbane       Queensland  …
3  Katherine Murray  Home Office         Berlin           Berlin  …
```

```
4       Rick Hansen     Consumer          Dakar            Dakar  …

         Product ID    Category Sub-Category  \
0   TEC-AC-10003033  Technology  Accessories
1   FUR-CH-10003950   Furniture       Chairs
2   TEC-PH-10004664  Technology       Phones
3   TEC-PH-10004583  Technology       Phones
4  TEC-SHA-10000501  Technology      Copiers

                                        Product Name     Sales Quantity  \
0  Plantronics CS510 - Over-the-Head monaural Wir…  2309.650        7
1          Novimex Executive Leather Armchair, Black  3709.395        9
2                  Nokia Smart Phone, with Caller ID  5175.171        9
3                     Motorola Smart Phone, Cordless  2892.510        5
4                     Sharp Wireless Fax, High-Speed  2832.960        8

  Discount    Profit  Shipping Cost  Order Priority
0      0.0  762.1845         933.57        Critical
1      0.1 -288.7650         923.63        Critical
2      0.1  919.9710         915.49          Medium
3      0.1  -96.5400         910.16          Medium
4      0.0  311.5200         903.04        Critical

[5 rows x 24 columns]
```

[3]:
```python
# Calculate total sales
total_sales = orders_df['Sales'].sum()
print(f"Total Sales: ${total_sales:.2f}")
```

```
Total Sales: $12642501.91
```

0.2.4 Data Cleaning and Preparation

Handling Missing Values

[38]:
```python
# Fill or drop missing values based on your analysis
data.fillna(0, inplace=True)  # Example of filling missing values with 0
```

[39]:
```python
print(data.isnull().sum())  # Displays the count of missing values for each␣
↪column
```

```
Row ID           0
Order ID         0
Order Date       0
Ship Date        0
Ship Mode        0
Customer ID      0
Customer Name    0
Segment          0
City             0
```

```
State             0
Country           0
Postal Code       0
Market            0
Region            0
Product ID        0
Category          0
Sub-Category      0
Product Name      0
Sales             0
Quantity          0
Discount          0
Profit            0
Shipping Cost     0
Order Priority    0
Year              0
Month             0
Season            0
dtype: int64
```

Remove Duplicate Entries

[42]:
```
# Drop duplicates if any
data.drop_duplicates(inplace=True)
```

[43]:
```
print(f"Remaining duplicates: {data.duplicated().sum()}")  # Prints the number of remaining duplicates
```

```
Remaining duplicates: 0
```

[46]:
```
data.drop_duplicates(subset=['Order ID', 'Customer ID'], inplace=True)
```

[47]:
```
data.drop_duplicates(inplace=True)
```

[48]:
```
# Drop duplicates based on specific columns
data.drop_duplicates(subset=['Order ID', 'Customer ID'], inplace=True)

# Display the DataFrame to see the changes
print(data.head())  # Shows the first few rows of the DataFrame
```

```
   Row ID         Order ID Order Date  Ship Date     Ship Mode Customer ID  \
0   32298   CA-2012-124891 2012-07-31 2012-07-31      Same Day    RH-19495
1   26341    IN-2013-77878 2013-02-05 2013-02-07  Second Class    JR-16210
2   25330    IN-2013-71249 2013-10-17 2013-10-18   First Class    CR-12730
3   13524  ES-2013-1579342 2013-01-28 2013-01-30   First Class    KM-16375
4   47221     SG-2013-4320 2013-11-05 2013-11-06      Same Day     RH-9495

      Customer Name      Segment           City            State  …  \
0       Rick Hansen     Consumer  New York City         New York  …
```

```
1     Justin Ritter    Corporate     Wollongong  New South Wales  …
2      Craig Reiter     Consumer       Brisbane       Queensland  …
3  Katherine Murray  Home Office         Berlin           Berlin  …
4       Rick Hansen     Consumer          Dakar            Dakar  …

                                      Product Name     Sales Quantity  \
0  Plantronics CS510 - Over-the-Head monaural Wir…  2309.650        7
1          Novimex Executive Leather Armchair, Black  3709.395        9
2                  Nokia Smart Phone, with Caller ID  5175.171        9
3                     Motorola Smart Phone, Cordless  2892.510        5
4                     Sharp Wireless Fax, High-Speed  2832.960        8

  Discount    Profit Shipping Cost Order Priority  Year  Month  Season
0      0.0  762.1845        933.57       Critical  2012      7  Summer
1      0.1 -288.7650        923.63       Critical  2013      2  Winter
2      0.1  919.9710        915.49         Medium  2013     10    Fall
3      0.1  -96.5400        910.16         Medium  2013      1  Winter
4      0.0  311.5200        903.04       Critical  2013     11    Fall

[5 rows x 27 columns]
```

[49]:
```python
# Count duplicates before dropping
initial_count = data.duplicated(subset=['Order ID', 'Customer ID']).sum()
print(f"Initial duplicate count: {initial_count}")

