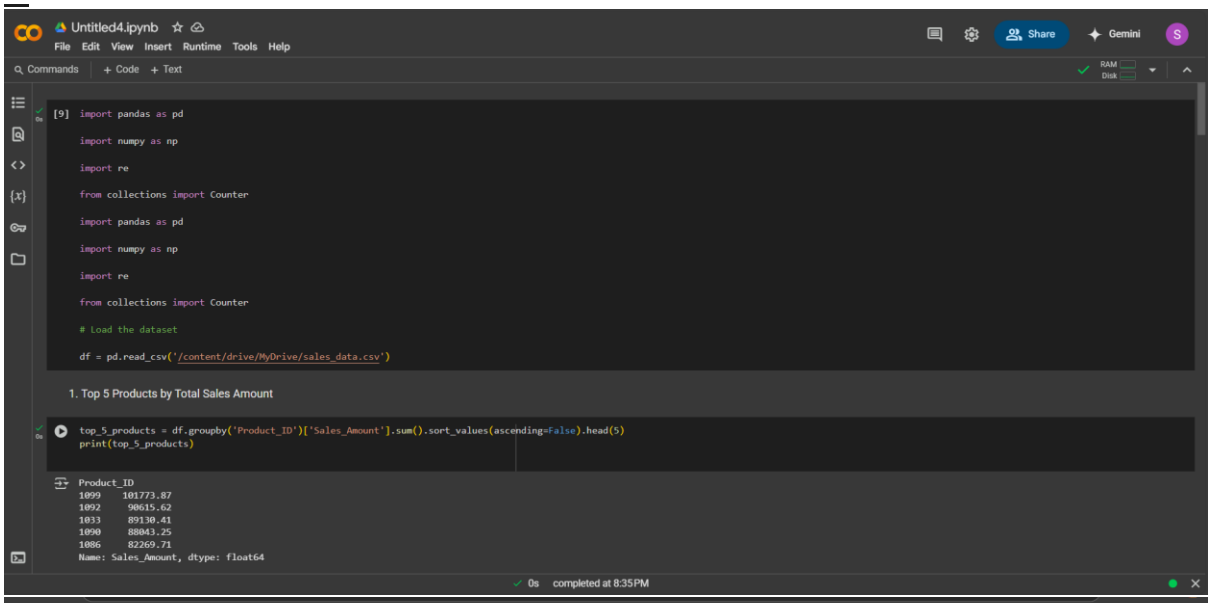
NAME: SHREYAS SUBHASH SALOKHE

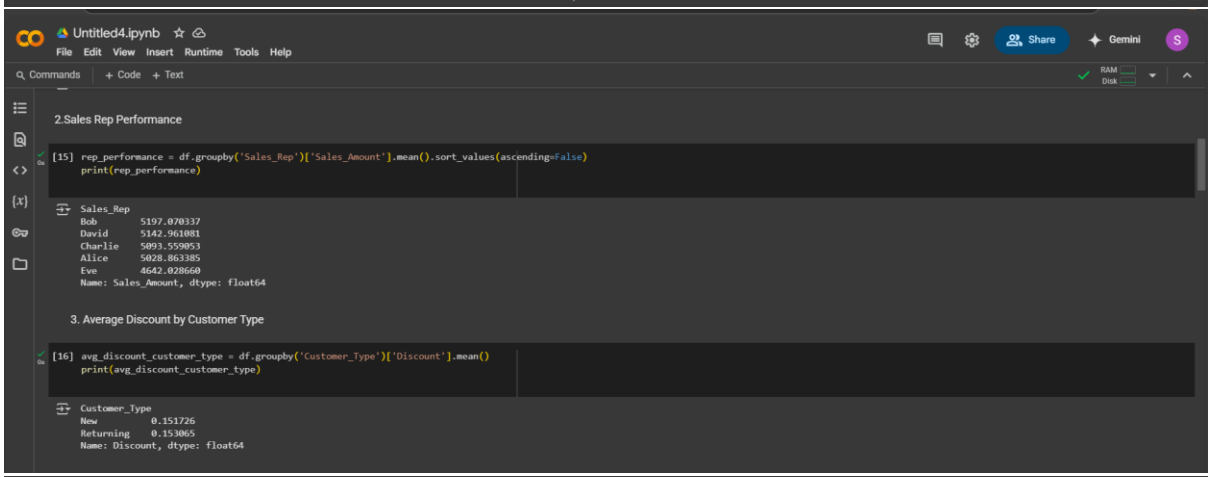
DIV: ET-1

