**# SuperStore Sales Analysis**

This analysis aims to explore the sales patterns, product performance, and regional trends using the SuperStoreOrders dataset. The objective is to clean the data, visualize key metrics, and derive actionable insights from the data.

#lets import the required libraries first

import pandas as pd

# Loading the dataset

df = pd.read\_csv('SuperStoreOrders.csv')

# Displaying the first few rows of the dataset

print(df.head(5))

output be like :

order\_id order\_date ship\_date ship\_mode customer\_name segment ... quantity discount profit shipping\_cost order\_priority year

0 AG-2011-2040 1/1/2011 6/1/2011 Standard Class Toby Braunhardt Consumer ... 2 0.0 106.140 35.46 Medium 2011

1 IN-2011-47883 1/1/2011 8/1/2011 Standard Class Joseph Holt Consumer ... 3 0.1 36.036 9.72 Medium 2011

2 HU-2011-1220 1/1/2011 5/1/2011 Second Class Annie Thurman Consumer ... 4 0.0 29.640 8.17 High 2011

3 IT-2011-3647632 1/1/2011 5/1/2011 Second Class Eugene Moren Home Office ... 3 0.5 -26.055 4.82 High 2011

4 IN-2011-47883 1/1/2011 8/1/2011 Standard Class Joseph Holt Consumer ... 5 0.1 37.770 4.70 Medium 2011

# Displaying data types of each column so we can make required changes for the datatypes

print(df.dtypes)

output:

order\_id object

order\_date object

ship\_date object

ship\_mode object

customer\_name object

segment object

state object

country object

market object

region object

product\_id object

category object

sub\_category object

product\_name object

sales object

quantity int64

discount float64

profit float64

shipping\_cost float64

order\_priority object

year int64

#the order\_date, ship\_date, and sales columns need attention. Let's go through the necessary steps to clean and prepare your dataset for analysis.

#Since order\_date is currently an object type, convert it to datetime format. The same applies to the ship\_date.

# Convert 'order\_date' and 'ship\_date' to datetime

df['order\_date'] = pd.to\_datetime(df['order\_date'], format='%m/%d/%Y', errors='coerce')

df['ship\_date'] = pd.to\_datetime(df['ship\_date'], format='%m/%d/%Y', errors='coerce')

# Check for any conversion errors

print(df['order\_date'].isnull().sum())  # Number of NaT values in 'Order Date'

print(df['ship\_date'].isnull().sum())   # Number of NaT values in 'Ship Date'

print(df.dtypes)

#we saw the order date and shipdate were now hanged to datetime and now lets make some more required changes

#sales columnn in also in object fprmat so we need to change that to numeric type

df['sales'] = pd.to\_numeric(df['sales'], errors='coerce')

print(df.dtypes)

# nowthat we changed the object to numeric so kets check for any conversional errors

print(df['sales'].isnull().sum())  # Number of NaN values in 'Sales'

#we found 2630 errors here when converting the data type so we should fix them now

# Check unique values in the 'Sales' column to identify potential issues

print(df['sales'].unique())

# Convert 'Sales' to numeric, coercing errors to NaN

df['sales'] = pd.to\_numeric(df['sales'], errors='coerce')

# Check for any NaN values in 'Sales' after conversion

print("NaN values in 'sales':", df['sales'].isnull().sum())

# Display rows where Sales is NaN

problematic\_sales = df[df['sales'].isnull()]

print("Problematic\_sales\_Entries:")

print(problematic\_sales[['sales']])  # Display only the 'Sales' column

# Convert 'Sales' to string first (this handles NaN as well)

df['sales'] = df['sales'].astype(str)

# Now you can safely strip whitespace

df['sales'] = df['sales'].str.strip()

# Remove currency symbols and commas

df['sales'] = df['sales'].replace({'\$': '', ',': ''}, regex=True)

df['sales'] = pd.to\_numeric(df['sales'], errors='coerce')

print("NaN values in 'sales' after conversion:", df['sales'].isnull().sum())

# Load the dataset again if needed

df = pd.read\_csv('SuperStoreOrders.csv')