# Drop duplicates
data.drop_duplicates(subset=['Order ID', 'Customer ID'], inplace=True)

# Count duplicates after dropping
final_count = data.duplicated(subset=['Order ID', 'Customer ID']).sum()
print(f"Final duplicate count: {final_count}")
```

```
Initial duplicate count: 0
Final duplicate count: 0
```

[50]:
```python
print(data.shape)  # Shows the number of rows and columns in the DataFrame
```

```
(25753, 27)
```

[51]:
```python
print(data.columns)  # Displays the column names in the DataFrame
```

```
Index(['Row ID', 'Order ID', 'Order Date', 'Ship Date', 'Ship Mode',
       'Customer ID', 'Customer Name', 'Segment', 'City', 'State', 'Country',
       'Postal Code', 'Market', 'Region', 'Product ID', 'Category',
       'Sub-Category', 'Product Name', 'Sales', 'Quantity', 'Discount',
       'Profit', 'Shipping Cost', 'Order Priority', 'Year', 'Month', 'Season'],
      dtype='object')
```

Data Type Conversion

[14]:
```
# Convert 'Order Date' to datetime
data['Order Date'] = pd.to_datetime(data['Order Date'], errors='coerce')

# Verify conversion
print(data['Order Date'].head())
```

```
0   2012-07-31
1   2013-02-05
2   2013-10-17
3   2013-01-28
4   2013-11-05
Name: Order Date, dtype: datetime64[ns]
```

0.2.5 Exploratory Data Analysis (EDA)

Summary Statistics

[15]:
```
data.describe()
```

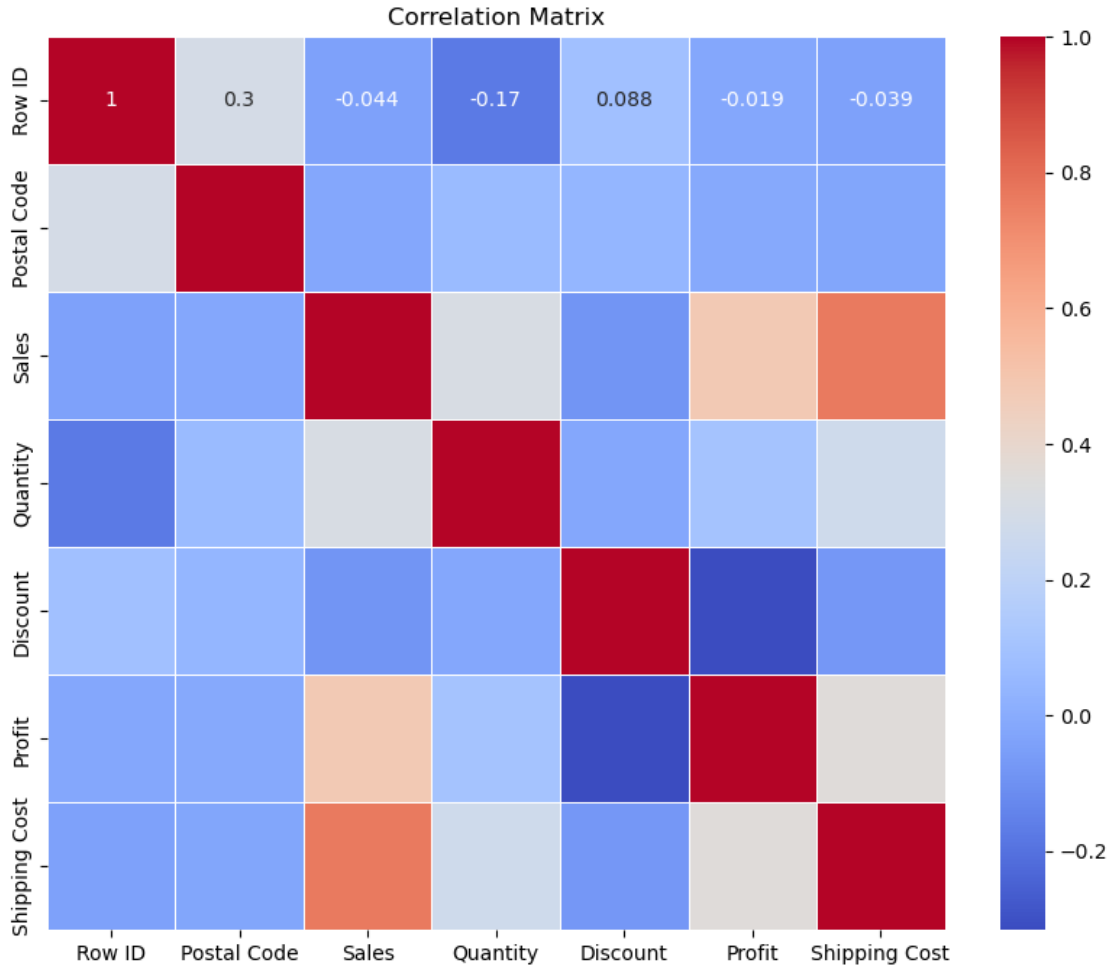
[15]:

	Row ID	Order Date	Ship Date	Postal Code	Sales	Quantity	Discount	Profit	Shipping Cost
count	51290.00000	51290	51290	51290.000000	51290.000000	51290.000000	51290.000000	51290.000000	51290.000000
mean	25645.50000	2013-05-11 21:26:49.155781120	2013-05-15 20:42:42.745174528	10753.999844	246.490581	3.476545	0.142908	28.610982	26.375818
min	1.00000	2011-01-01 00:00:00	2011-01-03 00:00:00	0.000000	0.444000	1.000000	0.000000	-6599.978000	0.002000
25%	12823.25000	2012-06-19 00:00:00	2012-06-23 00:00:00	0.000000	30.758625	2.000000	0.000000	0.000000	2.610000
50%	25645.50000	2013-07-08 00:00:00	2013-07-12 00:00:00	0.000000	85.053000	3.000000	0.000000	9.240000	7.790000
75%	38467.75000	2014-05-22 00:00:00	2014-05-26 00:00:00	0.000000	251.053200	5.000000	0.200000	36.810000	24.450000
max	51290.00000	2014-12-31 00:00:00	2015-01-07 00:00:00	99301.000000	22638.480000	14.000000	0.850000	8399.976000	933.570000
std	14806.29199	NaN	NaN	26042.011167	487.565361	2.278766	0.212280	174.340972	57.296810

Correlation Analysis

[17]:
```
# Filter out only numeric columns
numeric_data = data.select_dtypes(include=[np.number])

