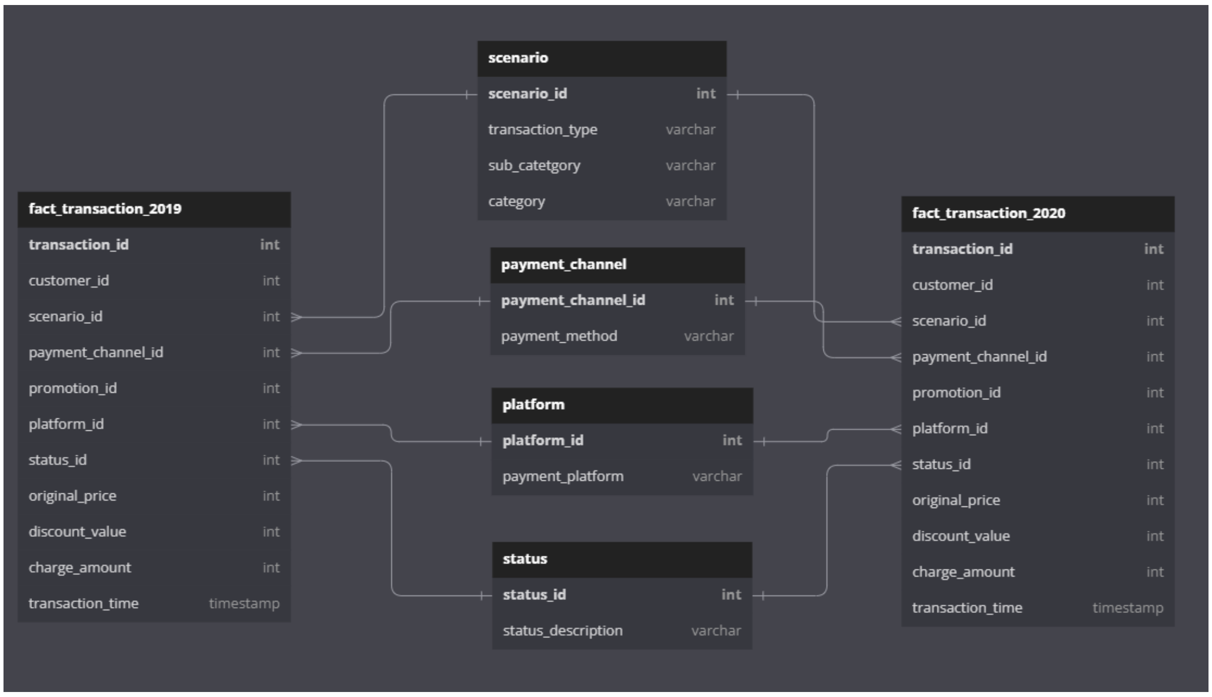
SQL Project: Paytm

I.DataBase Diagram



II. Overview

Paytm is an Indian multinational financial technology company.
It specialises in digital payment systems, 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 fulfil 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 the 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 (2019 and 2020)
- **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

