**SESSION 6 – HOMEWORK**

**Practice GROUP BY - WINDOW FUNCTION - Time series analysis**

|  |
| --- |
| **Description:**  *Paytm is an Indian multinational financial technology company. It specializes in digital payment system, e-commerce and financial services. Paytm wallet is a secure and RBI (Reserve Bank of India)approved digital/mobile wallet that provides a myriad of financial features to fulfill every consumer’s payment needs. Paytm wallet can be topped up through UPI (Unified Payments Interface), internet banking, or credit/debit cards. Users can also transfer money from a Paytm wallet to recipient’s bank account or their own Paytm wallet.*  Below is a small database of payment transactions from 2019 to 2020 of Paytm Wallet. The database includes 6 tables:   * fact\_transaction: Store information of all types of transactions: Payments, Top-up, Transfers, Withdrawals * dim\_scenario: Detailed description of transaction types * dim\_payment\_channel: Detailed description of payment methods * dim\_platform: Detailed description of payment devices * dim\_status: Detailed description of the results of the transaction |

**PART 1: PRACTICE SKILLS**

**Understand and use flexibly between GROUP BY and WINDOW FUNCTION**

# 1. GROUP BY

When we need to group together data lines of the same nature and calculate the SUM, COUNT, MIN, MAX, AVG commands.

**Task 1:** Calculate the number of successful transactions of each month in 2019

*--Your code here:*

SELECT MONTH(transaction\_time),

COUNT(transaction\_id) AS number\_success\_trans

FROM fact\_transaction\_2019 AS fact\_19

LEFT JOIN dim\_scenario AS dsc

ON fact\_19.scenario\_id = dsc.scenario\_id

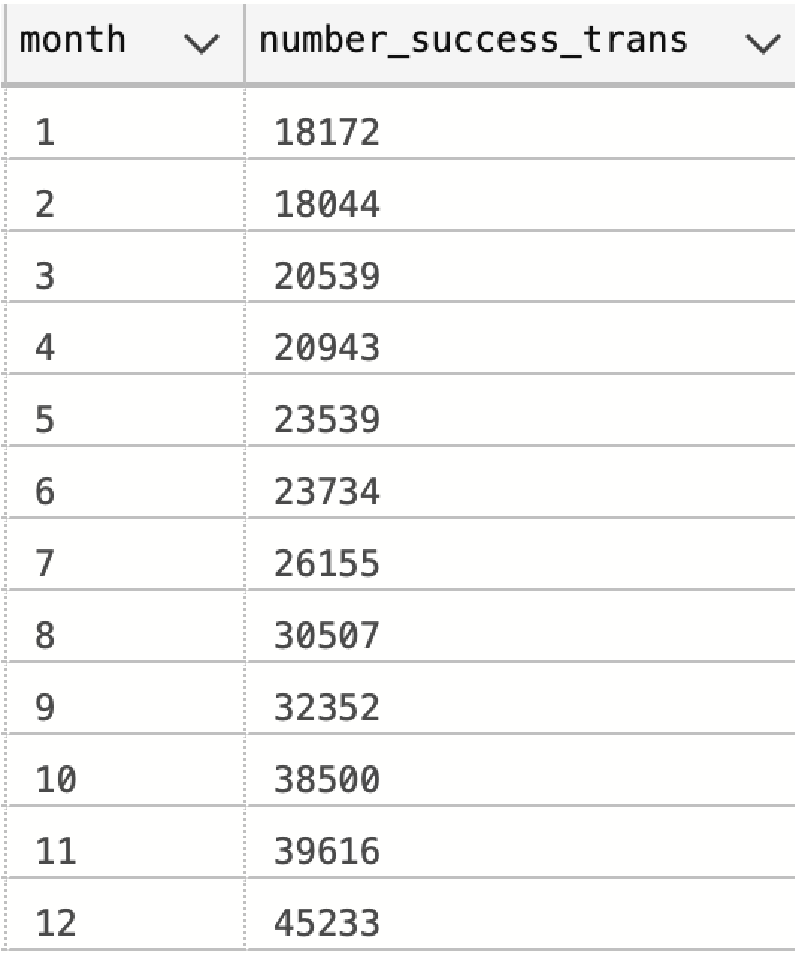
LEFT JOIN dim\_status AS dst

ON fact\_19.status\_id = dst.status\_id

WHERE fact\_19.status\_id = 1

GROUP BY MONTH(transaction\_time)

