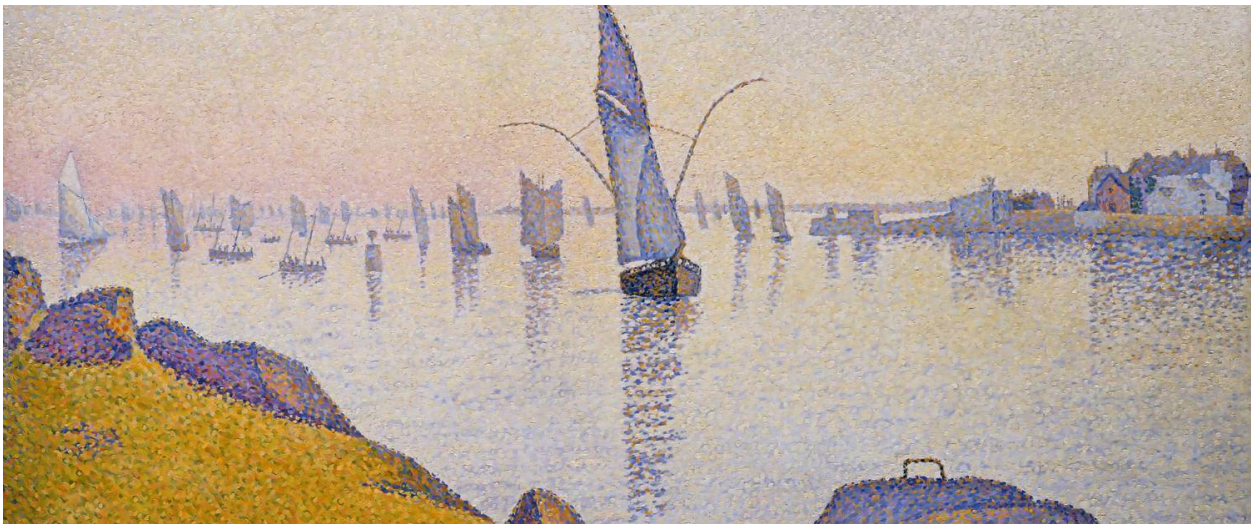

RFM TRANSITION

▼ RFM TRANSITION LOGIC

1. Recency

Recency measures how recently a customer made a purchase. The logic differs based on whether the `time_unit` is **Month** or **Quarter**.

For Month:

- **Definition**: The number of days between the customer's **last transaction date** in the month and the **end of that month**.
- **Logic**:
  - Identify the **last transaction date** (`MAX(Date)`) for each customer within the month.
  - Calculate the difference in days between this date and the **last day of the month** (`EOMONTH(MAX(Date))`).
  - This gives the number of days since the customer's last purchase relative to the end of the month.

For Quarter:

- **Definition**: The number of days between the customer's **last transaction date** in the quarter and the **end of that quarter**.
- **Logic**:
  - Identify the **last transaction date** (`MAX(Date)`) for each customer within the quarter.
  - Determine the **end of the quarter** based on the quarter of the last transaction date:
    - Q1 ends on March 31.
    - Q2 ends on June 30.
    - Q3 ends on September 30.
    - Q4 ends on December 31.
  - Calculate the difference in days between the last transaction date and the end of the quarter.

2. Frequency

Frequency measures how often a customer makes purchases. The logic is the same for both **Month** and **Quarter**, but it is calculated within the respective time period.

For Month and Quarter:

- **Definition**: The average number of days between purchases within the time period (month or quarter).
- **Logic**:
  - Identify the **first transaction date** (`MIN(Date)`) and the **last transaction date** (`MAX(Date)`) for each customer within the time period.
  - Calculate the total number of days between the first and last transaction dates.
  - Divide this by the **number of gaps between transactions**, which is `COUNT(*) - 1` (where `COUNT(*)` is the total number of transactions).
  - This gives the average number of days between purchases.

