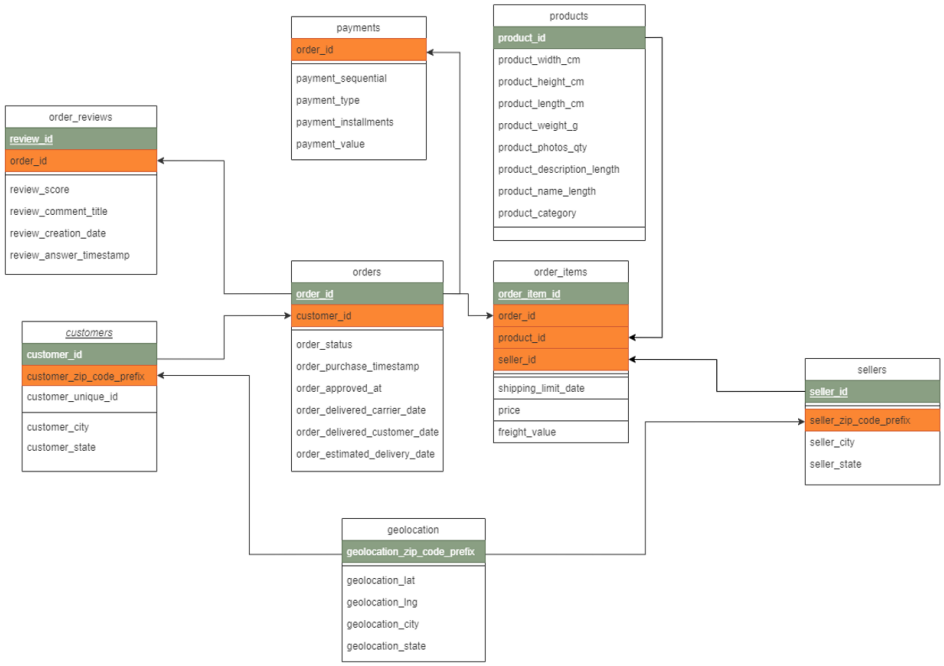
Target Schema:



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

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

   Query:

```
select
column_name,
data_type
from `scaler-dsml-sql-402614.Target.INFORMATION_SCHEMA.COLUMNS`
where table_name = 'customers';
```

   Output:

Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | CHART PREVIEW | 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: The information obtained from this query is of Customer Table.
The output provides column names of the customer table and the type of data we can store in those columns. We can clearly observe majority of the fields are of STRING data type.

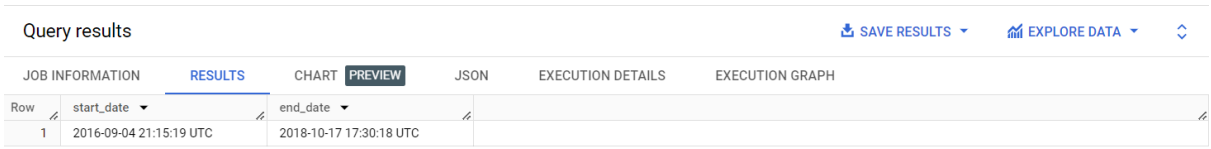
Recommendations: NA

→ Get the time range between which the orders were placed.

Query:

```
select
min(order_purchase_timestamp) as start_date,
max(order_purchase_timestamp) as end_date
from `Target.orders`;
```

Output:



Insights: The output of the query describes the date range between where the orders were placed. The date range clearly starts from September 4th 2016 to October 17th 2018.

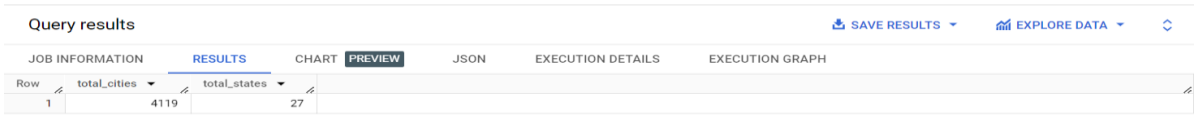
Recommendations: NA

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

Query:

```
select
count(distinct customer_city) as total_cities,
count(distinct customer_state) as total_states
from `Target.customers` c join `Target.orders` o
on o.customer_id = c.customer_id
```

Output:



Insights: There are total of 27 states and 4119 cities where the orders have been placed.

Recommendations: Introducing a franchise mechanism and expanding warehouse locations aligns with the goal of reaching a broader customer base and adapting to the growing demand. This strategy may strengthen Target's position in the market and capitalize on the opportunities presented by a widespread presence across the country.

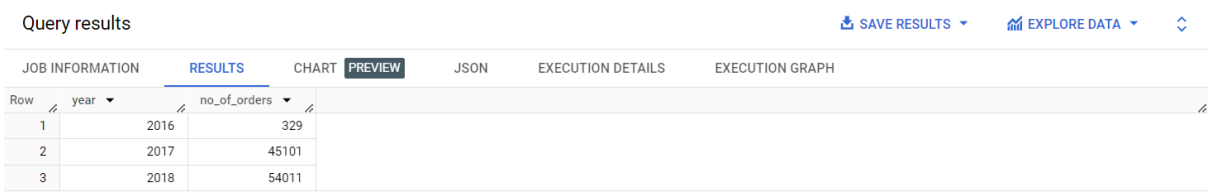
2. **In-depth Exploration:**

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

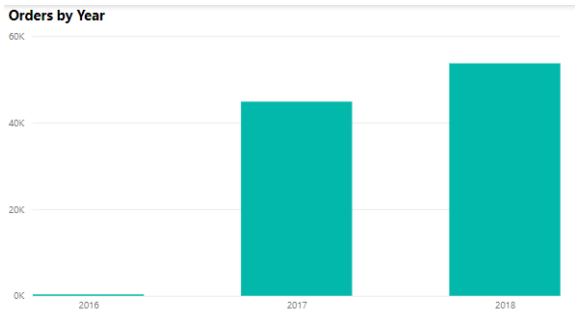
Query:

```
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
```

Output:



Insights:



Observing the data, it becomes evident that the number of orders exhibits a consistent upward trend, showing a year-over-year increase.

