Name:Himanshu Sharma
project_name:Customer segmentation analysis to improve conversion
---------->

1. Summary of active vs closed accounts.

```
SELECT
      "ACCOUNT_STATUS",
      COUNT("ACCOUNT_STATUS") AS TOTAL_ACCOUNT_STATUS
FROM
      TRANSACTION_LINE TL
GROUP BY
      1
ORDER BY
      1;

-- 2. Breakdown of account types (e.g., loans, credit cards) and their current balances.

SELECT
      "ACCOUNT_CATEGORY",
      COUNT(*) as count,
      SUM("ACCOUNT_BALANCE") AS TOTAL_CURRENT_BALANCE
FROM
      TRANSACTION_LINE TL
GROUP BY
      1
ORDER BY
      1;

-- 3. Analysis of loan amounts vs. account balances.

SELECT
      "ACCOUNT_CATEGORY",
      AVG("SANCTIONED_AMOUNT") AS AVG_LOAN_AMOUNT,
      AVG("ACCOUNT_BALANCE") AS AVG_ACCOUNT_BALANCE
FROM
      TRANSACTION_LINE TL
GROUP BY
      1;

-- 4. overview of the closure percentages for different loan types by ownership type
(Individual vs Joint Account)

-- Joint Account
WITH TOTAL_LOANS AS (
SELECT
      "ACCOUNT_CATEGORY",
      COUNT("ACCOUNT_STATUS") AS TOTAL_LOAN
FROM
      TRANSACTION_LINE TL
WHERE
      "OWNERSHIP_TYPE" = 'Joint Account'
GROUP BY
      1
),
TOTAL_CLOSED_LOANS AS (
SELECT
      "ACCOUNT_CATEGORY",
      COUNT("ACCOUNT_STATUS") AS TOTAL_CLOSED_LOAN
FROM
      TRANSACTION_LINE TL
WHERE
      "OWNERSHIP_TYPE" = 'Joint Account'
      AND "ACCOUNT_STATUS" = 'Closed'
GROUP BY
      1
)
```

```
SELECT
      TL."ACCOUNT_CATEGORY",
      TL.TOTAL_LOAN,
      TCL.TOTAL_CLOSED_LOAN,
      ROUND((TCL.TOTAL_CLOSED_LOAN::DECIMAL * 100 / TL.TOTAL_LOAN),
      2) AS CLOSURE_PERCENTAGE
FROM
      TOTAL_LOANS TL
JOIN TOTAL_CLOSED_LOANS TCL ON
      TL."ACCOUNT_CATEGORY" = TCL."ACCOUNT_CATEGORY"
GROUP BY
      1,
      2,
      3;

-- Individual
WITH TOTAL_LOANS AS (
SELECT
      "ACCOUNT_CATEGORY",
      COUNT("ACCOUNT_STATUS") AS TOTAL_LOAN
FROM
      TRANSACTION_LINE TL
WHERE
      "OWNERSHIP_TYPE" = 'Individual'
GROUP BY
      1
),
TOTAL_CLOSED_LOANS AS (
SELECT
      "ACCOUNT_CATEGORY",
      COUNT("ACCOUNT_STATUS") AS TOTAL_CLOSED_LOAN
FROM
      TRANSACTION_LINE TL
WHERE
      "OWNERSHIP_TYPE" = 'Individual'
      AND "ACCOUNT_STATUS" = 'Closed'
GROUP BY
      1
)
SELECT
      TL."ACCOUNT_CATEGORY",
      TL.TOTAL_LOAN,
      TCL.TOTAL_CLOSED_LOAN,
      ROUND((TCL.TOTAL_CLOSED_LOAN::DECIMAL * 100 / TL.TOTAL_LOAN),
      2) AS CLOSURE_PERCENTAGE
FROM
      TOTAL_LOANS TL
JOIN TOTAL_CLOSED_LOANS TCL ON
      TL."ACCOUNT_CATEGORY" = TCL."ACCOUNT_CATEGORY"
GROUP BY
      1,
      2,
      3;


-- 5. Let’s start by segmenting customers based on FICO scores and account categories to
see the distribution.

