

## SQL - Case Study

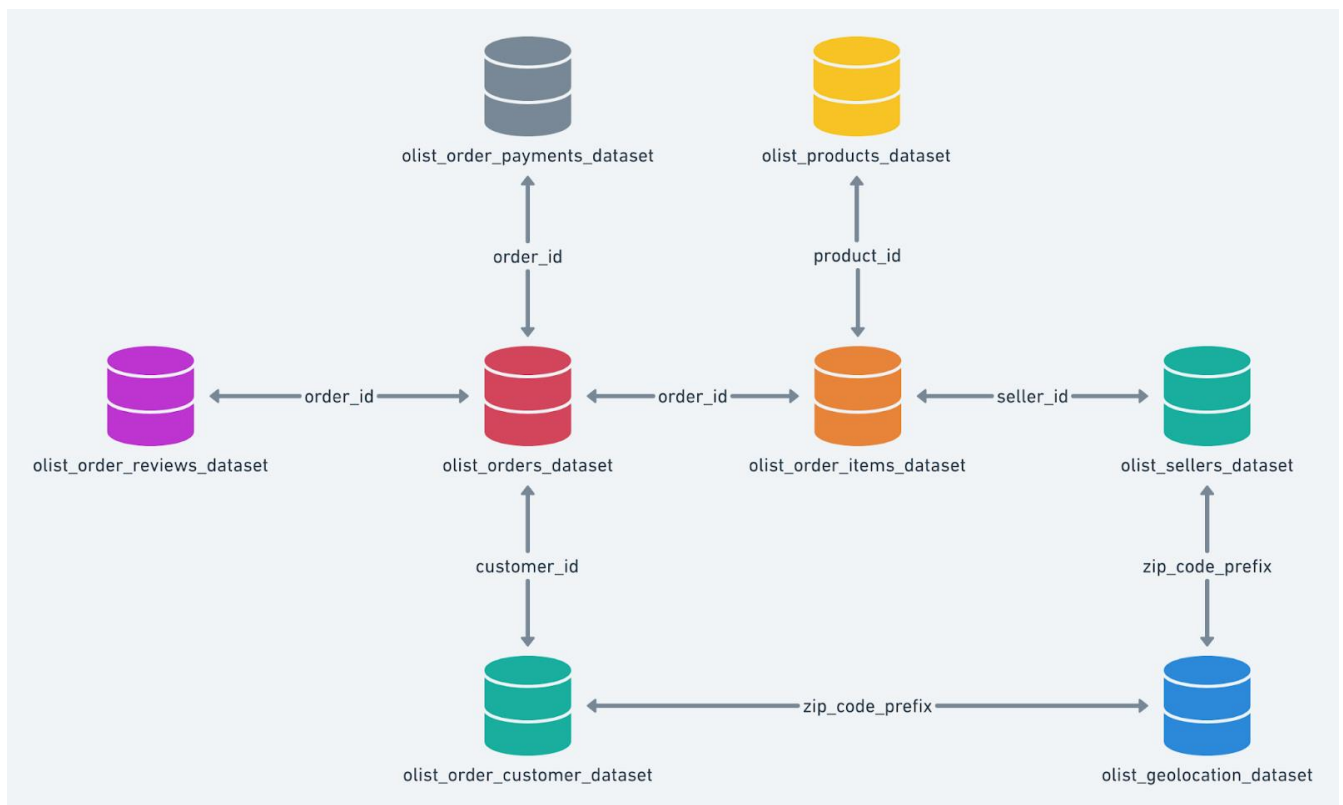
|                     |                     |
|---------------------|---------------------|
| <b>Company Name</b> | XYZ                 |
| <b>Category</b>     | E-Commerce Retailer |
| <b>Country Data</b> | Brazil              |
| <b>No of Tables</b> | 8                   |

**By**

**Naman Garg**

## Table Names:

1. customers
2. geolocation
3. order\_items
4. payments
5. reviews
6. orders
7. products
8. sellers