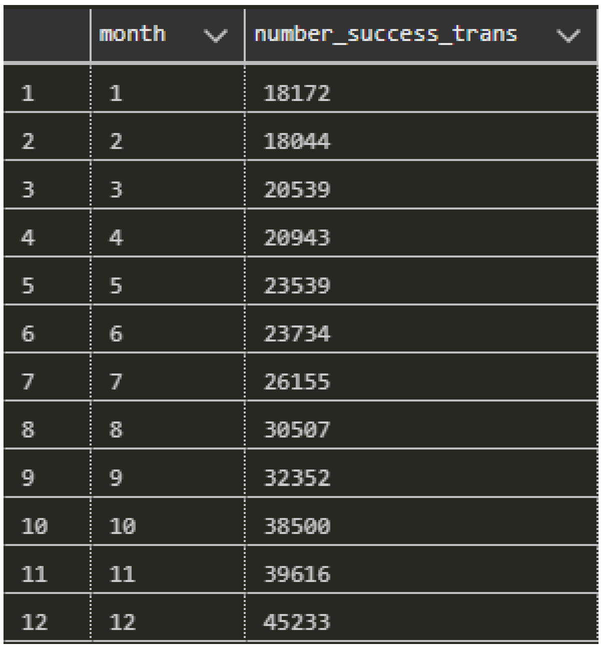
III. SQL Queries

a. Return the result of **successful transactions** for **each month in 2019**.

```
SELECT MONTH (transaction_time) AS [month]
       , COUNT (transaction_id ) AS number_success_trans
FROM dbo.fact_transaction_2019
```

```
LEFT JOIN dim_scenario
ON dbo.fact_transaction_2019.scenario_id = dim_scenario.scenario_id
WHERE status_id = 1
GROUP BY MONTH (transaction_time)
ORDER BY [month]
```

Output:



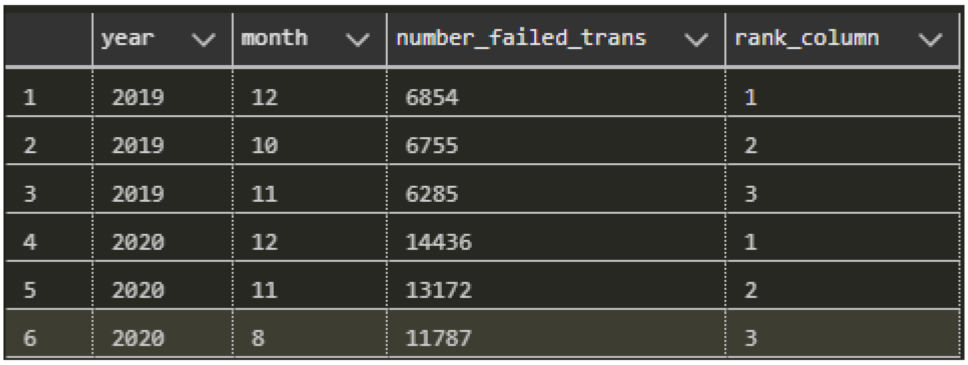
	month	number_success_trans
1	1	18172
2	2	18044
3	3	20539
4	4	20943
5	5	23539
6	6	23734
7	7	26155
8	8	30507
9	9	32352
10	10	38500
11	11	39616
12	12	45233

b. Find the **TOP 3 months** with the **highest** number of **failed transactions** for **each year**.

```
WITH failed_month_table AS (
  SELECT YEAR (transaction_time) AS [year]
        , MONTH (transaction_time) AS [month]
        , COUNT (transaction_id ) AS number_failed_trans
  FROM (SELECT transaction_id, transaction_time, scenario_id, status_id FROM
dbo.fact_transaction_2019
        UNION
        SELECT transaction_id, transaction_time, scenario_id, status_id FROM
dbo.fact_transaction_2020) AS fact_table
  WHERE status_id != 1 -- giao dịch lỗi
  GROUP BY YEAR (transaction_time), MONTH (transaction_time)
--    ORDER BY [year], [month]
)
, rank_table AS (
   SELECT *
   , RANK () OVER ( PARTITION BY [year] ORDER BY number_failed_trans DESC ) AS
rank_column
   FROM failed_month_table
   -- ORDER BY [year], [month]
)
SELECT *
FROM rank_table
```

```
WHERE rank_column < 4
```

Output:

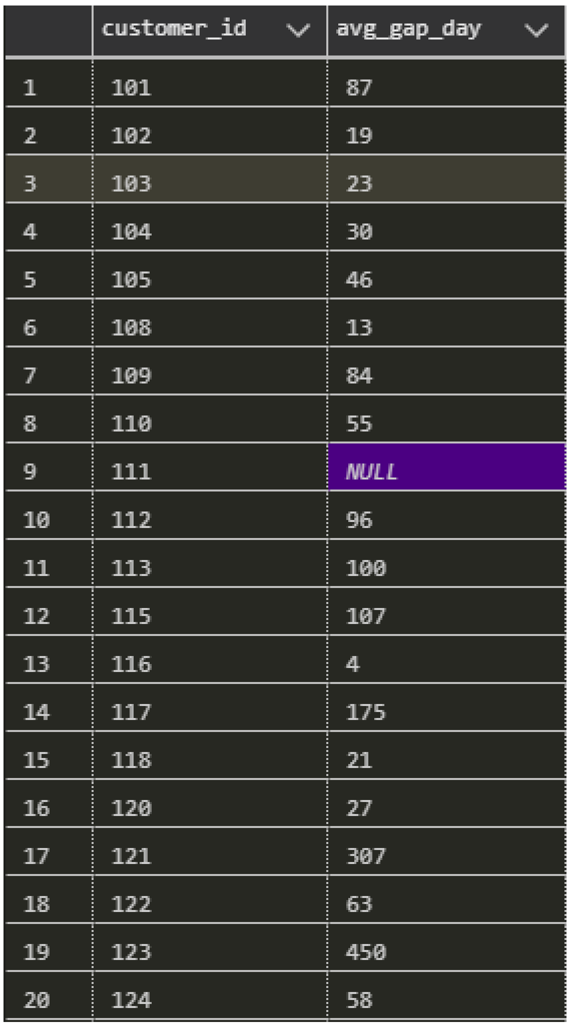


	year	month	number_failed_trans	rank_column
1	2019	12	6854	1
2	2019	10	6755	2
3	2019	11	6285	3
4	2020	12	14436	1
5	2020	11	13172	2
6	2020	8	11787	3

c. **Calculate the average distance between successful payments for each customer in the Telecom group in 2019.**

```
WITH customer_table AS (
  SELECT TOP 1000 customer_id, transaction_id, transaction_time
       , LAG (transaction_time, 1) OVER ( PARTITION BY customer_id ORDER BY
transaction_time ASC ) AS previous_time
       , DATEDIFF (day, LAG (transaction_time, 1) OVER ( PARTITION BY customer_id
ORDER BY transaction_time ASC )
                       , transaction_time) AS gap_day
  FROM (SELECT customer_id, transaction_id, transaction_time, scenario_id FROM
dbo.fact_transaction_2019
        UNION
        SELECT customer_id, transaction_id, transaction_time, scenario_id FROM
dbo.fact_transaction_2020) AS fact_table
  LEFT JOIN dbo.dim_scenario
  ON fact_table.scenario_id = dbo.dim_scenario.scenario_id
  WHERE category = 'Telco'
)
SELECT customer_id
    , AVG (gap_day)  AS avg_gap_day
FROM customer_table
GROUP BY customer_id
```

Output:



	customer_id	avg_gap_day
1	101	87
2	102	19
3	103	23
4	104	30
5	105	46
6	108	13
7	109	84
8	110	55
9	111	NULL
10	112	96
11	113	100
12	115	107
13	116	4
14	117	175
15	118	21
16	120	27
17	121	307
18	122	63
19	123	450
20	124	58