ORDER BY MONTH(transaction\_time)



# 2. WINDOW FUNCTION

* The window function helps us to process external grouping commands and fill the results in a new column in the data table query we are working on.
* Syntax:

# <AGGREGATE FUNCTION> OVER (PARTITION BY ... ORDER BY ...)

- **Lưu ý**: In the same query, if there is a Group By, the Window Function cannot be used anymore and vice versa. The solution is that I will divide them into CTEs for processing.

**Task 2.1:** Calculate the number of successful transactions of each month in 2019 and 2020 *(using Group By)*. Then create a column of the total number of successful transactions of each year *(using Window Function).* Finally calculate the successful transaction rate **(success\_rate)** of each month.

*--Your code here:*

WITH table\_year AS (

SELECT YEAR(transaction\_time) [year],

       MONTH(transaction\_time) [month],

       COUNT(transaction\_id) AS number\_success\_trans

FROM(SELECT \* FROM fact\_transaction\_2019 UNION SELECT \* FROM fact\_transaction\_2020) AS fact

LEFT JOIN dim\_scenario AS dsc

ON fact.scenario\_id = dsc.scenario\_id

LEFT JOIN dim\_status AS dst

ON fact.status\_id = dst.status\_id

WHERE fact.status\_id = 1

GROUP BY YEAR(transaction\_time), MONTH(transaction\_time)

)

