Target SQL Business Case

Case study :

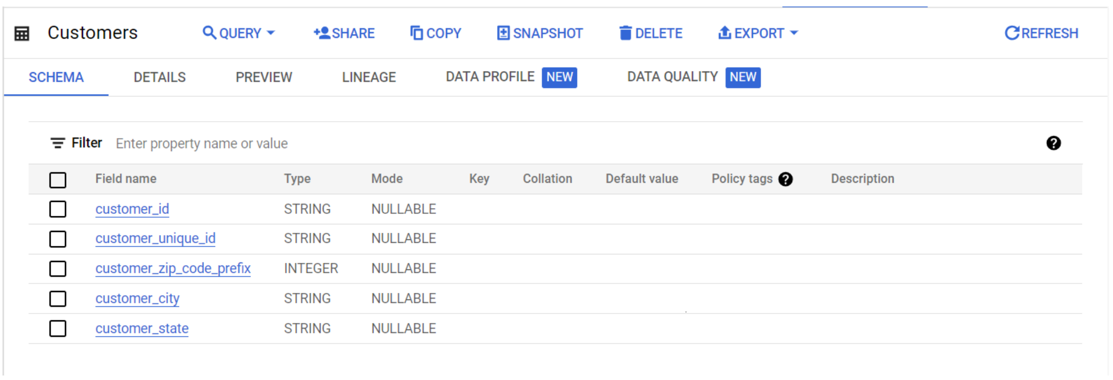
● **Target** is a globally renowned brand and a prominent retailer in the United States. Target makes itself a preferred shopping destination by offering outstanding value, inspiration, innovation and an exceptional guest experience that no other retailer can deliver.

● This particular business case focuses on the operations of Target in Brazil and provides insightful information about 100,000 orders placed between 2016 and 2018. The dataset offers a comprehensive view of various dimensions including the order status, price, payment and freight performance, customer location, product attributes, and customer reviews.

● By analyzing this extensive dataset, it becomes possible to gain valuable insights into Target's operations in Brazil. The information can shed light on various aspects of the business, such as order processing, pricing strategies, payment and shipping efficiency, customer demographics, product characteristics, and customer satisfaction levels.

Insights and recommendations based on the analysis of the target dataset :

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



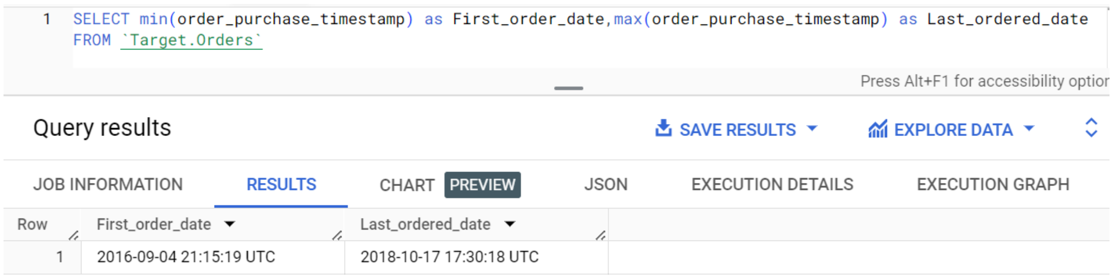
Field name	Type	Mode	Key	Collation	Default value	Policy tags	Description
customer_id	STRING	NULLABLE					
customer_unique_id	STRING	NULLABLE					
customer_zip_code_prefix	INTEGER	NULLABLE					
customer_city	STRING	NULLABLE					
customer_state	STRING	NULLABLE					

Insights : The customer table includes essential information such as customer_id, customer_unique_id which serves as a unique identifier for each customer. The customer_state field provides insights into the geographical location of customers, allowing for regional analysis, while the customer_city field further specifies their residential areas. This data can be leveraged to understand the distribution of customers, target specific regions for marketing efforts.

1B) Get the time range between which the orders were placed.

Query :

SELECT min(order_purchase_timestamp) as First_order_date,max(order_purchase_timestamp) as Last_ordered_date FROM `Target.Orders`



```
SELECT min(order_purchase_timestamp) as First_order_date,max(order_purchase_timestamp) as Last_ordered_date
FROM `Target.Orders`
```

Row	First_order_date	Last_ordered_date
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC

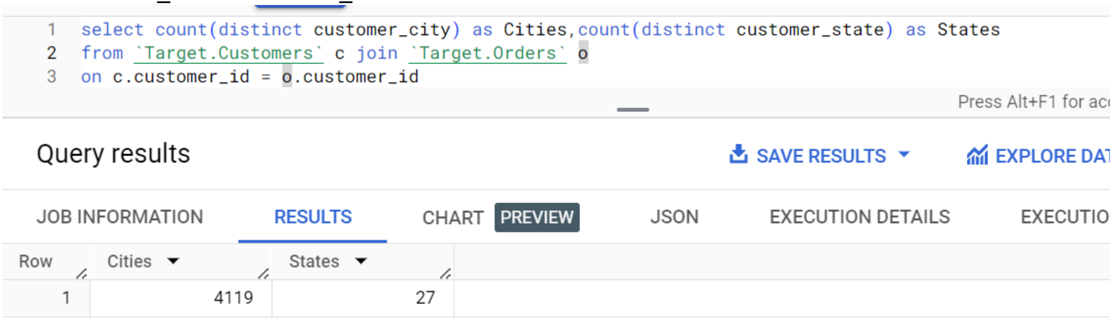
Insights : The orders were placed between 9th Sept 2016 and 17th October 2018.
First_order_date refers to the order date of the first order that was placed in this dataset and last_ordered_date refers to the order date of the latest or last order that was placed in this dataset.

Recommendations: NA

1C) Count the Cities & States of customers who ordered during the given period.

Query :

```
select count(distinct customer_city) as Cities,count(distinct customer_state) as States
from `Target.Customers` c join `Target.Orders` o
on c.customer_id = o.customer_id
```



```
select count(distinct customer_city) as Cities,count(distinct customer_state) as States
from `Target.Customers` c join `Target.Orders` o
on c.customer_id = o.customer_id
```

Press Alt+F1 for ac

Query results

Row	Cities	States
1	4119	27

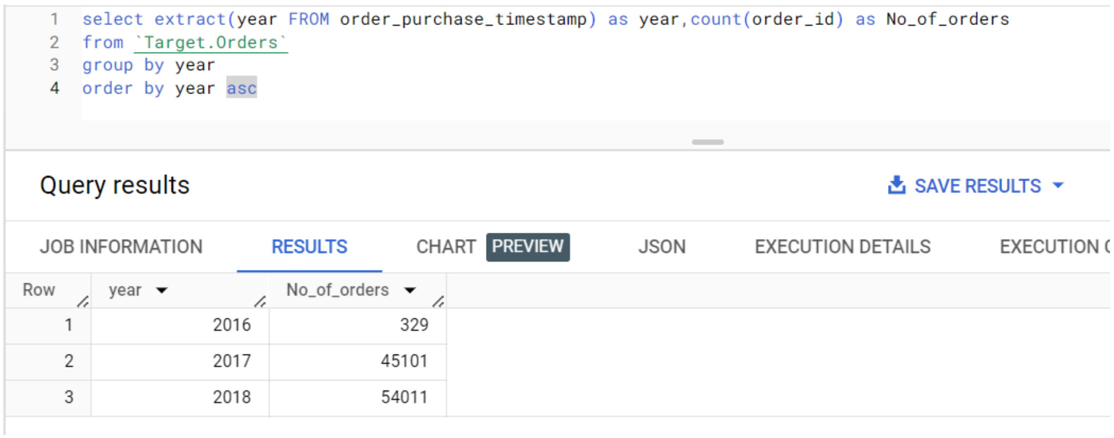
Insights : Customers across 4119 Cities and 27 States have ordered the products between 2016 and 2018.

Recommendation : NA

2A) Is there a growing trend in the no. of orders placed over the past years?

