Target Business Case Study

- Kshitij Kapoor

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

customers QUERY SHARE COPY S

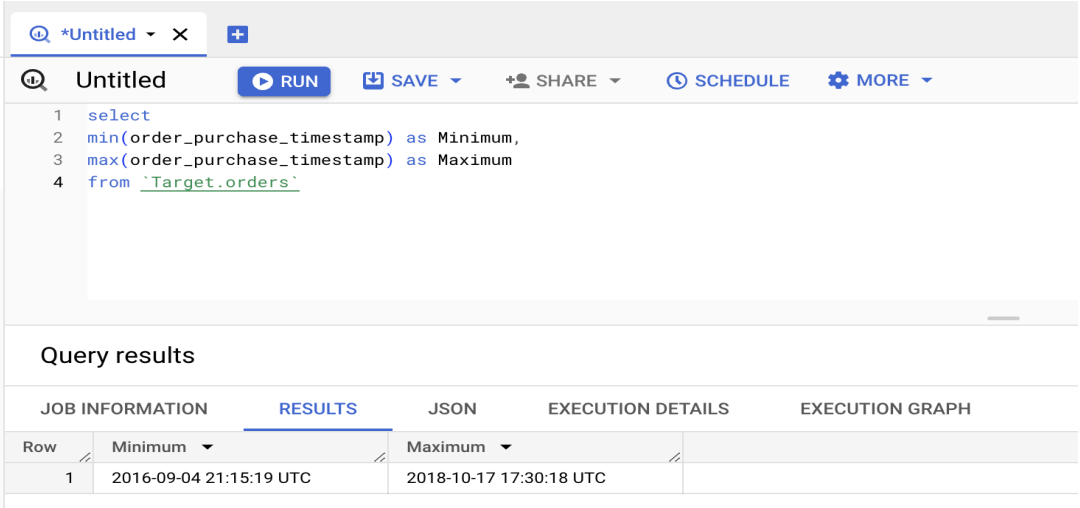
SCHEMA DETAILS PREVIEW LINEAGE

Filter Enter property name or value

Field name	Type	Mode	Key
customer_id	STRING	NULLABLE	
customer_unique_id	STRING	NULLABLE	
customer_zip_code_prefix	INTEGER	NULLABLE	
customer_city	STRING	NULLABLE	
customer_state	STRING	NULLABLE	

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

```
select
min(order_purchase_timestamp) as Minimum,
max(order_purchase_timestamp) as Maximum
from `Target.orders`
```



Insights:

The Dataset present with us showcases the orders which were placed from 4th September 2016 night till 17th October 2018 Evening.

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

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

Insights:

Data present in the Dataset is of customers from 27 States and 4119 Cities with in the states.

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

```
select
distinct(year) as Year,
count(order_id) over(partition by year order by year) as
Total_orders_per_year
from
(
select
*,
extract(year FROM order_purchase_timestamp) as year
from `Target.orders`
)
```

Untitled RUN SAVE SHARE SCHEDULE MORE

```
select
distinct(year) as Year,
count(order_id) over(partition by year order by year) as Total_orders_per_year
from
(
select
*,
extract(year FROM order_purchase_timestamp) as year
from `Target.orders`
)
```

Query results

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

Row	Year	Total_orders_per_yea
1	2016	329
2	2017	45101
3	2018	54011

Insights:

We can see that there is growth in the total no. of orders per year, however we can also see in 2016 either less amount pf data is available for the sale or limited months data is there.

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

```
select
distinct(year) as Year,
month,
count(order_id) over(partition by year,month order by year,month) as
Total_orders_per_year
from
(
select
*,
extract(year FROM order_purchase_timestamp) as year,
extract(month FROM order_purchase_timestamp) as month
from `Target.orders`
)
```

Untitled RUN SAVE SHARE SCHEDULE MORE

```
select
distinct(year) as Year,
month,
count(order_id) over(partition by year,month order by year,month) as Total_orders_per_year
from
(
select
*,
extract(year FROM order_purchase_timestamp) as year,
extract(month FROM order_purchase_timestamp) as month
from `Target.orders`
)
```

Query results

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

Row	Year	month	Total_orders_per_yea
1	2016	9	4
2	2016	10	324
3	2016	12	1
4	2017	1	800
5	2017	2	1780
6	2017	3	2682
7	2017	4	2404
8	2017	5	3700
9	2017	6	3245
10	2017	7	4026
11	2017	8	4331

Insights:

We can see there is a month on month growth from Jan 2017 onwards and the percentage increase in orders can be increased via marketing activities.

