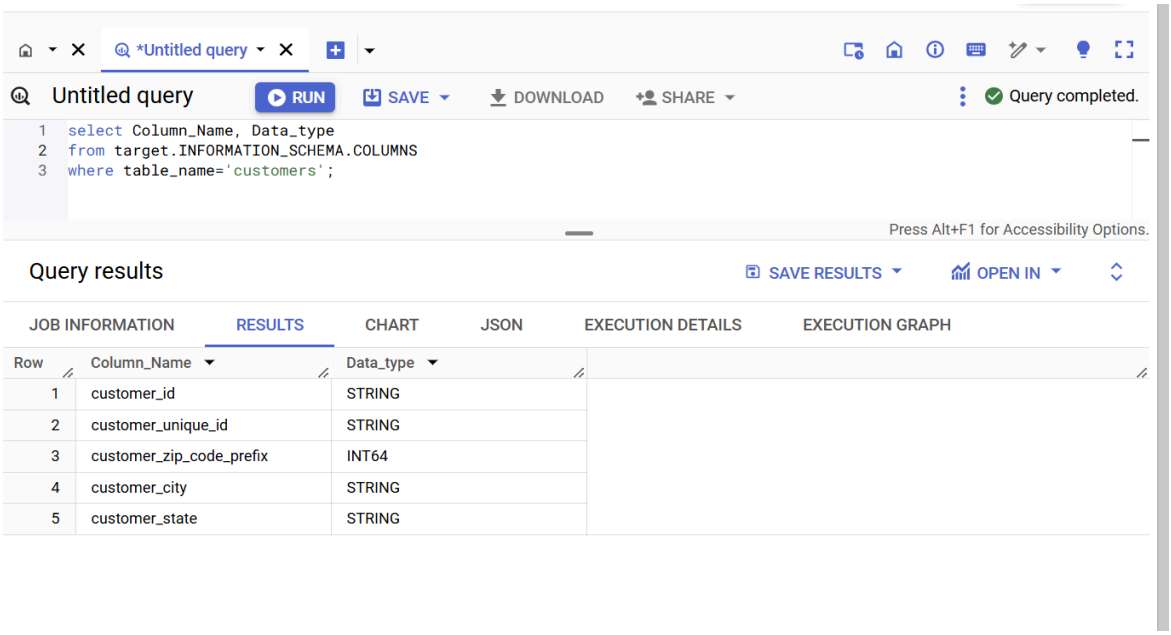
I. Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset.

