**Task:**

Requested to create a query that satisfies the below requirements.

**Resulting schema:**

- id, serial

- dt\_report, date (yyyy-mm-dd), the date of report (i.e. the date of the relevant close time of trades)

- login\_hash, text

- server\_hash, text

- symbol, text

- currency, text

- sum\_volume\_prev\_7d, double, sum of volume traded by login/server/symbol in previous 7 days

including current dt\_report

- sum\_volume\_prev\_all, double, sum of volume traded by login/server/symbol all previous days

including current dt\_report

- rank\_volume\_symbol\_prev\_7d, int, dense rank of most volume traded by login/symbol in

previous 7 days including current dt\_report

- rank\_count\_prev\_7d, int, dense rank of most trade count traded by login in previous 7 days

including current dt\_report

- sum\_volume\_2020\_08, double, sum of volume traded by login/server/symbol for August 2020

only, up to and including current dt\_report

- date\_first\_trade, timestamp, datetime of first trade by login/server/symbol, up to and

including current dt\_report

Need to return a row for every combination of dt\_report/login/server/symbol for every day in June,

July, August and September 2020. Your method should work even if there is no data on a

particular day in this period with in the data.

Need to run this query on users that exist in the users table only and include enabled

accounts only.

**Data Catalog**

login\_hash - hashed user login ID

ticket\_hash - hashed trade ID

server\_hash - hashed machine ID (note that logins and tickets belong to servers)

symbol - financial instrument being traded

digits - number of significant digits after the decimal place

cmd - 0 = buy, 1 = sell

volume - size of the trade

open\_time - open time of the trade

open\_price - open time of the trade

close\_time - close time of the trade (epoch means trade is still open)

contractsize - size of a single contract of the financial instrument

country\_hash - hash of the country of the user

currency - denomination of the account currency

enable - if the login account is enabled or not

**Query and Explanation:**

**active\_users AS (**

**SELECT login\_hash, server\_hash, currency**

**FROM Users**

**WHERE enable = 1**

**)**

This CTE filters the Users table to select only the enabled accounts (enable = 1). It returns the login\_hash, server\_hash, and currency for these users.

We only want to report on trades made by active users.

**WITH report\_dates AS (**

**SELECT generate\_series('2020-06-01'::date, '2020-09-30'::date, '1 day'::interval)::date AS dt\_report**

**)**

This Common Table Expression (CTE) generates a continuous series of dates from June 1, 2020, to September 30, 2020. Each date represents a reporting day (dt\_report).

**user\_trades AS (**

**SELECT**

**t.login\_hash,**

**t.server\_hash,**

**t.symbol,**

**t.volume,**

**t.close\_time::date AS trade\_date,**

**t.contractsize,**

**CASE**

**-- Volume for trades in August 2020**

**WHEN t.close\_time::date BETWEEN '2020-08-01' AND '2020-08-31' THEN t.volume**

**ELSE 0**

**END AS august\_volume,**

**t.close\_time**

**FROM Trades t**

**JOIN active\_users au ON t.login\_hash = au.login\_hash AND t.server\_hash = au.server\_hash**

**)**

This CTE joins the Trades table with active\_users to include only trades made by active users. It also extracts key trade details, such as the date of the trade and whether the trade volume falls within August 2020.

In this,

1. trade\_date: Extracts the date part from close\_time, which is used for filtering and grouping.

2. august\_volume: A calculated column that holds the trade volume if the trade occurred in August 2020. If the trade is outside August 2020, this column is set to 0.