3. Monetary

Monetary measures the average revenue generated per transaction. The logic is the same for both **Month** and **Quarter**.

For Month and Quarter:

- **Definition**: The average revenue per transaction within the time period (month or quarter).
- **Logic**:
  - Calculate the **total revenue** for each customer within the time period by summing up the product of `Quantity` and `Price` for all transactions.
  - Divide the total revenue by the **total number of transactions** (`COUNT(*)`).
  - This gives the average revenue per transaction.

Key Differences Between Month and Quarter

1. **Time Period**:
   - For **Month**, all calculations are scoped to a single month.
   - For **Quarter**, all calculations are scoped to a single quarter (3 months).
2. **Recency**:
   - For **Month**, the reference point is the **end of the month**.
   - For **Quarter**, the reference point is the **end of the quarter**.
3. **Frequency**:
   - For **Month**, the average days between purchases are calculated within the month.
   - For **Quarter**, the average days between purchases are calculated within the quarter.
4. **Monetary**:
   - The calculation is the same for both, but the transactions considered are limited to the respective time period (month or quarter).

---

1. Input parameters for flexible analysis

```
DECLARE @year INT = 2019;                -- Analysis year
DECLARE @time_unit VARCHAR(10) = 'Quarter';    -- Time unit ('MONTH' or 'QUARTER')
DECLARE @from_period INT = NULL;           -- Starting period (NULL for all)
```

```
DECLARE @to_period INT = NULL;              -- Ending period (NULL for all)

-- Define date range
DECLARE @start_date DATE = DATEFROMPARTS(@year, 1, 1);
DECLARE @end_date DATE = DATEADD(DAY, -1, DATEFROMPARTS(@year + 1, 1, 1));
```

2. Calculate RFM metrics based on selected time unit

```
-- Calculate RFM metrics based on selected time unit
WITH metrics AS (
   SELECT
      CustomerNo,
      YEAR(Date) AS year,
      CASE
         WHEN @time_unit = 'MONTH' THEN MONTH(Date)
         ELSE DATEPART(QUARTER, Date)
      END AS period,
  -- Recency: Number of days since the last transaction to the end of the period
CASE
   -- If time_unit is 'MONTH', calculate recency as days since last transaction to end of month
   WHEN @time_unit = 'MONTH' THEN
      DATEDIFF(DAY, MAX(Date), EOMONTH(MAX(Date)))
   -- If time_unit is 'QUARTER', calculate recency as days since last transaction to end of quarter
   WHEN @time_unit = 'QUARTER' THEN
      DATEDIFF(DAY, MAX(Date),
         DATEFROMPARTS(
            YEAR(MAX(Date)),
            CASE
               WHEN DATEPART(QUARTER, MAX(Date)) = 1 THEN 3  -- End of Q1 is March 31
               WHEN DATEPART(QUARTER, MAX(Date)) = 2 THEN 6  -- End of Q2 is June 30
               WHEN DATEPART(QUARTER, MAX(Date)) = 3 THEN 9  -- End of Q3 is September 30
               WHEN DATEPART(QUARTER, MAX(Date)) = 4 THEN 12 -- End of Q4 is December 31
            END,
            CASE
               WHEN DATEPART(QUARTER, MAX(Date)) = 1 THEN 31 -- March has 31 days
               WHEN DATEPART(QUARTER, MAX(Date)) = 2 THEN 30 -- June has 30 days
               WHEN DATEPART(QUARTER, MAX(Date)) = 3 THEN 30 -- September has 30 days
               WHEN DATEPART(QUARTER, MAX(Date)) = 4 THEN 31 -- December has 31 days
            END
         )
      )
END AS recency,
      -- Frequency: Average number of days between purchases
      CASE
         -- If time_unit is 'MONTH', calculate frequency within the month
         WHEN @time_unit = 'MONTH' THEN
            (DATEDIFF(DAY, MIN(Date), MAX(Date))) / NULLIF(COUNT(*), 0)
         -- If time_unit is 'QUARTER', calculate frequency within the quarter
         WHEN @time_unit = 'QUARTER' THEN
            (DATEDIFF(DAY, MIN(Date), MAX(Date))) / NULLIF(COUNT(*), 0)
      END AS frequency,
```

```
    -- Monetary: Average revenue per transaction
    SUM(Quantity * Price) / NULLIF(COUNT(*), 0) AS monetary
  FROM [sale_transaction-data]
  WHERE YEAR(Date) = @year
  GROUP BY
    CustomerNo,
    YEAR(Date),
    CASE
      WHEN @time_unit = 'MONTH' THEN MONTH(Date)
      ELSE DATEPART(QUARTER, Date)
    END
  HAVING
    (DATEDIFF(DAY, MIN(Date), MAX(Date))) / NULLIF(COUNT(*), 0) > 1  -- Exclude customers with only one transaction
)

-- Save metrics to temporary table
SELECT * INTO #metrics FROM metrics;
```