#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**

```sql
select
distinct(Time_of_the_day),
count(order_id) over(partition by Time_of_the_day) as No_of_orders
from
(
select
order_id,
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 24
then 'Night'
End as Time_of_the_day
from
`target-393507.Target.orders`
)
order by No_of_orders desc
```

Untitled RUN SAVE SHARE SCHEDULE MORE

```
select
distinct(Time_of_the_day),
count(order_id) over(partition by Time_of_the_day) as No_of_orders
from
(
select
order_id,
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 24 then 'Night'
End as Time_of_the_day
from
`target-393507.Target.orders`
)
order by No_of_orders desc
```

Query results

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

Row	Time_of_the_day	No_of_orders
1	Afternoon	38135
2	Night	28331
3	Morning	27733
4	Dawn	5242

Insights:

From the Data above we can tell that a good amount Brazilian customers order during the afternoons and then equally good amount customers order during night and morning hours, whereas significantly less amount of customers order during the dawn.

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

```
select
distinct(customer_state),
year,
month,
```

```
count(order_id) over(partition by customer_state,year,month order by
customer_state,year,month) as No_of_orders
from
(
(
select
order_id,
customer_id,
extract(year FROM order_purchase_timestamp) as year,
extract(month FROM order_purchase_timestamp) as month
from `Target.orders`
) as o
join
(
select
customer_id,
customer_state
from
`target-393507.Target.customers`
) as c
on c.customer_id = o.customer_id
)
```

Untitled

RUN SAVE SHARE SCHEDULE MORE

```
select
distinct(customer_state),
year,
month,
count(order_id) over(partition by customer_state,year,month order by customer_state,year,month) as No_of_orders
from
(
(
select
order_id,
customer_id,
extract(year FROM order_purchase_timestamp) as year,
extract(month FROM order_purchase_timestamp) as month
from `Target.orders`
) as o
join
(
select
customer_id,
```

Query results

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

Row	customer_state	year	month	No_of_orders
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

Insights:

For some states the month on month sales are increasing DF, and the rest more or less remain the same through out the year. More marketing and customer acquisition activities shall take place in the areas with same or decreasing sales.

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

```
select
distinct(customer_state) as States,
count(customer_id) over(partition by customer_state order by customer_state) as Customer_Per_State
from
```

`Target.customers`

Untitled RUN SAVE SHARE SCHEDULE MORE

```
select
distinct(customer_state) as States,
count(customer_id) over(partition by customer_state order by customer_state) as Customer_Per_State
from
`Target.customers`
```

Query results

JOB INFORMATION RESULTS JSON EXECUTION DETAILS EXECUTION GRAPH

Row	States	Customer_Per_State
1	AC	81
2	AL	413
3	AM	148
4	AP	68
5	BA	3380
6	CE	1336
7	DF	2140
8	ES	2033
9	GO	2020
10	MA	747
11	MG	11635
12	MS	715
13	MT	907
14	PA	975
15	PB	536
16	PE	1652

Insights:

The above data gives us an over view of which state got more density of customers and the areas with less density are to be focused more for increase in revenue.

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

```
with cte as(
select
distinct(year) as Year,
```

```
sum(payment_value) over(partition by year order by year) as sum1
from
(
select
order_id,
extract(year from order_purchase_timestamp) as year,
extract(month from order_purchase_timestamp)as month
from `target-393507.Target.orders`
) as o
join
(
select
order_id,
payment_value
from `target-393507.Target.payments`
) as p
on o.order_id = p.order_id
where month <= 8
)select
*,
ifnull(lag(sum1) over(order by year),sum1) as next,
((sum1-(ifnull(lag(sum1) over(order by
year),sum1)))/ifnull(lag(sum1) over(order by year),sum1))*100 as
percentage_increase_from_last_year
from cte
```

Untitled

RUN SAVE SHARE SCHEDULE MORE

```google_sql
with cte as(
select
distinct(year) as Year,
sum(payment_value) over(partition by year order by year) as sum1
from
(
select
order_id,
extract(year from order_purchase_timestamp) as year,
extract(month from order_purchase_timestamp)as month
from `target-393507.Target.orders`
) as o
join
(
select
order_id,
payment_value
from `target-393507.Target.payments`
) as p
on o.order_id = p.order_id
where month <= 8
)select
*,
ifnull(lag(sum1) over(order by year),sum1) as next,
((sum1-(ifnull(lag(sum1) over(order by year),sum1)))/ifnull(lag(sum1) over(order by year),sum1))*100 as percentage_increase_from_last_year
from cte
```

Query results SAVE RE

JOB INFORMATION RESULTS JSON EXECUTION DETAILS EXECUTION GRAPH

Row	Year	sum1	next	percentage_increase
1	2018	8694733.84	3669022.12	136.9768716466...
2	2017	3669022.12	3669022.12	0.0

Insights:

The data is only for january to august for both years and there rought 137% year on year increase in orders.

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

```google_sql
select
distinct(customer_state) as State,
round(sum(payment_value) over(partition by customer_state order by customer_state),2) as Total_Order_Value_of_Each_state,
round(avg(payment_value) over(partition by customer_state order by customer_state),2) as Avg_Order_Value_of_Each_state
from `target-393507.Target.orders` as o
left join `target-393507.Target.customers` as c on o.customer_id = c.customer_id
left join `target-393507.Target.payments` as p on o.order_id = p.order_id
```

Untitled

RUN SAVE SHARE SCHEDULE MORE

```
select
distinct(customer_state) as State,
round(sum(payment_value) over(partition by customer_state order by customer_state),2) as Total_Order_Value_of_Each_state,
round(avg(payment_value) over(partition by customer_state order by customer_state),2) as Avg_Order_Value_of_Each_state
from `target-393507.Target.orders` as o
left join `target-393507.Target.customers` as c on o.customer_id = c.customer_id
left join `target-393507.Target.payments` as p on o.order_id = p.order_id
```

Query results

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

Row	State	Total_Order_Value_of_Each_state	Avg_Order_Value_of_Each_state
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
17	PI	108523.97	207.11

Insights:

The above data gives an overview of total order value for each state and an average value of each order from each state.

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

```
select
distinct(customer_state) as State,
round(sum(freight_value) over(partition by customer_state order by customer_state),2) as Total_Order_Value_of_Each_state,
round(avg(freight_value) over(partition by customer_state order by customer_state),2) as Avg_Order_Value_of_Each_state
from `target-393507.Target.orders` as o
```

```
left join `target-393507.Target.customers` as c on o.customer_id =
c.customer_id
left join `target-393507.Target.order_items` as p on o.order_id =
p.order_id
```

Untitled

RUN SAVE SHARE SCHEDULE MORE

```
select
distinct(customer_state) as State,
round(sum(freight_value) over(partition by customer_state order by customer_state),2) as Total_Freight_Value_of_Each_state,
round(avg(freight_value) over(partition by customer_state order by customer_state),2) as Avg_Freight_Value_of_Each_state
from `target-393507.Target.orders` as o
left join `target-393507.Target.customers` as c on o.customer_id = c.customer_id
left join `target-393507.Target.order_items` as p on o.order_id = p.order_id
```

Query results

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

Row	State	Total_Freight_Value_	Avg_Freight_Value_o
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

Insights:

The above data gives an overview of total freight value shipped to each state and an average freight value shipped to that state.

#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.

```sql
select
order_id,
date_diff(order_delivered_customer_date, order_purchase_timestamp,
day) as time_to_deliver,
date_diff(order_estimated_delivery_date,
order_delivered_customer_date, day) as diff_estimated_delivery
from
(
select
order_id,
extract(date from order_delivered_customer_date) as
order_delivered_customer_date,
extract (date from order_purchase_timestamp) as
order_purchase_timestamp,
extract(date from order_estimated_delivery_date) as
order_estimated_delivery_date
from `target-393507.Target.orders`
where order_delivered_customer_date is not null
)
```

Untitled RUN SAVE SHARE SCHEDULE MORE

```
select
order_id,
date_diff(order_delivered_customer_date, order_purchase_timestamp, day) as time_to_deliver,
date_diff(order_estimated_delivery_date, order_delivered_customer_date, day) as diff_estimated_delivery
from
(
select
order_id,
extract(date from order_delivered_customer_date) as order_delivered_customer_date,
extract (date from order_purchase_timestamp) as order_purchase_timestamp,
extract(date from order_estimated_delivery_date) as order_estimated_delivery_date
from `target-393507.Target.orders`
where order_delivered_customer_date is not null
)
```

