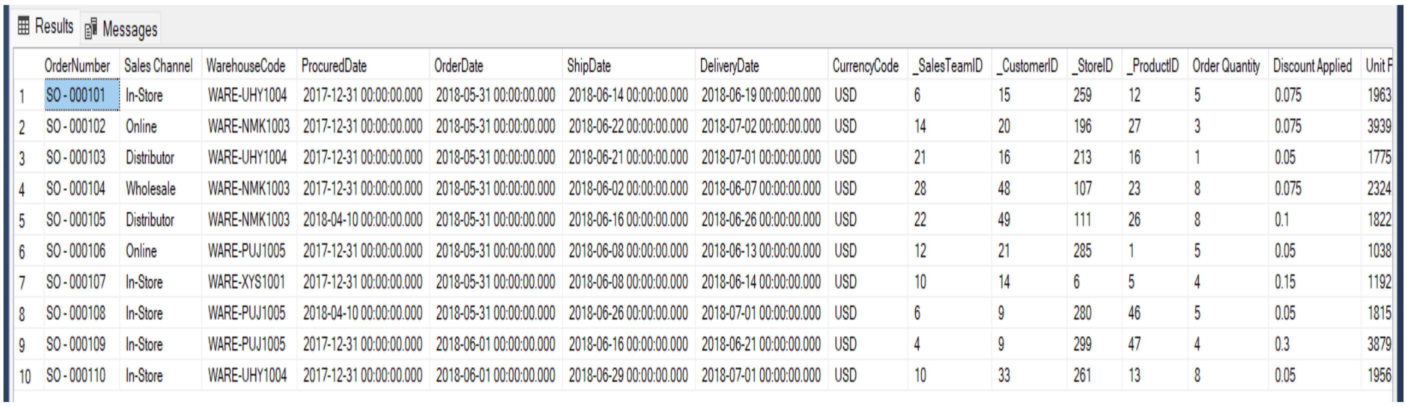
RFM Analysis

using SQL

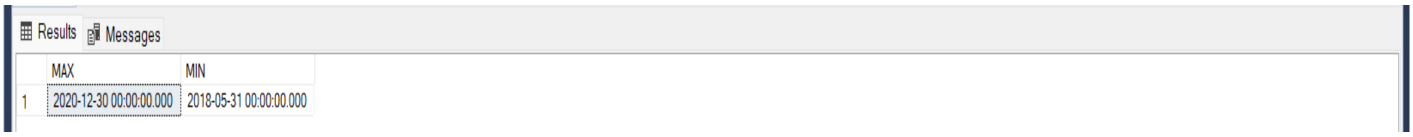
```sql
/*
RFM Analysis | Value Segmentation | Customer Segmentation

Skills used: Joins, Unions, CTE's, Temp Tables,Views, Windows Functions, Aggregate Functions, CASE,
Converting Data Types

--==> This means insights/inferences
*/

--Lets have a look at the data
Select top 10 * FROM PortfolioProjects..['Sales Orders Data']
```



Results | Messages

	OrderNumber	Sales Channel	WarehouseCode	ProcuredDate	OrderDate	ShipDate	DeliveryDate	CurrencyCode	_SalesTeamID	_CustomerID	_StoreID	_ProductID	Order Quantity	Discount Applied	Unit F
1	SO - 000101	In-Store	WARE-UHY1004	2017-12-31 00:00:00.000	2018-05-31 00:00:00.000	2018-06-14 00:00:00.000	2018-06-19 00:00:00.000	USD	6	15	259	12	5	0.075	1963
2	SO - 000102	Online	WARE-NMK1003	2017-12-31 00:00:00.000	2018-05-31 00:00:00.000	2018-06-22 00:00:00.000	2018-07-02 00:00:00.000	USD	14	20	196	27	3	0.075	3939
3	SO - 000103	Distributor	WARE-UHY1004	2017-12-31 00:00:00.000	2018-05-31 00:00:00.000	2018-06-21 00:00:00.000	2018-07-01 00:00:00.000	USD	21	16	213	16	1	0.05	1775
4	SO - 000104	Wholesale	WARE-NMK1003	2017-12-31 00:00:00.000	2018-05-31 00:00:00.000	2018-06-02 00:00:00.000	2018-06-07 00:00:00.000	USD	28	48	107	23	8	0.075	2324
5	SO - 000105	Distributor	WARE-NMK1003	2018-04-10 00:00:00.000	2018-05-31 00:00:00.000	2018-06-16 00:00:00.000	2018-06-26 00:00:00.000	USD	22	49	111	26	8	0.1	1822
6	SO - 000106	Online	WARE-PUJ1005	2017-12-31 00:00:00.000	2018-05-31 00:00:00.000	2018-06-08 00:00:00.000	2018-06-13 00:00:00.000	USD	12	21	285	1	5	0.05	1038
7	SO - 000107	In-Store	WARE-XYS1001	2017-12-31 00:00:00.000	2018-05-31 00:00:00.000	2018-06-08 00:00:00.000	2018-06-14 00:00:00.000	USD	10	14	6	5	4	0.15	1192
8	SO - 000108	In-Store	WARE-PUJ1005	2018-04-10 00:00:00.000	2018-05-31 00:00:00.000	2018-06-26 00:00:00.000	2018-07-01 00:00:00.000	USD	6	9	280	46	5	0.05	1815
9	SO - 000109	In-Store	WARE-PUJ1005	2017-12-31 00:00:00.000	2018-06-01 00:00:00.000	2018-06-16 00:00:00.000	2018-06-21 00:00:00.000	USD	4	9	299	47	4	0.3	3879
10	SO - 000110	In-Store	WARE-UHY1004	2017-12-31 00:00:00.000	2018-06-01 00:00:00.000	2018-06-29 00:00:00.000	2018-07-01 00:00:00.000	USD	10	33	261	13	8	0.05	1956

```sql
--Get the range of dates for the order data
Select
	MAX(OrderDate) AS MAX,
	MIN(OrderDate) AS MIN
FROM PortfolioProjects..['Sales Orders Data']
```

