Segmenting customers with Google Big-query

Problem statement:

Perform a RFM analysis for a chain of retail stores that sells a lot of different items and categories. The stores need to adjust their marketing budget and have better targeting of customers so they need to know which customers to focus on and how important they are for the business.

What are Customer segmentation models:

There are four main customer segmentation models:

1. Technographic segmentation
2. Customer behavior segmentation
3. Needs-based segmentation
4. Customer status segmentation

- **Technographic segmentation** refers to segmenting your customers based on a technology or a group of technologies. Based data about the technology products and services that customers use, such as the type of devices they own, the software they use, and the online services they subscribe to.

- **Behavioral segmentation** divides the market into minor groups based on people’s buying habits, likes, and wants. Customers performing similar buying patterns can be clubbed together in a group that will be targeted with higher precision. For example price-focused segment, quality and the brand-focused segment.

- **Needs-based segmentation** involves segmenting customer groups by their financial, emotional, and physical needs. Whether they want to find a budget-friendly gift, or a desk chair cushion for their back pain, you can discover what your target customer needs through targeted needs-based segmentation.

- **Customer status or customer lifecycle segmentation** refers to grouping customers based on their place in the customer lifecycle. This includes leads, new customers, loyal/long-time customers, at-risk customers, and churned customers. RFM is a method to achieve this.

What is RFM score?

We all know that valuing customers based on a single parameter is flawed. The biggest value customer may have only purchased once or twice in a year, or the most frequent purchaser may have a value so low that it is almost not profitable to service them.
One parameter will never give you an accurate view of your customer base, and you’ll ignore customer lifetime value.

We calculate the RFM score by attributing a numerical value for each of the criteria.

The customer gets more points -
- if they bought in the recent past,
- bought many times or
- if the purchase value is larger.

Combine these three values to create the RFM score.

This RFM score can then be used to segment your customer data platform.

Analysis of the customer RFM values will create some standard segments. The UK Data & Marketing Association (DMA) laid out 11 segments, and specified marketing strategies according to their respective characteristics.

Data:

- **InvoiceNo**: Invoice number. Nominal, a 6-digit integral number uniquely assigned to each transaction. If this code starts with letter 'c', it indicates a cancellation.
- **StockCode**: Product (item) code. Nominal, a 5-digit integral number uniquely assigned to each distinct product.
- **Description**: Product (item) name. Nominal.
- **Quantity**: The quantities of each product (item) per transaction. Numeric.
- **InvoiceDate**: Invoice Date and time. Numeric, the day and time when each transaction was generated.
- **UnitPrice**: Unit price. Numeric, Product price per unit in sterling.
- **CustomerID**: Customer number. Nominal, a 5-digit integral number uniquely assigned to each customer.
- **Country**: Country name. Nominal, the name of the country where each customer resides

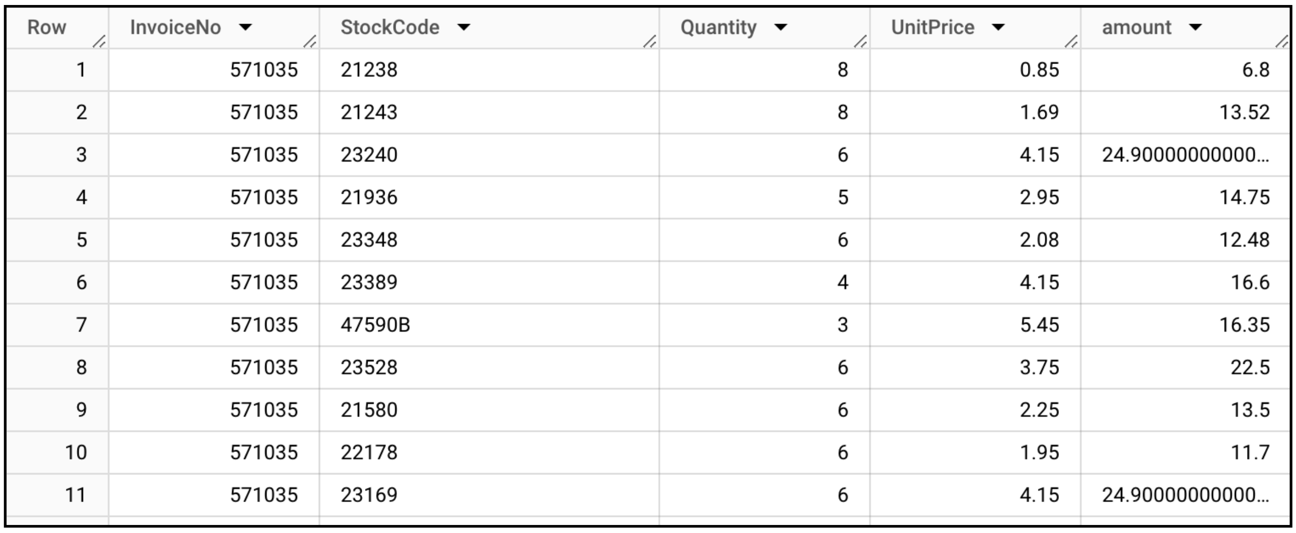
RFM Segmentation in BigQuery The RFM Segmentation can be executed using these five steps:
1. Data processing
2. Compute for recency, frequency, and monetary values per customer
3. Determine quantiles for each RFM metric
4. Assign scores for each RFM metric
5. Define the RFM segments using the scores in step 4

Data set link: sales.csv

Data processing:

If we look at the data we can see that there are products that have been bought in quantities more than one and we have unit price for those products but we do not have the total cost of that product.
So the first thing we’re gonna do is find the total cost for that product i.e., quantity * unit price -

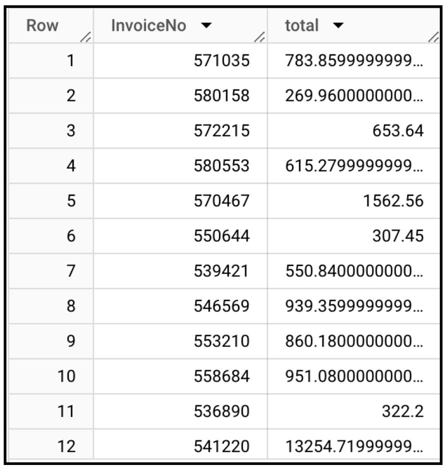
```sql
SELECT
  InvoiceNo,
  StockCode,
  Quantity,
  UnitPrice,
  (Quantity*UnitPrice) AS amount
FROM
  `customer_segmentation.sales`
```



Row	InvoiceNo	StockCode	Quantity	UnitPrice	amount
1	571035	21238	8	0.85	6.8
2	571035	21243	8	1.69	13.52
3	571035	23240	6	4.15	24.90000000000...
4	571035	21936	5	2.95	14.75
5	571035	23348	6	2.08	12.48
6	571035	23389	4	4.15	16.6
7	571035	47590B	3	5.45	16.35
8	571035	23528	6	3.75	22.5
9	571035	21580	6	2.25	13.5
10	571035	22178	6	1.95	11.7
11	571035	23169	6	4.15	24.90000000000...

Now that we have got the total cost for each product we need to find out the amount spent on each visit.
For each invoice id there may be different products, and till now we have calculated the total for each product, but we do not have the total bill amount for individual invoice ids.
For this we use the above query and create a CTE. Then group it by invoice id and sum the total cost, getting the actual bill amount.

```
WITH
  bills AS (
  SELECT
    InvoiceNo,
    (Quantity * UnitPrice) AS amount
  FROM
    `customer_segmentation.sales` )
SELECT
  InvoiceNo,
  SUM(amount) AS total
FROM
  bills
GROUP BY
  InvoiceNo;
```



Row	InvoiceNo	total
1	571035	783.8599999999...
2	580158	269.9600000000...
3	572215	653.64
4	580553	615.2799999999...
5	570467	1562.56
6	550644	307.45
7	539421	550.8400000000...
8	546569	939.3599999999...
9	553210	860.1800000000...
10	558684	951.0800000000...
11	536890	322.2
12	541220	13254.71999999...

Save this data as a **'bill'** table in the same dataset by using the save button below the query editor.
Note: we can do this without saving this a table but that will make the query very long.

Compute for recency, frequency and monetary value per customer:

Because we will be joining the bill and sales table we will get multiple rows on the key that we are going to join ie: InvoiceNo, so we'll just take one row per InvoiceNo.
For that we will use the row number and get the data from the sales table that we need.

Now for calculating RFM

- For monetary, this is just a simple sum of sales,
- while for frequency, this is a count of distinct invoice numbers per customer for the time they have been a customer ie: the number of separate purchases/ num of months they have been a customer. So we will get the first and last purchase for all customers and also the number of purchases
- For calculating recency we will first get the last purchase for each customer We will join the `bill` table that we saved with the `sales` table and add the total cost on the customer level for monetary value.

We will join the `bill` table that we saved with the `sales` table and add the total cost on the customer level for monetary value.

```
SELECT
  CustomerID,
  DATE(MAX(InvoiceDate)) AS last_purchase_date,
  DATE(MIN(InvoiceDate)) AS first_purchase_date,
  COUNT(DISTINCT InvoiceNo) AS num_purchases,
  SUM(total) AS monetary,
FROM (
  SELECT
    s.CustomerID,
    s.InvoiceDate,
    s.InvoiceNo,
    b.total,
    ROW_NUMBER() OVER(PARTITION BY s.InvoiceNo ORDER BY
s.InvoiceNo) AS RN
  FROM
    `customer_segmentation.sales` s
  LEFT JOIN
    `customer_segmentation.bill` b
  ON
    s.InvoiceNo=b.InvoiceNo ) A
WHERE A.RN = 1
GROUP BY CustomerID;
```

We can save this table **'monetary'**.

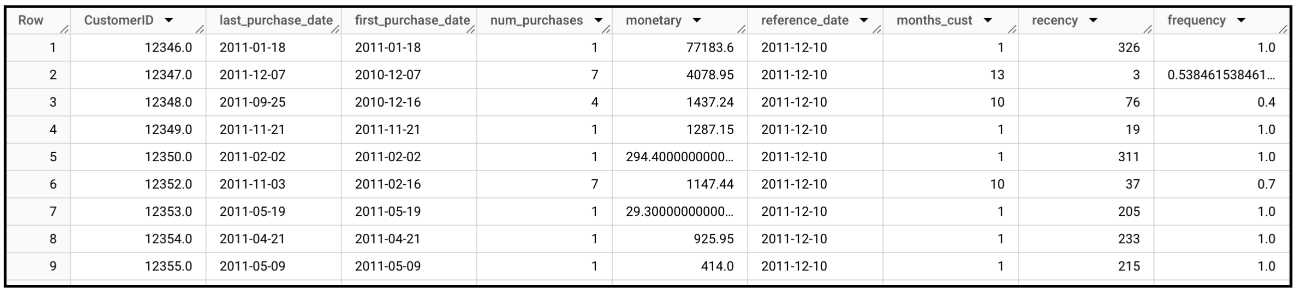
Recency

For recency, we chose a reference date, which is the most recent purchase in the dataset. In other situations, one may select the date when the data was analyzed instead.

After choosing the reference date, we get the date difference between the reference date and the last purchase date of each customer. This is the recency value for that particular customer.

For frequency we calculate the months the person has been a customer by difference in first and last purchase +1 (for when first and last month are same and the customer should be considered a customer for at least 1 month)

```
SELECT
  *,
  DATE_DIFF(reference_date, last_purchase_date, DAY) AS recency,
  num_purchases/ (months_cust) AS frequency,
FROM (
  SELECT
    *,
    MAX(last_purchase_date) OVER () + 1 AS reference_date,
    DATE_DIFF(last_purchase_date, first_purchase_date, month)+1 AS
months_cust
  FROM
    `customer_segmentation.monetary` )
ORDER BY
  CustomerID;
```



Row	CustomerID	last_purchase_date	first_purchase_date	num_purchases	monetary	reference_date	months_cust	recency	frequency
1	12346.0	2011-01-18	2011-01-18	1	77183.6	2011-12-10	1	326	1.0
2	12347.0	2011-12-07	2010-12-07	7	4078.95	2011-12-10	13	3	0.538461538461…
3	12348.0	2011-09-25	2010-12-16	4	1437.24	2011-12-10	10	76	0.4
4	12349.0	2011-11-21	2011-11-21	1	1287.15	2011-12-10	1	19	1.0
5	12350.0	2011-02-02	2011-02-02	1	294.4000000000…	2011-12-10	1	311	1.0
6	12352.0	2011-11-03	2011-02-16	7	1147.44	2011-12-10	10	37	0.7
7	12353.0	2011-05-19	2011-05-19	1	29.30000000000…	2011-12-10	1	205	1.0
8	12354.0	2011-04-21	2011-04-21	1	925.95	2011-12-10	1	233	1.0
9	12355.0	2011-05-09	2011-05-09	1	414.0	2011-12-10	1	215	1.0

Now save this as **‘RFM’** table.

Determine quantiles for each RFM values

The next step would be to group the customers into quintiles in terms of their RFM values — we divide the customers into 5 equal groups, according to how high and low they scored in the RFM metrics.
<u>The main advantage of using percentile is we do not have to change or set the values. It will be automatically calculated.</u>

We do this for each of recency, frequency and monetary values per customer.
I used BigQuery’s **APPROX_QUANTILES()** and **OFFSET()** to achieve this.

- APPROX_QUANTILES() : This is one of the approximate aggregate functions, which are scalable in terms of memory usage and time, but produce approximate results instead of exact results.

- OFFSET() : accesses an ARRAY element by position and returns the element. The approximate_quantiles will return an array for each percentile and for creating quintiles out of it we will need values at 20, 40 and so on. We save those values as m20, m40 for monetary and f, r for frequency and recency respectively.

```
SELECT
  a.*,
  --All percentiles for MONETARY
  b.percentiles[offset(20)] AS m20,
  b.percentiles[offset(40)] AS m40,
  b.percentiles[offset(60)] AS m60,
  b.percentiles[offset(80)] AS m80,
  b.percentiles[offset(100)] AS m100,
  --All percentiles for FREQUENCY
  c.percentiles[offset(20)] AS f20,
  c.percentiles[offset(40)] AS f40,
  c.percentiles[offset(60)] AS f60,
  c.percentiles[offset(80)] AS f80,
  c.percentiles[offset(100)] AS f100,
  --All percentiles for RECENCY
  d.percentiles[offset(20)] AS r20,
  d.percentiles[offset(40)] AS r40,
  d.percentiles[offset(60)] AS r60,
  d.percentiles[offset(80)] AS r80,
  d.percentiles[offset(100)] AS r100
FROM
  `customer_segmentation.RFM` a,
  (SELECT APPROX_QUANTILES(monetary, 100) percentiles FROM
  `customer_segmentation.RFM`) b,
  (SELECT APPROX_QUANTILES(frequency, 100) percentiles FROM
  `customer_segmentation.RFM`) c,
  (SELECT APPROX_QUANTILES(recency, 100) percentiles FROM
  `customer_segmentation.RFM`) d
ORDER BY CustomerID;
```

Save this as a new table called **‘quantile’**.

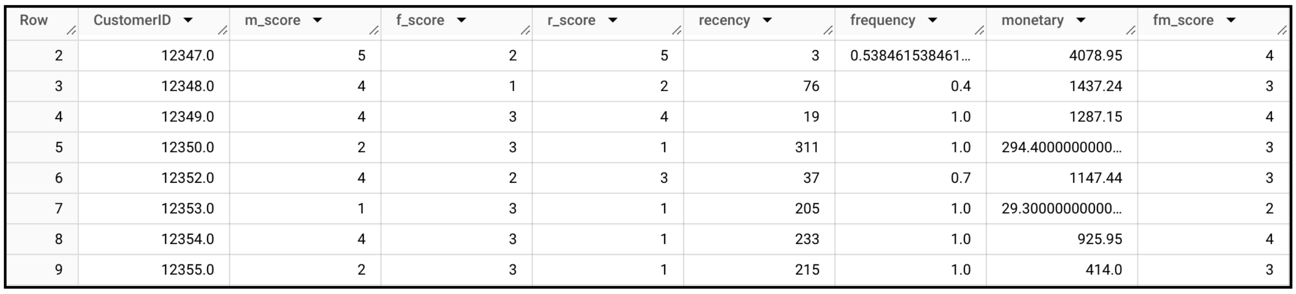
Assigning scores for each RFM metric:

Now that we know how each customer fares relative to other customers in terms of RFM values, we can now assign scores from 1 to 5.

Just keep in mind that while with F and M, we give higher scores for higher quintiles, R should be reversed as more recent customers should be scored higher in this metric. Frequency and Monetary value are combined (as both of them are indicative to purchase volume anyway) to reduce the possible options from 125 to 50.

We will use CASE to get values and assign scores accordingly, so we just get the data from the **‘quantile’** table that we stored assign scores.

```
SELECT
  CustomerID,
  m_score, f_score, r_score,
  recency, frequency, monetary,
  CAST(ROUND((f_score + m_score) / 2, 0) AS INT64) AS fm_score
FROM (
  SELECT
    *,
    CASE
      WHEN monetary <= m20 THEN 1
      WHEN monetary <= m40 AND monetary > m20 THEN 2
      WHEN monetary <= m60 AND monetary > m40 THEN 3
      WHEN monetary <= m80 AND monetary > m60 THEN 4
      WHEN monetary <= m100 AND monetary > m80 THEN 5
    END AS m_score,
    CASE
      WHEN frequency <= f20 THEN 1
      WHEN frequency <= f40 AND frequency > f20 THEN 2
      WHEN frequency <= f60 AND frequency > f40 THEN 3
      WHEN frequency <= f80 AND frequency > f60 THEN 4
      WHEN frequency <= f100 AND frequency > f80 THEN 5
    END AS f_score,
    --Recency scoring is reversed
    CASE
      WHEN recency <= r20 THEN 5
      WHEN recency <= r40 AND recency > r20 THEN 4
      WHEN recency <= r60 AND recency > r40 THEN 3
      WHEN recency <= r80 AND recency > r60 THEN 2
      WHEN recency <= r100 AND recency > r80 THEN 1
    END AS r_score,
  FROM `customer_segmentation.quantile`
);
```



Row	CustomerID	m_score	f_score	r_score	recency	frequency	monetary	fm_score
2	12347.0	5	2	5	3	0.538461538461...	4078.95	4
3	12348.0	4	1	2	76	0.4	1437.24	3
4	12349.0	4	3	4	19	1.0	1287.15	4
5	12350.0	2	3	1	311	1.0	294.4000000000...	3
6	12352.0	4	2	3	37	0.7	1147.44	3
7	12353.0	1	3	1	205	1.0	29.30000000000...	2
8	12354.0	4	3	1	233	1.0	925.95	4
9	12355.0	2	3	1	215	1.0	414.0	3

We can save this as a new table called **‘score’**.

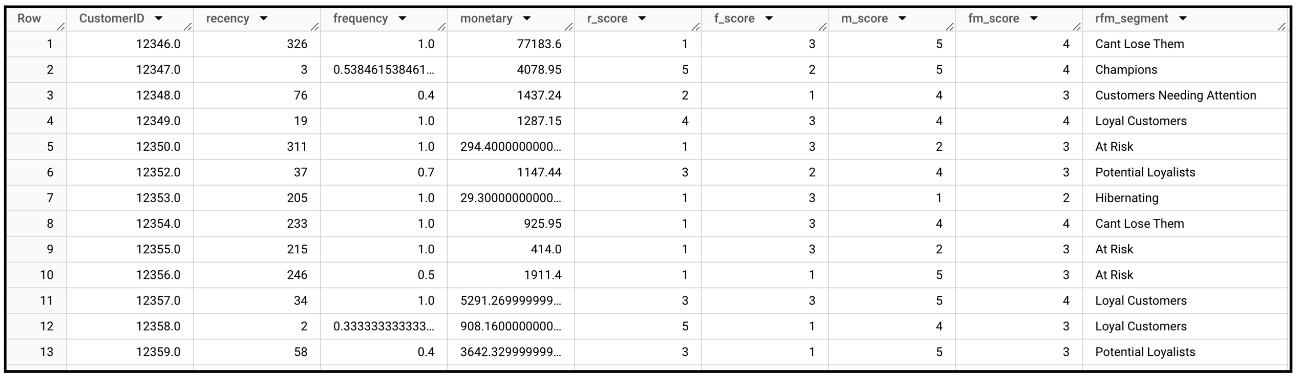
Define the RFM segments using these scores:

The next step is to combine the scores we obtained to define the RFM segment each customer will belong to.
As there are 5 groups for each of the R, F, and M metrics, there are 125 potential permutations.

We will be using the 11 personas in the DMA as a guide and define the R vs. FM scores accordingly.

Below is the SQL code for the same.

```sql
SELECT
  CustomerID,
  recency, frequency, monetary,
  r_score, f_score, m_score,
  fm_score,
  CASE
    WHEN (r_score = 5 AND fm_score = 5)
      OR (r_score = 5 AND fm_score = 4)
      OR (r_score = 4 AND fm_score = 5) THEN 'Champions'
    WHEN (r_score = 5 AND fm_score = 3)
      OR (r_score = 4 AND fm_score = 4)
      OR (r_score = 3 AND fm_score = 5)
      OR (r_score = 3 AND fm_score = 4) THEN 'Loyal Customers'
    WHEN (r_score = 5 AND fm_score = 2)
      OR (r_score = 4 AND fm_score = 2)
      OR (r_score = 3 AND fm_score = 3)
      OR (r_score = 4 AND fm_score = 3) THEN 'Potential Loyalists'
    WHEN r_score = 5 AND fm_score = 1 THEN 'Recent Customers'
    WHEN (r_score = 4 AND fm_score = 1)
      OR (r_score = 3 AND fm_score = 1) THEN 'Promising'
    WHEN (r_score = 3 AND fm_score = 2)
      OR (r_score = 2 AND fm_score = 3)
      OR (r_score = 2 AND fm_score = 2) THEN 'Customers Needing
Attention'
    WHEN r_score = 2 AND fm_score = 1 THEN 'About to Sleep'
    WHEN (r_score = 2 AND fm_score = 5)
      OR (r_score = 2 AND fm_score = 4)
      OR (r_score = 1 AND fm_score = 3) THEN 'At Risk'
    WHEN (r_score = 1 AND fm_score = 5)
      OR (r_score = 1 AND fm_score = 4) THEN 'Cant Lose Them'
    WHEN r_score = 1 AND fm_score = 2 THEN 'Hibernating'
    WHEN r_score = 1 AND fm_score = 1 THEN 'Lost'
  END AS rfm_segment
FROM
  `customer_segmentation.score`
ORDER BY
  CustomerID;
```



Row	CustomerID	recency	frequency	monetary	r_score	f_score	m_score	fm_score	rfm_segment
1	12346.0	326	1.0	77183.6	1	3	5	4	Cant Lose Them
2	12347.0	3	0.538461538461…	4078.95	5	2	5	4	Champions
3	12348.0	76	0.4	1437.24	2	1	4	3	Customers Needing Attention
4	12349.0	19	1.0	1287.15	4	3	4	4	Loyal Customers
5	12350.0	311	1.0	294.4000000000…	1	3	2	3	At Risk
6	12352.0	37	0.7	1147.44	3	2	4	3	Potential Loyalists
7	12353.0	205	1.0	29.30000000000…	1	3	1	2	Hibernating
8	12354.0	233	1.0	925.95	1	3	4	4	Cant Lose Them
9	12355.0	215	1.0	414.0	1	3	2	3	At Risk
10	12356.0	246	0.5	1911.4	1	1	5	3	At Risk
11	12357.0	34	1.0	5291.269999999…	3	3	5	4	Loyal Customers
12	12358.0	2	0.333333333333…	908.1600000000…	5	1	4	3	Loyal Customers
13	12359.0	58	0.4	3642.329999999…	3	1	5	3	Potential Loyalists

We are done with our customer segmentation using RFM model.

This type of segmentation focuses on the actual buying behavior and ignores the differences in motivations, intentions, and lifestyles of consumers.

RFM is nonetheless a useful start-off point, and because of its simplicity can be executed fast and in an automated way, giving companies the power to act and decide on business strategies swiftly.

Final data, after segmentation: cust_segments.csv