WITH FICO_SCORE_RANGES AS (
SELECT
      "CUSTOMER_ID",
      "FICO_SCORE",
      "ACCOUNT_CATEGORY",
      CASE
            WHEN "FICO_SCORE" BETWEEN 300 AND 549 THEN 'VERY POOR'
            WHEN "FICO_SCORE" BETWEEN 550 AND 649 THEN 'POOR'
```

```
            WHEN "FICO_SCORE" BETWEEN 650 AND 749 THEN 'FAIR'
            WHEN "FICO_SCORE" BETWEEN 750 AND 849 THEN 'GOOD'
            WHEN "FICO_SCORE" BETWEEN 850 AND 949 THEN 'EXCELLENT'
      END AS CREDIT_SEGMENT
FROM
      TRANSACTION_LINE TL
) ,
TOTAL_CUSTOMERS AS (
SELECT
      COUNT(DISTINCT "CUSTOMER_ID") AS TOTAL_CUSTOMER
FROM
      TRANSACTION_LINE TL
)
SELECT
      CREDIT_SEGMENT,
      "ACCOUNT_CATEGORY",
      COUNT("CUSTOMER_ID") AS CUSTOMER_SEGMENT_COUNT,
      TC.TOTAL_CUSTOMER,
      (COUNT("CUSTOMER_ID") * 100 / TC.TOTAL_CUSTOMER) AS PERCENTAGE
FROM
      FICO_SCORE_RANGES FSR
CROSS JOIN TOTAL_CUSTOMERS TC
GROUP BY
      1,
      2,
      4
ORDER BY
      2;


-- 6. Product Usage Segmentation: Categorizing customers by the types of accounts they
hold (e.g., Auto Loans, Credit Cards, etc.).

WITH PRODUCT_SEGMENTATION AS (
SELECT
      "CUSTOMER_ID",
      CASE
            WHEN AUTO_LOAN = 1 THEN
             CASE
                  WHEN CONSUMER_LOAN + CREDIT_CARD + GOLD_LOAN + HOUSING_LOAN +
PERSONAL_LOAN + TWO_WHEELER_LOAN = 0 THEN 'AUTO LOAN ONLY'
                  ELSE 'AUTO LOAN, ' ||
                      CASE
                        WHEN CONSUMER_LOAN = 1 THEN 'CONSUMER LOAN, '
                        ELSE ''
                  END ||
                      CASE
                        WHEN CREDIT_CARD = 1 THEN 'CREDIT CARD, '
                        ELSE ''
                  END ||
                      CASE
                        WHEN GOLD_LOAN = 1 THEN 'GOLD LOAN, '
                        ELSE ''
                  END ||
                      CASE
                        WHEN HOUSING_LOAN = 1 THEN 'HOUSING LOAN, '
                        ELSE ''
                  END ||
                      CASE
                        WHEN PERSONAL_LOAN = 1 THEN 'PERSONAL LOAN, '
                        ELSE ''
                  END ||
                      CASE
                        WHEN TWO_WHEELER_LOAN = 1 THEN 'TWO WHEELER LOAN'
                        ELSE ''
                  END
```

```
            END
            WHEN CONSUMER_LOAN = 1 THEN
             CASE
                  WHEN CREDIT_CARD + GOLD_LOAN + HOUSING_LOAN + PERSONAL_LOAN +
TWO_WHEELER_LOAN = 0 THEN 'CONSUMER LOAN ONLY'
                  ELSE 'CONSUMER LOAN, ' ||
                      CASE
                        WHEN CREDIT_CARD = 1 THEN 'CREDIT CARD, '
                        ELSE ''
                  END ||
                      CASE
                        WHEN GOLD_LOAN = 1 THEN 'GOLD LOAN, '
                        ELSE ''
                  END ||
                      CASE
                        WHEN HOUSING_LOAN = 1 THEN 'HOUSING LOAN, '
                        ELSE ''
                  END ||
                      CASE
                        WHEN PERSONAL_LOAN = 1 THEN 'PERSONAL LOAN, '
                        ELSE ''
                  END ||
                      CASE
                        WHEN TWO_WHEELER_LOAN = 1 THEN 'TWO WHEELER LOAN'
                        ELSE ''
                  END
            END
            WHEN CREDIT_CARD = 1 THEN
             CASE