Recommendations: It's great that we're getting more orders every year. To keep this going, we should plan ahead. Since orders are going up consistently, it's smart to make our setup bigger. This way, we can handle the current demand and be

ready for even more orders in the future. The idea is to keep growing and make smart investments that match our increasing number of orders.

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

Query:

```
select
extract(year from order_purchase_timestamp) as year,
extract(month from order_purchase_timestamp) as month,
count(order_id) as no_of_orders
from `Target.orders`
group by year, month
order by year, month
```

Output:

Query results

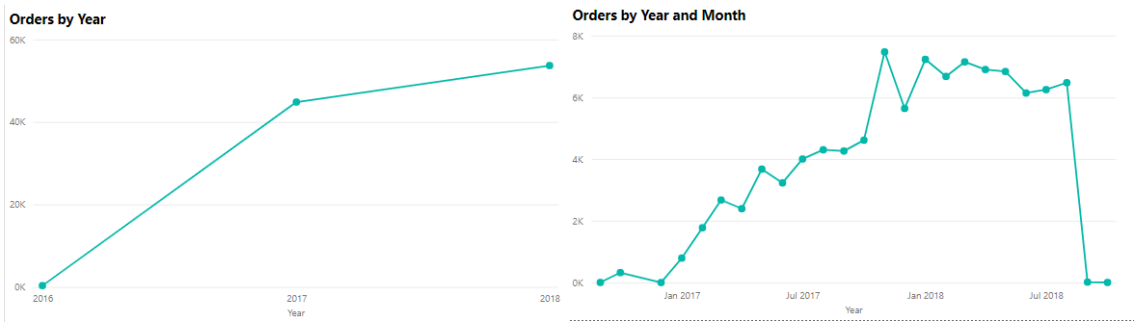
SAVE RESULTS | EXPLORE DATA

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

Row	year	month	no_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

Results per page: 50 | 1 – 25 of 25

Insights:



Looking at the data, it's clear that the most orders were in November 2017. However, something unusual happened after August 2018 orders dropped from 6477 to just 16. Before that, from the beginning until November 2017, orders were steadily going up. But starting January 2018, there was a consistent decrease, and after that, orders didn't pick up again.

Recommendations: From January 2018, there was a consistent decrease, and after that, orders didn't pick up again. This suggests a change in how people are buying, and it might be a good idea to figure out why and make some changes in the business strategy.

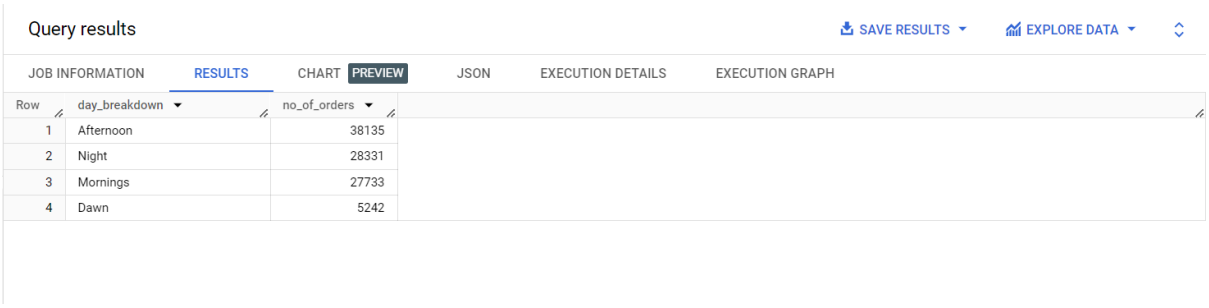
→During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)

- **0-6 hrs : Dawn**
- **7-12 hrs : Mornings**
- **13-18 hrs : Afternoon**
- **19-23 hrs : Night**

Query:

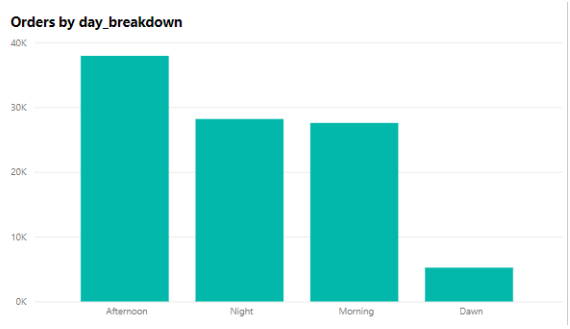
```
select
case
when extract(hour from order_purchase_timestamp) between 0 and 6 then 'Dawn'
when extract(hour from order_purchase_timestamp) between 7 and 12 then
'Mornings'
when extract(hour from order_purchase_timestamp) between 13 and 18 then
'Afternoon'
when extract(hour from order_purchase_timestamp) between 19 and 23 then 'Night'
end as day_breakdown,
count(order_id) as no_of_orders
from `Target.orders`
group by day_breakdown
order by no_of_orders desc
```

Output:



Row	day_breakdown	no_of_orders
1	Afternoon	38135
2	Night	28331
3	Mornings	27733
4	Dawn	5242

Insights:



Looking at the data, it's clear that most customers in Brazil like to place their orders in the afternoon.