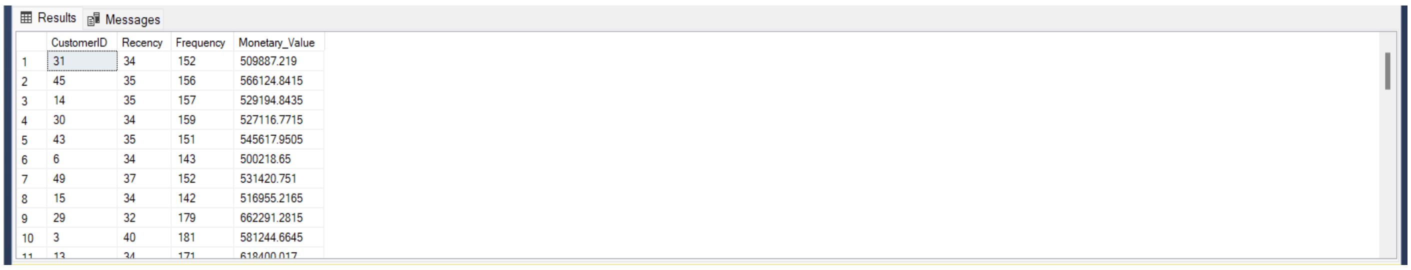


Results | Messages

	MAX	MIN
1	2020-12-30 00:00:00.000	2018-05-31 00:00:00.000

```sql
--==> Data is from May 2018 to Dec 2020
```

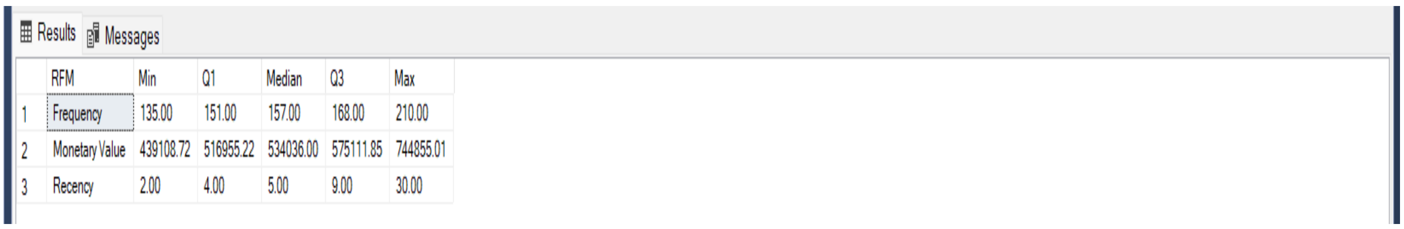
```sql
--Since its a bit out-dated data, so lets declare a today variable for better calculations
DECLARE @today_date AS DATE = '2021-01-31';

--Calculating the RFM
SELECT
	_CustomerID AS CustomerID
	,Datediff(day,MAX(OrderDate),@today_date) AS Recency
	,Count(OrderNumber) AS Frequency
	,Sum([Unit Price] - ([Unit Price]*[Discount Applied] - [Unit Cost])) AS Monetary_Value
FROM PortfolioProjects..['Sales Orders Data']
GROUP BY _CustomerID
```



Results | Messages

	CustomerID	Recency	Frequency	Monetary_Value
1	31	34	152	509887.219
2	45	35	156	566124.8415
3	14	35	157	529194.8435
4	30	34	159	527116.7715
5	43	35	151	545617.9505
6	6	34	143	500218.65
7	49	37	152	531420.751
8	15	34	142	516955.2165
9	29	32	179	662291.2815
10	3	40	181	581244.6645
11	13	34	171	618400.017