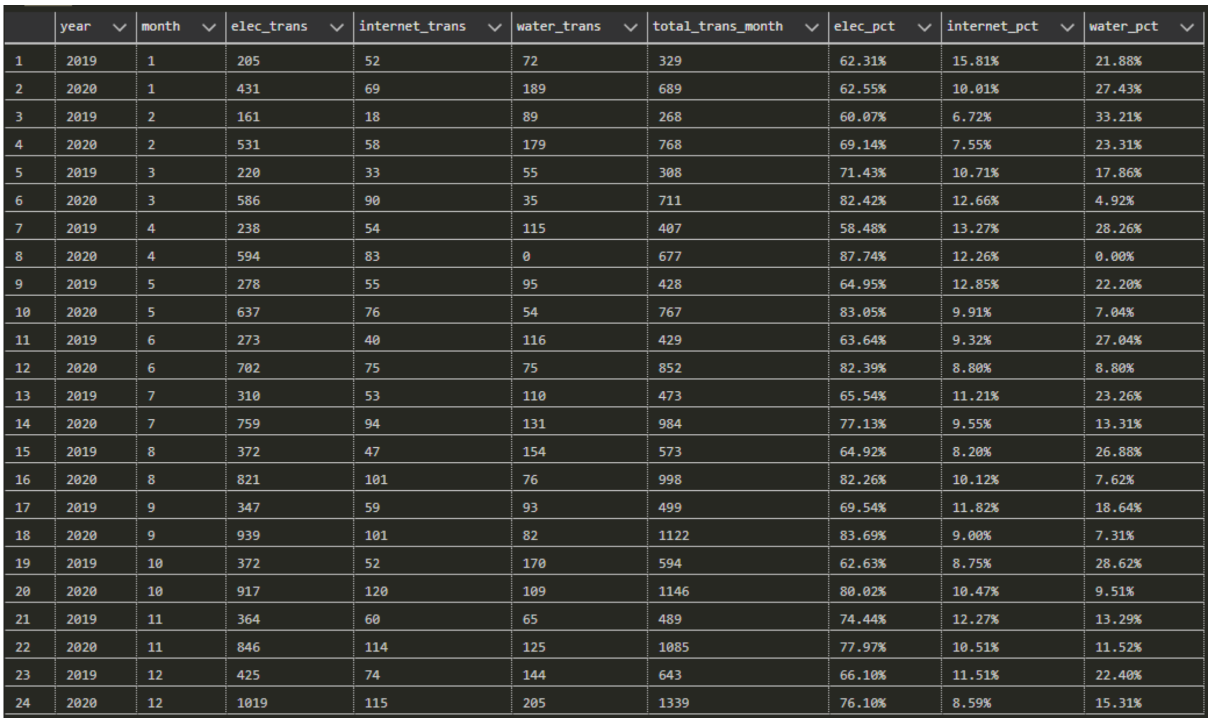
d. Time Series Analysis

Requirements: Create a table that shows the number of transactions for the 3 sub-categories: Electricity, Internet, and Water, as well as the total transactions for all 3 sub-categories in each month from 2019-2020. Then calculate the percentage of sales for each sub-category in each month compared to the total transactions.

```
WITH month_table AS (

   SELECT

     YEAR(transaction_time) AS [year]

     , MONTH (transaction_time) AS [month]

     , sub_category

     , COUNT (transaction_id) AS number_trans

  FROM (SELECT transaction_id, transaction_time, scenario_id, status_id FROM
dbo.fact_transaction_2019

        UNION

        SELECT transaction_id, transaction_time, scenario_id, status_id FROM
dbo.fact_transaction_2020)
```

```
      AS fact_table

  LEFT JOIN dbo.dim_scenario

  ON fact_table.scenario_id = dbo.dim_scenario.scenario_id

  LEFT JOIN dbo.dim_status

  ON fact_table.status_id = dbo.dim_status.status_id

  WHERE category = 'Billing'

     AND fact_table.status_id = 1

   GROUP BY YEAR(transaction_time), MONTH (transaction_time), sub_category

)

, pivot_table AS (

   SELECT [year], [month]

       , SUM ( CASE WHEN sub_category = 'electricity' THEN number_trans ELSE 0 END
) AS elec_trans

       , SUM ( CASE WHEN sub_category = 'internet' THEN number_trans ELSE 0 END )
AS internet_trans

       , SUM ( CASE WHEN sub_category = 'water' THEN number_trans ELSE 0 END ) AS
water_trans

   FROM month_table

   GROUP BY [year], [month]

-- ORDER BY [year], [month]

)

SELECT *

   , elec_trans + internet_trans + water_trans AS total_trans_month

   , FORMAT ( elec_trans *1.0 / ( elec_trans + internet_trans + water_trans ), 'p')
AS elec_pct

   , FORMAT ( internet_trans *1.0 / ( elec_trans + internet_trans + water_trans ),
'p') AS internet_pct

   , FORMAT ( water_trans *1.0 / ( elec_trans + internet_trans + water_trans ),
'p') AS water_pct

FROM pivot_table
```