# Plot correlation heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(numeric_data.corr(), annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Matrix')
plt.show()
```

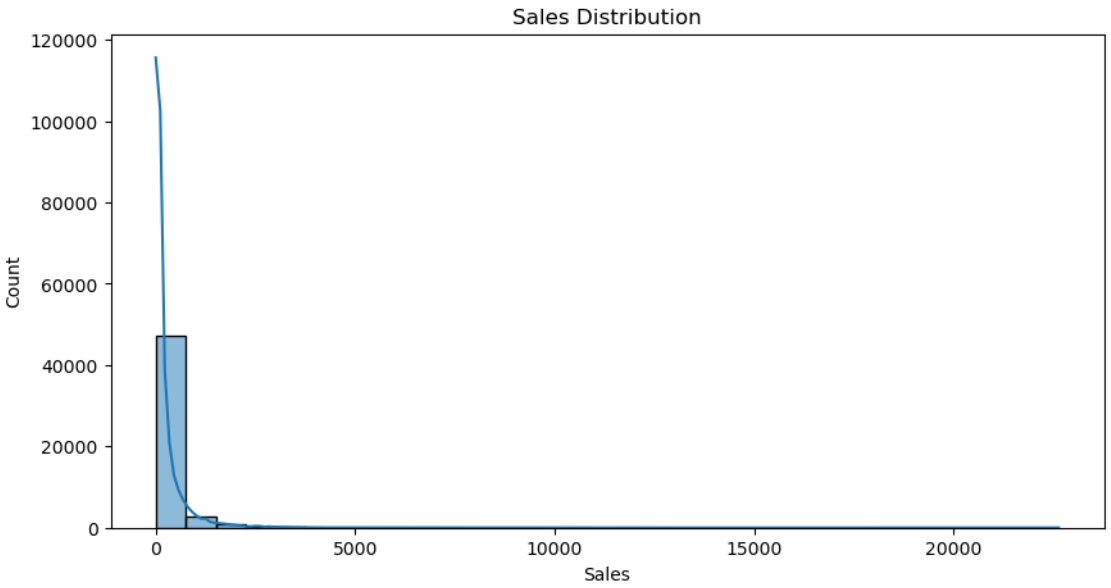


Distribution of Sales

[18]:
```
plt.figure(figsize=(10, 5))
sns.histplot(data['Sales'], kde=True, bins=30)
plt.title('Sales Distribution')
plt.show()
```

