SQL
PROJECT

BY ARJIT DABRAL

8WeekSQLChallenge.com

CASE STUDY #4

DATA BANK

That's money.

DataWithDanny.com

INTRODUCTION

There is a new innovation in the financial industry called Neo-Banks: new aged digital only banks without physical branches.
Danny thought that there should be some sort of intersection between these new age banks, cryptocurrency and the data world...so he decides to launch a new initiative - Data Bank!
Data Bank runs just like any other digital bank - but it isn't only for banking activities, they also have the world's most secure distributed data storage platform!
Customers are allocated cloud data storage limits which are directly linked to how much money they have in their accounts. There are a few interesting caveats that go with this business model, and this is where the Data Bank team need your help!
The management team at Data Bank want to increase their total customer base - but also need some help tracking just how much data storage their customers will need.
This case study is all about calculating metrics, growth and helping the business analyse their data in a smart way to better forecast and plan for their future developments!

DATA MODEL

The Data Bank team have prepared a data model for this case study as well as a few example rows from the complete dataset below to get you familiar with their tables.

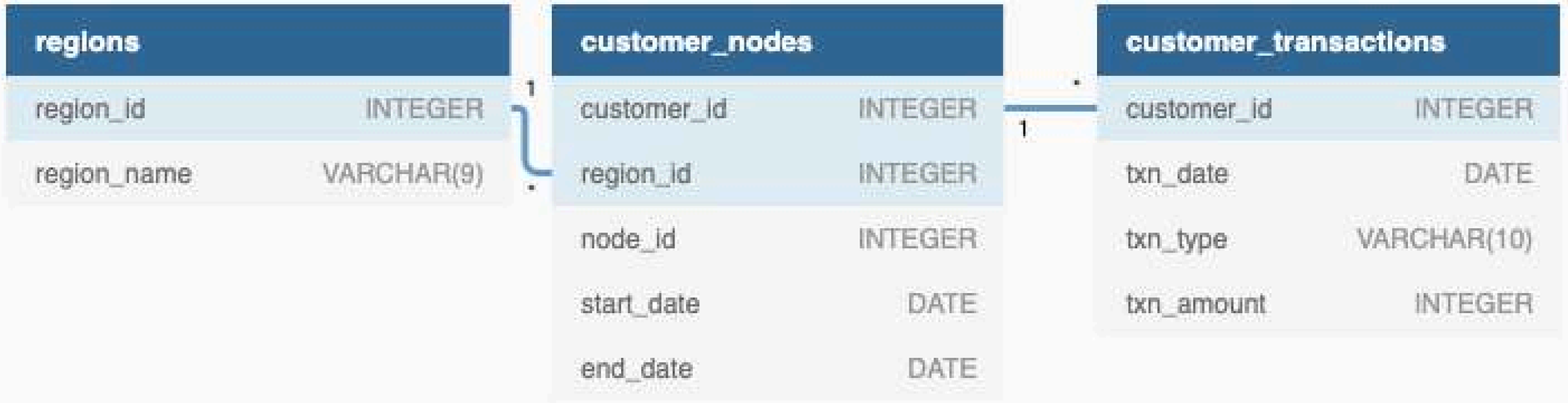


Table : regions

Just like popular cryptocurrency platforms - Data Bank is also run off a network of nodes where both money and data is stored across the globe. In a traditional banking sense - you can think of these nodes as bank branches or stores that exist around the world.

This regions table contains the region_id and their respective region_name values

region_id	region_name
1	Africa
2	America
3	Asia
4	Europe
5	Oceania

Table : customer_nodes

Customers are randomly distributed across the nodes according to their region - this also specifies exactly which node contains both their cash and data.
This random distribution changes frequently to reduce the risk of hackers getting into Data Bank's system and stealing customer's money and data!