                  WHEN GOLD_LOAN + HOUSING_LOAN + PERSONAL_LOAN + TWO_WHEELER_LOAN = 0
THEN 'CREDIT CARD ONLY'
                  ELSE 'CREDIT CARD, ' ||
                      CASE
                        WHEN GOLD_LOAN = 1 THEN 'GOLD LOAN, '
                        ELSE ''
                  END ||
                      CASE
                        WHEN HOUSING_LOAN = 1 THEN 'HOUSING LOAN, '
                        ELSE ''
                  END ||
                      CASE
                        WHEN PERSONAL_LOAN = 1 THEN 'PERSONAL LOAN, '
                        ELSE ''
                  END ||
                      CASE
                        WHEN TWO_WHEELER_LOAN = 1 THEN 'TWO WHEELER LOAN'
                        ELSE ''
                  END
            END
            WHEN GOLD_LOAN = 1 THEN
             CASE
                  WHEN HOUSING_LOAN + PERSONAL_LOAN + TWO_WHEELER_LOAN = 0 THEN 'GOLD LOAN
ONLY'
                  ELSE 'GOLD LOAN, ' ||
                      CASE
                        WHEN HOUSING_LOAN = 1 THEN 'HOUSING LOAN, '
                        ELSE ''
                  END ||
                      CASE
                        WHEN PERSONAL_LOAN = 1 THEN 'PERSONAL LOAN, '
                        ELSE ''
                  END ||
                      CASE
                        WHEN TWO_WHEELER_LOAN = 1 THEN 'TWO WHEELER LOAN'
                        ELSE ''
                  END
```

```
            END
            WHEN HOUSING_LOAN = 1 THEN
             CASE
                  WHEN PERSONAL_LOAN + TWO_WHEELER_LOAN = 0 THEN 'HOUSING LOAN ONLY'
                  ELSE 'HOUSING LOAN, ' ||
                      CASE
                        WHEN PERSONAL_LOAN = 1 THEN 'PERSONAL LOAN, '
                        ELSE ''
                  END ||
                      CASE
                        WHEN TWO_WHEELER_LOAN = 1 THEN 'TWO WHEELER LOAN'
                        ELSE ''
                  END
            END
            WHEN PERSONAL_LOAN = 1 THEN
             CASE
                  WHEN TWO_WHEELER_LOAN = 0 THEN 'PERSONAL LOAN ONLY'
                  ELSE 'PERSONAL LOAN, TWO WHEELER LOAN'
            END
            WHEN TWO_WHEELER_LOAN = 1 THEN 'TWO WHEELER LOAN ONLY'
            ELSE 'NO LOANS'
      END AS CUSTOMER_SEGMENT
FROM
      (
      SELECT
            "CUSTOMER_ID",
            MAX(CASE WHEN "ACCOUNT_CATEGORY" = 'Auto Loan' THEN 1 ELSE 0 END) AS
AUTO_LOAN,
            MAX(CASE WHEN "ACCOUNT_CATEGORY" = 'Consumer Loan' THEN 1 ELSE 0 END) AS
CONSUMER_LOAN,
            MAX(CASE WHEN "ACCOUNT_CATEGORY" = 'Credit Card' THEN 1 ELSE 0 END) AS
CREDIT_CARD,
            MAX(CASE WHEN "ACCOUNT_CATEGORY" = 'Gold Loan' THEN 1 ELSE 0 END) AS
GOLD_LOAN,
            MAX(CASE WHEN "ACCOUNT_CATEGORY" = 'Housing Loan' THEN 1 ELSE 0 END) AS
HOUSING_LOAN,
            MAX(CASE WHEN "ACCOUNT_CATEGORY" = 'Personal Loan' THEN 1 ELSE 0 END) AS
PERSONAL_LOAN,
            MAX(CASE WHEN "ACCOUNT_CATEGORY" = 'Two-Wheeler Loan' THEN 1 ELSE 0 END) AS
TWO_WHEELER_LOAN
      FROM
            TRANSACTION_LINE TL
      GROUP BY
            1
    ) AS CUSTOMER_ACCOUNTS
)
SELECT
      CUSTOMER_SEGMENT,
      COUNT("CUSTOMER_ID") AS TOTAL_CUSTOMER
FROM
      PRODUCT_SEGMENTATION PS
GROUP BY
      1
ORDER BY
      1;


-- 7. Account Activity Segmentation: Segmenting customers by the status of their accounts
(whether they have more active or closed accounts).