# Convert 'Sales' to string and strip whitespace

df['sales'] = df['sales'].astype(str).str.strip()

# Check unique values that couldn't be converted to numeric

problematic\_sales = df[pd.to\_numeric(df['sales'], errors='coerce').isnull()]

print("Problematic\_Sale\_Entries:")

print(problematic\_sales[['sales']])  # Display problematic entries

# Remove any unwanted characters (for example, letters)

df['sales'] = df['sales'].replace({'[^0-9.]': ''}, regex=True)

df['sales'] = df['sales'].replace({'\$': '', ',': ''}, regex=True)

df['sales'] = pd.to\_numeric(df['sales'], errors='coerce')

# Check for NaN values after conversion

print("NaN values in 'sales' after conversion:", df['sales'].isnull().sum())

print(df.describe())

print(df.dtypes)  # Check the data types of all columns

# Convert 'Order Date' to datetime

df['order\_date'] = pd.to\_datetime(df['order\_date'], errors='coerce')

print(df['order\_date'].isnull().sum())  # Check for NaT (not-a-time) values

print(df.dtypes)  # Check the data types again

# Check the data types

print(df.dtypes)  # This will show the data types of all columns

# Convert 'Sales' to numeric, coercing errors to NaN

df['sales'] = pd.to\_numeric(df['sales'].replace({'\$': '', ',': ''}, regex=True), errors='coerce')

# Check for any NaN values in 'Sales'

print("NaN values in 'sales':", df['sales'].isnull().sum())

import pandas as pd

# Load the dataset

df = pd.read\_csv('SuperStoreOrders.csv')

# Display the first few rows and check column names

print(df.head())

print(df.columns)  # Print column names

print(df.dtypes)   # Print data types

print("NaN values in 'sales':", df['sales'].isnull().sum())

# Check if 'Order Date' is of datetime type

if not pd.api.types.is\_datetime64\_any\_dtype(df['order\_date']):

    df['order\_date'] = pd.to\_datetime(df['order\_date'], errors='coerce')

# Now group sales by month and sum them

sales\_over\_time = df.groupby(df['order\_date'].dt.to\_period('M'))['sales'].sum()

import pandas as pd

# Load the dataset

df = pd.read\_csv('SuperStoreOrders.csv')

# Display the first few rows and check column names and data types

print(df.head())

print(df.columns)  # Print column names

print(df.dtypes)   # Print data types

# Convert 'Order Date' to datetime

df['order\_date'] = pd.to\_datetime(df['order\_date'], errors='coerce')

# Check for NaT (not-a-time) values in 'Order Date'

print("NaT values in 'order\_date':", df['order\_date'].isnull().sum())

# Clean 'Sales' column and convert to numeric

df['sales'] = pd.to\_numeric(df['sales'].replace({'\$': '', ',': ''}, regex=True), errors='coerce')

# Check for any NaN values in 'Sales' after conversion

print("NaN values in 'sales':", df['sales'].isnull().sum())

# Check if 'Sales' is of numeric type

print("Data type of 'sales':", df['sales'].dtype)

# Group by product\_name and sum sales

top\_products = df.groupby('product\_name')['sales'].sum().nlargest(10)

# Group by product\_name and sum only numeric columns

numeric\_df = df.select\_dtypes(include=['number'])  # Select only numeric columns

top\_products = df.groupby('product\_name')['sales'].sum().nlargest(10)

import pandas as pd

# Load the dataset

df = pd.read\_csv('SuperStoreOrders.csv')

# Display the first few rows and check column names and data types

print(df.head())

print(df.columns)  # Print column names

print(df.dtypes)   # Print data types

# Convert 'Order Date' to datetime

df['order\_date'] = pd.to\_datetime(df['order\_date'], errors='coerce')

# Check for NaT (not-a-time) values in 'Order Date'

print("NaT values in 'order\_date':", df['order\_date'].isnull().sum())

# Clean 'Sales' column and convert to numeric

df['sales'] = pd.to\_numeric(df['sales'].replace({'\$': '', ',': ''}, regex=True), errors='coerce')