Below is a sample of the top 10 rows of the data_bank.customer_nodes

customer_id	region_id	node_id	start_date	end_date
1	3	4	2020-01-02	2020-01-03
2	3	5	2020-01-03	2020-01-17
3	5	4	2020-01-27	2020-02-18
4	5	4	2020-01-07	2020-01-19
5	3	3	2020-01-15	2020-01-23
6	1	1	2020-01-11	2020-02-06
7	2	5	2020-01-20	2020-02-04
8	1	2	2020-01-15	2020-01-28
9	4	5	2020-01-21	2020-01-25
10	3	4	2020-01-13	2020-01-14

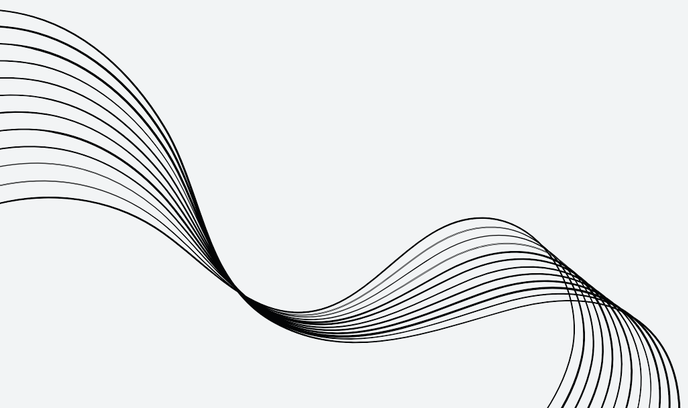


Table : customer_transactions

This table stores all customer deposits, withdrawals and purchases made using their Data Bank debit card.

customer_id	txn_date	txn_type	txn_amount
429	2020-01-21	deposit	82
155	2020-01-10	deposit	712
398	2020-01-01	deposit	196
255	2020-01-14	deposit	563
185	2020-01-29	deposit	626
309	2020-01-13	deposit	995
312	2020-01-20	deposit	485
376	2020-01-03	deposit	706
188	2020-01-13	deposit	601
138	2020-01-11	deposit	520

CASE STUDY QUESTIONS

The following case study questions include some general data exploration analysis for the nodes and transactions before diving right into the core business questions and finishes with a challenging final request!

There are 3 Parts to this project

Customer Node Exploration

Customer Transaction

Data Allocation Challenge

Each part contains few data analysis tasks that are done using my Sql by me

CASE STUDY QUESTIONS

Customer Nodes Exploration

1. How many unique nodes are there on the Data Bank system?
2. What is the number of nodes per region?
3. How many customers are allocated to each region?
4. How many days on average are customers reallocated to a different node?
5. What is the median, 80th and 95th percentile for this same reallocation days metric for each region?

CUSTOMER NODES EXPLORATION

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

```
/*distinct node_id*/
select distinct count(node_id) from customer_nodes;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	count(node_id)
▶	3500

CUSTOMER NODES EXPLORATION

What is the number of nodes per region?

```sql
/*nodes per region */
select
  count(cn.node_id) as number_of_nodes,
   r.region_name
    from customer_nodes as cn
    join
    regions as r using(region_id)
   group by r.region_name;
```

Result Grid | Filter Rows: | Export:

number_of_nodes	region_name
714	Africa
616	Europe
770	Australia
735	America
665	Asia

CUSTOMER NODES EXPLORATION

How many customers are allocated to each region?

```sql
/*customers allocated to each region*/
select
 count(distinct cn.customer_id) as number_of_customers,
 r.region_name
  from customer_nodes as cn
  join
  regions as r using(region_id)
 group by r.region_name;
```

Result Grid | Filter Rows: | Export:

number_of_customers	region_name
102	Africa
105	America
95	Asia
110	Australia
88	Europe

CUSTOMER NODES EXPLORATION

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

```sql
 /* average allocation days of customers*/

SELECT
    AVG(DATEDIFF(end_date, start_date)) AS average_allocation_days
FROM customer_nodes
WHERE end_date != '9999-12-31';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Co

average_allocation_days
14.6340

CUSTOMER NODES EXPLORATION

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

```sql
/*What is the median, 80th and 95th percentile for this same reallocation days metric for each region*/
WITH cte AS (
    SELECT
        r.region_name,
        DATEDIFF(cn.end_date, cn.start_date) AS allocation_days,
        PERCENT_RANK() OVER (PARTITION BY r.region_name ORDER BY DATEDIFF(cn.end_date, cn.start_date)) * 100 AS percentile_rank
    FROM
        customer_nodes AS cn
    JOIN
        regions AS r USING (region_id)
    WHERE
        cn.end_date != '9999-12-31'
),
```