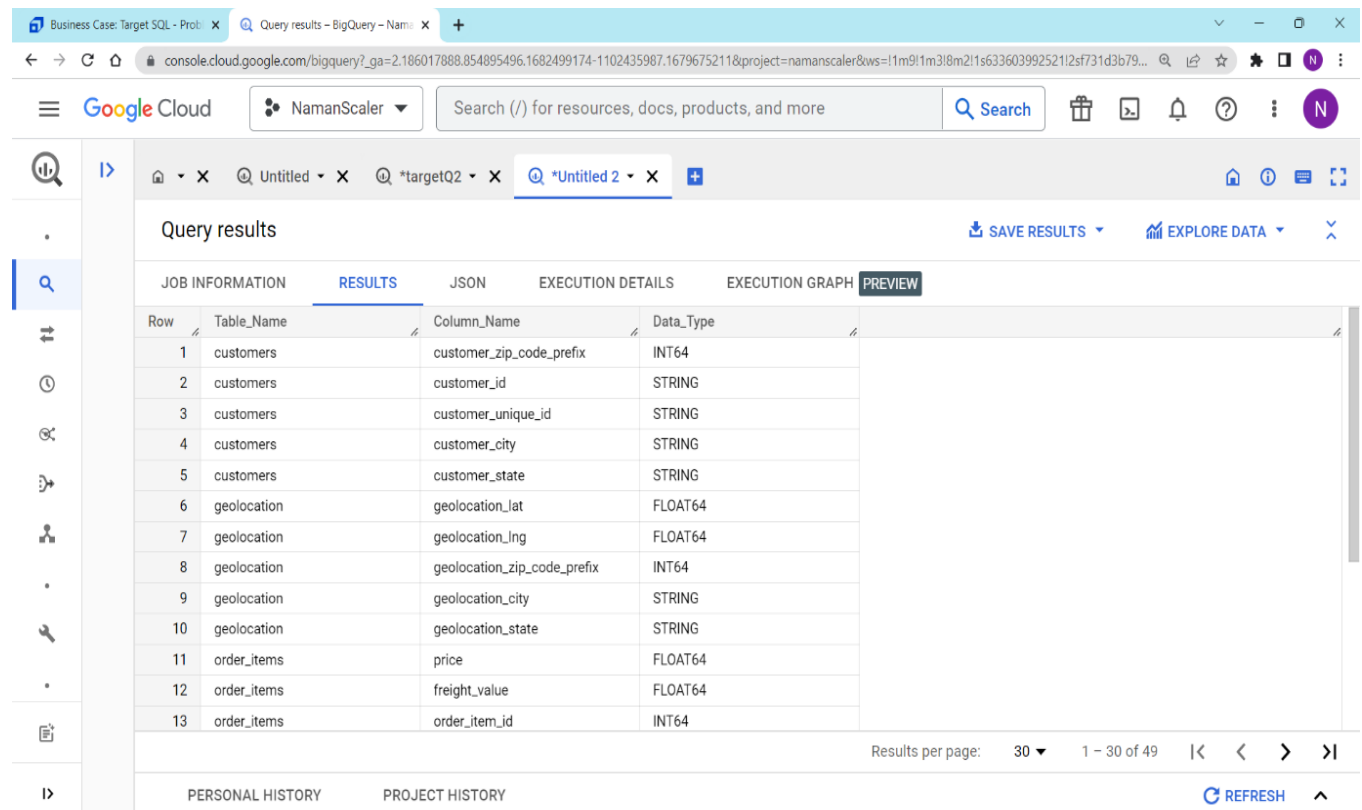


## Database Schema

### Q1.1 Data type of columns in a table.

Ans - I have fetched all column names and their datatypes from Database using below query.

```
select Table_Name, Column_Name, Data_Type
from namanscaler.Target_Database.INFORMATION_SCHEMA.COLUMNS
order by Table_name,Data_Type
```



Query results

| Row | Table_Name  | Column_Name                 | Data_Type |
|-----|-------------|-----------------------------|-----------|
| 1   | customers   | customer_zip_code_prefix    | INT64     |
| 2   | customers   | customer_id                 | STRING    |
| 3   | customers   | customer_unique_id          | STRING    |
| 4   | customers   | customer_city               | STRING    |
| 5   | customers   | customer_state              | STRING    |
| 6   | geolocation | geolocation_lat             | FLOAT64   |
| 7   | geolocation | geolocation_lng             | FLOAT64   |
| 8   | geolocation | geolocation_zip_code_prefix | INT64     |
| 9   | geolocation | geolocation_city            | STRING    |
| 10  | geolocation | geolocation_state           | STRING    |
| 11  | order_items | price                       | FLOAT64   |
| 12  | order_items | freight_value               | FLOAT64   |
| 13  | order_items | order_item_id               | INT64     |

