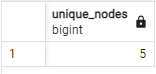
**--A. Customer Nodes Exploration**

***-- 1 How many unique nodes are there on the Data Bank system?***

SELECT COUNT(DISTINCT node\_id) AS unique\_nodes

FROM customer\_nodes;

**RESULT:**

**

***-- 2 What is the number of nodes* *per region?***

SELECT r.region\_name,COUNT(DISTINCT cn.node\_id)

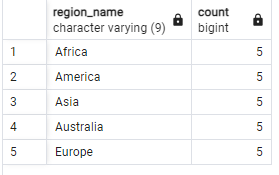
FROM customer\_nodes cn JOIN regions r

USING(region\_id)

GROUP BY r.region\_name

ORDER BY r.region\_name;

**RESULT:**



***-- 3 How many customers are allocated to each region?***

SELECT r.region\_name,COUNT(DISTINCT cn.node\_id)

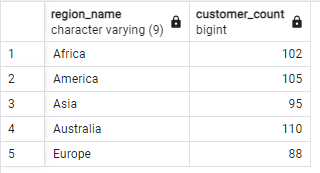
FROM customer\_nodes cn JOIN regions r

USING(region\_id)

GROUP BY r.region\_name

ORDER BY r.region\_name;

**RESULT:**



***-- 4 How many days on average are customers reallocated to a different node?***

WITH DAYS\_IN\_NODE AS (

SELECT

customer\_id,

node\_id,

SUM(DATEDIFF('days',start\_date,end\_date)) as days\_in\_node

FROM customer\_nodes

WHERE end\_date <> '9999-12-31'

GROUP BY customer\_id,

node\_id

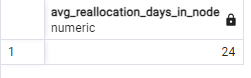
)

SELECT

ROUND(AVG(days\_in\_node),0) as average\_days\_in\_node

FROM DAYS\_IN\_NODE;

**RESULT:**



***-- 5 What is the median, 80th and 95th percentile for this same reallocation days metric for each region?***

WITH reallocation\_days\_cte AS

(

SELECT

cn.region\_id,

r.region\_name,

cn.customer\_id,

cn.node\_id,

SUM(cn.end\_date-cn.start\_date) AS reallocation\_days

FROM customer\_nodes cn

JOIN regions r

USING(region\_id)

WHERE end\_date <> '9999-12-31'

GROUP BY

cn.region\_id,r.region\_name,

cn.customer\_id,

cn.node\_id

)

SELECT

region\_name,

ROUND(PERCENTILE\_CONT(0.5) WITHIN GROUP (ORDER BY reallocation\_days)) AS median\_days,

ROUND(PERCENTILE\_CONT(0.8) WITHIN GROUP (ORDER BY reallocation\_days)) AS percentile\_80\_days,

ROUND(PERCENTILE\_CONT(0.95) WITHIN GROUP (ORDER BY reallocation\_days)) AS percentile\_95\_days

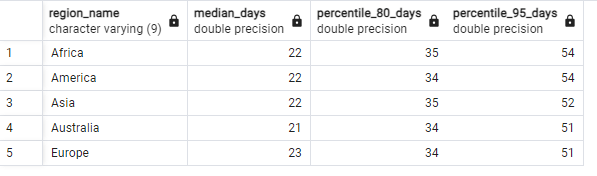
FROM

reallocation\_days\_cte

GROUP BY region\_name

ORDER BY region\_name;

**RESULT:**



**--B. Customer Transactions**

***1 What is the unique count and total amount for each transaction type?***

SELECT

DISTINCT(txn\_type) AS transaction\_type,

COUNT(\*) AS unique\_transaction\_count,

SUM(txn\_amount) AS total\_amount

FROM customer\_transactions

GROUP BY transaction\_type

ORDER BY transaction\_type;

**RESULT:**



***-- 2 What is the average total historical deposit counts and amounts for all customers?***

WITH CTE AS (

SELECT

customer\_id,

AVG(txn\_amount) as avg\_deposit,

COUNT(\*) as transaction\_count

FROM customer\_transactions

WHERE txn\_type = 'deposit'

GROUP BY customer\_id

)

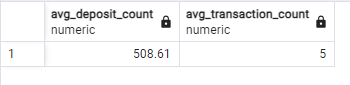
SELECT

ROUND(AVG(avg\_deposit),2) as avg\_deposit\_amount,

ROUND(AVG(transaction\_count),0) as avg\_transactions

FROM CTE;

**RESULT:**



***-- 3 For each month - how many Data Bank customers make more than 1 deposit and either 1 purchase or 1 withdrawal in a single month?***