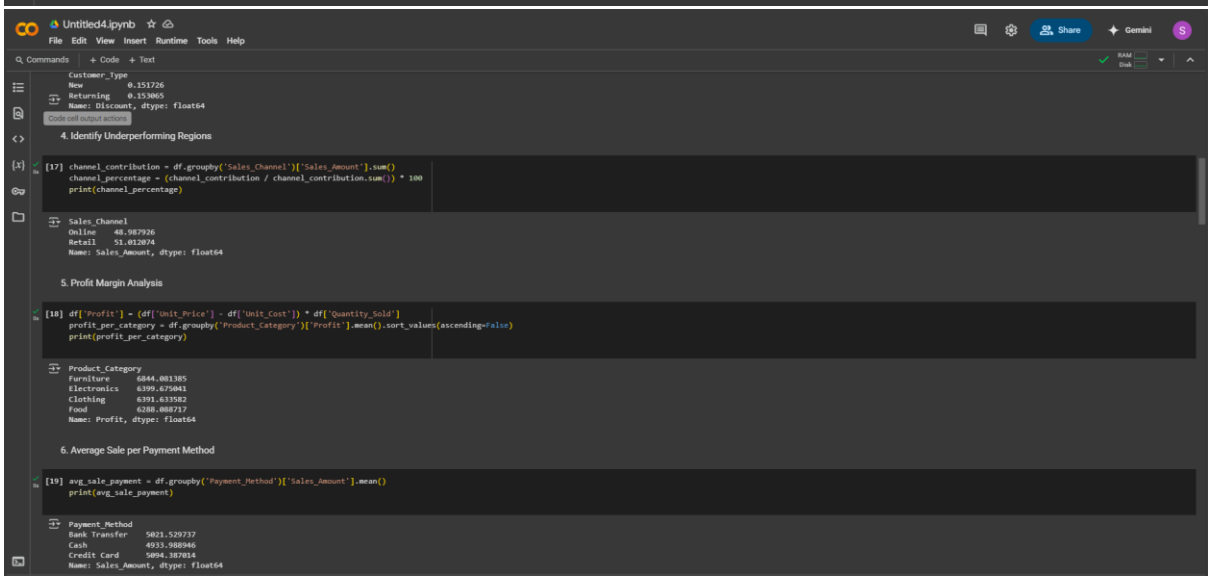
ROLL NO: ET1-44

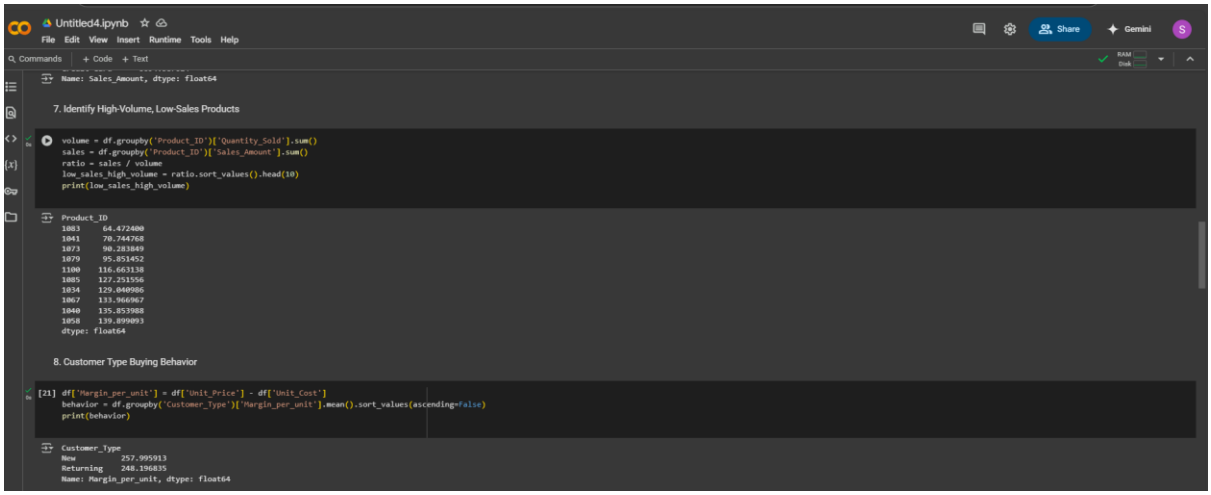
PRN : 202401070129