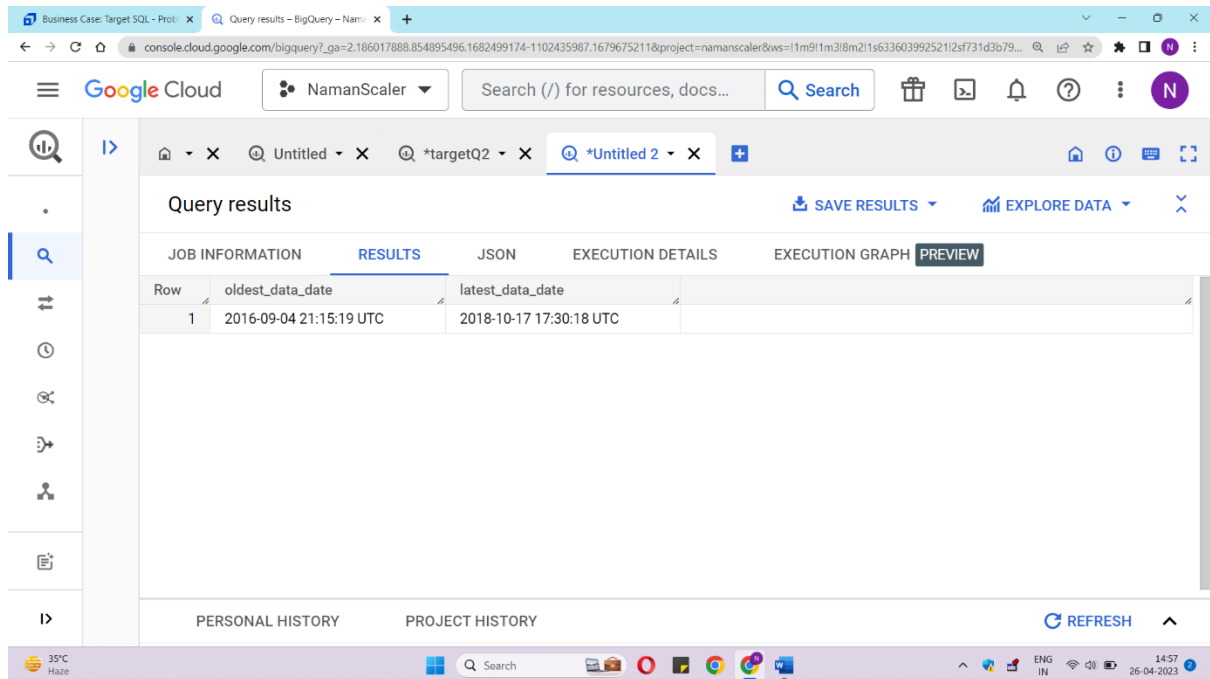
### Insight:

From this I understood the structure and type of data present in the database. Mostly string, integer, timestamp are used as datatype and most of the data are connected through **order\_id** and **column\_id**.

## Q1.2 Time period for which the data is given.

Ans- After going through the schema I have found out that 'order\_purchase\_timestamp' column can tell us about the time period of data using below query.

```
select min(order_purchase_timestamp) as oldest_data_date,  
max(order_purchase_timestamp) as latest_data_date  
from `Target_Database.orders`
```



The screenshot shows the Google Cloud BigQuery console interface. The top navigation bar includes the Google Cloud logo, the user name 'NamanScaler', and a search bar. Below the navigation bar, the 'Query results' tab is active, displaying a table with the following data:

| Row | oldest_data_date        | latest_data_date        |
|-----|-------------------------|-------------------------|
| 1   | 2016-09-04 21:15:19 UTC | 2018-10-17 17:30:18 UTC |

The table is displayed in a single row, indicating that the query returned only one result. The console also shows tabs for 'JOB INFORMATION', 'RESULTS', 'JSON', 'EXECUTION DETAILS', and 'EXECUTION GRAPH'. The bottom status bar shows the current time as 14:57 on 26-04-2023.

