BUSINESS CASE: TARGET SQL

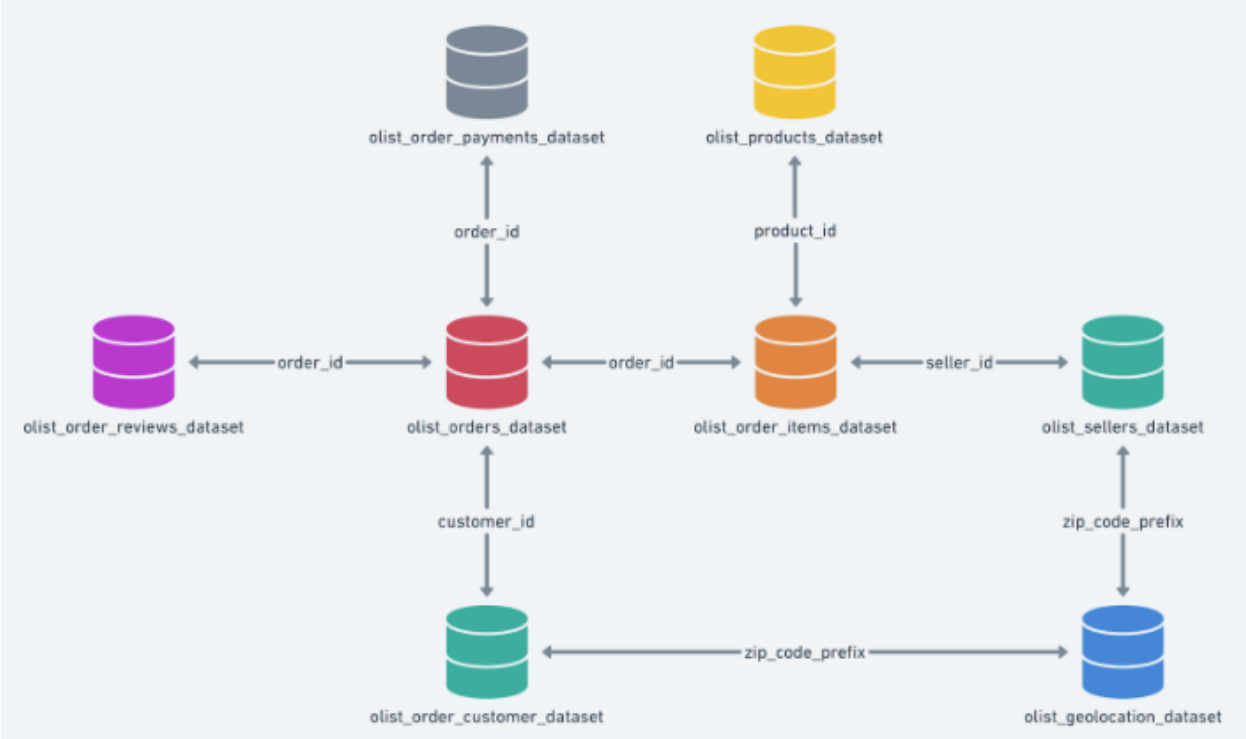
CONTEXT:

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.

DATA SCHEMA:



Problem Statement:

Assuming you are a data analyst/ scientist at Target, you have been assigned the task of analyzing the given dataset to extract valuable insights and provide actionable recommendations.

What does 'good' look like?

1. Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset:
    1. Data type of all columns in the "customers" table.

- Explore the type of data, find a relation, pattern in the data and is it ready for use or some sorting or to be brought in order

QUERY:

```
select column_name, data_type from `case`.INFORMATION_SCHEMA.COLUMNS where table_name
= "customers"
```

OUTPUT DATA:

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

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 & VIEWS:

The “customers” table has the information regarding customer id, customer unique id, customer zip codes, their city and state

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

-So, what basically asked is to check the first order and the last order in the given data set

QUERY:

```
select min(order_purchase_timestamp) as first_order, max(order_purchase_timestamp) as
most_recent from `case.orders`;
```

OUTPUT DATA:

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

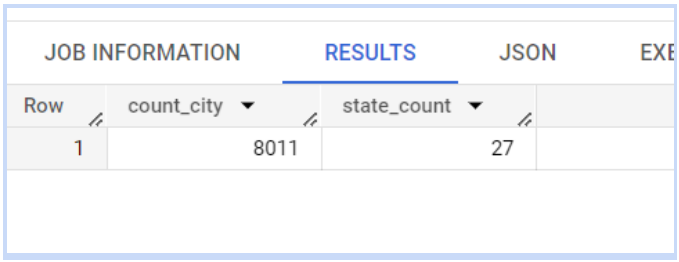
INSIGHTS & VIEWS:

3. Count the number of Cities and States in our dataset.

QUERY:

```
select count(distinct geolocation_city) as count_city, count(distinct
geolocation_state) as state_count from `case.geolocation`;
```

OUTPUT DATA:



INSIGHTS & VIEWS:

We have 8011 cities and 27 states.