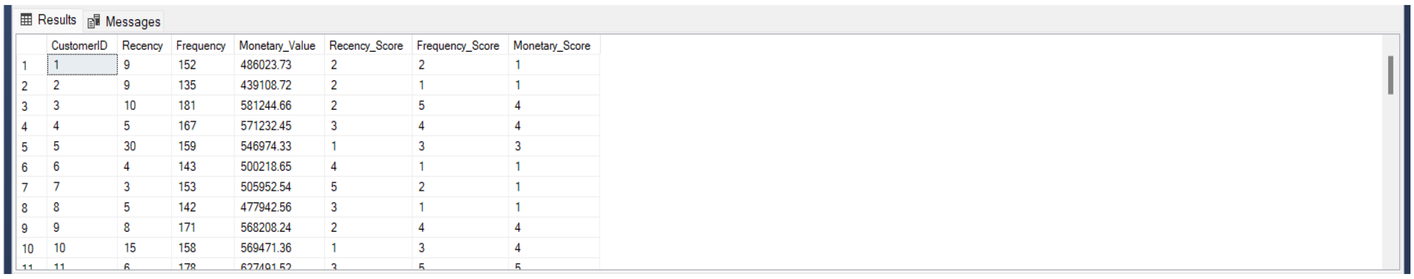
```sql
---Lets understand the distribution of RFM Values by Five Number Summary

--Calculate RFM Values
DECLARE @today_date AS DATE = '2021-01-01';
WITH RFM_CALC AS (
       SELECT
              _CustomerID AS CustomerID
              ,Datediff(day,MAX(OrderDate),@today_date) AS Recency
              ,Count(OrderNumber) AS Frequency
              ,CAST(Sum([Unit Price] - ([Unit Price]*[Discount Applied] - [Unit Cost])) AS
decimal(16,2)) AS Monetary_Value
       FROM PortfolioProjects..['Sales Orders Data']
       GROUP BY _CustomerID
),
--Minimum & Maximum Values
MinMax AS (
       Select
              Min(Recency) AS Rmin,
              Max(Recency) AS Rmax,
              Min(Frequency) AS Fmin,
              Max(Frequency) AS Fmax,
              Min(Monetary_Value) AS Mmin,
              Max(Monetary_Value) AS Mmax
       FROM RFM_CALC
)
--Fivenumber Summary for Monetary Value
SELECT DISTINCT
       'Monetary Value' AS RFM,
       M.Mmin AS Min,
       PERCENTILE_DISC(0.25) WITHIN GROUP (ORDER BY Monetary_Value) OVER () as Q1,
       PERCENTILE_DISC(0.50) WITHIN GROUP (ORDER BY Monetary_Value) OVER () as Median,
       PERCENTILE_DISC(0.75) WITHIN GROUP (ORDER BY Monetary_Value) OVER () as Q3,
       M.Mmax AS Max
FROM MinMax M JOIN RFM_CALC ON 1=1
UNION
--Fivenumber Summary for Frequency
SELECT DISTINCT
       'Frequency' AS RFM,
       F.Fmin AS Min,
       PERCENTILE_DISC(0.25) WITHIN GROUP (ORDER BY Frequency) OVER () as Q1,
       PERCENTILE_DISC(0.50) WITHIN GROUP (ORDER BY Frequency) OVER () as Median,
       PERCENTILE_DISC(0.75) WITHIN GROUP (ORDER BY Frequency) OVER () as Q3,
       F.Fmax AS Max
FROM MinMax F JOIN RFM_CALC ON 1=1
UNION
--Fivenumber Summary for Recency
SELECT DISTINCT
       'Recency' AS RFM,
       R.Rmin AS Min,
       PERCENTILE_DISC(0.25) WITHIN GROUP (ORDER BY Recency) OVER () as Q1,
       PERCENTILE_DISC(0.50) WITHIN GROUP (ORDER BY Recency) OVER () as Median,
       PERCENTILE_DISC(0.75) WITHIN GROUP (ORDER BY Recency) OVER () as Q3,
       R.Rmax AS MAX
FROM MinMax R JOIN RFM_CALC ON 1=1
```



	RFM	Min	Q1	Median	Q3	Max
1	Frequency	135.00	151.00	157.00	168.00	210.00
2	Monetary Value	439108.72	516955.22	534036.00	575111.85	744855.01
3	Recency	2.00	4.00	5.00	9.00	30.00