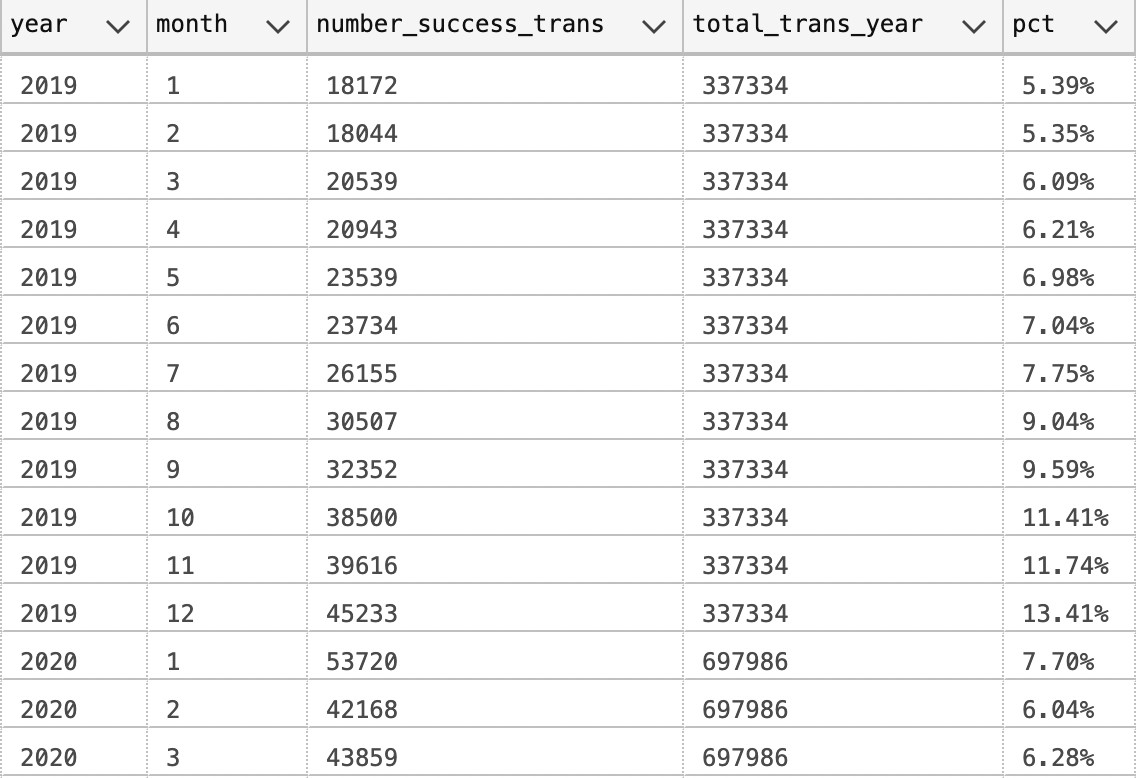
SELECT \*,

       SUM(number\_success\_trans) OVER (PARTITION BY [year]) AS total\_trans\_year,

       FORMAT(CAST(number\_success\_trans AS decimal)/SUM(number\_success\_trans) OVER (PARTITION BY [year]),'p') AS pct

FROM table\_year

ORDER BY [year], [month]



**Task 2.2:** Find out the TOP 3 months with the most failed transactions of each year *(using window function)*

***Hint:*** *Choose the right ranking function to use*

*--Your code here:*

WITH table\_rank AS (

SELECT \*,

RANK() OVER (PARTITION BY [year] ORDER BY number\_failed\_trans DESC) total\_rank

FROM (

SELECT YEAR(transaction\_time) [year],

       MONTH(transaction\_time) [month],

       COUNT(transaction\_id) AS number\_failed\_trans

FROM(SELECT \* FROM fact\_transaction\_2019 UNION SELECT \* FROM fact\_transaction\_2020) AS fact

LEFT JOIN dim\_scenario AS dsc

ON fact.scenario\_id = dsc.scenario\_id

LEFT JOIN dim\_status AS dst

ON fact.status\_id = dst.status\_id

WHERE fact.status\_id != 1

GROUP BY YEAR(transaction\_time), MONTH(transaction\_time)

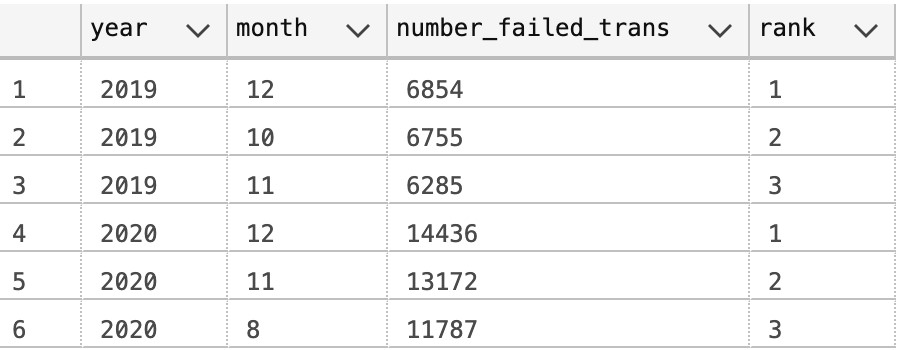
) [table\_year]

)

SELECT \*

FROM table\_rank

WHERE total\_rank <4



**Task 2.3** : Calculate the average distance between successful payments per customer in Telecom group 2019.

***Hint****: Use LAG() function in combination with Window Function*

WITH table\_day AS (

SELECT customer\_id,

AVG(gap\_day) OVER (PARTITION BY customer\_id) AS avg\_gap\_day

FROM(

SELECT customer\_id,

transaction\_id,

transaction\_time,

DATEDIFF(day,LAG(transaction\_time) OVER (partition BY customer\_id ORDER BY transaction\_time),transaction\_time) AS gap\_day

FROM fact\_transaction\_2019 AS fact\_19

LEFT JOIN dim\_scenario AS dsc

ON fact\_19.scenario\_id = dsc.scenario\_id

LEFT JOIN dim\_status AS dst

ON fact\_19.status\_id = dst.status\_id

WHERE fact\_19.status\_id = 1 AND category = 'Telco'