A. Data type of all columns in the “customers” table.

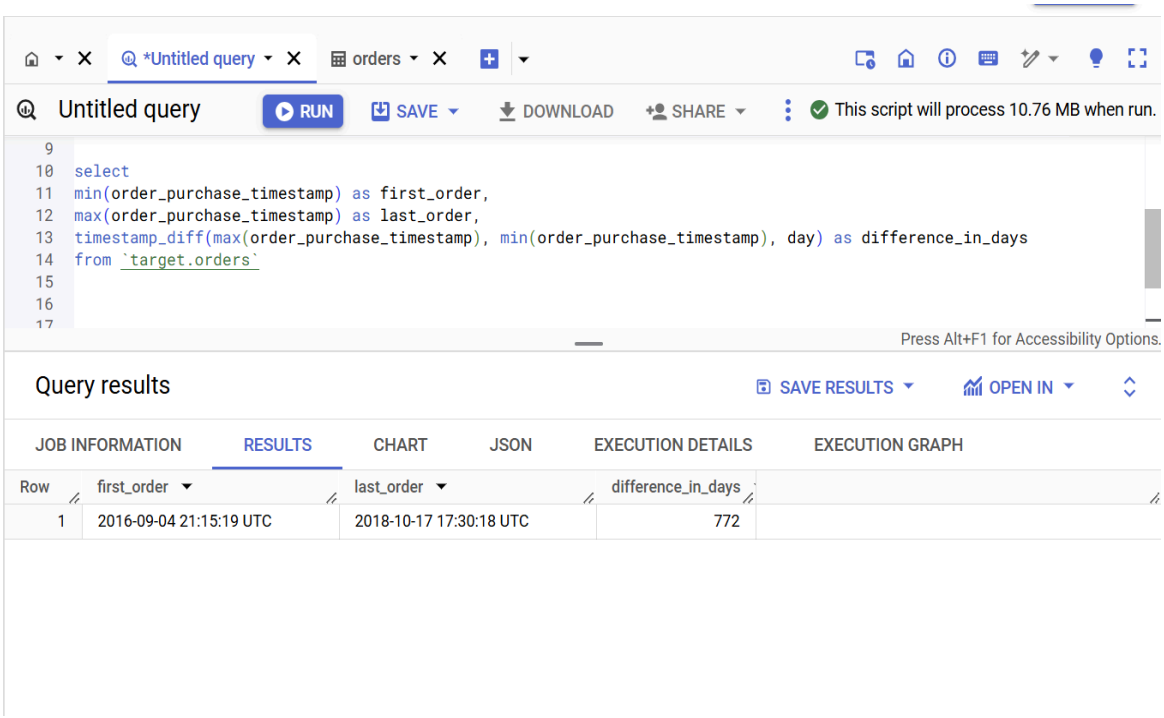
```
select Column_Name, Data_type
from target.INFORMATION_SCHEMA.COLUMNS
where table_name ='customers';
```



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

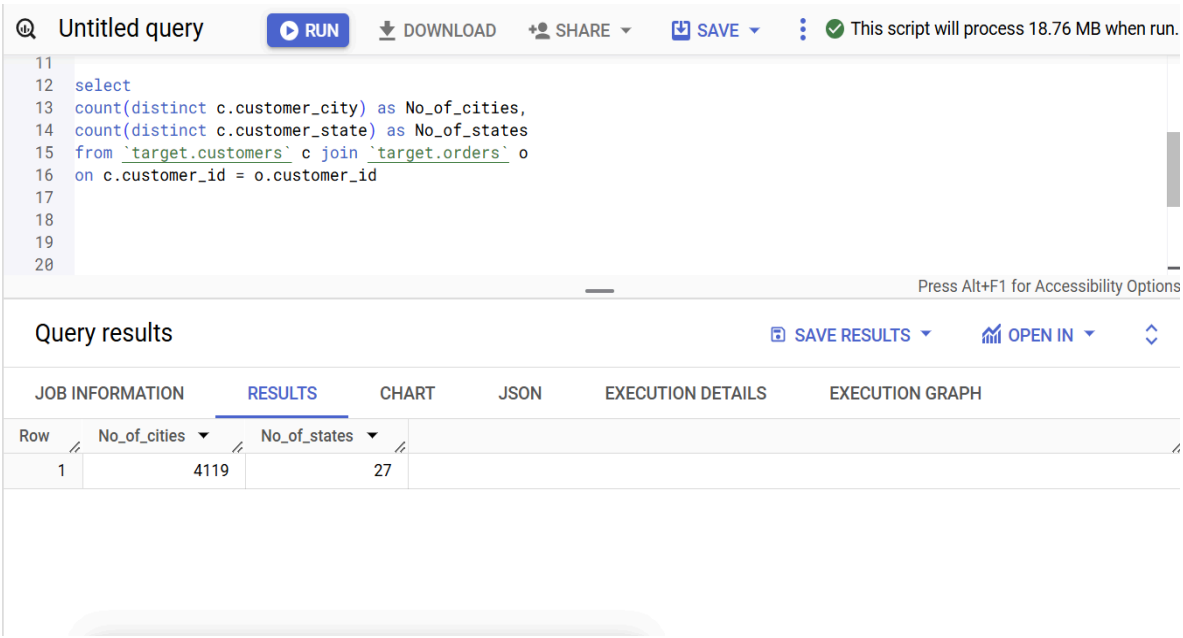
B. Get the time range between which the orders were placed.

```
select
min(order_purchase_timestamp) as first_order,
max(order_purchase_timestamp) as last_order,
timestamp_diff( max(order_purchase_timestamp),
min(order_purchase_timestamp), day) as difference_in_days
from `target.orders`;
```