# Check for any NaN values in 'Sales' after conversion

print("NaN values in 'sales':", df['sales'].isnull().sum())

# Check if 'Sales' is of numeric type

print("Data type of 'sales':", df['sales'].dtype)

# Ensure there are no datetime types in the 'Sales' or any other relevant columns

# Group by product\_name and sum sales

top\_products = df.groupby('product\_name')['sales'].sum().nlargest(10)

#we do the following to visualize the sales over time

import pandas as pd

import matplotlib.pyplot as plt

# Load the dataset

df = pd.read\_csv('SuperStoreOrders.csv')

# Display the first few rows and check column names

print(df.head())

print(df.columns)  # Print column names

print(df.dtypes)   # Print data types

# Convert 'Order Date' to datetime

df['order\_date'] = pd.to\_datetime(df['order\_date'], errors='coerce')

# Check for NaT (not-a-time) values in 'Order Date'

print("NaT values in 'order\_date':", df['order\_date'].isnull().sum())

# Clean 'Sales' column and convert to numeric

df['sales'] = pd.to\_numeric(df['sales'].replace({'\$': '', ',': ''}, regex=True), errors='coerce')

# Check for any NaN values in 'Sales' after conversion

print("NaN values in 'sales':", df['sales'].isnull().sum())

# Check if 'Sales' is of numeric type

print("Data type of 'sales':", df['sales'].dtype)

# Now group sales by month and sum them

sales\_over\_time = df.groupby(df['order\_date'].dt.to\_period('M'))['sales'].sum()

# Print the grouped sales to check if the operation is successful

print("Sales\_Over\_Time:")

print(sales\_over\_time)

# Plot sales over time

plt.figure(figsize=(12, 6))

plt.plot(sales\_over\_time.index.astype(str), sales\_over\_time.values)

plt.title('Sales\_Over\_Time')

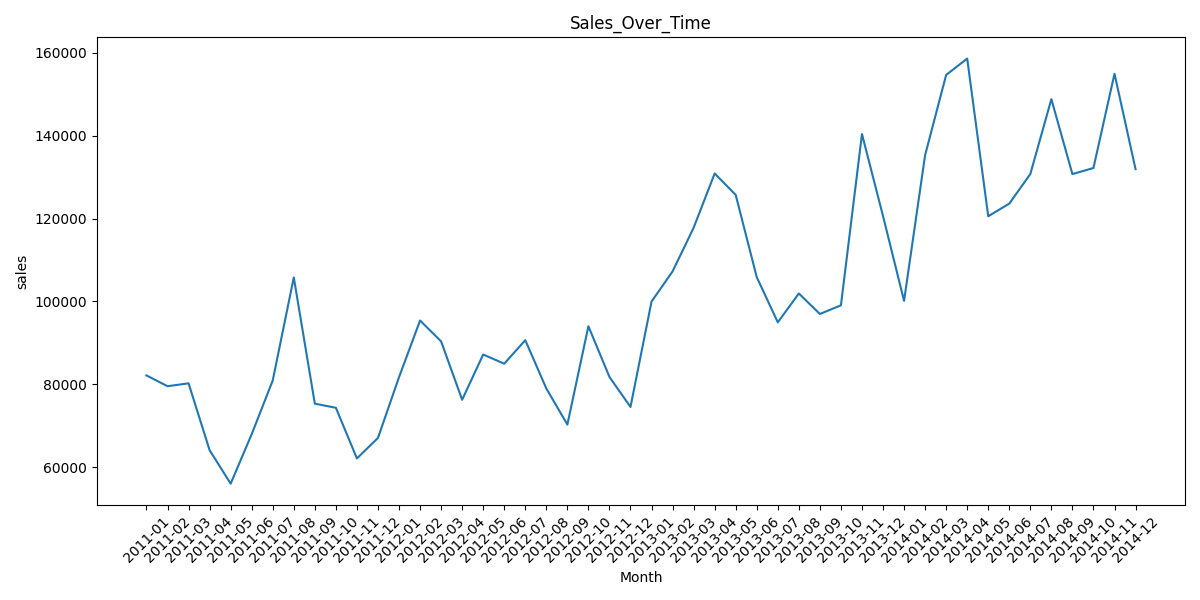
plt.xlabel('Month')

plt.ylabel('sales')

