**Colab Link:**

<https://colab.research.google.com/drive/1vH7cN_292cwxmw8rohM69d0ITm7qCfRZ?usp=sharing>

**Python Script:**

## \*\*Import needed libraries and dataset\*\*

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import datetime

df=pd.read\_csv('/content/drive/MyDrive/sales\_data\_sample[1].csv', encoding='Latin-1')

df.head(5)

df.info()

## \*\*Data Cleaning\*\*

Convert 'ORDERDATE' column to datetime format and check null values

#change data type

df['ORDERDATE'] = pd.to\_datetime(df['ORDERDATE'])

df['YEAR\_ID'] = df['ORDERDATE'].dt.year

df['MONTH\_ID'] = df['ORDERDATE'].dt.month

df['QTR\_ID'] = df['ORDERDATE'].dt.quarter

# get month names from orderdate column

df['MONTH\_NAME'] = pd.DatetimeIndex(df['ORDERDATE']).month\_name()

df.info()

df.isnull().sum()

df.head()

drop Unneeded Columns

to\_drop = ['PHONE','STATE',

'POSTALCODE', 'TERRITORY', 'CONTACTFIRSTNAME', 'CONTACTLASTNAME']

df = df.drop(to\_drop, axis = 1)

Merge the Two Adress Columns into one Column

and Rename PRICEEACH column to Product Cost

df["ADDRESS"] = df["ADDRESSLINE1"].fillna(' ') + " " + df["ADDRESSLINE2"].fillna('')

df.drop(['ADDRESSLINE1','ADDRESSLINE2'], axis=1, inplace=True)

df = df.rename(columns={'PRICEEACH': 'Product Cost'})

df.head(10)

Check if there is any null Values

df.isnull().sum()

#Data analysis

Data Discription with Basic Statistics

df.describe()

Combine Similar Product Description

pd.pivot\_table(

df,

index=['PRODUCTLINE'],

values=['Product Cost','QUANTITYORDERED','SALES'],

aggfunc=['sum']

)

# Grouping by PRODUCTCODE and aggregating relevant columns for consistency

product\_summary = df.groupby('PRODUCTCODE').agg({

'QUANTITYORDERED': 'sum',

'SALES': 'sum',

'ORDERNUMBER': 'nunique',

'Product Cost': 'sum'

}).reset\_index()

# Renaming columns for clarity

product\_summary.columns = ['PRODUCTCODE' ,'TOTAL\_QUANTITY', 'TOTAL\_SALES', 'UNIQUE\_ORDERS', 'Total\_cost']

product\_summary

# Top Customers By Number of orders

top\_customer = df.groupby(['CUSTOMERNAME']).count().sort\_values('ORDERNUMBER', ascending = False).head(5)

top\_customer = top\_customer[['ORDERNUMBER']].round(3)

top\_customer.reset\_index(inplace = True)

top\_customer.head()

TopCustomers= sns.barplot(x='CUSTOMERNAME',y='ORDERNUMBER',data=top\_customer,hue='CUSTOMERNAME',palette='icefire',

edgecolor='black', dodge=False, width= 0.8)

plt.xticks(rotation=30)

plt.show()

# TOP CUSTOMERS BY SALES

top\_customer = df.groupby(['CUSTOMERNAME'])['SALES'].sum().sort\_values(ascending = False).head(5).reset\_index()

top\_customer = top\_customer.round(3)

top\_customer.head()

Top\_customers= sns.barplot(x='CUSTOMERNAME',y='SALES',data=top\_customer,hue='CUSTOMERNAME',palette='icefire',

edgecolor='black', dodge=False, width= 0.8)

plt.xticks(rotation=30)

plt.show()

# Customer Segments

Customer\_Segment = df.groupby('CUSTOMERNAME').agg(

first\_order\_date=pd.NamedAgg(column='ORDERDATE', aggfunc='min'),

Last\_order\_date=pd.NamedAgg(column='ORDERDATE', aggfunc='max'),

Total\_Quantity\_ordered=pd.NamedAgg(column='QUANTITYORDERED', aggfunc='count'),

Number\_of\_orders=pd.NamedAgg(column='ORDERNUMBER', aggfunc='nunique'),

SALES=pd.NamedAgg(column='SALES', aggfunc='sum')

).reset\_index()

Customer\_Segment['AVG\_ORDER\_VALUE'] = Customer\_Segment['SALES'] / Customer\_Segment['Number\_of\_orders']

Customer\_Segment['CLTV']=Customer\_Segment['AVG\_ORDER\_VALUE']\*Customer\_Segment['Number\_of\_orders']\*3

Customer\_Segment.sort\_values(by='SALES', ascending=False)

#Customer Segmentaion:

def Customer\_Segment\_Categorization(QUANTITYORDERED):

if QUANTITYORDERED <= 30.68:

return 'Low'

elif 30.68<=QUANTITYORDERED <= 30.68\*1.1:

return 'Medium'

else:

return 'High'

# Apply the categorization function to the QUANTITYORDERED column

Customer\_Segment['CUSTOMER\_ORDERSVALUE'] = Customer\_Segment['Total\_Quantity\_ordered'].apply(Customer\_Segment\_Categorization)

#CUSROMERS ORDERS VALUE

def Customer\_Segment\_Categorization(SALES):

if SALES <= 109050.31:

return 'Low'

elif 109050.31<=SALES <= 109050.31\*1.1:

return 'Medium'

else:

return 'High'

