CASE STUDY #4

8weeksqlchallenge.com

DATA BANK

Case Study by: Satya Ranjan Ray

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!

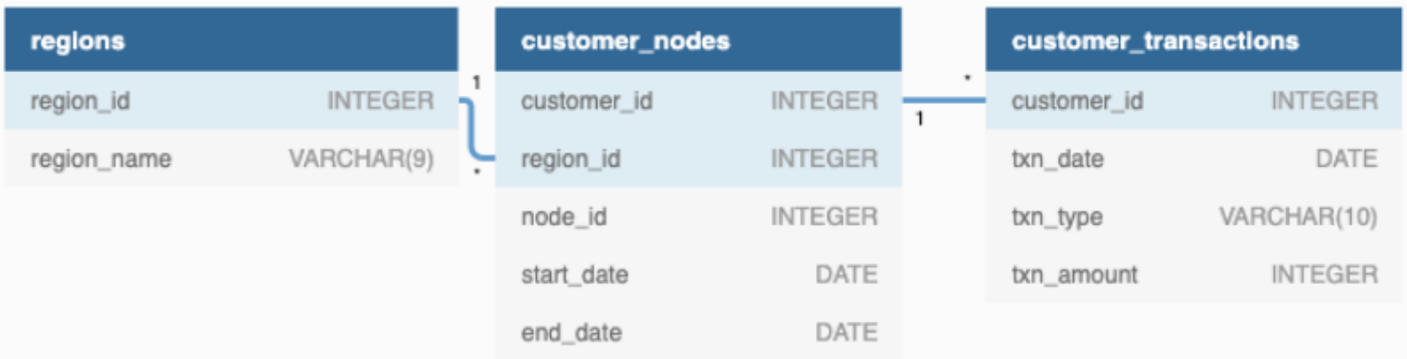
The management team at Data Bank want to increase their total customer base - but also need some help tracking just how much **data storage** their customers will need.

This case study is all about calculating **metrics**, **growth** and helping the business **analyze** their data in a smart way to better forecast and plan for their future developments!

Available Data

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.

Entity Relationship Diagram



Datasets

Table 1: 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 2: 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 **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 3: Customer Transactions

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

Below is a sample of the10 rows of the **customer_transactions**

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

Customer Nodes Exploration

Q1: How many unique nodes are there on the Data Bank system?

```
SELECT COUNT(DISTINCT node_id) AS node_count
FROM customer_nodes;
```

node_count
5

Q2. What is the number of nodes per region?

```
SELECT r.region_name,
       COUNT(DISTINCT cn.node_id) AS regionwise_node_count
FROM regions r
INNER JOIN customer_nodes cn ON r.region_id = cn.region_id
GROUP BY r.region_name;
```

region_name	regionwise_node_count
Africa	5
America	5
Asia	5
Australia	5
Europe	5

Q3: How many customers are allocated to each region?

```
SELECT r.region_name,
       COUNT(DISTINCT cn.customer_id) AS regionwise_customer_count
FROM regions r
INNER JOIN customer_nodes cn ON r.region_id = cn.region_id
GROUP BY r.region_name
ORDER BY regionwise_customer_count DESC;
```

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

Q4: How many days on average are customers reallocated to a different node?

```
SELECT ROUND(AVG(end_date - start_date)) AS average_reallocation_days
FROM customer_nodes
WHERE end_date <>'9999-12-31';
```

average_reallocation_days
15

Q5: What is the median, 80th and 95th percentile for this same reallocation days metric for each region?

```sql
WITH reallocation_cte AS (
  SELECT region_name,
         ROUND(end_date - start_date) AS reallocation_days
  FROM regions r
  INNER JOIN customer_nodes cn ON r.region_id = cn.region_id
  WHERE end_date <> '9999-12-31'
)

SELECT region_name,
       PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY reallocation_days) AS median,
       PERCENTILE_CONT(0.8) WITHIN GROUP (ORDER BY reallocation_days) AS percentile_80th,
       PERCENTILE_CONT(0.95) WITHIN GROUP (ORDER BY reallocation_days) AS percentile_95th
FROM reallocation_cte
GROUP BY region_name;
```

region_name	median	percentile_80th	percentile_95th
Africa	15	24	28
America	15	23	28
Asia	15	23	28
Australia	15	23	28
Europe	15	24	28