Recommendations: Because more people prefer to make purchases in the afternoon, it's a smart move to ensure our servers can handle the extra traffic at that

time. We should also have more staff available to handle the increased number of sales during the afternoon.

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

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

Query:

```
select
c.customer_state,
extract(year from order_purchase_timestamp) as year,
extract(month from order_purchase_timestamp) as month,
count(order_id) as no_of_orders
from `Target.orders` o join `Target.customers` c using(customer_id)
group by c.customer_state, year, month
order by year,month,no_of_orders
```

Output:

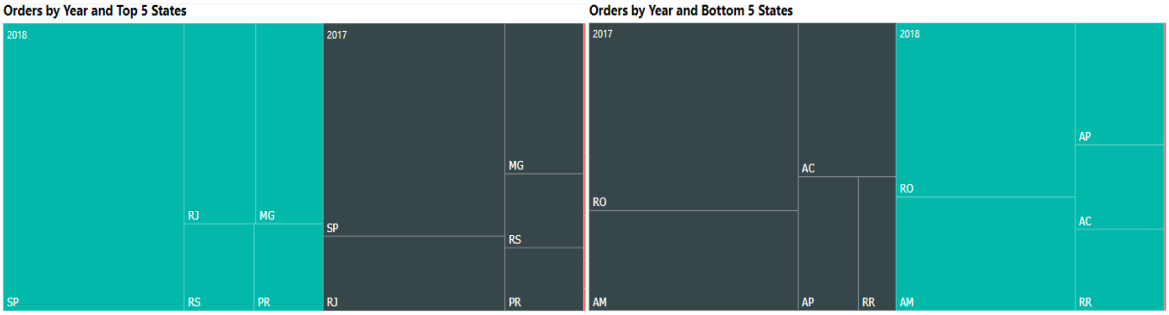
Query results

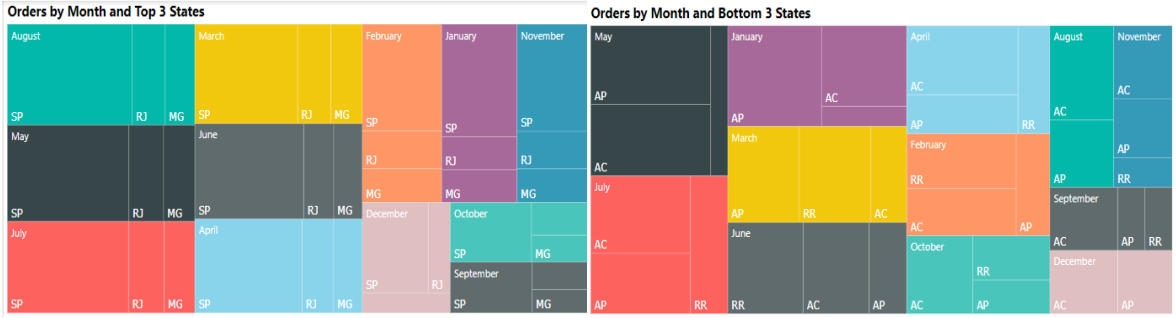
SAVE RESULTS | EXPLORE DATA

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

Row	customer_state	year	month	no_of_orders
1	RR	2016	9	1
2	RS	2016	9	1
3	SP	2016	9	2
4	RR	2016	10	1
5	PB	2016	10	1
6	PI	2016	10	1
7	AL	2016	10	2
8	MT	2016	10	3
9	SE	2016	10	3

Insights:





From the above visualizations we can clearly observe the orders are very good in the states such as SP, RJ, MG, RS, PR and the orders are very poor in the states such as RO, AM, AC, AP, RR.

Recommendations: In places where lots of people are ordering, we should make our warehouses better and improve how we deliver orders quickly. But in areas where fewer people are ordering, we need to figure out why? To attract more customers in those places, we might offer extra discounts, run special campaigns, or plan some exciting promotions.

→How are the customers distributed across all the states?

Query:

```
select
customer_state,
count(customer_id) as no_of_customers
from `Target.customers`
group by customer_state
order by no_of_customers desc
```

Output:

Query results

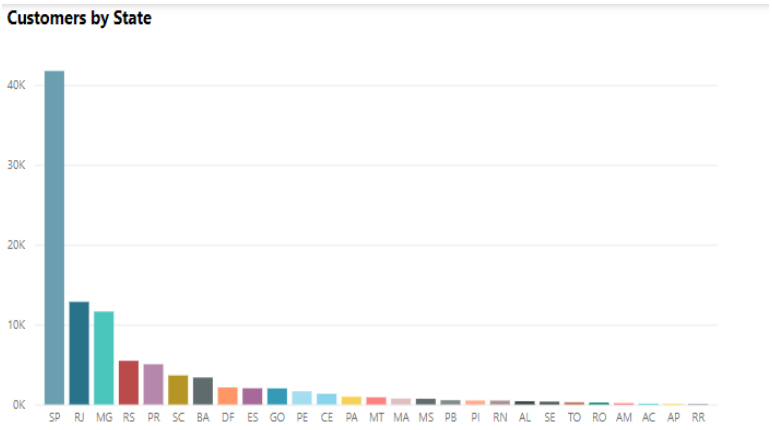
SAVE RESULTS | EXPLORE DATA

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

Row	customer_state	no_of_customers
1	SP	41746
2	RJ	12852
3	MG	11635
4	RS	5466
5	PR	5045
6	SC	3637

Results per page: 50 1 – 27 of 27

Insights:



Looking at the above insight, it's clear that there are a lot of customers in SP state. However, the number gradually goes down in the following states like RJ, MG, RS, and PR.