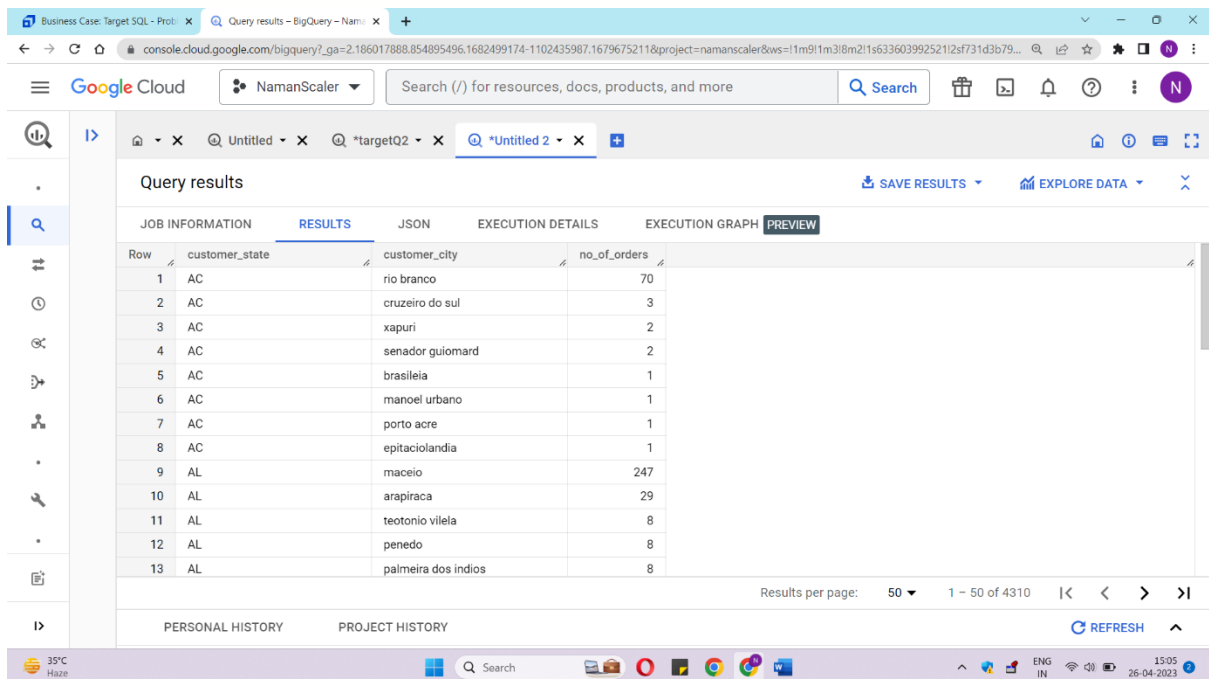
## Insight:

From this I got to know about the oldest data date and latest data date and that is from September 2016 to October 2018.

### Q1.3 Cities and States of customers ordered during the given period.

**Ans-** I have taken all the states and their respective cities from which orders are placed and their amount by using below query.

```
select customer_state, customer_city, count(*) as no_of_orders
from `Target_Database.orders` o1
join `Target_Database.customers` cust on o1.customer_id=cust.customer_id
group by customer_state, customer_city
order by customer_state, no_of_orders desc
```



The screenshot shows the Google Cloud BigQuery console interface. The query results are displayed in a table with the following data:

| Row | customer_state | customer_city       | no_of_orders |
|-----|----------------|---------------------|--------------|
| 1   | AC             | rio branco          | 70           |
| 2   | AC             | cruzeiro do sul     | 3            |
| 3   | AC             | xapuri              | 2            |
| 4   | AC             | senador guiomard    | 2            |
| 5   | AC             | brasileia           | 1            |
| 6   | AC             | manoel urbano       | 1            |
| 7   | AC             | porto acre          | 1            |
| 8   | AC             | epitaciolandia      | 1            |
| 9   | AL             | maceio              | 247          |
| 10  | AL             | arapiraca           | 29           |
| 11  | AL             | teotonio vilela     | 8            |
| 12  | AL             | penedo              | 8            |
| 13  | AL             | palmeira dos indios | 8            |

The interface includes tabs for JOB INFORMATION, RESULTS, JSON, EXECUTION DETAILS, EXECUTION GRAPH, and PREVIEW. The RESULTS tab is active, showing the query results. The bottom of the screen displays the Google Cloud logo, search bar, and system status (35°C, Haze, 15:05, 26-04-2023).