Query :

```
select extract(year FROM order_purchase_timestamp) as year,count(order_id) as No_of_orders
from `Target.Orders`
group by year
order by year asc
```



```
select extract(year FROM order_purchase_timestamp) as year,count(order_id) as No_of_orders
from `Target.Orders`
group by year
order by year asc
```

Query results

Row	year	No_of_orders
1	2016	329
2	2017	45101
3	2018	54011

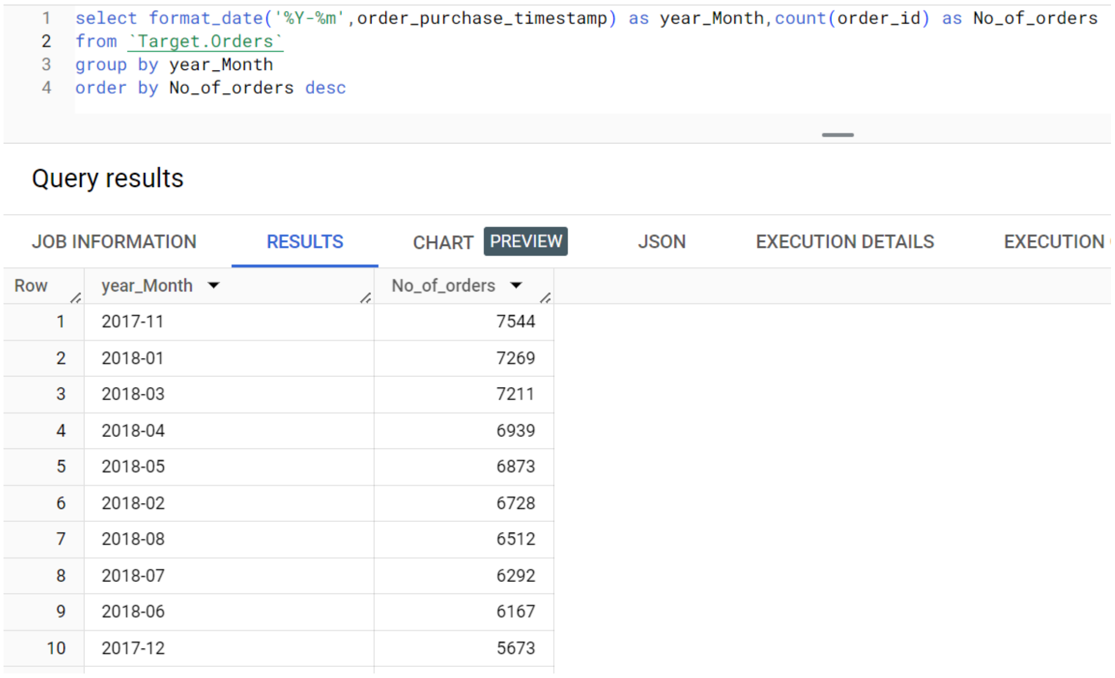
Insights : There is a growing trend over the past years. In 2016, we had fewer orders because we only have data from September. However, there was a 20% increase in orders placed in 2018 compared to the previous year.

Recommendations : NA

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

Query :

```
select format_date('%Y-%m',order_purchase_timestamp) as year_Month,count(order_id) as
No_of_orders
from `Target.Orders`
group by year_Month
order by No_of_orders desc
```



```
select format_date('%Y-%m',order_purchase_timestamp) as year_Month,count(order_id) as No_of_orders
from `Target.Orders`
group by year_Month
order by No_of_orders desc
```

Query results

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAILS | EXECUTION

Row	year_Month	No_of_orders
1	2017-11	7544
2	2018-01	7269
3	2018-03	7211
4	2018-04	6939
5	2018-05	6873
6	2018-02	6728
7	2018-08	6512
8	2018-07	6292
9	2018-06	6167
10	2017-12	5673

Insights :
The trend shows a peak in November 2017 with 7,544 orders, and December 2017 follows with 5,673 orders, likely influenced by holiday season shopping. There are certain months consistently attracting higher numbers of customers, possibly due to seasonal factors or promotional events.

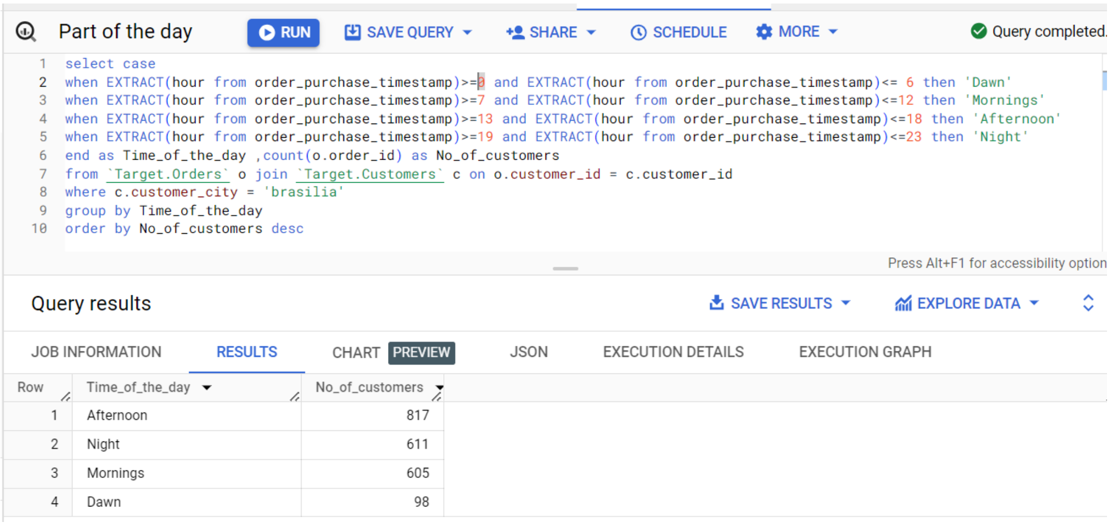
Recommendations :
Target seasonal promotions during high-demand months, such as November and December. Tailoring marketing strategies to align with these patterns can attract more customers during peak seasons and ensure sufficient inventory levels to meet increased demand.

2C) During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)

Query :

```
select case
when EXTRACT(hour from order_purchase_timestamp)>=0 and EXTRACT(hour from
order_purchase_timestamp)<= 6 then 'Dawn'
when EXTRACT(hour from order_purchase_timestamp)>=7 and EXTRACT(hour from
order_purchase_timestamp)<=12 then 'Mornings'
when EXTRACT(hour from order_purchase_timestamp)>=13 and EXTRACT(hour from
order_purchase_timestamp)<=18 then 'Afternoon'
when EXTRACT(hour from order_purchase_timestamp)>=19 and EXTRACT(hour from
order_purchase_timestamp)<=23 then 'Night'
end as Time_of_the_day ,count(o.order_id) as No_of_Orders
from `Target.Orders` o join `Target.Customers` c on o.customer_id = c.customer_id
where c.customer_city = 'brasilia'
```

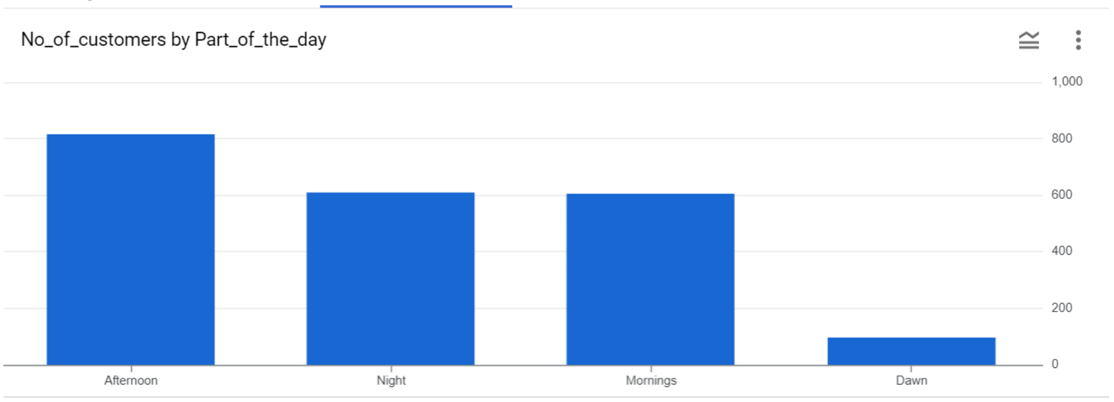
```
group by Time_of_the_day
order by No_of_Orders desc
```



```
select case
when EXTRACT(hour from order_purchase_timestamp)>=0 and EXTRACT(hour from order_purchase_timestamp)<= 6 then 'Dawn'
when EXTRACT(hour from order_purchase_timestamp)>=7 and EXTRACT(hour from order_purchase_timestamp)<=12 then 'Mornings'
when EXTRACT(hour from order_purchase_timestamp)>=13 and EXTRACT(hour from order_purchase_timestamp)<=18 then 'Afternoon'
when EXTRACT(hour from order_purchase_timestamp)>=19 and EXTRACT(hour from order_purchase_timestamp)<=23 then 'Night'
end as Time_of_the_day ,count(o.order_id) as No_of_customers
from `Target.Orders` o join `Target.Customers` c on o.customer_id = c.customer_id
where c.customer_city = 'brasilia'
group by Time_of_the_day
order by No_of_customers desc
```