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()



#we do this to show the visualization of top 10 products by sales

import pandas as pd

import matplotlib.pyplot as plt

# Load the dataset

df = pd.read\_csv('SuperStoreOrders.csv')

# Display the first few rows and check column names and data types

print(df.head())

print(df.columns)  # Print column names

print(df.dtypes)   # Print data types

# Convert 'Order Date' to datetime

df['order\_date'] = pd.to\_datetime(df['order\_date'], errors='coerce')

# Clean 'Sales' column and convert to numeric

df['sales'] = pd.to\_numeric(df['sales'].replace({'\$': '', ',': ''}, regex=True), errors='coerce')

# Check for NaN values in 'Sales' after conversion

print("NaN values in 'sales':", df['sales'].isnull().sum())

# Verify if the Sales column is of numeric type

print("Data type of 'sales':", df['sales'].dtype)

# Ensure no datetime types are included in the sum

# Group by 'Product Name' and sum only the 'Sales' column

top\_products = df.groupby('product\_name')['sales'].sum().nlargest(10)

# Print the top products

print("Top 10 Products by sales:")

print(top\_products)

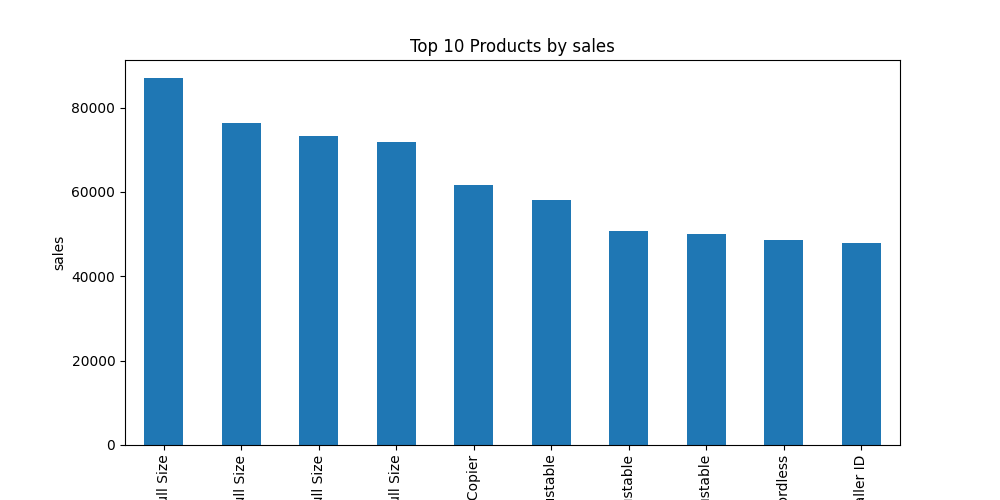
# Bar plot for top 10 products

plt.figure(figsize=(10, 5))

top\_products.plot(kind='bar', title='Top 10 Products by sales')

plt.ylabel('sales')

plt.show()



#we do this to find the image or plotting oe we call it the visual representation of sales by region

import matplotlib.pyplot as plt

import pandas as pd

import matplotlib.pyplot as plt  # Import matplotlib

import seaborn as sns  # Import seaborn for advanced visualizations

# Load the dataset

df = pd.read\_csv('SuperStoreOrders.csv')

# Clean 'Sales' column and convert to numeric

df['sales'] = pd.to\_numeric(df['sales'].replace({'\$': '', ',': ''}, regex=True), errors='coerce')

# Example: Visualizing Sales by Region

plt.figure(figsize=(10, 5))  # Create a figure

sns.barplot(x='region', y='sales', data=df)  # Create a bar plot

plt.title('sales\_by\_region')  # Title for the plot

plt.ylabel('sales')  # Label for Y-axis

plt.xticks(rotation=45)  # Rotate X-axis labels if necessary

plt.show()  # Display the plot