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

```
select
count(distinct c.customer_city) as No_of_cities,
count(distinct c.customer_state) as No_of_states
from `target.customers` c join `target.orders` o
on c.customer_id = o.customer_id;
```



II. In-depth Exploration:

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

```
select

format_timestamp("%Y-%m", order_purchase_timestamp) as order_by_year_month,

count(order_id) as Total_orders

from `target.orders`

group by 1

order by 1;
```

Untitled query RUN DOWNLOAD SHARE

```
18  select order_by_year_month, Total_orders,
19  from( select
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTI

Row	order_by_year_month	Total_orders
1	2016-09	4
2	2016-10	324
3	2016-12	1
4	2017-01	800
5	2017-02	1780
6	2017-03	2682
7	2017-04	2404
8	2017-05	3700
9	2017-06	3245
10	2017-07	4026

By observing the output we can conclude the order from month to month over years. Mostly there is a significant raise in the orders placed by the customers.

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

```
select

format_timestamp('%m', order_purchase_timestamp) as order_placed_by_months,
```

```
count(order_id) as Total_orders

from `target.orders`

group by 1

order by 1;
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON

Row	order_placed_by_months	Total_orders
1	01	8069
2	02	8508
3	03	9893
4	04	9343
5	05	10573
6	06	9412
7	07	10318
8	08	10843
9	09	4305
10	10	4959
11	11	7544
12	12	5674

As we observe the output we can analyze that

In the months of may, june, july, august most of the orders are placed

In the months of january, february, march, april the order placed are moderate.

In the months of September, October, November, December orders being placed are reduced.

C. 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

```
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"
else "Night"
end as Time_of_day,
count(order_id) as Total_orders
from `target.orders`
group by 1
order by 2 desc
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON

Row	Time_of_day	Total_orders
1	Afternoon	38135
2	Night	28331
3	Morning	27733
4	Dawn	5242

From the results we can conclude most of the orders were being placed at the Afternoon of the day and less number of orders were being placed at Dawn of the day.

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

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

```
select

c.customer_state,

format_timestamp("%m", o.order_purchase_timestamp) as Order_placed_by_month,

count(o.order_id) as No_of_order

from `target.orders` o left join  `target.customers` c

on o.customer_id = c.customer_id

group by 1,2

order by 1,2
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS

Row	customer_state	Order_placed_by_month	No_of_order
1	AC	01	8
2	AC	02	6
3	AC	03	4
4	AC	04	9
5	AC	05	10
6	AC	06	7
7	AC	07	9
8	AC	08	7
9	AC	09	5
10	AC	10	6
11	AC	11	5
12	AC	12	5
13	AL	01	39
14	AL	02	39
15	AL	03	40

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

```
select

customer_state,

count(distinct customer_unique_id) as Customers_by_unique_id,

count(distinct customer_id) as Customers_by_id

from `target.customers`

group by customer_state

order by 2 desc,3 desc;
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTIO

Row	customer_state	Customers_by_uniqu	Customers_by_id
1	SP	40302	41746
2	RJ	12384	12852
3	MG	11259	11635
4	RS	5277	5466
5	PR	4882	5045
6	SC	3534	3637
7	BA	3277	3380
8	DF	2075	2140
9	ES	1964	2033
10	GO	1952	2020
11	PE	1609	1652
12	CE	1313	1336
13	PA	949	975
14	MT	876	907
15	MA	726	747

From the customers we can identify that there were two columns of customer_id and customer_unique_id By observing the result we can conclude that some of customer_uniqe_id has been associated with multiple customer_id. We can consider Customer_unique_id gives accurate results. SP,RJ,MG states have the most number of customers.

IV. 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).