```
C:\Users\laksh\anaconda3\Lib\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):
```



0.2.6 Key Sales Insights

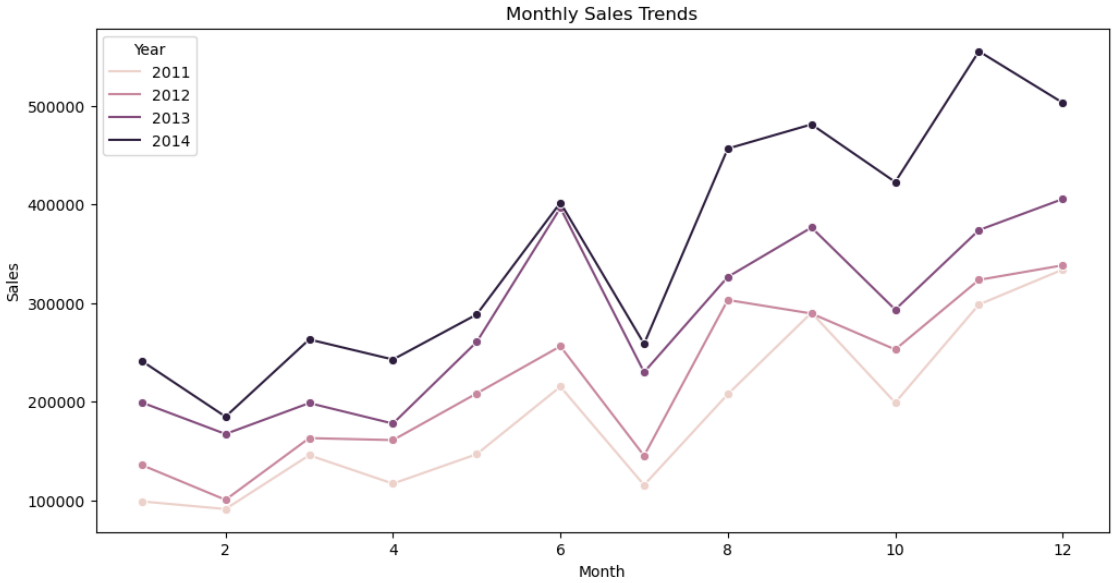
Monthly Sales Trends

[20]:
```
# Extract month and year from 'Order Date'
data['Year'] = data['Order Date'].dt.year
data['Month'] = data['Order Date'].dt.month

# Group by year and month for sales analysis
monthly_sales = data.groupby(['Year', 'Month'])['Sales'].sum().reset_index()

# Plot monthly sales trends
plt.figure(figsize=(12, 6))
sns.lineplot(x='Month', y='Sales', hue='Year', data=monthly_sales, marker='o')
plt.title('Monthly Sales Trends')
plt.show()
```

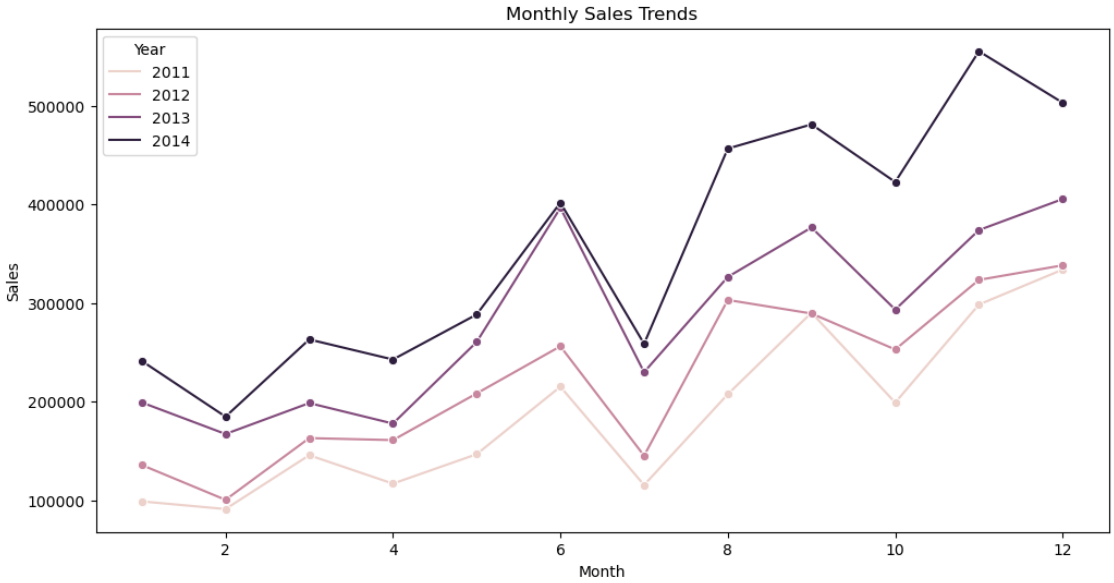
```
C:\Users\laksh\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\laksh\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: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



[23]:
```
# Replace infinite values with NaN if any exist in monthly_sales
monthly_sales.replace([np.inf, -np.inf], np.nan, inplace=True)

# Now plot
plt.figure(figsize=(12, 6))
sns.lineplot(x='Month', y='Sales', hue='Year', data=monthly_sales, marker='o')
plt.title('Monthly Sales Trends')
plt.show()
```