	CustomerNo	year	period	recency	frequency	monetary
1	141410	2019	4	24	3	79.180000
2	174910	2019	1	7	2	251.575714
3	159390	2019	2	18	2	1043.985161
4	141980	2019	3	16	3	106.434117
5	142170	2019	4	23	2	80.303703
6	181540	2019	4	25	2	112.551428
7	130580	2019	4	46	8	134.026000
8	129890	2019	1	14	6	1191.830000
9	182110	2019	3	15	3	150.970000
10	160110	2019	1	2	2	136.618620
11	126950	2019	4	29	8	75.215000
12	157370	2019	4	36	2	101.964000
13	125570	2019	1	3	9	1896.473333
14	133970	2019	3	2	3	167.992727
15	131130	2019	4	22	2	187.868333
16	178570	2019	4	26	2	3949.427777

3. Calculate percentile values for RFM scoring

```
-- Calculate percentile values for RFM scoring
SELECT
  -- Recency: lower is better
  percentile_disc(0.2) WITHIN GROUP (ORDER BY recency) OVER() AS recency_20,
  percentile_disc(0.4) WITHIN GROUP (ORDER BY recency) OVER() AS recency_40,
  percentile_disc(0.6) WITHIN GROUP (ORDER BY recency) OVER() AS recency_60,
  percentile_disc(0.8) WITHIN GROUP (ORDER BY recency) OVER() AS recency_80,

  -- Frequency: lower is better (days between purchases)
```

```
    percentile_disc(0.2) WITHIN GROUP (ORDER BY frequency) OVER() AS frequency_20,
    percentile_disc(0.4) WITHIN GROUP (ORDER BY frequency) OVER() AS frequency_40,
    percentile_disc(0.6) WITHIN GROUP (ORDER BY frequency) OVER() AS frequency_60,
    percentile_disc(0.8) WITHIN GROUP (ORDER BY frequency) OVER() AS frequency_80,

    -- Monetary: higher is better
    -1 * percentile_disc(0.2) WITHIN GROUP (ORDER BY monetary * -1) OVER() AS monetary_20,
    -1 * percentile_disc(0.4) WITHIN GROUP (ORDER BY monetary * -1) OVER() AS monetary_40,
    -1 * percentile_disc(0.6) WITHIN GROUP (ORDER BY monetary * -1) OVER() AS monetary_60,
    -1 * percentile_disc(0.8) WITHIN GROUP (ORDER BY monetary * -1) OVER() AS monetary_80
INTO #percentile_values
FROM #metrics;
```

recency_20	**recency_40**	recency_60	recency_80	frequency_20	frequency_40	frequency_60	frequency_80
6	15	24	37	2	2	3	6

