Business Case: Target SQL

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 analysing 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.

```
/* Question 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.
Get the time range between which the orders were placed.
Count the number of Cities and States in our dataset.
*/

#1.1 Data type of all columns in the "customers" table
```

Query:

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

Query Screenshot:

Query results

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

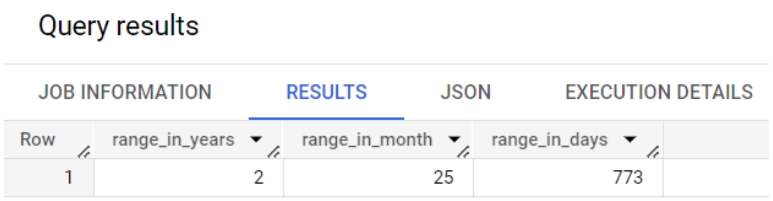
#1.2 Get the time range between which the orders were placed.

Query:

```
with cte as
(
select *,
extract ( date from order_purchase_timestamp) as order_date
from `target.orders`
)

select
date_diff(max(order_date), min(order_date), year) as range_in_years,
date_diff(max(order_date), min(order_date), month) as range_in_month,
date_diff(max(order_date), min(order_date), day) as range_in_days
from cte
```

Query Screenshot:

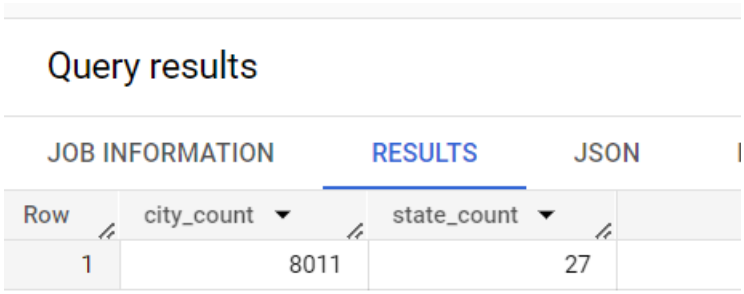


#1.3 Count the number of Cities and States in our dataset

Query:

```
select count (distinct(geolocation_city)) as city_count,
count(distinct(geolocation_state)) as state_count
from `target.geolocation`
```

Query Screenshot:



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

```
with cte as
(
select *,
extract ( date from order_purchase_timestamp) as order_date,
extract ( year from order_purchase_timestamp) as order_year,
extract ( month from order_purchase_timestamp) as order_month,
from `target.orders`
)


select order_year,
count(order_id) as total_orders
from cte
group by order_year
order by order_year
```

Query Screenshot:

Query results

JOB INFORMATION | RESULTS | JSON

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

Insights: growth rate of orders placed from 2107 to 2018 = (54011-45101)/45101 = 19.2%

#2.2Can we see some kind of monthly seasonality in terms of the no. of orders being placed?

Query:

```
with cte as
(
select *,
extract ( date from order_purchase_timestamp) as order_date,
extract ( year from order_purchase_timestamp) as order_year,
extract ( month from order_purchase_timestamp) as order_month,
from `target.orders`
)


select order_month, order_year,
count(order_id) as total_orders
```

```
from cte
group by order_month, order_year
order by order_year, order_month
```

Query Screenshot:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DE

Row	order_month	order_year	total_orders
1	9	2016	4
2	10	2016	324
3	12	2016	1
4	1	2017	800
5	2	2017	1780
6	3	2017	2682
7	4	2017	2404
8	5	2017	3700
9	6	2017	3245
10	7	2017	4026
11	8	2017	4331
12	9	2017	4285
13	10	2017	4631
14	11	2017	7544
15	12	2017	5673
16	1	2018	7269

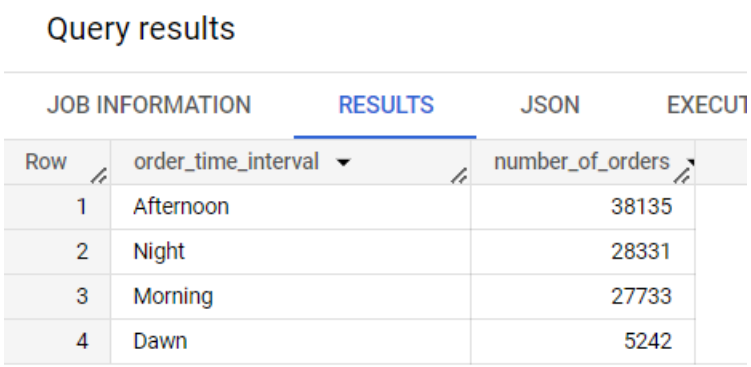
Insights:
-- We see a seasonal trend for Nov 2017 where there was Black Friday and there is a huge increase in the orders placed. No data for 2016, 2018 available
-- There is also a growth trend in Jan 2017 and Jan 2018 where New Years is experienced and also people may have preordered for the Carnival in Feb
-- There is also an increasing in orders in Q1 of 2018 during which FIFA World Cup was scheduled.

#2.3 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 EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 0 AND 6 THEN 'Dawn'
    WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 7 AND 12 THEN
'Morning'
    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 order_time_interval,
  COUNT(*) AS number_of_orders
FROM `target.orders`
GROUP BY order_time_interval
ORDER BY number_of_orders desc
```

Query Screenshot:



Row	order_time_interval	number_of_orders
1	Afternoon	38135
2	Night	28331
3	Morning	27733
4	Dawn	5242

Insights:

-- We see that the Brazillian customers order most in the afternoon followed by night time and the least is seen at dawn.
-- There is a chance of high sales if proper marketing strategy is applied in the afternoon and night times
-- While most of the advertisements can be pushed during the morning times, there is possibility of increasing the order count as well

#Evolution of E-commerce orders in the Brazil region:
#3.1 Get the month on month no. of orders placed in each state.

Query:

```
SELECT
EXTRACT(MONTH FROM o.order_purchase_timestamp) AS order_month,
c.customer_state,
COUNT(*) AS number_of_order
FROM `target.orders` AS o
JOIN `target.customers` AS c
ON o.customer_id = c.customer_id
GROUP BY order_month, c.customer_state
ORDER BY c.customer_state, order_month;
```

Query Screenshot:

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	order_month	customer_state	number_of_order
1	1	AC	8
2	2	AC	6
3	3	AC	4
4	4	AC	9
5	5	AC	10
6	6	AC	7
7	7	AC	9
8	8	AC	7
9	9	AC	5
10	10	AC	6
11	11	AC	5
12	12	AC	5
13	1	AL	39
14	2	AL	39
15	3	AL	40
16	4	AL	51

#3.2 How are the customers distributed across all the states?

Query:

```
SELECT customer_state, COUNT(DISTINCT customer_id) AS total_custmers
FROM `target.customers`
GROUP BY customer_state
ORDER BY total_custmers DESC;
```

Query Screenshot:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUT

Row	customer_state	total_custmers
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
13	PA	975
14	MT	907
15	MA	747
16	MS	715

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

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

Query:

```
with cte as
(
SELECT
  EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
  ROUND(SUM(p.payment_value),2) AS total_payment_value
FROM `target.payments` AS p
JOIN `target.orders` AS o
ON p.order_id = o.order_id
WHERE
  EXTRACT(YEAR FROM o.order_purchase_timestamp) IN (2017, 2018)
  AND EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8
GROUP BY year
ORDER BY year
)

select
year,
```

```
round(((tb.total_payment_value - tb.lead_value)*100/tb.lead_value),2) as
growth_rate
from
(
select
year, total_payment_value,
lag(total_payment_value) over ( order by cte.YEAR asc) as lead_value
from cte
order by year asc
) as tb
```

Query Screenshot:

Query results

JOB INFORMATION | RESULTS | JSON

Row	year	growth_rate
1	2017	null
2	2018	136.98

Insights:
-- we see a growth rate of approximately 137% from 2017 to 2018 for the months of January to August

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

Query:

```
select
c.customer_state,
round(sum(p.payment_value),2) as total_order_value,
round(avg(p.payment_value),2) as avg_order_value
from `target.orders` as o
join `target.payments` as p
on o.order_id = p.order_id
join `target.customers` as c
on o.customer_id = c.customer_id

group by c.customer_state
order by c.customer_state
```

Query Screenshot:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	total_order_value	avg_order_value
1	AC	19680.62	234.29
2	AL	96962.06	227.08
3	AM	27966.93	181.6
4	AP	16262.8	232.33
5	BA	616645.82	170.82
6	CE	279464.03	199.9
7	DF	355141.08	161.13
8	ES	325967.55	154.71
9	GO	350092.31	165.76
10	MA	152523.02	198.86
11	MG	1872257.26	154.71
12	MS	137534.84	186.87
13	MT	187029.29	195.23
14	PA	218295.85	215.92
15	PB	141545.72	248.33
16	PE	324850.44	187.99

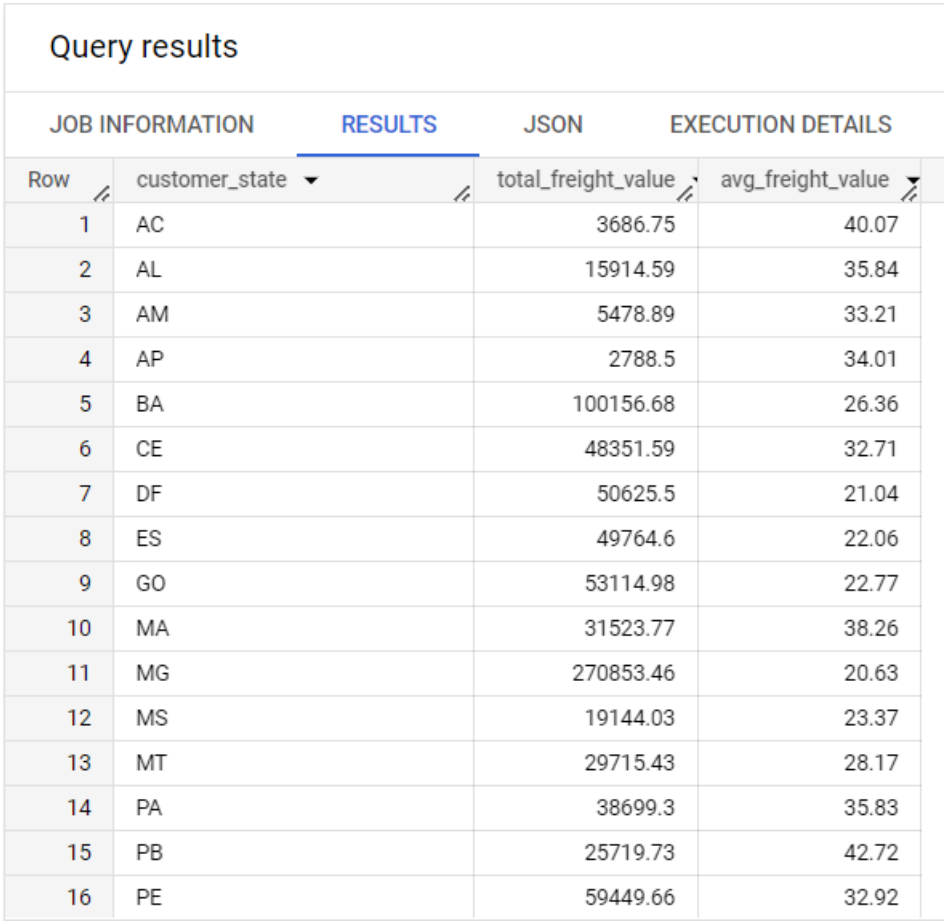
#4.3 Calculate the Total & Average value of order freight for each state

Query:

```
select
c.customer_state,
round(sum(p.freight_value),2) as total_freight_value,
round(avg(p.freight_value),2) as avg_freight_value
from `target.orders` as o
join `target.order_items` as p
on o.order_id = p.order_id
join `target.customers` as c
on o.customer_id = c.customer_id

group by c.customer_state
order by c.customer_state
```

Query Screenshot:



Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	total_freight_value	avg_freight_value
1	AC	3686.75	40.07
2	AL	15914.59	35.84
3	AM	5478.89	33.21
4	AP	2788.5	34.01
5	BA	100156.68	26.36
6	CE	48351.59	32.71
7	DF	50625.5	21.04
8	ES	49764.6	22.06
9	GO	53114.98	22.77
10	MA	31523.77	38.26
11	MG	270853.46	20.63
12	MS	19144.03	23.37
13	MT	29715.43	28.17
14	PA	38699.3	35.83
15	PB	25719.73	42.72
16	PE	59449.66	32.92

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

#5.1 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.

Query:

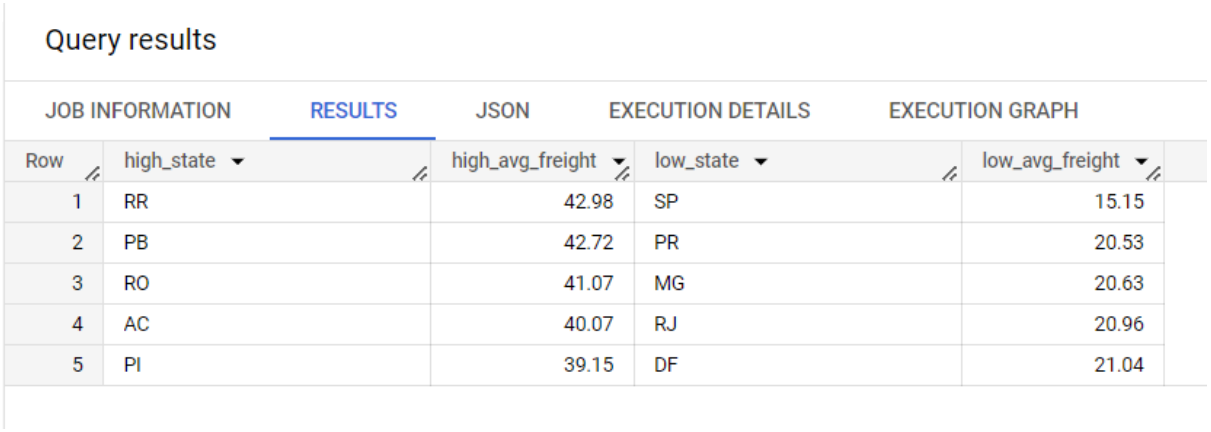
```
select
timestamp_diff(order_delivered_customer_date, order_purchase_timestamp, day) as
delivery_time,
timestamp_diff(order_estimated_delivery_date, order_delivered_customer_date, day)
as diff_estimated_delivery

from `target.orders`

where order_status = 'delivered' and order_delivered_customer_date is not null

order by order_purchase_timestamp asc
```

Query Screenshot:



Query results

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

Row	high_state	high_avg_freight	low_state	low_avg_freight
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

#5.2 Find out the top 5 states with the highest & lowest average freight value

Query:

```
select
high.customer_state as high_state,
high.average_freight_value as high_avg_freight,
low.customer_state as low_state,
low.average_freight_value as low_avg_freight
from
(
select
c.customer_state,
round(avg(p.freight_value),2) as average_freight_value,
row_number() over(order by (round(avg(p.freight_value),2))desc) as rowval1
from `target.orders` as o
join `target.order_items` as p
on o.order_id = p.order_id
join `target.customers` as c
on o.customer_id = c.customer_id

group by c.customer_state
order by average_freight_value desc
limit 5
) as high
join
(
select
c.customer_state,
round(avg(p.freight_value),2) as average_freight_value,
row_number() over(order by (round(avg(p.freight_value),2))) as rowval2
from `target.orders` as o
join `target.order_items` as p
on o.order_id = p.order_id
join `target.customers` as c
on o.customer_id = c.customer_id

group by c.customer_state
order by average_freight_value
```

```
limit 5
) as low

on high.rowval1 = low.rowval2
```

Query Screenshot:

Query results

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

Row	low_state	low_avg_delivery_time	high_state	high_avg_delivery_ti
1	SP	8.3	RR	28.98
2	PR	11.53	AP	26.73
3	MG	11.54	AM	25.99
4	DF	12.51	AL	24.04
5	SC	14.48	PA	23.32

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

Query:

```
with cte as
(
select
c.customer_state,
round(avg(t1.delivery_time),2) as avg_delivery_time

from
(
select *,
timestamp_diff(order_delivered_customer_date, order_purchase_timestamp, day) as
delivery_time,
from `target.orders`

where order_status = 'delivered' and order_delivered_customer_date is not null

order by order_purchase_timestamp asc
) as t1

join `target.customers` as c

on t1.customer_id = c.customer_id

group by c.customer_state

order by avg_delivery_time
```

```
)

select
c1.customer_state as low_state,
c1.avg_delivery_time as low_avg_delivery_time,
c2.customer_state as high_state,
c2.avg_delivery_time as high_avg_delivery_time
from
(
select *,
row_number() over (order by cte.avg_delivery_time desc) as rowval2
from cte
order by rowval2
) as c2

join

(
select *,
row_number() over (order by cte.avg_delivery_time) as rowval1
from cte
order by rowval1
) as c1

on c1.rowval1 = c2.rowval2

limit 5
```

Query Screenshot:

Query results

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

Row	low_state	low_avg_delivery_time	high_state	high_avg_delivery_ti
1	SP	8.3	RR	28.98
2	PR	11.53	AP	26.73
3	MG	11.54	AM	25.99
4	DF	12.51	AL	24.04
5	SC	14.48	PA	23.32

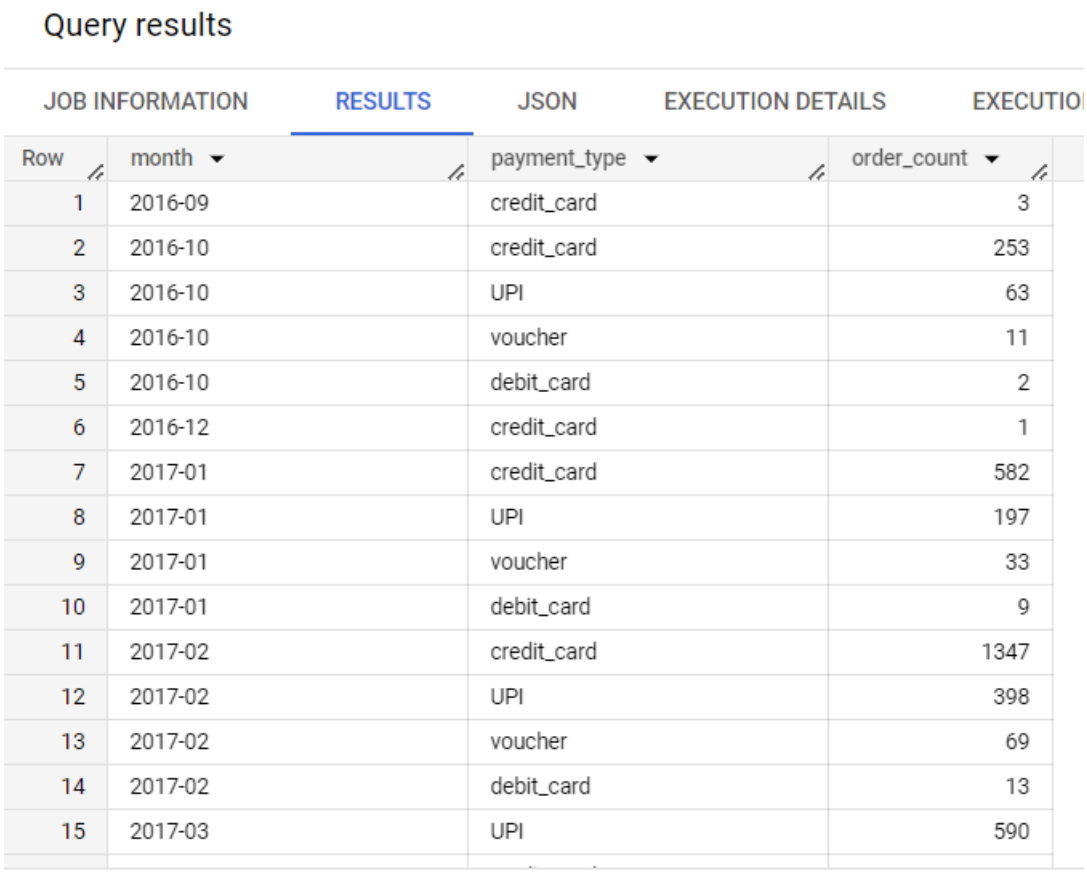
Analysis based on the payments:

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

Query:

```
SELECT
  FORMAT_TIMESTAMP('%Y-%m', o.order_purchase_timestamp) AS
month,
  p.payment_type,
  COUNT(DISTINCT o.order_id) AS order_count
FROM `target.orders` AS o
JOIN `target.payments` AS p
ON o.order_id = p.order_id
GROUP BY month, p.payment_type
ORDER BY month;
```

Query Screenshot:



Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTIO

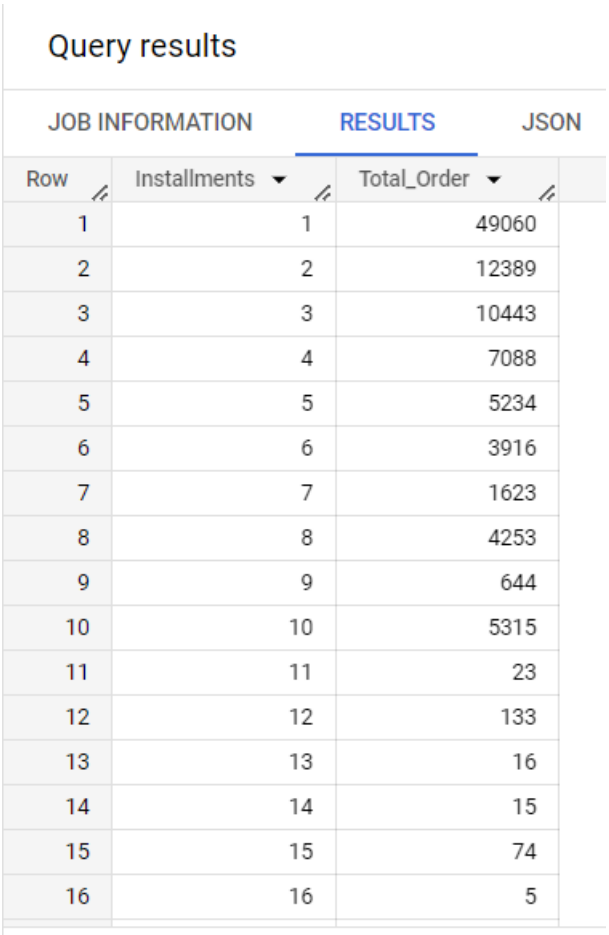
Row	month	payment_type	order_count
1	2016-09	credit_card	3
2	2016-10	credit_card	253
3	2016-10	UPI	63
4	2016-10	voucher	11
5	2016-10	debit_card	2
6	2016-12	credit_card	1
7	2017-01	credit_card	582
8	2017-01	UPI	197
9	2017-01	voucher	33
10	2017-01	debit_card	9
11	2017-02	credit_card	1347
12	2017-02	UPI	398
13	2017-02	voucher	69
14	2017-02	debit_card	13
15	2017-03	UPI	590

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

Query:

```
SELECT payment_installments, COUNT(DISTINCT order_id) AS
order_count
FROM `target.payments`
GROUP BY payment_installments
ORDER BY payment_installments;
```

Query Screenshot:



Query results

JOB INFORMATION | RESULTS | JSON

Row	Installments	Total_Order
1	1	49060
2	2	12389
3	3	10443
4	4	7088
5	5	5234
6	6	3916
7	7	1623
8	8	4253
9	9	644
10	10	5315
11	11	23
12	12	133
13	13	16
14	14	15
15	15	74
16	16	5