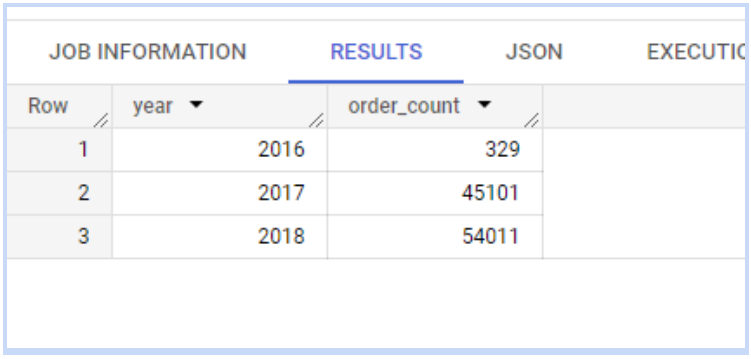
2. In-depth Exploration:

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

QUERY:

```
select extract(year from date(order_purchase_timestamp)) as year,count(distinct
order_id) as order_count from `case.orders`
group by 1
order by 1;
```

OUTPUT DATA:



JOB INFORMATION | RESULTS | JSON | EXECUTIC

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

INSIGHTS & VIEWS:

As we can see here the number of order count in 2016 is far less compared to succeeding years which implies that in 2017 and 2018 appropriate models created and proper results are obtained.

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

QUERY:

```
select extract(month from order_purchase_timestamp) as month, count(*) as order_count
from `case.orders`
group by 1
order by 1;
```

OUTPUT DATA:

Query results

JOB INFORMATION | RESULTS | JSON

Row	month	order_count
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

INSIGHTS & VIEWS:

Target made the most of the sales in the months of May, July & August. But in the months of September, October & December the sales were low as compared to others.

3. 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 cast(time(order_purchase_timestamp) as string) between '00:00:00' and
'06:59:59' then 'Dawn'
when cast(time(order_purchase_timestamp) as string) between '07:00:00' and '12:59:59'
then 'Morning'
when cast(time(order_purchase_timestamp) as string) between '13:00:00' and '18:59:59'
then 'Afternoon'
else 'Night' end as time_window, count(distinct order_id) from `case.orders`
group by 1;
```

OUTPUT DATA:

Row	time_window	f0_
1	Morning	27733
2	Dawn	5242
3	Afternoon	38135
4	Night	28331

INSIGHTS & VIEWS:

As we can see here maximum number of orders are placed in the Afternoon and the least orders are placed at Dawn as its the sleeping time in Brazil.

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

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

QUERY:

```
select customer_state,extract(year from date(order_purchase_timestamp)) as year,
extract(month from date(order_purchase_timestamp)) as month,
count(distinct order_id) as order_count from `case.customers` as a
inner join `case.orders` as b
on a.customer_id=b.customer_id
group by 1,2,3
order by 1,2,3;
```

OUTPUT DATA:

Query results

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

Row	customer_state	year	month	order_count
1	AC	2017	1	2
2	AC	2017	2	3
3	AC	2017	3	2
4	AC	2017	4	5
5	AC	2017	5	8
6	AC	2017	6	4
7	AC	2017	7	5
8	AC	2017	8	4
9	AC	2017	9	5
10	AC	2017	10	6
11	AC	2017	11	5
12	AC	2017	12	5
13	AC	2018	1	6

INSIGHTS & VIEWS:

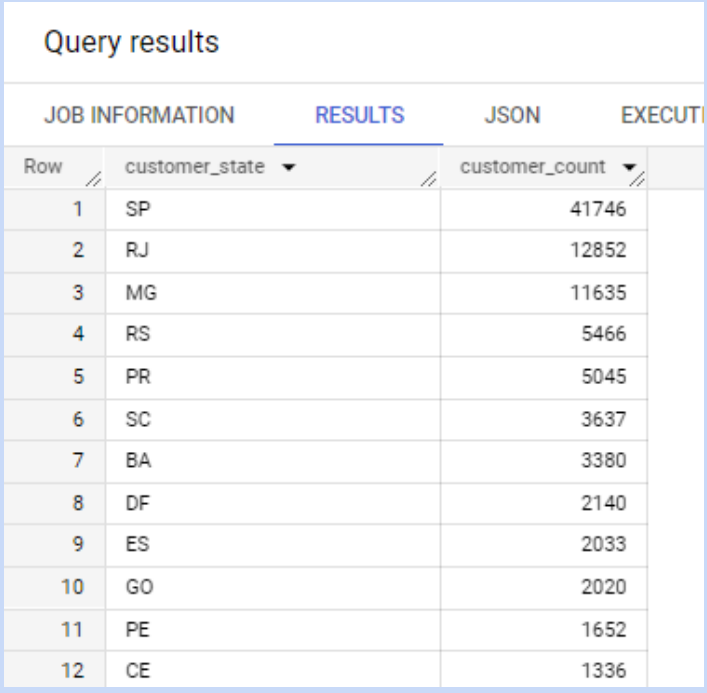
There is low count on order when seen on a monthly basis for some states. We need to bring offers, discounts and promotions to boost the order count for the respective states.

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

QUERY:

```
select customer_state,count(distinct customer_id) as customer_count from
`case.customers`
group by 1
order by 2 desc;
```

OUTPUT DATA:



Query results

JOB INFORMATION | RESULTS | JSON | EXECUTI

Row	customer_state	customer_count
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
12	CE	1336

INSIGHTS & VIEWS:

SP state has the highest customers and RR has the lowest.
More Discount and appealing schemes can be targeted to the lower state count customers to increase the number of orders.

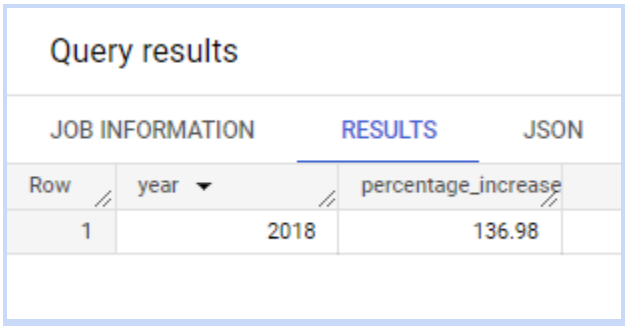
4. Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight and others.
    1. Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only).

You can use the "payment_value" column in the payments table to get the cost of orders.

QUERY:

```
select
table2.year,
round(((amount / lead(table2.amount) over(order by table2.year desc))-
1)*100,2) AS percentage_increase
from
(select
table1.year as year,
sum(table1.total_amount) as amount
from
(select
extract(year from o.order_purchase_timestamp) AS year,
extract(month from o.order_purchase_timestamp) AS month,
sum(payment_value) AS total_amount
from `case.orders` as o join `case.payments` AS p
on o.order_id=p.order_id
group by year, month
having year IN (2017, 2018) AND (month BETWEEN 1 AND 8)) AS table1
GROUP BY year) AS table2
order by year DESC
limit 1;
```

OUTPUT DATA:



INSIGHTS & VIEWS:

The growth rate is great for year 2018.

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

QUERY:

```
select s.seller_state,
round(sum(o.price),2) as total_price,
round(avg(o.price),2) as avg_price,
from `case.order_items` as o join `case.sellers` as s
on o.seller_id = s.seller_id
group by s.seller_state;
```

OUTPUT DATA:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | E

Row	seller_state	total_price	avg_price
1	SP	8753396.21	108.95
2	MG	1011564.74	114.6
3	PR	1261887.21	145.53
4	SC	632426.07	155.2
5	RS	378559.54	172.15
6	DF	97749.48	108.73
7	ES	47689.61	128.2
8	RJ	843984.22	175.17
9	GO	66399.21	127.69
10	PA	1238.0	154.75
11	RN	9992.6	178.44
12	CE	20240.64	215.33
13	BA	285561.56	444.11

INSIGHTS & VIEWS:

The total price of the orders as per the state and the average price.

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

QUERY:

```
select s.seller_state,
round(sum(o.freight_value),2) as total_freight_price,
round(avg(o.freight_value),2) as avg_freight_price,
from `case.order_items` as o join `case.sellers` as s
on o.seller_id = s.seller_id
group by s.seller_state;
```

OUTPUT DATA:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	seller_state	total_freight_price	avg_freight_price
1	SP	1482487.67	18.45
2	MG	212595.06	24.08
3	PR	197013.52	22.72
4	SC	106547.06	26.15
5	RS	57243.09	26.03
6	DF	18494.06	20.57
7	ES	12171.13	32.72
8	RJ	93829.9	19.47
9	GO	12565.5	24.16
10	PA	155.11	19.39

INSIGHTS & VIEWS:

The Freight price of a few states are very high, we should search for a better and cost beneficial delivery partner or negotiate on the existing terms.

5. 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.
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_estimated_delivery_date - order_delivered_customer_date

QUERY:

```
select order_id,
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 `case.orders`
```

OUTPUT DATA:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	order_id	time_to_deliver	diff_estimated_deliv
1	1950d777989f6a877539f5379...	30	-12
2	2c45c33d2f9cb8ff8b1c86cc28...	30	28
3	65d1e226dfaeb8cdc42f66542...	35	16
4	635c894d068ac37e6e03dc54e...	30	1
5	3b97562c3aee8bdedcb5c2e45...	32	0
6	68f47f50f04c4cb6774570cfde...	29	1
7	276e9ec344d3bf029ff83a161c...	43	-4
8	54e1a3c2b97fb0809da548a59...	40	-4
9	fd04fa4105ee8045f6a0139ca5...	37	-1
10	302bb8109d097a9fc6e9cefc5...	33	-5
11	66057d37308e787052a32828...	38	-6

INSIGHTS & VIEWS:

The negative figures indicate the excessive days taken after the given date to deliver, so we can check with the supply chain for the packaging, initiating and delivering of the products for a better result.

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

QUERY:

```
(select s.seller_state,
round(avg(freight_value)) as avg_value, 'high_avg' as category,
from `case.order_items` as o join `case.sellers` as s
on o.seller_id = s.seller_id
group by s.seller_state
order by avg_value desc
limit 5)
union all
(select s.seller_state,
round(avg(freight_value)) as avg_value, 'low_avg' as category
from `case.order_items` as o join `case.sellers` as s
on o.seller_id = s.seller_id
group by s.seller_state
order by avg_value asc
limit 5);
```

OUTPUT DATA:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTI

Row	seller_state	avg_value	category
1	RO	51.0	high_avg
2	CE	46.0	high_avg
3	PB	39.0	high_avg
4	PI	37.0	high_avg
5	ES	33.0	high_avg
6	SP	18.0	low_avg
7	PA	19.0	low_avg
8	RJ	19.0	low_avg
9	DF	21.0	low_avg
10	PR	23.0	low_avg

INSIGHTS & VIEWS:

We need to decrease the difference of the freight value average.

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

QUERY:

```
(select c.customer_state,
round(avg(timestamp_diff(order_delivered_customer_date,order_purchase_timestamp,
day)), 2) avg_time_deliver, 'highest_time' as avg_time_category,
from `case.orders` as o join `case.customers` as c
on o.customer_id = c.customer_id
group by c.customer_state
order by avg_time_deliver desc
limit 5)
union all
(select c.customer_state,
round(avg(timestamp_diff(order_delivered_customer_date,order_purchase_timestamp,
day)), 2) avg_time_deliver, 'lowest_time' as avg_time_category,
from `case.orders` as o join `case.customers` as c
on o.customer_id = c.customer_id
group by c.customer_state
order by avg_time_deliver asc
```

```
limit 5);
```

OUTPUT DATA:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTIO

Row	customer_state	avg_time_deliver	avg_time_category
1	RR	28.98	highest_time
2	AP	26.73	highest_time
3	AM	25.99	highest_time
4	AL	24.04	highest_time
5	PA	23.32	highest_time
6	SP	8.3	lowest_time
7	PR	11.53	lowest_time
8	MG	11.54	lowest_time
9	DF	12.51	lowest_time
10	SC	14.48	lowest_time

INSIGHTS & VIEWS:

The average time for delivery for a high average time is 22 days, so here we should work on the terms of our delivery partner for getting the days reduced.

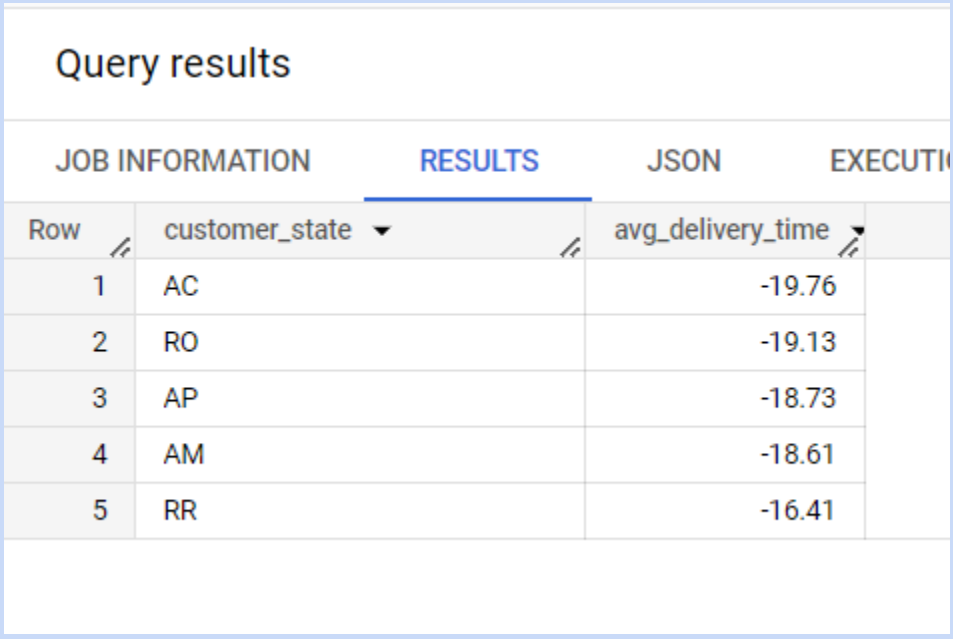
4. Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery.
   You can use the difference between the averages of actual & estimated delivery date to figure out how fast the delivery was for each state.

QUERY:

```
select c.customer_state,
round(avg(timestamp_diff(order_delivered_customer_date,order_estimated_delivery_date,d
ay)),2) as avg_delivery_time
from `case.orders` as o join `case.customers` as c
on o.customer_id = c.customer_id
group by c.customer_state
```

```
order by avg_delivery_time asc
limit 5
```

OUTPUT DATA:



Query results

JOB INFORMATION | RESULTS | JSON | EXECUTI

Row	customer_state	avg_delivery_time
1	AC	-19.76
2	RO	-19.13
3	AP	-18.73
4	AM	-18.61
5	RR	-16.41

INSIGHTS & VIEWS:
NA

6. Analysis based on the payments:
    1. Find the month on month no. of orders placed using different payment types.

QUERY:

```
select p.payment_type,
extract(month from o.order_purchase_timestamp) as months,
count(p.order_id) as order_count
from `case.payments` as p join `case.orders` as o
on p.order_id = o.order_id
group by 1, 2
order by 1, 2;
```

OUTPUT DATA:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	payment_type	months	order_count
1	UPI	1	1715
2	UPI	2	1723
3	UPI	3	1942
4	UPI	4	1783
5	UPI	5	2035
6	UPI	6	1807
7	UPI	7	2074
8	UPI	8	2077
9	UPI	9	903

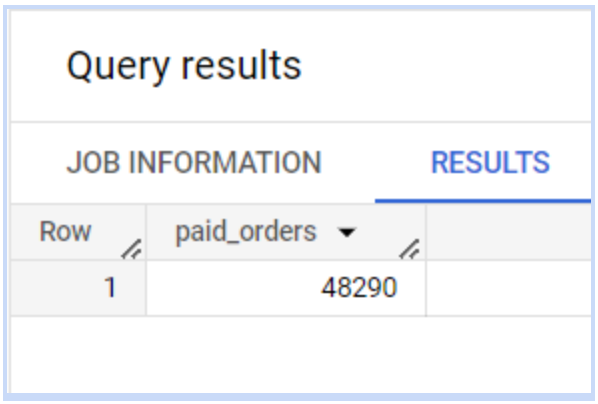
INSIGHTS & VIEWS:

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

QUERY:

```
with table1 as
(select
payment_installments, payment_sequential,
count (order_id) as order_count
from `case.payments`
group by payment_installments, payment_sequential)
select sum(order_count) as paid_orders
from table1
where payment_installments = payment_sequential
```

OUTPUT DATA:



INSIGHTS & VIEWS:

NA