Row	Time_of_the_day	No_of_customers
1	Afternoon	817
2	Night	611
3	Mornings	605
4	Dawn	98

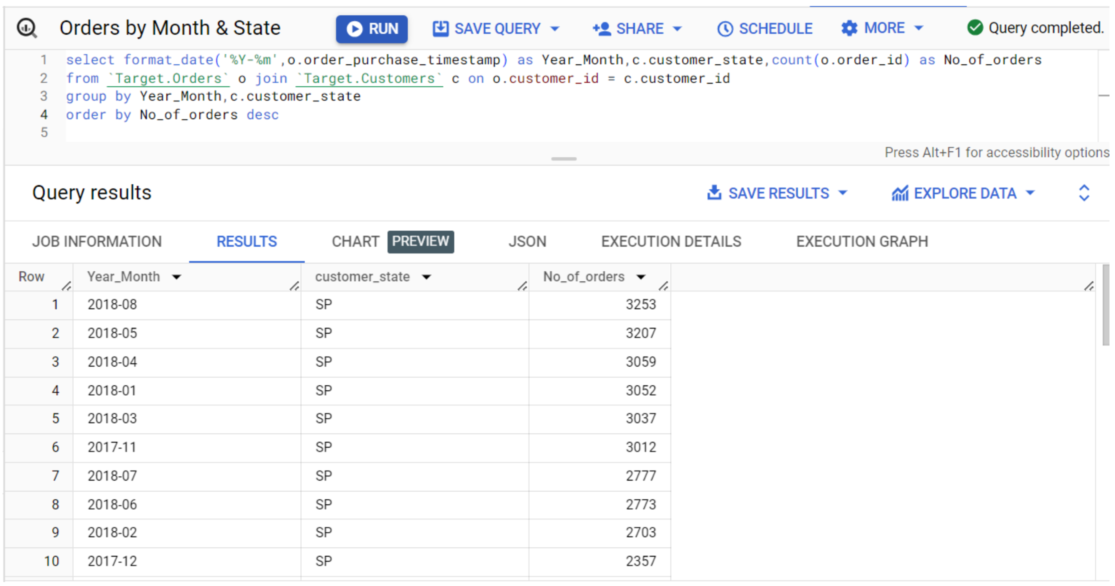
Insights :
Most of the Brazilian customers place their orders during Afternoon(817) followed by Night(611) and Mornings(605).



Recommendations :
Concentrate marketing efforts during the Afternoon, the peak ordering period.Efficient order processing and timely deliveries can enhance customer satisfaction during high-demand periods.To boost engagement during the Dawn period, consider introducing special promotions and exclusive early-morning deals.

3A) Get the month on month no. of orders placed in each state.
Query :

```
select format_date('%Y-%m',o.order_purchase_timestamp) as
Year_Month,c.customer_state,count(o.order_id) as No_of_orders
from `Target.Orders` o join `Target.Customers` c on o.customer_id = c.customer_id
group by Year_Month,c.customer_state
order by No_of_orders desc
```



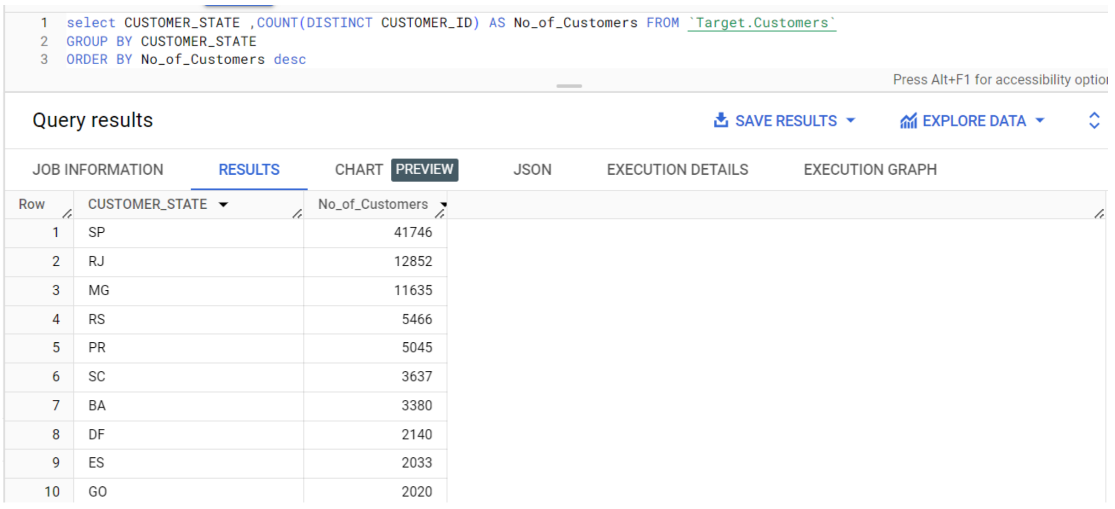
```
select format_date('%Y-%m',o.order_purchase_timestamp) as Year_Month,c.customer_state,count(o.order_id) as No_of_orders
from `Target.Orders` o join `Target.Customers` c on o.customer_id = c.customer_id
group by Year_Month,c.customer_state
order by No_of_orders desc
```

Row	Year_Month	customer_state	No_of_orders
1	2018-08	SP	3253
2	2018-05	SP	3207
3	2018-04	SP	3059
4	2018-01	SP	3052
5	2018-03	SP	3037
6	2017-11	SP	3012
7	2018-07	SP	2777
8	2018-06	SP	2773
9	2018-02	SP	2703
10	2017-12	SP	2357

Insights :
SP has the highest number of the orders(3253) in August 2018 compared to other states followed by **RJ**(1048) in Nov 2017.

Recommendations :
Analyze the operational and promotional strategies that contribute to **SP's** success and consider implementing similar strategies in other states.

3B) How are the customers distributed across all the states?
Query :
select CUSTOMER_STATE ,COUNT(DISTINCT CUSTOMER_ID) AS No_of_Customers FROM `Target.Customers`
GROUP BY CUSTOMER_STATE
ORDER BY No_of_Customers desc



```
select CUSTOMER_STATE ,COUNT(DISTINCT CUSTOMER_ID) AS No_of_Customers FROM `Target.Customers`
GROUP BY CUSTOMER_STATE
ORDER BY No_of_Customers desc
```

Row	CUSTOMER_STATE	No_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

Insights :
SP has the majority of the customers(41746) compared to other states followed by **RJ**(12852) and **MG**(11635).

Recommendations :
Analyze the operational and logistical practices that contribute to **SP's** success and consider implementing similar strategies in other states.

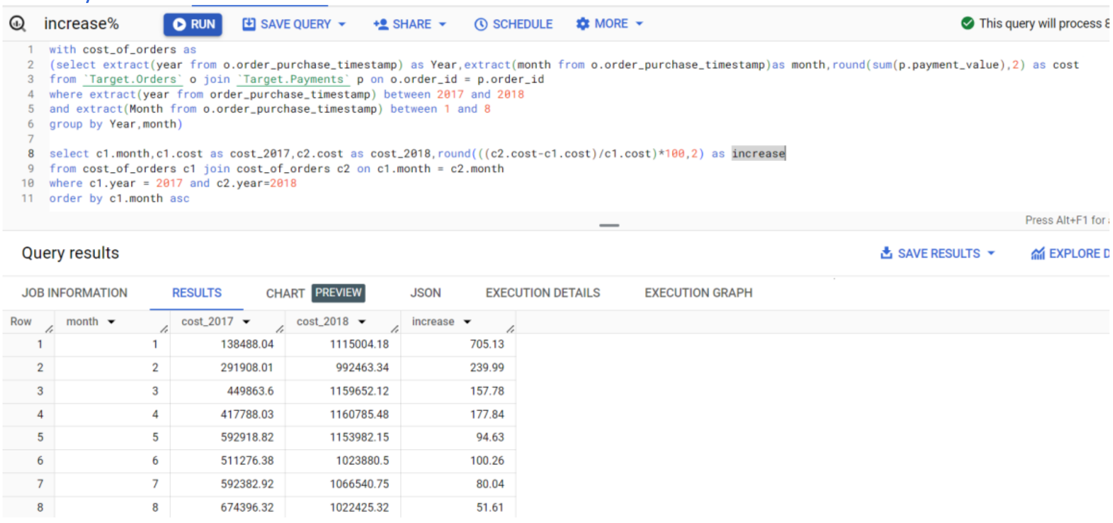
4A) Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only)

