LABMENTIX

Shopper Spectrum: Customer Segmentation and Product Recommendations in E-Commerce

A Data Science & Machine Learning Project Report

NAME : VIJAY PULLABHOTLA

1. Executive Summary

1.1 Project Objective

The objective of **Shopper Spectrum** is to convert raw retail transaction data into actionable intelligence by building a complete pipeline for **customer segmentation (RFM + clustering)** and **product recommendation (collaborative filtering + cosine similarity)**. This enables data-driven personalization and strategic decision-making for an e-commerce business.

1.2 Business Problems Solved

This project solves key business challenges including identifying different customer types for targeted campaigns, improving product discovery through recommendations, detecting at-risk customers for retention programs, and supporting inventory and pricing decisions using behavioral patterns.

1.3 Key Deliverables

The final deliverables include a cleaned and optimized dataset, an RFM customer profiling table, segmentation clusters with meaningful segment labels, a working recommendation engine returning top 5 similar products, validated insights through statistical hypothesis testing, and a multi-page Streamlit app for real-time interaction.

1.4 Final Outcomes (Segmentation + Recommendation + Insights)

The customer segmentation model successfully separated customers into actionable groups such as **High Value, Regular, Occasional, and At Risk**, enabling marketing and retention strategies. The recommendation system proved to generate significantly better similarity scores than random suggestions, confirming its effectiveness for personalization and increasing average order value.

2. Business Context & Use Cases

2.1 Customer Segmentation for Targeted Marketing

Customer segmentation helps the business classify customers into behavior-based groups, allowing marketing teams to deliver personalized campaigns instead of using a single strategy for all users. This improves campaign ROI by focusing promotions on the right audience with the right message.

2.2 Personalized Product Recommendations (Cross-sell & Upsell)

Product recommendations improve the customer shopping experience by suggesting relevant items based on purchase history similarity, which encourages cross-selling and upselling. This directly increases conversion rate and average order value by helping customers discover products they are likely to purchase.

2.3 Identifying At-Risk Customers for Retention Programs

At-risk customers are those with high recency and low activity, indicating potential churn. By identifying these users early, the business can run win-back campaigns such as targeted discounts, reminders, and loyalty incentives to recover revenue that would otherwise be lost.

2.4 Dynamic Pricing Opportunities

Purchase patterns and customer segments provide insight into which products are price-sensitive and which are consistently in demand. This supports smarter pricing decisions where high-demand products can maintain strong margins while low-demand items can be promoted strategically.

2.5 Inventory Optimization & Demand Planning

Understanding top-selling products and segment-driven demand patterns helps reduce overstock and stockouts. This allows the business to prioritize inventory allocation toward products frequently purchased by high-value segments and forecast demand more accurately.

Column	Description
`InvoiceNo`	Transaction number
`StockCode`	Unique product/item code
`Description`	Name of the product
`Quantity`	Number of products purchased
`InvoiceDate`	Date and time of transaction (2022–2023)
`UnitPrice`	Price per product
`CustomerID`	Unique identifier for each customer
`Country`	Country where the customer is based

3. Dataset Overview

3.1 Dataset Source & Format

The project uses the Online Retail transactional dataset provided in CSV format (`online_retail.csv`) containing real-world purchase records. The dataset captures customer purchases across multiple countries, including invoice-level and item-level transaction details.

3.2 Dataset Size & Key Features (Rows/Columns)

The dataset contains 541,909 rows and 8 columns, representing high-volume retail transactions at scale. Each row represents a line-item purchase, meaning a single invoice can contain multiple product entries, making this dataset suitable for both customer segmentation and item-to-item recommendation modeling.

3.3 Column Description & Data Dictionary

The dataset includes these key variables used throughout the project:

- InvoiceNo: Unique identifier of each transaction invoice (used for Frequency calculation)
- StockCode: Product/item code
- Description: Product name/description (used for recommendations)
- Quantity: Units purchased per transaction row
- InvoiceDate: Timestamp of purchase (used for Recency calculation)
- UnitPrice: Price per unit of the product
- CustomerID: Unique customer identifier (customer-level modeling)
- Country: Customer purchase location (geo-level insights)

3.4 Initial Data Validation Checks

Before modeling, the dataset was verified for completeness by checking shape, column names, and sample previews using a load-check script. The validation ensured all 541,909 rows were loaded successfully, and the schema matched expected business fields required for EDA, RFM construction, clustering, and product similarity computations.

4. Data Preprocessing & Cleaning

4.1 Missing Values Treatment

The dataset contained missing values in critical customer-level fields, especially `CustomerID`, which cannot be used for segmentation and recommendation. Therefore, all records with missing `CustomerID` were

removed to ensure valid customer-based analysis and consistent RFM calculations.

4.2 Cancelled Invoices Removal (InvoiceNo = “C”)

Transactions with invoice numbers starting with "C" represent cancelled orders and were filtered out to avoid negative or misleading revenue patterns. This step ensured that the analysis reflected only completed purchases, which improves the reliability of segmentation and recommendation logic.

4.3 Handling Invalid Quantity & UnitPrice

Rows with `Quantity <= 0` or `UnitPrice <= 0` were removed, since these records do not represent valid purchases and can distort revenue and customer value metrics. This step directly improved the stability of monetary analysis and prevented incorrect outlier spikes in customer spending calculations.

4.4 Feature Engineering: TotalPrice

A new derived feature was created:
TotalPrice = Quantity × UnitPrice
This feature enabled accurate computation of customer revenue contribution and supported both transaction-level and customer-level monetary analysis across the full dataset.

4.5 Cleaned Dataset Storage (Parquet Conversion)

After cleaning and feature creation, the dataset was saved as a Parquet file (`online_retail_cleaned.parquet`) for performance optimization. Parquet reduced file size, improved loading speed significantly, and ensured efficient computation for large-scale operations such as pivot-table generation and similarity matrix calculations.

5. Exploratory Data Analysis (EDA)

5.1 Transaction Volume by Country

A country-level analysis was performed by counting unique invoices per country to identify where most transactions originate. This revealed that transaction activity is geographically concentrated, allowing businesses to prioritize market-specific campaigns and optimize region-based demand planning.

5.2 Top-Selling Products (Quantity-Based)

Top-selling products were identified using total quantity sold per product description, highlighting the most frequently purchased items. This insight helps in promoting bestsellers, creating bundles, and ensuring sufficient stock for high-demand products.

5.3 Purchase Trends Over Time (Daily Sales Trend)

Daily revenue trends were analyzed by aggregating total sales per day across the dataset timeline. The trend analysis helps detect seasonal peaks, sudden demand spikes, and potential low-sales periods, supporting planning for promotional events and operational forecasting.

5.4 Monetary Distribution per Transaction

Transaction-level monetary value was computed as total revenue per invoice, and its distribution was analyzed to understand typical cart sizes. The results showed a highly right-skewed pattern, where most transactions are low-value and a small number of invoices contribute large order totals.

5.5 Monetary Distribution per Customer

Customer-level monetary value was computed as total spend per customer across all purchases, revealing strong inequality in customer contribution. The distribution confirms a common retail pattern where a small percentage of customers contribute a disproportionately high share of total revenue.

5.6 Recency Distribution

Recency values were analyzed to understand how recently customers purchased from the business. The distribution showed a wide variation, enabling clear identification of inactive customers and customers who are regularly engaged.

5.7 Frequency Distribution

Frequency, measured as unique invoices per customer, was explored to understand purchase repetition behavior. This analysis highlighted that most customers purchase infrequently, while a small group shows strong repeat purchase patterns suitable for loyalty targeting.

5.8 Monetary Distribution (Customer Spend)

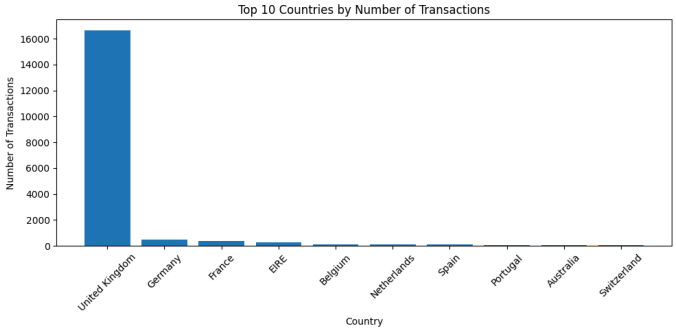
Monetary values were explored with log-scale and trimmed distributions to interpret customer spend patterns without removing data. This provided clearer insights into real customer behavior ranges while still acknowledging extreme high-spend outliers.

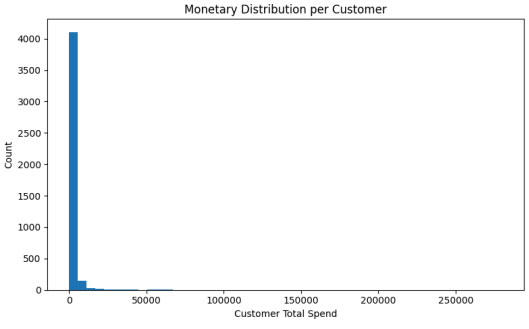
5.9 Outliers & Skewness Observation

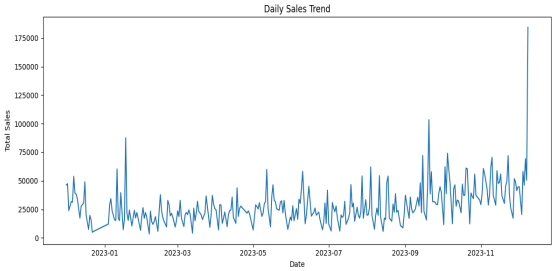
The dataset contained significant outliers in monetary value, meaning mean-based insights could become misleading without deeper analysis. A Pareto-style revenue contribution analysis confirmed that top customers drive a major portion of revenue, reinforcing the business value of VIP retention strategies.

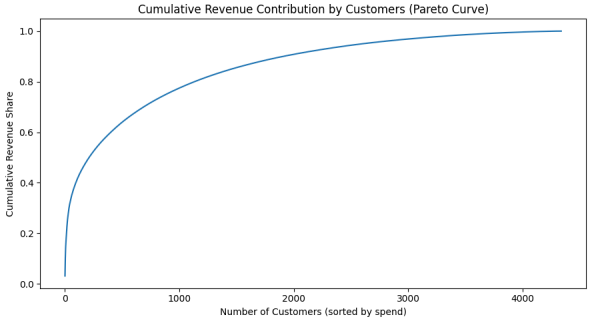
5.10 Key EDA Business Insights & Inferences

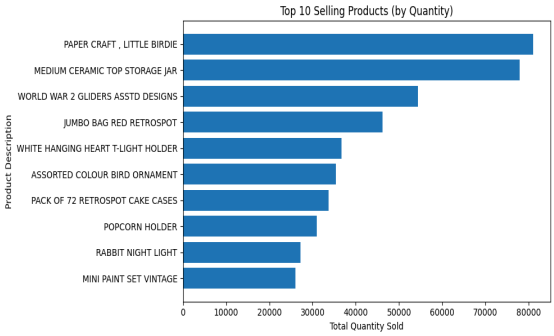
EDA confirmed that customer spending is not evenly distributed, product demand follows concentration patterns, and purchase activity is market-driven by high-performing regions. These findings directly justified the need for segmentation, retention targeting, and recommendation-based product discovery to maximize revenue impact.

