**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

SELECT month(transaction\_time) AS [month]

    , count(transaction\_id) AS [number\_success\_trans]

FROM fact\_transaction\_2019 AS fact\_19

WHERE status\_id = 1

GROUP BY month(transaction\_time)

**2. WINDOW FUNCTION**

**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.

-- Calculate the number of successful transactions of each month in 2019 and 2020 (using Group By).

WITH success\_trans\_table AS (

SELECT year(transaction\_time) AS [year]

    , month(transaction\_time) AS [month]

    , count(transaction\_id) AS [number\_success\_trans]

FROM (SELECT \* FROM fact\_transaction\_2019

UNION

SELECT \* FROM fact\_transaction\_2020) AS month\_table

WHERE status\_id = 1

GROUP BY month(transaction\_time), year(transaction\_time)

)

-- Then create a column of the total number of successful transactions of each year (using Window Function).

, year\_table AS(

SELECT [year]

    , [month]

    , [number\_success\_trans]

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

FROM success\_trans\_table

)

-- Finally calculate the successful transaction rate (success\_rate) of each month.

SELECT \*

    , FORMAT(([number\_success\_trans]\*1.0)/([trans\_number\_year]\*1.0),'p') AS pct

FROM year\_table

ORDER BY [year], [month] ASC

**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*

-- Find out the TOP 3 months with the most failed transactions of each year (using window function)

WITH failed\_trans\_table AS (

SELECT year(transaction\_time) AS [year]

    , month(transaction\_time) AS [month]

    , count(transaction\_id) AS [number\_failed\_trans]

FROM (SELECT \* FROM fact\_transaction\_2019

UNION

SELECT \* FROM fact\_transaction\_2020) AS month\_table

WHERE status\_id <> 1

GROUP BY month(transaction\_time), year(transaction\_time)

)

-- Rank top 3 month with most failed transaction

, rank\_table AS (

SELECT [year]

    , [month]

    , [number\_failed\_trans]

    , RANK() OVER (PARTITION BY [year] ORDER BY [number\_failed\_trans] DESC) AS rank

FROM failed\_trans\_table

)

SELECT \*

FROM rank\_table

WHERE 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 cus\_table AS (

SELECT success\_trans.customer\_id

    , success\_trans.transaction\_time

    , LAG(success\_trans.transaction\_time,1) OVER (PARTITION BY success\_trans.customer\_id ORDER BY success\_trans.transaction\_time) AS lag

FROM fact\_transaction\_2019 success\_trans

LEFT JOIN dim\_scenario AS scen

ON success\_trans.scenario\_id = scen.scenario\_id

WHERE success\_trans.status\_id = 1 AND scen.category = 'Telco'

)

SELECT customer\_id

    , AVG(DATEDIFF(day, lag, transaction\_time)) AS avg\_gap\_day

FROM cus\_table

GROUP BY customer\_id

**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**

# 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.

## **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**.

WITH success\_table AS (

SELECT DISTINCT(year(transaction\_time)) AS [year]

    , month(transaction\_time) AS [month]