Query continues on next page

CUSTOMER NODES EXPLORATION

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

```sql
-- Get the closest value to each desired percentile for each region
percentiles AS (
    SELECT
        region_name,

        -- Median
        MIN(CASE WHEN percentile_rank >= 50 THEN allocation_days END) AS median_allocation_days,

        -- 80th Percentile
        MIN(CASE WHEN percentile_rank >= 80 THEN allocation_days END) AS percentile_80_allocation_days,

        -- 95th Percentile
        MIN(CASE WHEN percentile_rank >= 95 THEN allocation_days END) AS percentile_95_allocation_days

    FROM
        cte
    GROUP BY
        region_name
)
SELECT * FROM percentiles;
```

Output on next page

CUSTOMER NODES EXPLORATION

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

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

region_name	median_allocation_days	percentile_80_allocation_days	percentile_95_allocation_days
Africa	16	25	29
America	16	24	29
Asia	16	24	29
Australia	16	24	29
Europe	16	25	29

CASE STUDY QUESTIONS

Customer Transactions

1. What is the unique count and total amount for each transaction type?
2. What is the average total historical deposit counts and amounts for all customers?
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?
4. What is the closing balance for each customer at the end of the month?
5. What is the percentage of customers who increase their closing balance by more than 5%?

CUSTOMER TRANSACTIONS

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

```sql
/*What is the unique count and total amount for each transaction type*/

select
     txn_type,
     count(txn_type) as unique_count,
     sum(txn_amount) as total_amount
     from customer_transactions
     group by txn_type;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

txn_type	unique_count	total_amount
deposit	2671	1359168
withdrawal	1580	793003
purchase	1617	806537

CUSTOMER TRANSACTIONS

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

```sql
/*What is the average total historical deposit counts and amounts for all customers*/

with cte as(select
     customer_id,
     count(txn_type) as unique_count,
     sum(txn_amount) as total_amount
     from customer_transactions
     where txn_type ='deposit'
     group by customer_id
     order by customer_id

)
select
     avg(unique_count) as average_deposit_count,
     avg(total_amount) as average_total_amount
     from cte;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

average_deposit_count	average_total_amount
5.3420	2718.3360

CUSTOMER TRANSACTIONS

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

```
/*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 transaction_count_per_month_cte AS
  (SELECT customer_id,
          month(txn_date) AS txn_month,
          SUM(IF(txn_type="deposit", 1, 0)) AS deposit_count,
          SUM(IF(txn_type="withdrawal", 1, 0)) AS withdrawal_count,
          SUM(IF(txn_type="purchase", 1, 0)) AS purchase_count
   FROM customer_transactions
   GROUP BY customer_id,
            month(txn_date))
SELECT txn_month,
       count(DISTINCT customer_id) as customer_count
FROM transaction_count_per_month_cte
WHERE deposit_count>1
  AND (purchase_count = 1
       OR withdrawal_count = 1)
GROUP BY txn_month;
```

Output on next page

CUSTOMER TRANSACTIONS

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

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

txn_month	customer_count
1	115
2	108
3	113
4	50

CUSTOMER TRANSACTIONS

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

```sql
/*What is the closing balance for each customer at the end of the month?*/

   WITH txn_monthly_balance_cte AS
  (SELECT customer_id,
          month(txn_date) AS txn_month,
          SUM(CASE
                  WHEN txn_type="deposit" THEN txn_amount
                  ELSE -txn_amount
              END) AS net_transaction_amt
   FROM customer_transactions
   GROUP BY customer_id,month(txn_date)
   ORDER BY customer_id)
SELECT customer_id,
       txn_month,
       net_transaction_amt,
       sum(net_transaction_amt) over(PARTITION BY customer_id
                                     ORDER BY txn_month ROWS BETWEEN UNBOUNDED preceding AND CURRENT ROW) AS closing_balance
FROM txn_monthly_balance_cte;
```

Output on next page

CUSTOMER TRANSACTIONS

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

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

customer_id	txn_month	net_transaction_amt	closing_balance
1	1	312	312
1	3	-952	-640
2	1	549	549
2	3	61	610
3	1	144	144
3	2	-965	-821
3	3	-401	-1222
3	4	493	-729
4	1	848	848
4	3	-193	655
5	1	954	954
5	3	-2877	-1923
5	4	-490	-2413
6	1	733	733
6	2	-785	-52
6	3	392	340
7	1	964	964
7	2	2209	3173
7	3	-640	2533
7	4	90	2623
8	1	587	587
8	2	-180	407
8	3	-464	-57
8	4	-972	-1029