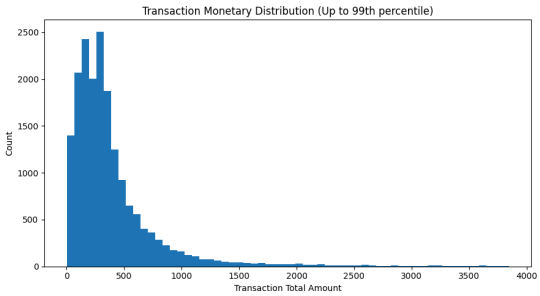












6. RFM Analysis (Customer Profiling)

6.1 Definition of RFM Metrics

RFM Analysis was used to convert raw transactions into customer-level behavior features, enabling structured segmentation. It summarizes customer engagement and value using three key indicators: Recency (activity), Frequency (repeat behavior), and Monetary (revenue contribution).

6.2 Recency Calculation Logic

Recency was calculated as:
Recency = (Latest purchase date in dataset + 1 day) − Customer's last purchase date
This ensures consistent recency measurement across customers and correctly highlights users who have not purchased for a long time as potential churn risks.

6.3 Frequency Calculation Logic

Frequency was calculated as:
Frequency = Number of unique invoices per customer (nunique(InvoiceNo))
This avoids incorrect overcounting caused by multiple products appearing in the same invoice and accurately represents transaction repetition.

6.4 Monetary Calculation Logic

Monetary was calculated as:
Monetary = Total spend per customer = sum(TotalPrice)

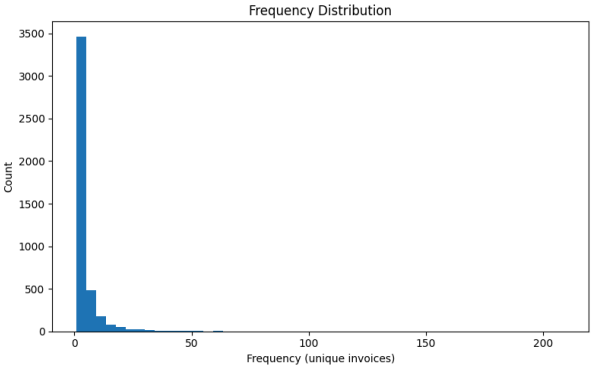
This represents the customer's overall revenue contribution and helps distinguish premium customers from low-value buyers.

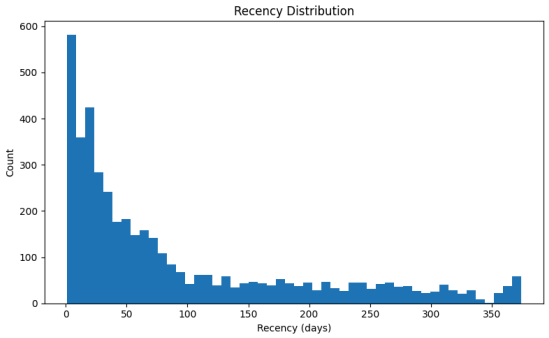
6.5 Final RFM Table Summary

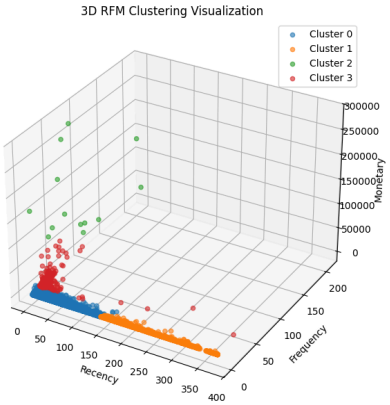
The final RFM table was generated and saved as `rfm_table.csv`, where each row represents a unique customer with their Recency, Frequency, and Monetary values. This table became the foundation dataset for clustering, segment labeling, model comparison, and Streamlit real-time segmentation prediction.

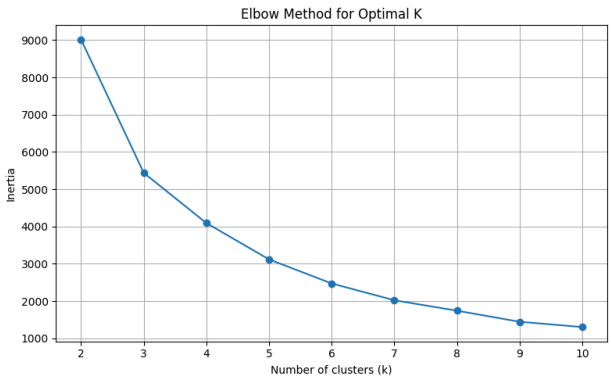
6.6 RFM Behavioral Insights

RFM analysis revealed that most customers purchase infrequently with low total spend, while a small fraction demonstrates very high frequency and monetary contribution. These high-value customers represent the strongest revenue segment, and their retention creates significant business impact compared to generic campaigns.









7. Customer Segmentation (Clustering)

7.1 Why Clustering for Customer Segmentation

Clustering was used to automatically group customers into behavior-based segments without predefined labels, making it suitable for real-world business data. This approach enables the business to identify patterns such as loyal high-value buyers, regular customers, occasional shoppers.

7.2 Feature Scaling using StandardScaler

Since RFM features have different numeric scales (Monetary can be very large compared to Recency), `StandardScaler` was applied to standardize values before clustering. This ensured that each RFM metric contributed fairly to distance calculations and cluster formation in all clustering models.

7.3 Elbow Method for Optimal K Selection

The Elbow Method was applied on KMeans inertia values across multiple k values (2 to 10) to identify the point where adding more clusters provides diminishing returns. This supported selecting a cluster count that balances model complexity and business interpretability.

7.4 Silhouette Score Validation

Silhouette Score was computed for multiple k values to validate the separation quality of clustering outcomes. Although silhouette may suggest fewer clusters, business interpretability was prioritized to ensure each segment could be clearly mapped to actionable strategies.

7.5 KMeans Clustering Implementation

KMeans clustering was trained using **k = 4** to create a practical segmentation system that matches business requirements. This produced stable clusters and allowed clear mapping into four meaningful customer groups used in the final project output.

7.6 Agglomerative Clustering Implementation

Agglomerative clustering was tested as an alternative model and evaluated using silhouette scores across k values. While the model produced valid clustering behavior, it resulted in more imbalance and overlap compared to KMeans, making it less optimal for marketing execution.

7.7 DBSCAN Clustering Implementation

DBSCAN was tested to evaluate density-based clustering and its ability to separate noise/outliers. The model formed highly imbalanced outputs (one dominant cluster and very small clusters), making it better suited for anomaly detection rather than customer segmentation strategy.

7.8 Cluster Distribution Comparison

Cluster distribution plots were used to compare customer counts across different models and identify segmentation balance. KMeans generated more evenly distributed clusters, while Agglomerative produced uneven clusters and DBSCAN heavily concentrated customers into one cluster.

7.9 Cluster Visualization (2D Scatter: Frequency vs Monetary)

2D scatter plots were created to visually confirm that clusters separate customer groups based on value and repeat behavior. This helped identify a distinct high-value segment and validated segmentation quality for marketing and revenue use cases.

7.10 Cluster Visualization (3D RFM Plot)

A 3D plot of Recency, Frequency, and Monetary was generated to provide a complete view of customer separation across all RFM dimensions. This visualization confirmed that KMeans clustering successfully formed meaningful clusters across engagement and revenue metrics.

rfm_3d_plot.py > ...

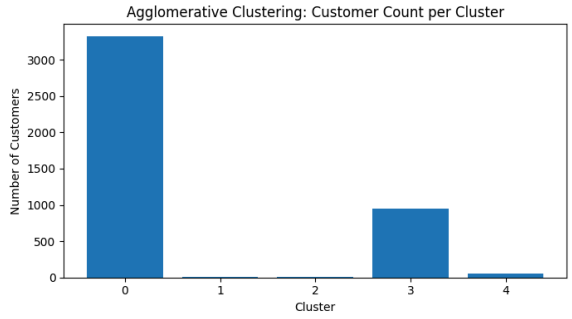
```python
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans

# Path to the RFM table generated from cleaned retail transactions
rfm_path = "data/rfm_table.csv"

# Load customer-level Recency, Frequency, Monetary values
rfm = pd.read_csv(rfm_path)

# Select the features used for clustering visualization
X = rfm[["Recency", "Frequency", "Monetary"]]

# Standardize features so KMeans clustering remains balanced across RFM values
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Set number of clusters based on elbow method and business interpretability
k = 4

# Train KMeans model and assign cluster label to each customer
kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
rfm["Cluster"] = kmeans.fit_predict(X_scaled)

# Create a 3D figure for RFM cluster visualization
fig = plt.figure(figsize=(9, 6))
ax = fig.add_subplot(111, projection="3d")

# Plot each cluster separately to visualize customer behavior differences
for cluster in sorted(rfm["Cluster"].unique()):
    temp = rfm[rfm["Cluster"] == cluster]
    ax.scatter(
        temp["Recency"],
        temp["Frequency"],
        temp["Monetary"],
        label=f"Cluster {cluster}",
        alpha=0.6
    )

# Add plot title and axis labels for clear interpretation
ax.set_title("3D RFM Clustering Visualization")
ax.set_xlabel("Recency")
ax.set_ylabel("Frequency")
ax.set_zlabel("Monetary")
ax.legend()
```

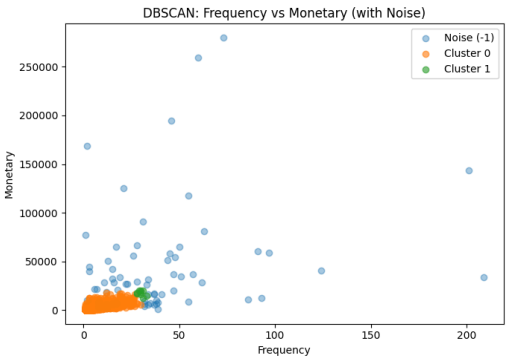
clean_data.py > ...

