RFM segmentation

https://www.kaggle.com/datasets/gabrielramos87/an-online-shop-business

1. Overview RFM

RFM Analysis (Recency, Frequency, Monetary) is a widely used customer segmentation technique that analyzes customer behavior based on three key dimensions:

1. **Recency (R):**
   - **Definition:** How recently a customer has made a purchase.
   - **Interpretation:** Customers who purchased more recently are typically more engaged and likely to respond to promotions.
   - **In Practice:** A lower recency value (i.e., fewer days since the last transaction) indicates a higher likelihood of repeat business.
2. **Frequency (F):**
   - **Definition:** How often a customer makes a purchase within a given timeframe.
   - **Interpretation:** More frequent purchases indicate a higher level of loyalty and engagement.
   - **In Practice:** This is often calculated as the average number of days between purchases (or the total number of transactions in a period). A lower average interval means the customer is purchasing more frequently.
3. **Monetary (M):**
   - **Definition:** How much money a customer spends on average per purchase.
   - **Interpretation:** Customers who spend more per transaction are typically more valuable.
   - **In Practice:** This is usually measured as the average order value or total spending divided by the number of transactions.

How RFM Analysis Works in My Project

- **Data Preparation:**

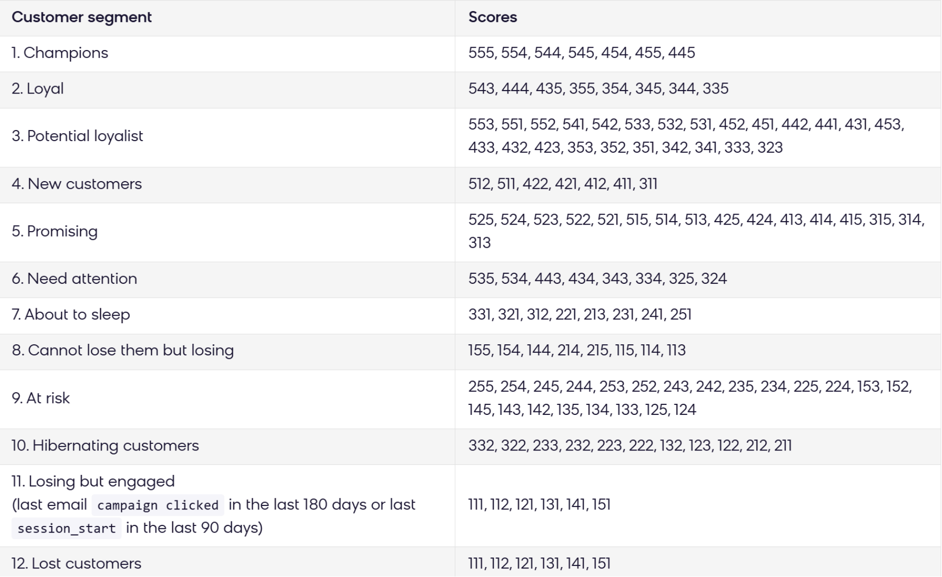
  I aggregate transactional data from our sales database to compute:

  - **Recency:** Calculated as the number of days from the most recent transaction to a defined reporting date.

    - **Frequency:** Calculated as the average number of days between the first and the most recent transaction (if the customer has more than one purchase).
    - **Monetary:** Calculated as the average revenue per transaction.
- **Scoring:**

  Using percentile-based thresholds (derived via functions by `PERCENTILE_DISC`), I assign scores (1 to 5) for each RFM dimension:

    - **Recency Score:** Lower recency (more recent purchase) gets a higher score (e.g., 5 is best).
    - **Frequency Score:** Lower average gap (i.e., more frequent purchases) gets a higher score.
    - **Monetary Score:** Higher average spending gets a higher score when inverted, so that a score of 5 represents the top 20% of spenders.