i) Comparision between months of 2017 and 2018

Query :

```
with cost_of_orders as
(select extract(year from o.order_purchase_timestamp) as Year,extract(month from
o.order_purchase_timestamp)as month,round(sum(p.payment_value),2) as cost
from `Target.Orders` o join `Target.Payments` p on o.order_id = p.order_id
where extract(year from order_purchase_timestamp) between 2017 and 2018
and extract(Month from o.order_purchase_timestamp) between 1 and 8
group by Year,month)

select c1.month,c1.cost as cost_2017,c2.cost as cost_2018,
round(((c2.cost-c1.cost)/c1.cost)*100,2) as increase
from cost_of_orders c1 join cost_of_orders c2 on c1.month = c2.month
where c1.year = 2017 and c2.year=2018
order by c1.month asc
```



```
with cost_of_orders as
(select extract(year from o.order_purchase_timestamp) as Year,extract(month from o.order_purchase_timestamp)as month,round(sum(p.payment_value),2) as cost
from `Target.Orders` o join `Target.Payments` p on o.order_id = p.order_id
where extract(year from order_purchase_timestamp) between 2017 and 2018
and extract(Month from o.order_purchase_timestamp) between 1 and 8
group by Year,month)

select c1.month,c1.cost as cost_2017,c2.cost as cost_2018,round(((c2.cost-c1.cost)/c1.cost)*100,2) as increase
from cost_of_orders c1 join cost_of_orders c2 on c1.month = c2.month
where c1.year = 2017 and c2.year=2018
order by c1.month asc
```

Query results

Row	month	cost_2017	cost_2018	increase
1	1	138488.04	1115004.18	705.13
2	2	291908.01	992463.34	239.99
3	3	449863.6	1159652.12	157.78
4	4	417788.03	1160785.48	177.84
5	5	592918.82	1153982.15	94.63
6	6	511276.38	1023880.5	100.26
7	7	592382.92	1066540.75	80.04
8	8	674396.32	1022425.32	51.61

Insights :
There is a noticeable increase in the cost of orders from 2017 to 2018 across all months. The percentage increase ranges from 51.61% in August to as high as 705.13% in January.The increase is particularly high in the early months (January and February), and the rate of increase slows down in the subsequent months.

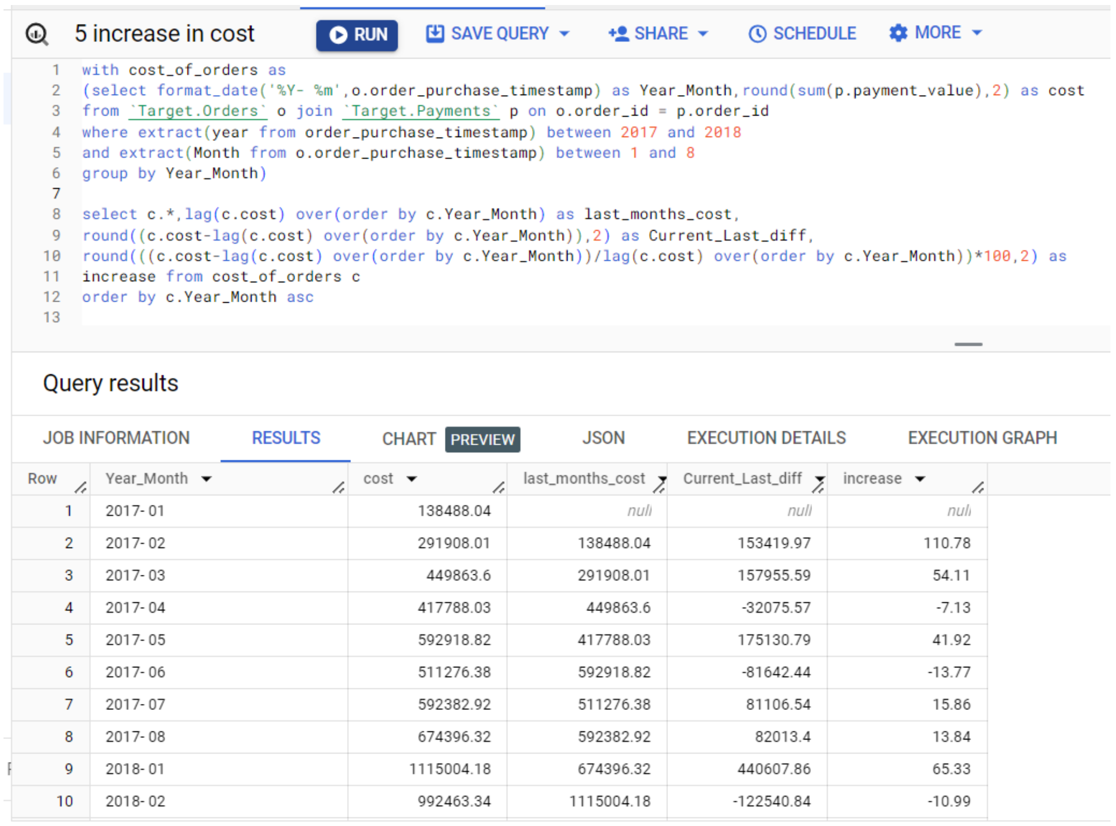
Recommendations :
The substantial increase in January (705.13%) warrants further investigation. Determine if there were any special promotions, marketing campaigns, or external factors contributing to this surge. If successful, consider replicating similar strategies during peak periods.

ii) **Comparision between consecutive months**

Query :

```
with cost_of_orders as
(select format_date('%Y- %m',o.order_purchase_timestamp) as Year_Month,
round(sum(p.payment_value),2) as cost
from `Target.Orders` o join `Target.Payments` p on o.order_id = p.order_id
where extract(year from order_purchase_timestamp) between 2017 and 2018
and extract(Month from o.order_purchase_timestamp) between 1 and 8
group by Year_Month)
```

```
select c.*,lag(c.cost) over(order by c.Year_Month) as last_months_cost,
round((c.cost-lag(c.cost) over(order by c.Year_Month)),2) as Current_Last_diff,
round(((c.cost-lag(c.cost) over(order by c.Year_Month))/lag(c.cost) over(order by
c.Year_Month))*100,2) as increase from cost_of_orders c
order by c.Year_Month asc
```



```
with cost_of_orders as
(select format_date('%Y- %m',o.order_purchase_timestamp) as Year_Month,round(sum(p.payment_value),2) as cost
from `Target.Orders` o join `Target.Payments` p on o.order_id = p.order_id
where extract(year from order_purchase_timestamp) between 2017 and 2018
and extract(Month from o.order_purchase_timestamp) between 1 and 8
group by Year_Month)

select c.*,lag(c.cost) over(order by c.Year_Month) as last_months_cost,
round((c.cost-lag(c.cost) over(order by c.Year_Month)),2) as Current_Last_diff,
round(((c.cost-lag(c.cost) over(order by c.Year_Month))/lag(c.cost) over(order by c.Year_Month))*100,2) as
increase from cost_of_orders c
order by c.Year_Month asc
```

Query results

Row	Year_Month	cost	last_months_cost	Current_Last_diff	increase
1	2017- 01	138488.04	null	null	null
2	2017- 02	291908.01	138488.04	153419.97	110.78
3	2017- 03	449863.6	291908.01	157955.59	54.11
4	2017- 04	417788.03	449863.6	-32075.57	-7.13
5	2017- 05	592918.82	417788.03	175130.79	41.92
6	2017- 06	511276.38	592918.82	-81642.44	-13.77
7	2017- 07	592382.92	511276.38	81106.54	15.86
8	2017- 08	674396.32	592382.92	82013.4	13.84
9	2018- 01	1115004.18	674396.32	440607.86	65.33
10	2018- 02	992463.34	1115004.18	-122540.84	-10.99