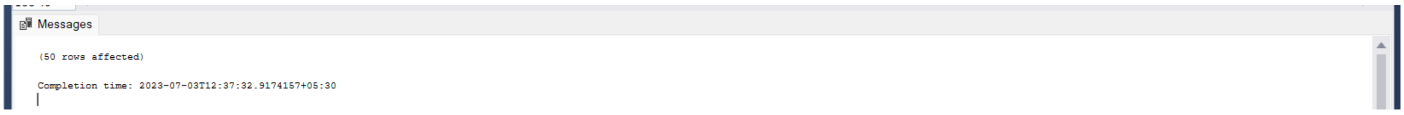
```sql
--==> Data is rightly-skewed
```

```sql
----lets partition RFM Values on the scale of 1 to 5 scores as the ranges of RFM are not very big

--Lets calculate RFM Values
DECLARE @today_date AS DATE = '2021-01-01';
WITH RFM_CALC AS (
	SELECT
		_CustomerID AS CustomerID
		,Datediff(day,MAX(OrderDate), @today_date) AS Recency
		,Count(OrderNumber) AS Frequency
		,CAST(Sum([Unit Price] - ([Unit Price]*[Discount Applied] - [Unit Cost])) AS
decimal(16,2)) AS Monetary_Value
	FROM PortfolioProjects..['Sales Orders Data']
	GROUP BY _CustomerID
)
-- Calculate RMF Scores
SELECT
	CustomerID
	,Recency
	,Frequency
	,Monetary_Value
	,NTILE(5) OVER(ORDER BY Recency DESC) AS Recency_Score
	,NTILE(5) OVER(ORDER BY Frequency ASC) AS Frequency_Score
	,NTILE(5) OVER(ORDER BY Monetary_Value ASC) AS Monetary_Score
FROM
	RFM_CALC
ORDER BY
	CustomerID
```



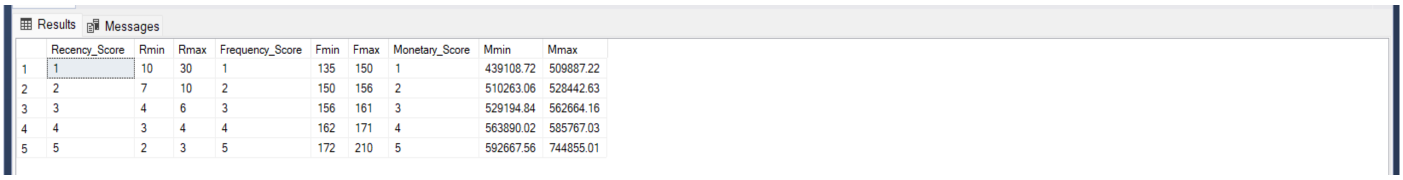
	CustomerID	Recency	Frequency	Monetary_Value	Recency_Score	Frequency_Score	Monetary_Score
1	1	9	152	486023.73	2	2	1
2	2	9	135	439108.72	2	1	1
3	3	10	181	581244.66	2	5	4
4	4	5	167	571232.45	3	4	4
5	5	30	159	546974.33	1	3	3
6	6	4	143	500218.65	4	1	1
7	7	3	153	505952.54	5	2	1
8	8	5	142	477942.56	3	1	1
9	9	8	171	568208.24	2	4	4
10	10	15	158	569471.36	1	3	4
11	11	6	178	627491.52	3	5	5

```sql
----Lets store the above result as a temporary table for further analytics

--Lets calculate RFM Values
WITH RFM_CALC AS (
	SELECT
		_CustomerID AS CustomerID
		,Datediff(day,MAX(OrderDate),'2021-01-01') AS Recency
		,Count(OrderNumber) AS Frequency
		,CAST(Sum([Unit Price] - ([Unit Price]*[Discount Applied] - [Unit Cost])) AS
decimal(16,2)) AS Monetary_Value
	FROM PortfolioProjects..['Sales Orders Data']
	GROUP BY _CustomerID
)
-- Calculate RMF Scores
SELECT
	CustomerID
	,Recency
	,Frequency
	,Monetary_Value
	,NTILE(5) OVER(ORDER BY Recency DESC) AS Recency_Score
	,NTILE(5) OVER(ORDER BY Frequency ASC) AS Frequency_Score
	,NTILE(5) OVER(ORDER BY Monetary_Value ASC) AS Monetary_Score
INTO #RFM_Value_Score
FROM
	RFM_CALC
```