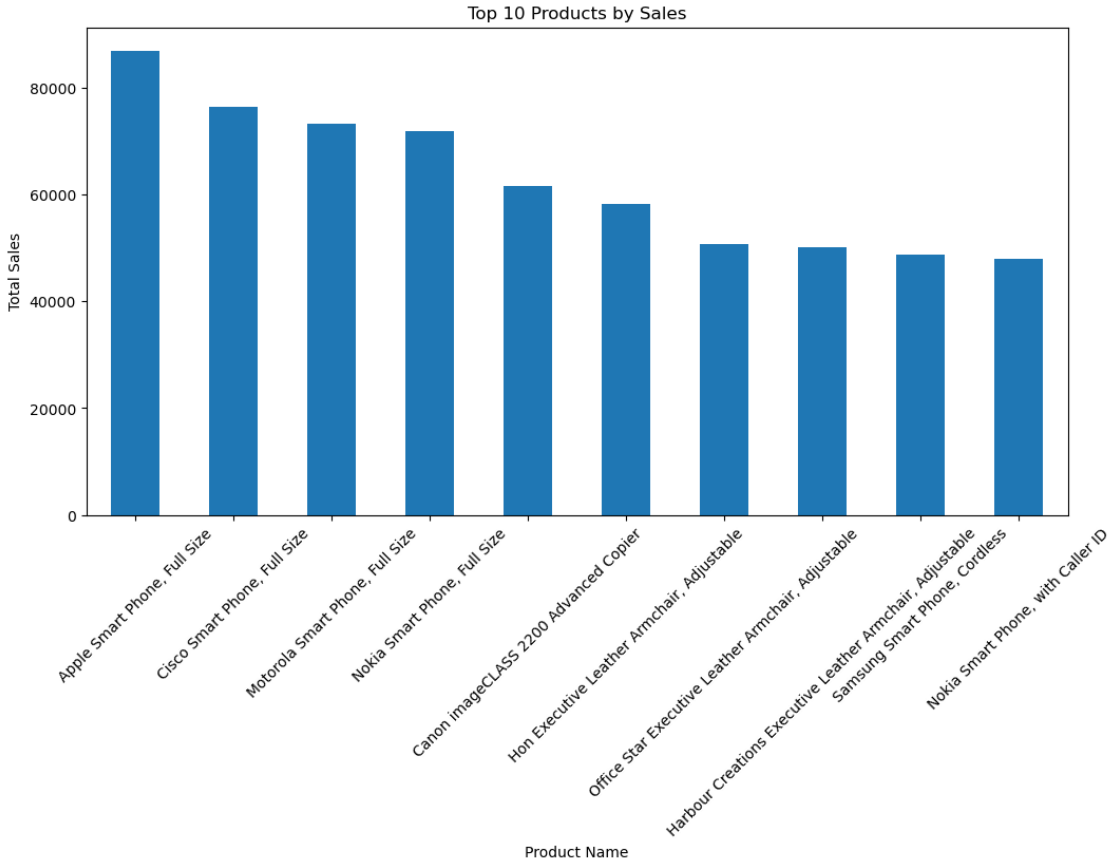
```
C:\Users\laksh\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\laksh\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: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



Top Products by Sales

[25]:
```
# Aggregate sales by product
top_products = data.groupby('Product Name')['Sales'].sum().nlargest(10)

# Plot top products
plt.figure(figsize=(12, 6))
top_products.plot(kind='bar')
plt.title('Top 10 Products by Sales')
plt.ylabel('Total Sales')
plt.xlabel('Product Name')
plt.xticks(rotation=45)
plt.show()
```



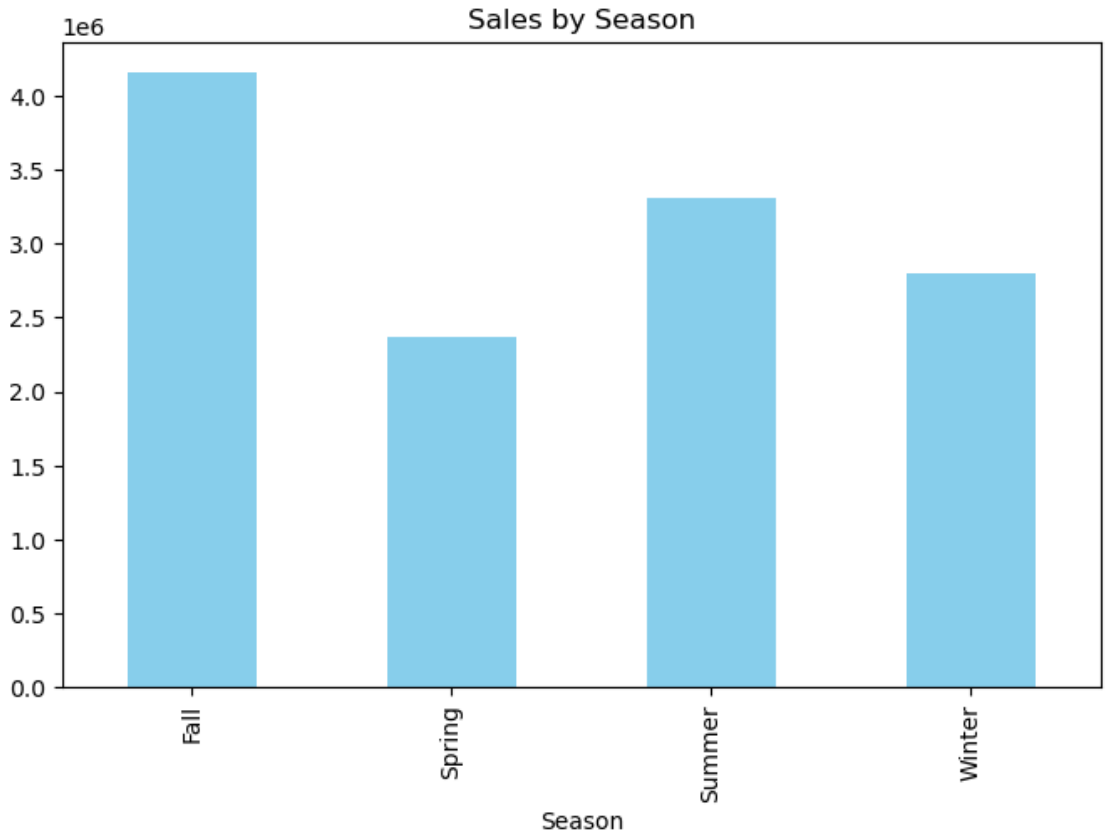
0.2.7 Advanced Insights

Seasonal Analysis

[26]:
```
# Extract season (this is an example and may vary based on data specifics)
data['Season'] = data['Month'].apply(lambda x: 'Winter' if x in [12, 1, 2]
                                     else 'Spring' if x in [3, 4, 5]
                                     else 'Summer' if x in [6, 7, 8]
                                     else 'Fall')