Customer segment	Scores
1. Champions	555, 554, 544, 545, 454, 455, 445
2. Loyal	543, 444, 435, 355, 354, 345, 344, 335
3. Potential loyalist	553, 551, 552, 541, 542, 533, 532, 531, 452, 451, 442, 441, 431, 453, 433, 432, 423, 353, 352, 351, 342, 341, 333, 323
4. New customers	512, 511, 422, 421, 412, 411, 311
5. Promising	525, 524, 523, 522, 521, 515, 514, 513, 425, 424, 413, 414, 415, 315, 314, 313
6. Need attention	535, 534, 443, 434, 343, 334, 325, 324
7. About to sleep	331, 321, 312, 221, 213, 231, 241, 251
8. Cannot lose them but losing	155, 154, 144, 214, 215, 115, 114, 113
9. At risk	255, 254, 245, 244, 253, 252, 243, 242, 235, 234, 225, 224, 153, 152, 145, 143, 142, 135, 134, 133, 125, 124
10. Hibernating customers	332, 322, 233, 232, 223, 222, 132, 123, 122, 212, 211
11. Losing but engaged (last email `campaign clicked` in the last 180 days or last `session_start` in the last 90 days)	111, 112, 121, 131, 141, 151
12. Lost customers	111, 112, 121, 131, 141, 151

https://documentation.bloomreach.com/engagement/docs/rfm-segmentation

Benefits of `PERCENTILE_DISC` Over `NTILE`

Feature	PERCENTILE_DISC	NTILE
Precision	Picks an actual value from the dataset, ensuring percentile cutoffs align with real data points.	May create artificial boundaries by evenly distributing rows.
Use Case	Ideal for percentile-based segmentation, such as RFM scoring (top 20%, 40%, etc.).	Good for distributing data into equal-sized buckets but may not reflect true percentiles.
Handling of Ties	Always selects a valid data point, avoiding interpolation.	Can place tied values into different buckets, making segmentation less precise.
Flexibility	Allows defining exact percentile cutoffs (e.g., top 20% customers).	Forces equal groups, which may not align with business needs.
Suitability for RFM	Matches percentile-based ranking well, ensuring fair customer segmentation.	May create unequal groups when data is skewed.

Example: Disadvantage of `NTILE` (Distributing Data into Equal-Sized Buckets)

Scenario:

Imagine we have **100 customers** and we want to segment them into **4 RFM groups** based on `recency`. We use `NTILE(4) OVER (ORDER BY recency)` to create quartiles.

Dataset (Sorted by Recency - Days Since Last Purchase)

Customer	Recency (days)
C1	1
C2	2
C3	2
C4	3
C5	4
...	...
C96	200
C97	210
C98	220
C99	230
C100	250

Using `NTILE(4) OVER (ORDER BY recency)`

Since `NTILE(4)` forces an **equal number of customers in each bucket**, each quartile will contain **25 customers**.

NTILE Group	Recency Range (Days)
Quartile 1 (Top 25%)	**1 - 50 days**
Quartile 2 (25-50%)	**51 - 100 days**
Quartile 3 (50-75%)	**101 - 150 days**
Quartile 4 (Bottom 25%)	**151 - 250 days**

Problem 1: Artificial Cutoffs

- A customer with **51 days recency** is placed in **Quartile 2**, while a customer with **50 days recency** is in **Quartile 1**.
- These two customers are **almost identical** in behavior but are **assigned to different groups**.

Problem 2: Uneven Real-World Distribution

Imagine **80 customers have recency ≤ 50 days**, but `NTILE(4)` forces each quartile to have **only 25 customers**.

- As a result, `NTILE(4)` **ignores the natural data distribution** and forces an **even split** instead of reflecting the **true purchasing behavior**.

How `PERCENTILE_DISC` Solves This Issue

Instead of blindly splitting into 4 equal groups, `PERCENTILE_DISC(0.25)` would **pick the actual recency value** closest to the 25th percentile.

