Task 1: Exploratory Data Analysis (EDA) and business insights

**Insights Derived from the Data:**

**1.Regional Performance**:

South America leads with the highest total sales of $219,352.56, followed by Europe ($166,254.63) and North America ($152,313.40).

Asia is close behind with sales of $152,074.97.

**2.Category-wise Sales**:

Books generated the highest revenue ($192,147.47), followed by Electronics ($180,783.50).

Clothing and Home Decor contributed $166,170.66 and $150,893.93, respectively.

**3.Customer Acquisition Trend:**

The majority of customers signed up in 2024 (79 new customers), indicating a rising customer acquisition trend.

Customer growth in 2023 and 2022 was 57 and 64, respectively.

**4.Top Products by Quantity Sold:**

The most sold products are:

ActiveWear Smartwatch: 100 units.

SoundWave Headphones: 97 units.

HomeSense Desk Lamp: 81 units.

**5.Actionable Insight**:

Focusing on South America and promoting the Books and Electronics categories could further boost revenue.

Continuously analyzing customer acquisition trends helps refine marketing strategies.

# Saving the EDA code as a Python script for the user

eda\_code = """

# Importing necessary libraries

import pandas as pd

# Loading the datasets

customers = pd.read\_csv('Customers.csv')

products = pd.read\_csv('Products.csv')

transactions = pd.read\_csv('Transactions.csv')

# Converting date columns to datetime format for easier analysis

customers['SignupDate'] = pd.to\_datetime(customers['SignupDate'])

transactions['TransactionDate'] = pd.to\_datetime(transactions['TransactionDate'])

# Checking for duplicates in all datasets

customers\_duplicates = customers.duplicated().sum()

products\_duplicates = products.duplicated().sum()

transactions\_duplicates = transactions.duplicated().sum()

# Basic descriptive statistics for numerical columns in Products and Transactions datasets

products\_describe = products.describe()

transactions\_describe = transactions.describe()

# Checking unique counts for categorical columns

customers\_unique\_regions = customers['Region'].value\_counts()

products\_unique\_categories = products['Category'].value\_counts()

# Merging datasets for detailed analysis

transactions\_merged = transactions.merge(customers, on='CustomerID', how='left').merge(products, on='ProductID', how='left')

# Analyzing transactions by region

region\_sales = transactions\_merged.groupby('Region')['TotalValue'].sum().sort\_values(ascending=False)

# Analyzing sales by product category

category\_sales = transactions\_merged.groupby('Category')['TotalValue'].sum().sort\_values(ascending=False)

# Analyzing customer acquisition over time

customers['SignupYear'] = customers['SignupDate'].dt.year

signup\_trend = customers['SignupYear'].value\_counts().sort\_index()

# Analyzing most sold products

most\_sold\_products = transactions\_merged.groupby('ProductName')['Quantity'].sum().sort\_values(ascending=False).head(5)

# Printing Results

print("Duplicates in Customers:", customers\_duplicates)

print("Duplicates in Products:", products\_duplicates)

print("Duplicates in Transactions:", transactions\_duplicates)

print("Products Description:\\n", products\_describe)

print("Transactions Description:\\n", transactions\_describe)

print("Region-wise Sales:\\n", region\_sales)

print("Category-wise Sales:\\n", category\_sales)

print("Signup Trends:\\n", signup\_trend)

print("Most Sold Products:\\n", most\_sold\_products)

"""

# Saving the Python script to a file

eda\_script\_path = "/mnt/data/EDA\_Script.py"

with open(eda\_script\_path, "w") as file:

file.write(eda\_code)

# Re-creating the PDF report as it wasn't generated earlier

pdf = PDFReport()

pdf.add\_page()

# Title

pdf.set\_font('Arial', 'B', 16)

pdf.cell(0, 10, 'Exploratory Data Analysis (EDA) and Business Insights', ln=True, align='C')

pdf.ln(10)

# Insights Section

pdf.chapter\_title('Business Insights')

pdf.chapter\_body(

"1. Regional Performance:\n"

" - South America is the top-performing region with total sales of $219,352.56, followed by Europe and North America.\n"

"\n"

"2. Category-wise Sales:\n"

" - Books and Electronics lead in revenue generation with $192,147.47 and $180,783.50, respectively.\n"

"\n"

"3. Customer Acquisition Trends:\n"

" - 2024 saw the highest customer growth with 79 new signups. Customer acquisition is steadily increasing.\n"

"\n"

"4. Top Products by Quantity Sold:\n"

" - ActiveWear Smartwatch and SoundWave Headphones are the most popular products.\n"

"\n"

"5. Actionable Insights:\n"

" - Strengthening promotions in South America and focusing on Books and Electronics could further boost revenue.\n"

" - Monitoring customer signups and adjusting marketing strategies based on trends is essential."

)

# Save the PDF

pdf\_report\_path = "/mnt/data/eCommerce\_Transactions\_Report.pdf"

pdf.output(pdf\_report\_path)

eda\_script\_path, pdf\_report\_path

Task 2: Lookalike Model:

from sklearn.metrics.pairwise import cosine\_similarity

from sklearn.preprocessing import StandardScaler

# Preparing the data by aggregating transaction and customer information

# Aggregating transaction history for each customer

customer\_transactions = transactions\_merged.groupby('CustomerID').agg({

'TotalValue': 'sum', # Total spending

'Quantity': 'sum', # Total quantity purchased

'Category': lambda x: ','.join(x) # Categories purchased (for preference analysis)

}).reset\_index()

# Merging with customer information

customer\_profiles = customers.merge(customer\_transactions, on='CustomerID', how='left')

# Filling missing values for customers without transactions

customer\_profiles.fillna({'TotalValue': 0, 'Quantity': 0, 'Category': ''}, inplace=True)

# Encoding categorical data for similarity analysis

region\_dummies = pd.get\_dummies(customer\_profiles['Region'], prefix='Region')

# Adding numeric features

numeric\_features = customer\_profiles[['TotalValue', 'Quantity']]

scaler = StandardScaler()

numeric\_features\_scaled = scaler.fit\_transform(numeric\_features)

# Combining all features

final\_features = pd.concat([

pd.DataFrame(numeric\_features\_scaled, columns=['TotalValue\_Scaled', 'Quantity\_Scaled']),

region\_dummies

], axis=1)

# Calculating similarity scores for the first 20 customers (C0001 - C0020)

customer\_ids = customer\_profiles['CustomerID']

similarity\_matrix = cosine\_similarity(final\_features)

lookalike\_map = {}

for i in range(20): # First 20 customers

customer\_id = customer\_ids.iloc[i]

similarities = list(enumerate(similarity\_matrix[i]))

# Exclude the customer itself and sort by similarity score

top\_similar = sorted([sim for sim in similarities if sim[0] != i], key=lambda x: x[1], reverse=True)[:3]

lookalike\_map[customer\_id] = [(customer\_ids.iloc[sim[0]], sim[1]) for sim in top\_similar]

# Creating a DataFrame for Lookalike.csv

lookalike\_list = [{'CustomerID': cust\_id, 'Lookalikes': lookalikes} for cust\_id, lookalikes in lookalike\_map.items()]

lookalike\_df = pd.DataFrame(lookalike\_list)

# Saving the Lookalike map as a CSV file

lookalike\_csv\_path = "/mnt/data/Lookalike.csv"

lookalike\_df.to\_csv(lookalike\_csv\_path, index=False)

lookalike\_csv\_path

Task 3 : Customer Segmentation / clustering

from sklearn.preprocessing import StandardScaler, OneHotEncoder

from sklearn.cluster import KMeans

from sklearn.metrics import davies\_bouldin\_score

from sklearn.decomposition import PCA

import matplotlib.pyplot as plt

# Preparing data for clustering

# Encoding categorical variables (e.g., Region) and scaling numerical variables

encoder = OneHotEncoder(sparse=False)

regions\_encoded = encoder.fit\_transform(customers[['Region']])

scaler = StandardScaler()

scaled\_transactions = scaler.fit\_transform(customer\_profile[['TotalValue']])

# Combining encoded regions, scaled transaction values, and signup year as features

customers['SignupYear'] = customers['SignupDate'].dt.year

signup\_year\_scaled = scaler.fit\_transform(customers[['SignupYear']])

# Combining all features

feature\_matrix = np.hstack([regions\_encoded, scaled\_transactions, signup\_year\_scaled])

# Performing clustering using KMeans

db\_index\_values = []

cluster\_range = range(2, 11)

cluster\_results = {}

for k in cluster\_range:

kmeans = KMeans(n\_clusters=k, random\_state=42, n\_init=10)

labels = kmeans.fit\_predict(feature\_matrix)

db\_index = davies\_bouldin\_score(feature\_matrix, labels)

db\_index\_values.append(db\_index)

cluster\_results[k] = labels

# Selecting the best number of clusters based on the DB Index (lower is better)

optimal\_k = cluster\_range[np.argmin(db\_index\_values)]

# Re-running KMeans with the optimal number of clusters

final\_kmeans = KMeans(n\_clusters=optimal\_k, random\_state=42, n\_init=10)

final\_labels = final\_kmeans.fit\_predict(feature\_matrix)

# Adding cluster labels to the customer data for visualization

customers['Cluster'] = final\_labels

# Dimensionality reduction using PCA for visualization

pca = PCA(n\_components=2)

reduced\_features = pca.fit\_transform(feature\_matrix)

# Visualizing the clusters

plt.figure(figsize=(10, 7))

plt.scatter(reduced\_features[:, 0], reduced\_features[:, 1], c=final\_labels, cmap='viridis', s=50)

plt.title(f'Customer Clusters (k={optimal\_k})', fontsize=14)

plt.xlabel('PCA Component 1', fontsize=12)

plt.ylabel('PCA Component 2', fontsize=12)

plt.colorbar(label='Cluster ID')

plt.show()

# Saving clustering results and DB Index values for the report

clustering\_report\_path = "/mnt/data/Customer\_Clustering\_Report.txt"

with open(clustering\_report\_path, "w") as report\_file:

report\_file.write(f"Optimal Number of Clusters: {optimal\_k}\n")

report\_file.write(f"Davies-Bouldin Index for Optimal Clusters: {min(db\_index\_values):.4f}\n")

report\_file.write(f"DB Index Values for Cluster Range (2-10):\n")

for k, db\_index in zip(cluster\_range, db\_index\_values):

report\_file.write(f" k={k}: DB Index={db\_index:.4f}\n")

clustering\_report\_path