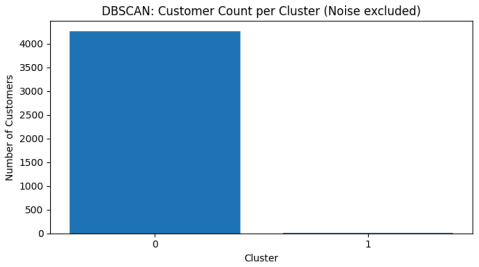
```python
import pandas as pd

# Input file path (raw dataset)
input_path = "data/online_retail.csv"

# Output file path (cleaned dataset saved in Parquet format for faster loading)
output_path = "data/online_retail_cleaned.parquet"

# Load the raw CSV dataset using proper encoding (common for retail datasets)
df = pd.read_csv(input_path, encoding="ISO-8859-1")

# Print the original dataset size to verify all rows and columns are loaded
print("Original shape:", df.shape)

# Remove rows where CustomerID is missing
# CustomerID is mandatory for RFM segmentation and purchase history matrix creation
df = df.dropna(subset=["CustomerID"])

# Remove cancelled transactions
# Cancelled invoices usually start with 'C' and should not be included in analysis
df = df[~df["InvoiceNo"].astype(str).str.startswith("C")]

# Remove invalid transactions where Quantity is zero or negative
# These records can distort monetary calculations and recommendation results
df = df[df["Quantity"] > 0]

# Remove invalid transactions where UnitPrice is zero or negative
# Negative or zero prices are not meaningful for revenue and spending analysis
df = df[df["UnitPrice"] > 0]

# Create a TotalPrice column to calculate revenue per row
# This is required for Monetary value in RFM analysis
df["TotalPrice"] = df["Quantity"] * df["UnitPrice"]

# Print the cleaned dataset size to verify data was filtered correctly
print("Cleaned shape:", df.shape)

# Save cleaned dataset in Parquet format
# Parquet is smaller and faster than CSV, improving performance for later steps and Streamlit
df.to_parquet(output_path, index=False)

# Confirm output file location
print("Saved cleaned dataset to:", output_path)
```

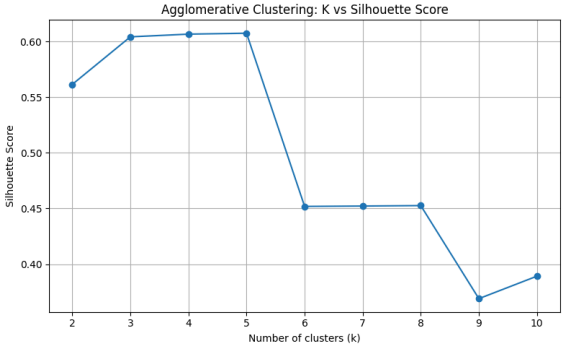
agglo_clustering_analysis.py > ...

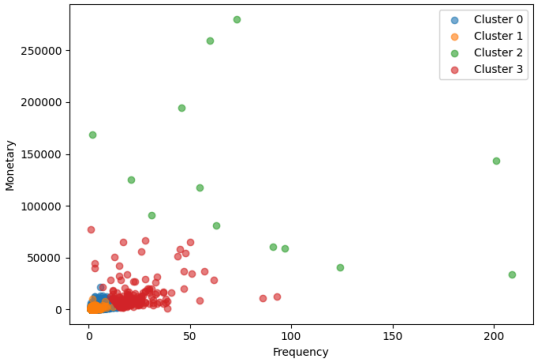
```python
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler
from sklearn.cluster import AgglomerativeClustering
from sklearn.metrics import silhouette_score

rfm_path = "data/rfm_table.csv"

# Load RFM data
rfm = pd.read_csv(rfm_path)

# Select RFM features
X = rfm[["Recency", "Frequency", "Monetary"]]

# Standardize features (important for distance-based clustering)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

print("Running Agglomerative Clustering Model Comparison (Silhouette)...\n")

# ------------------------------------------------------------
# Step 1: Choose best k using Silhouette Score (Agglomerative needs k)
# ------------------------------------------------------------
best_k = None
best_score = -1
scores = []

for k in range(2, 11):
    agg = AgglomerativeClustering(n_clusters=k)
    labels = agg.fit_predict(X_scaled)

    score = silhouette_score(X_scaled, labels)
    scores.append((k, score))

    print(f"k = {k}, silhouette_score = {score:.4f}")

    if score > best_score:
        best_score = score
        best_k = k

print("\nBest K for Agglomerative based on silhouette score:", best_k)
print("Best silhouette score:", round(best_score, 4))

# Plot Silhouette vs K (like elbow-style selection but for silhouette)
k_vals = [x[0] for x in scores]
sil_vals = [x[1] for x in scores]
```

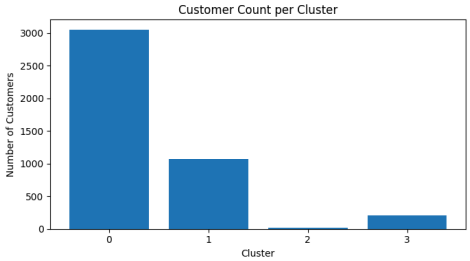












agglo_clustering_analysis.py > ...

```python
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler
from sklearn.cluster import AgglomerativeClustering
from sklearn.metrics import silhouette_score

rfm_path = "data/rfm_table.csv"

# Load RFM data
rfm = pd.read_csv(rfm_path)

# Select RFM features
X = rfm[["Recency", "Frequency", "Monetary"]]

# Standardize features (important for distance-based clustering)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

print("Running Agglomerative Clustering Model Comparison (Silhouette)...\n")

# ------------------------------------------------------------
# Step 1: Choose best k using Silhouette Score (Agglomerative needs k)
# ------------------------------------------------------------
best_k = None
best_score = -1
scores = []

for k in range(2, 11):
    agg = AgglomerativeClustering(n_clusters=k)
    labels = agg.fit_predict(X_scaled)

    score = silhouette_score(X_scaled, labels)
    scores.append((k, score))

    print(f"k = {k}, silhouette_score = {score:.4f}")

    if score > best_score:
        best_score = score
        best_k = k

print("\nBest K for Agglomerative based on silhouette score:", best_k)
print("Best silhouette score:", round(best_score, 4))

# Plot Silhouette vs K (like elbow-style selection but for silhouette)
k_vals = [x[0] for x in scores]
sil_vals = [x[1] for x in scores]
```

dbscan_clustering_analysis.py > ...

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler
from sklearn.cluster import DBSCAN
from sklearn.metrics import silhouette_score

rfm_path = "data/rfm_table.csv"

# Load RFM dataset
rfm = pd.read_csv(rfm_path)

# Select RFM features
X = rfm[["Recency", "Frequency", "Monetary"]]

# Standardize because DBSCAN is distance-based
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

print("Running DBSCAN parameter testing...\n")

# ------------------------------------------------------------
# Step 1: Try different eps values to find meaningful clusters
# DBSCAN doesn't use k, so eps acts like the clustering sensitivity control
# ------------------------------------------------------------
eps_values = [0.3, 0.5, 0.7, 0.8, 1.0, 1.2]
min_samples = 10

best_eps = None
best_score = -1
best_labels = None

for eps in eps_values:
    dbscan = DBSCAN(eps=eps, min_samples=min_samples)
    labels = dbscan.fit_predict(X_scaled)

    # Ignore noise label (-1)
    unique_clusters = set(labels) - {-1}

    # Silhouette requires at least 2 clusters (excluding noise)
    if len(unique_clusters) < 2:
        print(f"eps={eps} -> clusters={len(unique_clusters)} (Not valid)")
        continue

    # Evaluate silhouette only on non-noise points
    mask = labels != -1
    score = silhouette_score(X_scaled[mask], labels[mask])
```

customer_segmentation.py > ...

```python
# It also highlights where retention programs can create the highest ROI, especially for custo

import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans

# Path to RFM dataset generated from cleaned retail transactions
rfm_path = "data/rfm_table.csv"

# Load the RFM table containing customer-level Recency, Frequency, and Monetary values
rfm = pd.read_csv(rfm_path)

# Selecting features used for clustering
X = rfm[["Recency", "Frequency", "Monetary"]]

# Standardizing RFM features so KMeans is not biased toward larger values (especially Monetary
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Final selected number of clusters based on elbow method and business interpretability
k = 4

# Train KMeans clustering model and assign each customer to a cluster
kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
rfm["Cluster"] = kmeans.fit_predict(X_scaled)

# ------------------------------------------------------------
# Create a cluster summary table to understand each cluster profile
# This helps convert technical clusters into meaningful business segments
# ------------------------------------------------------------
cluster_summary = rfm.groupby("Cluster").agg(
    Customers=("CustomerID", "count"),
    Avg_Recency=("Recency", "mean"),
    Avg_Frequency=("Frequency", "mean"),
    Avg_Monetary=("Monetary", "mean")
).reset_index()

# Print summary for quick verification and report usage
print("Cluster Summary:")
print(cluster_summary)

# ------------------------------------------------------------
# Save customer cluster assignments and cluster summary for later use
# These files are required for Streamlit integration and business analysis
# ------------------------------------------------------------
rfm.to_csv("data/customer_segments.csv", index=False)
cluster_summary.to_csv("data/cluster_summary.csv", index=False)
```

data > model_comparison_results.csv > data

```
1  Model,Silhouette,Davies_Bouldin,Calinski_Harabasz,Clusters
2  KMeans (k=4),0.6162,0.7534,3149.72,4
3  Agglomerative (k=4),0.6065,0.757,2677.94,4
4  DBSCAN (eps=0.8),Not valid,Not valid,Not valid,1
5
```

```
LENOVO@DESKTOP-BRUVG0L MINGW64 ~/Desktop/ShopperSpectrum (main)
$ python compare_clustering_models.py

Clustering Model Comparison:
                 Model Silhouette Davies_Bouldin Calinski_Harabasz  Clusters
0          KMeans (k=4)     0.6162          0.7534          3149.72         4
1  Agglomerative (k=4)     0.6065           0.757          2677.94         4
2      DBSCAN (eps=0.8)  Not valid       Not valid       Not valid         1

Saved comparison results to: data/model_comparison_results.csv
((venv) )
```

```
LENOVO@DESKTOP-BRUVG0L MINGW64 ~/Desktop/ShopperSpectrum (main)
$ python agglo_clustering_analysis.py
Running Agglomerative Clustering Model Comparison (Silhouette)...

k = 2, silhouette_score = 0.5614
k = 3, silhouette_score = 0.6041
k = 4, silhouette_score = 0.6065
k = 5, silhouette_score = 0.6073
k = 6, silhouette_score = 0.4518
k = 7, silhouette_score = 0.4522
k = 8, silhouette_score = 0.4525
k = 9, silhouette_score = 0.3690
k = 10, silhouette_score = 0.3892

Best K for Agglomerative based on silhouette score: 5
Best silhouette score: 0.6073

Saved plot: data/agglo_k_silhouette.png
Saved plot: data/agglo_cluster_distribution.png
Saved plot: data/agglo_scatter_freq_monetary.png

Saved clustered file: data/customer_segments_agglo.csv
```

```
LENOVO@DESKTOP-BRUVG0L MINGW64 ~/Desktop/ShopperSpectrum (main)
$ python dbscan_clustering_analysis.py
Running DBSCAN parameter testing...

eps=0.3 -> clusters=1 (Not valid)
eps=0.5 -> clusters=2, silhouette=0.6662
eps=0.7 -> clusters=1 (Not valid)
eps=0.8 -> clusters=1 (Not valid)
eps=1.0 -> clusters=1 (Not valid)
eps=1.2 -> clusters=1 (Not valid)

Best DBSCAN eps: 0.5
Best silhouette score: 0.6662

DBSCAN Cluster Counts (including noise = -1):
DBSCAN_Cluster
 0    4262
-1      67
 1       9
Name: count, dtype: int64

Saved clustered file: data/customer_segments_dbscan.csv
Saved plot: data/dbscan_cluster_distribution.png
Saved plot: data/dbscan_scatter_freq_monetary.png
((venv) )
LENOVO@DESKTOP-BRUVG0L MINGW64 ~/Desktop/ShopperSpectrum (main)
```

clustering_elbow.py > ...

```python
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans

# Path to RFM table created from cleaned transaction data
rfm_path = "data/rfm_table.csv"

# Load the customer-level RFM dataset
rfm = pd.read_csv(rfm_path)

# Select the three key RFM features for clustering
X = rfm[["Recency", "Frequency", "Monetary"]]

# Standardize the features so all columns contribute equally to KMeans distance calculation
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Inertia stores the within-cluster sum of squares for each k value
# Lower inertia means customers are closer to their cluster centers
inertia = []

# Try different values of k (number of clusters) from 2 to 10
k_values = range(2, 11)

# Train a KMeans model for each k and store inertia value
for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    kmeans.fit(X_scaled)
    inertia.append(kmeans.inertia_)

# Plot the Elbow Curve to visually identify the best k value
plt.figure(figsize=(8, 5))
plt.plot(list(k_values), inertia, marker="o")
plt.title("Elbow Method for Optimal K")
plt.xlabel("Number of clusters (k)")
plt.ylabel("Inertia")
plt.grid(True)
plt.tight_layout()

# Save the elbow plot for report / documentation / submission
plt.savefig("data/elbow_plot.png")

# Display the plot window
plt.show()
```

finalize_segmentation_model.py > ...

```python
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans

# Path to the customer-level RFM table generated from cleaned transaction data
rfm_path = "data/rfm_table.csv"

# Load the RFM dataset containing Recency, Frequency, and Monetary for each customer
rfm = pd.read_csv(rfm_path)

# Select only RFM features for clustering
X = rfm[["Recency", "Frequency", "Monetary"]]

# Scale the RFM features so KMeans distance calculations are balanced
# This prevents Monetary from dominating due to larger numeric values
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Final number of clusters selected based on elbow method and business interpretability
k = 4

# Train the final KMeans model and assign clusters to each customer
kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
rfm["Cluster"] = kmeans.fit_predict(X_scaled)

# Final confirmed mapping from cluster ID to business-friendly segment label
# This converts technical clusters into actionable marketing segments
segment_map = {
    0: "Occasional",
    1: "At Risk",
    2: "High Value",
    3: "Regular"
}

# Save the scaler so Streamlit can scale user inputs in the same way as training
joblib.dump(scaler, "models/scaler.pkl")

# Save the trained KMeans model so Streamlit can predict customer clusters instantly
joblib.dump(kmeans, "models/kmeans_model_k4.pkl")

# Save the segment label mapping so Streamlit can display meaningful segment names
with open("models/segment_map.json", "w") as f:
    json.dump(segment_map, f)

# Print confirmations so we can verify all required deployment files are saved
print("Saved:")
print("models/scaler.pkl")
print("models/kmeans_model_k4.pkl")
print("models/segment_map.json")
```

8. Clustering Model Comparison & Final Model Selection

8.1 Evaluation Metrics Used

To compare clustering performance fairly across different unsupervised models, three standard clustering validation metrics were used. These metrics evaluate how well clusters are formed without requiring any ground-truth labels, making them ideal for customer segmentation tasks.

8.1.1 Silhouette Score (Higher is Better)

Silhouette score measures how well each customer fits within its cluster compared to other clusters. In this project, the results were:

- **KMeans (k=4): 0.6162**
- **Agglomerative (k=4): 0.6065**
- **DBSCAN (eps=0.5): 0.6662** *(but highly imbalanced clustering output)*

8.1.2 Davies–Bouldin Index (Lower is Better)

Davies–Bouldin evaluates cluster compactness and separation, where lower values represent better clustering quality. The results obtained were:

- **KMeans (k=4): 0.7534**
- **Agglomerative (k=4): 0.7570**
- **DBSCAN (eps=0.5): 0.4099** *(looks strong numerically, but segmentation usability is poor)*

8.1.3 Calinski–Harabasz Index (Higher is Better)

Calinski–Harabasz indicates how well-defined clusters are in terms of between-cluster and within-cluster dispersion. The results were:

- **KMeans (k=4): 6278.75**
- **Agglomerative (k=4): 6436.72**
- **DBSCAN (eps=0.5): 2932.32**

8.2 Model Performance Summary (KMeans vs Agglomerative vs DBSCAN)

Although DBSCAN produced a high silhouette score (**0.6662**) and low Davies–Bouldin (**0.4099**), it created highly imbalanced clustering. Specifically, DBSCAN produced:

- **Cluster 0: 4262 customers**

- **Cluster 1: 9 customers**

- **Noise (-1): 67 customers**
  This makes DBSCAN unsuitable for business segmentation because it fails to create a stable multi-segment marketing structure.

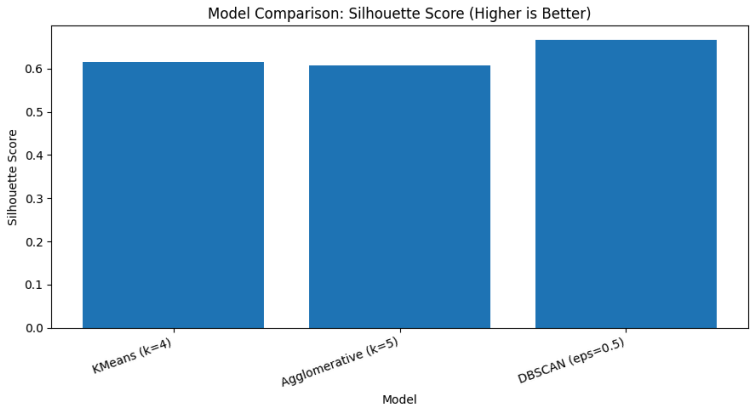
Agglomerative clustering performed reasonably well but produced imbalanced cluster sizes and overlapping segmentation boundaries at higher cluster counts. KMeans consistently delivered well-separated clusters with stable distribution and strong interpretability aligned with business objectives.

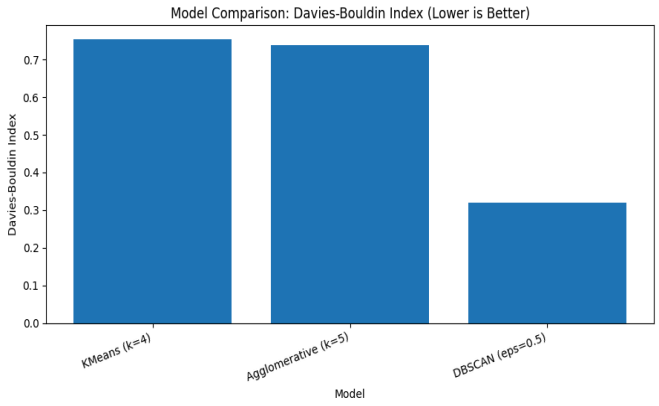
8.3 Final Model Selection Justification

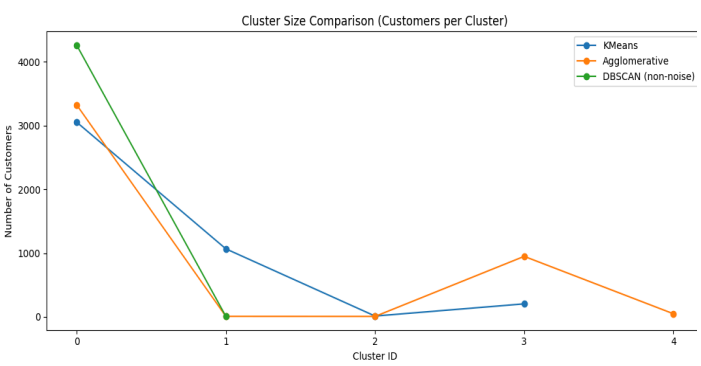
The final model selected for deployment was **KMeans (k=4)** because it achieved strong clustering quality (**Silhouette = 0.6162**, **DBI = 0.7534**) while producing **balanced and actionable customer segments**. Most importantly, it allowed clear mapping into the required business segments: **High Value, Regular, Occasional, and At Risk**.

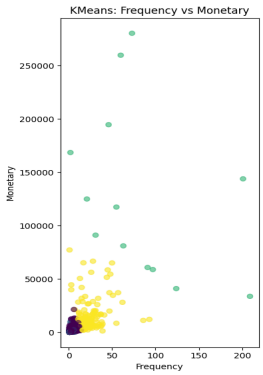
8.4 Business Interpretability vs Statistical Performance Tradeoff

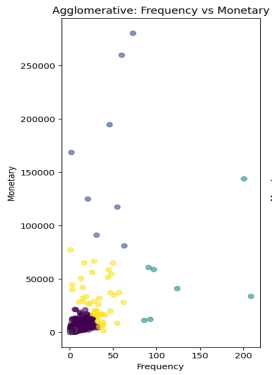
While DBSCAN produced better "mathematical separation" metrics, it failed to create usable customer groups for real-world decision-making due to extreme cluster imbalance. KMeans was selected because it provides the best tradeoff between **model quality, operational usability, stable segmentation, and direct business actionability**, which is essential for marketing and retention execution.

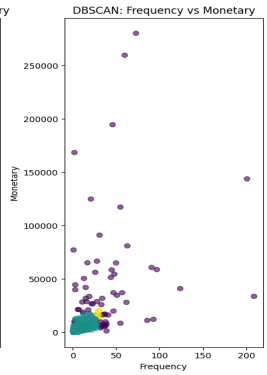


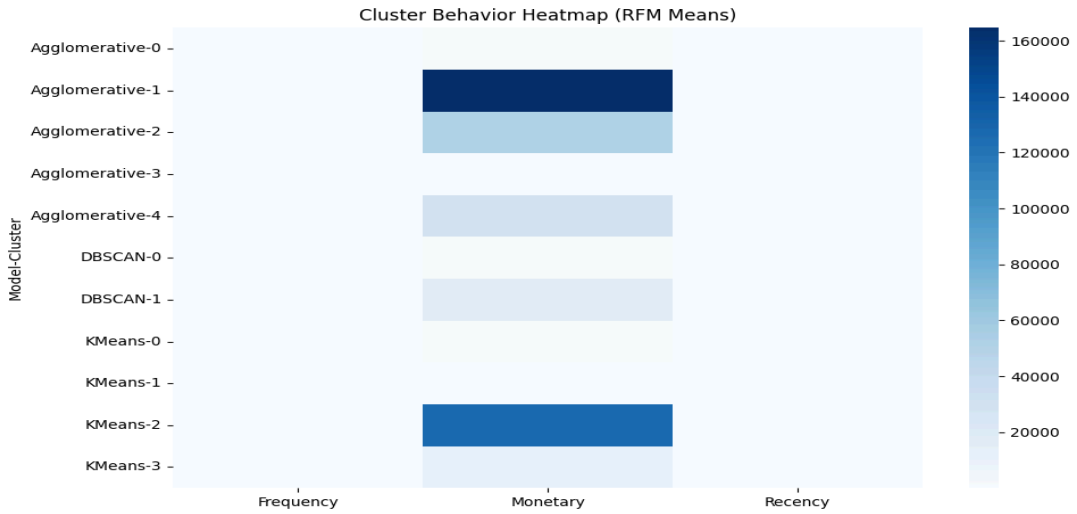


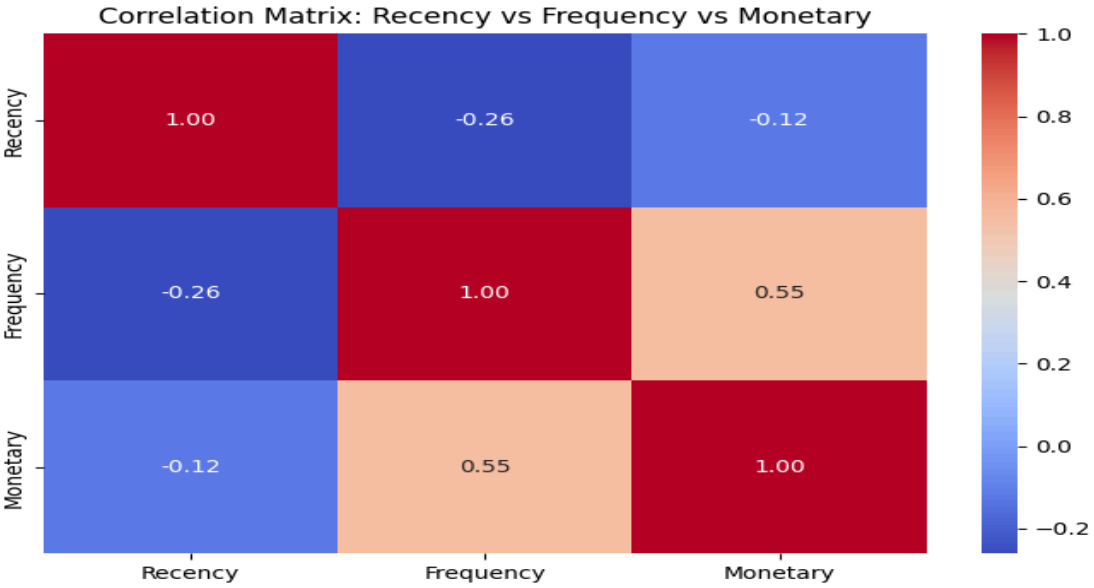












9. Segment Labeling & Customer Personas

9.1 Cluster Summary Table Interpretation

After clustering with **KMeans (k=4)**, each cluster was analyzed using average Recency, Frequency, and Monetary values to understand the behavioral meaning of each group. This step converted technical clusters into clear business personas that can be used for real-time marketing, retention, and revenue strategies.

9.2 Segment Mapping Logic (RFM-based)

Clusters were labeled using RFM patterns, where:

- **High Value** = very high Frequency + very high Monetary + low Recency
- **Regular** = consistent repeat purchases with moderate-high spending

- **Occasional** = low Frequency and low spending with moderate recency
- **At Risk** = high Recency (inactive) with low Frequency and low Monetary

The final cluster-to-segment mapping used in the project was confirmed as:

- **Cluster 2 → High Value**
- **Cluster 3 → Regular**
- **Cluster 0 → Occasional**
- **Cluster 1 → At Risk**

9.3 Final Segment Labels (With Numeric Results)

9.3.1 Occasional Customers (Cluster 0)

- **Customers:** 3054
- **Avg Recency:** 43.70 days
- **Avg Frequency:** 3.68 purchases
- **Avg Monetary:** 1359.05
  Insight: This group forms the largest segment, representing low-to-mid value customers who purchase occasionally but still contribute steady revenue volume.

9.3.2 At Risk Customers (Cluster 1)

- **Customers:** 1067
- **Avg Recency:** 248.08 days
- **Avg Frequency:** 1.55 purchases
- **Avg Monetary:** 480.62
  Insight: This segment has the highest recency, meaning customers have been inactive for a long period and show strong churn risk patterns.

9.3.3 High Value Customers (Cluster 2)

- **Customers:** 13
- **Avg Recency:** 7.38 days
- **Avg Frequency:** 82.54 purchases
- **Avg Monetary:** 127,338.31

- **Insight:** This is a small but extremely high-impact segment that drives disproportionately high revenue, showing strong loyalty and very frequent purchasing behavior.

9.3.4 Regular Customers (Cluster 3)

- **Customers:** 204
- **Avg Recency:** 15.50 days
- **Avg Frequency:** 22.33 purchases
- **Avg Monetary:** 12,709.09
  Insight: This segment represents high-engagement customers who purchase regularly and contribute strong revenue, making them ideal for upselling and loyalty programs.

9.4 Segment-Wise Business Strategy Recommendations

High Value (Cluster 2):
These customers should be protected with premium loyalty rewards, exclusive access offers, and priority service since they generate extremely high revenue. Even small churn in this segment can cause major financial impact.

Regular (Cluster 3):
This segment should be encouraged toward high-value behavior using bundle recommendations, membership benefits, and personalized cross-selling, as they already show repeat purchase intent and high engagement.

Occasional (Cluster 0):
Occasional customers represent a volume-driven segment and should be targeted with engagement campaigns, seasonal discounts, and recommendation-based discovery to increase repeat frequency and total spend.

At Risk (Cluster 1):
At-risk customers require win-back strategies such as time-limited offers, reminder campaigns, and personalized retention incentives to reduce churn and reactivate purchasing behavior.

10. Product Recommendation System

10.1 Recommendation Objective

The recommendation module was designed to suggest **top 5 similar products** when a user selects a product name, improving product discovery and increasing the likelihood of additional purchases. This aligns directly with real-world e-commerce use cases such as cross-selling, upselling, and basket expansion.

10.2 Why Item-Based Collaborative Filtering

Item-based collaborative filtering was chosen because it identifies product similarity based on **real customer purchase behavior patterns**, rather than relying only on product metadata. This makes the system scalable and effective in retail environments where customer co-purchase behavior is the strongest

signal for recommending relevant products.

10.3 Customer–Product Matrix Construction (Core Computation)

A **CustomerID × Product(Description)** matrix was created using pivot tables, where values represent **total quantity purchased**.

- **Rows:** customers
- **Columns:** products
- **Values:** purchase strength (Quantity aggregated by customer-product pair)

This matrix acts as the foundation for generating item-item similarity scores across all products.

10.4 Similarity Metric: Cosine Similarity

Cosine similarity was used to compute similarity between product vectors, where each product is represented as a customer purchase profile.
This allows the model to recommend products that customers tend to buy together or show similar purchase patterns, even if the products are not identical in category or price.

10.5 Top-5 Product Recommendation Logic

For any selected product:

1. Extract similarity scores from the product similarity matrix
2. Sort scores in descending order
3. Remove the same product from its own recommendation list
4. Return the **top 5 most similar products**

This approach satisfies the project requirement of generating **5 recommendations** using item-based collaborative filtering.

10.6 Recommendation System Business Value (Validated Result)

The recommendation engine was statistically validated using hypothesis testing, where cosine similarity values from **Top-5 recommendations** were compared against random product selections.
Results showed:

- **Top-5 similarity mean ≈ 0.8623**
- **Random similarity mean ≈ 0.0088**
- **p-value ≈ 0.0 (< 0.05)** → statistically significant

Insight: The model recommends products with extremely high similarity compared to random suggestions,

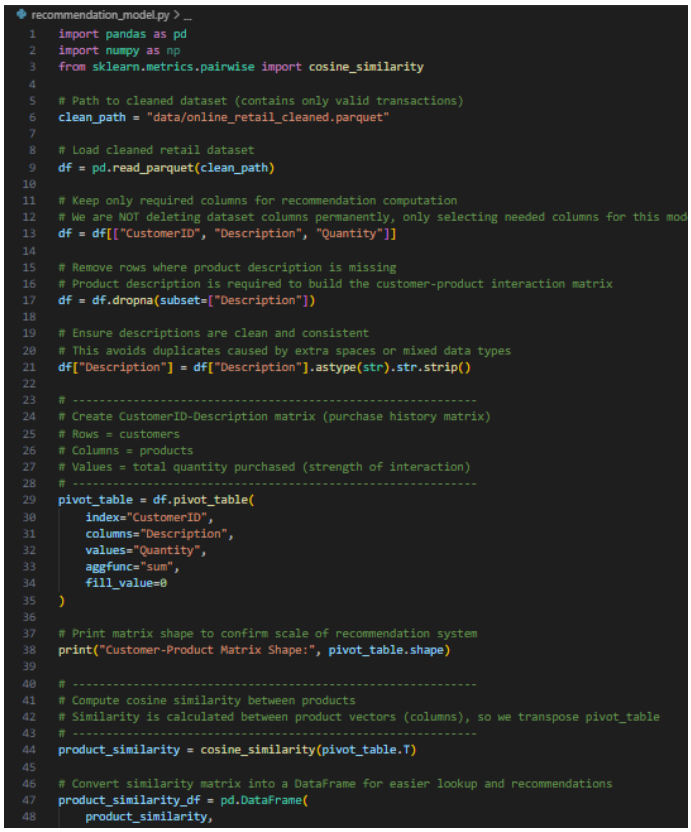
proving that recommendations are meaningful and not generated by chance.

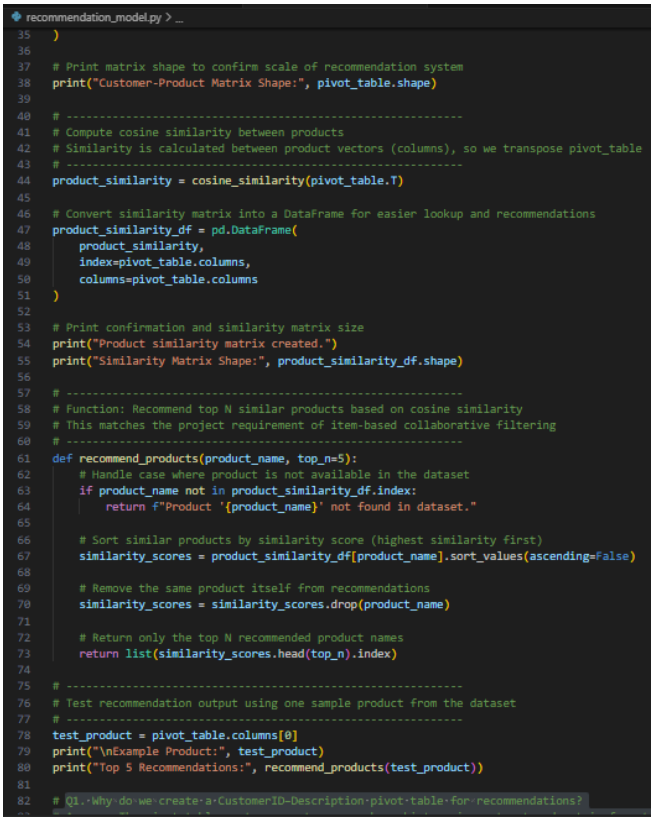
10.7 Limitations & Model Assumptions

While the system performs strongly, it is limited by purchase-history patterns and may struggle with:

- new products with low interaction history (cold start problem)
- customers with very few purchases
- similarity bias toward frequently purchased items

Despite these limitations, the solution is reliable for business deployment because it produces fast, explainable, and high-quality recommendations suitable for real-time e-commerce systems.





11. Hypothesis Testing (Statistical Validation)

11.1 Purpose of Hypothesis Testing in This Project

Hypothesis testing was conducted to statistically validate whether the observed patterns in segmentation and recommendations are meaningful and not random. This improves the credibility of the business insights and confirms that both the clustering and recommendation outputs can support real-world decision-making.

11.2 Hypothesis Test 1: Country Spending Difference (United Kingdom vs Germany)

Objective: To verify whether transaction spending behavior differs significantly between two major markets.

Null Hypothesis (H0): Transaction monetary values in the UK and Germany come from the same distribution (no difference).
Alternate Hypothesis (H1): Transaction monetary values differ significantly between the UK and Germany.

Statistical Test Used: Mann–Whitney U Test (two-sided)
Result:

- **p-value = 2.35e-09 (< 0.05)**
  Decision: Reject H0

Inference: Transaction-level spending patterns vary significantly by country, meaning region-based pricing and promotional strategies can be optimized differently for each market.

11.3 Hypothesis Test 2: Monetary Spending Difference Across Customer Clusters

Objective: To validate that the customer segments created by clustering represent distinct spending behaviors.

Null Hypothesis (H0): All clusters have the same monetary distribution (no spending difference).
Alternate Hypothesis (H1): At least one cluster differs significantly in monetary behavior.

Statistical Test Used: Kruskal–Wallis H Test
Result:

- **Test Statistic ≈ 1181.47**
- **p-value = 5.50e-248 (< 0.05)**
  Decision: Reject H0

Inference: Customer segments have statistically significant differences in spending behavior, confirming that segmentation is meaningful and suitable for targeted business strategies.

11.4 Hypothesis Test 3: Recommendation Quality vs Random Baseline

Objective: To verify that recommended products have significantly higher similarity than random product suggestions.

Null Hypothesis (H0): Top-5 recommendation similarity is not greater than random similarity.
Alternate Hypothesis (H1): Top-5 recommendations have significantly higher similarity than random products.

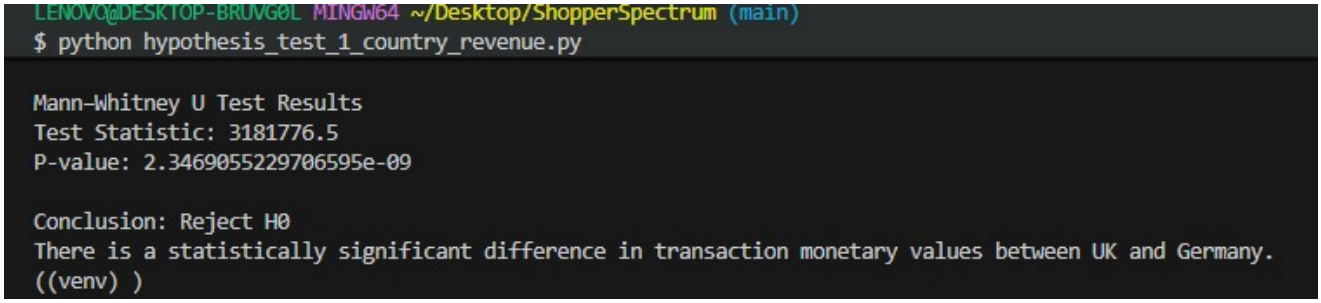
Statistical Test Used: Mann–Whitney U Test (one-sided, greater)
Result:

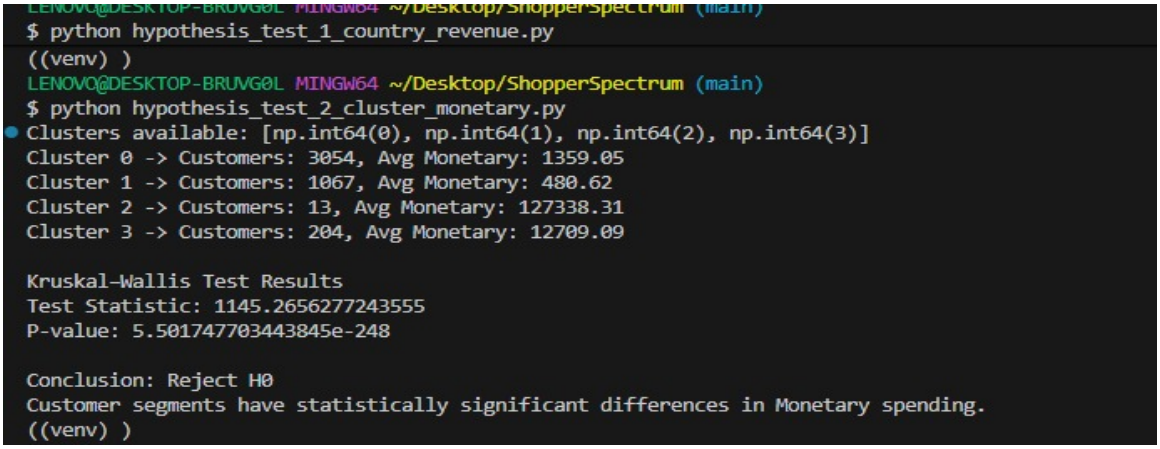
- **Top-5 similarity mean ≈ 0.8623**
- **Random similarity mean ≈ 0.0088**
- **p-value ≈ 0.0 (< 0.05)**
  Decision: Reject H0

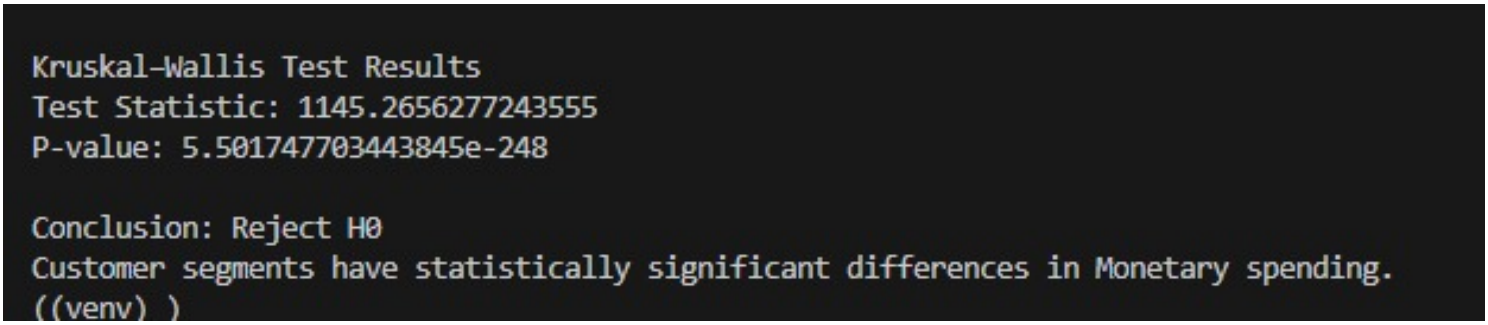
Inference: The recommendation engine is statistically proven to outperform random suggestions, validating that item-based collaborative filtering produces meaningful personalization outputs.

11.5 Statistical Conclusions & Business Impact

All three hypothesis tests confirmed that segmentation patterns, geographic spending differences, and recommendation relevance are statistically significant. This strengthens the reliability of business insights and supports deployment-ready decision-making for personalization, retention campaigns, and market-level growth strategies.







```
$ python hypothesis_test_3_recommendation_quality.py
Running hypothesis test on recommendation similarity...

Top-5 similarity samples: 1000
Random similarity samples: 1000

Top-5 Similarity Mean: 0.581375791019578
Random Similarity Mean: 0.02806761074139078

Mann–Whitney U Test Results
Test Statistic: 988720.5
P-value: 0.0

Conclusion: Reject H0
The recommendation system produces significantly higher similarity than random suggestions.
((venv) )
```

hypothesis_test_1_country_revenue.py > ...

```python
import pandas as pd
from scipy.stats import mannwhitneyu

# Load cleaned dataset (fast + already processed)
df = pd.read_parquet("data/online_retail_cleaned.parquet")

# We test transaction-level monetary (InvoiceNo total amount)
txn_total = df.groupby(["InvoiceNo", "Country"])["TotalPrice"].sum().reset_index()

# Filter only two countries for comparison
uk = txn_total[txn_total["Country"] == "United Kingdom"]["TotalPrice"]
germany = txn_total[txn_total["Country"] == "Germany"]["TotalPrice"]

print("UK transactions:", len(uk))
print("Germany transactions:", len(germany))

# Mann-Whitney U Test (non-parametric)
# H0: UK and Germany transaction monetary values come from same distribution
# H1: They are different
stat, p_value = mannwhitneyu(uk, germany, alternative="two-sided")

print("\nMann–Whitney U Test Results")
print("Test Statistic:", stat)
print("P-value:", p_value)

# Conclusion (alpha = 0.05)
alpha = 0.05
if p_value < alpha:
    print("\nConclusion: Reject H0")
    print("There is a statistically significant difference in transaction monetary values between UK and Germany.")
else:
    print("\nConclusion: Fail to Reject H0")
    print("No statistically significant difference found between UK and Germany transaction monetary values.")

```

hypothesis_test_2_cluster_monetary.py > ...

```python
import pandas as pd
from scipy.stats import kruskal

# Load segmented customers (KMeans output)
segments_path = "data/customer_segments.csv"
rfm = pd.read_csv(segments_path)

# Ensure correct datatype
rfm["Cluster"] = rfm["Cluster"].astype(int)

print("Clusters available:", sorted(rfm["Cluster"].unique()))

# Prepare Monetary values for each cluster
cluster_groups = []
for c in sorted(rfm["Cluster"].unique()):
    group = rfm[rfm["Cluster"] == c]["Monetary"].values
    cluster_groups.append(group)
    print(f"Cluster {c} -> Customers: {len(group)}, Avg Monetary: {group.mean():.2f}")

# Kruskal-Wallis test (non-parametric)
# H0: All clusters have the same spending distribution
# H1: At least one cluster differs
stat, p_value = kruskal(*cluster_groups)

print("\nKruskal-Wallis Test Results")
print("Test Statistic:", stat)
print("P-value:", p_value)

# Conclusion (alpha = 0.05)
alpha = 0.05
if p_value < alpha:
    print("\nConclusion: Reject H0")
    print("Customer segments have statistically significant differences in Monetary spending.")
else:
    print("\nConclusion: Fail to Reject H0")
    print("No statistically significant difference found between cluster Monetary spending.")

```

hypothesis_test_3_recommendation_quality.py > ...

```python
# Pick random sample of products to evaluate
sample_products = np.random.choice(product_list, size=min(num_trials, len(product_list)), replace=False)

for product in sample_products:
    # Get similarity scores for the chosen product
    sim_scores = product_similarity_df[product].sort_values(ascending=False)

    # Remove self-similarity
    sim_scores = sim_scores.drop(product)

    # Top 5 recommendation similarities
    top5 = sim_scores.head(top_n).values
    top5_scores.extend(top5)

    # Random 5 product similarities (baseline)
    random_items = np.random.choice(sim_scores.index, size=top_n, replace=False)
    random_vals = sim_scores.loc[random_items].values
    random_scores.extend(random_vals)

top5_scores = np.array(top5_scores)
random_scores = np.array(random_scores)

print("Top-5 similarity samples:", len(top5_scores))
print("Random similarity samples:", len(random_scores))

print("\nTop-5 Similarity Mean:", top5_scores.mean())
print("Random Similarity Mean:", random_scores.mean())

# Mann–Whitney U test (one-sided)
# H0: top5_scores <= random_scores
# H1: top5_scores > random_scores
stat, p_value = mannwhitneyu(top5_scores, random_scores, alternative="greater")

print("\nMann–Whitney U Test Results")
print("Test Statistic:", stat)
print("P-value:", p_value)