Output:



	year	month	elec_trans	internet_trans	water_trans	total_trans_month	elec_pct	internet_pct	water_pct
1	2019	1	205	52	72	329	62.31%	15.81%	21.88%
2	2020	1	431	69	189	689	62.55%	10.01%	27.43%
3	2019	2	161	18	89	268	60.07%	6.72%	33.21%
4	2020	2	531	58	179	768	69.14%	7.55%	23.31%
5	2019	3	220	33	55	308	71.43%	10.71%	17.86%
6	2020	3	586	90	35	711	82.42%	12.66%	4.92%
7	2019	4	238	54	115	407	58.48%	13.27%	28.26%
8	2020	4	594	83	0	677	87.74%	12.26%	0.00%
9	2019	5	278	55	95	428	64.95%	12.85%	22.20%
10	2020	5	637	76	54	767	83.05%	9.91%	7.04%
11	2019	6	273	40	116	429	63.64%	9.32%	27.04%
12	2020	6	702	75	75	852	82.39%	8.80%	8.80%
13	2019	7	310	53	110	473	65.54%	11.21%	23.26%
14	2020	7	759	94	131	984	77.13%	9.55%	13.31%
15	2019	8	372	47	154	573	64.92%	8.20%	26.88%
16	2020	8	821	101	76	998	82.26%	10.12%	7.62%
17	2019	9	347	59	93	499	69.54%	11.82%	18.64%
18	2020	9	939	101	82	1122	83.69%	9.00%	7.31%
19	2019	10	372	52	170	594	62.63%	8.75%	28.62%
20	2020	10	917	120	109	1146	80.02%	10.47%	9.51%
21	2019	11	364	60	65	489	74.44%	12.27%	13.29%
22	2020	11	846	114	125	1085	77.97%	10.51%	11.52%
23	2019	12	425	74	144	643	66.10%	11.51%	22.40%
24	2020	12	1019	115	205	1339	76.10%	8.59%	15.31%