**aggregated\_data AS (**

**SELECT**

**rd.dt\_report,**

**au.login\_hash,**

**au.server\_hash,**

**ut.symbol,**

**au.currency,**

**COALESCE(SUM(CASE**

**WHEN ut.trade\_date BETWEEN rd.dt\_report - INTERVAL '6 days' AND rd.dt\_report THEN ut.volume**

**ELSE 0**

**END), 0) AS sum\_volume\_prev\_7d,**

**COALESCE(SUM(ut.volume), 0) AS sum\_volume\_prev\_all,**

**DENSE\_RANK() OVER (PARTITION BY au.login\_hash, ut.symbol ORDER BY SUM(CASE**

**WHEN ut.trade\_date BETWEEN rd.dt\_report - INTERVAL '6 days' AND rd.dt\_report THEN ut.volume**

**ELSE 0**

**END) DESC) AS rank\_volume\_symbol\_prev\_7d,**

**DENSE\_RANK() OVER (PARTITION BY au.login\_hash ORDER BY COUNT(CASE**

**WHEN ut.trade\_date BETWEEN rd.dt\_report - INTERVAL '6 days' AND rd.dt\_report THEN 1**

**ELSE NULL**

**END) DESC) AS rank\_count\_prev\_7d,**

**COALESCE(SUM(CASE**

**WHEN ut.trade\_date BETWEEN '2020-08-01' AND rd.dt\_report THEN ut.august\_volume**

**ELSE 0**

**END), 0) AS sum\_volume\_2020\_08,**

**MIN(ut.close\_time) AS date\_first\_trade**

**FROM**

**report\_dates rd**

**CROSS JOIN active\_users au**

**LEFT JOIN user\_trades ut**

**ON au.login\_hash = ut.login\_hash AND au.server\_hash = ut.server\_hash AND ut.trade\_date <= rd.dt\_report**

**GROUP BY rd.dt\_report, au.login\_hash, au.server\_hash, ut.symbol, au.currency**

**)**

This CTE aggregates the trade data for each combination of dt\_report, login\_hash, server\_hash, and symbol. It calculates the metrics required for the final output.

In this,

1.sum\_volume\_prev\_7d: The sum of trade volumes for the last 7 days (including the report date) for the given login\_hash, server\_hash, and symbol.

If there are no trades in that 7-day window, the SUM function would return NULL. To avoid this, the COALESCE function ensures that 0 is returned instead of NULL in such cases. This prevents potential issues when interpreting or using the result.

2. sum\_volume\_prev\_all: The sum of all trade volumes up to and including the current report date.

If there are no trades up to the current date, the SUM function would return NULL. The COALESCE function replaces this NULL with 0, ensuring that the column always has a numeric value.

3. rank\_volume\_symbol\_prev\_7d: The dense rank based on the total volume traded for the given login\_hash and symbol over the last 7 days.

1. **Partition By**: The ranking is done separately for each combination of login\_hash and symbol. Each unique pair has its own ranking.
2. **Order By**: The ranking is based on the total volume traded in the previous 7 days (including the report date). The highest volume gets rank 1.
3. **Result**: Symbols with the same volume receive the same rank, and no ranks are skipped

4. rank\_count\_prev\_7d: The dense rank based on the total trade count for the given login\_hash over the last 7 days.

1. **Partition By**: The ranking is done separately for each login\_hash, meaning each user’s trades are ranked independently.
2. **Order By**: The ranking is based on the count of trades in the previous 7 days. The user with the most trades gets rank 1.
3. **Result**: Users with the same trade count receive the same rank, and no ranks are skipped

5. sum\_volume\_2020\_08: The sum of trade volumes for August 2020 up to and including the current report date.

If there are no trades in August 2020, the SUM function would return NULL. COALESCE ensures that the column has a value of 0 instead of NULL

6. date\_first\_trade: The timestamp of the first trade made by the user for the given login\_hash, server\_hash, and symbol up to the current report date.

**final\_output AS (**

**SELECT**

**ROW\_NUMBER() OVER () AS id,**

**dt\_report,**

**login\_hash,**

**server\_hash,**

**COALESCE(symbol, '') AS symbol,**

**currency,**

**sum\_volume\_prev\_7d,**

**sum\_volume\_prev\_all,**

**rank\_volume\_symbol\_prev\_7d,**

**rank\_count\_prev\_7d,**

**sum\_volume\_2020\_08,**

**date\_first\_trade**

**FROM aggregated\_data**

**)**

**SELECT \* FROM final\_output**

**ORDER BY dt\_report, login\_hash, server\_hash, symbol;**

This final CTE selects the relevant columns from the aggregated\_data CTE and adds a unique id for each row.

In this,

1 id: A sequential identifier generated using ROW\_NUMBER() for uniqueness.

2. dt\_report: The report date generated from report\_dates

3. login\_hash, server\_hash, symbol, currency: The identifying fields for the user, server, and financial instrument.

4. sum\_volume\_prev\_7d: The sum of trade volumes in the last 7 days, including the report date.

5. sum\_volume\_prev\_all: The sum of all trade volumes up to and including the report date.

6. rank\_volume\_symbol\_prev\_7d: The rank of total volume traded by the login\_hash and symbol over the last 7 days.

7. rank\_count\_prev\_7d: The rank of total trade count by the login\_hash over the last 7 days.

8. sum\_volume\_2020\_08: The sum of trade volumes for August 2020 up to and including the report date.