CUSTOMER TRANSACTIONS

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

```
/*What is the percentage of customers who increase their closing balance by more than 5%?*/
 WITH txn_monthly_balance_cte AS
  (SELECT customer_id,
          month(txn_date) AS txn_month,
          SUM(CASE
                  WHEN txn_type="deposit" THEN txn_amount
                  ELSE -txn_amount
              END) AS net_transaction_amt
   FROM customer_transactions
   GROUP BY customer_id,month(txn_date)
   ORDER BY customer_id),
cte2 as(SELECT customer_id,
       txn_month,
       net_transaction_amt,
       sum(net_transaction_amt) over(PARTITION BY customer_id
                                     ORDER BY txn_month ROWS BETWEEN UNBOUNDED preceding AND CURRENT ROW) AS closing_balance,
       rank() over(partition by customer_id order by txn_month asc) as rank_asc,
       rank() over(partition by customer_id order by txn_month desc) as rank_desc

FROM txn_monthly_balance_cte),
```

Query continues on next page.....

CUSTOMER TRANSACTIONS

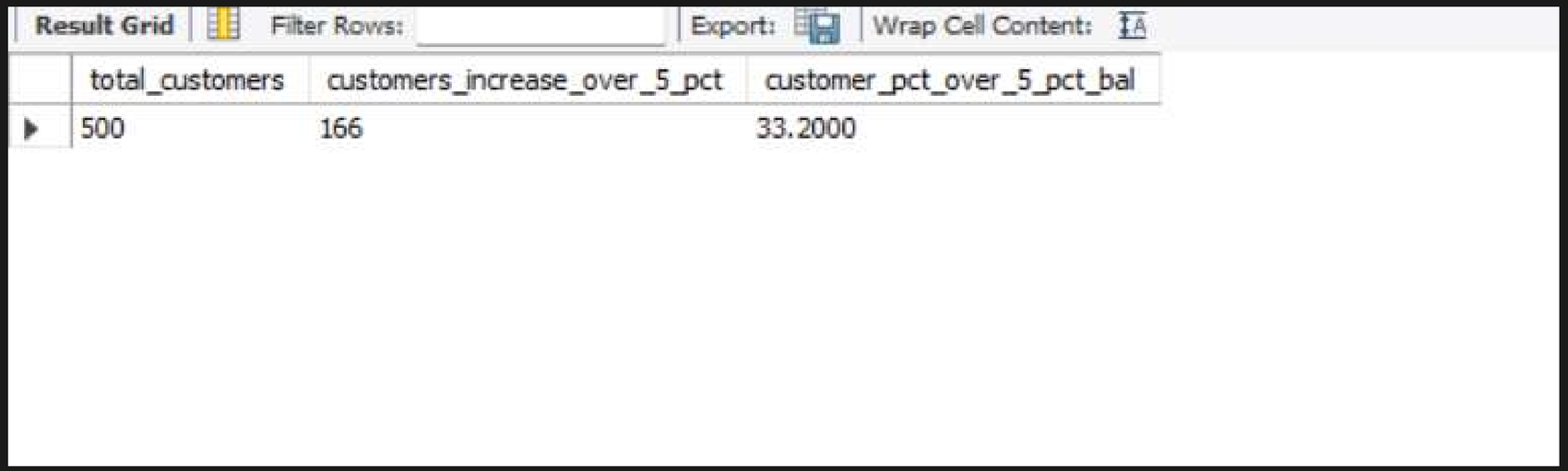
What is the percentage of customers who increase their closing balance by more than 5%?

```
cte3 as (select
    customer_id,
     max(case when rank_asc =1 then closing_balance end) as initial_balance,
     max(case when rank_desc =1 then closing_balance end) as final_balance
     from cte2
     group by customer_id
     ),
   cte4 as (select
        *,
         (final_balance-initial_balance)/abs(initial_balance) * 100 as pct_chg
        from cte3
        )
select
     count(customer_id) as total_customers,
     sum(if(pct_chg>5,1,0)) as customers_increase_over_5_pct,
    sum(if(pct_chg>5,1,0)) * 100 / count(customer_id)   as customer_pct_over_5_pct_bal
     from cte4;
```

Output on next page

CUSTOMER TRANSACTIONS

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


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	total_customers	customers_increase_over_5_pct	customer_pct_over_5_pct_bal
▶	500	166	33.2000

CASE STUDY QUESTIONS

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?