SELECT
      "CUSTOMER_ID",
      COUNT(CASE WHEN "ACCOUNT_STATUS" = 'Active' THEN 1 END) AS ACTIVE_ACCOUNTS,
      COUNT(CASE WHEN "ACCOUNT_STATUS" = 'Closed' THEN 1 END) AS CLOSED_ACCOUNTS,
      CASE
```

```
          WHEN COUNT(CASE WHEN "ACCOUNT_STATUS" = 'Closed' THEN 1 END) = 0 THEN 'FULLY
ACTIVE'
          WHEN COUNT(CASE WHEN "ACCOUNT_STATUS" = 'Active' THEN 1 END) > COUNT(CASE WHEN
"ACCOUNT_STATUS" = 'Closed' THEN 1 END) THEN 'MOSTLY ACTIVE'
          WHEN COUNT(CASE WHEN "ACCOUNT_STATUS" = 'Active' THEN 1 END) = COUNT(CASE WHEN
"ACCOUNT_STATUS" = 'Closed' THEN 1 END) THEN 'BALANCED'
          WHEN COUNT(CASE WHEN "ACCOUNT_STATUS" = 'Active' THEN 1 END) < COUNT(CASE WHEN
"ACCOUNT_STATUS" = 'Closed' THEN 1 END) THEN 'MOSTLY INACTIVE'
          WHEN COUNT(CASE WHEN "ACCOUNT_STATUS" = 'Active' THEN 1 END) = 0 THEN 'FULLY
INACTIVE'
     END AS ACTIVITY_SEGMENT
FROM
     TRANSACTION_LINE TL
GROUP BY
     1;


-- 8. Account Activity Segmentation Count
WITH ACTIVITY_SEGMENTS AS (
SELECT
         "CUSTOMER_ID",
         COUNT(CASE WHEN "ACCOUNT_STATUS" = 'Active' THEN 1 END) AS ACTIVE_ACCOUNTS,
         COUNT(CASE WHEN "ACCOUNT_STATUS" = 'Closed' THEN 1 END) AS CLOSED_ACCOUNTS,
         CASE
          WHEN COUNT(CASE WHEN "ACCOUNT_STATUS" = 'Closed' THEN 1 END) = 0 THEN 'FULLY
ACTIVE'
          WHEN COUNT(CASE WHEN "ACCOUNT_STATUS" = 'Active' THEN 1 END) > COUNT(CASE WHEN
"ACCOUNT_STATUS" = 'Closed' THEN 1 END) THEN 'MOSTLY ACTIVE'
          WHEN COUNT(CASE WHEN "ACCOUNT_STATUS" = 'Active' THEN 1 END) = COUNT(CASE WHEN
"ACCOUNT_STATUS" = 'Closed' THEN 1 END) THEN 'BALANCED'
          WHEN COUNT(CASE WHEN "ACCOUNT_STATUS" = 'Active' THEN 1 END) < COUNT(CASE WHEN
"ACCOUNT_STATUS" = 'Closed' THEN 1 END) THEN 'MOSTLY INACTIVE'
          WHEN COUNT(CASE WHEN "ACCOUNT_STATUS" = 'Active' THEN 1 END) = 0 THEN 'FULLY
INACTIVE'
     END AS ACTIVITY_SEGMENT
FROM
         TRANSACTION_LINE TL
GROUP BY
         1
)
SELECT
     ACTIVITY_SEGMENT,
     COUNT("CUSTOMER_ID") AS TOTAL_CUSTOMER
FROM
     ACTIVITY_SEGMENTS
GROUP BY
     1
ORDER BY
     1;

-- 9. Create a cohort to reveal insights into customer retention and the life cycle of
accounts.
