Business Case: Target – SQL – PRADEEP D

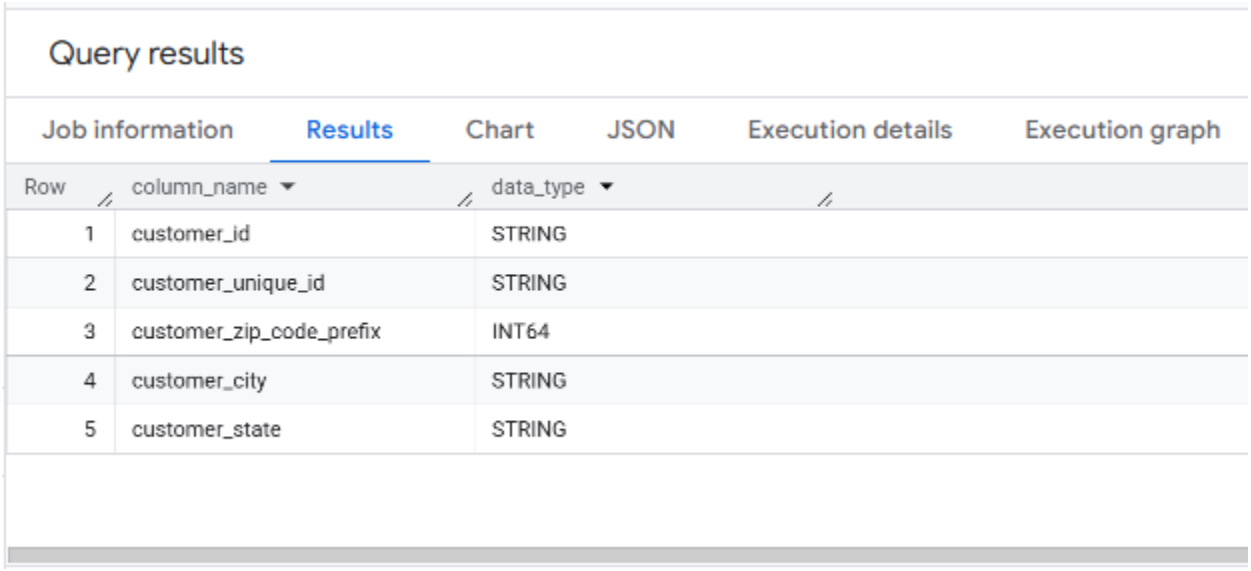
Problem Statement:

Target, a leading global retailer, operates extensively in Brazil. The provided dataset contains detailed information on 100,000 orders placed between 2016 and 2018, covering order status, pricing, payments, freight performance, customer locations, product attributes, and customer reviews. The goal is to analyze this dataset to uncover key insights into Target's Brazilian operations and provide actionable recommendations for improving operational efficiency, customer satisfaction, and overall business performance.

I. Importing the dataset and doing usual exploratory analysis steps like checking the structure & characteristics of the dataset:

1) Data type of all columns in the "customers" table.

```
SELECT column_name, data_type
FROM `scaler-target-csdy.dataset.INFORMATION_SCHEMA.COLUMNS`
WHERE table_name = 'customers';
```



Query results

Job information | Results | Chart | JSON | Execution details | Execution graph

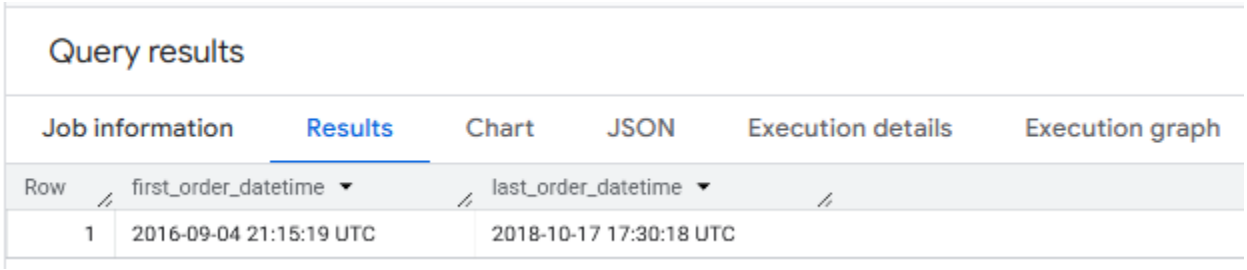
Row	column_name	data_type
1	customer_id	STRING
2	customer_unique_id	STRING
3	customer_zip_code_prefix	INT64
4	customer_city	STRING
5	customer_state	STRING

Insights/Recommendations:

All the columns present in customers table is of string data type and only the column customer_zip_code_prefix is of integer data type.

2) Getting the time range between which the orders were placed.

```
SELECT MIN(order_purchase_timestamp) AS first_order_datetime,
MAX(order_purchase_timestamp) AS last_order_datetime
FROM `scaler-target-csdy.dataset.orders`;
```



Insights/Recommendations:

In the dataset the first order was placed on 2016-09-04 around 21:15:19 UTC and the last order was placed on 2018-10-17 around 17:30:18 UTC.

3) Count the Cities & States of customers who ordered during the given period.

```
WITH date_range AS (SELECT
MIN(order_purchase_timestamp) AS min_date,
MAX(order_purchase_timestamp) AS max_date
FROM `scaler-target-csdy.dataset.orders`)

SELECT
COUNT(DISTINCT c.customer_city) AS count_of_customer_city,
COUNT(DISTINCT c.customer_state) AS count_of_customer_state
FROM
`scaler-target-csdy.dataset.customers` c
JOIN
`scaler-target-csdy.dataset.orders` o
ON
c.customer_id = o.customer_id
JOIN
```

```
date_range dr
ON
o.order_purchase_timestamp BETWEEN dr.min_date AND dr.max_date;
```

Insights/Recommendations:

i) The Distinct count of cities of customers in Brazil who ordered during the given period is 4119.

ii) The Distinct count of states of customers in Brazil who ordered during the given period is 27.

II. In-depth Exploration:

1) Is there a growing trend in the no. of orders placed over the past years?

```
SELECT EXTRACT(YEAR FROM o.order_purchase_timestamp) AS
order_year,
count(o.order_id) AS count_of_orders
FROM
`scaler-target-csdy.dataset.customers` c
JOIN
 `scaler-target-csdy.dataset.orders` o
ON c.customer_id = o.customer_id
GROUP BY 1
ORDER BY order_year;
```

Query results

Job information | Results | Chart | JSON | Execution details | Execution graph

Row	order_year	count_of_orders
1	2016	329
2	2017	45101
3	2018	54011

Insights/Recommendations:

i) There is an increase in the count of orders placed over the period of years.
ii) There is a huge increase in the count of orders placed in 2017 i.e 45101 compared to the orders placed in 2016 i.e 329.
iii) There is a slight increase in the count of orders placed in 2018 i.e 54011 compared to the orders placed in 2017 i.e 45101.

2) Can we see some kind of monthly seasonality in terms of the no. of orders being placed?

Based on Years & Months:

```
SELECT EXTRACT(YEAR FROM o.order_purchase_timestamp) AS
order_year,
EXTRACT(MONTH FROM o.order_purchase_timestamp) AS order_month,
count(o.order_id) AS count_of_orders
FROM
`scaler-target-csdy.dataset.customers` c
JOIN
`scaler-target-csdy.dataset.orders` o
ON c.customer_id = o.customer_id
GROUP BY order_year,order_month
ORDER BY order_year,order_month;
```

Query results

Save results

Job information | Results | Chart | JSON | Execution details | Execution graph

Row	order_year	order_month	count_of_orders
1	2016	9	4
2	2016	10	324
3	2016	12	1
4	2017	1	800
5	2017	2	1780
6	2017	3	2682
7	2017	4	2404
8	2017	5	3700
9	2017	6	3245

Results per page: 200 1 – 25 of 25

Based on Months of all Years:

```
SELECT
EXTRACT(MONTH FROM o.order_purchase_timestamp) AS order_month,
count(o.order_id) AS count_of_orders
FROM
`scaler-target-csdy.dataset.customers` c
JOIN
`scaler-target-csdy.dataset.orders` o
ON c.customer_id = o.customer_id
GROUP BY order_month
ORDER BY order_month;
```

Query results

Save results

Job information | Results | Visualisation | JSON | Execution details | Execution graph

Row	order_month	count_of_orders
1	1	8069
2	2	8508
3	3	9893
4	4	9343
5	5	10573
6	6	9412
7	7	10318
8	8	10843
9	9	4305
10	10	4959
11	11	7544
12	12	5674

Results per page: 50 1 – 12 of 12

Insights/Recommendations:

There is no particular monthly seasonality trend in terms of no of orders being placed since the month by month per year data doesn't show consistent peaks in the same months each year. However across all years in Aug month the maximum count of orders were placed (i.e 10843) compared to other months.