```
select 100 * ((cost - lag(cost) over(order by year)) / lag(cost) over(order by year) ) as inc_in_percentage

from (

    select

    extract(year from o.order_purchase_timestamp) as year,
```

```
    sum(p.payment_value) as cost

    from `target.orders` o join `target.payments` p

    on o.order_id = p.order_id

    where extract(month from o.order_purchase_timestamp) between 1 and 8 and

          extract(year from o.order_purchase_timestamp) in (2017 , 2018)

    group by year

    order by year

      ) tbl

order by 1 desc

limit 1
```

```
select 100 * ((cost - lag(cost) over(order by year)) / lag(cost) over(order by year) ) as inc_in_percentage
from (
    select
    extract(year from o.order_purchase_timestamp) as year,
    sum(p.payment_value) as cost
    from `target.orders` o join `target.payments` p
    on o.order_id = p.order_id
    where extract(month from o.order_purchase_timestamp) between 1 and 8 and
          extract(year from o.order_purchase_timestamp) in (2017 , 2018)
    group by year
    order by year
      ) tbl
order by 1 desc
limit 1
```

ıery results

B INFORMATION	RESULTS	CHART	JSON	EXECUTION DETAILS	EXECUTION GRAPH

	inc_in_percentage
1	136.9768716466...

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

```sql
select

c.customer_state,

round(sum(tbl.Total_value),2) as Total_value,

round(avg(tbl.Total_value), 2 ) as Avg_value

from `target.customers` c  join

  (

  select

  o.customer_id as customer_id,

  o.order_id,

  round(sum(p.payment_value), 2 ) as Total_value,

  from `target.orders` o join `target.payments` p

  on o.order_id = p.order_id

  group by customer_id, o.order_id

  order by o.order_id

  ) tbl on c.customer_id = tbl.customer_id

group by c.customer_state

order by c.customer_state;
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION

Row	customer_state	Total_value	Avg_value
1	AC	19680.62	242.97
2	AL	96962.06	234.77
3	AM	27966.93	188.97
4	AP	16262.8	239.16
5	BA	616645.82	182.44
6	CE	279464.03	209.18
7	DF	355141.08	165.95
8	ES	325967.55	160.34
9	GO	350092.31	173.31
10	MA	152523.02	204.18
11	MG	1872257.26	160.92
12	MS	137534.84	192.36
13	MT	187029.29	206.21
14	PA	218295.85	223.89
15	PB	141545.72	264.08

For a single order there are multiple payment values. SO instead of calculating the total and average we had used sub query.

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

```
select c.customer_state,
   sum(tbl.Total_freight_value) as Total_freight_value,
   avg(tbl.Total_freight_value) as Avg_freight_value
from `target.customers` c join
    (select
    o.customer_id as customer_id,
    oi.order_id as order_id,
    sum(oi.freight_value) as Total_freight_value
    from `target.order_items` oi join `target.orders` o
```

```
 on oi.order_id = o.order_id

 group by oi.order_id, o.customer_id

 ) tbl

on c.customer_id = tbl.customer_id

group by c.customer_state

order by c.customer_state
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS

Row	customer_state	Total_freight_value	Avg_freight_value
1	AC	3686.749999999…	45.51543209876…
2	AL	15914.58999999…	38.72163017031…
3	AM	5478.890000000…	37.27136054421…
4	AP	2788.500000000…	41.00735294117…
5	BA	100156.6799999…	29.82628945801…
6	CE	48351.58999999…	36.43676714393…
7	DF	50625.49999999…	23.82376470588…
8	ES	49764.59999999…	24.57511111111…
9	GO	53114.97999999…	26.46486297957…
10	MA	31523.77000000…	42.59968918918…
11	MG	270853.4600000…	23.46270443520…
12	MS	19144.03	27.00145275035…
13	MT	29715.43000000…	32.90745293466…
14	PA	38699.30000000…	39.89618556701…

V. 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.