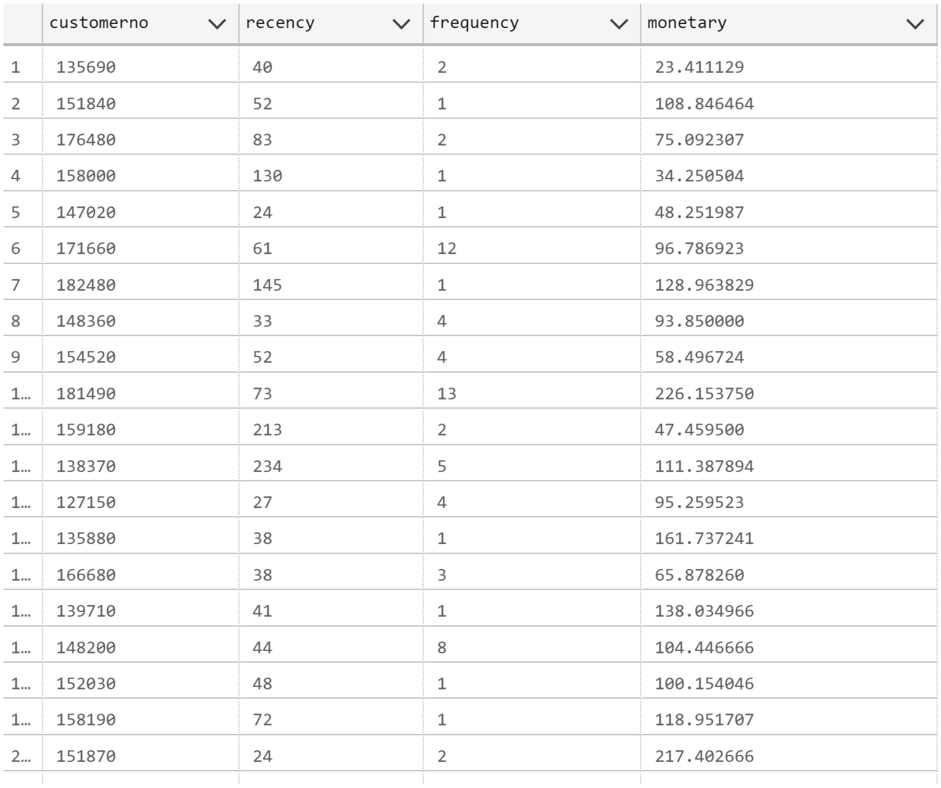
- This ensures that **recency cutoffs align with actual customer behavior**, rather than forcing an **equal distribution** of customers.

2. Analysis

Caculate Recency , Frequency , Monetary

```
DECLARE @year INT = 2019;
DECLARE @start_date DATE = DATEFROMPARTS(@year, 1, 1);
DECLARE @end_date DATE = DATEADD(DAY, 1, DATEFROMPARTS(@year, 12, 31));

with RFM_data as (
 SELECT
        customerno,
        DATEDIFF(DAY, MAX(date), @end_date) AS recency,
DATEDIFF(DAY, MIN(date), MAX(date))/ COUNT(*)  AS frequency,
        SUM(price*quantity) / NULLIF(COUNT(*), 0) AS monetary
    FROM [Uk_Sale_Transaction]
    WHERE YEAR(date) = @year
    GROUP BY customerno
    HAVING
    DATEDIFF(DAY, MIN(Date), MAX(Date)) / NULLIF(COUNT(*) , 0) > 0)
```



	customerno	recency	frequency	monetary
1	135690	40	2	23.411129
2	151840	52	1	108.846464
3	176480	83	2	75.092307
4	158000	130	1	34.250504
5	147020	24	1	48.251987
6	171660	61	12	96.786923
7	182480	145	1	128.963829
8	148360	33	4	93.850000
9	154520	52	4	58.496724
1…	181490	73	13	226.153750
1…	159180	213	2	47.459500
1…	138370	234	5	111.387894
1…	127150	27	4	95.259523
1…	135880	38	1	161.737241
1…	166680	38	3	65.878260
1…	139710	41	1	138.034966
1…	148200	44	8	104.446666
1…	152030	48	1	100.154046
1…	158190	72	1	118.951707
2…	151870	24	2	217.402666

There are 1935 rows on the table above

Use Percentile_Disc to deviding data into 5 parts

```
SELECT
    -- Recency: lower values are 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 values are better
    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 values are better (negative order to reverse)
    -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 rfm_data;
```

Using DECLARE allows the percentile_disc values to update dynamically as the data is updated

```
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_80

-- Gán giá trị từ bảng tạm vào biến
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;
   select
    @recency_20 as recency_20,
    @recency_40 as recency_40,
    @recency_60 as recency_60,
    @recency_80 as recency_80 ,
     @frequency_20 as  frequency_20,
    @frequency_40 as  frequency_40,
    @frequency_60 as frequency_60,
    @frequency_80 as frequency_80,

    @monetary_20 as monetary_20,
    @monetary_40 as monetary_40,
    @monetary_60 as monetary_60,
    @monetary_80 as  monetary_80;
```

recency_20	recency_40	recency_60	recency_80	frequency_20	frequency_40	frequency_60	frequency_80
33	46	69	114	1	2	3	5

Recency (R):

- `recency_20 = 33` → The top 20% of customers (best customers in terms of recency) made a purchase within the last 33 days.
- `recency_40 = 46` → The top 40% of customers made a purchase within the last 46 days.
- Similarly, `recency_60` and `recency_80` show that as recency increases, customers are less recent.