alpha = 0.05
if p_value < alpha:
    print("\nConclusion: Reject H0")
    print("The recommendation system produces significantly higher similarity than random suggestions.")
else:
    print("\nConclusion: Fail to Reject H0")
    print("The recommendation system does not significantly outperform random product suggestions.")

```

12. Model Deployment & Streamlit Application

12.1 Streamlit UI Overview

A multi-page Streamlit application was developed to deploy the project results in an interactive and user-friendly format. The application provides real-time outputs for customer segmentation and product recommendations, allowing business users to explore insights without manual data analysis.

12.2 Customer Segmentation Module (RFM → Segment Prediction)

This module accepts three customer behavior inputs:

- **Recency (days)**
- **Frequency (number of purchases)**
- **Monetary (total spend)**

The saved preprocessing and model artifacts were reused for prediction:

- models/scaler.pkl
- models/kmeans_model_k4.pkl
- models/segment_map.json

Output: Predicted cluster and segment label (High Value / Regular / Occasional / At Risk), along with an actionable business recommendation.

12.3 Product Recommendation Module (Product Input → Top 5 Similar Products)

This module allows users to select a product name and instantly receive **5 similar product recommendations** based on cosine similarity. It uses saved recommendation artifacts:

- models/product_similarity.pkl
- models/product_list.pkl

Output: Top 5 recommended product names displayed in a clean card-style UI for better usability and presentation.

12.4 Business Insights Dashboard (KPIs + Segment Distribution + Use Cases)

The dashboard module presents core business indicators computed from the cleaned dataset and labeled segments:

- **Total Customers** (based on CustomerID)
- **Total Transactions** (based on unique InvoiceNo)

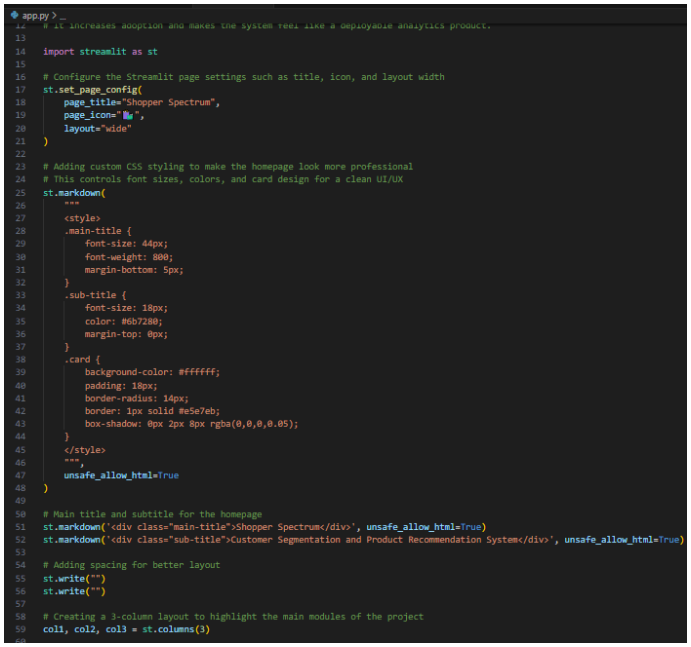
- **Unique Products** (based on unique Description)
- **Total Revenue** (sum of TotalPrice)

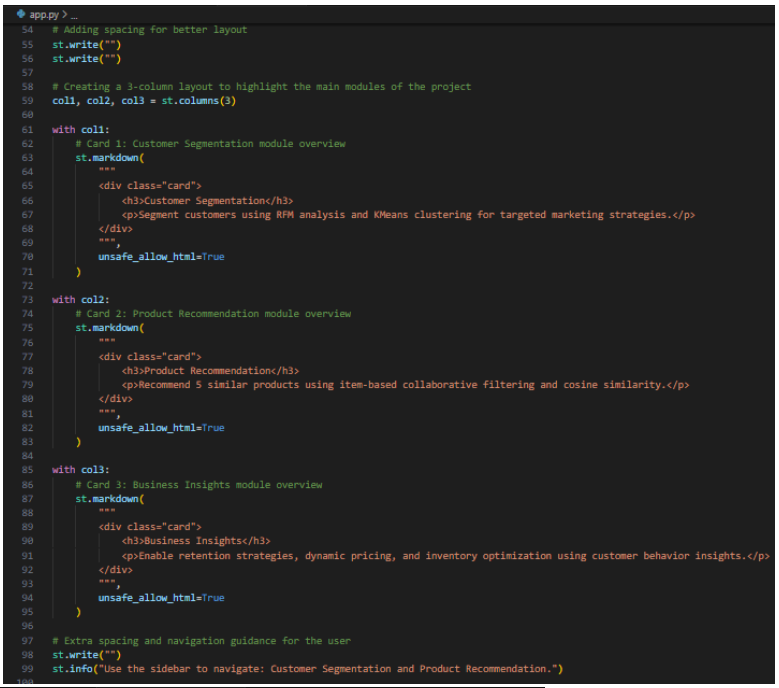
It also displays customer segment distribution and explains real-time business use cases such as retention, targeted marketing, dynamic pricing, and inventory planning.

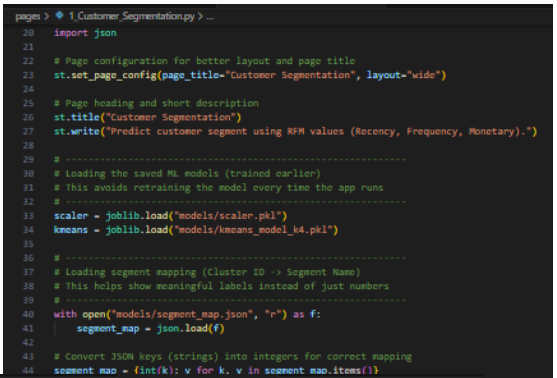
12.5 Model Saving & Loading Strategy (Joblib + JSON)

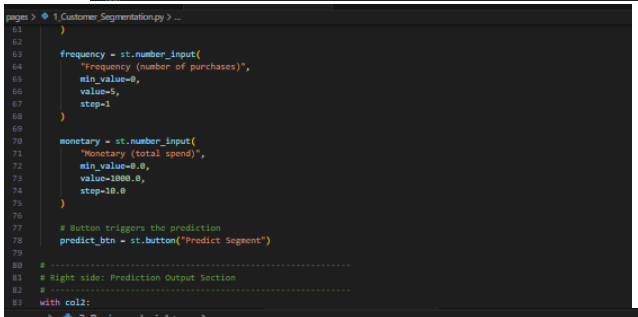
To ensure real-time prediction performance and consistent results across runs, models and artifacts were stored using:

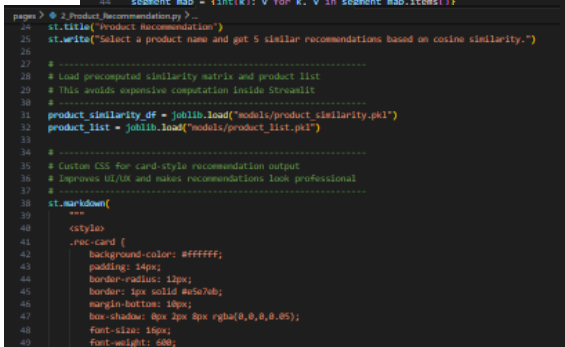
- **Joblib (.pkl)** for trained models and recommendation data
- **JSON** for segment mapping labels

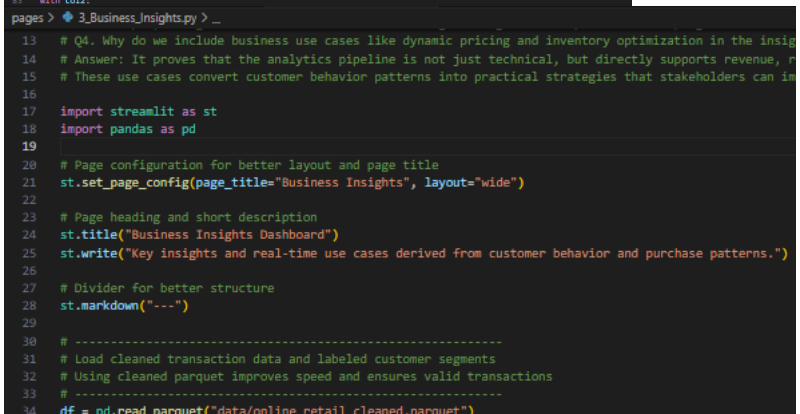


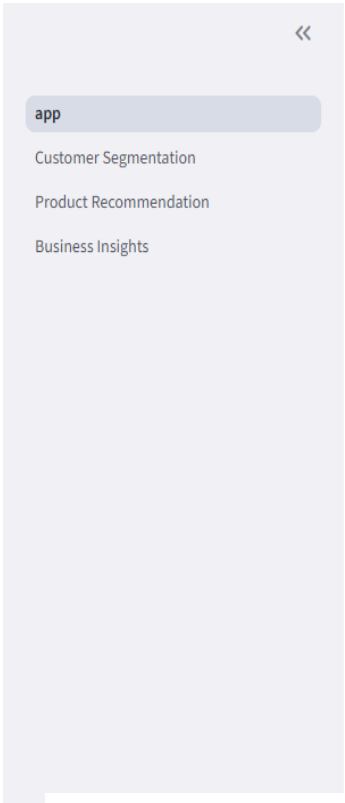












Deploy

Shopper Spectrum

Customer Segmentation and Product Recommendation System

Customer Segmentation

Segment customers using RFM analysis and KMeans clustering for targeted marketing strategies.

Product Recommendation

Recommend 5 similar products using item-based collaborative filtering and cosine similarity.

Business Insights

Enable retention strategies, dynamic pricing, and inventory optimization using customer behavior insights.

Use the sidebar to navigate: Customer Segmentation and Product Recommendation.

Customer Segmentation

Predict customer segment using RFM values (Recency, Frequency, Monetary).

Recency (days since last purchase)

12 − +

Frequency (number of purchases)

15 − +

Monetary (total spend)

10500.00 − +

Predict Segment

Prediction Output

Predicted Cluster: 3

Customer Segment: Regular

Recommended Action: Cross-sell bundles and membership benefits to increase repeat purchases.

Customer Segmentation

Predict customer segment using RFM values (Recency, Frequency, Monetary).

Recency (days since last purchase)

3

Frequency (number of purchases)

11

Monetary (total spend)

10500.00

Predict Segment

Prediction Output

Predicted Cluster: 0

Customer Segment: Occasional

Recommended Action: Offer discounts and personalized recommendations to increase engagement.

Product Recommendation

Select a product name and get 5 similar recommendations based on cosine similarity.

Select Product Name

12 HANGING EGGS HAND PAINTED

Get Recommendations

Top 5 Recommended Products

1. SET OF 6 EASTER RAINBOW CHICKS
2. BAG OF SILVER STONES
3. UBO-LIGHT TRIOBASE BLUE
4. ORIENTAL RED CUSHION COVER
5. GREEN PEONY CUSHION COVER

Product Recommendation

Select a product name and get 5 similar recommendations based on cosine similarity.

Select Product Name

3 PIECE SPACEBOY COOKIE CUTTER SET

Get Recommendations

Top 5 Recommended Products

1. MEMO BOARD COTTAGE DESIGN
2. SET 12 COLOURING PENCILS DOILEY
3. SET 36 COLOURING PENCILS DOILEY
4. I LOVE LONDON MINI RUCKSACK
5. DOLLCRAFT GIRL AMELIE KIT

Business Insights Dashboard

Key insights and real-time use cases derived from customer behavior and purchase patterns.

Total Customers	Total Transactions	Unique Products	Total Revenue
4,338	18,532	3,866	8,911,407.90

Customer Segment Distribution

	Segment	Customers
0	Occasional	3054
1	At Risk	1067
2	Regular	204
3	High Value	13

Real-time Business Use Cases

Customer Segmentation for Targeted Marketing Campaigns

- High Value: premium loyalty rewards, exclusive offers, personalized bundles
- Regular: cross-sell campaigns, membership benefits, repeat-purchase reminders
- Occasional: discounts, trending products, engagement campaigns
- At Risk: win-back offers, reminders, limited-time coupons

Personalized Product Recommendations on E-Commerce Platforms

- Recommend 5 similar products using cosine similarity
- Improves customer experience and increases average order value
- Helps cross-selling and product discovery

Identifying At-Risk Customers for Retention Programs

- At Risk customers have high recency and low frequency
- Retention actions: reminder emails, personalized discounts, win-back campaigns
- Prevents churn and improves repeat purchases

Dynamic Pricing Strategies Based on Purchase Behavior

- High demand products can maintain price with minimal discounts
- Low demand products can be promoted with discounts to increase sales
- Segment-based offers improve revenue without over-discounting

13. Business Insights Dashboard (Final Outputs)

13.1 Key Business KPIs

The dashboard summarizes key operational metrics that reflect overall business health and performance. These KPIs include total unique customers, total invoice transactions, unique product count, and total revenue derived from all valid purchases using the engineered TotalPrice feature

13.2 Segment Distribution Summary (With Numeric Results)

Customer segmentation revealed a strong imbalance in customer value contribution, which is common in retail. The final segment distribution was:

- **Occasional Customers:** 3054
- **At Risk Customers:** 1067
- **Regular Customers:** 204
- **High Value Customers:** 13

Insight: Although High Value customers are the smallest group, they represent the highest revenue opportunity, while Occasional customers represent the largest conversion opportunity by volume.

13.3 Revenue Contribution Patterns (Pareto Insight)

Revenue distribution analysis confirmed that customer spend is highly skewed, where a small fraction of customers contribute disproportionately to total revenue. This validates the need to retain and prioritize high-value and regular customers through loyalty strategies rather than applying uniform discounts across the customer bases.

13.4 Actionable Recommendations per Segment

Each segment enables direct business decision-making:

- **High Value (13 customers):** premium rewards, exclusive access offers, priority support
- **Regular (204 customers):** bundle offers, cross-sell recommendations, membership upgrades
- **Occasional (3054 customers):** discounts, engagement campaigns, trending product discovery
- **At Risk (1067 customers):** win-back offers, reminders, time-limited incentives

Inference: Segment-based targeting improves campaign efficiency by aligning marketing actions with customer profitability and churn risk.

13.5 Decision-Making Impact (Marketing, Retention, Inventory)

The dashboard connects analytics to business strategy by enabling:

- smarter allocation of marketing budget across segments
- early identification of churn-prone customers for retention intervention
- inventory prioritization based on high-frequency customer demand signals
- product recommendation-driven cross-selling to improve average order value

14. Project Constraints & Practical Considerations

14.1 Handling Large Dataset Efficiently

The raw dataset contained **541,909 rows**, making repeated CSV loading and heavy pivot computations expensive for a local system. To optimize performance, the dataset was cleaned once and converted into a **Parquet file**, reducing load time and enabling faster downstream processing for EDA, RFM generation, and model training.

14.2 Local Machine Limitations (RAM/Storage Constraints)

The project was executed on a system with **8 GB RAM** and limited storage availability (~8 GB free). This required efficient memory handling and avoiding unnecessary recomputation, especially during recommendation matrix creation and visualization generation.

14.3 Recommendation Matrix Complexity & Scalability

The recommendation model required building a customer-product interaction matrix and computing cosine similarity across products. The pivot table size was:

- **4338 customers × 3866 products**

This means product-to-product similarity can scale to millions of similarity pairs, making it computationally heavy. To address this, similarity results were saved using Joblib so that recommendations load instantly inside Streamlit without recomputation.

14.4 Model Output Imbalance and Business Interpretability

Model comparison revealed that DBSCAN produced highly imbalanced clustering results:

- **4262 customers in one cluster**, **9 in another**, and **67 noise points**
  Even though DBSCAN gave a high silhouette score (**0.6662**), it was not suitable for segmentation-driven marketing because it could not generate usable customer groups. KMeans was selected because it produced stable and interpretable segments aligned with business requirements.

15. Final Conclusion

15.1 Summary of Achievements

The project successfully transformed raw retail transaction data (**541,909 rows**) into a complete analytics pipeline including data cleaning, EDA, RFM profiling, customer segmentation, product recommendations, statistical validation, and real-time Streamlit deployment. The final outputs are both technically solid and directly aligned with real-world business decision-making.

15.2 Business Impact Delivered (With Numeric Results)

Customer segmentation created four business-ready segments with clear behavior patterns:

- **Occasional:** 3054 customers
- **At Risk:** 1067 customers
- **Regular:** 204 customers
- **High Value:** 13 customers

This distribution confirms that the business has a large conversion opportunity in the Occasional group and a major revenue-protection requirement for the small but extremely valuable High Value group.

15.3 What Worked Best & Why (Model + Recommendation Validation)

Among three clustering models (KMeans, Agglomerative, DBSCAN), **KMeans (k=4)** was selected because it provided the most balanced combination of clustering performance and business interpretability (**Silhouette = 0.6162, DBI = 0.7534**). The recommendation system also proved statistically meaningful with **Top-5 similarity mean ≈ 0.8623**, compared to a random baseline (**≈ 0.0088**), confirming strong personalization quality.

15.4 Final Recommendations for Stakeholders

The business should prioritize retention of High Value and Regular customers through loyalty incentives and premium offers, while using recommendations and discounts to increase frequency among Occasional customers. At Risk customers should be targeted with win-back campaigns, reminders, and time-based offers to reduce churn and maximize customer lifetime value.