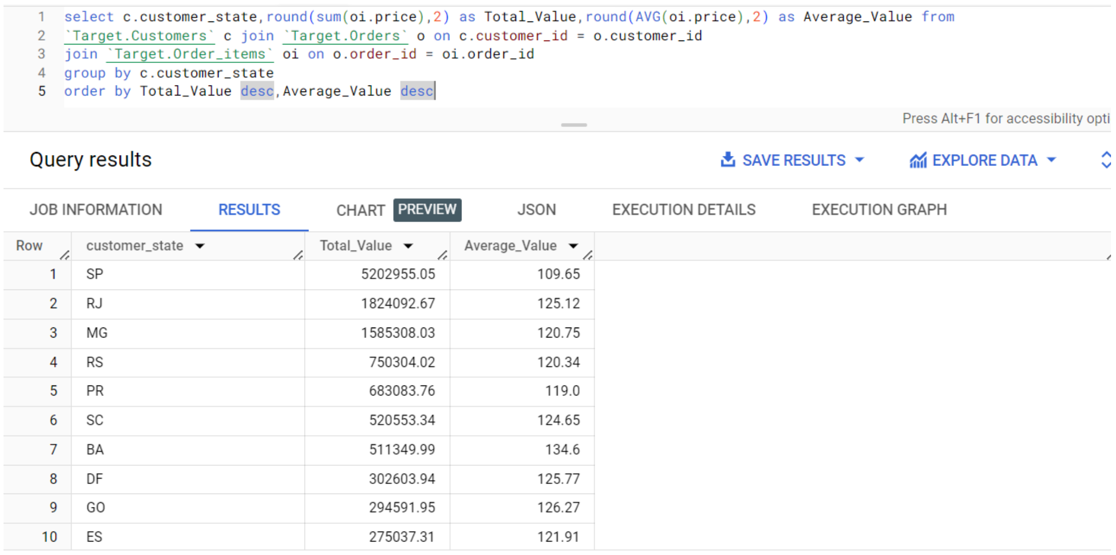
Insights :
Cost Fluctuations: There are fluctuations in costs over the specified time period, with notable increases and decreases in successive months.
Monthly Changes: The "Current_Last_diff" column indicates the month-to-month difference in cost.
Percentage Increase: The "increase" column represents the percentage change in costs compared to the previous month. Positive percentages denote an increase, while negative percentages indicate a decrease.

Recommendations :
Forecasting : Implement forecasting models to predict future payment trends based on historical data. This can help in proactive financial planning and decision-making, allowing the business to anticipate and adapt to changes.
Strategies : Develop and implement targeted marketing campaigns and promotions during months with notable decreases in payment values.

4B) Calculate the Total & Average value of order price for each state.

Query :

```
select c.customer_state,round(sum(oi.price),2) as Total_Value,round(AVG(oi.price),2) as
Average_Value from `Target.Customers` c join `Target.Orders` o on c.customer_id = o.customer_id
join `Target.Order_items` oi on o.order_id = oi.order_id
group by c.customer_state
order by Total_Value desc,Average_Value desc
```



```
select c.customer_state,round(sum(oi.price),2) as Total_Value,round(AVG(oi.price),2) as Average_Value from
`Target.Customers` c join `Target.Orders` o on c.customer_id = o.customer_id
join `Target.Order_items` oi on o.order_id = oi.order_id
group by c.customer_state
order by Total_Value desc,Average_Value desc
```

Query results

Row	customer_state	Total_Value	Average_Value
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

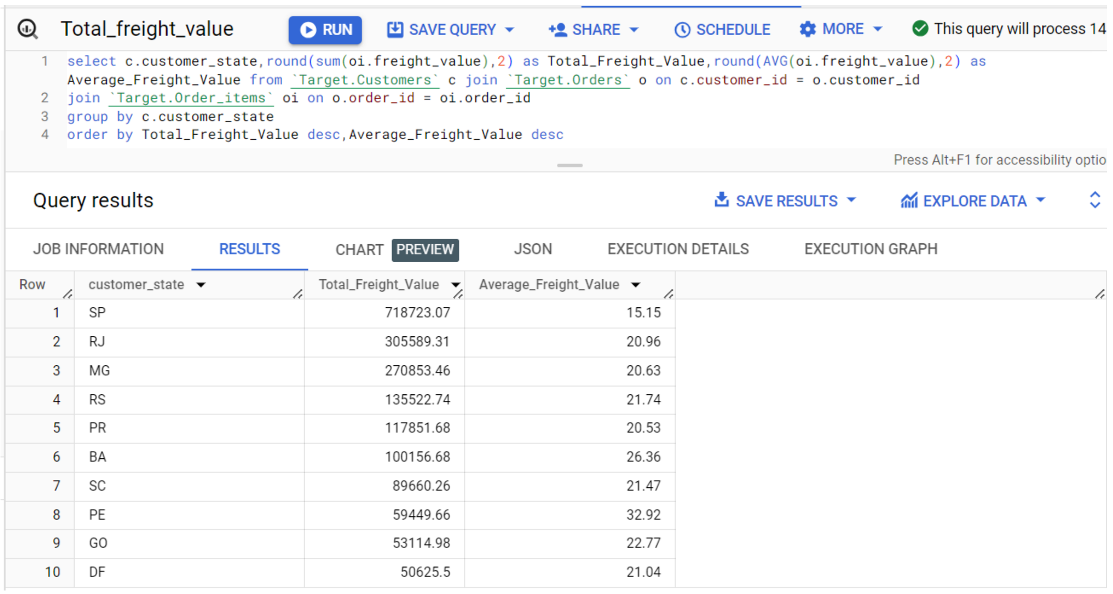
Insights :
SP has the highest order price value with 5.2 Million and Average order price is 109.65.
RR has the lowest order price value with 7829 and Average order price is 150.

Recommendations :
Since **SP** has the highest order price, ensure efficient order processing and timely deliveries can enhance customer satisfaction.To boost engagement in states having low order price value such as **RR, AP** consider introducing special promotions and coupons.

4C) Calculate the Total & Average value of order freight for each state.

Query :

```
select c.customer_state,round(sum(oi.freight_value),2) as Total_Freight_Value,
round(AVG(oi.freight_value),2) as Average_Freight_Value
from `Target.Customers` c join `Target.Orders` o on c.customer_id = o.customer_id
join `Target.Order_items` oi on o.order_id = oi.order_id
group by c.customer_state
order by Total_Freight_Value desc,Average_Freight_Value desc
```



Total_freight_value

```
select c.customer_state,round(sum(oi.freight_value),2) as Total_Freight_Value,round(AVG(oi.freight_value),2) as
Average_Freight_Value from `Target.Customers` c join `Target.Orders` o on c.customer_id = o.customer_id
join `Target.Order_items` oi on o.order_id = oi.order_id
group by c.customer_state
order by Total_Freight_Value desc,Average_Freight_Value desc
```

Query results

Row	customer_state	Total_Freight_Value	Average_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	59449.66	32.92
9	GO	53114.98	22.77
10	DF	50625.5	21.04

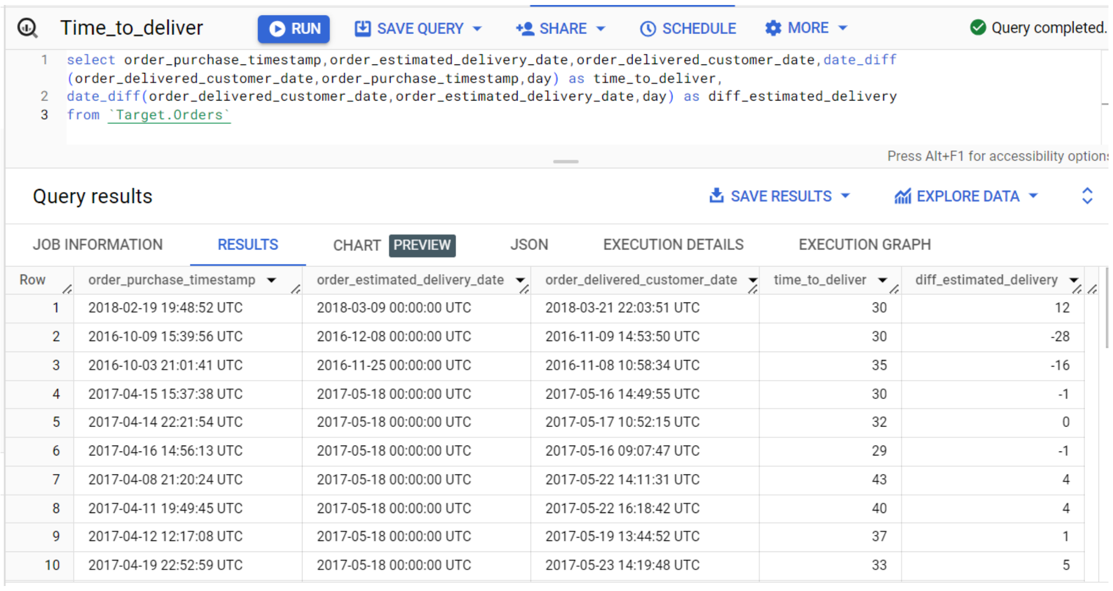
Insights :
SP has the highest order freight value with 7.2 Lakh and Average order freight is 15.15.
RR has the lowest order freight value with 2235.19 and Average order freight is 42.98.