Recommendations: Customers in the states with consistently high in numbers can be seen as loyal supporters. To show our appreciation, we could consider offering them special gifts or incentives. On the other hand, for the bottom 5 states with significantly fewer numbers, it's important to analyse the reasons behind this decline. Taking suitable actions based on this analysis will help us address any issues and improve the situation in those regions.

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

→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:
Year on Year Approach:

```
with cte1 as
(
SELECT
SUM(
CASE WHEN EXTRACT(YEAR FROM order_purchase_timestamp) = 2018 AND
EXTRACT(MONTH FROM order_purchase_timestamp) BETWEEN 1 AND 8 THEN
payment_value
ELSE 0
END) as sales_2018,
SUM(
CASE WHEN EXTRACT(YEAR FROM order_purchase_timestamp) = 2017 AND
EXTRACT(MONTH FROM order_purchase_timestamp) BETWEEN 1 AND 8 THEN
payment_value
ELSE 0
END) as sales_2017
FROM `Target.orders` JOIN `Target.payments` USING(order_id)
)

select
round(sales_2017,2) as sales_2017,
round(sales_2018,2) as sales_2018,
concat(round(((sales_2018-sales_2017)/sales_2017)*100,2),'%') as
sales_percentage from cte1
```

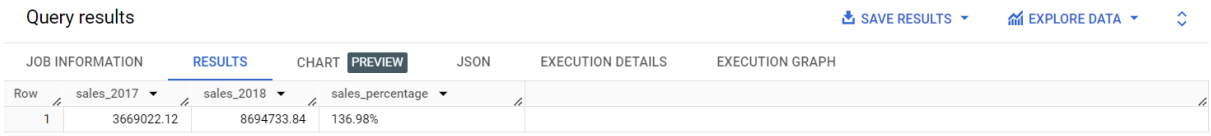
Month on Month Approach:

```
with cte1 as
(
SELECT
```

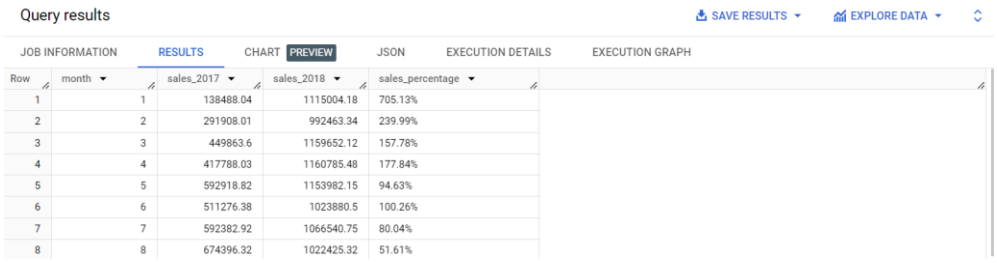
```
extract(month from order_purchase_timestamp) as month,
round(sum(payment_value),2) as sales_2017
FROM `Target.orders` JOIN `Target.payments` USING(order_id)
where extract(year from order_purchase_timestamp) = 2017
and extract(month from order_purchase_timestamp) between 1 and 8
group by month
order by month
),
cte2 as
(
SELECT
extract(month from order_purchase_timestamp) as month,
round(sum(payment_value),2) as sales_2018
FROM `Target.orders` JOIN `Target.payments` USING(order_id)
where extract(year from order_purchase_timestamp) = 2018
and extract(month from order_purchase_timestamp) between 1 and 8
group by month
order by month
)

select
month,
sales_2017,
sales_2018,
round(((sales_2018-sales_2017)/sales_2017)*100,2) as sales_percentage
from cte1 join cte2 using(month)
order by month
```

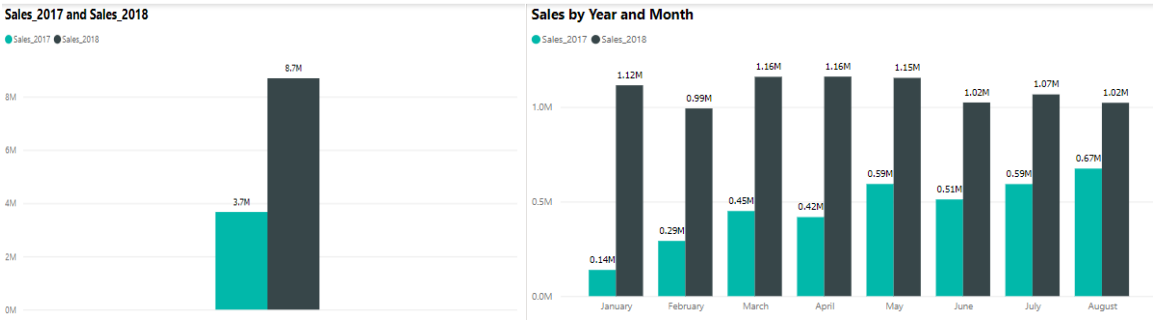
Output-1:



Output-2:



Insights:



Sales in 2018 showed a big jump compared to 2017. When we look at the sales each month in these two years, the total money earned went up a lot. The sales were almost doubled when compared with previous year sales but the biggest increase happened in January 2018, almost seven times more than what we made in January the previous year.

Recommendations: It's awesome to see the sales going up so much. Now, it's a good idea to hire more people, make our setup bigger, and even open new stores. This way, we can grow the business and be ready for the new customers coming in.