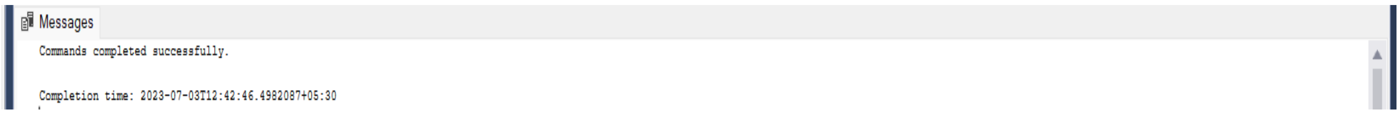
```
(50 rows affected)

Completion time: 2023-07-03T12:37:32.9174157+05:30
```

```sql
----Lets check the Ranges of RFM by Scores using the temp table created above

WITH Recency_Range AS (
	Select
		row_number() Over(Order by Recency_Score) AS I,
		Recency_Score,
		Min(Recency) AS Rmin,
		Max(Recency) AS Rmax
	FROM #RFM_Value_Score
	GROUP BY Recency_Score
),
Frequency_Range AS (
	Select
		row_number() Over(Order by Frequency_Score) AS I,
		Frequency_Score,
		Min(Frequency) AS Fmin,
		Max(Frequency) AS Fmax
	FROM #RFM_Value_Score
	GROUP BY Frequency_Score
),
Monetary_Range AS (
	Select
		row_number() Over(Order by Monetary_Score) AS I,
		Monetary_Score,
		Min(Monetary_Value) AS Mmin,
		Max(Monetary_Value) AS Mmax
	FROM #RFM_Value_Score
	GROUP BY Monetary_Score
)
Select
	Recency_Score,Rmin,Rmax,
	Frequency_Score,Fmin,Fmax,
	Monetary_Score,Mmin,Mmax
FROM Recency_Range R
Join Frequency_Range F
On R.I = F.I
Join Monetary_Range M
On R.I = M.I
```


Results | Messages

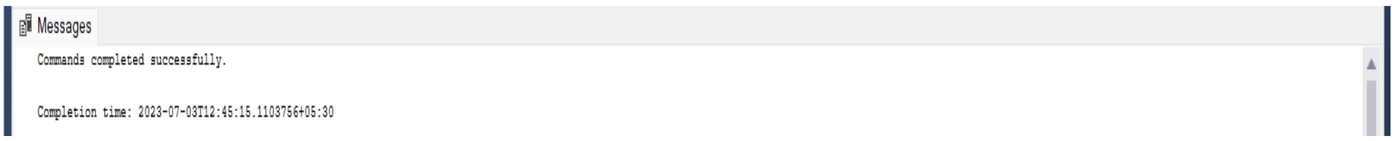
	Recency_Score	Rmin	Rmax	Frequency_Score	Fmin	Fmax	Monetary_Score	Mmin	Mmax
1	1	10	30	1	135	150	1	439108.72	509887.22
2	2	7	10	2	150	156	2	510263.06	528442.63
3	3	4	6	3	156	161	3	529194.84	562664.16
4	4	3	4	4	162	171	4	563890.02	585767.03
5	5	2	3	5	172	210	5	592667.56	744855.01

```sql
----Create the Value Segments & Customer Segments based on RFM Score & Average RFM Score & store as a View for further Analytics & Visualization

--As we can't use the variable directly in the View, Lets create a Function to get the Recency``1
CREATE FUNCTION GetRecency(@today_date DATE, @orderDate DATE)
RETURNS INT
AS
BEGIN
    RETURN DATEDIFF(day, @orderDate, @today_date);
END;
```


Messages

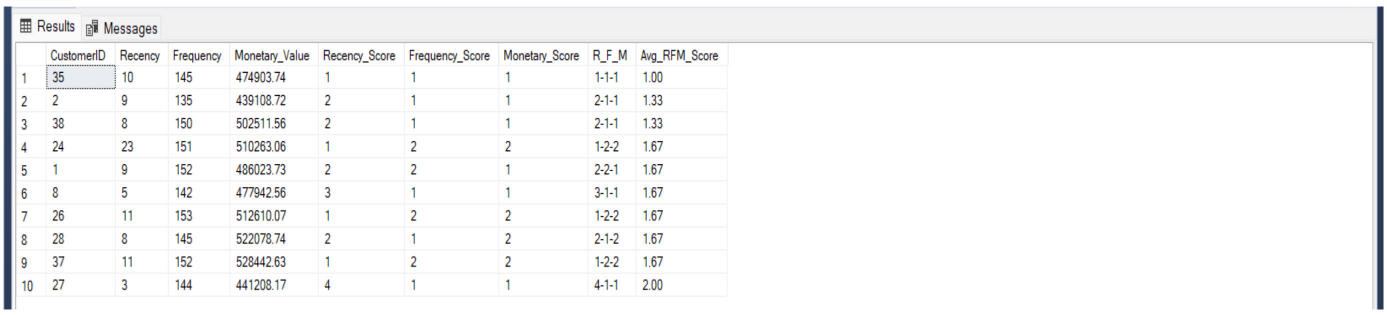
```
Commands completed successfully.


Completion time: 2023-07-03T12:42:46.4982087+05:30
```

```sql
--DropView if exixted
DROP VIEW IF EXISTS RFM_View;

--Create a View for RFM Values & RFM Scores
CREATE VIEW RFM_View AS
--Calculate RFM Values
WITH RFM_CALC AS (
    SELECT
        _CustomerID AS CustomerID,
        dbo.GetRecency('2021-01-01', MAX(OrderDate)) AS Recency,
        COUNT(OrderNumber) AS Frequency,
        CAST(SUM([Unit Price] - ([Unit Price]*[Discount Applied] - [Unit Cost])) AS decimal(16,2))
AS Monetary_Value
    FROM PortfolioProjects..['Sales Orders Data']
    GROUP BY _CustomerID
),
-- Calculate RMF Scores
RFM_SCORES AS (
SELECT
       CustomerID
       ,Recency
       ,Frequency
       ,Monetary_Value
       ,NTILE(5) OVER(ORDER BY Recency DESC) AS Recency_Score
       ,NTILE(5) OVER(ORDER BY Frequency ASC) AS Frequency_Score
       ,NTILE(5) OVER(ORDER BY Monetary_Value ASC) AS Monetary_Score
FROM RFM_CALC
),
-- Calculate Avg RFM Score
RFM_AVG_SCORE AS (
       Select
              CustomerID
              ,CONCAT_WS('-',Recency_Score,Frequency_Score,Monetary_Score) AS R_F_M
              ,CAST((CAST(Recency_Score AS Float) + Frequency_Score + Monetary_Score)/3 AS
DECIMAL(16,2)) AS Avg_RFM_Score
       FROM RFM_SCORES
)
Select
       T1.CustomerID
       ,Recency,Frequency,Monetary_Value
       ,Recency_Score,Frequency_Score,Monetary_Score
       ,R_F_M,Avg_RFM_Score
FROM RFM_SCORES T1
JOIN RFM_AVG_SCORE T2
ON T1.CustomerID = T2.CustomerID
```