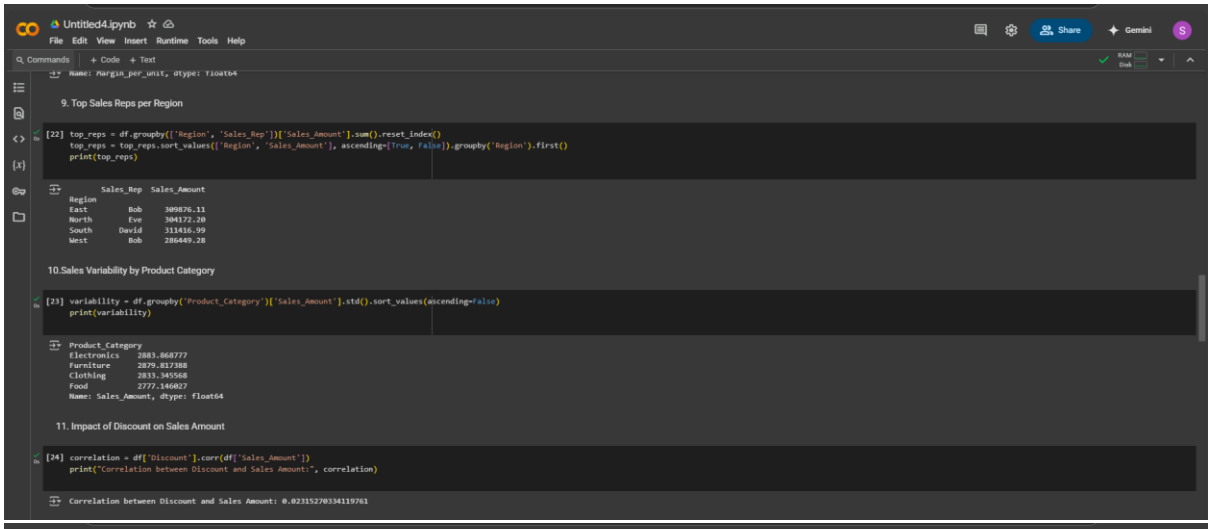
GOOGLE COLAB LINK :
https://colab.research.google.com/drive/1ViN2_lOir-NADjl86Mfz0G6X610cjA_V?usp=sharing