Customer Transactions

Q1: What is the unique count and total amount for each transaction type?

```
SELECT txn_type AS transaction_type,
       COUNT(txn_type) AS count,
       TO_CHAR(SUM(txn_amount), 'FM$ 999,999,999') AS total_amount
FROM customer_transactions
GROUP BY txn_type
ORDER BY count DESC;
```

transaction_type	count	total_amount
deposit	2,671	$ 1,359,168
purchase	1,617	$ 806,537
withdrawal	1,580	$ 793,003

Q2: What is the average total historical deposit counts and amounts for all customers?

```
WITH transaction_details AS (
  SELECT customer_id,
         COUNT(txn_type) AS deposit_count,
         AVG(txn_amount) AS avg_amount
  FROM customer_transactions
  WHERE txn_type = 'deposit'
  GROUP BY customer_id
)

SELECT ROUND(AVG(deposit_count)) AS deposit_count,
       CONCAT('$ ', ROUND(AVG(avg_amount), 2)) AS avg_deposit_amount
FROM transaction_details;
```

deposit_count	avg_deposit_amount
5	$ 508.61

Q3: For each month - how many Data Bank customers make more than 1 deposit and either 1 purchase or 1 withdrawal in a single month?

```sql
WITH transaction_details AS (
  SELECT customer_id,
         EXTRACT(MONTH FROM txn_date) AS month,
         UPPER(TO_CHAR(txn_date, 'month')) AS month_name,
         COUNT(CASE WHEN txn_type = 'deposit' THEN 1 END) AS deposit,
         COUNT(CASE WHEN txn_type = 'purchase' THEN 1 END) AS purchase,
         COUNT(CASE WHEN txn_type = 'withdrawal' THEN 1 END) AS withdrawal
  FROM customer_transactions
  GROUP BY customer_id, month, month_name
)

SELECT month,
       month_name,
       COUNT(DISTINCT customer_id) AS customer_counts
FROM transaction_details
WHERE deposit > 1 AND (purchase > 0 OR withdrawal > 0)
GROUP BY month, month_name
ORDER BY month;
```

123 month	ABC month_name	123 customer_counts
1	JANUARY	168
2	FEBRUARY	181
3	MARCH	192
4	APRIL	70

Q4: What is the closing balance for each customer at the end of the month?

```
SELECT customer_id,
       EXTRACT(MONTH FROM txn_date) AS month,
       UPPER(TO_CHAR(txn_date, 'month')) AS month_name,
       SUM(CASE WHEN txn_type = 'deposit' THEN txn_amount ELSE -txn_amount END) AS closing_balance
FROM customer_transactions
GROUP BY customer_id, month, month_name
ORDER BY customer_id, month;
```

Sample Output:

customer_id	month	month_name	closing_balance
1	1	JANUARY	312
1	3	MARCH	-952
2	1	JANUARY	549
2	3	MARCH	61
3	1	JANUARY	144
3	2	FEBRUARY	-965
3	3	MARCH	-401
3	4	APRIL	493
4	1	JANUARY	848
4	3	MARCH	-193
5	1	JANUARY	954
5	3	MARCH	-2,877
5	4	APRIL	-490
6	1	JANUARY	733
6	2	FEBRUARY	-785
6	3	MARCH	392
7	1	JANUARY	964
7	2	FEBRUARY	2,209

Q5: What is the percentage of customers who increase their closing balance by more than 5%?

```
WITH monthly_balance_cte as (
  SELECT customer_id,
         EXTRACT(MONTH FROM txn_date) AS month,
         SUM(CASE WHEN txn_type = 'deposit' THEN txn_amount ELSE -txn_amount END) AS
closing_balance
  FROM customer_transactions
  GROUP BY customer_id, month
),
closingbalance_gt5_cte AS (
  SELECT COUNT(DISTINCT customer_id) AS customer_count
  FROM (SELECT customer_id,
              (LEAD(closing_balance) OVER(PARTITION BY customer_id ORDER BY month) -
               closing_balance) / closing_balance::numeric * 100 AS percent_change
        FROM monthly_balance_cte
        ) sb
    WHERE percent_change > 5
)
SELECT COUNT(DISTINCT ct.customer_id) AS total_customers,
       MAX(customer_count) AS customer_count,
       CONCAT(ROUND(MAX(customer_count) / COUNT(DISTINCT ct.customer_id)::numeric * 100,
2), ' %') AS customer_percentage
FROM closingbalance_gt5_cte cb
CROSS JOIN customer_transactions ct;
```

123 total_customers	123 customer_count	ABC customer_percentage
500	269	53.80 %

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?