) table\_time

),

table\_col AS (

SELECT \*,

ROW\_NUMBER() OVER (partition BY customer\_id ORDER BY avg\_gap\_day) AS num\_col

FROM table\_day

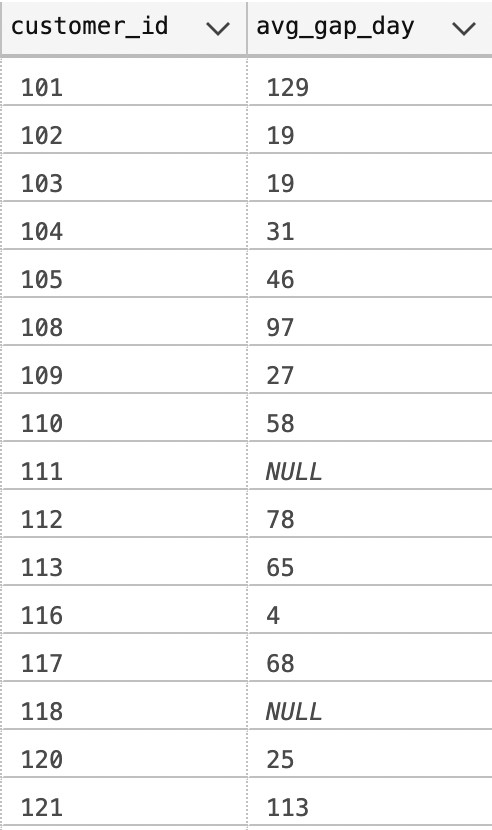
)

SELECT customer\_id,

       avg\_gap\_day

FROM table\_col

WHERE num\_col = 1



**SUMMARY**:

* Window Function calculates the value for each line, while GROUP BY does not.
* GROUP BY reduces the original number of lines to the correct number of groups, Window Function keeps the number of lines.

*\** For example, in Task 1 above, if we use the window function, it will take a long time to process the data because the window function processes on each line.

|  |  |
| --- | --- |
| **GROUP BY** | **WINDOW FUNCTION** |
| SELECT  MONTH(transaction\_time) AS month  ,COUNT(transaction\_id) AS number\_trans  FROM fact\_transaction\_2019  WHERE status\_id = 1  GROUP BY MONTH(transaction\_time); | SELECT transaction\_id  ,MONTH(transaction\_time) AS month  ,COUNT(transaction\_id) OVER (PARTITION BY  MONTH(transaction\_time)) AS number\_trans  FROM fact\_transaction\_2019  WHERE status\_id = 1 |
|  |  |

**PART 2: SQL APPLIED TO REAL PROBLEMS**

## INTRODUCTION: Time Series Analysis

Time series analysis is one of the most common types of analysis done with SQL. A time series is a sequence of measurements or data points recorded in time order, often at regularly spaced intervals. There are many examples of time series data in daily life, such as the daily high temperature, the closing value of the S&P 500 stock index, or the number of daily steps recorded by your fitness tracker. Time series analysis is used in a wide variety of industries and disciplines, from statistics and engineering to weather forecasting and business planning. Time series analysis is a way to understand and quantify how things change over time.