Query results

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

Row	order_id	time_to_deliver	diff_estimated_delivery
1	1950d777989f6a877539f5379...	30	-12
2	2c45c33d2f9cb8ff8b1c86cc28...	31	29
3	65d1e226dfaeb8cdc42f66542...	36	17
4	635c894d068ac37e6e03dc54e...	31	2
5	3b97562c3aee8bdedcb5c2e45...	33	1
6	68f47f50f04c4cb6774570cfde...	30	2
7	276e9ec344d3bf029ff83a161c...	44	-4
8	54e1a3c2b97fb0809da548a59...	41	-4
9	fd04fa4105ee8045f6a0139ca5...	37	-1
10	302bb8109d097a9fc6e9cefc5...	34	-5
11	66057d37308e787052a32828...	39	-6
12	19135c945c554eebfd7576c73...	36	-2
13	4493e45e7ca1084efcd38ddeb...	34	0

Insights:

The above data gives an overview of the time taken for each order in days to reach the customer, where as the column with difference in estimated delivery showcases the how much was the date difference when the customer received the order compared to the estimated delivery time. The negative values showcase the it was delivered before the estimated date and Supply chain is to managed for the values in positive.

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

```
with cte1 as(
select
distinct(customer_state) as State,
ceil(avg(freight_value) over(partition by customer_state order by
customer_state)) as Avg_Value_of_Each_state
```

```
from `target-393507.Target.orders` as o
left join `target-393507.Target.customers` as c on o.customer_id =
c.customer_id
left join `target-393507.Target.order_items` as p on o.order_id =
p.order_id
)(
select
*
from cte1
order by Avg_Value_of_Each_state desc
limit 5 )
union all
(select
*
from cte1
order by Avg_Value_of_Each_state asc
limit 5)
```

Untitled RUN SAVE SHARE SCHEDULE MORE

```
with cte1 as(
select
distinct(customer_state) as State,
ceil(avg(freight_value) over(partition by customer_state order by customer_state)) as Avg_Value_of_Each_state
from `target-393507.Target.orders` as o
left join `target-393507.Target.customers` as c on o.customer_id = c.customer_id
left join `target-393507.Target.order_items` as p on o.order_id = p.order_id
)(
select
*
from cte1
order by Avg_Value_of_Each_state desc
limit 5 )
union all
(select
*
from cte1
order by Avg_Value_of_Each_state asc
limit 5)
```

Query results

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

Row	State	Avg_Value_of_Each_
1	SP	16.0
2	MG	21.0
3	PR	21.0
4	RJ	21.0
5	RS	22.0
6	PB	43.0
7	RR	43.0
8	RO	42.0
9	AC	41.0
10	PI	40.0

Insights:

The first five states are the states with the lowest average freight value and the bottom 5 are the states with maximum average freight value.

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

```
with cte as(
select
distinct(customer_state),
Round(avg(days_to_deliver) over(partition by customer_state),2) as
Average_days
from
(
```

```
select
customer_id,
date_diff(order_delivered_customer_date, order_purchase_timestamp,
day) as days_to_deliver
from
(
select
customer_id,
extract(date from order_delivered_customer_date) as
order_delivered_customer_date,
extract (date from order_purchase_timestamp) as
order_purchase_timestamp
from `target-393507.Target.orders`
where order_delivered_customer_date is not null
)
) as o left join `target-393507.Target.customers` as c
on o.customer_id = c.customer_id
)
(select
*
from cte
order by Average_days asc
limit 5)
union all
(select
*
from cte
order by Average_days desc
limit 5)
```

Untitled

RUN SAVE SHARE SCHEDULE MORE

```google_sql
with cte as(
select
distinct(customer_state),
Round(avg(days_to_deliver) over(partition by customer_state),2) as Average_days
from
(
select
customer_id,
date_diff(order_delivered_customer_date, order_purchase_timestamp, day) as days_to_deliver
from
(
select
customer_id,
extract(date from order_delivered_customer_date) as order_delivered_customer_date,
extract (date from order_purchase_timestamp) as order_purchase_timestamp
from `target-393507.Target.orders`
where order_delivered_customer_date is not null
)
) as o left join `target-393507.Target.customers` as c
on o.customer_id = c.customer_id
)
(select
```

Query results

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

Row	customer_state	Average_days
1	SP	8.7
2	PR	11.94
3	MG	11.95
4	DF	12.9
5	SC	14.91
6	RR	29.34
7	AP	27.18
8	AM	26.36
9	AL	24.5
10	PA	23.73

Insights:

The top five states are the ones with lowest average delivery time and the bottom five states are the ones with highest average delivery time.