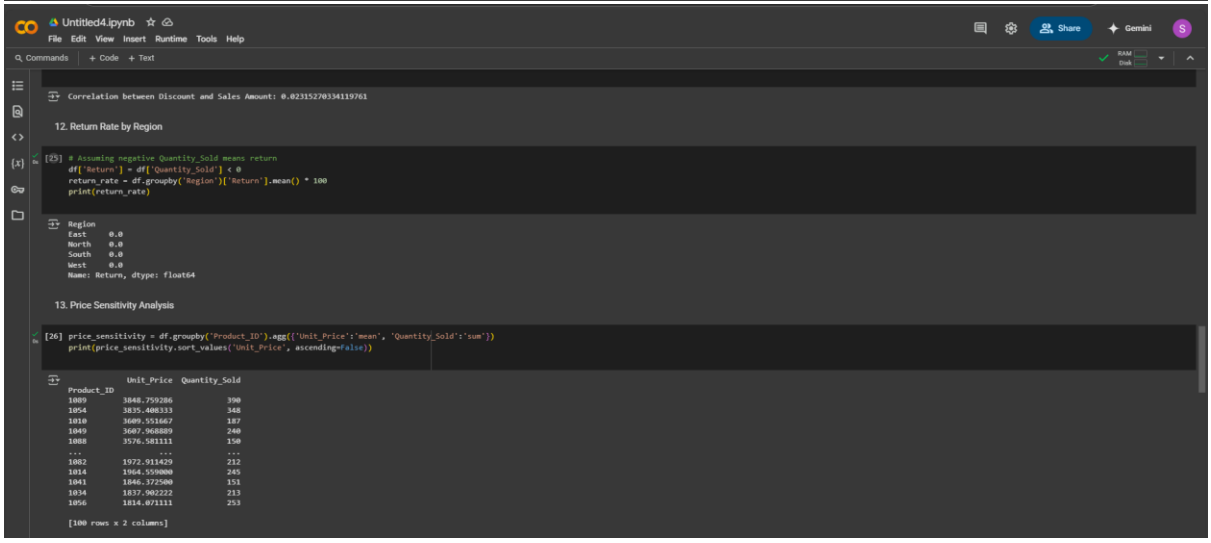

Untitled4.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
[9] import pandas as pd
import numpy as np
import re
from collections import Counter
import pandas as pd
import numpy as np
import re
from collections import Counter
# Load the dataset
df = pd.read_csv('/content/drive/MyDrive/sales_data.csv')
1. Top 5 Products by Total Sales Amount
top_5_products = df.groupby('Product_ID')['Sales_Amount'].sum().sort_values(ascending=False).head(5)
print(top_5_products)
Product_ID
1099 101773.87
1092 90615.62
1033 89130.41
1090 88043.25
1086 82269.71
Name: Sales_Amount, dtype: float64
0s completed at 8:35PM

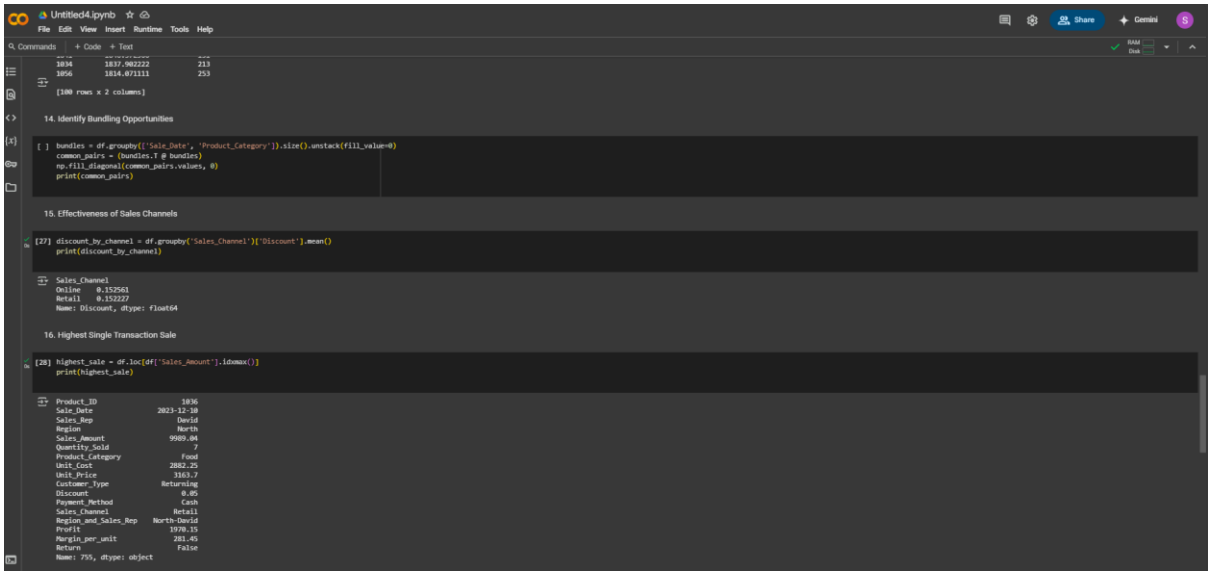

Untitled4.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
2.Sales Rep Performance
[15] rep_performance = df.groupby('Sales_Rep')['Sales_Amount'].mean().sort_values(ascending=False)
print(rep_performance)
Sales_Rep
Bob 5197.070337
David 5142.961081
Charlie 5093.559053
Alice 5028.863385
Eve 4642.028660
Name: Sales_Amount, dtype: float64
3. Average Discount by Customer Type
[16] avg_discount_customer_type = df.groupby('Customer_Type')['Discount'].mean()
print(avg_discount_customer_type)
Customer_Type
New 0.151726
Returning 0.153065
Name: Discount, dtype: float64