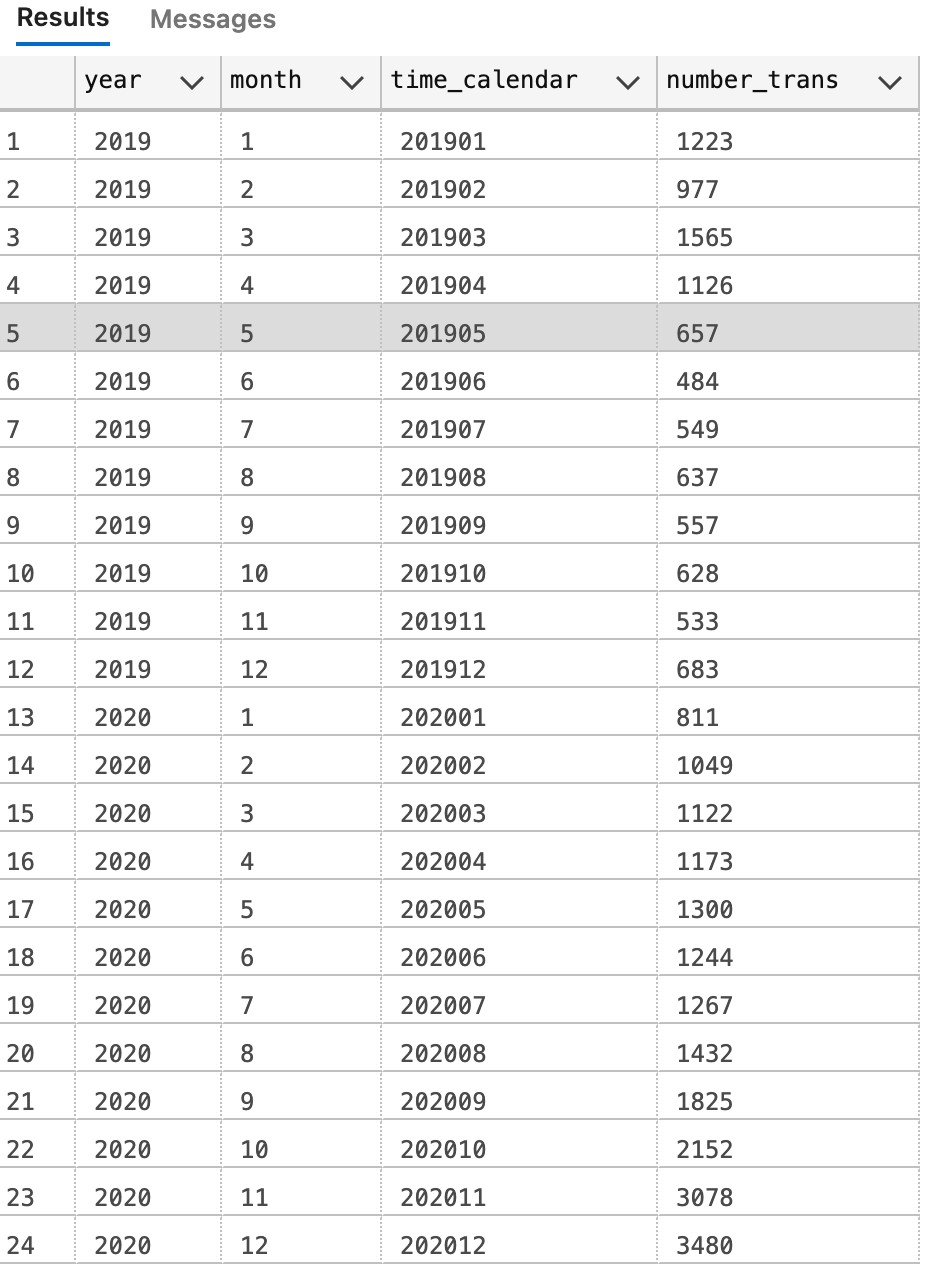
## TIME SERIES ANALYSIS METHODS

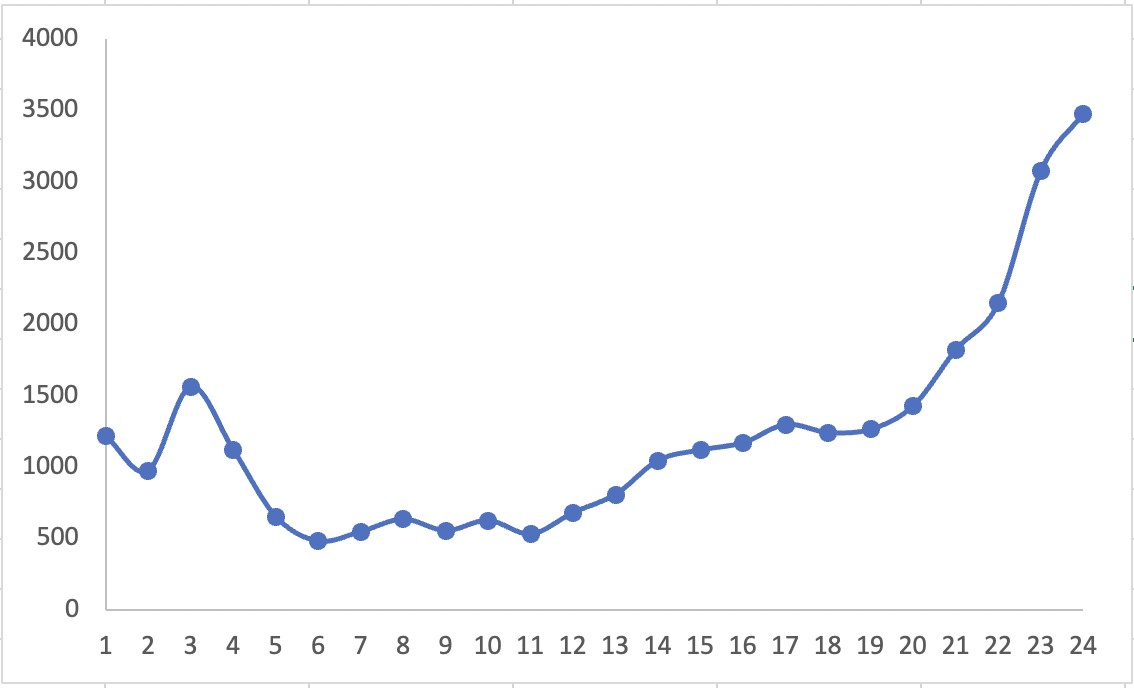
**1. Trending the Data:** With time series data, we often want to look for trends in the data. A trend is simply the direction in which the data is moving. It may be moving up or increasing over time, or it may be moving down or decreasing over time. It can remain more or less flat, or there could be so much noise, or movement up and down, that it’s hard to determine a trend at all.

### 1.1. Simple trend

**Task**: You need to analyze the trend of payment transactions of the **Billing category** from 2019 to 2020.

First, let’s show **the trend of the number of successful transactions by month**.





-- Your code here:

WITH table\_trans AS (

SELECT transaction\_id, transaction\_time,

       YEAR(transaction\_time) [year],

       MONTH(transaction\_time) [month]

FROM (

    SELECT \*

    FROM fact\_transaction\_2019

    UNION

    SELECT \*

    FROM fact\_transaction\_2020) AS fact

LEFT JOIN dim\_scenario AS dsc

ON fact.scenario\_id = dsc.scenario\_id

LEFT JOIN dim\_status AS dst

ON fact.status\_id = dst.status\_id