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

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

FROM (SELECT \* FROM fact\_transaction\_2019

      UNION

      SELECT \* FROM fact\_transaction\_2020) AS month\_table

LEFT JOIN dim\_scenario AS scen

ON month\_table.scenario\_id = scen.scenario\_id

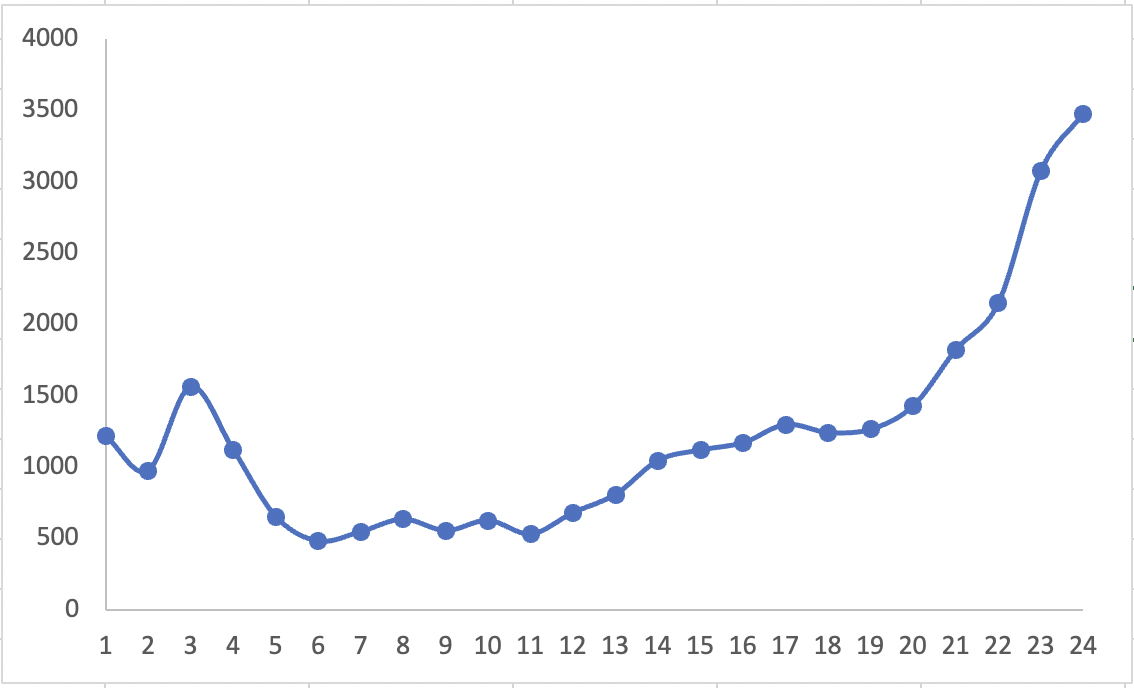
WHERE status\_id = 1 AND category = 'Billing'

)

SELECT \*

FROM success\_table

ORDER BY [year], [month] ASC



## **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**.

WITH success\_table AS (

SELECT DISTINCT(year(transaction\_time)) AS [year]

    , month(transaction\_time) AS [month]

    , sub\_category

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

FROM (SELECT \* FROM fact\_transaction\_2019

      UNION

      SELECT \* FROM fact\_transaction\_2020) AS month\_table

LEFT JOIN dim\_scenario AS scen

ON month\_table.scenario\_id = scen.scenario\_id

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

)

SELECT \*

FROM success\_table

ORDER BY [year], [month] ASC

**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

WITH success\_table AS (

SELECT DISTINCT(year(transaction\_time)) AS [year]

    , month(transaction\_time) AS [month]

    , sub\_category

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

FROM (SELECT \* FROM fact\_transaction\_2019

      UNION

      SELECT \* FROM fact\_transaction\_2020) AS month\_table

LEFT JOIN dim\_scenario AS scen

ON month\_table.scenario\_id = scen.scenario\_id

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

)

SELECT \*

FROM success\_table

PIVOT (

    SUM (number\_trans)

    FOR sub\_category IN ([Water], [Internet], [Electricity])

) AS pivot\_table

ORDER BY [year],[month]

## **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:

WITH success\_table AS (

SELECT DISTINCT(year(transaction\_time)) AS [year]

    , month(transaction\_time) AS [month]

    , sub\_category

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

FROM (SELECT \* FROM fact\_transaction\_2019

      UNION

      SELECT \* FROM fact\_transaction\_2020) AS month\_table

LEFT JOIN dim\_scenario AS scen

ON month\_table.scenario\_id = scen.scenario\_id

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

)

, sub\_cat\_table AS (

    SELECT \*

FROM success\_table

PIVOT (

    SUM (number\_trans)

    FOR sub\_category IN ([Water], [Internet], [Electricity])

) AS pivot\_table

)

, pct\_table AS (

    SELECT \*

        , [Water] + [Electricity] + [Internet] AS [total\_trans\_month]

        , FORMAT(([Water]\*1.0)/ (([Water] + [Electricity] + [Internet])\*1.0),'p') AS water\_pct

        , FORMAT(([Electricity]\*1.0)/ (([Water] + [Electricity] + [Internet])\*1.0),'p') AS elec\_pct

        , FORMAT(([Internet]\*1.0)/ (([Water] + [Electricity] + [Internet])\*1.0),'p') AS internet\_pct

FROM sub\_cat\_table

)

SELECT \*

FROM pct\_table