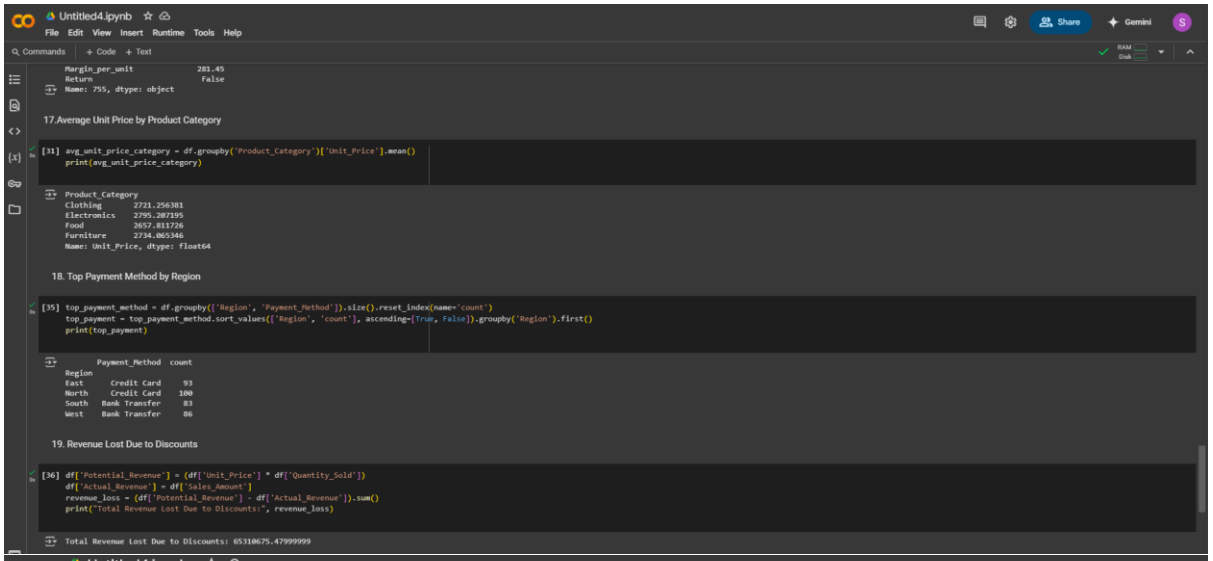

Untitled4.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Customer_Type
New 0.151726
Returning 0.153065
Name: Discount, dtype: float64
Code cell output actions
4. Identify Underperforming Regions
[17] channel_contribution = df.groupby('Sales_Channel')['Sales_Amount'].sum()
channel_percentage = (channel_contribution / channel_contribution.sum()) * 100
print(channel_percentage)
Sales_Channel
Online 48.987926
Retail 51.012074
Name: Sales_Amount, dtype: float64
5. Profit Margin Analysis
[18] df['Profit'] = (df['Unit_Price'] - df['Unit_Cost']) * df['Quantity_Sold']
profit_per_category = df.groupby('Product_Category')['Profit'].mean().sort_values(ascending=False)
print(profit_per_category)
Product_Category
Furniture 6844.081385
Electronics 6399.675041
Clothing 6391.633582
Food 6288.008717
Name: Profit, dtype: float64
6. Average Sale per Payment Method
[19] avg_sale_payment = df.groupby('Payment_Method')['Sales_Amount'].mean()
print(avg_sale_payment)
Payment_Method
Bank Transfer 5021.529737
Cash 4933.988946
Credit Card 5094.387014
Name: Sales_Amount, dtype: float64