DATA ALLOCATION CHALLENGE

Running customer balance column that includes the impact each transaction

```
/*running customer balance that includes the impact each transaction*/
 select * from customer_transactions order by customer_id,txn_type,txn_date;

 WITH txn_running_balance_cte AS
  (SELECT customer_id,
       txn_date,
       max(txn_type) as transaction_type,
          SUM(CASE
                  WHEN txn_type="deposit" THEN txn_amount
                  ELSE -txn_amount
              END) net_transaction_amt
   FROM customer_transactions
   GROUP BY customer_id,txn_date
   ORDER BY customer_id)
SELECT customer_id,
       txn_date,
       transaction_type,
       net_transaction_amt,
       sum(net_transaction_amt) over(PARTITION BY customer_id
                                     ORDER BY txn_date ROWS BETWEEN UNBOUNDED preceding AND CURRENT ROW) AS Running_balance
FROM txn_running_balance_cte;
```

Output on next page

DATA ALLOCATION CHALLENGE

Running customer balance column that includes the impact each transaction

customer_id	txn_date	transaction_type	net_transaction_amt	Running_balance
1	2020-01-02	deposit	312	312
1	2020-03-05	purchase	-612	-300
1	2020-03-17	deposit	324	24
1	2020-03-19	purchase	-664	-640
2	2020-01-03	deposit	549	549
2	2020-03-24	deposit	61	610
3	2020-01-27	deposit	144	144
3	2020-02-22	purchase	-965	-821
3	2020-03-05	withdrawal	-213	-1034
3	2020-03-19	withdrawal	-188	-1222
3	2020-04-12	deposit	493	-729
4	2020-01-07	deposit	458	458
4	2020-01-21	deposit	390	848
4	2020-03-25	purchase	-193	655
5	2020-01-15	deposit	974	974
5	2020-01-25	deposit	806	1780
5	2020-01-31	withdrawal	-826	954
5	2020-03-02	purchase	-886	68
5	2020-03-19	deposit	718	786
5	2020-03-26	withdrawal	-786	0
5	2020-03-27	withdrawal	-288	-288
5	2020-03-29	purchase	-852	-1140
5	2020-03-31	purchase	-783	-1923

DATA ALLOCATION CHALLENGE

Customer balance at the end of each month

```sql
/*customer balance at end of each month*/
 WITH txn_monthly_balance_cte AS
  (SELECT customer_id,
          month(txn_date) AS txn_month,
          SUM(CASE
                  WHEN txn_type="deposit" THEN txn_amount
                  ELSE -txn_amount
              END) AS net_transaction_amt
   FROM customer_transactions
   GROUP BY customer_id,month(txn_date)
   ORDER BY customer_id)
SELECT customer_id,
       txn_month,
       net_transaction_amt,
       sum(net_transaction_amt) over(PARTITION BY customer_id
                                     ORDER BY txn_month ROWS BETWEEN UNBOUNDED preceding AND CURRENT ROW) AS closing_balance
FROM txn_monthly_balance_cte;
```

Output on next page

DATA ALLOCATION CHALLENGE

Customer balance at the end of each month

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	txn_month	net_transaction_amt	closing_balance
1	1	312	312
1	3	-952	-640
2	1	549	549
2	3	61	610
3	1	144	144
3	2	-965	-821
3	3	-401	-1222
3	4	493	-729
4	1	848	848
4	3	-193	655
5	1	954	954
5	3	-2877	-1923
5	4	-490	-2413
6	1	733	733
6	2	-785	-52
6	3	392	340
7	1	964	964
7	2	2209	3173
7	3	-640	2533
7	4	90	2623
8	1	587	587
8	2	-180	407
8	3	-464	-57
8	4	-972	-1029