9. date\_first\_trade: The timestamp of the first trade made by the user for the given login\_hash, server\_hash, and symbol up to the report date.

Final Code:

**WITH** *active\_users* **AS** (

**SELECT** login\_hash, server\_hash, currency

**FROM** Users

**WHERE** **enable** = 1

),

*report\_dates* **AS** (

**SELECT** **generate\_series**(**'2020-06-01'**::**date**, **'2020-09-30'**::**date**, **'1 day'**::**interval**)::**date** **AS** *dt\_report*

),

*user\_trades* **AS** (

**SELECT**

*t*.login\_hash,

*t*.server\_hash,

*t*.symbol,

*t*.volume,

*t*.close\_time::**date** **AS** *trade\_date*,

*t*.contractsize,

**CASE**

-- Volume for trades in August 2020

**WHEN** *t*.close\_time::**date** **BETWEEN** **'2020-08-01'** **AND** **'2020-08-31'** **THEN** *t*.volume

**ELSE** 0

**END** **AS** *august\_volume*,

*t*.close\_time

**FROM** Trades *t*

**JOIN** *active\_users* *au* **ON** *t*.login\_hash = *au*.login\_hash **AND** *t*.server\_hash = *au*.server\_hash

),

*aggregated\_data* **AS** (

**SELECT**

*rd*.*dt\_report*,

*au*.login\_hash,

*au*.server\_hash,

*ut*.symbol,

*au*.currency,

**COALESCE**(**SUM**(**CASE**

**WHEN** ut.trade\_date **BETWEEN** rd.dt\_report - **INTERVAL** **'6 days'** **AND** rd.dt\_report **THEN** *ut*.volume

**ELSE** 0

**END**), 0) **AS** *sum\_volume\_prev\_7d*,

**COALESCE**(**SUM**(*ut*.volume), 0) **AS** *sum\_volume\_prev\_all*,

**DENSE\_RANK**() **OVER** (**PARTITION** **BY** *au*.login\_hash, *ut*.symbol **ORDER** **BY** **SUM**(**CASE**

**WHEN** ut.trade\_date **BETWEEN** rd.dt\_report - **INTERVAL** **'6 days'** **AND** rd.dt\_report **THEN** *ut*.volume

**ELSE** 0

**END**) **DESC**) **AS** *rank\_volume\_symbol\_prev\_7d*,

**DENSE\_RANK**() **OVER** (**PARTITION** **BY** *au*.login\_hash **ORDER** **BY** **COUNT**(**CASE**

**WHEN** ut.trade\_date **BETWEEN** rd.dt\_report - **INTERVAL** **'6 days'** **AND** rd.dt\_report **THEN** 1

**ELSE** **NULL**

**END**) **DESC**) **AS** *rank\_count\_prev\_7d*,

**COALESCE**(**SUM**(**CASE**

-- Accumulate volume for trades in August 2020, up to the current dt\_report

**WHEN** *ut*.*trade\_date* **BETWEEN** **'2020-08-01'** **AND** *rd*.*dt\_report* **THEN** *ut*.*august\_volume*

**ELSE** 0

**END**), 0) **AS** *sum\_volume\_2020\_08*,

**MIN**(*ut*.close\_time) **AS** *date\_first\_trade*

**FROM**

*active\_users* *au*

**CROSS** **JOIN** *report\_dates* *rd*

**LEFT** **JOIN** *user\_trades* *ut*

**ON** *au*.login\_hash = *ut*.login\_hash **AND** *au*.server\_hash = *ut*.server\_hash **AND** *ut*.*trade\_date* <= *rd*.*dt\_report*

**GROUP** **BY** *rd*.*dt\_report*, *au*.login\_hash, *au*.server\_hash, *ut*.symbol, *au*.currency

),

*final\_output* **AS** (

**SELECT**

**ROW\_NUMBER**() **OVER** () **AS** *id*,

*dt\_report*,

login\_hash,

server\_hash,

**COALESCE**(symbol, **''**) **AS** *symbol*,

currency,

*sum\_volume\_prev\_7d*,

*sum\_volume\_prev\_all*,

*rank\_volume\_symbol\_prev\_7d*,

*rank\_count\_prev\_7d*,

*sum\_volume\_2020\_08*,

*date\_first\_trade*

**FROM** *aggregated\_data*

)

**SELECT** \* **FROM** *final\_output*

**ORDER** **BY** *dt\_report*, login\_hash, server\_hash, *symbol*;