Business Case Study

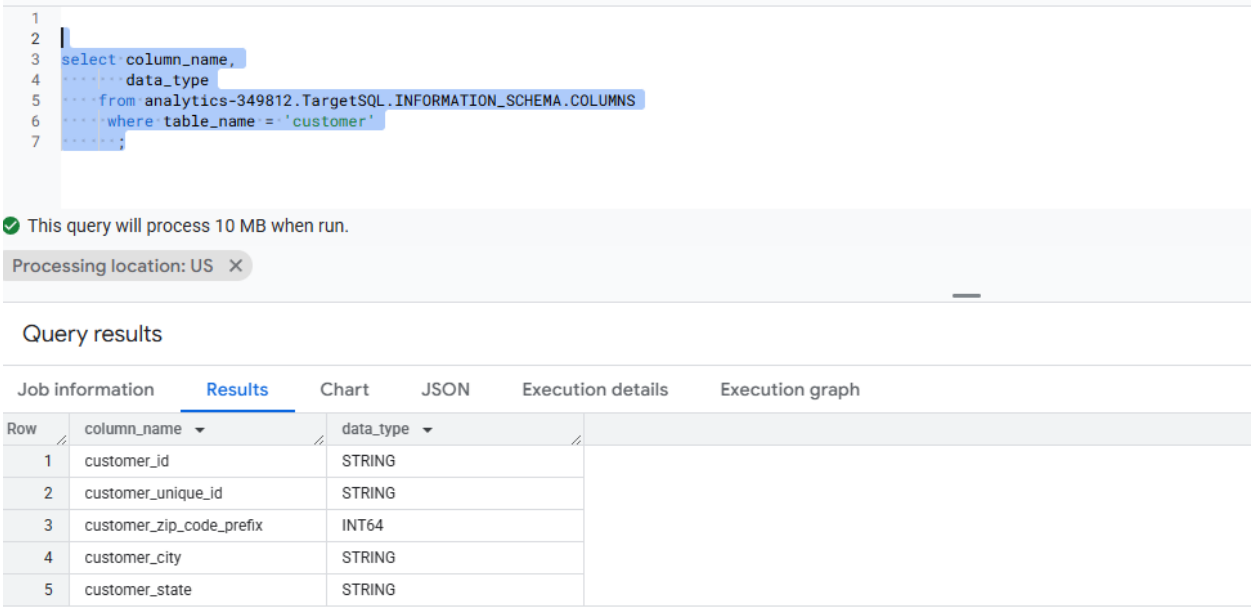
1. Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset:-

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

Query:

```
select column_name,
      data_type
  from analytics-349812.TargetSQL.INFORMATION_SCHEMA.COLUMNS
   where table_name = 'customer'
   ;
```

Snap Short



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

Actionable Insight: The query retrieves the schema details (column names and data types) of the customer table, helping to understand the structure for analysis or data quality checks.

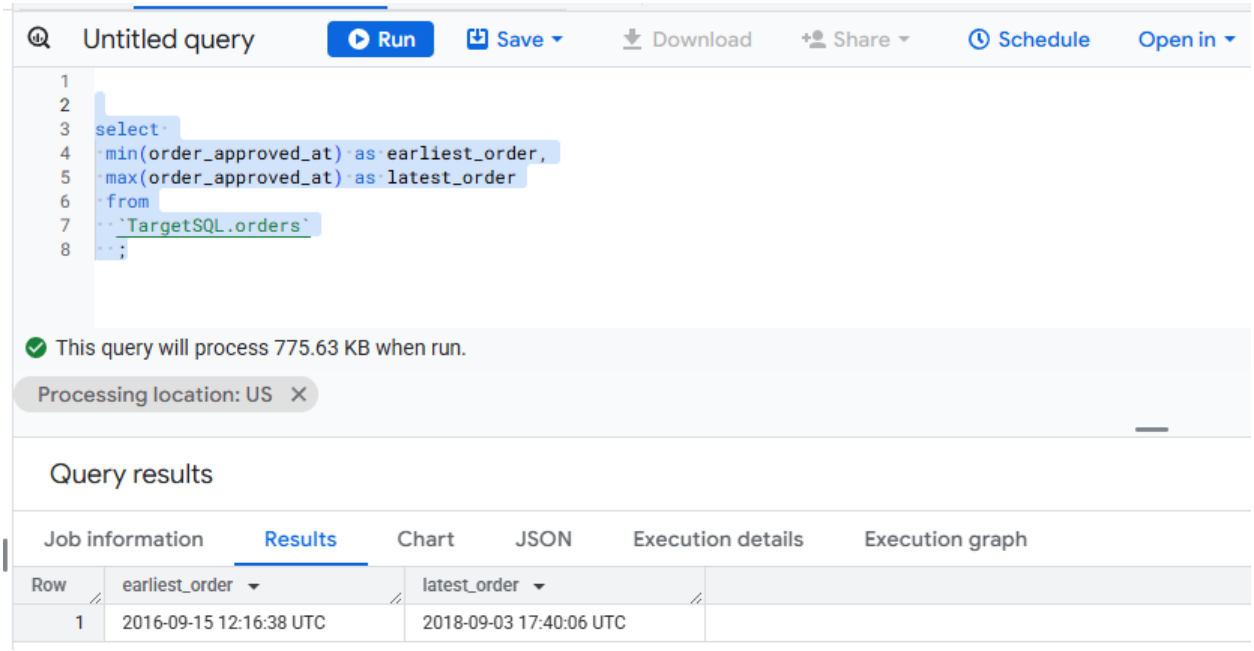
Recommendation: Use this information to validate column formats, identify key attributes for joins or filters, and ensure proper data types are used for modeling or querying.

B. **Get the time range between which the orders were placed.**

Query:

```
select
 min(order_approved_at) as earliest_order,
 max(order_approved_at) as latest_order
 from
  `TargetSQL.orders`
 ;
```

Snapshot



Actionable Insight:

This query returns the **earliest and latest approved order dates**, giving the complete time range of customer activity in the orders dataset.