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

Query:

```
select
c.customer_state,
round(avg(price),2) as avg_order_price,
round(sum(price),2) as total_order_price,
from `Target.orders` o join `Target.customers` c using(customer_id)
join `Target.order_items` using(order_id)
group by c.customer_state
order by customer_state
```

Output:

Query results

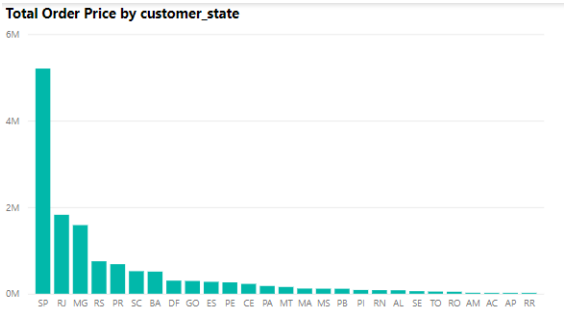
SAVE RESULTS | EXPLORE DATA

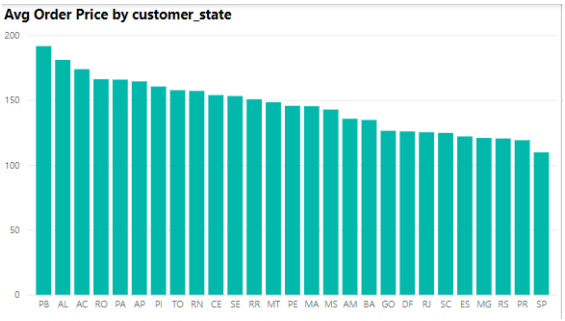
JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	customer_state	avg_order_price	total_order_price
1	AC	173.73	15982.95
2	AL	180.89	80314.81
3	AM	135.5	22356.84
4	AP	164.32	13474.3
5	BA	134.6	511349.99

Results per page: 50 1 – 27 of 27

Insights:





Looking at the insight, it's clear that SP state has the highest order price, followed by RJ, MG, RS, and PR. However, surprisingly, when we look at the average payment for each order, SP has the lowest. This suggests that even though there are a lot of orders in SP, each order doesn't have a high order price. This could be causing an increase in the freight cost for SP state. The similar is the case for RJ, MG, RS and PR states as well.

Recommendations: To make people spend more on each order, we can have special deals like "buy one, get one free." This way, we can pair a popular item with one that's not selling as much. This should encourage customers to spend more, and it helps us make better profits.

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

Query:

```
select
c.customer_state as state,
round(avg(freight_value),2) as avg_freight_value,
round(sum(freight_value),2) as total_freight_value
from `Target.order_items` oi join `Target.orders` o using(order_id)
join `Target.customers` c  using(customer_id)
group by c.customer_state
order by c.customer_state
```

Output:

Query results

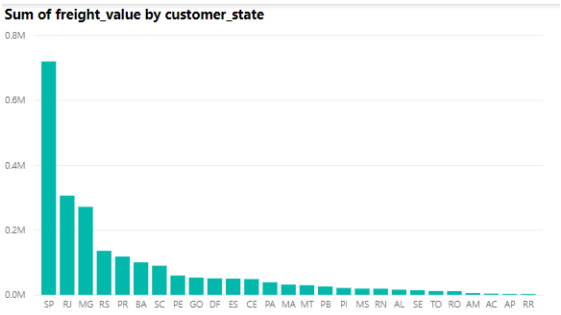
SAVE RESULTS | EXPLORE DATA

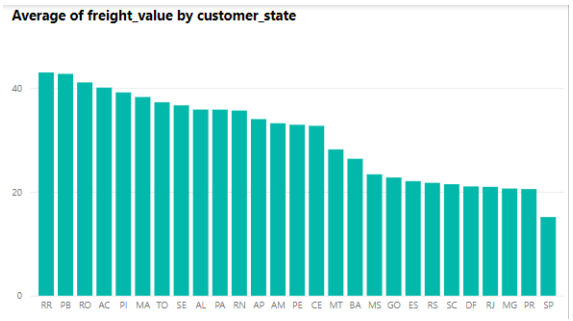
JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	state	avg_freight_value	total_freight_value
1	AC	40.07	3686.75
2	AL	35.84	15914.59
3	AM	33.21	5478.89
4	AP	34.01	2788.5
5	BA	26.36	100156.68

Results per page: 50 | 1 – 27 of 27

Insights:





The shipping costs for SP, RJ, MG, RS, and PR states are quite high based on the information we've gathered.

Recommendations: Since the shipping costs are high in those areas (SP, RJ, MG, RS, and PR), it's a good idea to set up franchises or warehouses there. These states are bringing in good sales for us, and by reducing the shipping costs, we can make bigger profits.

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

→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:

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

Query:

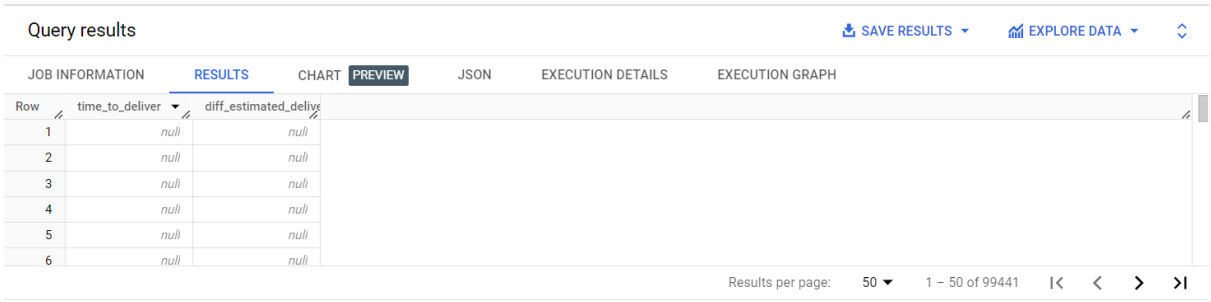
```
select
timestamp_diff(order_delivered_customer_date, order_purchase_timestamp, day) as
time_to_deliver,
timestamp_diff(order_estimated_delivery_date, order_delivered_customer_date,
day) as diff_estimated_delivery
from `Target.orders`
order by diff_estimated_delivery;
```