Recommendations : SP has been dominating over the time period in terms of order price and customers.So negotiate bulk shipping discounts with carriers and explore efficient packaging solutions to reduce overall freight costs .

5A) 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.

Query :

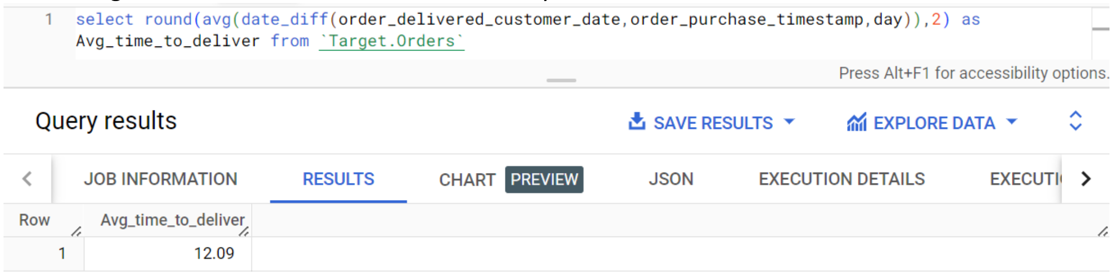
```
select order_purchase_timestamp,order_estimated_delivery_date,order_delivered_customer_date,
date_diff(order_delivered_customer_date,order_purchase_timestamp,day) as time_to_deliver,
date_diff(order_delivered_customer_date,order_estimated_delivery_date,day) as
diff_estimated_delivery from `Target.Orders`
```



```
select 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,
date_diff(order_delivered_customer_date,order_estimated_delivery_date,day) as diff_estimated_delivery
from `Target.Orders`
```

Row	order_purchase_timestamp	order_estimated_delivery_date	order_delivered_customer_date	time_to_deliver	diff_estimated_delivery
1	2018-02-19 19:48:52 UTC	2018-03-09 00:00:00 UTC	2018-03-21 22:03:51 UTC	30	12
2	2016-10-09 15:39:56 UTC	2016-12-08 00:00:00 UTC	2016-11-09 14:53:50 UTC	30	-28
3	2016-10-03 21:01:41 UTC	2016-11-25 00:00:00 UTC	2016-11-08 10:58:34 UTC	35	-16
4	2017-04-15 15:37:38 UTC	2017-05-18 00:00:00 UTC	2017-05-16 14:49:55 UTC	30	-1
5	2017-04-14 22:21:54 UTC	2017-05-18 00:00:00 UTC	2017-05-17 10:52:15 UTC	32	0
6	2017-04-16 14:56:13 UTC	2017-05-18 00:00:00 UTC	2017-05-16 09:07:47 UTC	29	-1
7	2017-04-08 21:20:24 UTC	2017-05-18 00:00:00 UTC	2017-05-22 14:11:31 UTC	43	4
8	2017-04-11 19:49:45 UTC	2017-05-18 00:00:00 UTC	2017-05-22 16:18:42 UTC	40	4
9	2017-04-12 12:17:08 UTC	2017-05-18 00:00:00 UTC	2017-05-19 13:44:52 UTC	37	1
10	2017-04-19 22:52:59 UTC	2017-05-18 00:00:00 UTC	2017-05-23 14:19:48 UTC	33	5

Insights : The above query results provides information about the time taken to deliver each order and difference between the estimated delivery date and order delivered date.
The average time taken to deliver a order is 12 days.



```
select round(avg(date_diff(order_delivered_customer_date,order_purchase_timestamp,day)),2) as
Avg_time_to_deliver from `Target.Orders`
```

Row	Avg_time_to_deliver
1	12.09

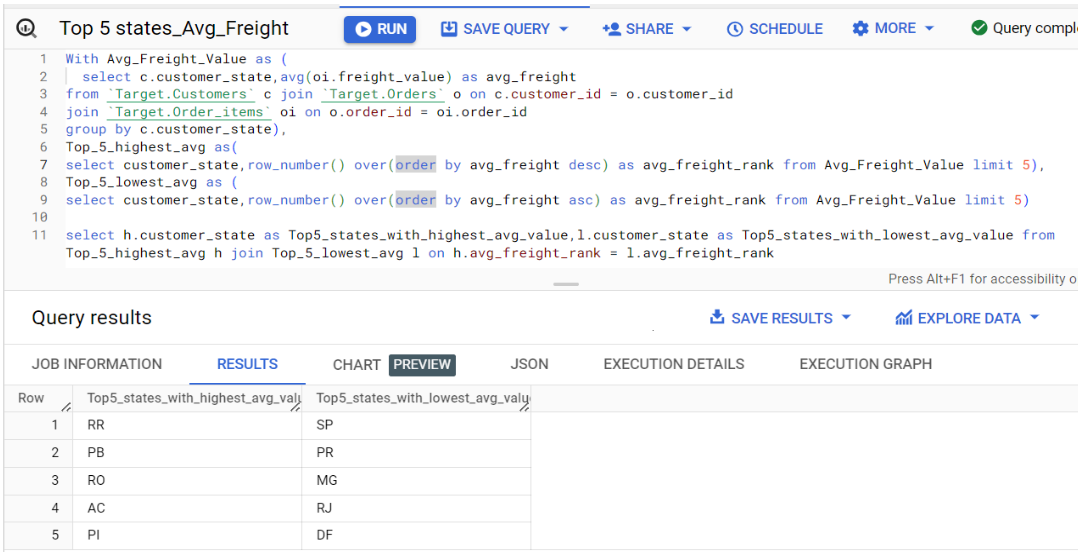
Recommendations:
Plan ahead for seasonal peaks or high-demand periods to ensure that fast delivery standards are maintained even during busy times.

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

Query :

```
With Avg_Freight_Value as (
 select c.customer_state,avg(oi.freight_value) as avg_freight
from `Target.Customers ` c join `Target.Orders` o on c.customer_id = o.customer_id
join `Target.Order_items` oi on o.order_id = oi.order_id
group by c.customer_state),
Top_5_highest_avg as(
select customer_state,row_number() over(order by avg_freight desc) as avg_freight_rank from
Avg_Freight_Value limit 5),
Top_5_lowest_avg as (
select customer_state,row_number() over(order by avg_freight asc) as avg_freight_rank from
Avg_Freight_Value limit 5)

select h.customer_state as Top5_states_with_highest_avg_value,l.customer_state as
Top5_states_with_lowest_avg_value from Top_5_highest_avg h join Top_5_lowest_avg l on
h.avg_freight_rank = l.avg_freight_rank
```



```
With Avg_Freight_Value as (
  select c.customer_state,avg(oi.freight_value) as avg_freight
from `Target.Customers` c join `Target.Orders` o on c.customer_id = o.customer_id
join `Target.Order_items` oi on o.order_id = oi.order_id
group by c.customer_state),
Top_5_highest_avg as(
select customer_state,row_number() over(order by avg_freight desc) as avg_freight_rank from Avg_Freight_Value limit 5),
Top_5_lowest_avg as (
select customer_state,row_number() over(order by avg_freight asc) as avg_freight_rank from Avg_Freight_Value limit 5)

select h.customer_state as Top5_states_with_highest_avg_value,l.customer_state as Top5_states_with_lowest_avg_value from
Top_5_highest_avg h join Top_5_lowest_avg l on h.avg_freight_rank = l.avg_freight_rank
```

Row	Top5_states_with_highest_avg_valu	Top5_states_with_lowest_avg_valu
1	RR	SP
2	PB	PR
3	RO	MG
4	AC	RJ
5	PI	DF

Insights :
The Top 5 States with highest average freight values are RR,PB,RO,AC and PI.
The Top 5 States with lowest average freight values are SP,PR,MG,RJ and DF.

