TASK 1

Exploratory Data Analysis (EDA) and Business Insights

To provide a comprehensive response, I would need access to the specific dataset you are referring to for the Exploratory Data Analysis (EDA). However, I can guide you through the general steps of performing EDA and suggest how to derive business insights based on typical findings.

1 datasets for Exploratory Data Analysis (EDA)

1. Data Collection: Gather the dataset and load it into a suitable environment (e.g., Python with Pandas, R).

2. Data Cleaning: Check for missing values, duplicates, and outliers. Handle them appropriately through imputation or removal.

3. Descriptive Statistics: Generate summary statistics (mean, median, mode, standard deviation) to understand the data distribution.

4. Data Visualization: Use visual tools like histograms, box plots, scatter plots, and heatmaps to identify patterns and relationships.

5. Correlation Analysis: Analyze correlations between variables to identify potential relationships that could inform business decisions.

2 five hypothetical business insights derived from EDA:

1.Customer Segmentation: The analysis reveals distinct customer segments based on purchasing behavior. Targeted marketing strategies can be developed for each segment to enhance engagement and sales.

2.Sales Trends: Seasonal sales trends indicate a significant increase during holiday periods. This insight suggests that promotional campaigns should be intensified during these times to maximize revenue.

3. Product Performance: Certain products consistently outperform others across different regions. Focusing inventory and marketing efforts on high-performing products can improve overall profitability.

4. Churn Analysis: A notable percentage of customers who make infrequent purchases tend to churn. Implementing loyalty programs or personalized follow-ups could help retain these customers.

5. Pricing Sensitivity: The correlation between price changes and sales volume indicates high sensitivity in certain product categories. Strategic pricing adjustments can optimize revenue without alienating price-sensitive customers.

TASK 2

To build a Lookalike Model that recommends similar customers based on user information, you can follow these structured steps:

Step 1: Data Collection

Gather the necessary data, which includes:

Customer Information: Demographics (age, gender, location), preferences, and behavior patterns.

Transaction History: Purchase history, frequency of purchases, and product categories.

Step 2: Data Preprocessing

1. Data Cleaning: Remove duplicates and handle missing values.

2. Feature Engineering: Create relevant features that represent customer behavior and preferences effectively.

3. Normalization: Scale numerical features to ensure uniformity.

Step 3: Similarity Calculation

Utilize techniques such as:

Cosine Similarity: Measures the cosine of the angle between two non-zero vectors of an inner product space, useful for high-dimensional data.

Euclidean Distance: Calculates the straight-line distance between two points in multi-dimensional space.

Step 4: Model Building

1. Choose a Model: Use machine learning algorithms like K-Nearest Neighbors (KNN) or clustering methods (e.g., K-Means) to identify similar customers.

2. Train the Model: Fit the model on historical customer data to learn patterns.

Step 5: Recommendation Generation

1. Input User Profile: When a user inputs their information, extract their features.

2. Find Similar Customers: Use the trained model to find customers with the highest similarity scores based on the input profile.

3. Assign Similarity Scores: Calculate and assign a similarity score to each recommended customer.

Step 6: Output Recommendations

Provide a list of recommended customers along with their similarity scores. For example:

Recommended Customer	Similarity Score
Customer A	0.92
Customer B	0.88
Customer C	0.85

Example Implementation

Here's a simplified Python code snippet illustrating how you might implement this model using KNN:

```python
import pandas as pd
from sklearn.neighbors import NearestNeighbors
# Load customer data
data = pd.read_csv('customer_data.csv')
# Preprocess data (cleaning, feature engineering)
# Define features for modeling
features = data[['age', 'gender', 'purchase_frequency', 'product_category']]
# Fit KNN model
knn = NearestNeighbors(n_neighbors=3)
knn.fit(features)
# Input user profile
user_profile = [[25, 'Female', 5, 'Electronics']]  # Example input
# Find similar customers
distances, indices = knn.kneighbors(user_profile)
# Output recommended customers and their similarity scores
recommended_customers = data.iloc[indices[0]]
similarity_scores = 1 / (1 + distances[0])  # Inverse distance for similarity score
recommended_customers['Similarity Score'] = similarity_scores
print(recommended_customers[['Customer ID', 'Similarity Score']])
```

This approach will allow you to effectively recommend similar customers based on their profiles and transaction histories while assigning a meaningful similarity score to each recommendation.

TASK 3

Customer segementation / cluster

To perform customer segmentation using clustering techniques based on the provided profile information (from `Customers.csv`) and transaction information (from `Transactions.csv`), follow these structured steps. Below is an outline of the process, including the clustering algorithm, metrics, and visualization.

Step 1: Data Preparation

1.1 Load Data

Load the customer and transaction datasets using Pandas.

python

import pandas as pd

Load datasets

customers = pd.read_csv('Customers.csv')

transactions = pd.read_csv('Transactions.csv')

1.2 Data Cleaning

Clean the data by handling missing values and duplicates**.**

python

Check for missing values

customers.dropna(inplace=True)

transactions.dropna(inplace=True)

Remove duplicates

customers.drop_duplicates(inplace=True)

transactions.drop_duplicates(inplace=True)

1.3 Merge Datasets

Merge the customer and transaction data on a common identifier (e.g., Customer ID).

python

data = pd.merge(customers, transactions, on='CustomerID')

1.4 Feature Engineering

Create relevant features for clustering, such as total spending, frequency of purchases, and recency of purchases.

python

Example feature engineering

data['Total_Spending'] = data.groupby('CustomerID')['Amount'].transform('sum')

data['Purchase_Frequency'] = data.groupby('CustomerID')['TransactionID'].transform('count') data['Recency'] = (data['Date'].max() - data['Date']).dt.days

Step 2: Clustering

2.1 Choose Clustering Algorithm

Select a clustering algorithm (e.g., K-Means) and determine the number of clusters (between 2 and 10).

python

from sklearn.cluster import KMeans

Select features for clustering

features = data[['Total_Spending', 'Purchase_Frequency', 'Recency']]

Determine optimal number of clusters using Elbow method

```
inertia = []
for k in range(2, 11):
    kmeans = KMeans(n_clusters=k)
    kmeans.fit(features)
    inertia.append(kmeans.inertia_)
```

2.2 Fit K-Means Model

Fit the K-Means model with the chosen number of clusters.

python

```
optimal_k = 4  # Chosen based on Elbow method analysis
kmeans = KMeans(n_clusters=optimal_k)
data['Cluster'] = kmeans.fit_predict(features)
```

Step 3: Evaluation Metrics

3.1 Calculate DB Index

Calculate the Davies-Bouldin Index to evaluate clustering quality.

python

```
from sklearn.metrics import davies_bouldin_score
db_index = davies_bouldin_score(features, data['Cluster'])
```

3.2 Other Relevant Metrics

Consider other metrics such as silhouette score.

python

```
from sklearn.metrics import silhouette_score
silhouette_avg = silhouette_score(features, data['Cluster'])
```

Step 4: Visualization

4.1 Visualize Clusters

Use Matplotlib or Seaborn to visualize the clusters.

python

```
import matplotlib.pyplot as plt

import seaborn as sns

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

sns.scatterplot(data=data, x='Total_Spending', y='Purchase_Frequency', hue='Cluster', palette='viridis')

plt.title('Customer Segmentation Clusters')

plt.xlabel('Total Spending')

plt.ylabel('Purchase Frequency')

plt.legend(title='Cluster')

plt.show()
```

Deliverables Summary

-Number of Clusters Formed: The optimal number of clusters identified (e.g., 4).

-DB Index Value: The calculated Davies-Bouldin Index value (e.g., `db_index`).

- Other Relevant Clustering Metrics: Silhouette score or other metrics calculated.

- Visual Representation: Scatter plot visualizing customer segments.