Fixed Query:

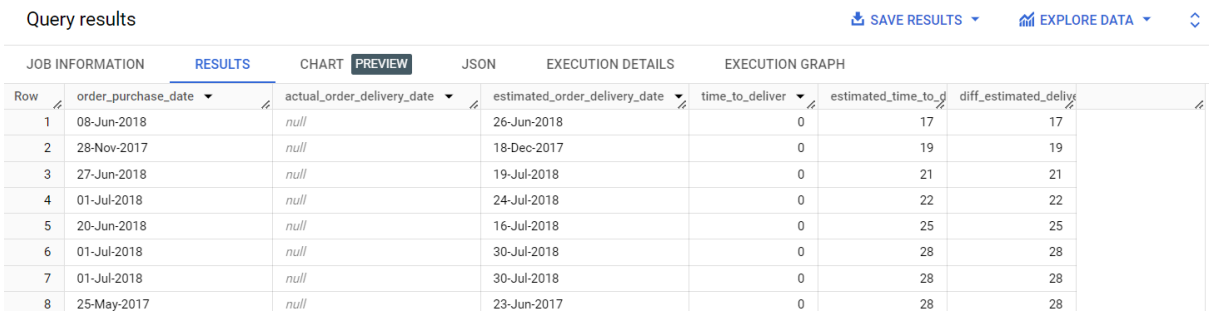
```
select
format_date('%d-%b-%Y', order_purchase_timestamp) as order_purchase_date,
format_date('%d-%b-%Y', order_delivered_customer_date) as
actual_order_delivery_date,
```

```
format_date('%d-%b-%Y', order_estimated_delivery_date) as
estimated_order_delivery_date,
ifnull(timestamp_diff(order_delivered_customer_date, order_purchase_timestamp,
day),0) as time_to_deliver,
timestamp_diff(order_estimated_delivery_date, order_purchase_timestamp, day) as
estimated_time_to_deliver,
ifnull(timestamp_diff(order_estimated_delivery_date,
order_delivered_customer_date, day),
timestamp_diff(order_estimated_delivery_date, order_purchase_timestamp, day)) as
diff_estimated_delivery,
from `Target.orders`
where order_status = 'delivered'
and order_delivered_customer_date is null
order by diff_estimated_delivery;
```

Output (Before Fixing):



Output (After Fixing)



Row	order_purchase_date	actual_order_delivery_date	estimated_order_delivery_date	time_to_deliver	estimated_time_to_d	diff_estimated_delive
1	08-Jun-2018	null	26-Jun-2018	0	17	17
2	28-Nov-2017	null	18-Dec-2017	0	19	19
3	27-Jun-2018	null	19-Jul-2018	0	21	21
4	01-Jul-2018	null	24-Jul-2018	0	22	22
5	20-Jun-2018	null	16-Jul-2018	0	25	25
6	01-Jul-2018	null	30-Jul-2018	0	28	28
7	01-Jul-2018	null	30-Jul-2018	0	28	28
8	25-May-2017	null	23-Jun-2017	0	28	28

Insights:

The output reveal orders with missing ‘time to deliver’ and significant differences between estimated and actual delivery times. Investigating these cases can help identify potential issues or delays in the delivery process.

Recommendations:

Identifying orders with missing delivery dates and knowing the reason behind why they are missing.

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

Query:

```
with cte1 as
(
	select
	customer_state,
	round(avg(freight_value),2) as avg_freight_value
	from
	`Target.order_items`  join `Target.orders`  using(order_id)
	join `Target.customers`  using(customer_id)
	group by customer_state
),
cte2 as
(
	select * from
	(
	select
	customer_state as state,
	avg_freight_value as highest_avg_freight_value,
	dense_rank() over(order by avg_freight_value desc) as rnk
	from cte1
	order by rnk
	) t
	where t.rnk <= 5
),
cte3 as
(
	select * from
	(
	select
	customer_state as state,
	avg_freight_value as lowest_avg_freight_value,
	dense_rank() over(order by avg_freight_value) as rnk
	from cte1
	order by rnk
	) t
	where t.rnk <= 5
)

select
cte2.state as states_with_highest,
cte2.highest_avg_freight_value,
cte3.state as states_with_lowest,
cte3.lowest_avg_freight_value
from cte2 join cte3
on cte2.rnk = cte3.rnk
order by cte2.rnk
```

Output:

Query results

SAVE RESULTS | EXPLORE DATA

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

Row	states_with_highest	highest_avg_freight_	states_with_lowest	lowest_avg_freight_v
1	RR	42.98	SP	15.15
2	PB	42.72	PR	20.53
3	RO	41.07	MG	20.63
4	AC	40.07	RJ	20.96
5	PI	39.15	DF	21.04

Insights:

From the above output its clearly evident that the top 5 states with highest average freight value include RR, PB, RO, AC, PI and the bottom 5 states with the lowest average freight value include SP, PR, MG, RJ, DF

Recommendations: For the top 5 states with high freight costs, let's try to make our delivery routes more efficient and, if possible, open local franchises to cut down on shipping expenses. This could help us save money.