Running balance

```sql
SELECT *,
       SUM(CASE WHEN txn_type = 'deposit' THEN txn_amount ELSE -txn_amount END)
       OVER(PARTITION BY customer_id ORDER BY txn_date) AS running_balance
FROM customer_transactions;
```

Sample Output:

customer_id	txn_date	txn_type	txn_amount	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	-1,034
3	2020-03-19	withdrawal	188	-1,222
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	1,780
5	2020-01-31	withdrawal	826	954

Monthly balance

```sql
SELECT customer_id,
       EXTRACT(MONTH FROM txn_date) AS month,
       UPPER(TO_CHAR(txn_date, 'month')) AS month_name,
       SUM(CASE WHEN txn_type = 'deposit' THEN txn_amount ELSE -txn_amount END) AS closing_balance
FROM customer_transactions
GROUP BY customer_id, month, month_name
ORDER BY customer_id, month;
```

Sample Output:

customer_id	month	month_name	closing_balance
1	1	JANUARY	312
1	3	MARCH	-952
2	1	JANUARY	549
2	3	MARCH	61
3	1	JANUARY	144
3	2	FEBRUARY	-965
3	3	MARCH	-401
3	4	APRIL	493
4	1	JANUARY	848
4	3	MARCH	-193
5	1	JANUARY	954
5	3	MARCH	-2,877
5	4	APRIL	-490
6	1	JANUARY	733
6	2	FEBRUARY	-785
6	3	MARCH	392
7	1	JANUARY	964

Min, Max & Average Transaction

```sql
WITH running_bal_cte AS (
  SELECT *,
         SUM(CASE WHEN txn_type = 'deposit' THEN txn_amount ELSE -txn_amount END)
         OVER (PARTITION BY customer_id ORDER BY txn_date) AS running_balance
  FROM customer_transactions
)

SELECT customer_id,
       MIN(running_balance) AS min_transaction,
       MAX(running_balance) AS max_transaction,
       ROUND(AVG(running_balance), 2) AS avg_transaction
FROM running_bal_cte
GROUP BY customer_id
ORDER BY customer_id;
```

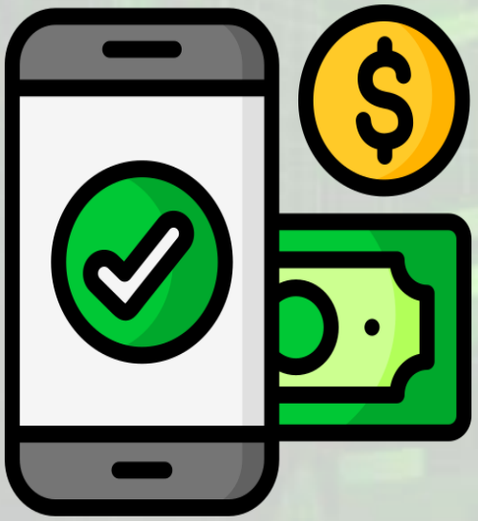
Sample Output:

customer_id	min_transaction	max_transaction	avg_transaction
1	-640	312	-151
2	549	610	579.5
3	-1,222	144	-732.4
4	458	848	653.67
5	-2,413	1,780	-135.45
6	-552	2,197	624
7	887	3,539	2,268.69
8	-1,029	1,363	173.7
9	-91	2,030	1,021.7
10	-5,090	556	-2,229.83
11	-2,529	60	-1,950.82
12	-647	295	-14.5
13	379	1,444	901.15
14	205	989	751
15	379	1,102	740.5
16	-4,284	421	-1,921.53
17	-892	465	-292.33

Insights

- There are Five unique nodes which contain customer cash and data.
- Most of the customers are allocated to Australia. Also Customers are reallocated to a different node every two weeks, on average.
- Deposits occur more frequently than withdrawals and purchases, which account for half of all transactions.
- Customers deposited an average of 5 times for an average of $509 each.
- 53.8% of customers have increased their closing balance by at least 5%

THANK YOU !



Case Study by: Satya Ranjan Ray

Email ID: srray200@gmail.com