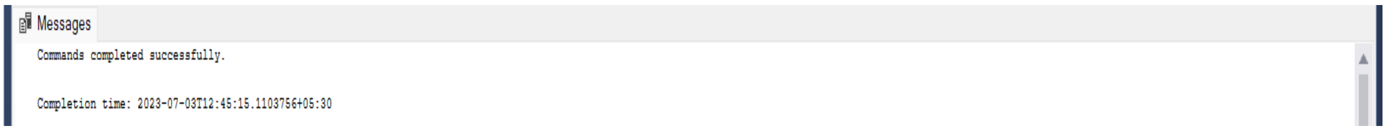
```
Commands completed successfully.

Completion time: 2023-07-03T12:45:15.1103756+05:30
```

```sql
SELECT TOP 10 * FROM RFM_View ORDER BY Avg_RFM_Score
```



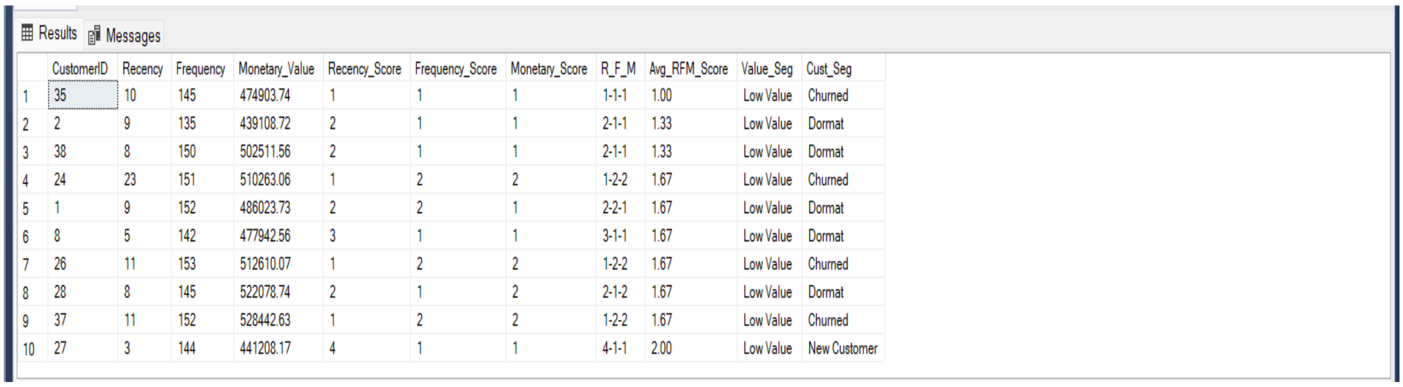
	CustomerID	Recency	Frequency	Monetary_Value	Recency_Score	Frequency_Score	Monetary_Score	R_F_M	Avg_RFM_Score
1	35	10	145	474903.74	1	1	1	1-1-1	1.00
2	2	9	135	439108.72	2	1	1	2-1-1	1.33
3	38	8	150	502511.56	2	1	1	2-1-1	1.33
4	24	23	151	510263.06	1	2	2	1-2-2	1.67
5	1	9	152	486023.73	2	2	1	2-2-1	1.67
6	8	5	142	477942.56	3	1	1	3-1-1	1.67
7	26	11	153	512610.07	1	2	2	1-2-2	1.67
8	28	8	145	522078.74	2	1	2	2-1-2	1.67
9	37	11	152	528442.63	1	2	2	1-2-2	1.67
10	27	3	144	441208.17	4	1	1	4-1-1	2.00

```sql
--Drop View if already exists
DROP VIEW IF EXISTS Customer_Segmentaion;

----Create a View for the Customer Segments & Value Segments using the View "RFM_View"
CREATE VIEW Customer_Segmentaion AS
Select *
       , CASE WHEN Avg_RFM_Score >= 4 THEN 'High Value'
                   WHEN Avg_RFM_Score >= 2.5 AND Avg_RFM_Score < 4 THEN 'Mid Value'
                   WHEN Avg_RFM_Score > 0 AND Avg_RFM_Score < 2.5 THEN 'Low Value'
       END AS Value_Seg --Value Segment
       , CASE WHEN Frequency_Score >= 4 and Recency_Score >= 4 and Monetary_Score >= 4 THEN 'VIP'
                   WHEN Frequency_Score >= 3 and Monetary_Score < 4 THEN 'Regular'
                   WHEN Recency_Score <= 3 and Recency_Score > 1 THEN 'Dormat'
                   WHEN Recency_Score = 1 THEN 'Churned'
                   WHEN Recency_Score >= 4 and Frequency_Score <= 4 THEN 'New Customer'
       END AS Cust_Seg --Customer Segment
FROM RFM_View
```