For the bottom 5 states where freight costs are low, we should do some market research. If there's a chance, we can run special promotions or campaigns to boost sales in those states. Even though the shipping is cheaper, let's try to make more sales and increase our profits.

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

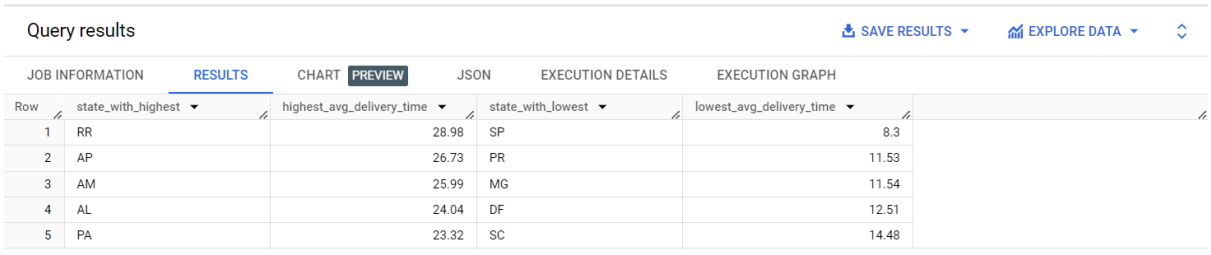
Query:

```
with cte1 as
(
        select
        c.customer_state,
        round(avg(date_diff(order_delivered_customer_date,
        order_purchase_timestamp, day)),2) as avg_delivery_time
        from
        `Target.orders` o join `Target.customers` c  using(customer_id)
        group by c.customer_state
),
cte2 as
(
```

```
        select * from
        (
        select
        customer_state as state,
        avg_delivery_time as highest_avg_delivery_time,
        dense_rank() over(order by avg_delivery_time desc) as rnk
        from cte1
        order by rnk
        ) t
        where t.rnk <= 5
),
cte3 as
(
        select * from
        (
        select
        customer_state as state,
        avg_delivery_time as lowest_avg_delivery_time,
        dense_rank() over(order by avg_delivery_time) as rnk
        from cte1
        order by rnk
        ) t
        where t.rnk <= 5
)

select
cte2.state as state_with_highest,
cte2.highest_avg_delivery_time,
cte3.state as state_with_lowest,
cte3.lowest_avg_delivery_time
from cte2 join cte3
on cte2.rnk = cte3.rnk
order by cte2.rnk
```

Output:



Row	state_with_highest	highest_avg_delivery_time	state_with_lowest	lowest_avg_delivery_time
1	RR	28.98	SP	8.3
2	AP	26.73	PR	11.53
3	AM	25.99	MG	11.54
4	AL	24.04	DF	12.51
5	PA	23.32	SC	14.48

Insights: From the data its clearly evident that the top 5 states with highest delivery time are RR, AP, AM, AL, PA and states with the lowest delivery time are SP, PR, MG, DF, SC

Recommendations: For top 5 states with highest delivery time we can evaluate logistics partners, optimize delivery routs, utilize local warehousing, communicate realistic delivery times, and also we can implement tracking systems. For bottom 5 states we can promote quick delivery and plan for an expansion.

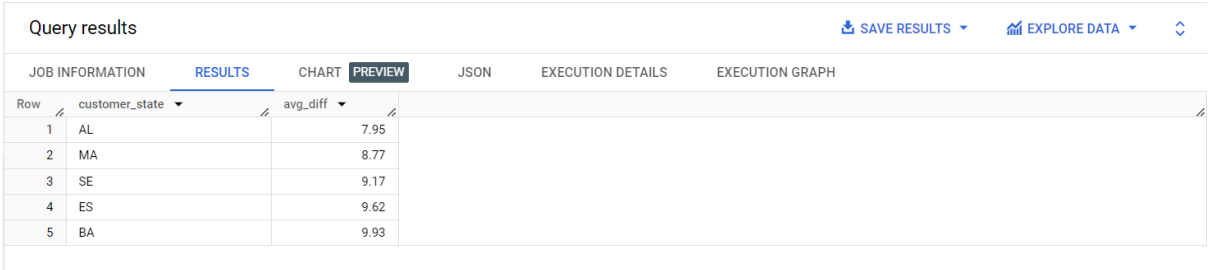
→Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery.
You can use the difference between the averages of actual & estimated delivery date to figure out how fast the delivery was for each state.

Query:

```
with cte1 as
(
  select
  customer_state,
  avg
  (
  timestamp_diff(order_estimated_delivery_date,
order_delivered_customer_date,day)
  ) as avg_diff
  from `Target.orders` join `Target.customers` using(customer_id)
  group by customer_state
),
cte2 as
(
  select
  *,
  dense_rank() over(order by avg_diff) as rnk from cte1
)

select
customer_state,
round(avg_diff,2) as avg_diff
from cte2
where rnk<=5
order by rnk
```

Output:



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

Insights: From the data its clearly evident that the top 5 states(AL, MA, SE, ES, BA) which have the actual delivery fast when compared to the estimated delivery.

Recommendations: In the top 5 states where deliveries are faster (AL, MA, SE, ES, BA), let's promote our quick delivery as a special feature. We should check if customers are happy and loyal. Using smart technology, we can keep a close eye on our stock and make sure it's always updated. This way, we can plan to grow our business in these places and build a strong reputation for our brand.