WITH customer\_monthly\_transactions AS (

SELECT

customer\_id,

EXTRACT(YEAR FROM txn\_date) AS txn\_year,

EXTRACT(MONTH FROM txn\_date) AS txn\_month,

SUM(CASE WHEN txn\_type = 'deposit' THEN 1 ELSE 0 END) AS deposit\_count,

SUM(CASE WHEN txn\_type = 'purchase' THEN 1 ELSE 0 END) AS purchase\_count,

SUM(CASE WHEN txn\_type = 'withdrawal' THEN 1 ELSE 0 END) AS withdrawal\_count

FROM

customer\_transactions

GROUP BY

customer\_id, EXTRACT(YEAR FROM txn\_date), EXTRACT(MONTH FROM txn\_date)

)

-- Filter customers who meet the criteria

SELECT

txn\_year,

txn\_month,

COUNT(DISTINCT customer\_id) AS customer\_count

FROM

customer\_monthly\_transactions

WHERE

deposit\_count > 1

AND (purchase\_count >= 1 OR withdrawal\_count >= 1)

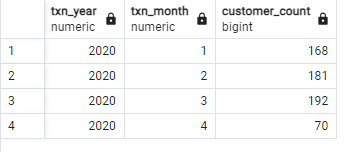
GROUP BY

txn\_year, txn\_month

ORDER BY

txn\_year, txn\_month;

**RESULT:**



***-- 4 What is the closing balance for each customer at the end of the month?***

WITH cust\_monthly\_trans AS (

SELECT

customer\_id,

EXTRACT(YEAR FROM txn\_date) AS txn\_yr,

EXTRACT(MONTH FROM txn\_date) AS txn\_month,

TO\_CHAR(txn\_date, 'Month') AS txn\_month\_name,

SUM(CASE

WHEN txn\_type = 'deposit' THEN txn\_amount

WHEN txn\_type = 'withdrawal' THEN -txn\_amount

ELSE 0

END) AS monthly\_balance

FROM

customer\_transactions

GROUP BY

customer\_id, EXTRACT(YEAR FROM txn\_date), EXTRACT(MONTH FROM txn\_date), TO\_CHAR(txn\_date, 'Month')

),

running\_balance AS

(

SELECT customer\_id,

txn\_yr,

txn\_month,

txn\_month\_name,

SUM(monthly\_balance) OVER(PARTITION BY customer\_id

ORDER BY txn\_yr,txn\_month

ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW

) AS closing\_balance

FROM cust\_monthly\_trans

)

SELECT

customer\_id,

txn\_yr,

txn\_month,

txn\_month\_name,

closing\_balance

FROM running\_balance

ORDER BY

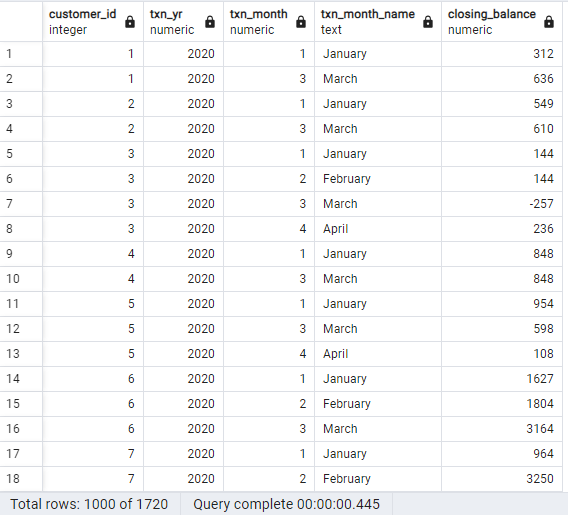
customer\_id,

txn\_yr,

txn\_month,

txn\_month\_name;

**RESULT:**



**--SOL2:**

WITH monthly\_transactions AS (

SELECT

customer\_id,

DATE\_TRUNC('month', txn\_date) AS txn\_month,

TO\_CHAR(txn\_date, 'Month') AS month\_name,

SUM(

CASE

WHEN txn\_type = 'deposit' THEN txn\_amount

WHEN txn\_type = 'withdrawal' THEN -txn\_amount

ELSE 0 -- Handle other transaction types as needed

END

) AS net\_change

FROM

customer\_transactions

GROUP BY

customer\_id,

DATE\_TRUNC('month', txn\_date),

TO\_CHAR(txn\_date, 'Month')