Recommendations : Negotiate bulk shipping discounts with carriers and explore efficient packaging solutions to reduce overall freight costs in the states where the freight value are high.

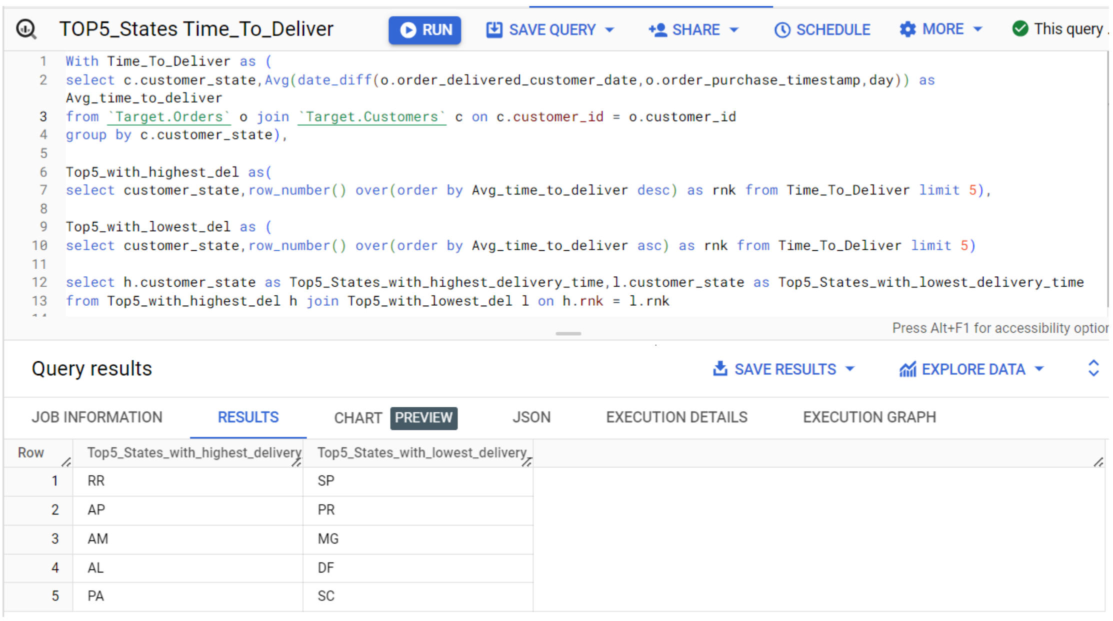
5C) Find out the top 5 states with the highest & lowest average delivery time.

Query :

```
With Time_To_Deliver as (
select c.customer_state,
Avg(date_diff(o.order_delivered_customer_date,o.order_purchase_timestamp,day)) as
Avg_time_to_deliver
from `Target.Orders` o join `Target.Customers` c on c.customer_id = o.customer_id
group by c.customer_state),

Top5_with_highest_del as(
select customer_state,row_number() over(order by Avg_time_to_deliver desc) as rnk from
Time_To_Deliver Order by rnk asc  limit 5),
Top5_with_lowest_del as (
select customer_state,row_number() over(order by Avg_time_to_deliver asc) as rnk from
Time_To_Deliver
Order by rnk asc limit 5)

select h.customer_state as Top5_States_with_highest_delivery_time,l.customer_state as
Top5_States_with_lowest_delivery_time
from Top5_with_highest_del h join Top5_with_lowest_del l on h.rnk = l.rnk
```



TOP5_States Time_To_Deliver

```
With Time_To_Deliver as (
select c.customer_state,Avg(date_diff(o.order_delivered_customer_date,o.order_purchase_timestamp,day)) as
Avg_time_to_deliver
from `Target.Orders` o join `Target.Customers` c on c.customer_id = o.customer_id
group by c.customer_state),

Top5_with_highest_del as(
select customer_state,row_number() over(order by Avg_time_to_deliver desc) as rnk from Time_To_Deliver limit 5),

Top5_with_lowest_del as (
select customer_state,row_number() over(order by Avg_time_to_deliver asc) as rnk from Time_To_Deliver limit 5)

select h.customer_state as Top5_States_with_highest_delivery_time,l.customer_state as Top5_States_with_lowest_delivery_time
from Top5_with_highest_del h join Top5_with_lowest_del l on h.rnk = l.rnk
```

Query results

Row	Top5_States_with_highest_delivery	Top5_States_with_lowest_delivery
1	RR	SP
2	AP	PR
3	AM	MG
4	AL	DF
5	PA	SC

Insights :

The Top 5 States with highest average delivery time are RR,AP,AM,AL and PA.
The Top 5 States with lowest average delivery time are SP,PR,MG,DF and SC.

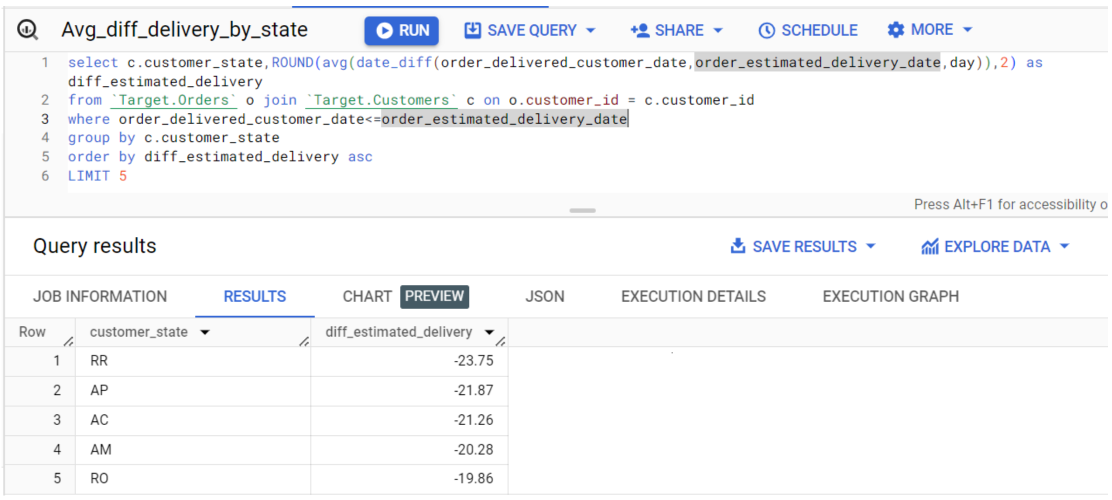
Recommendations:

Optimize logistics, explore local warehousing, and leverage technology to streamline processes, reducing the average delivery time in the states where we have highest delivery time to enhance customer satisfaction and competitiveness.

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

Query :

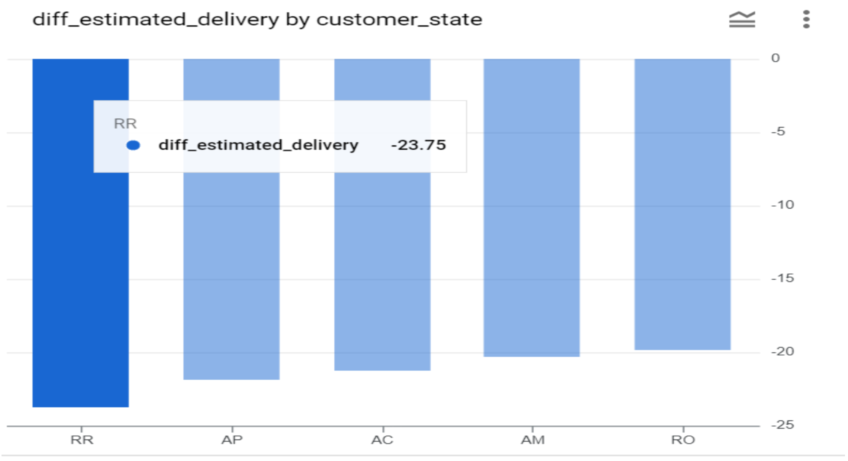
```
Select c.customer_state,
ROUND(avg(date_diff(order_delivered_customer_date,order_estimated_delivery_date,day)),2) as
diff_estimated_delivery
from `Target.Orders` o join `Target.Customers` c on o.customer_id = c.customer_id
where order_delivered_customer_date<=order_estimated_delivery_date
group by c.customer_state
order by diff_estimated_delivery asc
LIMIT 5
```



```
select c.customer_state,ROUND(avg(date_diff(order_delivered_customer_date,order_estimated_delivery_date,day)),2) as
diff_estimated_delivery
from `Target.Orders` o join `Target.Customers` c on o.customer_id = c.customer_id
where order_delivered_customer_date<=order_estimated_delivery_date
group by c.customer_state
order by diff_estimated_delivery asc
LIMIT 5
```