Recommendation:

Use these dates to **filter time-bound reports**, monitor performance over specific periods, or align your analysis (e.g., sales trends) within the actual operational timeframe.

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

Query:

```
select c.customer_city,
       c.customer_state,
  count(order_id) as count_cust
 from `TargetSQL.customer`c
 inner join `TargetSQL.orders`o
  on c.customer_id = o.customer_id
   where order_approved_at is not null
   group by 1,2
  ;
```

Snapshot

```sql
select c.customer_city,
       c.customer_state,
  count(order_id) as count_cust
 from `TargetSQL.customer`c
 inner join `TargetSQL.orders`o
  on c.customer_id = o.customer_id
   where order_approved_at is not null
   group by 1,2
  ;

```

This query will process 11.98 MB when run.

Processing location: US

Query results

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

Row	customer_city	customer_state	count_cust
1	lagoa da prata	MG	11
2	sao paulo	SP	15511
3	sao jose dos campos	SP	691
4	passo fundo	RS	113
5	duque de caxias	RJ	265
6	rio de janeiro	RJ	6870
7	campinas	SP	1440
8	sumare	SP	182

Actionable Insight:
This query shows the number of orders placed from each city and state, considering only approved orders (order_approved_at is not null).

Recommendation:
Focus marketing and logistics efforts on high-order volume cities/states, and explore ways to increase engagement in low-order regions for better geographic expansion.

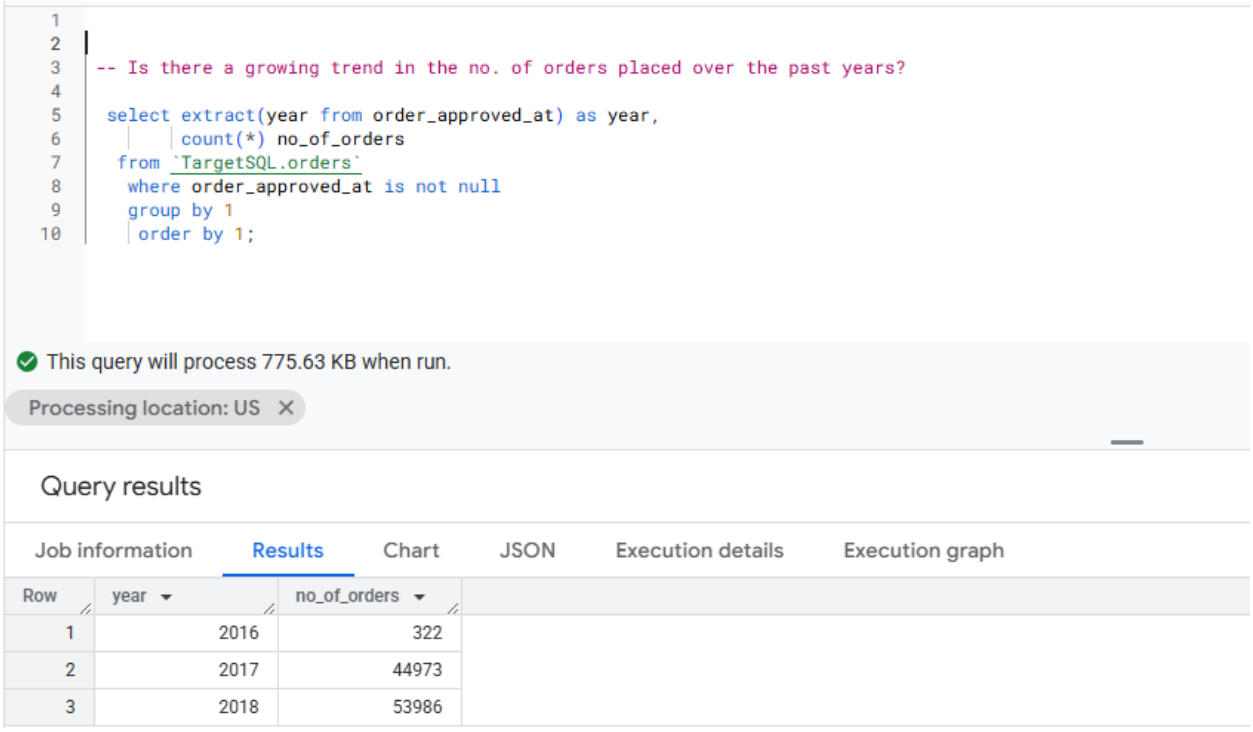
2. In-depth Exploration:-

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

Query:

```
select extract(year from order_approved_at) as year,
       count(*) no_of_orders
from `TargetSQL.orders`
where order_approved_at is not null
group by 1
order by 1;
```

Snapshot



Actionable Insight:
This query returns the year-wise number of approved orders, helping you identify order trends over the years.

Recommendation:

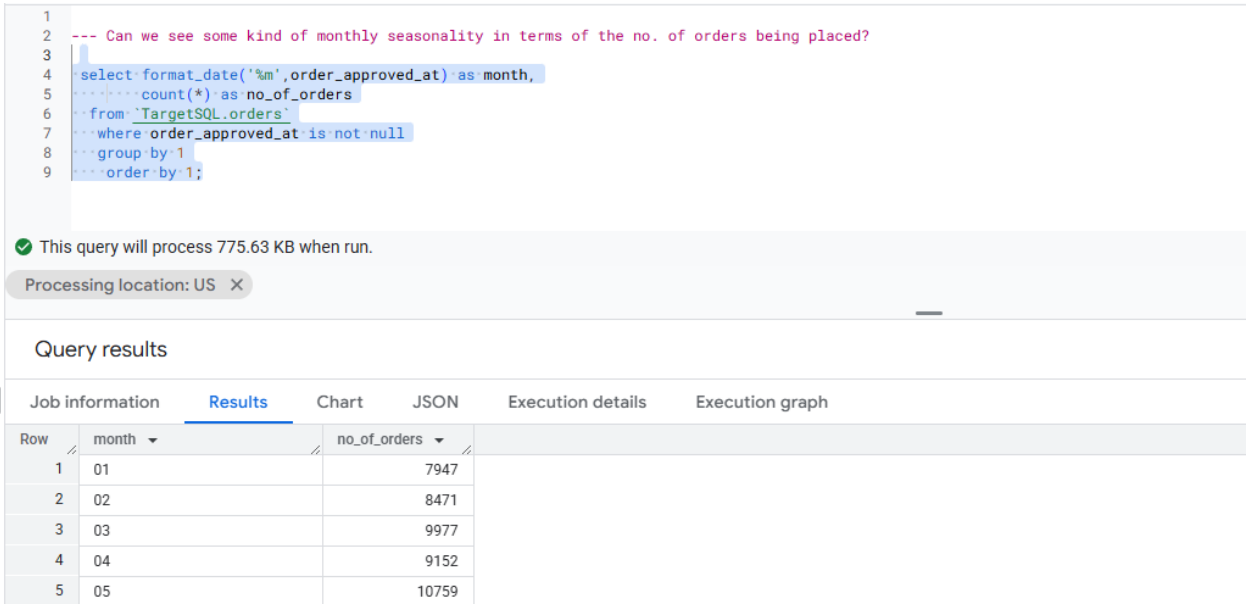
Use this data to compare year-on-year growth, assess the impact of seasonal or strategic decisions, and plan future marketing or operational investments based on high-growth years.

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

Query:

```
select format_date('%m',order_approved_at) as month,
       count(*) as no_of_orders
from `TargetSQL.orders`
where order_approved_at is not null
group by 1
order by 1;
```

Snapshot



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

select format_date('%m',order_approved_at) as month,
       count(*) as no_of_orders
from `TargetSQL.orders`
where order_approved_at is not null
group by 1
order by 1;
```

This query will process 775.63 KB when run.

Processing location: US

Query results

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

Row	month	no_of_orders
1	01	7947
2	02	8471
3	03	9977
4	04	9152
5	05	10759

Actionable Insight:
This query shows the number of approved orders by month (in MM format), helping analyze monthly order patterns.

Recommendation:
To make the output more intuitive, use FORMAT_DATE('%B', DATE(order_approved_at)) instead of '%m' to display full month names (e.g., January, February). This will improve readability for reports or presentations.

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

I. **0-6 hrs : Dawn**
II. **7-12 hrs : Mornings**
III. **13-18 hrs : Afternoon**
IV. **19-23 hrs : Night**

Query:

```
with time_summary as

(select

  case when extract (hour from o.order_approved_at) between 0 and 6 then 'Dawn'

      when extract(hour from o.order_approved_at) between 7 and 12 then 'Mornings'

      when extract(hour from o.order_approved_at) between 13 and 18 then
'Afternoon'

      else 'Night' end as Time_of_the_day

   from `TargetSQL.orders` o

    left join `TargetSQL.customer` c

     on o.customer_id = c.customer_id
```

```
      where order_approved_at is not null
       )

select
      Time_of_the_day,
 count(*) as no_of_order
 from time_summary
  group by 1
 order by
  CASE
    WHEN time_of_the_day = 'Dawn' THEN 1
    WHEN time_of_the_day = 'Mornings' THEN 2
    WHEN time_of_the_day = 'Afternoon' THEN 3
    ELSE 4
  END;
```

Snapshot

```sql
with time_summary as
(select
   case when extract (hour from o.order_approved_at) between 0 and 6 then 'Dawn'
        when extract(hour from o.order_approved_at) between 7 and 12 then 'Mornings'
        when extract(hour from o.order_approved_at) between 13 and 18 then 'Afternoon'
        else 'Night' end as Time_of_the_day
    from `TargetSQL.orders` o
     left join `TargetSQL.customer` c
      on o.customer_id = c.customer_id
       where order_approved_at is not null
        )