DATA ALLOCATION CHALLENGE

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

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

 WITH txn_running_balance_cte AS
  (SELECT customer_id,
       txn_date,
       max(txn_type) as transaction_type,
          SUM(CASE
                  WHEN txn_type="deposit" THEN txn_amount
                  ELSE -txn_amount
              END) net_transaction_amt
   FROM customer_transactions
   GROUP BY customer_id,txn_date
   ORDER BY customer_id),
running_balance as (SELECT customer_id,
       txn_date,
       transaction_type,
       net_transaction_amt,
       sum(net_transaction_amt) over(PARTITION BY customer_id
                                     ORDER BY txn_date ROWS BETWEEN UNBOUNDED preceding AND CURRENT ROW) AS Running_balance
FROM txn_running_balance_cte)
```

Query continues on next page....

DATA ALLOCATION CHALLENGE

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

```
select
      customer_id,
      min(Running_balance) as minimum_running_bal,
      max(Running_balance) as max_running_bal,
      avg(Running_balance) as avg_running_bal
    from
      running_balance
    group by customer_id;
```

Output on next page

DATA ALLOCATION CHALLENGE

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

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	minimum_running_bal	max_running_bal	avg_running_bal
1	-640	312	-151.0000
2	549	610	579.5000
3	-1222	144	-732.4000
4	458	848	653.6667
5	-2413	1780	-120.2000
6	-552	2197	654.5882
7	887	3539	2268.6923
8	-1029	1363	173.7000
9	-91	2030	1021.7000
10	-5090	556	-2164.1875
11	-2529	60	-1914.0833
12	-647	295	-14.5000
13	379	1444	869.6667
14	205	989	671.6667
15	379	1102	740.5000
16	-4284	421	-1985.2857
17	-892	465	-292.3333
18	-1216	757	-539.7500
19	-301	258	-41.4286
20	465	1017	783.2857
21	-3253	326	-1038.1875
22	-1444	794	-235.5263
23	-678	520	-33.3333

DATA ALLOCATION CHALLENGE

Data is allocated based off the amount of money at the end of the previous month

```
/*Data is allocated based off the amount of money at the end of the previous month*/
WITH txn_monthly_balance_cte AS
  (SELECT customer_id,
          month(txn_date) AS txn_month,
          SUM(CASE
                  WHEN txn_type="deposit" THEN txn_amount
                  ELSE -txn_amount
              END) AS net_transaction_amt
   FROM customer_transactions
   GROUP BY customer_id,month(txn_date)
   ORDER BY customer_id),
month_end_bal as (SELECT customer_id,
       txn_month,
       net_transaction_amt,
       sum(net_transaction_amt) over(PARTITION BY customer_id
                                     ORDER BY txn_month ROWS BETWEEN UNBOUNDED preceding AND CURRENT ROW) AS closing_balance
FROM txn_monthly_balance_cte
 )
```

Query continues on next page....

DATA ALLOCATION CHALLENGE

Data is allocated based off the amount of money at the end of the previous month

```
select
    txn_month,
    sum(if(closing_balance>0,closing_balance,0)) as data_required_per_month
  from month_end_bal
  group by txn_month
  order by txn_month;
```

txn_month	data_required_per_month
1	235595
2	238492
3	240065
4	157033

DATA ALLOCATION CHALLENGE

Data is allocated on the average amount of money kept in the account in the previous 30 days

```
/* Data is allocated on the average amount of money kept in the account in the previous 30 days */

 WITH txn_running_balance_cte AS
  (SELECT customer_id,
       txn_date,
       max(txn_type) as transaction_type,
          SUM(CASE
                  WHEN txn_type="deposit" THEN txn_amount
                  ELSE -txn_amount
              END) net_transaction_amt
   FROM customer_transactions
   GROUP BY customer_id,txn_date
   ORDER BY customer_id),
running_balance as (SELECT customer_id,
       txn_date,
       transaction_type,
       net_transaction_amt,
       sum(net_transaction_amt) over(PARTITION BY customer_id
                                     ORDER BY txn_date ROWS BETWEEN UNBOUNDED preceding AND CURRENT ROW) AS Running_balance
FROM txn_running_balance_cte),
```

Query continues on next page....

DATA ALLOCATION CHALLENGE

Data is allocated on the average amount of money kept in the account in the previous 30 days

```
avg_balance_cte as(
    select
          customer_id,
          month(txn_date) as txn_month,
          avg(Running_balance)  AS avg_balance

   from running_balance
   group by customer_id,month(txn_date)
   order by customer_id
)
select
     txn_month,
     round(sum(if(avg_balance>0,avg_balance,0))) as monthly_data_req
 from avg_balance_cte
   group by txn_month
   order by txn_month;
```

Result Grid | Filter Rows: | Export:

txn_month	monthly_data_req
1	225658
2	230421
3	234904
4	150552

DATA ALLOCATION CHALLENGE

Data is updated real-time