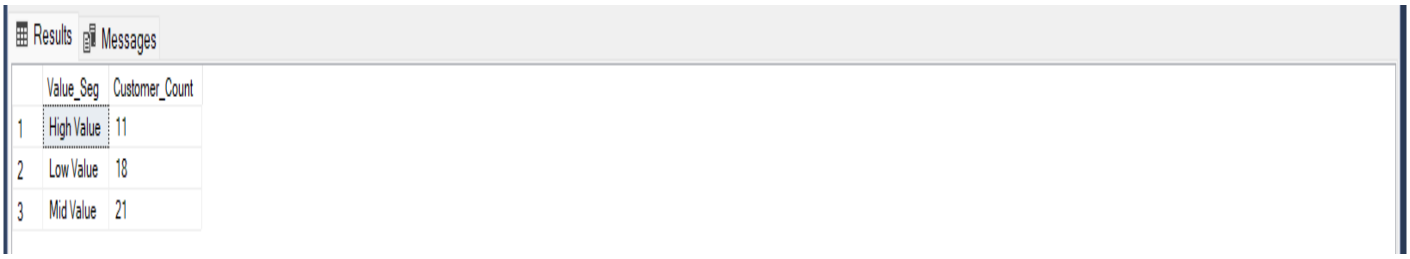
```
Commands completed successfully.

Completion time: 2023-07-03T12:45:15.1103756+05:30
```

```sql
SELECT TOP 10 * FROM Customer_Segmentaion ORDER BY Avg_RFM_Score
```



	CustomerID	Recency	Frequency	Monetary_Value	Recency_Score	Frequency_Score	Monetary_Score	R_F_M	Avg_RFM_Score	Value_Seg	Cust_Seg
1	35	10	145	474903.74	1	1	1	1-1-1	1.00	Low Value	Churned
2	2	9	135	439108.72	2	1	1	2-1-1	1.33	Low Value	Dormat
3	38	8	150	502511.56	2	1	1	2-1-1	1.33	Low Value	Dormat
4	24	23	151	510263.06	1	2	2	1-2-2	1.67	Low Value	Churned
5	1	9	152	486023.73	2	2	1	2-2-1	1.67	Low Value	Dormat
6	8	5	142	477942.56	3	1	1	3-1-1	1.67	Low Value	Dormat
7	26	11	153	512610.07	1	2	2	1-2-2	1.67	Low Value	Churned
8	28	8	145	522078.74	2	1	2	2-1-2	1.67	Low Value	Dormat
9	37	11	152	528442.63	1	2	2	1-2-2	1.67	Low Value	Churned
10	27	3	144	441208.17	4	1	1	4-1-1	2.00	Low Value	New Customer

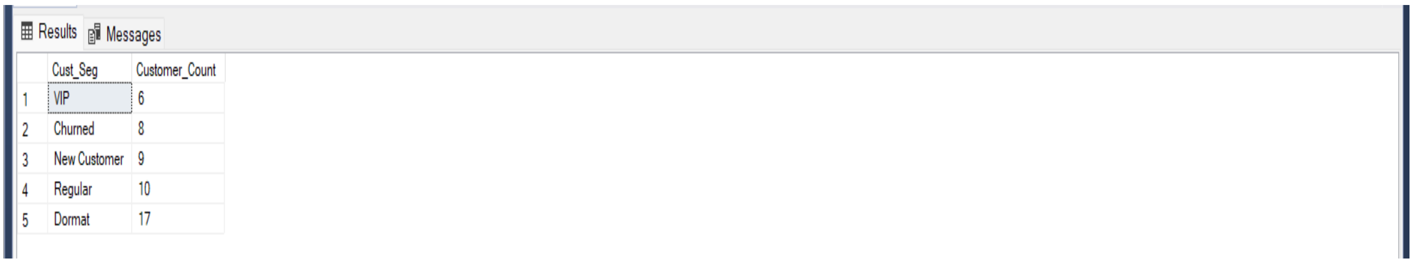
```sql
----Insights

--Distribution of Customers by Value Segment
SELECT
       Value_Seg,
       COUNT(CustomerID) AS Customer_Count
FROM Customer_Segmentaion
GROUP BY Value_Seg
ORDER BY Customer_Count
```