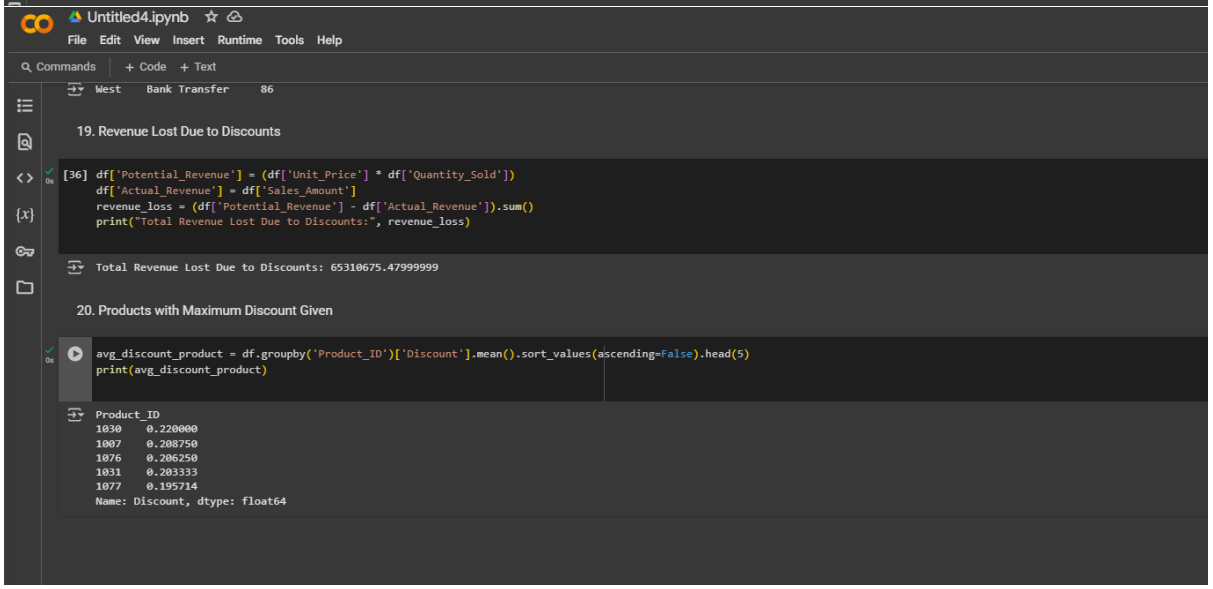

Untitled4.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Name: Sales_Amount, dtype: float64
7. Identify High-Volume, Low-Sales Products
volume = df.groupby('Product_ID')['Quantity_Sold'].sum()
sales = df.groupby('Product_ID')['Sales_Amount'].sum()
ratio = sales / volume
low_sales_high_volume = ratio.sort_values().head(10)
print(low_sales_high_volume)
Product_ID
1083 64.472400
1041 70.744768
1073 90.283849
1079 95.851452
1100 116.663138
1085 127.251556
1034 129.040986
1067 133.966967
1040 135.853988
1058 139.899093
dtype: float64
8. Customer Type Buying Behavior
[21] df['Margin_per_unit'] = df['Unit_Price'] - df['Unit_Cost']
behavior = df.groupby('Customer_Type')['Margin_per_unit'].mean().sort_values(ascending=False)
print(behavior)
Customer_Type
New 257.995913
Returning 248.196835
Name: Margin_per_unit, dtype: float64


Untitled4.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Name: Margin_per_unit, dtype: float64
9. Top Sales Reps per Region
[22] top_reps = df.groupby(['Region', 'Sales_Rep'])['Sales_Amount'].sum().reset_index()
top_reps = top_reps.sort_values(['Region', 'Sales_Amount'], ascending=[True, False]).groupby('Region').first()
print(top_reps)
Sales_Rep Sales_Amount
Region
East Bob 309876.11
North Eve 304172.20
South David 311416.99
West Bob 286449.28
10.Sales Variability by Product Category
[23] variability = df.groupby('Product_Category')['Sales_Amount'].std().sort_values(ascending=False)
print(variability)
Product_Category
Electronics 2883.868777
Furniture 2879.817388
Clothing 2833.345568
Food 2777.146027
Name: Sales_Amount, dtype: float64
11. Impact of Discount on Sales Amount
[24] correlation = df['Discount'].corr(df['Sales_Amount'])
print("Correlation between Discount and Sales Amount:", correlation)
Correlation between Discount and Sales Amount: 0.0231527034119761