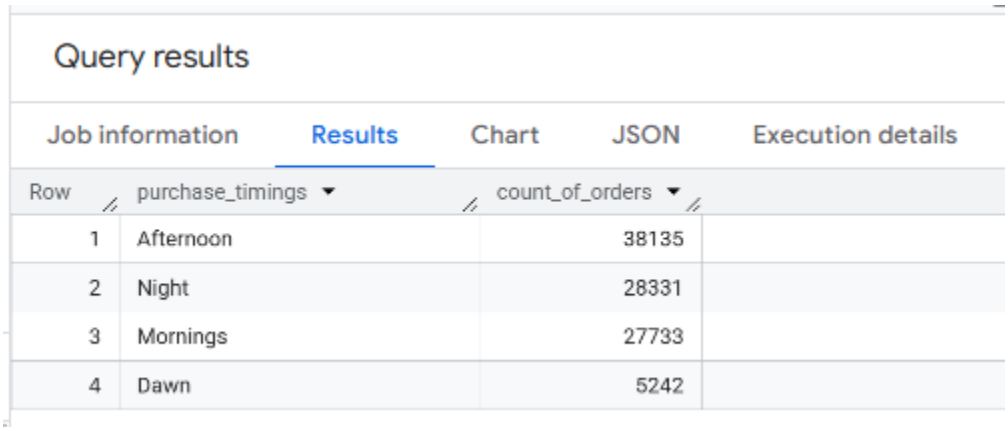
3) During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)

a. 0-6 hrs : Dawn
b. 7-12 hrs : Mornings
c. 13-18 hrs : Afternoon
d. 19-23 hrs : Night

```
WITH cte AS (SELECT *, EXTRACT(TIME FROM
order_purchase_timestamp) AS od_time_stamp FROM `scaler-target-
csdy.dataset.orders`),

ts AS (SELECT od_time_stamp,
        CASE WHEN od_time_stamp BETWEEN TIME '00:00:00' AND TIME
   '06:59:59' THEN 'Dawn'
              WHEN od_time_stamp BETWEEN TIME '07:00:00' AND TIME
   '12:59:59' THEN 'Mornings'
              WHEN od_time_stamp BETWEEN TIME '13:00:00' AND TIME
   '18:59:59' THEN 'Afternoon'
              WHEN od_time_stamp BETWEEN TIME '19:00:00' AND TIME
   '23:59:59' THEN 'Night'
        END AS purchase_timings
FROM cte)

SELECT purchase_timings, count(*) AS count_of_orders
FROM ts
GROUP BY purchase_timings
ORDER BY count_of_orders DESC;
```



Query results

Job information | Results | Chart | JSON | Execution details

Row	purchase_timings	count_of_orders
1	Afternoon	38135
2	Night	28331
3	Mornings	27733
4	Dawn	5242

Insights/Recommendations:

i) Brazilian customers place the highest number of orders in the afternoon, while the dawn period records the fewest orders.

ii) Order distribution in the dataset is as follows:

- Afternoon: 38,135 orders
- Night: 28,331 orders
- Morning: 27,733 orders
- Dawn: 5,242 orders

iii) Since the majority of orders are placed in the afternoon, the platform should ensure high server availability and performance during this period. Any planned maintenance should be scheduled during low traffic periods such as dawn to avoid impacting customers.

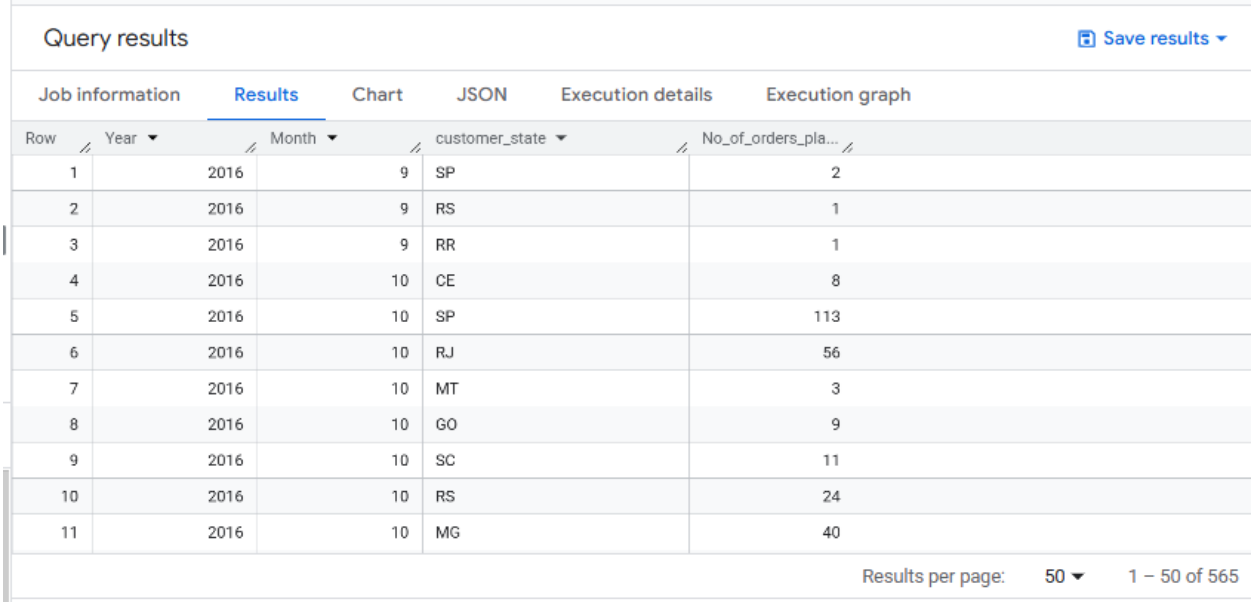
III. Evolution of E-commerce orders in the Brazil region:

1) Get the month-on-month no. of orders placed in each state.

Based on Years & Months:

```
SELECT
EXTRACT(YEAR FROM o.order_purchase_timestamp) AS Year,
EXTRACT(MONTH FROM o.order_purchase_timestamp) AS Month,
c.customer_state, COUNT(o.order_id) AS No_of_orders_placed
FROM
`scaler-target-csdy.dataset.customers` c
JOIN
`scaler-target-csdy.dataset.orders` o
```

```
ON c.customer_id = o.customer_id
GROUP BY Year, Month, c.customer_state
ORDER BY Year, Month;
```



Query results

Save results

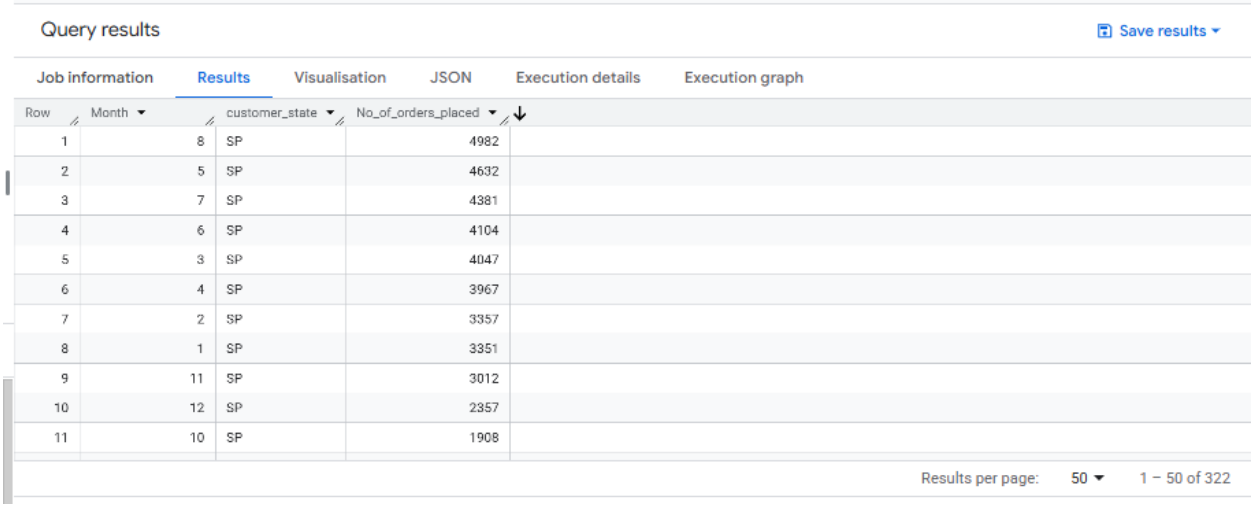
Job information | Results | Chart | JSON | Execution details | Execution graph