WHERE fact.status\_id = 1 AND category = 'Billing'

)

SELECT DISTINCT [year],

                [month],

                FORMAT(transaction\_time,'yyyyMM') AS time\_calendar,

                COUNT(transaction\_id) OVER (PARTITION BY [year], [month] ) AS number\_trans

FROM table\_trans

ORDER BY [year], [month]

### 1.2. Comparing Component

**Task A:** You know that there are many sub-categories of **the Billing group**. After reviewing the above result, you should **break down the trend into each sub-categories**.



*-- Your code here:*

WITH table\_trans AS (

SELECT transaction\_id, sub\_category,

       YEAR(transaction\_time) [year],

       MONTH(transaction\_time) [month]

FROM (

    SELECT \*

    FROM fact\_transaction\_2019

    UNION

    SELECT \*

    FROM fact\_transaction\_2020) AS fact

LEFT JOIN dim\_scenario AS dsc

ON fact.scenario\_id = dsc.scenario\_id

LEFT JOIN dim\_status AS dst

ON fact.status\_id = dst.status\_id

WHERE fact.status\_id = 1 AND category = 'Billing'

)

SELECT DISTINCT [year],

                [month],

                sub\_category,

                COUNT(transaction\_id) OVER (PARTITION BY [year], [month], sub\_category ) AS number\_trans

