**DATA ANALYST ASSESSMENT ANALYSIS**

**NAME: Roy Mariam**

**TOOLS USED: SQL**

**DATE: 18/06/2025**

**Q1: HIGH-VALUE CUSTOMERS WITH MULTIPLE PRODUCTS ANALYSIS**

**Title: High-Value Customers with Multiple Products**

**OBJECTIVE**

To identify high-value customers who have both a funded savings

account and an investment plan, in order to uncover cross-selling

opportunities.

**APPROACH**

1. **Data Extraction:**

• The dataset was provided as a MySQL dump

(adashi\_assessment.sql).

• Tables involved: users\_customuser, savings\_savingsaccount, and

plans\_plan.

2. **Schema Understanding:**

• Analyzed the structure of each table to understand relationships.

• Identified:

• owner\_id as the foreign key linking users to savings and plans.

• transaction\_status in savings\_savingsaccount to evaluate savings

activity.

• interest\_rate in plans\_plan to infer investment plans.

3. **Business Logic:**

• Customers with **at least one savings account** with a **successful**

**transaction status**.

• Customers with **at least one investment plan** (determined by

interest\_rate > 0).

• Aggregated total deposits using new\_balance from savings and

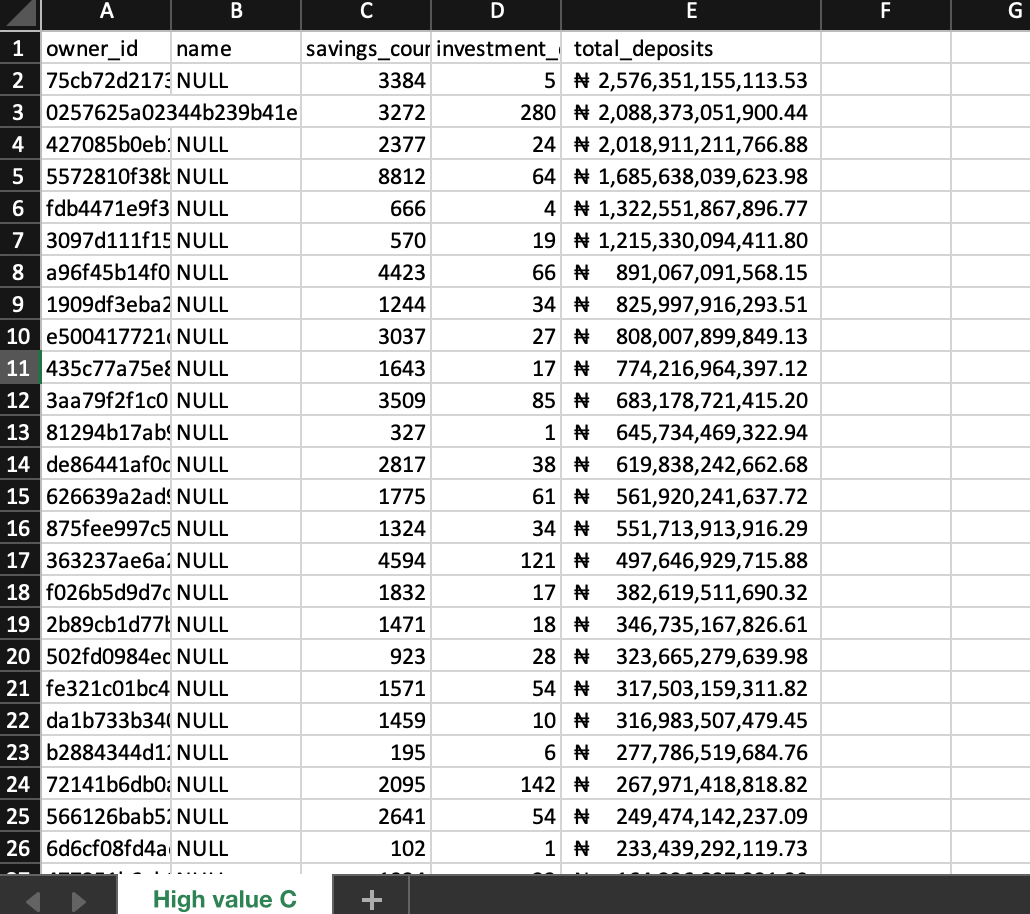
amount from plans.

4. **SQL Query Development:**

• Wrote SQL joins and applied filters to meet the logic above.

• Used GROUP BY, HAVING, and ORDER BY to generate final insights.

**TABLE OUTPUT**

****

**CHALLENGES FACED**

1. **Non-Standard Status Values:**

• The transaction\_status column in savings\_savingsaccount had many

custom and inconsistent status messages (e.g., “Charge attempt

cannot be fulfilled…”, “Successful”, “Abandoned”).

• Solution: Created a whitelist of known success indicators ('Success',

'Successful', 'Monthly\_success') to filter valid transactions.

2. **Missing/Unclear Fields:**

• There was no direct type field in plans\_plan to distinguish investment

plans.

• Solution: Used interest\_rate > 0 as a proxy to identify

investment-related entries.

3. **Data Quality Issues:**

• Some entries had missing or ambiguous information.

• Addressed this by using COALESCE () to handle NULL values in

calculations.

4. **Query Returns Empty Result:**

• Initially, the query returned no rows due to loose filtering criteria.

• Fixed by tightening filter logic and testing subqueries independently.

**Q2: TRANSACTION FREQUENCY ANALYSIS**

**Title: Transaction Frequency Analysis**

**OBJECTIVE**

To find and calculate the average number of transactions of each customer, for each month, and categorize them into:

"High Frequency" (≥10 transactions/month)

"Medium Frequency" (3-9 transactions/month)

"Low Frequency" (≤2 transactions/month)

**ANALYSIS APPROACH**

1. **Data Join:**

Joined users\_customuser and savings\_savingsaccount using the

foreign key owner\_id.

2. **Transaction Filtering:**

Only considered transactions with statuses:

• 'Success'

• 'Successful'

• 'Monthly\_success'

3. **Metric Calculation per User:**

• Total Transactions: Count of transaction records.

• Account Tenure (Months): MIN(created\_on) to MAX(created\_on).

• Average Transactions per Month:

avg = total\_transactions

—————————-

months active

4. **User Segmentation:**

Categorized customers based on monthly transaction average:

• High Frequency: ≥ 10

• Medium Frequency: 3 – 9

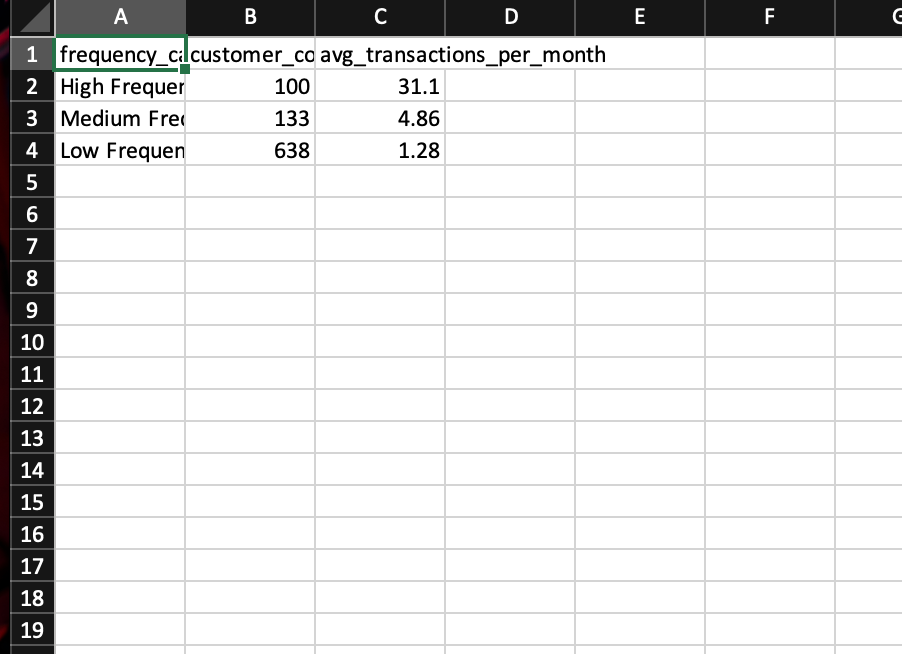
• Low Frequency: ≤ 2

5. **Final Aggregation:**

• Grouped by frequency category.

• Calculated total customers and average transaction rate per group.

**TABLE OUTPUT**

****

**CHALLENGES FACED**

1. **Uneven transaction data distribution:**

Some users had all their transactions in a single month, causing

inflated averages.

**Solution:** Included a minimum of +1 month to ensure fairness.

2. **No unified transaction table:**

The savings\_savingsaccount table holds balances and not explicit

transaction records.

**Solution:** Treated each entry with a successful status as one

transaction.

3. **Ambiguous transaction definitions:**

Some savings records may not truly reflect new transactions.

**Solution:** Filtered only successful transaction statuses to imply valid

activity.

4. **Edge cases with zero transactions or invalid dates:**

Some users may have savings accounts but no valid transactions.

**Solution:** Focused only on users with successful records and

ensured proper date handling.

**Q3: ACCOUNT INACTIVITY ALERT ANALYSIS**

**Title: Account Inactivity Alert**

**OBJECTIVE**

To find and identify active accounts (savings or investments) that have

had no inflow transactions in the last 1 year.

**APPROACH**

1. **Identified active accounts by filtering:**

• Savings accounts with successful transaction statuses.

• Investment plans with interest (interest\_rate > 0).

2.  **Calculated last transaction date** for each account using

MAX(created\_at).

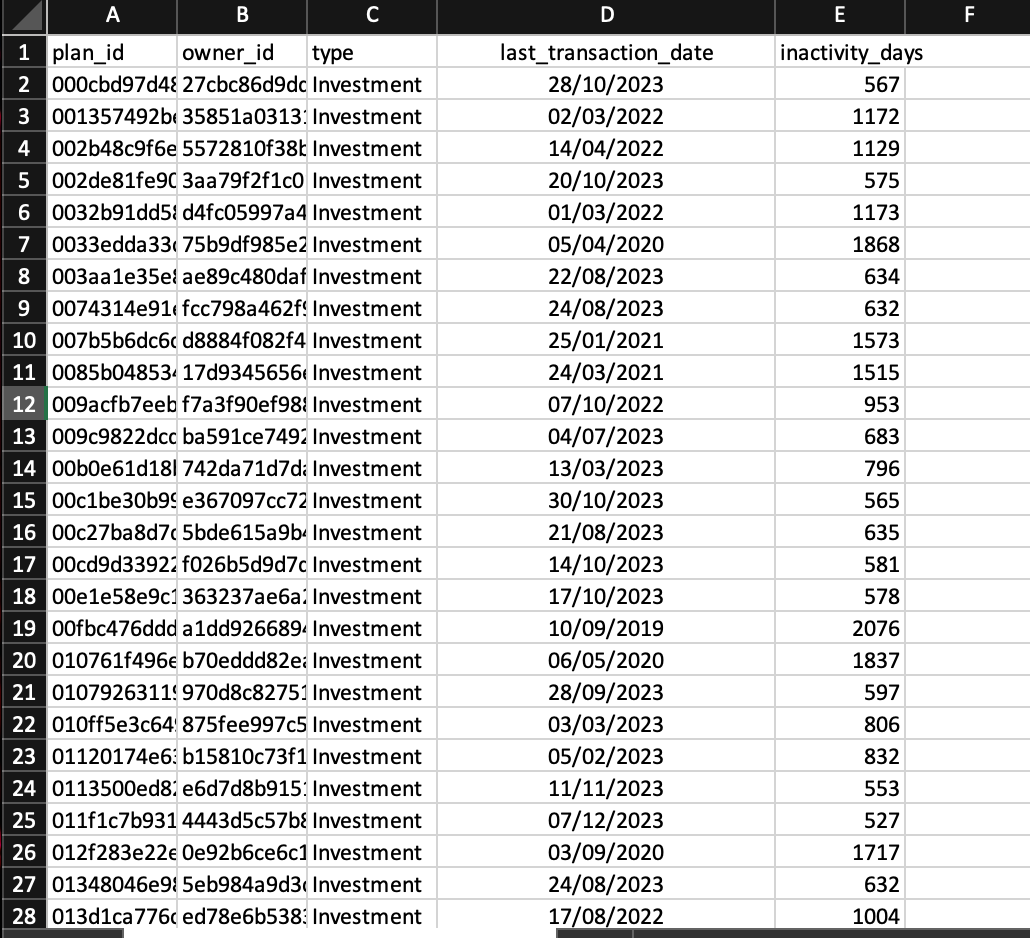
3. **Computed inactivity in days** using DATEDIFF(CURDATE(),

last\_transaction\_date).

4. **Combined savings and investment data** using UNION and flagged

accounts inactive for over 365 days.

**TABLE OUTPUT**

****

**CHALLENGES FACED**

1. **Inconsistent date formats or missing timestamps:**

Some rows may lack created\_on, which would block accurate

inactivity tracking.

**Solution:** Used fallback defaults and filtered only rows with valid

dates.

2. **Identifying successful transactions:**

Not all savings transactions imply inflows.

**Solution**: Applied a whitelist of statuses (e.g., Success, Successful).

3. **No standardized transaction table:**

Since inflow/outflow isn’t separated, we inferred activity from

created\_on.

**Solution:** Defined “activity” as any valid account creation or update.

**Q4: CUSTOMER LIFE VALUE (CLV) ESTIMATION ANALYSIS**

**Title: Customer Life Value (CLV) Estimation**

**OBJECTIVE**

To calculate Account tenure (months since signup), Total transaction Estimated

CLV (Assume: CLV = (total \_transactions / tenure) \* 12 \*,

avg\_profit\_per\_transaction), Order by estimated CLV from highest to lowest, for each customer assuming the profit\_per\_transaction is 0.1% of the

transaction value.

**APPROACH**

1. **Calculated tenure** using the difference between current date and

date\_joined.

2. **Summed all successful savings transactions** using new\_balance

and filtered for valid statuses.

3. **Estimated CLV using the formula:**

CLV = (Total Transactions)

——————————- X 12 X 0.001

(Tenure in Months)

Where 0.1% = 0.001 represents profit per transaction.

**TABLE OUTPUT**

****

**CHALLENGES FACED**

1. **Zero-tenure edge case:**

Some users had just joined, leading to division by zero in tenure.

**Solution:** Used NULLIF(..., 0) to prevent errors.

2. **Inconsistent transaction values:**

Some new\_balance entries may reflect balance rather than transaction

value.

**Solution:** Treated them as proxies for transaction totals due to

schema limitations.

3. **Investment data missing:**

CLV focused only on savings due to lack of clear investment

inflow/outflow tracking.

**Solution:** Highlighted this assumption and kept the model simplified.