Row	Year	Month	customer_state	No_of_orders_pla...
1	2016	9	SP	2
2	2016	9	RS	1
3	2016	9	RR	1
4	2016	10	CE	8
5	2016	10	SP	113
6	2016	10	RJ	56
7	2016	10	MT	3
8	2016	10	GO	9
9	2016	10	SC	11
10	2016	10	RS	24
11	2016	10	MG	40

Results per page: 50 1 – 50 of 565

Based on Months of all Years:

```
SELECT
EXTRACT(MONTH FROM o.order_purchase_timestamp) AS Month,
c.customer_state,
COUNT(o.order_id) AS No_of_orders_placed
FROM
`scaler-target-csdy.dataset.customers` c
JOIN
`scaler-target-csdy.dataset.orders` o
ON c.customer_id = o.customer_id
GROUP BY Month, c.customer_state
ORDER BY Month;
```



Query results

Save results

Job information | Results | Visualisation | JSON | Execution details | Execution graph

Row	Month	customer_state	No_of_orders_placed
1	8	SP	4982
2	5	SP	4632
3	7	SP	4381
4	6	SP	4104
5	3	SP	4047
6	4	SP	3967
7	2	SP	3357
8	1	SP	3351
9	11	SP	3012
10	12	SP	2357
11	10	SP	1908

Results per page: 50 1 – 50 of 322

Insights/Recommendations:

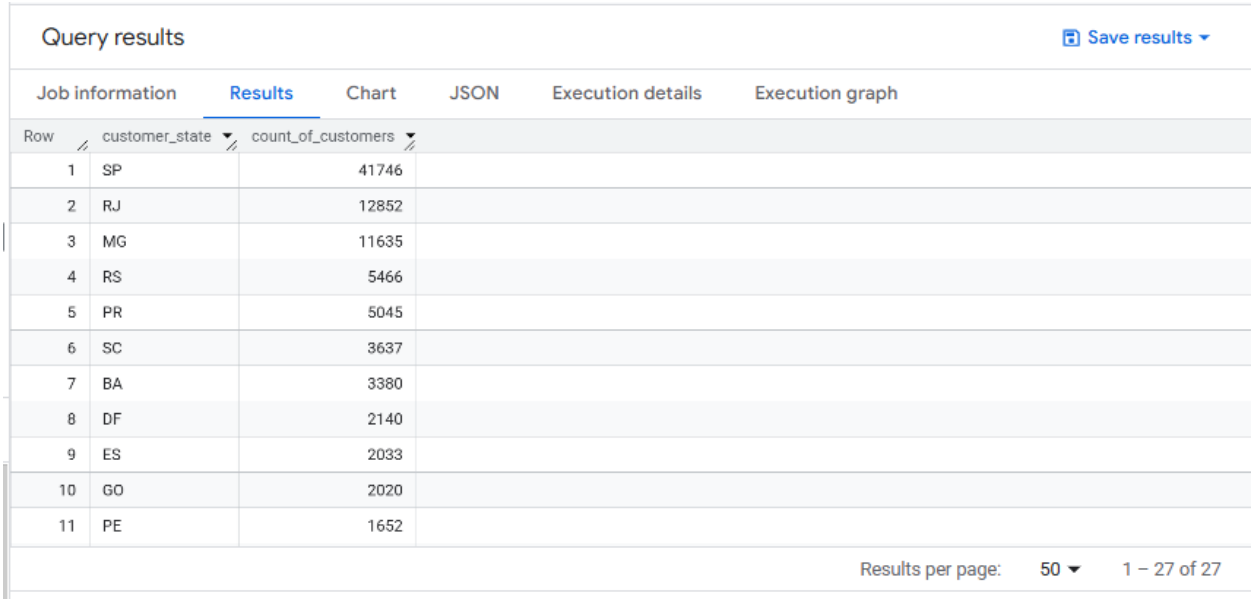
i) SP state dominates demand with the peak of 3,253 orders in August 2018 and SP state placed a total of 4982 orders in Aug month in all years making it the most critical market for the business.

ii) 2017 and 2018 recorded the highest overall orders, with SP state consistently leading with the no of orders placed.

iii) Several states like (AM, AP, MS, PB, PI, PR, RJ, RR, RS, SC, TO) recorded only 1 order in scattered months like June, July, January, October, December, September, May of the Years 2016, 2017 and 2018 respectively indicating low market penetration and significant growth potential.

2) How are the customers distributed across all the states?

```
SELECT customer_state, COUNT(*) AS count_of_customers
FROM `scaler-target-csdy.dataset.customers`
GROUP BY customer_state
ORDER BY count_of_customers DESC;
```



Row	customer_state	count_of_customers
1	SP	41746
2	RJ	12852
3	MG	11635
4	RS	5466
5	PR	5045
6	SC	3637
7	BA	3380
8	DF	2140
9	ES	2033
10	GO	2020
11	PE	1652

Insights/Recommendations:

i) Most customers are in SP state, with 41,746 customers. This shows that Target has a very strong customer base here.

ii) RR state has the fewest customers, with only 46 customers, which is very low compared to SP.

iii) Some other states also have very few customers, such as AC (81) and AP (68). Target can run special offers or campaigns to attract more customers in these regions.

IV. Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight and others.

1) Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only).

```
WITH value AS (SELECT *
FROM scaler-target-csdy.dataset.customers c
JOIN scaler-target-csdy.dataset.orders o
ON c.customer_id = o.customer_id
JOIN scaler-target-csdy.dataset.order_items oi
ON o.order_id = oi.order_id
```

```
WHERE EXTRACT(MONTH FROM order_purchase_timestamp) <= 08 AND
EXTRACT(YEAR FROM order_purchase_timestamp) IN (2017, 2018)),

val AS (SELECT EXTRACT(YEAR FROM value.order_purchase_timestamp)
AS Year, ROUND(SUM(value.price),2) AS Total_Price,
FROM value
GROUP BY 1
ORDER BY 1)

SELECT *, ROUND((Total_Price - LAG(Total_Price,1) OVER(ORDER BY
Year))
    / LAG(Total_Price,1) OVER(ORDER BY Year) * 100, 2)  AS
Percent_increase_cost_order
FROM val
ORDER BY Year;
```

Query results

Job information | Results | Visualisation | JSON | Execution details | Execution graph

Row	Year	Total_Price	Percent_increase_cost_order
1	2017	3113000.32	null
2	2018	7385905.8	137.26

Insights/Recommendations:

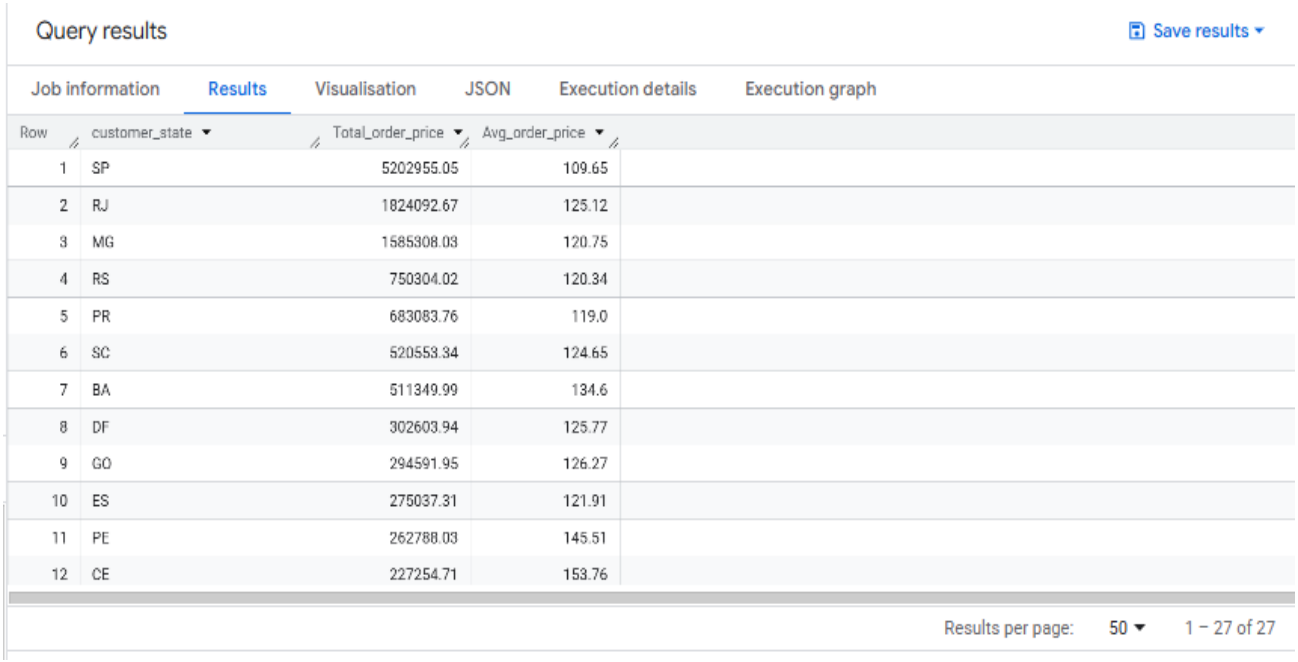
i) The total cost of orders between Jan and Aug in the year 2017 is 3113000.32 and the cost of orders between Jan and Aug in the year 2018 is 7385905.8

ii) There is 137.26% increase in the cost order in 2018 compared to the year 2017.

2) Calculate the Total & Average value of order price for each state.

```
SELECT DISTINCT c.customer_state,
```

```
SUM(oi.price) OVER(PARTITION BY c.customer_state) AS
Total_order_price,
ROUND(AVG(oi.price) OVER(PARTITION BY c.customer_state),2) AS
Avg_order_price
FROM scaler-target-csdy.dataset.customers c
JOIN scaler-target-csdy.dataset.orders o
ON c.customer_id = o.customer_id
JOIN scaler-target-csdy.dataset.order_items oi
ON o.order_id = oi.order_id
ORDER BY Total_order_price DESC;
```



Row	customer_state	Total_order_price	Avg_order_price
1	SP	5202955.05	109.65
2	RJ	1824092.67	125.12
3	MG	1585308.03	120.75
4	RS	750304.02	120.34
5	PR	683083.76	119.0
6	SC	520553.34	124.65
7	BA	511349.99	134.6
8	DF	302603.94	125.77
9	GO	294591.95	126.27
10	ES	275037.31	121.91
11	PE	262788.03	145.51
12	CE	227254.71	153.76

Insights/Recommendations:

i) Customers from SP state have the highest total order price of 5,202,955.05, however they have the lowest average order price of 109.65.

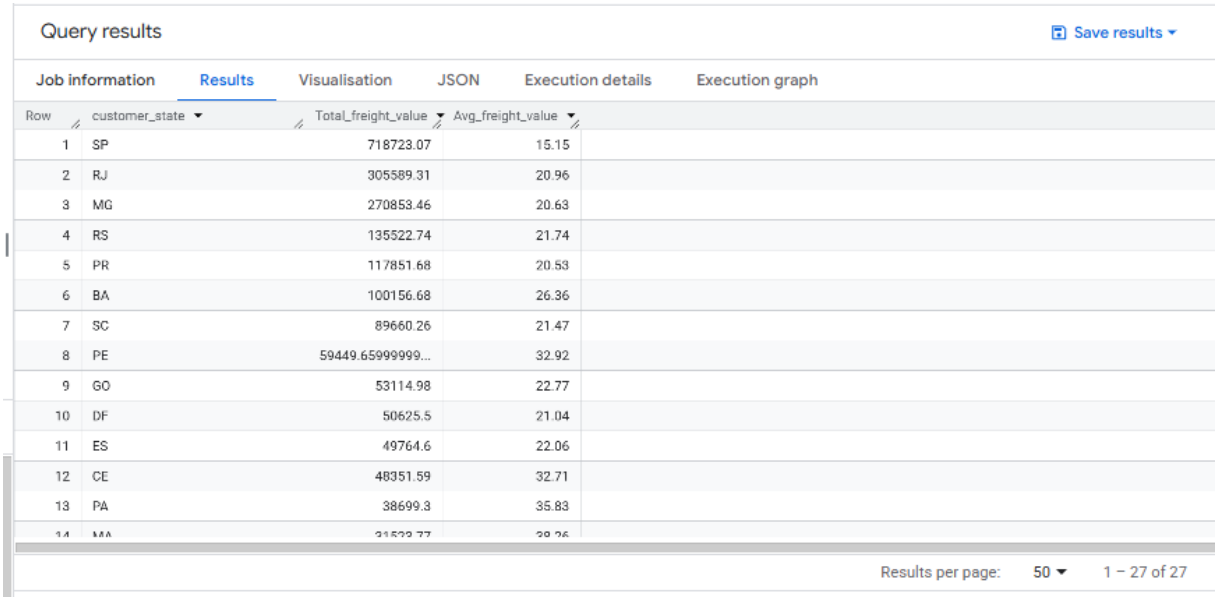
ii) Customers from PB state have the highest average order price of 191.48, however the total order price in the state is only 115,268.08.

iii) States like RJ (1,824,092.67) and MG (1,585,308.03) also contribute significantly to the total order price and can be considered the next major markets

after SP, while smaller states such as RR, AP, AC contribute very little overall and has a growth potential.

3) Calculate the Total & Average value of order freight for each state.

```
SELECT
DISTINCT c.customer_state,
SUM(oi.freight_value) OVER(PARTITION BY c.customer_state) AS
Total_freight_value,
ROUND(AVG(oi.freight_value) OVER(PARTITION BY
c.customer_state),2) AS Avg_freight_value
FROM scaler-target-csdy.dataset.customers c
JOIN scaler-target-csdy.dataset.orders o
ON c.customer_id = o.customer_id
JOIN scaler-target-csdy.dataset.order_items oi
ON o.order_id = oi.order_id
ORDER BY Total_freight_value DESC
```



Query results — Save results

Job information | Results | Visualisation | JSON | Execution details | Execution graph

Row	customer_state	Total_freight_value	Avg_freight_value
1	SP	718723.07	15.15
2	RJ	305589.31	20.96
3	MG	270853.46	20.63
4	RS	135522.74	21.74
5	PR	117851.68	20.53
6	BA	100156.68	26.36
7	SC	89660.26	21.47
8	PE	59449.65999999...	32.92
9	GO	53114.98	22.77
10	DF	50625.5	21.04
11	ES	49764.6	22.06
12	CE	48351.59	32.71
13	PA	38699.3	35.83
14	MA	31523.77	38.26

Results per page: 50 — 1 – 27 of 27

Insights/Recommendations:

i) Customers from SP state have the highest total freight value of 718,723.07, however they have the lowest average freight value of 15.15.

ii) Customers from RR state have the highest average freight value of 42.98, however the total freight value in the state is only 2,235.19.