	Value_Seg	Customer_Count
1	High Value	11
2	Low Value	18
3	Mid Value	21

```
--==> We have highest Mid Value Customers (42%)
```

```
--Distribution of Customers by Customer Segment
SELECT
       Cust_Seg,
       COUNT(CustomerID) AS Customer_Count
FROM Customer_Segmentaion
GROUP BY Cust_Seg
ORDER BY Customer_Count
```

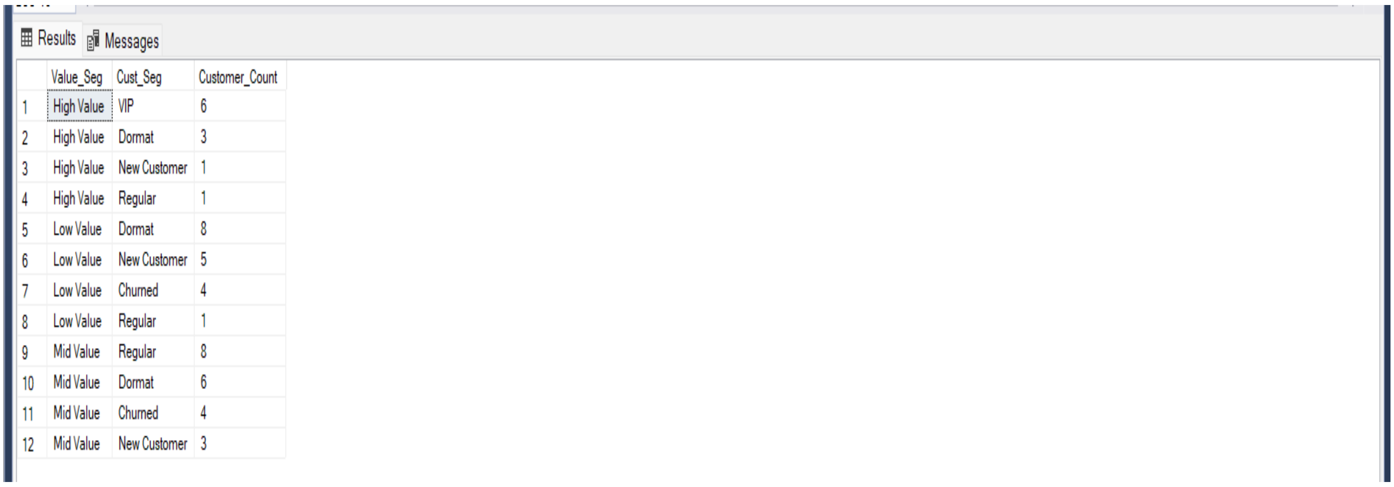


Results | Messages

	Cust_Seg	Customer_Count
1	VIP	6
2	Churned	8
3	New Customer	9
4	Regular	10
5	Dormat	17

```
--==>Company have highest Dormat Customers (34%), 20% Regular Customers, 18% New Customers, 16%
Churned Customers & Lowest VIP Customers (12%)
```

```
--Distribution of customers across different RFM customer segments within each value segment
SELECT
       Value_Seg,
       Cust_Seg,
       COUNT(CustomerID) AS Customer_Count
FROM Customer_Segmentaion
GROUP BY Cust_Seg,Value_Seg
ORDER BY Value_Seg,Customer_Count DESC
```



Results | Messages

	Value_Seg	Cust_Seg	Customer_Count
1	High Value	VIP	6
2	High Value	Dormat	3
3	High Value	New Customer	1
4	High Value	Regular	1
5	Low Value	Dormat	8
6	Low Value	New Customer	5
7	Low Value	Churned	4
8	Low Value	Regular	1
9	Mid Value	Regular	8
10	Mid Value	Dormat	6
11	Mid Value	Churned	4
12	Mid Value	New Customer	3

```
--==>Churned Customers are equally distributed among mid value & low value customers.
--==>Dormant Customers are distributed across all the value segments, low value segment have the
maximum dormant customers.
--==>Regular Customers are also distributed across all the value segments but majorly the Mid Value
segment.
--==>New Customers are also distributed across all the value segments but majorly low value & mid
value segment.
--==>55% of High Value segment customers are the VIP Customer
```