Frequency (F):

- `frequency_20 = 1` → The top 20% of customers **purchase every day**
- `frequency_40 = 2` → The top 40% of customers **purchase every 2 days on average**.
- `frequency_80 = 5` → The top 80% of customers **purchase every 5 days on average**, and so on.

Monetary (M):

- `monetary_20 = 194.70` → The top 20% of customers (best spenders) spent at least **$194.70**.
- `monetary_40 = 131.58` → The top 40% of customers spent at least **$131.58**.
- `monetary_80 = 63.99` → The top 80% of customers spent at least **$63.99**.
- Higher monetary values mean the customer spends more.

Assign scores (1 to 5) for each RFM dimension

```
DECLARE @year INT = 2019;
DECLARE @start_date DATE = DATEFROMPARTS(@year, 1, 1);
DECLARE @end_date DATE = DATEADD(DAY, 1, DATEFROMPARTS(@year, 12, 31));

with RFM_data as (
 SELECT
        customerno,
        DATEDIFF(DAY, MAX(date), @end_date) AS recency,
DATEDIFF(DAY, MIN(date), MAX(date))/ COUNT(*)  AS frequency,
        SUM(price*quantity) / NULLIF(COUNT(*), 0) AS monetary
    FROM Uk_Sale_Transaction
    WHERE YEAR(date) = @year
    GROUP BY customerno
    HAVING
    DATEDIFF(DAY, MIN(Date), MAX(Date)) / NULLIF(COUNT(*) - 1, 0) > 0)

SELECT
    CustomerNo,
    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 rfm_data;
```

Defining rfm_segement base on score 1-5

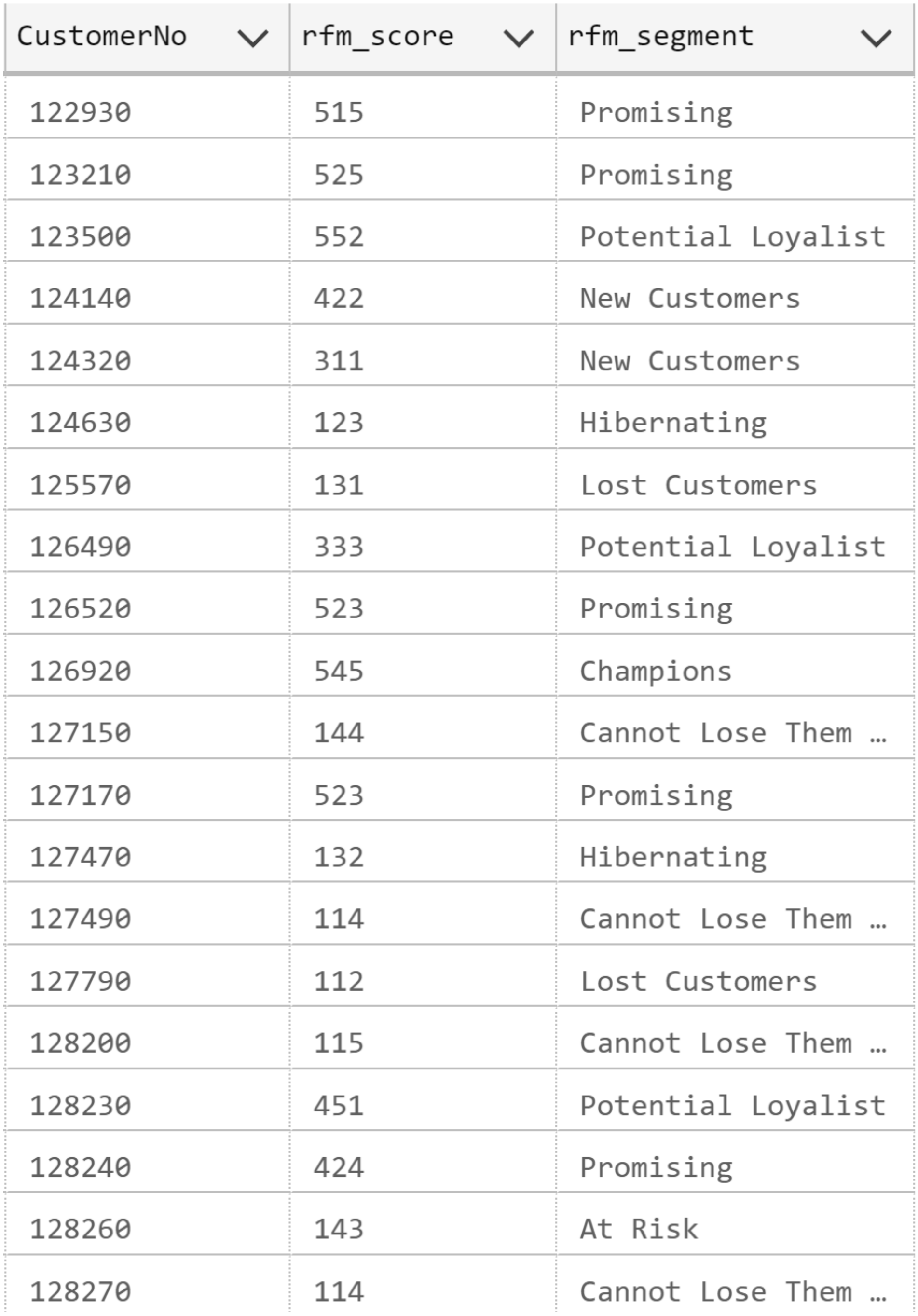
```
SELECT rfm_score,
 CASE
    -- Champions
    WHEN rfm_score IN ('555', '554', '544', '545', '454', '455', '445') THEN 'Champions'
    -- Loyal Customers
    WHEN Rfm_score IN ('543', '444', '435', '355', '354', '345', '344', '335') THEN 'Loyal'
    -- Potential Loyalist
    WHEN Rfm_score IN ('553', '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')
    -- 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', '14
    -- 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
```