seasonal_sales = data.groupby('Season')['Sales'].sum()

# Plot seasonal sales
plt.figure(figsize=(8, 5))
seasonal_sales.plot(kind='bar', color='skyblue')
plt.title('Sales by Season')
plt.show()
```



Customer Segmentation Analysis

[28]:
```
# Example: Average order value by customer
customer_data = data.groupby('Customer ID').agg({'Sales': 'sum', 'Order ID':␣
 ↪'count'}).rename(columns={'Order ID': 'Order_Count'})
customer_data['Avg_Order_Value'] = customer_data['Sales'] /␣
 ↪customer_data['Order_Count']

# Display top customers by order value
top_customers = customer_data.sort_values(by='Avg_Order_Value',␣
 ↪ascending=False).head(10)
print(top_customers)
```

```
                Sales  Order_Count  Avg_Order_Value
Customer ID
BW-1065       9027.48000           10       902.748000
SM-20320     31125.29496           39       798.084486
DJ-3510       5976.69000            9       664.076667
MG-8145       5229.11400            8       653.639250
HL-15040     29664.23058           47       631.153842
HM-4980       4409.29800            7       629.899714
```

```
TC-20980     34218.26900          59       579.970661
MD-7860       6853.51200          12       571.126000
AC-10450     16126.39040          29       556.082428
TA-21385     35668.12080          65       548.740320
```

0.2.8 Save Analysis Results

[29]:
```
# Export the modified DataFrame to Excel for further reporting if needed
data.to_excel('Processed_ECOMM_DATA.xlsx', index=False)
```

[32]:
```
data.to_excel('Processed_ECOMM_DATA.xlsx', index=False)
```

[33]:
```
import os

# Check the current working directory
print(os.getcwd())
```

```
C:\Users\laksh\Downloads\Advanced Sales Data Analysis
```

[36]:
```
data.to_excel(r'C:\Users\laksh\Downloads\Advanced Sales Data␣
↪Analysis\Processed_ECOMM_DATA.xlsx', index=False)
```

[37]:
```
print(data.head())
```

```
   Row ID         Order ID Order Date  Ship Date     Ship Mode Customer ID  \
0   32298   CA-2012-124891 2012-07-31 2012-07-31      Same Day    RH-19495
1   26341    IN-2013-77878 2013-02-05 2013-02-07  Second Class    JR-16210
2   25330    IN-2013-71249 2013-10-17 2013-10-18   First Class    CR-12730
3   13524  ES-2013-1579342 2013-01-28 2013-01-30   First Class    KM-16375
4   47221     SG-2013-4320 2013-11-05 2013-11-06      Same Day     RH-9495

      Customer Name      Segment           City            State  …  \
0       Rick Hansen     Consumer  New York City         New York  …
1     Justin Ritter    Corporate     Wollongong  New South Wales  …
2      Craig Reiter     Consumer       Brisbane       Queensland  …
3  Katherine Murray  Home Office         Berlin           Berlin  …
4       Rick Hansen     Consumer          Dakar            Dakar  …

                                      Product Name     Sales Quantity  \
0  Plantronics CS510 - Over-the-Head monaural Wir…  2309.650        7
1          Novimex Executive Leather Armchair, Black  3709.395        9
2                  Nokia Smart Phone, with Caller ID  5175.171        9
3                     Motorola Smart Phone, Cordless  2892.510        5
4                     Sharp Wireless Fax, High-Speed  2832.960        8

  Discount    Profit Shipping Cost Order Priority  Year  Month  Season
0      0.0  762.1845        933.57       Critical  2012      7  Summer
1      0.1 -288.7650        923.63       Critical  2013      2  Winter
2      0.1  919.9710        915.49         Medium  2013     10    Fall
```