```
/* Data is updated real-time*/
 WITH txn_running_balance_cte AS
 (SELECT customer_id,
      txn_date,
      max(txn_type) as transaction_type,
         SUM(CASE
                WHEN txn_type="deposit" THEN txn_amount
                ELSE -txn_amount
            END) net_transaction_amt
  FROM customer_transactions
  GROUP BY customer_id,txn_date
  ORDER BY customer_id),
running_balance as (SELECT customer_id,
      txn_date,
      month(txn_date) as txn_month,
      transaction_type,
      net_transaction_amt,
      sum(net_transaction_amt) over(PARTITION BY customer_id
                                    ORDER BY txn_date ROWS BETWEEN UNBOUNDED preceding AND CURRENT ROW) AS Running_balance
FROM txn_running_balance_cte)
```

Query continues on next page....

DATA ALLOCATION CHALLENGE

Data is updated real-time

```
select
      txn_month,
      sum(if(running_balance>0,running_balance,0)) as monthly_data_req
  from running_balance
  group by txn_month
  order by txn_month
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

txn_month	monthly_data_req
1	656456
2	907818
3	876809
4	367709

EXTENSION REQUEST

The Data Bank team wants you to use the outputs generated from the above sections to create a quick Powerpoint presentation which will be used as marketing materials for both external investors who might want to buy Data Bank shares and new prospective customers who might want to bank with Data Bank.

1. Using the outputs generated from the customer node questions, generate a few headline insights which Data Bank might use to market it’s world-leading security features to potential investors and customers.
2. With the transaction analysis - prepare a 1 page presentation slide which contains all the relevant information about the various options for the data provisioning so the Data Bank management team can make an informed decision.

HEADLINE INSIGHTS FOR INVESTORS AND CUSTOMERS

Security and Reliability:

“Data Bank’s innovative system reassigns customer nodes frequently, significantly enhancing security by minimizing exposure to potential breaches. With regular node reallocations, Data Bank ensures a reliable and resilient environment for customer assets.”

Customer Behavior Analytics:

"Data Bank’s in-depth transaction analytics allow us to anticipate customer needs, providing a seamless experience with monthly deposit and transaction patterns that show on an average over **19.3%** of active users utilizing multiple services each month."

Financial Growth and Customer Loyalty:

"Our analysis reveals that **33.2%** of customers increased their balances by over 5% month-over-month, showcasing Data Bank's potential for helping clients grow their finances steadily while encouraging long-term loyalty."

DATA PROVISIONING OPTIONS FOR EFFICIENT RESOURCE ALLOCATION

Provisioning Option	Methodology	Estimated monthly data requirement
1: End-of-Previous Month	Based on customer balances at end-of-month	217796
2: 30-Day Average Balance	Based on the average daily balance	210383
3: Real-Time Updates	Updated with each transaction	702198

DATA PROVISIONING OPTIONS SUMMARY

The Data Bank team has evaluated three provisioning options for efficient data management and customer experience:

1. **Option 1 (End-of-Month Balance)**: Requires **217,796** units of data per month.
   - Benefit: Efficient use of data while providing reliable end-of-month balance updates.
   - Ideal for: Customers interested in monthly financial overviews with minimal data costs.
2. **Option 2 (30-Day Average Balance)**: Requires **210,383** units of data per month.
   - Benefit: Reduces data load with a rolling average approach, offering balanced accuracy and data efficiency.
   - Ideal for: Customers valuing a consistent balance overview without needing real-time updates.
3. **Option 3 (Real-Time Balance Updates)**: Requires **702,198** units of data per month.
   - Benefit: Highest accuracy with real-time updates, ensuring customer balances are always current.
   - Ideal for: High-value accounts or customers who prioritize immediate access to updated balance information.

Recommendation

For a cost-effective yet reliable experience, Option 2 is recommended for most users. Option 3 is suitable for premium accounts where real-time updates are crucial, while Option 1 offers a monthly overview for customers focused on data efficiency.

THANK'S FOR WATCHING

Created by Arjit Dabral