Question

Question 1 : "What is the RFM score and corresponding customer segment for each distinct customer?”

```
select distinct sc.CustomerNo , rs.rfm_score, rs.rfm_segment
from #rfm_segment  rs inner join #rfm_scores sc on rs.rfm_score=sc.rfm_score
```

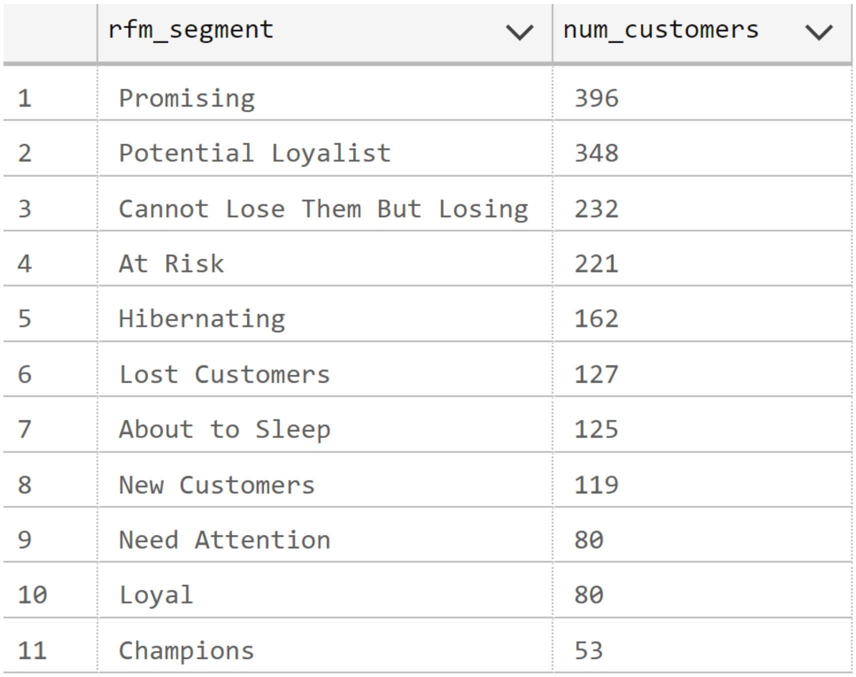


CustomerNo	rfm_score	rfm_segment
122930	515	Promising
123210	525	Promising
123500	552	Potential Loyalist
124140	422	New Customers
124320	311	New Customers
124630	123	Hibernating
125570	131	Lost Customers
126490	333	Potential Loyalist
126520	523	Promising
126920	545	Champions
127150	144	Cannot Lose Them …
127170	523	Promising
127470	132	Hibernating
127490	114	Cannot Lose Them …
127790	112	Lost Customers
128200	115	Cannot Lose Them …
128230	451	Potential Loyalist
128240	424	Promising
128260	143	At Risk
128270	114	Cannot Lose Them …

Question 2 : How many customers are in each Rfm segment ?

```
SELECT
   rs.rfm_segment,
   COUNT(DISTINCT r.CustomerNo) AS num_customers
FROM #rfm_scores r
JOIN #rfm_segment rs ON r.rfm_score = rs.rfm_score
GROUP BY rs.rfm_segment
```

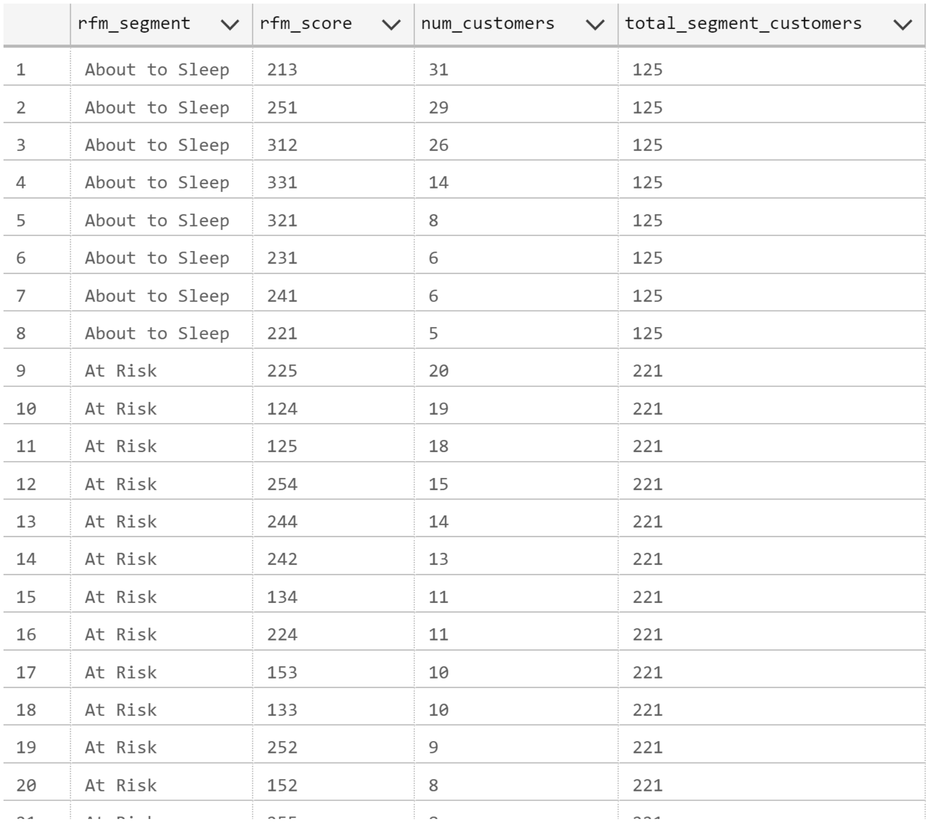
```
ORDER BY num_customers DESC;
```



	rfm_segment	num_customers
1	Promising	396
2	Potential Loyalist	348
3	Cannot Lose Them But Losing	232
4	At Risk	221
5	Hibernating	162
6	Lost Customers	127
7	About to Sleep	125
8	New Customers	119
9	Need Attention	80
10	Loyal	80
11	Champions	53

Question 3 : How many customers are in each rfm_score and total customers in each rfm_segment ?

```
SELECT
    rs.rfm_segment,
    r.rfm_score,
    COUNT(DISTINCT r.CustomerNo) AS num_customers,
    SUM(COUNT(DISTINCT r.CustomerNo)) OVER (PARTITION BY rs.rfm_segment) AS total_segment_customers
FROM #rfm_scores r
JOIN #rfm_segment rs ON r.rfm_score = rs.rfm_score
GROUP BY rs.rfm_segment, r.rfm_score
ORDER BY rs.rfm_segment, num_customers DESC;
```



	rfm_segment	rfm_score	num_customers	total_segment_customers
1	About to Sleep	213	31	125
2	About to Sleep	251	29	125
3	About to Sleep	312	26	125
4	About to Sleep	331	14	125
5	About to Sleep	321	8	125
6	About to Sleep	231	6	125
7	About to Sleep	241	6	125
8	About to Sleep	221	5	125
9	At Risk	225	20	221
10	At Risk	124	19	221
11	At Risk	125	18	221
12	At Risk	254	15	221
13	At Risk	244	14	221
14	At Risk	242	13	221
15	At Risk	134	11	221
16	At Risk	224	11	221
17	At Risk	153	10	221
18	At Risk	133	10	221
19	At Risk	252	9	221
20	At Risk	152	8	221
21	At Risk	255	8	221