6. Analysis based on the payments:

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

Query:

```
select
extract(year from o.order_purchase_timestamp) as year,
extract(month from o.order_purchase_timestamp) as month,
payment_type,
count(o.order_id) as no_of_orders
from `Target.orders` o join `Target.payments` p using(order_id)
group by year, month, payment_type
order by year, month, no_of_orders
```

Output:

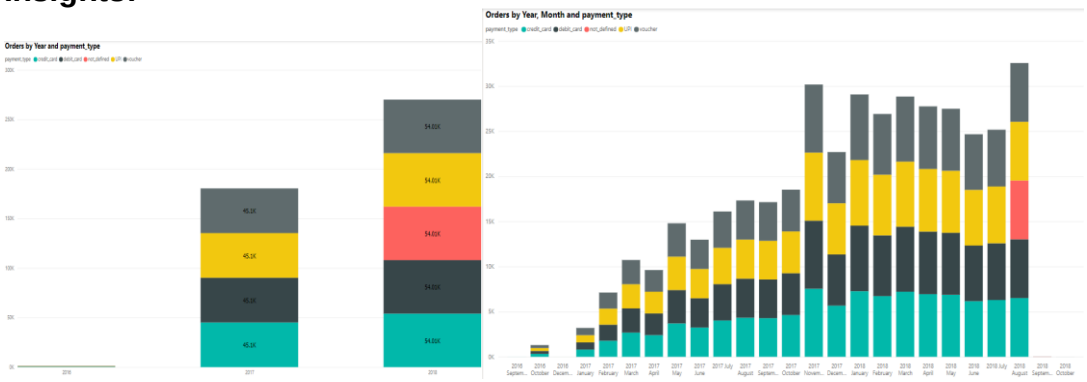
Query results

SAVE RESULTS | EXPLORE DATA

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

Row	year	month	payment_type	no_of_orders
1	2016	9	credit_card	3
2	2016	10	debit_card	2
3	2016	10	voucher	23
4	2016	10	UPI	63
5	2016	10	credit_card	254
6	2016	12	credit_card	1
7	2017	1	debit_card	9

Insights:



Looking at the data, it's clear that most customers prefer using vouchers for payments, followed by UPI, Debit cards, and Credit cards. Vouchers seem to be the most popular choice among our customers.

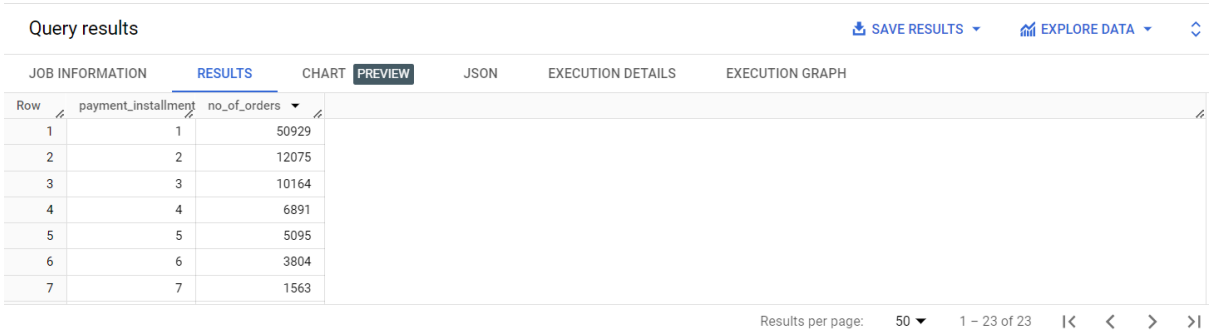
Recommendations: We should check which payment methods customers use the most and fix any issues with those options. Get feedback on how our payment system is working. Partner with payment providers to get better rates and use those methods more. Give extra discounts to customers who use the methods with lower commission to encourage them.

→Find the no. of orders placed on the basis of the payment instalments that have been paid.

Query:

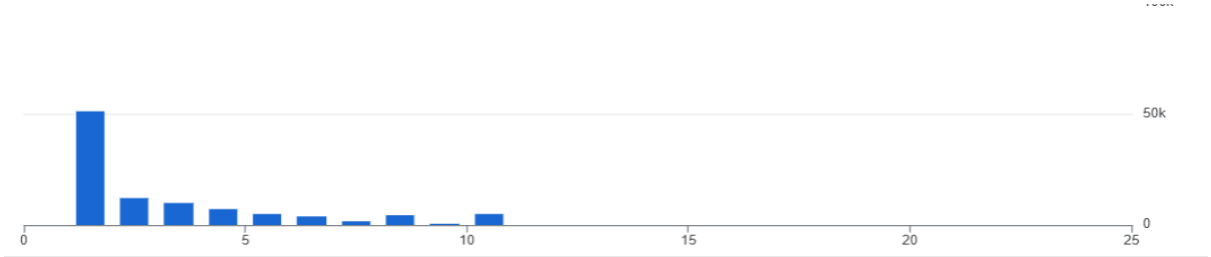
```
select
payment_installments,
count(o.order_id) as no_of_orders
from `Target.orders` o join `Target.payments` p using(order_id)
where order_status = 'delivered'
and payment_installments >= 1
group by payment_installments
order by payment_installments
```

Output:



Insights:

Orders by Payment Instalments where the order status is delivered



From the chart, it's clear that most customers prefer paying for their orders directly rather than opting for instalment plans. The majority of payments are made in one go, showing a preference for straightforward transactions.

Recommendations: Since we have a significant number of orders with instalment plans, it makes sense to promote this option based on our convenience. We can educate customers about the benefits of monthly instalments to attract as many customers as possible.