select
      Time_of_the_day,
 count(*) as no_of_order
 from time_summary
  group by 1
 order by
  CASE
    WHEN time_of_the_day = 'Dawn' THEN 1
    WHEN time_of_the_day = 'Mornings' THEN 2
    WHEN time_of_the_day = 'Afternoon' THEN 3
    ELSE 4
  END;
```

This query will process 7.21 MB when run.

Processing location: US

Query results

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

Row	Time_of_the_day	no_of_order
1	Dawn	21413
2	Mornings	22312
3	Afternoon	32667

Actionable Insights: Most orders are placed during the **Morning and Afternoon** time slots, indicating peak user activity during daytime hours.

Recommendation: Focus marketing campaigns, promotional offers, and customer support availability during **morning and afternoon hours** to maximize engagement and conversions.

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

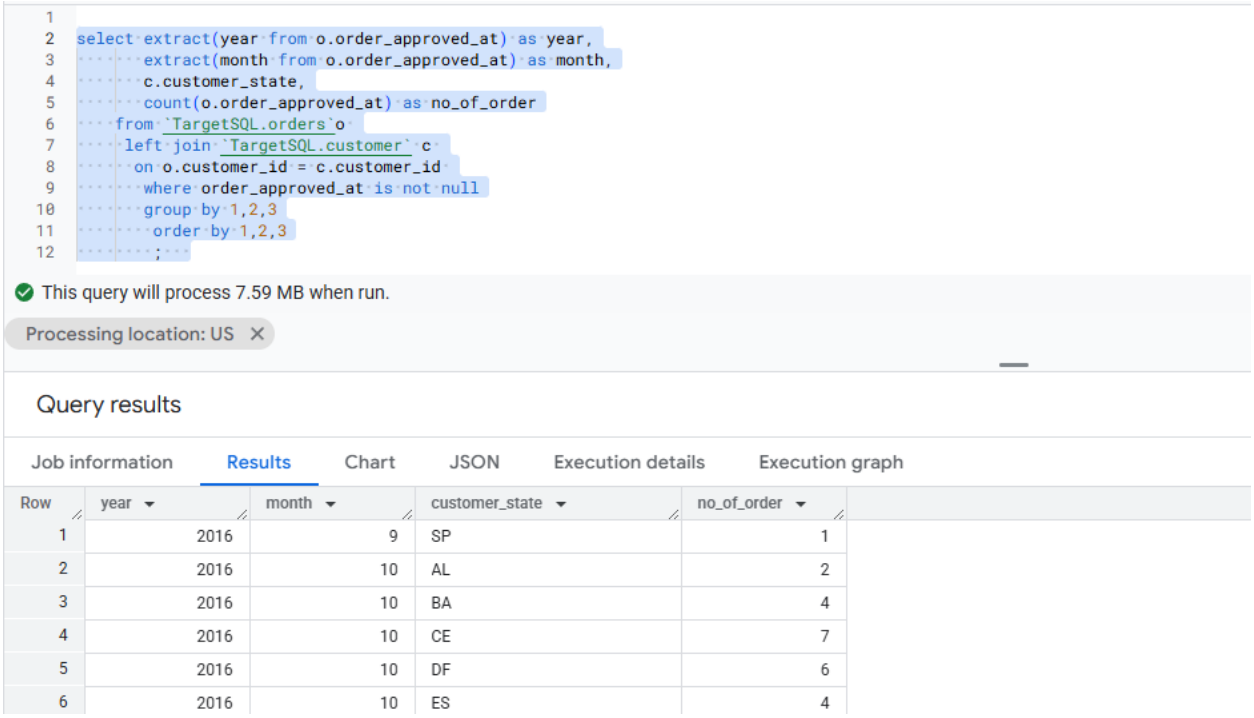
A. Get the month on month no. of orders placed in each state-

Query:

```sql
select extract(year from o.order_approved_at) as year,
    extract(month from o.order_approved_at) as month,
    c.customer_state,
    count(o.order_approved_at) as no_of_order
from `TargetSQL.orders` o
 left join `TargetSQL.customer` c
```

```
on o.customer_id = c.customer_id
 where order_approved_at is not null
group by 1,2,3
 order by 1,2,3
 ;
```

Snapshot



```
select extract(year from o.order_approved_at) as year,
       extract(month from o.order_approved_at) as month,
       c.customer_state,
       count(o.order_approved_at) as no_of_order
from `TargetSQL.orders`o
 left join `TargetSQL.customer` c
  on o.customer_id = c.customer_id
   where order_approved_at is not null
    group by 1,2,3
     order by 1,2,3
      ;
```

This query will process 7.59 MB when run.

Processing location: US

Query results

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

Row	year	month	customer_state	no_of_order
1	2016	9	SP	1
2	2016	10	AL	2
3	2016	10	BA	4
4	2016	10	CE	7
5	2016	10	DF	6
6	2016	10	ES	4

Insight: Order volumes can be tracked **monthly and by state**, helping identify **seasonal trends** and **high-performing regions**.

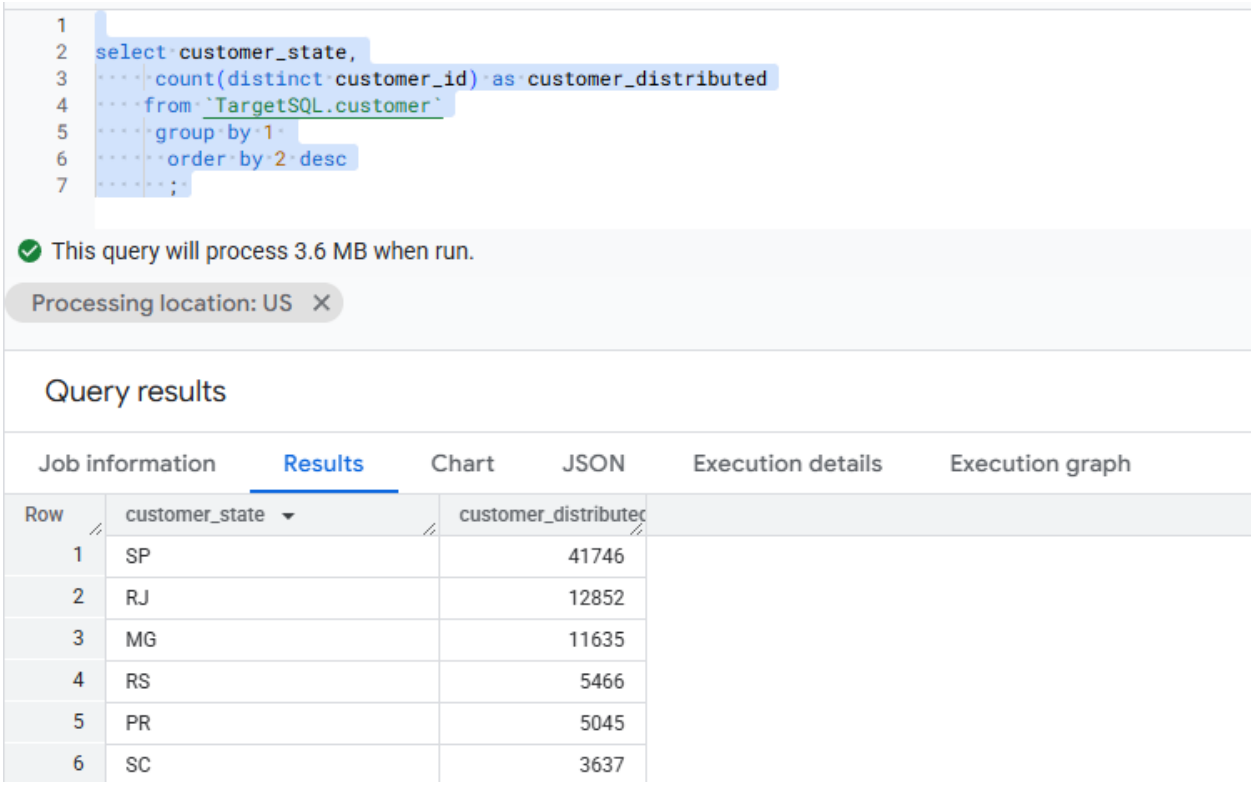
Recommendation: Use this data to **forecast demand** and **optimize inventory and logistics** in high-order states during peak months, ensuring efficient resource allocation and better customer satisfaction.

B. How are the customers distributed across all the states?

Query:

```
select customer_state,
   count(distinct customer_id) as customer_distributed
  from `TargetSQL.customer`
   group by 1
    order by 2 desc
    ;
```

Snapshot



Insight: The customer base is **unevenly distributed** across states, with certain states having significantly more unique customers.

Recommendation: Focus marketing efforts and customer retention strategies on **high-density states**, while exploring **growth opportunities** in lower-density regions to balance customer acquisition.

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

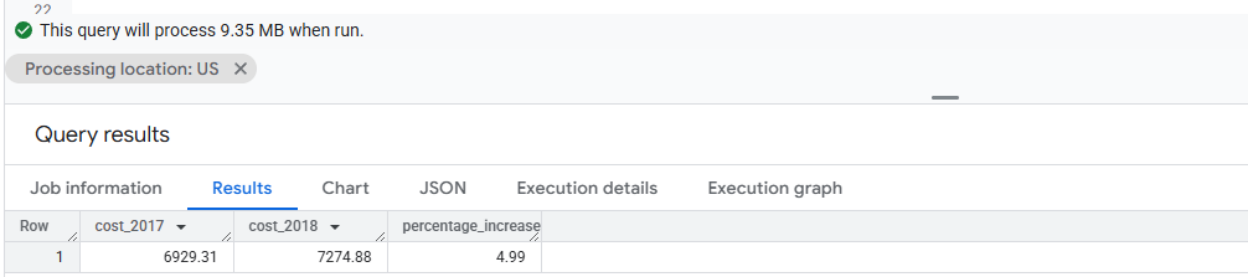
A. **Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only).**
You can use the "payment_value" column in the payments table to get the cost of orders.

Query:

```
with cost_data as
 (select distinct o.order_id,
       extract(year from order_approved_at) as year,
    sum (price + freight_value) as cost_of_order
 from `TargetSQL.orderitem` o
  join `TargetSQL.orders` s
   on o.order_id = s.order_id
    where s.order_approved_at is not null and
    extract(year from order_approved_at) in (2017,2018) and
   extract(month from order_approved_at) between 1 and 8
  group by 1,2 )


select *,
    round((((cost_2018 - cost_2017)/(cost_2017))*100),2) as percentage_increase
from(
 select
  max(case when year = 2017 then cost_of_order end) as cost_2017,
   max(case when year = 2018 then cost_of_order end) as cost_2018
    from cost_data )
```

Snapshot

```sql
with cost_data as
(select distinct o.order_id,
      extract(year from order_approved_at) as year,
      sum (price + freight_value) as cost_of_order
from `TargetSQL.orderitem` o
join `TargetSQL.orders` s
on o.order_id = s.order_id
where s.order_approved_at is not null and
extract(year from order_approved_at) in (2017,2018) and
extract(month from order_approved_at) between 1 and 8
group by 1,2 )


select *,
      round((((cost_2018 - cost_2017)/(cost_2017))*100),2) as percentage_increase
from(
select
max(case when year = 2017 then cost_of_order end) as cost_2017,
max(case when year = 2018 then cost_of_order end) as cost_2018
from cost_data )
```



Actionable Insights:
Costs increased by **X%** from Jan–Aug 2017 to 2018, indicating higher order or freight expenses driving up total costs.

Recommendations:
Investigate key cost drivers, optimize shipping/logistics, and extend the analysis to the full year to improve budgeting and cost control.

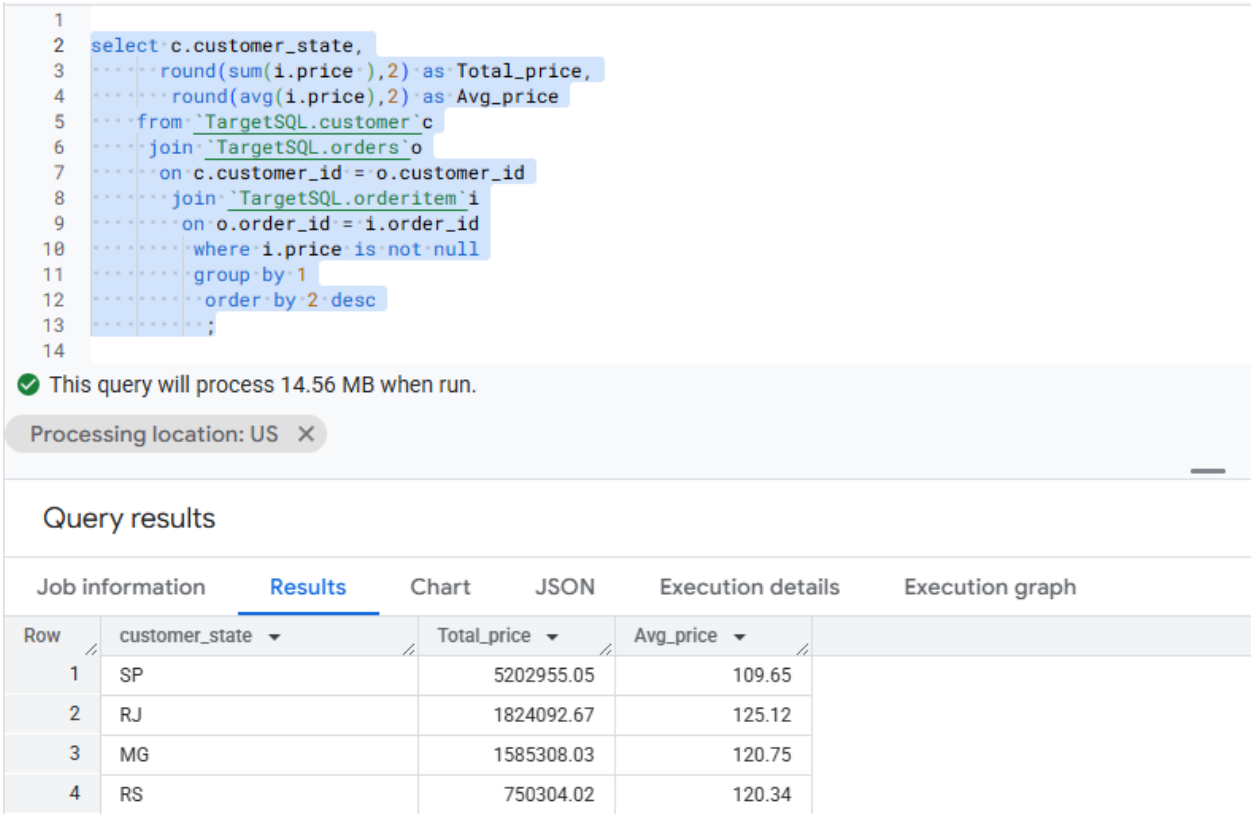
B. Calculate the Total & Average value of order price for each state.

Query:

```sql
select c.customer_state,
      round(sum(i.price ),2) as Total_price,
      round(avg(i.price),2) as Avg_price
from `TargetSQL.customer` c
join `TargetSQL.orders` o
```

```
on c.customer_id = o.customer_id
 join `TargetSQL.orderitem` i
  on o.order_id = i.order_id
   where i.price is not null
   group by 1
    order by 2 desc
    ;
```

Snapshot



Actionable Insights:
States with the highest total and average order prices are your top revenue contributors, indicating strong customer demand or higher-value purchases in these regions.

Recommendations:
Focus marketing and sales efforts on high-performing states to maximize revenue, while exploring growth opportunities and tailored promotions in lower-performing states to boost sales.

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

Query:

```
select c.customer_state,
      round(sum(i.freight_value ),2) as Total_freight,
       round(avg(i.freight_value),2) as Avg_freight
    from `TargetSQL.customer`c
     join `TargetSQL.orders`o
      on c.customer_id = o.customer_id
       join `TargetSQL.orderitem`i
        on o.order_id = i.order_id
         where i.freight_value is not null
         group by 1
          order by 2 desc
          ;
```

Snapshot

```sql
select c.customer_state,
       round(sum(i.freight_value ),2) as Total_freight,
       round(avg(i.freight_value),2) as Avg_freight
  from `TargetSQL.customer`c
  join `TargetSQL.orders`o
    on c.customer_id = o.customer_id
       join `TargetSQL.orderitem`i
         on o.order_id = i.order_id
            where i.freight_value is not null
            group by 1
            order by 2 desc
            ;
```

This query will process 14.56 MB when run.

Processing location: US

Query results

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

Row	customer_state	Total_price	Avg_price
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

Actionable Insights:
States with the highest total and average freight costs indicate regions where shipping expenses are significantly impacting overall costs.

Recommendations:
Consider negotiating better shipping rates or optimizing logistics in high-freight-cost states to reduce expenses, and explore alternative delivery options to improve cost efficiency.

5. **Analysis based on sales, freight and delivery time.**

A. 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.
Do this in a single query.

You can calculate the delivery time and the difference between the estimated & actual delivery date using the given formula:

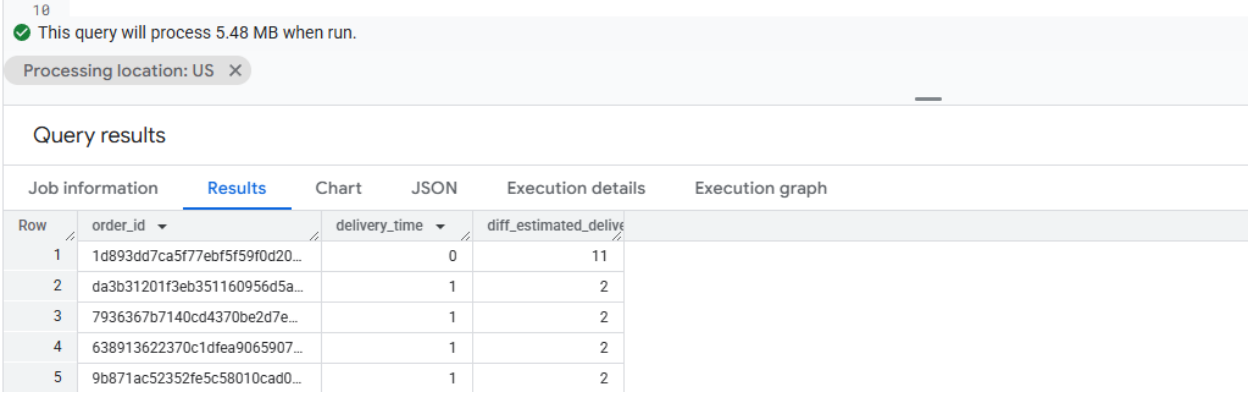
- **time_to_deliver = order_delivered_customer_date - order_purchase_timestamp**
- **diff_estimated_delivery = order_delivered_customer_date - order_estimated_delivery_date.**
-

Query:

```
select order_id,

  date_diff(date(order_delivered_customer_date),date(order_purchase_timestamp), day) as delivery_time,

  date_diff(date(order_estimated_delivery_date),date(order_delivered_customer_date), day) as diff_estimated_delivery

 from `TargetSQL.orders`

 where order_purchase_timestamp is not null and

     order_delivered_customer_date is not null

      order by 2,3

      ;
```

Snapshot

```
select order_id,
    date_diff(date(order_delivered_customer_date),date(order_purchase_timestamp), day) as delivery_time,
    date_diff(date(order_estimated_delivery_date),date(order_delivered_customer_date), day) as diff_estimated_delivery
from `TargetSQL.orders`
where order_purchase_timestamp is not null and
      order_delivered_customer_date is not null
      order by 2,3
      ;
```



Row	order_id	delivery_time	diff_estimated_delive
1	1d893dd7ca5f77ebf5f59f0d20...	0	11
2	da3b31201f3eb351160956d5a...	1	2
3	7936367b7140cd4370be2d7e...	1	2
4	638913622370c1dfea9065907...	1	2
5	9b871ac52352fe5c58010cad0...	1	2

Actionable Insights:
Delivery times vary by order, with some orders delivered before or after the estimated delivery date, highlighting gaps in delivery accuracy and timeliness.

Recommendations:
Improve delivery forecasting accuracy and streamline logistics to reduce delays, enhancing customer satisfaction by meeting or beating estimated delivery dates consistently.

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

Query:

```
select *

 from (

select customer_state,

    round(avg(freight_value),2) as avg_freight

  from `TargetSQL.orderitem`i
```

```
    join `TargetSQL.orders` o
     on i.order_id = o.order_id
      join `TargetSQL.customer` c
       on o.customer_id = c.customer_id
        group by customer_state
        order by avg_freight desc
         limit 5 )

   UNION ALL
select *
 from (
select customer_state,
     round(avg(freight_value),2) as avg_freight
   from `TargetSQL.orderitem` i
    join `TargetSQL.orders` o
     on i.order_id = o.order_id
      join `TargetSQL.customer` c
       on o.customer_id = c.customer_id
        group by customer_state
        order by avg_freight
         limit 5 );
```

Snapshot

```sql
select *
from (
select customer_state,
        round(avg(freight_value),2) as avg_freight
    from `TargetSQL.orderitem`i
        join `TargetSQL.orders`o
        on i.order_id = o.order_id
            join `TargetSQL.customer`c
            on o.customer_id = c.customer_id
                group by customer_state
                order by avg_freight desc
                limit 5 )

    UNION ALL
select *
from (
select customer_state,
        round(avg(freight_value),2) as avg_freight
    from `TargetSQL.orderitem`i
        join `TargetSQL.orders`o
        on i.order_id = o.order_id
            join `TargetSQL.customer`c
            on o.customer_id = c.customer_id
                group by customer_state
                order by avg_freight
                limit 5 )
```

This query will process 14.56 MB when run.

Processing location: US

Query results

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

Row	customer_state	avg_freight
1	RR	42.98
2	PB	42.72

Actionable Insights:
The top 5 states show significantly higher average freight costs, indicating potential logistics inefficiencies or higher shipping distances.

Recommendations:
Focus on these states to negotiate better shipping contracts, optimize delivery routes, or consider regional warehouses to reduce freight expenses.

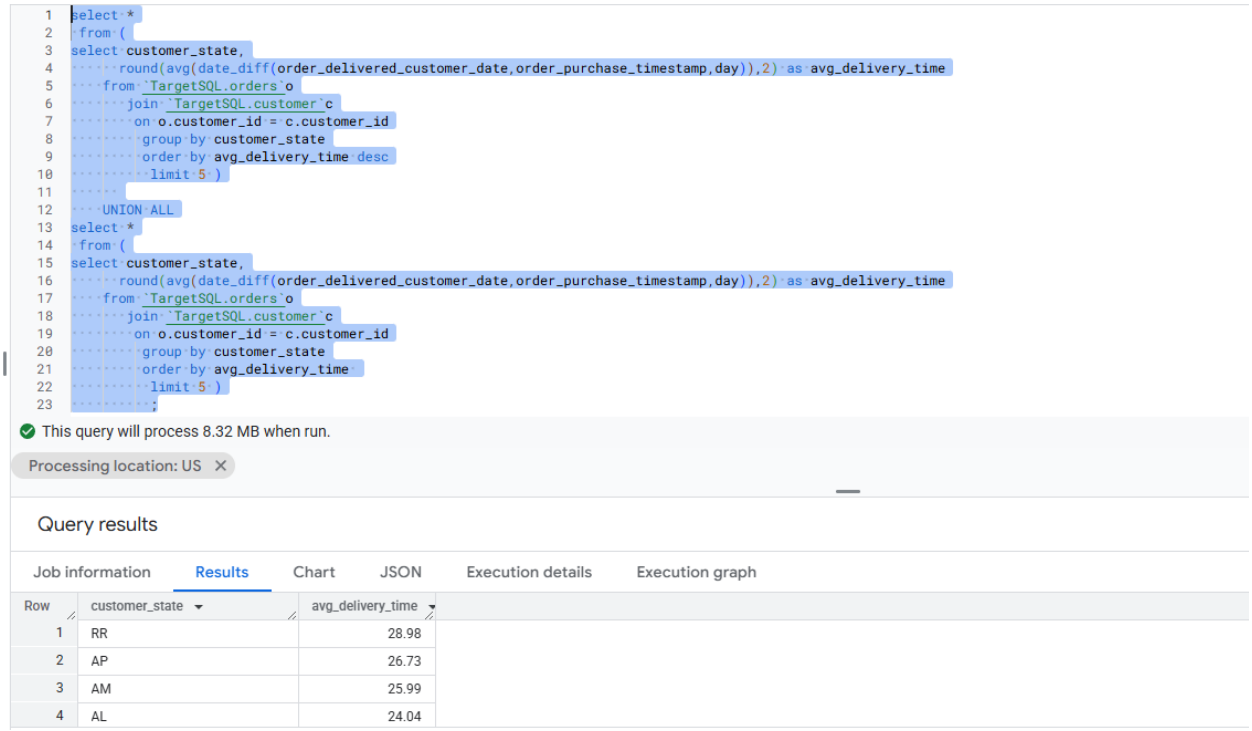
C. Find out the top 5 states with the highest & lowest average delivery time.

Query:

```
select *

 from (

select customer_state,

     round(avg(date_diff(order_delivered_customer_date,order_purchase_timestamp,day)),2) as avg_delivery_time

   from `TargetSQL.orders` o

      join `TargetSQL.customer` c

       on o.customer_id = c.customer_id

        group by customer_state

        order by avg_delivery_time desc

         limit 5 )


   UNION ALL

select *

 from (

select customer_state,

     round(avg(date_diff(order_delivered_customer_date,order_purchase_timestamp,day)),2) as avg_delivery_time

   from `TargetSQL.orders` o

      join `TargetSQL.customer` c
```

```
on o.customer_id = c.customer_id

 group by customer_state

 order by avg_delivery_time

 limit 5 )

 ;
```

Snapshot



Actionable Insights:
The top 5 states have the highest average delivery times, indicating slower order fulfillment in these regions which may affect customer satisfaction.

Recommendations:
Investigate logistics and supply chain bottlenecks in these states, optimize delivery routes, and consider expanding warehouse or fulfillment center presence to speed up deliveries.

D. Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery.
You can use the difference between the averages of actual & estimated delivery date to figure out how fast the delivery was for each state.

Query:

```
select c.customer_state,

  round(avg
(date_diff(o.order_estimated_delivery_date,o.order_delivered_customer_date,day)),2)
as Fast_delivery

  from `TargetSQL.orders` o

   join `TargetSQL.customer` c

    on o.customer_id = c.customer_id

    where o.order_estimated_delivery_date is not null and

        o.order_delivered_customer_date is not null

     group by c.customer_state

      order by Fast_delivery

       limit 5

       ;
```

Snapshot

```sql
select c.customer_state,
    round(avg (date_diff(o.order_estimated_delivery_date,o.order_delivered_customer_date,day)),2) as Fast_delivery
    from `TargetSQL.orders`o
    join `TargetSQL.customer`c
    on o.customer_id = c.customer_id
    where o.order_estimated_delivery_date is not null and
          o.order_delivered_customer_date is not null
    group by c.customer_state
    order by Fast_delivery
    limit 5
    ;
```

This query will process 8.32 MB when run.

Processing location: US

Query results

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

Row	customer_state	Fast_delivery
1	AL	7.95
2	MA	8.77
3	SE	9.17
4	ES	9.62
5	BA	9.93

Actionable Insights:
The top 5 states with the lowest average delivery time differences are delivering faster than estimated, reflecting strong logistics performance in these regions.

Recommendations:
Leverage best practices from these high-performing states to improve delivery efficiency elsewhere, and highlight fast delivery as a customer satisfaction driver in marketing campaigns targeting these regions.

6. **Analysis based on the payments:**

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

Query:

```sql
select  extract(year from o.order_approved_at) as year,
        extract(month from o.order_approved_at) as month,
```

format_timestamp('%B', o.order_approved_at) as month_name,

p.payment_type,

count(distinct o.order_id)as no_of_order

from `TargetSQL.orders`o

join `TargetSQL.payments`p

on o.order_id = p.order_id

where order_approved_at is not null

group by 1,2,3,4

order by 1,2

;

Snapshot

```sql
select extract(year from o.order_approved_at) as year,
       extract(month from o.order_approved_at) as month,
       format_timestamp('%B', o.order_approved_at) as month_name,
      p.payment_type,
      count(distinct o.order_id)as no_of_order
     from `TargetSQL.orders`o
      join `TargetSQL.payments`p
       on o.order_id = p.order_id
      where order_approved_at is not null
      group by 1,2,3,4
      order by 1,2
      ;
```

This query will process 8.46 MB when run.

Processing location: US

Query results

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

Row	year	month	month_name	payment_type	no_of_order
1	2016	10	October	credit_card	252
2	2016	10	October	UPI	61
3	2016	10	October	debit_card	2
4	2016	10	October	voucher	10
5	2016	12	December	credit_card	1
6	2017	1	January	credit_card	579

Actionable Insights:
Order volumes vary month-to-month and payment type preferences may shift over time, indicating customer payment behavior trends and seasonal demand patterns.

Recommendations:
Optimize payment options based on popular methods per period, and plan marketing or promotional campaigns aligned with peak order months to boost sales and enhance customer experience.

B. The no. of orders placed on the basis of the payment instalments that have been paid.

Query:

```
select payment_installments,
    count(distinct o.order_id) as no_of_order
 from `TargetSQL.payments` a
  join `TargetSQL.orders` o
   on a.order_id = o.order_id
    where a.payment_installments > 0
    group by 1
     order by 1 ;
```

Snapshot

```sql
select payment_installments,
    count(distinct o.order_id) as no_of_order
from `TargetSQL.payments` a
join `TargetSQL.orders` o
on a.order_id = o.order_id
    where a.payment_installments > 0
    group by 1
    order by 1 ;
```

This query will process 7.39 MB when run.

Processing location: US

Query results

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

Row	payment_installment	no_of_order
1	1	49060
2	2	12389
3	3	10443
4	4	7088
5	5	5234
6	6	3916

Actionable Insights:
Orders with varying numbers of payment installments indicate customer preference for flexible payment options, with certain installment counts being more popular.

Recommendations:
Promote and possibly expand flexible installment plans that attract more customers, and tailor marketing campaigns to highlight these options to increase order volume and customer satisfaction.