Untitled4.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Correlation between Discount and Sales Amount: 0.0231527034119761
12. Return Rate by Region
[25] # Assuming negative Quantity_Sold means return
df['Return'] = df['Quantity_Sold'] < 0
return_rate = df.groupby('Region')['Return'].mean() * 100
print(return_rate)
Region
East 0.0
North 0.0
South 0.0
West 0.0
Name: Return, dtype: float64
13. Price Sensitivity Analysis
[26] price_sensitivity = df.groupby('Product_ID').agg({'Unit_Price':'mean', 'Quantity_Sold':'sum'})
print(price_sensitivity.sort_values('Unit_Price', ascending=False))
Unit_Price Quantity_Sold
Product_ID
1089 3848.759286 390
1054 3835.408333 348
1010 3609.551667 187
1049 3607.968889 240
1088 3576.581111 150
... ...
1082 1972.911429 212
1014 1964.559000 245
1041 1846.372500 151
1034 1837.902222 213
1056 1814.071111 253
[100 rows x 2 columns]


Untitled4.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
1034 1837.982222 213
1056 1814.071111 253
[100 rows x 2 columns]
14. Identify Bundling Opportunities
bundles = df.groupby(['Sale_Date', 'Product_Category']).size().unstack(fill_value=0)
common_pairs = (bundles.T @ bundles)
np.fill_diagonal(common_pairs.values, 0)
print(common_pairs)
15. Effectiveness of Sales Channels
discount_by_channel = df.groupby('Sales_Channel')['Discount'].mean()
print(discount_by_channel)
Sales_Channel
Online 0.152561
Retail 0.152227
Name: Discount, dtype: float64
16. Highest Single Transaction Sale
highest_sale = df.loc[df['Sales_Amount'].idxmax()]
print(highest_sale)
Product_ID 1036
Sale_Date 2023-12-10
Sales_Rep David
Region North
Sales_Amount 9989.04
Quantity_Sold 7
Product_Category Food
Unit_Cost 2882.25
Unit_Price 3163.7
Customer_Type Returning
Discount 0.05
Payment_Method Cash
Sales_Channel Retail
Region_and_Sales_Rep North-David
Profit 1970.15
Margin_per_unit 281.45
Return False
Name: 755, dtype: object


Untitled4.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Margin_per_unit 281.45
Return False
Name: 755, dtype: object
17.Average Unit Price by Product Category
avg_unit_price_category = df.groupby('Product_Category')['Unit_Price'].mean()
print(avg_unit_price_category)
Product_Category
Clothing 2721.256381
Electronics 2795.207195
Food 2657.811726
Furniture 2734.065346
Name: Unit_Price, dtype: float64
18. Top Payment Method by Region
top_payment_method = df.groupby(['Region', 'Payment_Method']).size().reset_index(name='count')
top_payment = top_payment_method.sort_values(['Region', 'count'], ascending=[True, False]).groupby('Region').first()
print(top_payment)
Payment_Method count
Region
East Credit Card 93
North Credit Card 100
South Bank Transfer 83
West Bank Transfer 86
19. Revenue Lost Due to Discounts
df['Potential_Revenue'] = (df['Unit_Price'] * df['Quantity_Sold'])
df['Actual_Revenue'] = df['Sales_Amount']
revenue_loss = (df['Potential_Revenue'] - df['Actual_Revenue']).sum()
print("Total Revenue Lost Due to Discounts:", revenue_loss)
Total Revenue Lost Due to Discounts: 65310675.47999999


Untitled4.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text
West Bank Transfer 86
19. Revenue Lost Due to Discounts
[36] df['Potential_Revenue'] = (df['Unit_Price'] * df['Quantity_Sold'])
df['Actual_Revenue'] = df['Sales_Amount']
revenue_loss = (df['Potential_Revenue'] - df['Actual_Revenue']).sum()
print("Total Revenue Lost Due to Discounts:", revenue_loss)
Total Revenue Lost Due to Discounts: 65310675.47999999
20. Products with Maximum Discount Given
avg_discount_product = df.groupby('Product_ID')['Discount'].mean().sort_values(ascending=False).head(5)
print(avg_discount_product)
Product_ID
1030 0.220000
1007 0.208750
1076 0.206250
1031 0.203333
1077 0.195714
Name: Discount, dtype: float64