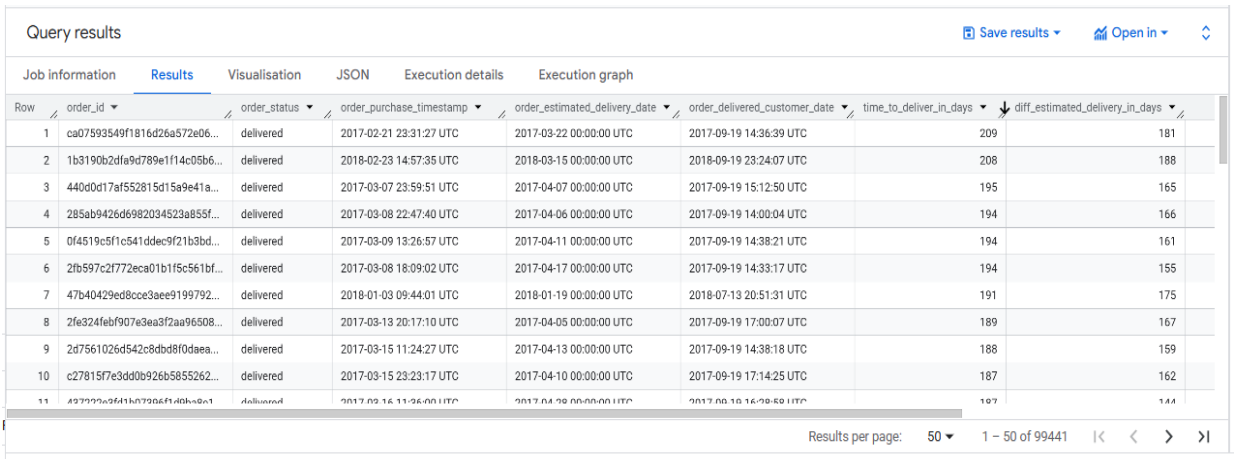
iii) States like RJ (305,589.31) and MG (270,853.46) also contribute significantly to the total freight value and can be considered as next major markets after SP, while smaller states such as RR, AP, and AC contribute very little overall but have comparatively high average freight costs.

iv) Should Focus on optimizing logistics with high cost especially in low-volume states (e.g., RR, AC, AP) to reduce freight burden, while continuing to strengthen sales in high-volume states like SP, RJ, and MG.

V. Analysis based on sales, freight and delivery time.

1) Find the no. of days taken to deliver each order from the order's purchase date as delivery time. Also, calculate the difference (in days) between the estimated & actual delivery date of an order.

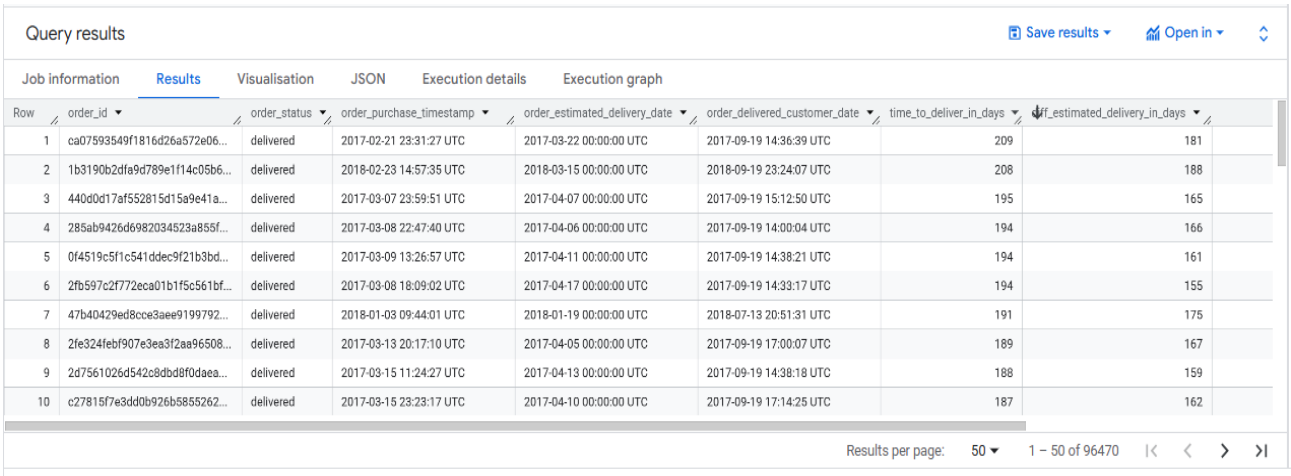
```
SELECT
order_id,
order_status,
order_purchase_timestamp,
order_estimated_delivery_date,
order_delivered_customer_date,
DATE_DIFF(order_delivered_customer_date,
order_purchase_timestamp, DAY) AS time_to_deliver_in_days,
DATE_DIFF(order_delivered_customer_date,
order_estimated_delivery_date, DAY) AS
diff_estimated_delivery_in_days
FROM scaler-target-csdy.dataset.orders;
```



Row	order_id	order_status	order_purchase_timestamp	order_estimated_delivery_date	order_delivered_customer_date	time_to_deliver_in_days	diff_estimated_delivery_in_days
1	ca07593549f1816d26a572e06...	delivered	2017-02-21 23:31:27 UTC	2017-03-22 00:00:00 UTC	2017-09-19 14:36:39 UTC	209	181
2	1b3190b2dfa9d789e1f14c05b6...	delivered	2018-02-23 14:57:35 UTC	2018-03-15 00:00:00 UTC	2018-09-19 23:24:07 UTC	208	188
3	440d0d17af552815d15a9e41a...	delivered	2017-03-07 23:59:51 UTC	2017-04-07 00:00:00 UTC	2017-09-19 15:12:50 UTC	195	165
4	285ab9426d6982034523a855f...	delivered	2017-03-08 22:47:40 UTC	2017-04-06 00:00:00 UTC	2017-09-19 14:00:04 UTC	194	166
5	0f4519c5f1c541ddec9f21b3bd...	delivered	2017-03-09 13:26:57 UTC	2017-04-11 00:00:00 UTC	2017-09-19 14:38:21 UTC	194	161
6	2fb597c2f772eca01b1f5c561bf...	delivered	2017-03-08 18:09:02 UTC	2017-04-17 00:00:00 UTC	2017-09-19 14:33:17 UTC	194	155
7	47b40429ed8cce3aee9199792...	delivered	2018-01-03 09:44:01 UTC	2018-01-19 00:00:00 UTC	2018-07-13 20:51:31 UTC	191	175
8	2fe324febf907e3ea3f2aa96508...	delivered	2017-03-13 20:17:10 UTC	2017-04-05 00:00:00 UTC	2017-09-19 17:00:07 UTC	189	167
9	2d7561026d542c8dbd8f0daea...	delivered	2017-03-15 11:24:27 UTC	2017-04-13 00:00:00 UTC	2017-09-19 14:38:18 UTC	188	159
10	c27815f7e3dd0b926b5855262...	delivered	2017-03-15 23:23:17 UTC	2017-04-10 00:00:00 UTC	2017-09-19 17:14:25 UTC	187	162

Filtering the dataset to include only orders with status ‘delivered’ and order_delivered_customer_date ‘IS NOT NULL’

```
SELECT
order_id,
order_status,
order_purchase_timestamp,
order_estimated_delivery_date,
order_delivered_customer_date,
DATE_DIFF(order_delivered_customer_date,
order_purchase_timestamp, DAY) AS time_to_deliver_in_days,
DATE_DIFF(order_delivered_customer_date,
order_estimated_delivery_date, DAY) AS
diff_estimated_delivery_in_days
FROM scaler-target-csdy.dataset.orders
WHERE order_status = 'delivered'
AND order_delivered_customer_date IS NOT NULL;
```