FROM table\_trans

ORDER BY [year], [month], sub\_category

**Task B**: Then modify the result as the following table: Only select the sub-categories belong to list (Electricity, Internet and Water) ***Hint:***

* **Method 1**: Use SUM in combination with CASE WHEN

*For example: SUM( CASE WHEN … THEN … ELSE …)*

* **Method 2**: Using the Pivot function

*-- Your code here*:

WITH table\_trans AS (

SELECT transaction\_id, sub\_category,

       YEAR(transaction\_time) [year],

       MONTH(transaction\_time) [month]

FROM (

    SELECT \*

    FROM fact\_transaction\_2019

    UNION

    SELECT \*

    FROM fact\_transaction\_2020) AS fact

LEFT JOIN dim\_scenario AS dsc

ON fact.scenario\_id = dsc.scenario\_id

LEFT JOIN dim\_status AS dst

ON fact.status\_id = dst.status\_id

WHERE fact.status\_id = 1 AND category = 'Billing'

)

SELECT [year], [month], [Electricity], [Internet], [Water]

FROM (

      SELECT DISTINCT [year],

                [month],

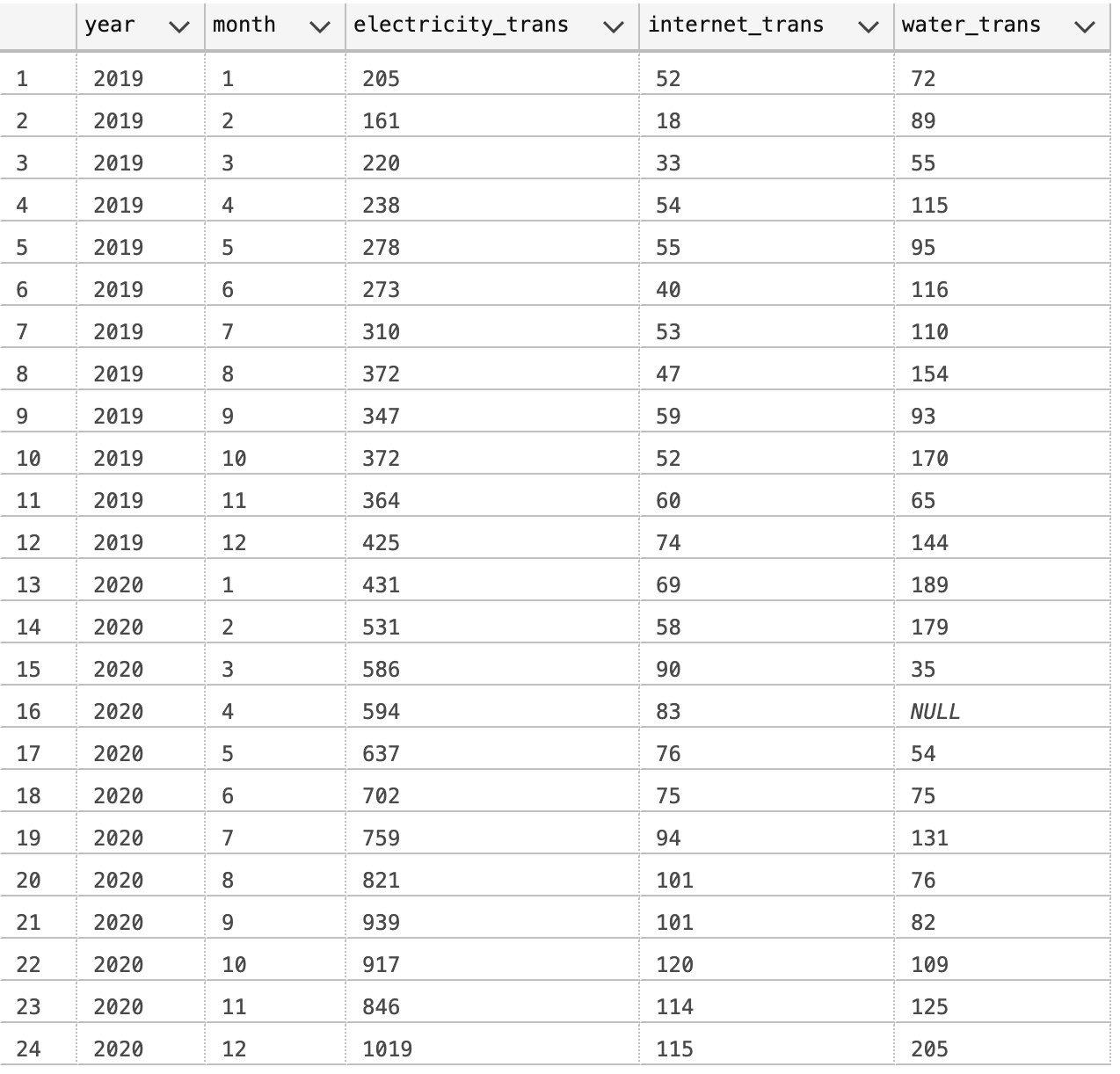
                sub\_category,

                COUNT(transaction\_id) OVER (PARTITION BY [year], [month], sub\_category ) AS number\_trans

      FROM table\_trans) table\_rank

PIVOT (SUM(number\_trans) FOR sub\_category IN ([Electricity],[Internet],[Water])) AS Pivot\_table

ORDER BY [year], [month]



20191

20192

20193

20194

20195

20196

20197

20198

20199

201910

201911

201912

20201

20202

20203

20204

20205

20206

20207

20208

20209

202010

202011

202012

1200

1000

800

600

400

200

0

**electricity\_trans**

**internet\_trans**

**water\_trans**

Trend by sub-category

**1.3. Percent of Total Calculations:** When working with time series data that has multiple parts or attributes that constitute a whole, it’s often useful to analyze each part’s contribution to the whole

and whether that has changed over time. Unless the data already contains a time series of the total values, we’ll need to calculate the overall total in order to calculate the percent of total for each row.