```
select

order_id,

date_diff(order_delivered_customer_date , order_purchase_timestamp, day) as No_of_days_to_deliver,

date_diff(order_delivered_customer_date ,order_estimated_delivery_date, day) as Diff_of_est_delivered

from `target.orders`

order by 2 desc, 3 desc
```

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION

ow	order_id	No_of_days_to_deliv	Diff_of_est_delivered
1	ca07593549f1816d26a572e06...	209	181
2	1b3190b2dfa9d789e1f14c05b...	208	188
3	440d0d17af552815d15a9e41a...	195	165
4	285ab9426d6982034523a855f...	194	166
5	0f4519c5f1c541ddec9f21b3bd...	194	161
6	2fb597c2f772eca01b1f5c561b...	194	155
7	47b40429ed8cce3aee9199792...	191	175
8	2fe324febf907e3ea3f2aa9650...	189	167
9	2d7561026d542c8dbd8f0daea...	188	159
10	c27815f7e3dd0b926b5855262...	187	162
11	437222e3fd1b07396f1d9ba8c...	187	144
12	dfe5f68118c2576143240b8d7...	186	153
13	6e82dcfb5eada6283dba34f16...	182	155
14	2ba1366baecad3c3536f27546...	181	152

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

From the query 4c we had already calculate the Avg of freight value by placing the same query in cte we can calculate the the top 5 and bottom 5 with highest and lowest Average freight Vvlue

```
with avg_freight_cte as (

    select

       c.customer_state as customer_state,

       sum(tbl.Total_freight_value) as Total_freight_value,

       avg(tbl.Total_freight_value) as Avg_freight_value

  from `target.customers` c join

        (select

        o.customer_id as customer_id,

        oi.order_id as order_id,

        sum(oi.freight_value) as Total_freight_value

        from `target.order_items` oi join `target.orders` o

        on oi.order_id = o.order_id

        group by oi.order_id, o.customer_id

        ) tbl

  on c.customer_id = tbl.customer_id
```

```sql
  group by customer_state

  order by customer_state

),

dense_rank_cte as (

select

avg_freight_cte.customer_state as customer_state,

dense_rank() over(order by avg_freight_cte.Avg_freight_value desc) as
high_rank,

dense_rank() over(order by avg_freight_cte.Avg_freight_value asc) as
low_rank

from avg_freight_cte

)

select

h.customer_state as High_avg_freight_value,

l.customer_state as low_avg_freight_value

from dense_rank_cte h join dense_rank_cte l

on h.high_rank = l.low_rank

where h.high_rank <= 5

order by h.high_rank
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON

Row	High_avg_freight_value	low_avg_freight_value
1	RR	SP
2	PB	MG
3	RO	PR
4	AC	DF
5	PI	RJ

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

```
with avg_cte as(

select

c.customer_state as customer_state,

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

from `target.orders` o  join `target.customers` c

on o.customer_id = c.customer_id

where o.order_delivered_customer_date is not null

group by c.customer_state

),

rank_avg as(

 select

 avg_cte.customer_state as customer_state,
```

```
 dense_rank() over(order by avg_cte.No_of_days_to_deliver desc) as
high_rank,

 dense_rank() over(order by avg_cte.No_of_days_to_deliver asc) as low_rank

 from avg_cte

)

select

h.customer_state as High_days_to_deliver,

l.customer_state as low_days_to_deliver

from rank_avg h join rank_avg l

on h.high_rank = l.low_rank

where h.high_rank <= 5

order by h.high_rank
```