```
3      0.1  -96.5400        910.16         Medium  2013      1  Winter
4      0.0  311.5200        903.04       Critical  2013     11    Fall

[5 rows x 27 columns]
```

[30]:
```
import os
print(os.getcwd())
```

```
C:\Users\laksh\Downloads\Advanced Sales Data Analysis
```

[31]:
```
import pandas as pd

# Load the exported Excel file
processed_data = pd.read_excel('Processed_ECOMM_DATA.xlsx')

# Display the first few rows
print(processed_data.head())
```

```
   Row ID         Order ID Order Date  Ship Date     Ship Mode Customer ID  \
0   32298   CA-2012-124891 2012-07-31 2012-07-31      Same Day    RH-19495
1   26341    IN-2013-77878 2013-02-05 2013-02-07  Second Class    JR-16210
2   25330    IN-2013-71249 2013-10-17 2013-10-18   First Class    CR-12730
3   13524  ES-2013-1579342 2013-01-28 2013-01-30   First Class    KM-16375
4   47221     SG-2013-4320 2013-11-05 2013-11-06      Same Day     RH-9495

      Customer Name      Segment           City            State  …  \
0       Rick Hansen     Consumer  New York City         New York  …
1     Justin Ritter    Corporate     Wollongong  New South Wales  …
2      Craig Reiter     Consumer       Brisbane       Queensland  …
3  Katherine Murray  Home Office         Berlin           Berlin  …
4       Rick Hansen     Consumer          Dakar            Dakar  …

                                      Product Name     Sales Quantity  \
0  Plantronics CS510 - Over-the-Head monaural Wir…  2309.650        7
1          Novimex Executive Leather Armchair, Black  3709.395        9
2                  Nokia Smart Phone, with Caller ID  5175.171        9
3                     Motorola Smart Phone, Cordless  2892.510        5
4                     Sharp Wireless Fax, High-Speed  2832.960        8

  Discount    Profit Shipping Cost Order Priority  Year  Month  Season
0      0.0  762.1845        933.57       Critical  2012      7  Summer
1      0.1 -288.7650        923.63       Critical  2013      2  Winter
2      0.1  919.9710        915.49         Medium  2013     10    Fall
3      0.1  -96.5400        910.16         Medium  2013      1  Winter
4      0.0  311.5200        903.04       Critical  2013     11    Fall

[5 rows x 27 columns]
```

[ ]:

[53]:
```
# Convert 'Order Date' to datetime format
orders_df['Order Date'] = pd.to_datetime(orders_df['Order Date'])

# Extract year and month from the 'Order Date'
orders_df['Year'] = orders_df['Order Date'].dt.year
orders_df['Month'] = orders_df['Order Date'].dt.month

# Group by year and month and sum the sales
sales_trends = orders_df.groupby(['Year', 'Month'])['Sales'].sum().reset_index()

# Display the sales trends
print(sales_trends)
```

```
    Year  Month         Sales
0   2011      1   98898.48886
1   2011      2   91152.15698
2   2011      3  145729.36736
3   2011      4  116915.76418
4   2011      5  146747.83610
5   2011      6  215207.38022
6   2011      7  115510.41912
7   2011      8  207581.49122
8   2011      9  290214.45534
9   2011     10  199071.26404
10  2011     11  298496.53752
11  2011     12  333925.73460
12  2012      1  135780.72024
13  2012      2  100510.21698
14  2012      3  163076.77116
15  2012      4  161052.26952
16  2012      5  208364.89124
17  2012      6  256175.69842
18  2012      7  145236.78512
19  2012      8  303142.94238
20  2012      9  289389.16564
21  2012     10  252939.85020
22  2012     11  323512.41690
23  2012     12  338256.96660
24  2013      1  199185.90738
25  2013      2  167239.65040
26  2013      3  198594.03012
27  2013      4  177821.31684
28  2013      5  260498.56470
29  2013      6  396519.61190
30  2013      7  229928.95200
31  2013      8  326488.78936
```

```
32  2013      9  376619.24568
33  2013     10  293406.64288
34  2013     11  373989.36010
35  2013     12  405454.37802
36  2014      1  241268.55566
37  2014      2  184837.35556
38  2014      3  263100.77262
39  2014      4  242771.86130
40  2014      5  288401.04614
41  2014      6  401814.06310
42  2014      7  258705.68048
43  2014      8  456619.94236
44  2014      9  481157.24370
45  2014     10  422766.62916
46  2014     11  555279.02700
47  2014     12  503143.69348
```

[54]:
```
# Convert 'Order Date' to datetime format
orders_df['Order Date'] = pd.to_datetime(orders_df['Order Date'])

# Extract year and month from the 'Order Date'
orders_df['Year'] = orders_df['Order Date'].dt.year
orders_df['Month'] = orders_df['Order Date'].dt.month

# Group by year and month and sum the sales
sales_trends = orders_df.groupby(['Year', 'Month'])['Sales'].sum().reset_index()

