TASK ----2

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

# Load dataset (replace with your path)

df = pd.read\_csv(r"C:\Users\ajayc\Downloads\GlobalSuperstore.csv\GlobalSuperstore.csv")

# Display basic info

print(df.info())

print(df.describe())

'''Step 2: Data Cleaning

1. Handle Missing Values'''

# Check missing values

print("Missing values:\n", df.isnull().sum())

# Fill missing numeric values with median

for col in df.select\_dtypes(include=np.number).columns:

df[col] = df[col].fillna(df[col].median())

# Fill missing categorical values with mode

for col in df.select\_dtypes(include='object').columns:

df[col] = df[col].fillna(df[col].mode()[0])

<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 object

3 Ship Date 51290 non-null object

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: float64(5), int64(2), object(17)

memory usage: 9.4+ MB

None

Row ID Postal Code Sales Quantity Discount \

count 51290.00000 9994.000000 51290.000000 51290.000000 51290.000000

mean 25645.50000 55190.379428 246.490581 3.476545 0.142908

std 14806.29199 32063.693350 487.565361 2.278766 0.212280

min 1.00000 1040.000000 0.444000 1.000000 0.000000

25% 12823.25000 23223.000000 30.758625 2.000000 0.000000

50% 25645.50000 56430.500000 85.053000 3.000000 0.000000

75% 38467.75000 90008.000000 251.053200 5.000000 0.200000

max 51290.00000 99301.000000 22638.480000 14.000000 0.850000

Profit Shipping Cost

count 51290.000000 51290.000000

mean 28.610982 26.375915

std 174.340972 57.296804

min -6599.978000 0.000000

25% 0.000000 2.610000

50% 9.240000 7.790000

75% 36.810000 24.450000

max 8399.976000 933.570000

Missing values:

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. Remove Duplicates'''

print("Before removing duplicates:", df.shape)

df.drop\_duplicates(inplace=True)

print("After removing duplicates:", df.shape)

Before removing duplicates: (51290, 24)

After removing duplicates: (51290, 24)

'''3. Handle Outliers using IQR Method'''

def remove\_outliers\_iqr(data, column):

Q1 = data[column].quantile(0.25)

Q3 = data[column].quantile(0.75)

IQR = Q3 - Q1

lower = Q1 - 1.5 \* IQR

upper = Q3 + 1.5 \* IQR

return data[(data[column] >= lower) & (data[column] <= upper)]

# Example: Removing outliers from 'Sales' and 'Profit'

df = remove\_outliers\_iqr(df, 'Sales')

df = remove\_outliers\_iqr(df, 'Profit')

'''Step 3: Statistical Analysis'''

# Summary statistics

print("Summary Stats:\n", df[['Sales', 'Profit']].describe())

# Correlation matrix

corr = df.select\_dtypes(include=np.number).corr()

print("Correlation Matrix:\n", corr)

Summary Stats:

Sales Profit

count 37703.000000 37703.000000

mean 88.494541 10.755792

std 100.037706 19.514843

min 0.444000 -40.020000

25% 22.620000 0.771600

50% 51.312000 7.200000

75% 114.390000 20.158400

max 581.250000 66.660000

Correlation Matrix:

Row ID Postal Code Sales Quantity Discount Profit \

Row ID 1.000000 -0.003876 -0.097034 -0.190959 0.067281 -0.033190

Postal Code -0.003876 1.000000 0.032769 0.008797 0.019318 -0.001941

Sales -0.097034 0.032769 1.000000 0.236746 -0.173398 0.400584

Quantity -0.190959 0.008797 0.236746 1.000000 -0.026605 0.155049

Discount 0.067281 0.019318 -0.173398 -0.026605 1.000000 -0.540408

Profit -0.033190 -0.001941 0.400584 0.155049 -0.540408 1.000000

Shipping Cost -0.072586 0.028767 0.759044 0.180057 -0.132758 0.304747

Shipping Cost

Row ID -0.072586

Postal Code 0.028767

Sales 0.759044

Quantity 0.180057

Discount -0.132758

Profit 0.304747

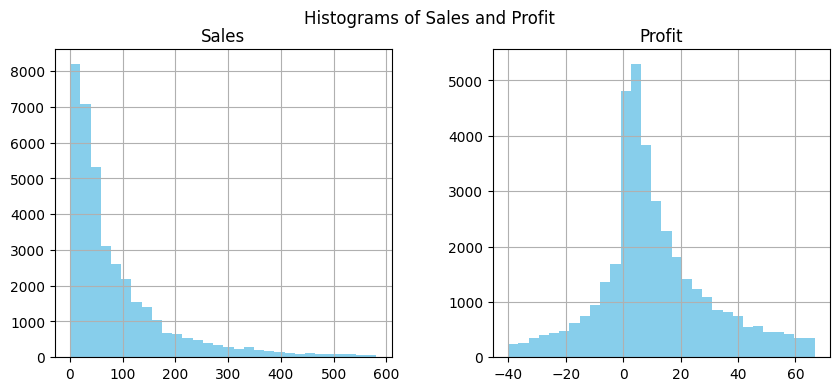
Shipping Cost 1.000000

'''Step 4: Data Visualization'''

df[['Sales', 'Profit']].hist(bins=30, figsize=(10, 4), color='skyblue')

plt.suptitle('Histograms of Sales and Profit')

plt.show()



'''2. Boxplots'''

plt.figure(figsize=(10, 4))

sns.boxplot(data=df[['Sales', 'Profit']])

plt.title("Boxplot of Sales and Profit")

plt.show()

A diagram of sales and profit

AI-generated content may be incorrect.

'''3. Heatmap (Correlations)'''

plt.figure(figsize=(10, 6))

sns.heatmap(corr, annot=True, cmap='coolwarm', fmt=".2f")

plt.title("Correlation Heatmap")

plt.show()

A screenshot of a graph

AI-generated content may be incorrect.

'''Cleaned Dataset:'''

df.to\_csv("Cleaned\_Global\_Superstore.csv", index=False)

print(df.head(5))

Row ID Order ID Order Date Ship Date Ship Mode Customer ID \

685 50937 TZ-2013-3060 9/24/2013 9/24/2013 Same Day HH-5010

815 30241 ID-2011-74175 11/23/2011 11/25/2011 First Class BM-11140

951 17222 ES-2011-4665208 12/10/2011 12/13/2011 First Class BP-11185

962 45019 AL-2013-3830 8/12/2013 8/14/2013 First Class TT-11460

996 34815 US-2013-113649 8/10/2013 8/13/2013 First Class HA-14920

Customer Name Segment City State ... \

685 Hilary Holden Corporate Dar es Salaam Dar Es Salaam ...

815 Becky Martin Consumer Jakarta Jakarta ...

951 Ben Peterman Corporate London England ...

962 Tonja Turnell Home Office Durres Durrës ...

996 Helen Andreada Consumer Fayetteville North Carolina ...

Product ID Category Sub-Category \

685 OFF-HAM-10003368 Office Supplies Appliances

815 TEC-MA-10004553 Technology Machines

951 OFF-AP-10000717 Office Supplies Appliances

962 TEC-CIS-10000436 Technology Phones

996 OFF-AP-10002518 Office Supplies Appliances

Product Name Sales Quantity Discount \

685 Hamilton Beach Blender, Black 553.440 8 0.00

815 Epson Receipt Printer, Red 473.847 5 0.17

951 Hoover Microwave, Black 554.526 2 0.10

962 Cisco Speaker Phone, with Caller ID 553.920 4 0.00

996 Kensington 7 Outlet MasterPiece Power Center 569.536 4 0.20

Profit Shipping Cost Order Priority

685 16.5600 246.05 Critical

815 -5.8530 226.73 Critical

951 -0.0540 207.93 Critical

962 22.0800 206.56 Critical

996 64.0728 203.22 High

[5 rows x 24 columns]

PROJECT 2

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import r2\_score, mean\_squared\_error

from sklearn.model\_selection import train\_test\_split

# Load dataset

df = pd.read\_csv(r"C:\Users\ajayc\Downloads\sales\_data.csv")

# Preview and inspect data

print(df.head())

print(df.info())

print(df.describe())

print("Missing Values:\n", df.isnull().sum())