-- for example: categorize customers into cohorts based on their account opening dates
and then analyse how these cohorts have behaved over time
-- analyse retention Rates (how long accounts stay open across different cohorts)

SET datestyle = 'DMY';

WITH cohorts AS (
    SELECT
        "CUSTOMER_ID",
        DATE_TRUNC('MONTH',
            CASE
                WHEN "OPENING_DATE" = '' THEN NULL
                ELSE TO_DATE("OPENING_DATE", 'DD-MM-YYYY')
            END
```

```
        )::DATE AS cohort_month,
        "ACCOUNT_CATEGORY",
        COALESCE(
            CASE
                WHEN "CLOSED_DATE" = '' THEN NULL
                ELSE TO_DATE("CLOSED_DATE", 'DD-MM-YYYY')
            END,
            NOW()
        )::DATE AS end_date
    FROM
        transaction_line tl
    WHERE
        "OPENING_DATE" != ''
),
cohort_sizes AS (
    SELECT
        cohort_month,
        "ACCOUNT_CATEGORY",
        COUNT(DISTINCT "CUSTOMER_ID") AS total_customers
    FROM
        cohorts c
    GROUP BY
        1,
        2
),
retention AS (
    SELECT
        c.cohort_month,
        c."ACCOUNT_CATEGORY",
        cs.total_customers,
        COUNT(DISTINCT
                CASE
                    WHEN DATE_TRUNC('MONTH', c.end_date) >= c.cohort_month + INTERVAL '12
MONTHS' THEN c."CUSTOMER_ID"
                END
            ) AS retained_customers_after_12_months
    FROM
        cohorts c
    JOIN cohort_sizes cs ON
        c.cohort_month = cs.cohort_month
        AND c."ACCOUNT_CATEGORY" = cs."ACCOUNT_CATEGORY"
    GROUP BY
        1,
        2,
        3
)
SELECT
    cohort_month,
    "ACCOUNT_CATEGORY",
    total_customers,
    retained_customers_after_12_months,
    ROUND((retained_customers_after_12_months::DECIMAL / total_customers::DECIMAL) * 100,
    2) AS retention_rate_after_12_months
FROM
    retention r
WHERE
    cohort_month <= NOW() - INTERVAL '12 MONTHS'
ORDER BY
    1,
    2;



-- 10. Identify Common Account Combinations: Look at customers who have multiple account
types and identify common combinations.
```

```
-- Predict Likely Next Products: Based on product holding patterns and customer behaviour
within cohorts.

-- This step aggregates data for each customer:
-- 1. Creates an array of unique account types they have
-- 2. Counts the number of different account types
-- 3. Calculates average FICO score, total balance, and maximum tenure
WITH customer_accounts AS (
    SELECT
        "CUSTOMER_ID",
        ARRAY_AGG(DISTINCT "ACCOUNT_CATEGORY" ORDER BY "ACCOUNT_CATEGORY") AS
account_types,
        COUNT(DISTINCT "ACCOUNT_CATEGORY") AS num_account_types,
        ROUND(AVG("FICO_SCORE"),2) AS avg_fico_score,
        SUM("ACCOUNT_BALANCE") AS total_balance,
        MAX("TENURE_MONTHS") AS max_tenure
    FROM transaction_line
    GROUP BY 1
),
-- This identifies the most common combinations of account types:
-- 1. Groups by the array of account types
-- 2. Counts how many customers have each combination
account_combinations AS (
    SELECT
        account_types,
        COUNT(*) AS combination_count
    FROM customer_accounts
    GROUP BY account_types
    ORDER BY COUNT(*) DESC
),
-- This step:
-- 1. Takes the customer account information
-- 2. Adds a new column 'potential_products' which is an array of account types the
customer doesn't currently have
customer_potential AS (
    SELECT
        ca."CUSTOMER_ID",
        ca.account_types,
        ca.num_account_types,
        ca.avg_fico_score,
        ca.total_balance,
        ca.max_tenure,
        (SELECT ARRAY_AGG(DISTINCT ac."ACCOUNT_CATEGORY")
         FROM transaction_line ac
         WHERE ac."ACCOUNT_CATEGORY" != ALL(ca.account_types)) AS potential_products
    FROM customer_accounts ca
)
-- This final step:
-- 1. Selects relevant information for each customer
-- 2. Adds a 'recommended_product' based on simple rules:
      -- 2.1. Credit Card for high FICO score customers without one
      -- 2.2. Investment Account for high balance customers without one
      -- 2.3. Loan Product for long-term customers without one
-- 3. Orders results by total balance and FICO score to prioritize high-value customers
SELECT
    cp."CUSTOMER_ID",
    cp.account_types AS current_products,
    cp.num_account_types,
    cp.avg_fico_score,
    cp.total_balance,
    cp.max_tenure,
    cp.potential_products as recommended_product
FROM customer_potential cp
ORDER BY cp.total_balance DESC, cp.avg_fico_score desc;
```

```
-- Summary of this query:
-- 1. Identify Common Account Combinations: The account_combinations CTE shows the most
common product combinations.
-- 2. Predict Likely Next Products: The potential_products column shows products the
customer doesn't have, and
-- the recommended_product column provides a simple recommendation based on their
profile.
```