Row	order_id	order_status	order_purchase_timestamp	order_estimated_delivery_date	order_delivered_customer_date	time_to_deliver_in_days	diff_estimated_delivery_in_days
1	ca07593549f1816d26a572e06...	delivered	2017-02-21 23:31:27 UTC	2017-03-22 00:00:00 UTC	2017-09-19 14:36:39 UTC	209	181
2	1b3190b2dfa9d789e1f14c05b6...	delivered	2018-02-23 14:57:35 UTC	2018-03-15 00:00:00 UTC	2018-09-19 23:24:07 UTC	208	188
3	440d0d17af552815d15a9e41a...	delivered	2017-03-07 23:59:51 UTC	2017-04-07 00:00:00 UTC	2017-09-19 15:12:50 UTC	195	165
4	285ab9426d6982034523a855f...	delivered	2017-03-08 22:47:40 UTC	2017-04-06 00:00:00 UTC	2017-09-19 14:00:04 UTC	194	166
5	0f4519c5f1c541ddec9f21b3bd...	delivered	2017-03-09 13:26:57 UTC	2017-04-11 00:00:00 UTC	2017-09-19 14:38:21 UTC	194	161
6	2fb597c2f772eca01b1f5c561bf...	delivered	2017-03-08 18:09:02 UTC	2017-04-17 00:00:00 UTC	2017-09-19 14:33:17 UTC	194	155
7	47b40429ed8cce3aee9199792...	delivered	2018-01-03 09:44:01 UTC	2018-01-19 00:00:00 UTC	2018-07-13 20:51:31 UTC	191	175
8	2fe324febf907e3ea3f2aa96508...	delivered	2017-03-13 20:17:10 UTC	2017-04-05 00:00:00 UTC	2017-09-19 17:00:07 UTC	189	167
9	2d7561026d542c8dbd8f0daea...	delivered	2017-03-15 11:24:27 UTC	2017-04-13 00:00:00 UTC	2017-09-19 14:38:18 UTC	188	159
10	c27815f7e3dd0b926b5855262...	delivered	2017-03-15 23:23:17 UTC	2017-04-10 00:00:00 UTC	2017-09-19 17:14:25 UTC	187	162

Insights/Recommendations:

i) The order with order_id = ca07593549f1816d26a572e06dc1eab6 took the longest delivery time of 209 days from the purchase date. It was delivered 181 days later than the estimated delivery date.

ii) Similarly, the order with order_id = 1b3190b2dfa9d789e1f14c05b647a14a was delivered in 208 days from purchase, which is 188 days beyond the estimated delivery date.

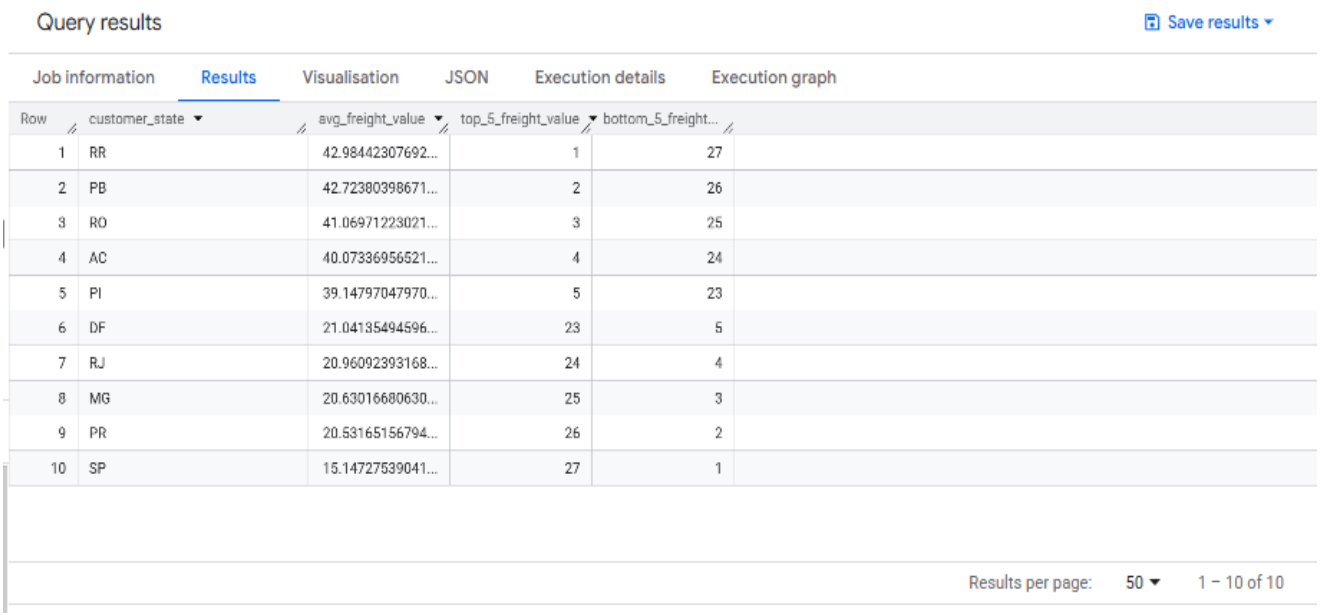
iii) On the other hand, the order with order_id = 0607f0efea4b566f1eb8f7d3c2397320 was delivered in just 3 days from purchase, which is 146 days earlier than the estimated delivery date.

iv) There are cases where orders are either delayed by over 180 days from the estimated delivery date or delivered much earlier than expected delivery date. This inconsistency highlights the need for improving logistics planning and accurate delivery time estimation.

2) Find out the top 5 states with the highest & lowest average freight value.

```
WITH value AS (
SELECT
c.customer_state,
AVG(oi.freight_value) AS avg_freight_value
FROM scaler-target-csdy.dataset.customers c
JOIN scaler-target-csdy.dataset.orders o
ON c.customer_id = o.customer_id
JOIN scaler-target-csdy.dataset.order_items oi
ON o.order_id = oi.order_id
GROUP BY c.customer_state
),
ranked AS (
SELECT *,
RANK() OVER (ORDER BY avg_freight_value DESC) AS rank_desc,
RANK() OVER (ORDER BY avg_freight_value ASC) AS rank_asc
FROM value
```

```
)
SELECT customer_state, avg_freight_value, rank_desc AS
top_5_freight_value, rank_asc AS bottom_5_freight_value
FROM ranked
WHERE rank_desc <= 5 OR rank_asc <= 5
ORDER BY avg_freight_value DESC;
```



Query results

Save results

Job information | Results | Visualisation | JSON | Execution details | Execution graph

Row	customer_state	avg_freight_value	top_5_freight_value	bottom_5_freight...
1	RR	42.98442307692...	1	27
2	PB	42.72380398671...	2	26
3	RO	41.06971223021...	3	25
4	AC	40.07336956521...	4	24
5	PI	39.14797047970...	5	23
6	DF	21.04135494596...	23	5
7	RJ	20.96092393168...	24	4
8	MG	20.63016680630...	25	3
9	PR	20.53165156794...	26	2
10	SP	15.14727539041...	27	1

Results per page: 50 1 – 10 of 10

Insights/Recommendations:

i) The customer states RR, PB, RO, AC, and PI contribute to the top 5 highest average freight values.

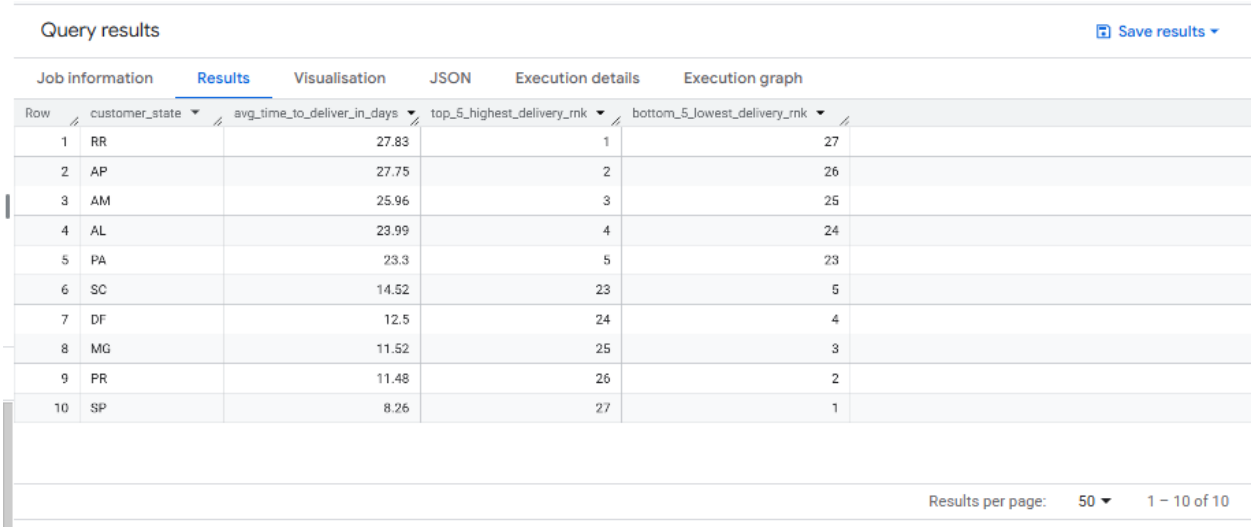
ii) The customer states SP, PR, MG, RJ, and DF contribute to the bottom 5 lowest average freight values.

iii) The state RR has the highest average freight value of 42.98.

iv) The state SP has the lowest average freight value of 15.14.

3) Find out the top 5 states with the highest & lowest average delivery time.

```
WITH value AS (
SELECT
c.customer_state,
ROUND(AVG(DATE_DIFF(o.order_delivered_customer_date,
o.order_purchase_timestamp, DAY)),2) AS
avg_time_to_deliver_in_days,
FROM scaler-target-csdy.dataset.customers c
JOIN scaler-target-csdy.dataset.orders o
ON c.customer_id = o.customer_id
JOIN scaler-target-csdy.dataset.order_items oi
ON o.order_id = oi.order_id
GROUP BY c.customer_state
),
ranked AS (
SELECT *,
RANK() OVER (ORDER BY avg_time_to_deliver_in_days DESC) AS
rank_desc,
RANK() OVER (ORDER BY avg_time_to_deliver_in_days ASC) AS
rank_asc
FROM value
)
SELECT customer_state, avg_time_to_deliver_in_days, rank_desc AS
top_5_highest_delivery_rnk, rank_asc AS
bottom_5_lowest_delivery_rnk
FROM ranked
WHERE rank_desc <= 5 OR rank_asc <= 5
ORDER BY avg_time_to_deliver_in_days DESC;
```



Query results

Save results

Job information | Results | Visualisation | JSON | Execution details | Execution graph

Row	customer_state	avg_time_to_deliver_in_days	top_5_highest_delivery_rnk	bottom_5_lowest_delivery_rnk
1	RR	27.83	1	27
2	AP	27.75	2	26
3	AM	25.96	3	25
4	AL	23.99	4	24
5	PA	23.3	5	23
6	SC	14.52	23	5
7	DF	12.5	24	4
8	MG	11.52	25	3
9	PR	11.48	26	2
10	SP	8.26	27	1

Results per page: 50 1 – 10 of 10

Insights/Recommendations:

i) The customer states RR, AP, AM, AL, PA have the highest average delivery times of 27.83, 27.75, 25.96, 23.99, and 23.3 days respectively.

ii) The customer states SP, PR, MG, DF, SC have the lowest average delivery times of 8.26, 11.48, 11.52, 12.5, and 14.52 days respectively.

iii) The state RR has the highest delivery delay with an average of 27.83 days.

iv) The state SP has the fastest deliveries, averaging just 8.26 days.

4) Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery.

```
WITH value AS (
SELECT
c.customer_state,
ROUND(AVG(DATE_DIFF(o.order_delivered_customer_date,
o.order_purchase_timestamp, DAY)),2) AS
avg_time_to_deliver_in_days,
ROUND(AVG(DATE_DIFF(o.order_estimated_delivery_date,
o.order_purchase_timestamp, DAY)),2) AS
estimated_avg_time_to_deliver_in_days,
```

```
FROM scaler-target-csdy.dataset.customers c
JOIN scaler-target-csdy.dataset.orders o
ON c.customer_id = o.customer_id
JOIN scaler-target-csdy.dataset.order_items oi
ON o.order_id = oi.order_id
GROUP BY c.customer_state
)

SELECT *, ROUND((avg_time_to_deliver_in_days -
estimated_avg_time_to_deliver_in_days),2) AS time_difference
FROM value
ORDER BY time_difference
LIMIT 5;
```

Query results

Save results

Job information | Results | Visualisation | JSON | Execution details | Execution graph

Row	customer_state	avg_time_to_deliver_in_days	estimated_avg_time_to_deliver_in_days	time_difference
1	AC	20.33	40.7	-20.37
2	RO	19.28	38.65	-19.37
3	AM	25.96	45.21	-19.25
4	RR	27.83	45.98	-18.15
5	AP	27.75	45.49	-17.74

Results per page: 50 1 – 5 of 5

Insights/Recommendations:

i) The top 5 states where deliveries were significantly faster than estimated are AC, RO, AM, RR, and AP. Their average delivery times were 20.33, 19.28, 25.96, 27.83, and 27.75 days, compared to estimated averages of 40.7, 38.65, 45.21, 45.98, and 45.49 days respectively.

ii) The state AC shows the highest improvement, with deliveries being 20.37 days faster than the estimated average.

iii) These states demonstrate strong logistics efficiency, best practices from these regions could be studied and replicated in the slower logistic performing states.

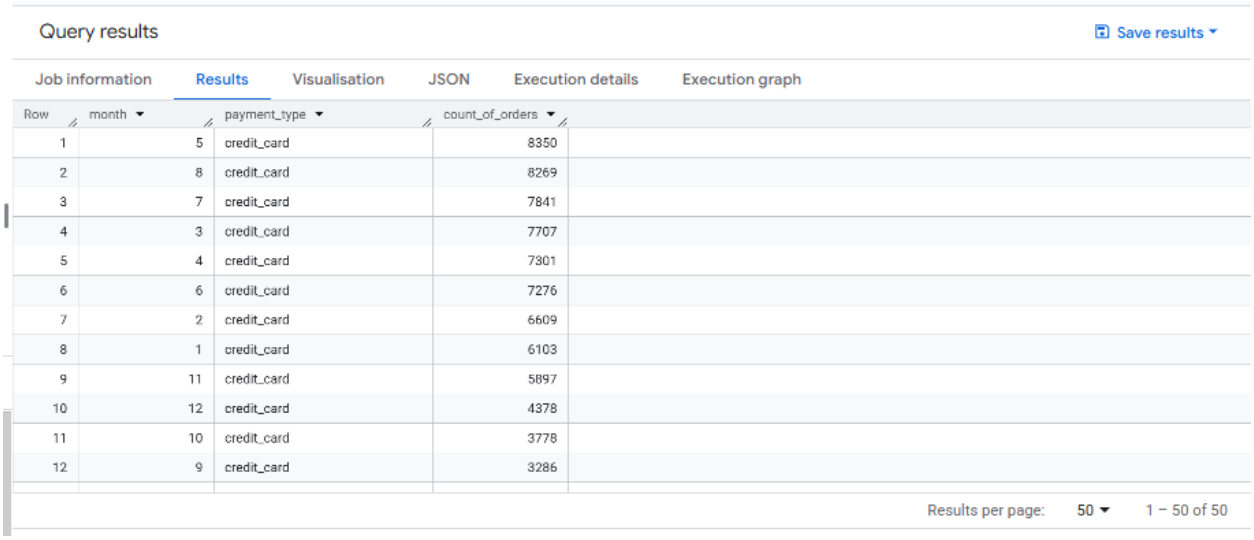
VI. Analysis based on the payments:

1) Find the month-on-month no. of orders placed using different payment types.

Based on only month (Seasonal insights):

```
WITH value AS (SELECT *, EXTRACT (MONTH FROM
o.order_purchase_timestamp) AS month
FROM scaler-target-csdy.dataset.orders o
JOIN scaler-target-csdy.dataset.payments p
ON o.order_id = p.order_id)

SELECT month, payment_type, count(payment_type) AS
count_of_orders
FROM value
GROUP BY month, payment_type
ORDER BY count_of_orders DESC;
```



Query results

Save results

Job information | Results | Visualisation | JSON | Execution details | Execution graph

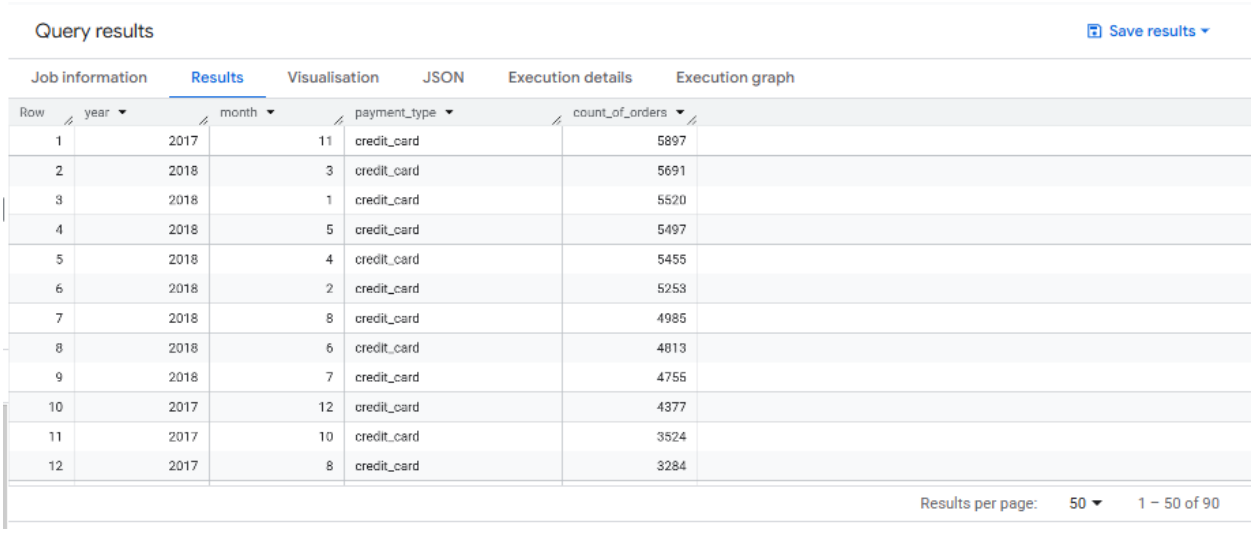
Row	month	payment_type	count_of_orders
1	5	credit_card	8350
2	8	credit_card	8269
3	7	credit_card	7841
4	3	credit_card	7707
5	4	credit_card	7301
6	6	credit_card	7276
7	2	credit_card	6609
8	1	credit_card	6103
9	11	credit_card	5897
10	12	credit_card	4378
11	10	credit_card	3778
12	9	credit_card	3286

Results per page: 50 1 – 50 of 50

Based on Year & month (Trend Analysis):

```
WITH value AS (SELECT *, EXTRACT (YEAR FROM
o.order_purchase_timestamp) AS year, EXTRACT (MONTH FROM
o.order_purchase_timestamp) AS month
FROM scaler-target-csdy.dataset.orders o
JOIN scaler-target-csdy.dataset.payments p
ON o.order_id = p.order_id)
```

```sql
SELECT year, month, payment_type, count(payment_type) AS
count_of_orders
FROM value
GROUP BY payment_type, year, month
ORDER BY count_of_orders DESC;
```



Query results

Save results

Job information | Results | Visualisation | JSON | Execution details | Execution graph

Row	year	month	payment_type	count_of_orders
1	2017	11	credit_card	5897
2	2018	3	credit_card	5691
3	2018	1	credit_card	5520
4	2018	5	credit_card	5497
5	2018	4	credit_card	5455
6	2018	2	credit_card	5253
7	2018	8	credit_card	4985
8	2018	6	credit_card	4813
9	2018	7	credit_card	4755
10	2017	12	credit_card	4377
11	2017	10	credit_card	3524
12	2017	8	credit_card	3284

Results per page: 50 1 – 50 of 90

Insights/Recommendations:

i) Credit card is the most preferred payment method compared to other payment modes.

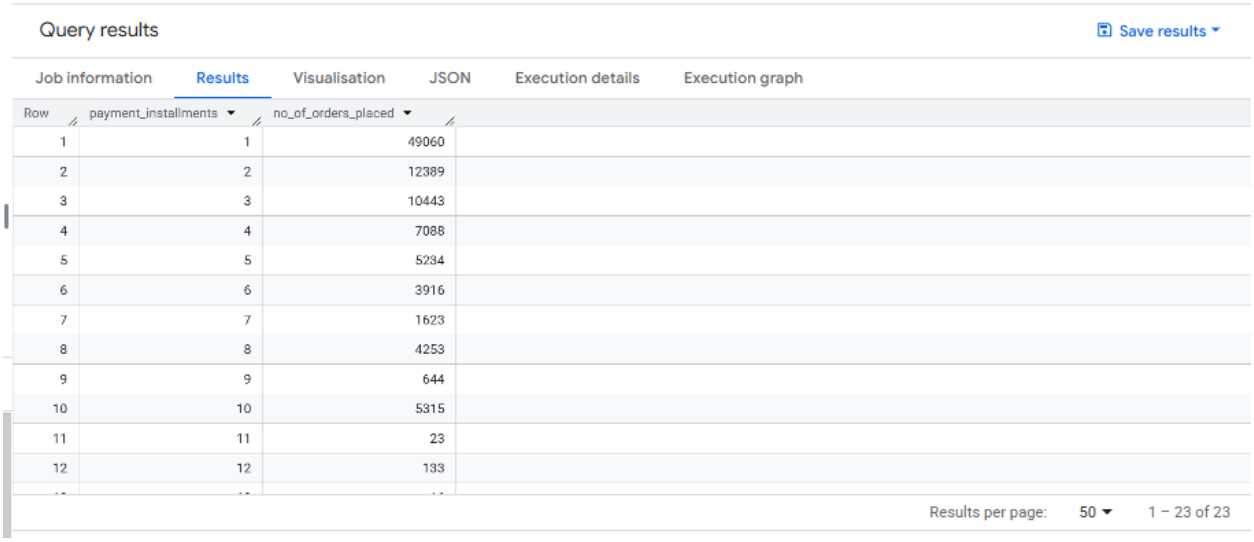
ii) In all Years May shows the highest number of orders 8350 using credit card payment type.

iii) In 2017 Nov maximum of 5897 orders were placed using the credit card payment type.

iv) Should focus on marketing campaigns and offers during May to leverage peak demand and strengthen the credit card payment experience.

2) Find the no. of orders placed on the basis of the payment installments that have been paid.

```sql
SELECT payment_installments,
COUNT(DISTINCT order_id) AS no_of_orders_placed
FROM scaler-target-csdy.dataset.payments
WHERE payment_installments >= 1
GROUP BY payment_installments
ORDER BY payment_installments;
```



Row	payment_installments	no_of_orders_placed
1	1	49060
2	2	12389
3	3	10443
4	4	7088
5	5	5234
6	6	3916
7	7	1623
8	8	4253
9	9	644
10	10	5315
11	11	23
12	12	133

Insights/Recommendations:

i) Majority of orders (49,060) were placed with a single installment, showing that most customers prefer one-time payments.

ii) Installment plans of 1, 2, or 3 are most commonly used, indicating limited adoption for longer EMI options.

iii) Very few customers opted for higher installments (i.e 22 or 23).

iv) Should focus on one-time payments while promoting short tenure EMIs (2-3 installments), and streamline very long installment options due to negligible usage.

Overall Insights:

- **Customer Distribution**: The majority of customers and orders come from SP state, making it the strongest market. States like RR, AC, AP contribute very little.

- **Order Trends:** Orders steadily increased from 2016 to 2018, with peak demand observed in May and November, and most purchases made in the afternoon & night.

- **Economy Impact:** From Jan to Aug 2017 to the same period in 2018, order costs increased by 137%, indicating strong growth in e commerce adoption.

- **Order Value & Freight**: SP leads in total order value and freight, but has lowest average freight & order price, showing economies of scale. Smaller states like RR have higher average freight costs and longer delivery times.

- **Delivery Performance:** Delivery times vary significantly. Some orders are delivered much earlier than expected (up to 146 days early), while others are delayed by over 180 days, showing inconsistencies in logistics and forecasting.

- **Payments:** Credit card is the dominant payment method, with the majority of customers choosing single installment payments. Very few opt for longer EMIs, but peak order counts via credit cards occur in May & November.

Key Recommendations:

- **Expand Beyond SP:** Strengthen Target's presence in low market states (RR, AC, AP) through localized campaigns, discounts, and partnerships with local logistics providers.

- **Logistics Optimization:** Reduce extreme delivery delays by improving delivery time estimation models, monitoring logistics SLAs, and leveraging faster shipping in remote states.

- **Seasonal Strategy:** Launch targeted marketing and promotional campaigns in May & November, aligning with peak demand periods.

- **Freight Cost Efficiency:** Focus on reducing freight costs in high-cost states (RR, PB, PI, etc) and better route optimization.

- **Payment Flexibility:** Since most customers use credit cards and 1 time installment payments, introduce small value EMI options to increase affordability and attract new customers.