# Display the sales trends
sales_trends.head()
```

[54]:
```
   Year  Month         Sales
0  2011      1   98898.48886
1  2011      2   91152.15698
2  2011      3  145729.36736
3  2011      4  116915.76418
4  2011      5  146747.83610
```

[55]:
```
# Group by product and sum the sales
best_selling_products = orders_df.groupby('Product Name')['Sales'].sum().
 ↪reset_index()

# Sort the products by sales in descending order
best_selling_products = best_selling_products.sort_values(by='Sales',␣
 ↪ascending=False)

# Display the top 10 best-selling products
best_selling_products.head(10)
```

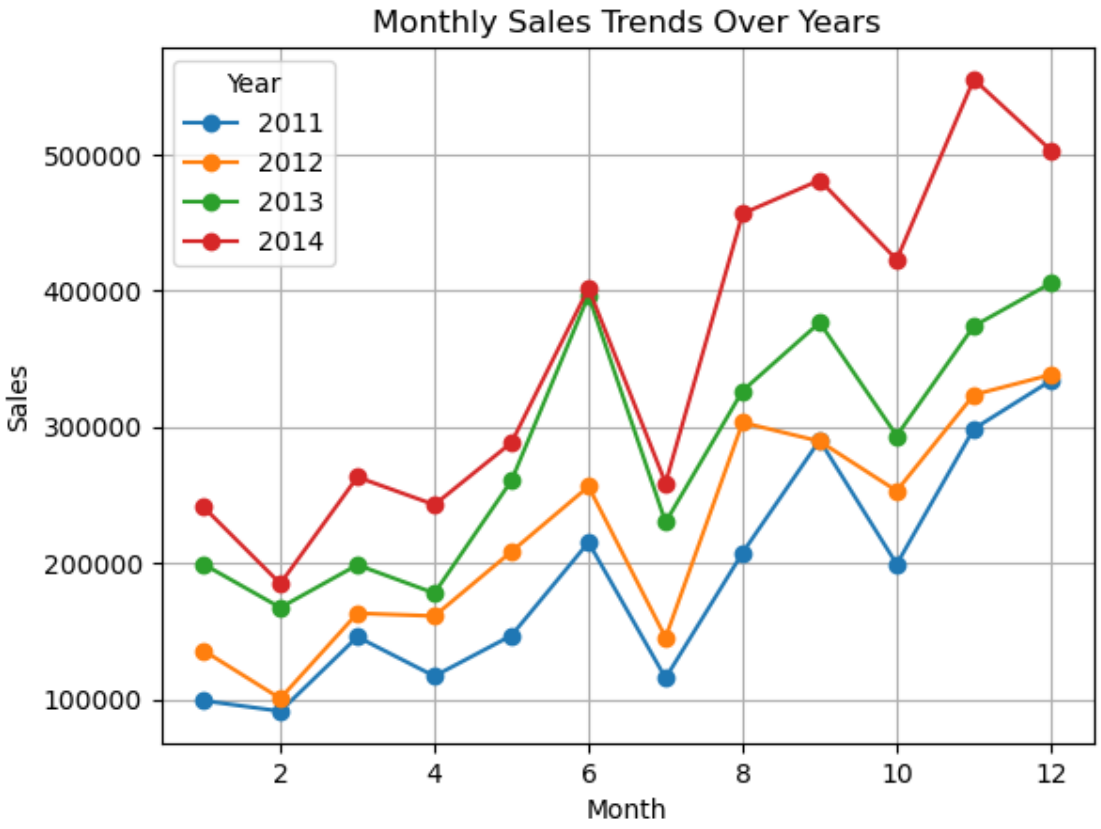
[55]:

	Product Name	Sales
310	Apple Smart Phone, Full Size	86935.7786
970	Cisco Smart Phone, Full Size	76441.5306
2415	Motorola Smart Phone, Full Size	73156.3030
2501	Nokia Smart Phone, Full Size	71904.5555
866	Canon imageCLASS 2200 Advanced Copier	61599.8240
1837	Hon Executive Leather Armchair, Adjustable	58193.4841
2631	Office Star Executive Leather Armchair, Adjust…	50661.6840
1714	Harbour Creations Executive Leather Armchair, …	50121.5160
2988	Samsung Smart Phone, Cordless	48653.4600
2502	Nokia Smart Phone, with Caller ID	47877.7857

[56]:
```
import matplotlib.pyplot as plt

# Create a pivot table for easier plotting
sales_trends_pivot = sales_trends.pivot(index='Month', columns='Year',␣
 ↪values='Sales')

# Plot the sales trends
sales_trends_pivot.plot(kind='line', marker='o')
plt.title('Monthly Sales Trends Over Years')
plt.xlabel('Month')
plt.ylabel('Sales')
plt.legend(title='Year')
plt.grid(True)
plt.show()
```


Monthly Sales Trends Over Years
Year
2011
2012
2013
2014
Sales
500000
400000
300000
200000
100000
2
4
6
8
10
12
Month

[ ]: