Customer Segmentation Using SQL

Table of Contents

Project Overview

This project analyzes transactional retail data to segment customers into groups based on behavioral metrics like Recency, Frequency, and Monetary (RFM).

Project Purpose

To help the business understand different types of customers through segmentation and identify strategies for increasing customer retention, loyalty, and sales.

Context

The dataset contains online retail transactions between 2010 and 2011. My goal is to:

- Identify loyal customers
- Find high-value customers
- Spot dormant or at-risk customers
- Make data-driven marketing decisions

Actions

- Clean and preprocess data
- Calculate RFM metrics
- Define scoring thresholds using SQL logic
- Segment customers based on RFM scores
- Profile each segment to understand characteristics

Results

I created 4 distinct customer segments:

- Champions: Recent, frequent, and high spenders
- Loyal Customers: Frequent but less recent
- At Risk: Not active recently
- Potential: Moderate spenders with recent activity

Growth/Next Steps

- Add time-based tracking to analyze customer lifecycle stages
- Integrate with marketing platforms using SQL exports
- Add product segmentation within each customer group

Data Overview

Column Name	**Description**
InvoiceNo	Transaction ID
StockCode	Product code
Description	Product description
Quantity	Items sold
InvoiceDate	Date of transaction
UnitPrice	Price per item
CustomerID	Unique customer ID
Country	Customer's country

Concept Overview

Recency: Number of days since last purchase.
Frequency: Total number of purchases.
Monetary: Total revenue (Quantity × UnitPrice).
RFM Score: Combined 3-digit score (e.g., 443).

Key SQL Concepts:

- `CASE WHEN` for scoring
- `GROUP BY` for aggregation
- `DATEDIFF` or `TIMESTAMPDIFF` for recency
- `JOIN`s to merge scores with customer data

Data Preprocessing

```
-- Remove NULLs
DELETE FROM ‘online retail‘
WHERE CustomerID IS NULL;

-- Remove negative or zero quantity
DELETE FROM ‘online retail‘
WHERE Quantity <= 0;

-- Create a view with TotalPrice
CREATE OR REPLACE VIEW clean_sales AS
SELECT *,
       Quantity * UnitPrice AS TotalPrice
FROM ‘online retail‘;

SELECT * FROM clean_sales LIMIT 4;
```

Invoice No	Stock Code	Description	Qty	Invoice Date	Unit Price	Cust ID	Country	Total
536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850	United Kingdom	15.30
536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850	United Kingdom	20.34
536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850	United Kingdom	22.00
536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850	United Kingdom	20.34

Segmentation Criteria

- Recency: How recently a customer purchased
- Frequency: Number of unique invoices
- Monetary: Total revenue from a customer

Customer Segmentation (RFM Calculation)

```sql
-- Set reference date for Recency
SELECT MAX(InvoiceDate) INTO @latest_date FROM clean_sales;

-- RFM metrics
CREATE OR REPLACE VIEW rfm_metrics AS
SELECT
  CustomerID,
  DATEDIFF(
    (SELECT MAX(InvoiceDate) FROM clean_sales),
    MAX(InvoiceDate)
  ) AS Recency,
  COUNT(DISTINCT InvoiceNo) AS Frequency,
  SUM(TotalPrice) AS Monetary
FROM clean_sales
GROUP BY CustomerID;
```

CustomerID	Recency	Frequency	Monetary
12347	5	1	711.79
12386	1	1	258.90
12395	9	1	346.10
12427	1	1	303.50
12429	3	1	1281.50
12431	11	1	358.25
12433	4	2	3787.20
12441	2	1	173.55
12471	2	1	2360.41
12472	7	1	1531.30

Append Segments to Customers (Scoring)

```sql
CREATE OR REPLACE VIEW rfm_scores AS
SELECT *,
  CASE
    WHEN Recency <= 30 THEN 4
    WHEN Recency <= 60 THEN 3
    WHEN Recency <= 90 THEN 2
    ELSE 1
  END AS R_score,

  CASE
    WHEN Frequency >= 50 THEN 4
    WHEN Frequency >= 20 THEN 3
    WHEN Frequency >= 10 THEN 2
    ELSE 1
  END AS F_score,

  CASE
    WHEN Monetary >= 10000 THEN 4
    WHEN Monetary >= 5000 THEN 3
    WHEN Monetary >= 1000 THEN 2
    ELSE 1
  END AS M_score
FROM rfm_metrics
ORDER BY CustomerID;
```

CustomerID	Recency	Frequency	Monetary	R_score	F_score	M_score
12347	5	1	711.79	4	1	1
12386	1	1	258.90	4	1	1
12395	9	1	346.10	4	1	1
12427	1	1	303.50	4	1	1
12429	3	1	1281.50	4	1	2
12431	11	1	358.25	4	1	1
12433	4	2	3787.20	4	1	2
12441	2	1	173.55	4	1	1
12471	2	1	2360.41	4	1	2
12472	7	1	1531.30	4	1	2

Segment Profiling

```
CREATE OR REPLACE VIEW customer_segments AS
SELECT *,
  CONCAT(R_score, F_score, M_score) AS RFM_Score,
  CASE
    WHEN R_score = 4 AND F_score = 4 AND M_score = 4 THEN 'Champions'
    WHEN R_score >= 3 AND F_score >= 3 THEN 'Loyal Customers'
    WHEN R_score <= 2 AND F_score <= 2 THEN 'At Risk'
    WHEN R_score >= 3 AND M_score <= 2 THEN 'Potential'
    ELSE 'Others'
  END AS Segment
FROM rfm_scores;

SELECT Segment, COUNT(*) AS CustomerCount
FROM customer_segments
GROUP BY Segment;
```

Count Customers per Segment

```
SELECT Segment, COUNT(*) AS customer_count
FROM customer_segments
GROUP BY Segment
ORDER BY customer_count DESC;
```

Segment	customer_count
Potential	628
Others	4
Loyal Customers	2

Total Monetary Value per Segment

```sql
SELECT Segment, ROUND(SUM(Monetary), 2) AS total_revenue
FROM customer_segments
GROUP BY Segment
ORDER BY total_revenue DESC;
```

Segment	total_revenue
Potential	29161.73
Others	57587.18
Loyal Customers	8809.54

Average Order Value per Segment

```sql
SELECT Segment, ROUND(AVG(Monetary / Frequency), 2) AS avg_order_value
FROM customer_segments
GROUP BY Segment
ORDER BY avg_order_value DESC;
```

Segment	avg_order_value
Others	2478.52
Potential	361.40
Loyal Customers	153.59