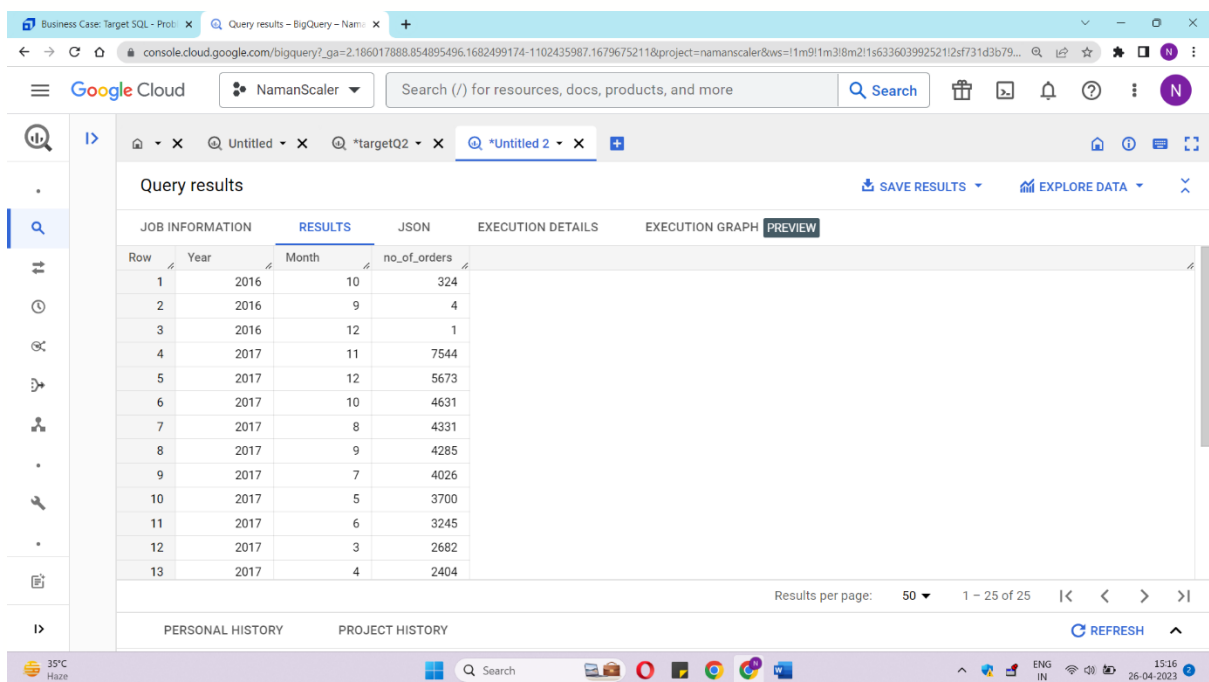
### Insight:

From this we can see the number of orders placed from each city of each state and by ordering we can also infer that city 'rio branco' is generating highest number of orders from customer state 'AC' and similarly we can see of other states as well.

**Q2.1 Is there a growing trend on e-commerce in Brazil? How can we describe a complete scenario? Can we see some seasonality with peaks at specific months?**

**Ans -** In this I have counted all orders from 2016 to 2018 month on month.

```
select Year, Month, count (*) as no_of_orders
from (select extract (year from order_purchase_timestamp) as Year,
extract (month from order_purchase_timestamp) as Month
from `Target_Database.orders`) as temp
group by Year, Month
order by Year, no_of_orders desc
```



The screenshot shows the Google Cloud BigQuery console interface. The query results are displayed in a table with the following data:

| Row | Year | Month | no_of_orders |
|-----|------|-------|--------------|
| 1   | 2016 | 10    | 324          |
| 2   | 2016 | 9     | 4            |
| 3   | 2016 | 12    | 1            |
| 4   | 2017 | 11    | 7544         |
| 5   | 2017 | 12    | 5673         |
| 6   | 2017 | 10    | 4631         |
| 7   | 2017 | 8     | 4331         |
| 8   | 2017 | 9     | 4285         |
| 9   | 2017 | 7     | 4026         |
| 10  | 2017 | 5     | 3700         |
| 11  | 2017 | 6     | 3245         |
| 12  | 2017 | 3     | 2682         |
| 13  | 2017 | 4     | 2404         |

The interface includes tabs for 'JOB INFORMATION', 'RESULTS', 'JSON', 'EXECUTION DETAILS', 'EXECUTION GRAPH', and 'PREVIEW'. The 'RESULTS' tab is active, showing the table data. The bottom of the screen shows a Windows taskbar with the date 26-04-2023 and time 15:16.