),

cumulative\_balances AS (

SELECT

customer\_id,

txn\_month,

month\_name,

net\_change,

SUM(net\_change) OVER (

PARTITION BY customer\_id

ORDER BY txn\_month

ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW

) AS closing\_balance

FROM

monthly\_transactions

)

SELECT

customer\_id,

TO\_CHAR(txn\_month + INTERVAL '1 month' - INTERVAL '1 day', 'YYYY-MM-DD') AS end\_of\_month,

closing\_balance

FROM

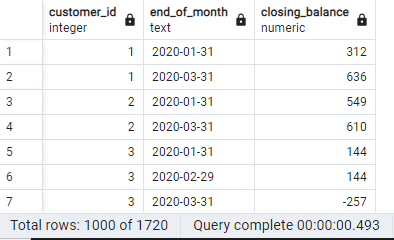
cumulative\_balances

ORDER BY

customer\_id,

txn\_month;

**RESULT:**



***- 5 What is the percentage of customers who increase their closing balance by more than 5%?***

WITH monthly\_balances AS (

-- Step 1: Get closing balances for each customer at the end of each month

SELECT

customer\_id,

DATE\_TRUNC('month', txn\_date) AS txn\_month,

SUM(

CASE

WHEN txn\_type = 'deposit' THEN txn\_amount

WHEN txn\_type = 'withdrawal' THEN -txn\_amount

ELSE 0

END

) AS closing\_balance

FROM

customer\_transactions

GROUP BY

customer\_id,

DATE\_TRUNC('month', txn\_date)

ORDER BY

customer\_id,

DATE\_TRUNC('month', txn\_date)

),

balance\_changes AS (

-- Step 2: Calculate the percentage change in balance compared to the previous month

SELECT

customer\_id,

txn\_month,

closing\_balance,

LAG(closing\_balance) OVER (PARTITION BY customer\_id ORDER BY txn\_month) AS previous\_balance,

CASE

WHEN LAG(closing\_balance) OVER (PARTITION BY customer\_id ORDER BY txn\_month) > 0

THEN (closing\_balance - LAG(closing\_balance) OVER (PARTITION BY customer\_id ORDER BY txn\_month)) / LAG(closing\_balance) OVER (PARTITION BY customer\_id ORDER BY txn\_month) \* 100

ELSE NULL -- Handle cases where the previous balance is 0 or doesn't exist

END AS percentage\_change

FROM

monthly\_balances

),

customers\_with\_increase AS (

-- Step 3: Identify customers whose closing balance increased by more than 5%

SELECT

customer\_id

FROM

balance\_changes

WHERE

percentage\_change > 5

GROUP BY

customer\_id

),

total\_customers AS (

-- Step 4: Count total distinct customers

SELECT

COUNT(DISTINCT customer\_id) AS total\_customer\_count

FROM

customer\_transactions

)

-- Step 5: Calculate the percentage of customers who increased their balance by more than 5%

SELECT

ROUND( COUNT(DISTINCT cwi.customer\_id) \* 100.0 / tc.total\_customer\_count,2) AS percentage\_increased

FROM

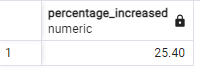
customers\_with\_increase cwi,

total\_customers tc

GROUP BY

tc.total\_customer\_count;

**RESULT:**

******

**--C. Data Allocation Challenge**

/\*To test out a few different hypotheses - the Data Bank team wants to run an experiment where different groups of customers would be allocated data using 3 different options:

Option 1: data is allocated based off the amount of money at the end of the previous month

Option 2: data is allocated on the average amount of money kept in the account in the previous 30 days

Option 3: data is updated real-time

For this multi-part challenge question - you have been requested to generate the following data elements to help the Data Bank team estimate how much data will need to be provisioned for each option:

running customer balance column that includes the impact each transaction

customer balance at the end of each month

minimum, average and maximum values of the running balance for each customer

Using all of the data available - how much data would have been required for each option on a monthly basis?

\*/

/\*

1. Running Customer Balance (Impact of Each Transaction)

We will calculate a running balance that reflects the impact of each transaction on the customer's account.

2. Customer Balance at the End of Each Month

For Option 1, we need to calculate the customer balance at the end of each month, which will help allocate data based on this balance for the next month.

3. Minimum, Average, and Maximum Running Balance for Each Customer

For Option 2 (average balance over the last 30 days) and for understanding trends (Option 3), we will compute the minimum, average, and maximum running balance for each customer.

\*/

-- ***Step 1:Calculate the Running Balance for Each Customer***

--We first calculate the running balance based on the transactions (deposits and withdrawals) for each customer.