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

```
with cte as(select
order_id,
customer_id,
date_diff(order_estimated_delivery_date,
order_delivered_customer_date, day) as estimated_delivery_time
from
(
select
order_id,customer_id,
extract(date from order_delivered_customer_date) as
order_delivered_customer_date,
extract(date from order_estimated_delivery_date) as
order_estimated_delivery_date
from `target-393507.Target.orders`
where lower(order_status) like '%delivered%'
)as tbl
)
select
distinct(customer_state) as State,
round(avg(estimated_delivery_time) over(partition by customer_state
),2) as Average
from cte o left join `target-393507.Target.customers` as c
on o.customer_id = c.customer_id
order by average desc
limit 5
```

Untitled RUN SAVE SHARE SCHEDULE MORE

```google_sql
with cte as(select
order_id,
customer_id,
date_diff(order_estimated_delivery_date, order_delivered_customer_date, day) as estimated_delivery_time
from
(
select
order_id,customer_id,
extract(date from order_delivered_customer_date) as order_delivered_customer_date,
extract(date from order_estimated_delivery_date) as order_estimated_delivery_date
from `target-393507.Target.orders`
where lower(order_status) like '%delivered%'
)as tbl
)
select
distinct(customer_state) as State,
round(avg(estimated_delivery_time) over(partition by customer_state ),2) as Average
from cte o left join `target-393507.Target.customers` as c
on o.customer_id = c.customer_id
order by average desc
limit 5
```

Query results

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

Row	State	Average
1	AC	20.73
2	RO	20.1
3	AP	19.69
4	AM	19.57
5	RR	17.29

Insights:

These are the 5 states have the averag fastest delivery time compared to estimated delivery time.

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

```google_sql
select
distinct(payment_type),
year,
month,
count(order_id) over(partition by payment_type, year, month order by year, month)
from
```

```
(select
payment_type,
extract(year from order_purchase_timestamp) as year,
extract(month from order_purchase_timestamp)as month,
o.order_id
from
`target-393507.Target.orders` as o left join
`target-393507.Target.payments` p
on o.order_id = p.order_id)
```

Untitled RUN SAVE SHARE SCHEDULE MORE

```
select
distinct(payment_type),
year,
month,
count(order_id) over(partition by payment_type, year, month order by year, month) as No_of_orders
from
(select
payment_type,
extract(year from order_purchase_timestamp) as year,
extract(month from order_purchase_timestamp)as month,
o.order_id
from
`target-393507.Target.orders` as o left join `target-393507.Target.payments` p
on o.order_id = p.order_id)
where payment_type is not null
```

Query results

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

Row	payment_type	year	month	No_of_orders
1	UPI	2016	10	63
2	UPI	2017	1	197
3	UPI	2017	2	398
4	UPI	2017	3	590
5	UPI	2017	4	496
6	UPI	2017	5	772
7	UPI	2017	6	707
8	UPI	2017	7	845
9	UPI	2017	8	938
10	UPI	2017	9	903
11	UPI	2017	10	993
12	UPI	2017	11	1509
13	UPI	2017	12	1160

Insights:

The above data displays the count of orders placed month on month with different payment types.

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

```
select
distinct(payment_installments),
count(order_id) over(partition by payment_installments order by
payment_installments ) as No_of_orders
from
(
select
payment_installments,
o.order_id
from
`target-393507.Target.orders` as o left join
`target-393507.Target.payments` p
on o.order_id = p.order_id
)
```

Untitled

RUN SAVE SHARE SCHEDULE MORE

```
select
distinct(payment_installments),
count(order_id) over(partition by payment_installments order by payment_installments ) as No_of_orders
from
(
select
payment_installments,
o.order_id
from
`target-393507.Target.orders` as o left join `target-393507.Target.payments` p
on o.order_id = p.order_id
)
where payment_installments is not null
```

Query results

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

Row	payment_installment	No_of_orders
1	0	2
2	1	52546
3	2	12413
4	3	10461
5	4	7098
6	5	5239
7	6	3920
8	7	1626
9	8	4268
10	9	644
11	10	5328
12	11	23
13	12	133
14	13	16