**Task**: Based on the previous query, you need to calculate the proportion of each sub-category (Electricity, Internet and Water) in the total for each month. Let’s see the desired outcome:



*-- Your code here:*

WITH table\_trans AS (

SELECT transaction\_id, sub\_category,

       YEAR(transaction\_time) [year],

       MONTH(transaction\_time) [month]

FROM (

    SELECT \*

    FROM fact\_transaction\_2019

    UNION

    SELECT \*

    FROM fact\_transaction\_2020) AS fact

LEFT JOIN dim\_scenario AS dsc

ON fact.scenario\_id = dsc.scenario\_id

LEFT JOIN dim\_status AS dst

ON fact.status\_id = dst.status\_id

WHERE fact.status\_id = 1 AND category = 'Billing'

),

table\_sub AS (

SELECT [year], [month], [Electricity], [Internet], [Water]

FROM (

    SELECT DISTINCT [year],

                    [month],

                    sub\_category,

                    COUNT(transaction\_id) OVER (PARTITION BY [year], [month], sub\_category ) AS number\_trans

    FROM table\_trans) table\_rank

PIVOT (SUM(number\_trans) FOR sub\_category IN ([Electricity],[Internet],[Water])) AS Pivot\_table )

SELECT [year],[month],[Electricity],[Internet],[Water], total\_trans\_month,

FORMAT(CAST([Electricity] AS decimal)/total\_trans\_month,'p') AS elec\_pct,

FORMAT(CAST([Internet] AS decimal)/total\_trans\_month,'p') AS internet\_pct,

FORMAT(CAST([Water] AS decimal)/total\_trans\_month,'p') AS water\_pct

 FROM (SELECT \*,

              SUM([Electricity1]+[Internet1]+[Water1]) OVER (PARTITION BY [year],[month]) AS total\_trans\_month

       FROM (SELECT \*,

                    ISNULL([Electricity],0) AS [Electricity1],

                    ISNULL([Internet],0) AS [Internet1],

                    ISNULL([Water],0) AS [Water1]

             FROM table\_sub) table\_null) table\_total