Query results

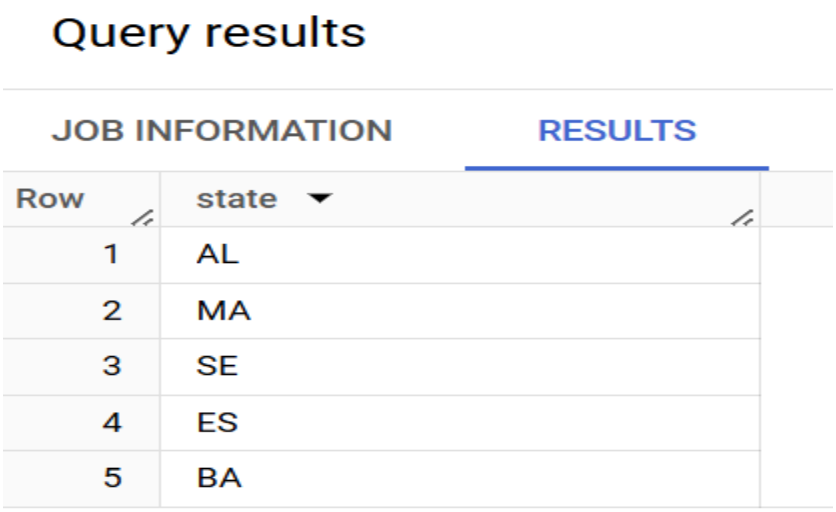
JOB INFORMATION | RESULTS | CHART | JSON

Row	High_days_to_deliver	low_days_to_deliver
1	RR	SP
2	AP	PR
3	AM	MG
4	AL	DF
5	PA	SC

D. Find out the top 5 states where the order delivery is really fast as

```
compared to the estimated date of delivery.

with avg_diff_est_cte as(

select

c.customer_state as customer_state,

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

from `target.orders` o join `target.customers` c

on o.customer_id = c.customer_id

where o.order_delivered_customer_date is not null

group by 1

),

ranks_cte as (

  select

  avg_diff_est_cte.customer_state as state,

  dense_rank() over(order by avg_diff_est_cte.Diff_of_est_delivered desc)
high_rank

  from avg_diff_est_cte

)

select

ranks_cte.state as state

from ranks_cte
```

```
where ranks_cte.high_rank <= 5

order by ranks_cte.high_rank
```



Query results

JOB INFORMATION | RESULTS

Row	state
1	AL
2	MA
3	SE
4	ES
5	BA

The cities with the lowest difference will deliver the order fast.

VI. Analysis based on the payments:

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

```
      with order_payment_cte as (

  select

  distinct o.order_id as order_id,

  format_timestamp("%Y-%m", order_purchase_timestamp) order_purchase,

  p.payment_type

  from `target.orders` o join `target.payments` p

  on o.order_id = p.order_id
```

```sql
  where p.payment_value >=0
),
date_payment_count as (
  select
  order_payment_cte.order_purchase as order_purchase,
  payment_type,
  count(*) as No_of_orders
  from order_payment_cte
  group by 1,2
  order by 1,2
)

select
payment_type,
order_purchase,
date_payment_count.No_of_orders as No_of_orders,
sum(date_payment_count.No_of_orders) over(partition by payment_type order by
order_purchase) as month_order
from date_payment_count
order by 1,2
```

```sql

with order_payment_cte as (
  select
  distinct o.order_id as order_id,
  format_timestamp("%Y-%m", order_purchase_timestamp) order_purchase,
  p.payment_type
  from `target.orders` o join `target.payments` p
  on o.order_id = p.order_id
  where p.payment_value >=0
),
date_payment_count as (
  select
  order_payment_cte.order_purchase as order_purchase,
  payment_type,
  count(*) as No_of_orders
  from order_payment_cte
  group by 1,2
  order by 1,2
)

select
payment_type,
order_purchase,
date_payment_count.No_of_orders as No_of_orders,
sum(date_payment_count.No_of_orders) over(partition by payment_type order by order_purchase) as month_order
from date_payment_count
order by 1,2
```

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

```sql
select

payment_installments,

count(distinct order_id) as order_id

from `target.payments`

where payment_type = "credit_card" and payment_installments > 1 and payment_value > 0

group by 1

having order_id >= 1
```

Query results

JOB INFORMATION | RESULTS | CHAR

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