Product\_ID Sale\_Date Sales\_Rep Region Sales\_Amount Quantity\_Sold \

0 1052 2023-02-03 Bob North 5053.97 18

1 1093 2023-04-21 Bob West 4384.02 17

2 1015 2023-09-21 David South 4631.23 30

3 1072 2023-08-24 Bob South 2167.94 39

4 1061 2023-03-24 Charlie East 3750.20 13

Product\_Category Unit\_Cost Unit\_Price Customer\_Type Discount \

0 Furniture 152.75 267.22 Returning 0.09

1 Furniture 3816.39 4209.44 Returning 0.11

2 Food 261.56 371.40 Returning 0.20

3 Clothing 4330.03 4467.75 New 0.02

4 Electronics 637.37 692.71 New 0.08

Payment\_Method Sales\_Channel Region\_and\_Sales\_Rep

0 Cash Online North-Bob

1 Cash Retail West-Bob

2 Bank Transfer Retail South-David

3 Credit Card Retail South-Bob

4 Credit Card Online East-Charlie

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 1000 entries, 0 to 999

Data columns (total 14 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Product\_ID 1000 non-null int64

1 Sale\_Date 1000 non-null object

2 Sales\_Rep 1000 non-null object

3 Region 1000 non-null object

4 Sales\_Amount 1000 non-null float64

5 Quantity\_Sold 1000 non-null int64

6 Product\_Category 1000 non-null object

7 Unit\_Cost 1000 non-null float64

8 Unit\_Price 1000 non-null float64

9 Customer\_Type 1000 non-null object

10 Discount 1000 non-null float64

11 Payment\_Method 1000 non-null object

12 Sales\_Channel 1000 non-null object

13 Region\_and\_Sales\_Rep 1000 non-null object

dtypes: float64(4), int64(2), object(8)

memory usage: 109.5+ KB

None

Product\_ID Sales\_Amount Quantity\_Sold Unit\_Cost Unit\_Price \

count 1000.000000 1000.000000 1000.000000 1000.000000 1000.000000

mean 1050.128000 5019.265230 25.355000 2475.304550 2728.440120

std 29.573505 2846.790126 14.159006 1417.872546 1419.399839

min 1001.000000 100.120000 1.000000 60.280000 167.120000

25% 1024.000000 2550.297500 13.000000 1238.380000 1509.085000

50% 1051.000000 5019.300000 25.000000 2467.235000 2696.400000

75% 1075.000000 7507.445000 38.000000 3702.865000 3957.970000

max 1100.000000 9989.040000 49.000000 4995.300000 5442.150000

Discount

count 1000.00000

mean 0.15239

std 0.08720

min 0.00000

25% 0.08000

50% 0.15000

75% 0.23000

max 0.30000

Missing Values:

Product\_ID 0

Sale\_Date 0

Sales\_Rep 0

Region 0

Sales\_Amount 0

Quantity\_Sold 0

Product\_Category 0

Unit\_Cost 0

Unit\_Price 0

Customer\_Type 0

Discount 0

Payment\_Method 0

Sales\_Channel 0

Region\_and\_Sales\_Rep 0

dtype: int64

[27]:

import pandas as pd

# Load dataset

df = pd.read\_csv(r"C:\Users\ajayc\Downloads\sales\_data.csv")

# Check structure

print(df.info())

# Check for missing values

print("Missing Values:\n", df.isnull().sum())

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 1000 entries, 0 to 999

Data columns (total 14 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Product\_ID 1000 non-null int64

1 Sale\_Date 1000 non-null object

2 Sales\_Rep 1000 non-null object

3 Region 1000 non-null object

4 Sales\_Amount 1000 non-null float64

5 Quantity\_Sold 1000 non-null int64

6 Product\_Category 1000 non-null object

7 Unit\_Cost 1000 non-null float64

8 Unit\_Price 1000 non-null float64

9 Customer\_Type 1000 non-null object

10 Discount 1000 non-null float64

11 Payment\_Method 1000 non-null object

12 Sales\_Channel 1000 non-null object

13 Region\_and\_Sales\_Rep 1000 non-null object

dtypes: float64(4), int64(2), object(8)

memory usage: 109.5+ KB

None

Missing Values:

Product\_ID 0

Sale\_Date 0

Sales\_Rep 0

Region 0

Sales\_Amount 0

Quantity\_Sold 0

Product\_Category 0

Unit\_Cost 0

Unit\_Price 0

Customer\_Type 0

Discount 0

Payment\_Method 0

Sales\_Channel 0

Region\_and\_Sales\_Rep 0

dtype: int64

'''2. Remove Duplicates'''

print("Before removing duplicates:", df.shape)

df.drop\_duplicates(inplace=True)

print("After removing duplicates:", df.shape)

Before removing duplicates: (1000, 14)

After removing duplicates: (1000, 14)

'''3. Handle Missing Values'''

# Fill numeric columns with median (if missing)

numeric\_cols = ['Sales\_Amount', 'Quantity\_Sold', 'Unit\_Cost', 'Unit\_Price', 'Discount']

for col in numeric\_cols:

if df[col].isnull().sum() > 0:

df[col].fillna(df[col].median(), inplace=True)

# Fill categorical columns with mode

categorical\_cols = ['Sales\_Rep', 'Region', 'Product\_Category', 'Customer\_Type',

'Payment\_Method', 'Sales\_Channel', 'Region\_and\_Sales\_Rep']

for col in categorical\_cols:

if df[col].isnull().sum() > 0:

df[col].fillna(df[col].mode()[0], inplace=True)

'''4. Convert Sale\_Date to Datetime'''

df['Sale\_Date'] = pd.to\_datetime(df['Sale\_Date'], errors='coerce')

df = df.dropna(subset=['Sale\_Date'])

''' Check for and Handle Outliers '''

def remove\_outliers(df, col):

Q1 = df[col].quantile(0.25)

Q3 = df[col].quantile(0.75)

IQR = Q3 - Q1

lower\_bound = Q1 - 1.5 \* IQR

upper\_bound = Q3 + 1.5 \* IQR

return df[(df[col] >= lower\_bound) & (df[col] <= upper\_bound)]

for col in ['Sales\_Amount', 'Unit\_Cost', 'Unit\_Price', 'Discount']:

df = remove\_outliers(df, col)

'''6. Create New Combined Column'''

df['Region\_and\_Sales\_Rep'] = df['Region'] + " - " + df['Sales\_Rep']

print(df.info())

print(df.head())

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 1000 entries, 0 to 999

Data columns (total 14 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Product\_ID 1000 non-null int64

1 Sale\_Date 1000 non-null datetime64[ns]

2 Sales\_Rep 1000 non-null object

3 Region 1000 non-null object

4 Sales\_Amount 1000 non-null float64

5 Quantity\_Sold 1000 non-null int64

6 Product\_Category 1000 non-null object

7 Unit\_Cost 1000 non-null float64

8 Unit\_Price 1000 non-null float64

9 Customer\_Type 1000 non-null object

10 Discount 1000 non-null float64

11 Payment\_Method 1000 non-null object

12 Sales\_Channel 1000 non-null object

13 Region\_and\_Sales\_Rep 1000 non-null object

dtypes: datetime64[ns](1), float64(4), int64(2), object(7)

memory usage: 109.5+ KB

None

Product\_ID Sale\_Date Sales\_Rep Region Sales\_Amount Quantity\_Sold \

0 1052 2023-02-03 Bob North 5053.97 18

1 1093 2023-04-21 Bob West 4384.02 17

2 1015 2023-09-21 David South 4631.23 30

3 1072 2023-08-24 Bob South 2167.94 39

4 1061 2023-03-24 Charlie East 3750.20 13

Product\_Category Unit\_Cost Unit\_Price Customer\_Type Discount \

0 Furniture 152.75 267.22 Returning 0.09

1 Furniture 3816.39 4209.44 Returning 0.11

2 Food 261.56 371.40 Returning 0.20

3 Clothing 4330.03 4467.75 New 0.02

4 Electronics 637.37 692.71 New 0.08

Payment\_Method Sales\_Channel Region\_and\_Sales\_Rep

0 Cash Online North - Bob

1 Cash Retail West - Bob

2 Bank Transfer Retail South - David

3 Credit Card Retail South - Bob

4 Credit Card Online East - Charlie

[ ]: