**Project Report: Credit Card Engagement and Share of Wallet Analysis**

Prepared by: Akarsh Anan  
Date: 30-06-2024

**Executive Summary**

This project focused on analysing customer engagement with Union Express (Imaginary Company) Internal cards versus external cards (Other Companies Competitors) , leveraging dummy data tailored for this analysis. The primary objective was to track Share of Wallet (SoW) trends, identify drivers of customer disengagement, and provide actionable insights to optimize customer engagement strategies.

Key findings highlight significant differences between with Union Express (Imaginary Company) internal and external cards in transaction behaviour, reward points allocation, and SoW (Share of wallet) percentages. Internal cards exhibited higher average transaction amounts and reward points, suggesting potential benefits for customer retention strategies. Conversely, external cards showed higher transaction volumes and SoW (Share of wallet) percentages, driven by competitive loyalty programs.

These insights underscore the importance of tailored engagement strategies based on card type to enhance overall customer satisfaction and retention.

## Introduction

In today's competitive financial services landscape, understanding customer preferences and engagement patterns is crucial for optimizing card offerings and enhancing customer satisfaction. This project focuses on analysing transactional data to uncover insights into customer behaviour and preferences regarding Union Express (Imaginary cards) cards compared to external competitors.

## Methodology

### **Data Collection and Preparation**

* **Data Sources**: Utilized dummy transactional data mimicking real-world scenarios.
* **Data Cleaning**: Processed data to ensure consistency and accuracy, including date formatting and data type adjustments.
* **Analysis Tools**: Leveraged SQL queries and Python for data analysis and visualization.

Since I do not have access to the specific findings and detailed project outcomes from your analysis, I will provide a generic example of how you might structure and write a report based on typical findings related to credit card engagement and Share of Wallet (SoW) analysis. You can adapt this template to include your actual findings and insights.

## Data Analysis

### Findings

#### Transaction Analysis by Card Type

* **Average Transaction Amount**:
  + External Cards: $250.64
  + Internal Cards: $294.12
* **Number of Transactions**:
  + External Cards: 18
  + Internal Cards: 14

#### Share of Wallet (SoW) Analysis

* **SoW Percentage**:
  + External Cards: 52.28%
  + Internal Cards: 47.72%
* **Average Reward Points**:
  + Internal Cards: 29.36 points
  + External Cards: 25.22 points

## Results

### Insights

1. **Transaction Behaviour**: Internal cardholders tend to make fewer but higher-value transactions compared to external cardholders.
2. **Share of Wallet Dynamics**: External cards capture a higher percentage of wallet share, suggesting stronger customer loyalty possibly driven by reward programs.
3. **Reward Point Comparison**: Despite higher average reward points on internal cards, the overall engagement and SoW metrics Favor external cards, highlighting their competitive advantages in customer retention.

## Discussion

The findings underscore the importance of enhancing internal card features to better compete with external offerings. Strategies should focus on improving transaction frequency and loyalty programs to increase customer engagement and wallet share.

## Conclusion

In conclusion, the analysis reveals distinct transaction patterns and loyalty dynamics between Union Express (Imaginary Company) internal and external cards. Moving forward, targeted initiatives to enhance internal card benefits and customer engagement strategies will be pivotal in strengthening market position and fostering long-term customer relationships.

**Appendices**

* SQL Queries Used:

Create database credit\_analysis;

use credit\_analysis

/\* Create table for customerDemographics and transactionalData\*/

CREATE TABLE CustomerDemographics (

customer\_id INT PRIMARY KEY,

Age INT,

Gender VARCHAR(10),

Income\_level VARCHAR(10),

Location VARCHAR(50),

Education\_level VARCHAR(20)

);

CREATE TABLE TransactionalData (

transaction\_id INT PRIMARY KEY,

customer\_id INT,

transaction\_date DATETIME,

amount\_spent FLOAT,

merchant\_category VARCHAR(20),

reward\_points\_earned INT,

card\_type varchar(20),

FOREIGN KEY (customer\_id) REFERENCES CustomerDemographics(customer\_id)

);

select \* from TransactionalData

/\* Customer Engagement Analysis\*/

-- Total amount spent per customer

SELECT customer\_id, SUM(amount\_spent) AS total\_spent

FROM TransactionalData

GROUP BY customer\_id;

-- Average spend per transaction by card type

SELECT card\_type, AVG(amount\_spent) AS avg\_spent

FROM TransactionalData

GROUP BY card\_type;

-- Number of transactions by card type

SELECT card\_type, COUNT(\*) AS transaction\_count

FROM TransactionalData

GROUP BY card\_type;

-- Share of Wallet (SoW) Calculation

-- Calculate total spend for internal and external cards per customer

SELECT customer\_id,

SUM(CASE WHEN card\_type = 'Internal' THEN amount\_spent ELSE 0 END) AS internal\_spent,

SUM(CASE WHEN card\_type = 'External' THEN amount\_spent ELSE 0 END) AS external\_spent

FROM TransactionalData

GROUP BY customer\_id;

-- Number of trnsactions on Internal Card

SELECT COUNT(\*)

FROM TransactionalData

WHERE card\_type = 'Internal';

-- Number of trnsactions on External Card

SELECT COUNT(\*)

FROM TransactionalData

WHERE card\_type = 'External';

-- Transaction date and customer Id of Internal card users

SELECT customer\_id, transaction\_date

FROM TransactionalData

WHERE card\_type = 'Internal'

ORDER BY transaction\_date DESC

LIMIT 10;

-- -- Transaction date and customer Id of External card users

SELECT customer\_id, transaction\_date

FROM TransactionalData

WHERE card\_type = 'External'

ORDER BY transaction\_date DESC

LIMIT 10;

-- This query calculates the total number of transactions, average transaction amount,

-- and the counts of internal and external card transactions for each customer.

SELECT

customer\_id AS CustomerID,

COUNT(DISTINCT transaction\_id) AS NumTransactions,

AVG(amount\_spent) AS AvgTransactionAmount,

SUM(CASE WHEN card\_type = 'Internal' THEN 1 ELSE 0 END) AS NumInternalTransactions,

SUM(CASE WHEN card\_type = 'External' THEN 1 ELSE 0 END) AS NumExternalTransactions

FROM

transactionaldata

GROUP BY

customer\_id;

-- Identify customers who have not made internal transactions in the past 3 months

-- but have made transactions with external cards, including reward points analysis.

SELECT

t1.customer\_id,

COUNT(DISTINCT t1.transaction\_id) AS num\_external\_transactions,

AVG(t1.amount\_spent) AS avg\_external\_transaction\_amount,

SUM(t1.reward\_points\_earned) AS total\_external\_reward\_points,

MAX(t1.transaction\_date) AS last\_external\_transaction\_date,

t2.num\_internal\_transactions,

t2.avg\_internal\_transaction\_amount,

t2.total\_internal\_reward\_points,

t2.last\_internal\_transaction\_date

FROM

transactionaldata t1

LEFT JOIN

(SELECT

customer\_id,

COUNT(DISTINCT transaction\_id) AS num\_internal\_transactions,

AVG(amount\_spent) AS avg\_internal\_transaction\_amount,

SUM(reward\_points\_earned) AS total\_internal\_reward\_points,

MAX(transaction\_date) AS last\_internal\_transaction\_date

FROM

transactionaldata

WHERE

card\_type = 'Internal'

GROUP BY

customer\_id) t2

ON t1.customer\_id = t2.customer\_id

WHERE

t1.card\_type = 'External'

AND (t2.last\_internal\_transaction\_date IS NULL OR t2.last\_internal\_transaction\_date < CURDATE() - INTERVAL 3 MONTH)

GROUP BY

t1.customer\_id;

-- Identify customers who have not made external transactions in the past 3 months

-- but have made transactions with internal cards, including reward points analysis.

SELECT

t1.customer\_id,

COUNT(DISTINCT t1.transaction\_id) AS num\_internal\_transactions,

AVG(t1.amount\_spent) AS avg\_internal\_transaction\_amount,

SUM(t1.reward\_points\_earned) AS total\_internal\_reward\_points,

MAX(t1.transaction\_date) AS last\_internal\_transaction\_date,

t2.num\_external\_transactions,

t2.avg\_external\_transaction\_amount,

t2.total\_external\_reward\_points,

t2.last\_external\_transaction\_date

FROM

transactionaldata t1

LEFT JOIN

(SELECT

customer\_id,

COUNT(DISTINCT transaction\_id) AS num\_external\_transactions,

AVG(amount\_spent) AS avg\_external\_transaction\_amount,

SUM(reward\_points\_earned) AS total\_external\_reward\_points,

MAX(transaction\_date) AS last\_external\_transaction\_date

FROM

transactionaldata

WHERE

card\_type = 'External'

GROUP BY

customer\_id) t2

ON t1.customer\_id = t2.customer\_id

WHERE

t1.card\_type = 'Internal'

AND (t2.last\_external\_transaction\_date IS NULL OR t2.last\_external\_transaction\_date < CURDATE() - INTERVAL 3 MONTH)

GROUP BY

t1.customer\_id;

-- Calculate average reward points earned per transaction for internal cards

SELECT

'Internal' AS CardType,

AVG(reward\_points\_earned) AS AvgRewardPoints

FROM

TransactionalData

WHERE

card\_type = 'Internal';

-- Calculate average reward points earned per transaction for external cards

SELECT

'External' AS CardType,

AVG(reward\_points\_earned) AS AvgRewardPoints

FROM

TransactionalData

WHERE

card\_type = 'External';

-- Calculate Share of Wallet (SoW) percentage for internal cards

SELECT

'Internal' AS CardType,

SUM(amount\_spent) / (SELECT SUM(amount\_spent) FROM TransactionalData) \* 100 AS SoW\_Percentage

FROM

TransactionalData

WHERE

card\_type = 'Internal';

-- Calculate Share of Wallet (SoW) percentage for external cards

SELECT

'External' AS CardType,

SUM(amount\_spent) / (SELECT SUM(amount\_spent) FROM TransactionalData) \* 100 AS SoW\_Percentage

FROM

TransactionalData

WHERE

card\_type = 'External';

**Python Code:**

# Approach to create dummy data for analysis purposes

# 1) Create a dummy data for Customer Demographics

# What is customer demographics data?

# Customer Demographics data is basic information about customers characteristics and attributes. It includes the following details:

# Customer ID: A unique identifier for each customer.

#Age: Age of the customer.

#Gender: Gender identity of the customer (e.g., male, female, non-binary).

#Income Level: Income bracket or range of the customer.

#Location: Geographical location where the customer resides (often includes city, state, or region).

#Education Level: Highest level of education completed by the customer.

#Other Demographic Variables: These could include marital status, household size, occupation, and more, depending on the specific needs of the analysis or project.

# Python code to create the following dummy data

import pandas as pd

import numpy as np

import random

# Generate Customer Demographics Data

num\_customers = 5000  # Increased number of customers

customer\_ids = range(1, num\_customers + 1)

ages = np.random.randint(18, 80, *size*=num\_customers)  # Increased age range

genders = np.random.choice(['Male', 'Female', 'Other'], *size*=num\_customers)  # Added 'Other' gender

income\_levels = np.random.choice(['High', 'Medium', 'Low'], *size*=num\_customers)

locations = np.random.choice(['New York, NY', 'Los Angeles, CA', 'Chicago, IL', 'Houston, TX', 'Miami, FL'], *size*=num\_customers)  # Expanded location choices

education\_levels = np.random.choice(['Graduate', 'Undergraduate', 'High School'], *size*=num\_customers)

# Create DataFrame

customer\_demographics = pd.DataFrame({

    'Customer ID': customer\_ids,

    'Age': ages,

    'Gender': genders,

    'Income Level': income\_levels,

    'Location': locations,

    'Education Level': education\_levels

})

# Display first few rows

print(customer\_demographics)

# Save to CSV

customer\_demographics.to\_csv('customer\_demographics.csv', *index*=False)

#2) Python code to create Transactional Data

# What is transactional data?

# Transactional Data contains the data about customers spending and transaction on different merchant types. It contains the followig details :

#Transaction ID: A unique identifier for each transaction.

#Customer ID: Identifier for the customer associated with the transaction.

#Transaction Date: Date and time when the transaction occurred.

#Amount Spent: The monetary value of the transaction.

#Merchant Category: Category of the merchant where the transaction took place (e.g., dining, retail, groceries).

#Reward Points Earned: Number of loyalty or reward points accrued for the transaction.

# Python code to create the following dummy data

 # Generate Transactional Data

import pandas as pd

import numpy as np

import random

from datetime import datetime, timedelta

# Function to generate dummy transactional data

def generate\_dummy\_data(*num\_rows*):

    transaction\_ids = range(1, num\_rows + 1)

    customer\_ids = np.random.choice(range(1, 1001), *size*=num\_rows)

    # Generate random transaction dates within the past year

    end\_date = datetime.now()

    start\_date = end\_date - timedelta(*days*=365)

    transaction\_dates = [start\_date + timedelta(*days*=random.randint(0, 365), *hours*=random.randint(0, 23), *minutes*=random.randint(0, 59)) for \_ in range(num\_rows)]

    amount\_spent = np.round(np.random.uniform(5, 500, *size*=num\_rows), 2)

    merchant\_categories = np.random.choice(['Dining', 'Retail', 'Groceries', 'Entertainment', 'Travel'], *size*=num\_rows)

    reward\_points = np.round(amount\_spent \* 0.1).astype(*int*)

    # Generating card types randomly

    card\_types = np.random.choice(['Internal', 'External'], *size*=num\_rows)

    transactional\_data = pd.DataFrame({

        'transaction\_id': transaction\_ids,

        'customer\_id': customer\_ids,

        'transaction\_date': transaction\_dates,

        'amount\_spent': amount\_spent,

        'merchant\_category': merchant\_categories,

        'reward\_points\_earned': reward\_points,

        'card\_type': card\_types

    })

    return transactional\_data

# Generating 50000 rows of dummy data

num\_rows = 50000

transactional\_data = generate\_dummy\_data(num\_rows)

# Displaying the generated DataFrame

print(transactional\_data.head())

# Cleaning the data

transactional\_data['transaction\_date'] = pd.to\_datetime(transactional\_data['transaction\_date'])

transactional\_data['amount\_spent'] = transactional\_data['amount\_spent'].astype(*float*)

# Saving to CSV

transactional\_data.to\_csv('transactional\_data.csv', *index*=False)

print("Data generation, cleaning, and saving completed.")

# Competitor data refers to information about other companies or entities

# operating in the same industry or market as a particular organization.

# typically includes details such as:

#

# - Competitor Name: Name of the competing company or entity.

# - Market Share (%): The percentage of the market controlled or captured by the competitor.

# - Key Features: Unique selling points or attributes of the competitor's products or services

#   (e.g., cashback offers, low interest rates, rewards programs).

# - Customer Satisfaction Rating: Ratings or feedback from customers regarding their

#   satisfaction with the competitor's products or services.

#

# Analyzing competitor data provides businesses with valuable insights into market dynamics,

# competitive landscape, customer preferences, and areas for improvement.

# This information helps organizations understand their position

# relative to competitors, identify opportunities for differentiation, and develop strategies

# to enhance their market presence and customer engagement.

#Python code to generate data for competitors

# Generate Competitor Data

competitors = ['Competitor A', 'Competitor B', 'Competitor C', 'Competitor D', 'Competitor E']

market\_share = [random.randint(20, 40) for \_ in range(len(competitors))]  # Random market share percentages

possible\_key\_features = [

    'Cashback, Lower Interest Rates',

    'Better Rewards, Lower Annual Fees',

    'Points for Every Purchase',

    'Exclusive Benefits',

    'No Annual Fees'

]

key\_features = [random.choice(possible\_key\_features) for \_ in range(len(competitors))]

customer\_satisfaction = [round(random.uniform(3.5, 5.0), 1) for \_ in range(len(competitors))]  # Random ratings

# Create DataFrame

competitor\_data = pd.DataFrame({

    'Competitor Name': competitors,

    'Market Share (%)': market\_share,

    'Key Features': key\_features,

    'Customer Satisfaction Rating': customer\_satisfaction

})

# Display DataFrame

print(competitor\_data)

# Save to CSV

competitor\_data.to\_csv('competitor\_data.csv', *index*=False)