e. RFM analysis

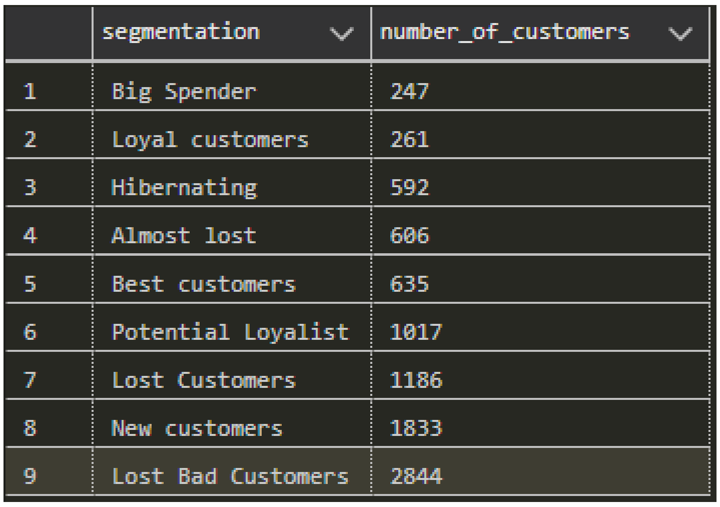
Segment all customers in 2019-2020 into 9 distinct groups by percent ranking and count the total customers of each group.

```
WITH fact_table AS -- tạo bảng fact chung
        (SELECT customer_id
              , transaction_id
              , transaction_time
              , charged_amount
              , sub_category
              , cast(transaction_time as date) as transaction_date
         FROM fact_transaction_2019 fact_2019
                  JOIN dim_scenario scen ON scen.scenario_id =
fact_2019.scenario_id
         WHERE status_id = 1
           AND category = 'Billing'
         UNION ALL
         SELECT customer_id
              , transaction_id
              , transaction_time
              , charged_amount
              , sub_category
              , cast(transaction_time as date) as transaction_date
         FROM fact_transaction_2020 fact_2020
                  JOIN dim_scenario scen ON scen.scenario_id =
fact_2020.scenario_id
         WHERE status_id = 1
           AND category = 'Billing')
,
```

```sql
   rfm_model as (SELECT customer_id,
                        DATEDIFF(DAY, MAX(transaction_time), '2020-12-31') AS
Recency,
                        COUNT(DISTINCT transaction_id)                     AS
Frequency,
                        SUM(charged_amount)                                AS
Monetary
                 FROM fact_table
                 GROUP BY customer_id
   )
,
    rfm_percent as (
        SELECT customer_id
                        , PERCENT_RANK() OVER ( ORDER BY Recency DESC)   AS
r_percent_rank
                        , PERCENT_RANK() OVER ( ORDER BY Frequency ASC) AS
f_percent_rank
                        , PERCENT_RANK() OVER ( ORDER BY Monetary ASC)  AS
m_percent_rank
                   FROM rfm_model
                   )
, tier_table as(
       SELECT customer_id,
              CASE WHEN r_percent_rank <= 0.25 then 4
                  WHEN r_percent_rank <= 0.5 then 3
                  WHEN r_percent_rank <= 0.75 then 2
               ELSE 1 end r_tier,
           CASE WHEN f_percent_rank <= 0.25 then 4
                  WHEN f_percent_rank <= 0.5 then 3
                  WHEN f_percent_rank <= 0.75 then 2
               ELSE 1 end f_tier,
           CASE WHEN m_percent_rank <= 0.25 then 4
                  WHEN m_percent_rank <= 0.5 then 3
                  WHEN m_percent_rank <= 0.75 then 2
               ELSE 1 end m_tier
       FROM rfm_percent
   )
, score_table as (
        SELECT customer_id
                         , CONCAT(r_tier, f_tier, m_tier) as score
       FROM tier_table)
, label_table as (
        SELECT *
                         , (CASE
                              WHEN score = '111' THEN 'Best customers'
                              WHEN score LIKE '[34][34][1-4]' THEN 'Lost Bad
Customers'
                              WHEN score LIKE '[34]2[1-4]' THEN 'Lost Customers'
                              WHEN score LIKE '21[1-4]' THEN 'Almost lost'
```

```
                                WHEN score LIKE '11[2-4]' THEN 'Loyal customers'
                                WHEN score LIKE '[12][1-3]1' THEN 'Big Spender'
                                WHEN score LIKE '[12]4[1-4]' THEN 'New customers'
                                WHEN score LIKE '[34]1[1-4]' THEN 'Hibernating'
                                WHEN score LIKE '[12][23][2-4]' THEN 'Potential
Loyalist'
            END
            ) as label
                    FROM score_table)
SELECT label
    , COUNT(customer_id) as number_of_customers
FROM label_table
GROUP BY label
ORDER BY number_of_customers
```

Output:



	segmentation	number_of_customers
1	Big Spender	247
2	Loyal customers	261
3	Hibernating	592
4	Almost lost	606
5	Best customers	635
6	Potential Loyalist	1017
7	Lost Customers	1186
8	New customers	1833
9	Lost Bad Customers	2844

Note: The percent_rank function allows for greater flexibility in filtering and sorting data, while the Ntile function divides data into a fixed number of equally-sized groups, which may not be suitable for all use cases. By using percent_rank, I can easily filter data based on a particular percentile, such as the top or bottom 50%, which can be useful in various analysis scenarios.