WITH running\_balances AS (

SELECT

customer\_id,

txn\_date,

SUM(

CASE

WHEN txn\_type = 'deposit' THEN txn\_amount

WHEN txn\_type = 'withdrawal' THEN -txn\_amount

ELSE 0

END

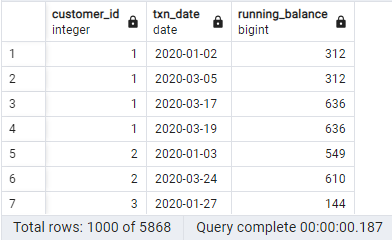
) OVER (PARTITION BY customer\_id ORDER BY txn\_date) AS running\_balance

FROM

customer\_transactions

)

**RESULT:**



***--(Option 1)***

***-- Step 2:Calculate the Customer Balance at the End of Each Month***

--For Option 1, we will use the customer balance at the end of each month.

--The LAG function is used to get the previous month's balance.

WITH end\_of\_month\_balances AS (

SELECT

customer\_id,

DATE\_TRUNC('month', txn\_date) AS txn\_month,

SUM(

CASE

WHEN txn\_type = 'deposit' THEN txn\_amount

WHEN txn\_type = 'withdrawal' THEN -txn\_amount

ELSE 0

END

) AS monthly\_balance,

ROW\_NUMBER() OVER (PARTITION BY customer\_id, DATE\_TRUNC('month', txn\_date) ORDER BY txn\_date DESC) AS rn

FROM

customer\_transactions

GROUP BY

customer\_id,

DATE\_TRUNC('month', txn\_date),

txn\_date

)

SELECT

customer\_id,

txn\_month,

LAG(monthly\_balance) OVER (PARTITION BY customer\_id ORDER BY txn\_month) AS data\_allocated\_for\_next\_month

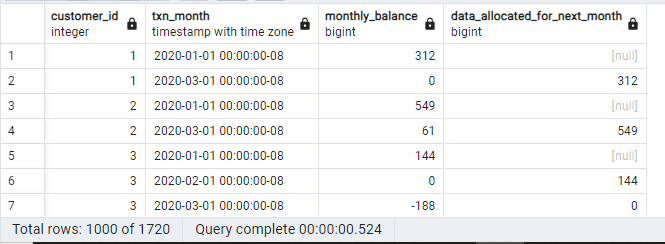
FROM

end\_of\_month\_balances

WHERE

rn = 1;

**RESULT:**



-- ***(Option 2)***

***--Step 3: Calculate Minimum, Average, and Maximum Running Balances***

--For Option 2, where the allocation is based on the average balance over the previous 30 days,

--we calculate the min, max, and average running balances.

WITH running\_balances AS (

SELECT

customer\_id,

txn\_date,

SUM(

CASE

WHEN txn\_type = 'deposit' THEN txn\_amount

WHEN txn\_type = 'withdrawal' THEN -txn\_amount

ELSE 0

END

) OVER (PARTITION BY customer\_id ORDER BY txn\_date) AS running\_balance

FROM

customer\_transactions

)

SELECT

customer\_id,

MIN(running\_balance) AS min\_balance,

AVG(running\_balance) AS avg\_balance,

MAX(running\_balance) AS max\_balance

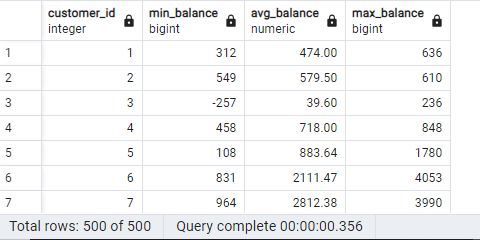
FROM

running\_balances

GROUP BY

customer\_id;

**RESULT:**



***--(Option 3)***

***--Step 4: Calculate Real-Time Balance***

--For Option 3, we need the total real-time running balance per day.

WITH daily\_balances AS (

SELECT

customer\_id,

txn\_date,

SUM(

CASE

WHEN txn\_type = 'deposit' THEN txn\_amount

WHEN txn\_type = 'withdrawal' THEN -txn\_amount

ELSE 0

END

) OVER (PARTITION BY customer\_id ORDER BY txn\_date) AS running\_balance

FROM

customer\_transactions

)

SELECT

customer\_id,

DATE\_TRUNC('month', txn\_date) AS txn\_month,

SUM(running\_balance) AS total\_real\_time\_balance

FROM

daily\_balances

GROUP BY

customer\_id,

DATE\_TRUNC('month', txn\_date);

**RESULT:**