# Apply the categorization function to the SALES column

Customer\_Segment['CUSTOMER\_SALESVALUE'] = Customer\_Segment['SALES'].apply(Customer\_Segment\_Categorization)

Customer\_Segment.head()

Customer\_Segment.describe()

Orders Mean=30.68

Sales Mean =109050.31

So i want to compare each customer sales and orders with the means, less than the mean are categorized as "Low", within 10% of the mean are categorized as "Medium", and greater than 10% above the mean are categorized as "High".

print(Customer\_Segment['CUSTOMER\_SALESVALUE'].value\_counts())

Customer\_Segment['CUSTOMER\_ORDERSVALUE'].value\_counts()

# Create two pie charts, one for customer orders value and one for customer sales value.

# Customer Orders Value Pie Chart

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 5))

labels = Customer\_Segment['CUSTOMER\_ORDERSVALUE'].value\_counts().index.to\_list()

values = Customer\_Segment['CUSTOMER\_ORDERSVALUE'].value\_counts().to\_numpy()

ax1.pie(values, labels=labels, autopct='%1.1f%%', startangle=90)

ax1.set\_title('Customer Orders Value')

# Customer Sales Value Pie Chart

labels = Customer\_Segment['CUSTOMER\_SALESVALUE'].value\_counts().index.to\_list()

values = Customer\_Segment['CUSTOMER\_SALESVALUE'].value\_counts().to\_numpy()

ax2.pie(values, labels=labels, autopct='%1.1f%%', startangle=90)

ax2.set\_title('Customer Sales Value')

# Adjust spacing between subplots

plt.tight\_layout()

plt.show()

customer\_lifespans = df.groupby('CUSTOMERNAME')['MONTH\_ID'].apply(lambda x: x.max() - x.min())

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

sns.histplot(customer\_lifespans, bins=10, kde=True)

plt.title('Customer Lifespan Distribution')

plt.xlabel('Lifespan (Years)')

plt.ylabel('Frequency')

plt.show()

Customer Lifespan Distribution

What It Shows: This chart visualizes the distribution of customer lifespans, measured in years. It represents how long customers continue to make purchases from the time they first make a purchase to their last recorded purchase.

Why It’s Useful: Understanding the distribution of customer lifespans helps businesses gauge customer retention and loyalty. A longer lifespan indicates better customer retention, which is crucial for sustained business growth

df.sort\_values(by=['ORDERNUMBER','ORDERDATE']).groupby('ORDERNUMBER').head()

df.sort\_values(by=['CUSTOMERNAME','ORDERDATE']).groupby('CUSTOMERNAME').head()

What are the trends in orders over time?

sns.jointplot(x='YEAR\_ID', y='QUANTITYORDERED', data=df,kind='hex')

data = df.groupby(['YEAR\_ID','MONTH\_ID'])['SALES'].sum()

data.plot(kind='bar')

# Total sales by year

total\_sales\_by\_year = df.groupby('YEAR\_ID')['SALES'].sum().reset\_index()

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

sns.barplot(x='YEAR\_ID', y='SALES', data=total\_sales\_by\_year)

plt.title('Total Sales by Year')

plt.show()

monthly\_orders = df.groupby(['YEAR\_ID', 'MONTH\_ID'])['QUANTITYORDERED'].sum().reset\_index()

# Pivoting data for plotting

monthly\_orders\_pivot = monthly\_orders.pivot(index='MONTH\_ID', columns='YEAR\_ID', values='QUANTITYORDERED')

# Plotting monthly orders trends

plt.figure(figsize=(14, 7))

monthly\_orders\_pivot.plot(kind='line', marker='o')

plt.title('Monthly QUANTITY ORDERED Trends')

plt.xlabel('Month')

plt.ylabel('Total Orders')

plt.grid(True)

plt.show()

THE HIGHST SALES ARE IN NOVEMBER

Description: Aggregating total sales by month to identify any recurring monthly patterns. Insights:

Certain months may consistently show higher sales due to factors like holiday seasons (e.g., November having higher sales which may be as a result of Christmas campaigns). Monthly trends can reveal the effectiveness of monthly marketing campaigns

sns.barplot(x='YEAR\_ID', y='SALES', hue='QTR\_ID', data=df)

plt.title("Bar plot of sales over year and quarter")

Quarters that align with fiscal year-end or significant business events may show increased sales. Helps in understanding broader seasonal impacts compared to monthly trends.

deal\_size\_by\_quarter = df.groupby(['QTR\_ID', 'DEALSIZE']).size().unstack().fillna(0)

# Plotting deal size distribution by quarter using a side-by-side bar chart

deal\_size\_by\_quarter.plot(kind='bar', figsize=(12, 6),

color=['red', 'violet', 'green'],

edgecolor='black')

plt.title('Deal Size Distribution by Quarter')

plt.xlabel('Quarter')

plt.ylabel('Number of Deals')

plt.legend(title='Deal Size')

plt.grid(axis='y')

plt.show()

What is the percentages of deal size?

Deal\_Size =df['DEALSIZE'].value\_counts()

plt.pie(

Deal\_Size,

labels=Deal\_Size.index,

autopct='%1.1f%%'

);

SALES OF EACH COUNTRY OVER YEARS

cou1 = px.treemap(df,path=['YEAR\_ID','COUNTRY'],values='SALES',

height = 600,width = 800)

cou1.show()

df.to\_csv('sales\_data\_sample.csv', index=False)

Customer\_Segment.to\_csv('Customer\_Segment.csv', index=False)