4.Declare variables for percentile boundaries

```
-- Declare variables for percentile boundaries
DECLARE @recency_20 INT, @recency_40 INT, @recency_60 INT, @recency_80 INT;
DECLARE @frequency_20 DECIMAL(10, 2), @frequency_40 DECIMAL(10, 2), @frequency_60 DECIMAL(10, 2), @frequency
DECLARE @monetary_20 DECIMAL(10, 2), @monetary_40 DECIMAL(10, 2), @monetary_60 DECIMAL(10, 2), @monetary_8

-- Assign values from temporary table to variables
SELECT
    @recency_20 = recency_20,
    @recency_40 = recency_40,
    @recency_60 = recency_60,
    @recency_80 = recency_80,

    @frequency_20 = frequency_20,
    @frequency_40 = frequency_40,
    @frequency_60 = frequency_60,
    @frequency_80 = frequency_80,

    @monetary_20 = monetary_20,
    @monetary_40 = monetary_40,
    @monetary_60 = monetary_60,
    @monetary_80 = monetary_80
FROM
    #percentile_values;
```

5. Calculate RFM scores for each customer and period

```
-- Calculate RFM scores for each customer and period
SELECT
   CustomerNo,
   year,
   period,
   CAST(
      CASE
         WHEN recency <= @recency_20 THEN 5
         WHEN recency <= @recency_40 THEN 4
         WHEN recency <= @recency_60 THEN 3
         WHEN recency <= @recency_80 THEN 2
         ELSE 1
      END AS VARCHAR
   ) + CAST(
      CASE
         WHEN frequency <= @frequency_20 THEN 5
         WHEN frequency <= @frequency_40 THEN 4
         WHEN frequency <= @frequency_60 THEN 3
         WHEN frequency <= @frequency_80 THEN 2
         ELSE 1
      END AS VARCHAR
   ) + CAST(
      CASE
         WHEN monetary >= @monetary_20 THEN 5
         WHEN monetary >= @monetary_40 THEN 4
         WHEN monetary >= @monetary_60 THEN 3
         WHEN monetary >= @monetary_80 THEN 2
         ELSE 1
      END AS VARCHAR
   ) AS rfm_score
INTO #rfm_scores
FROM #metrics;
```

	CustomerNo	year	period	rfm_score
1	141410	2019	4	331
2	174910	2019	1	453
3	159390	2019	2	355
4	141980	2019	3	332
5	142170	2019	4	351
6	181540	2019	4	252
7	130580	2019	4	112
8	129890	2019	1	425
9	182110	2019	3	432
10	160110	2019	1	552
11	126950	2019	4	211
12	157370	2019	4	251
13	125570	2019	1	515
14	133970	2019	3	533
15	131130	2019	4	353
16	178570	2019	4	255

6. Define RFM segments based on RFM scores

```
-- Define RFM segments based on RFM scores
SELECT DISTINCT rfm_score,
  CASE
      -- Champions
      WHEN rfm_score IN ('555', '554', '544', '545', '454', '455', '445') THEN 'Champions'
      -- Loyal Customers
      WHEN rfm_score IN ('543', '444', '435', '355', '354', '345', '344', '335') THEN 'Loyal'
      -- Potential Loyalist
      WHEN rfm_score IN ('553', '551', '552', '541', '542', '533', '532', '531', '452', '451', '442', '441', '431', '453', '433', '432',
      -- New Customers
      WHEN rfm_score IN ('512', '511', '422', '421', '412', '411', '311') THEN 'New Customers'
      -- Promising
      WHEN rfm_score IN ('525', '524', '523', '522', '521', '515', '514', '513', '425', '424', '413', '414', '415', '315', '314', '313') T
      -- Need Attention
      WHEN rfm_score IN ('535', '534', '443', '434', '343', '334', '325', '324') THEN 'Need Attention'
      -- About to Sleep
      WHEN rfm_score IN ('331', '321', '312', '221', '213', '231', '241', '251') THEN 'About to Sleep'
      -- Cannot Lose Them But Losing
```

```
        WHEN rfm_score IN ('155', '154', '144', '214', '215', '115', '114', '113') THEN 'Cannot Lose Them But Losing'
        -- At Risk
        WHEN rfm_score IN ('255', '254', '245', '244', '253', '252', '243', '242', '235', '234', '225', '224', '153', '152', '145', '143
        -- Hibernating
        WHEN rfm_score IN ('332', '322', '233', '232', '223', '222', '132', '123', '122', '212', '211') THEN 'Hibernating'
        -- Lost Customers
        WHEN rfm_score IN ('111', '112', '121', '131', '141', '151') THEN 'Lost Customers'
        ELSE 'Unclassified'
    END AS rfm_segment
INTO #rfm_segment
FROM #rfm_scores;
```