Row	customer_state	diff_estimated_delivery
1	RR	-23.75
2	AP	-21.87
3	AC	-21.26
4	AM	-20.28
5	RO	-19.86

Insights :



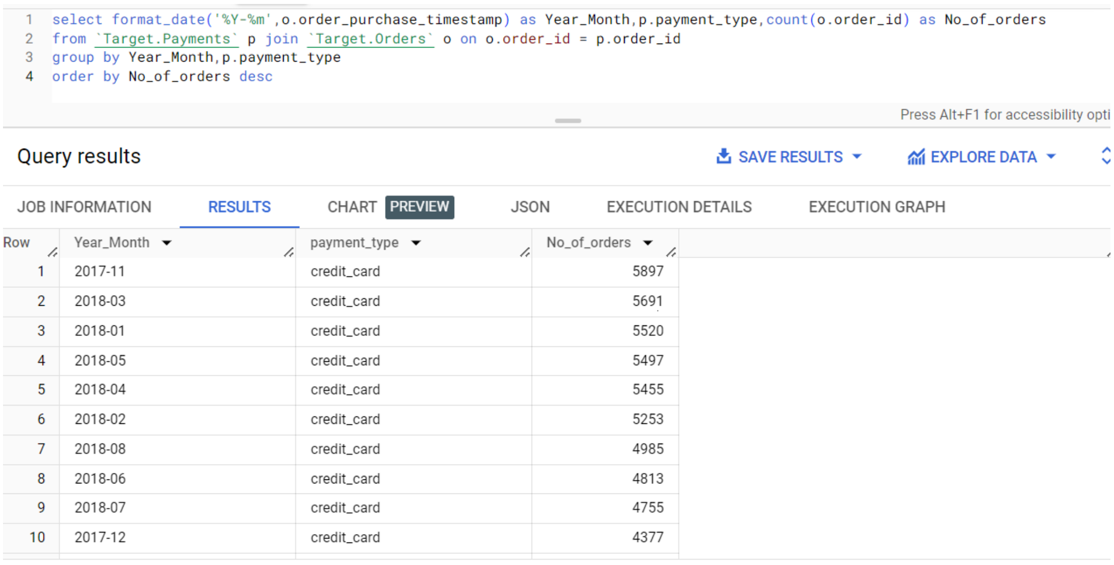
The graph above displays the top 5 states where the Delivery date either matches or precedes the Estimated delivery date. Among these states, RR holds the highest average number of orders delivered on or before the estimated delivery date with an average of 23 days, prior to the Estimated delivery date.

Recommendations :
Showcase **RR's** outstanding performance in delivery speed in your marketing materials and on your website. This can build trust and attract more customers from that state.Analyze the operational and logistical practices that contribute to **RR's** success and consider implementing similar strategies in other states to improve overall delivery performance.

6A) Find the month on month no. of orders placed using different payment types.

Query :

```
select format_date('%Y-%m',o.order_purchase_timestamp) as
Year_Month,p.payment_type,count(o.order_id) as No_of_orders
from `Target.Payments` p join `Target.Orders` o on o.order_id = p.order_id
group by Year_Month,p.payment_type
order by No_of_orders desc
```



```
select format_date('%Y-%m',o.order_purchase_timestamp) as Year_Month,p.payment_type,count(o.order_id) as No_of_orders
from `Target.Payments` p join `Target.Orders` o on o.order_id = p.order_id
group by Year_Month,p.payment_type
order by No_of_orders desc
```

Press Alt+F1 for accessibility opti

Query results

SAVE RESULTS EXPLORE DATA

JOB INFORMATION RESULTS CHART PREVIEW JSON EXECUTION DETAILS EXECUTION GRAPH

Row	Year_Month	payment_type	No_of_orders
1	2017-11	credit_card	5897
2	2018-03	credit_card	5691
3	2018-01	credit_card	5520
4	2018-05	credit_card	5497
5	2018-04	credit_card	5455
6	2018-02	credit_card	5253
7	2018-08	credit_card	4985
8	2018-06	credit_card	4813
9	2018-07	credit_card	4755
10	2017-12	credit_card	4377

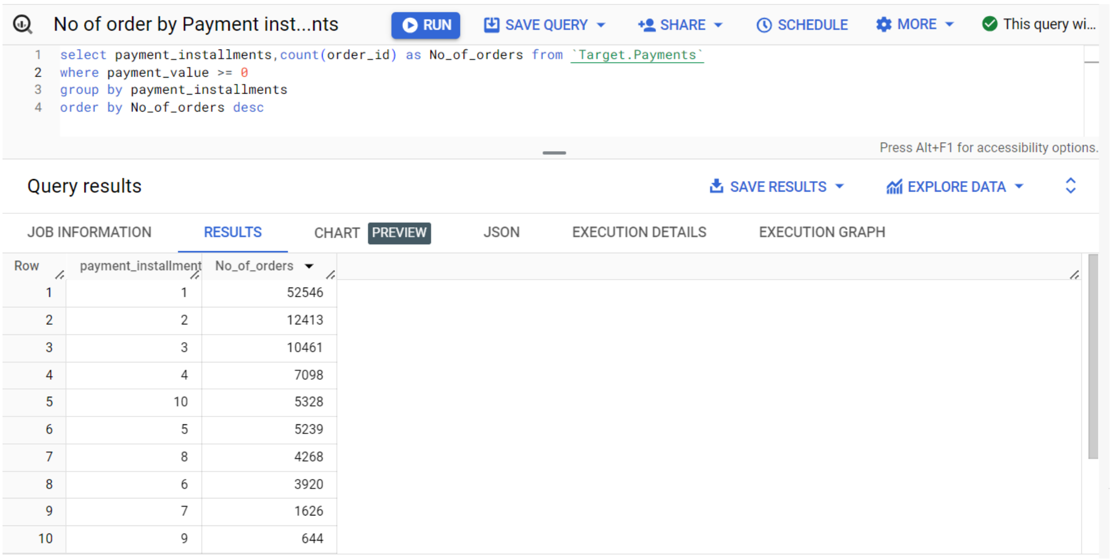
Insights :
The number of orders generally shows an increasing trend over time, with fluctuations in different months.Notable peaks are observed in months like November 2017.
"Credit Card" is consistently the dominant payment type across all months, with a significant number of orders.Other payment types such as "UPI," "Voucher," and "Debit Card" contribute to the overall order volume, but they are less dominant.

Recommendations :
Since credit card payments are most popular, consider running promotions specifically for credit card users to encourage continued use.
UPI Payments shows a consistent presence across all the months so initialize a campaign to offer them additional discounts or vouchers.

6B)Find the no. of orders placed on the basis of the payment installments that have been paid.

Query :

```
select payment_installments,count(order_id) as No_of_orders from `Target.Payments`
where payment_value >= 0
group by payment_installments
order by No_of_orders desc
```



No of order by Payment inst...nts RUN SAVE QUERY SHARE SCHEDULE MORE This query wi...

```
select payment_installments,count(order_id) as No_of_orders from `Target.Payments`
where payment_value >= 0
group by payment_installments
order by No_of_orders desc
```

Press Alt+F1 for accessibility options.

Query results

SAVE RESULTS EXPLORE DATA

JOB INFORMATION RESULTS CHART PREVIEW JSON EXECUTION DETAILS EXECUTION GRAPH

Row	payment_installment	No_of_orders
1	1	52546
2	2	12413
3	3	10461
4	4	7098
5	10	5328
6	5	5239
7	8	4268
8	6	3920
9	7	1626
10	9	644

Insights : The majority of orders (52,546) are paid in a single installment, followed by 12,413 orders with two payment installments and 10,461 orders with three installments.There is a decreasing trend in the number of orders for installments beyond the first few and Installments beyond 10 show lower frequencies, indicating that these are less common choices among customers.

Recommendations :
consider promoting **Single Installment Payments** to customers by including promotional offers or discounts.Provide clear information to customers about the available payment installment options. This can help them make better decisions based on their financial situations.Explore the possibility of introducing more flexible payment plans for customers who prefer multiple installments.