	rfm_score	rfm_segment
1	112	Lost Customers
2	113	Cannot Lose Them But Losing
3	114	Cannot Lose Them But Losing
4	115	Cannot Lose Them But Losing
5	121	Lost Customers
6	122	Hibernating
7	123	Hibernating
8	124	At Risk
9	125	At Risk
10	131	Lost Customers
11	132	Hibernating
12	133	At Risk
13	134	At Risk
14	135	At Risk
15	151	Lost Customers
16	152	At Risk

7. Analyze customer transitions between segments across periods

```
-- Analyze customer transitions between segments across periods
SELECT
   a.year,
   @time_unit AS time_unit,
   a.period AS period_from,
   b.period AS period_to,
   a.rfm_score AS rfm_from,
   rs_from.rfm_segment AS segment_from,
   b.rfm_score AS rfm_to,
   rs_to.rfm_segment AS segment_to,
   COUNT(DISTINCT a.CustomerNo) AS num_customers
FROM
   #rfm_scores a
JOIN
   #rfm_scores b ON a.CustomerNo = b.CustomerNo
            AND a.year = b.year
            AND a.period + 1 = b.period
JOIN
   #rfm_segment rs_from ON a.rfm_score = rs_from.rfm_score
JOIN
   #rfm_segment rs_to ON b.rfm_score = rs_to.rfm_score
WHERE
   a.year = @year
   AND (@from_period IS NULL OR a.period = @from_period)
   AND (@to_period IS NULL OR b.period = @to_period)
GROUP BY
   a.year,
   a.period,
   b.period,
   a.rfm_score,
   rs_from.rfm_segment,
   b.rfm_score,
   rs_to.rfm_segment
ORDER BY
   a.year,
   a.period,
   b.period,
   rs_from.rfm_segment,
   rs_to.rfm_segment;
```

	year	time_unit	period_from	period_to	rfm_from	segment_from	rfm_to	segment_to	num_customers
1	2019	Quarter	1	2	235	At Risk	255	At Risk	1
2	2019	Quarter	1	2	124	At Risk	115	Cannot Lose Them But Losing	1
3	2019	Quarter	1	2	224	At Risk	322	Hibernating	1
4	2019	Quarter	1	2	225	At Risk	535	Need Attention	1
5	2019	Quarter	1	2	125	At Risk	425	Promising	1
6	2019	Quarter	1	2	225	At Risk	414	Promising	1
7	2019	Quarter	1	2	215	Cannot Lose Th…	255	At Risk	1
8	2019	Quarter	1	2	215	Cannot Lose Th…	425	Promising	1
9	2019	Quarter	1	2	454	Champions	115	Cannot Lose Them But Losing	1
10	2019	Quarter	1	2	554	Champions	355	Loyal	1
11	2019	Quarter	1	2	454	Champions	434	Need Attention	1
12	2019	Quarter	1	2	123	Hibernating	124	At Risk	1
13	2019	Quarter	1	2	322	Hibernating	322	Hibernating	1
14	2019	Quarter	1	2	435	Loyal	435	Loyal	1
15	2019	Quarter	1	2	535	Need Attention	455	Champions	1