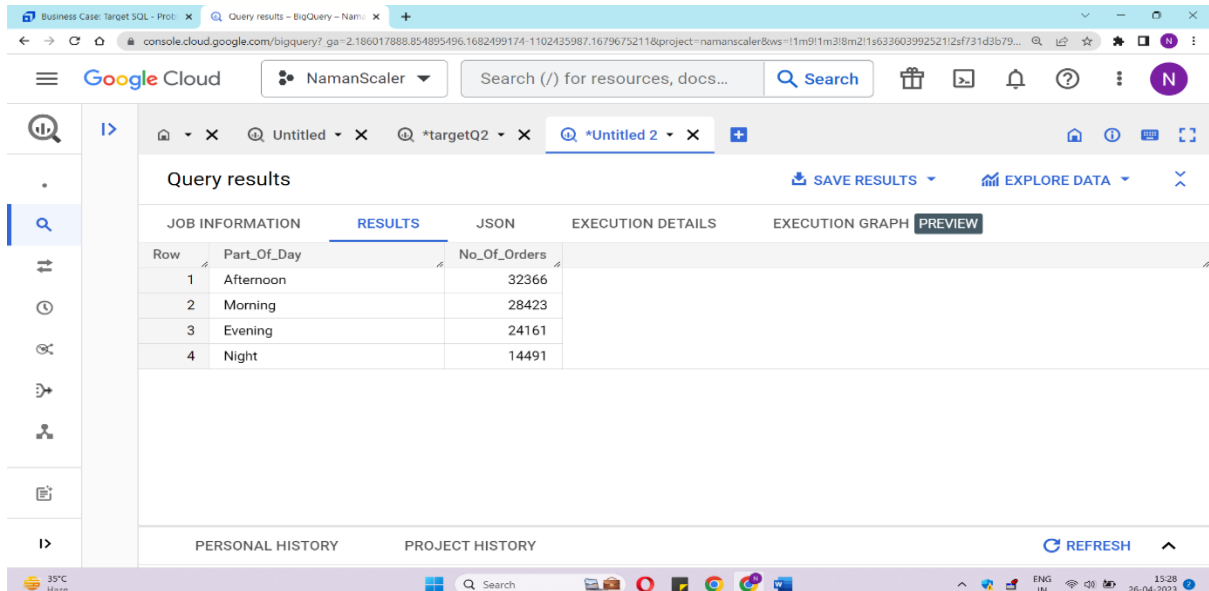
## Insight:

From this we infer that this e-commerce business is growing as no\_of\_orders are increasing month on month. And by ordering we figure out that most no\_of\_orders are in the months between October and February.

## Q2.2 What time do Brazilian customers tend to buy (Evening, Morning, Afternoon or Night)?

**Ans-** I have used case statement to indicate part of days and then have grouped them by taking no of orders in that part of day using below query.

```
Select Part_Of_Day, count (*) as No_Of_Orders
from (
select extract (hour from order_purchase_timestamp) as Hour,
case
when extract(hour from order_purchase_timestamp)>=5 and extract(hour from order_purchase_timestamp)<=12 then 'Morning'
when extract(hour from order_purchase_timestamp)>12 and extract(hour from order_purchase_timestamp)<=17 then 'Afternoon'
when extract(hour from order_purchase_timestamp)>17 and extract(hour from order_purchase_timestamp)<=21 then 'Evening'
when extract(hour from order_purchase_timestamp)>21 and extract(hour from order_purchase_timestamp)<=23 then 'Night'
when extract(hour from order_purchase_timestamp)>=0 and extract(hour from order_purchase_timestamp)<=4 then 'Night'
end as Part_Of_Day
from `Target_Database.orders`) as temp
group by Part_Of_Day
order by No_Of_Orders desc
```



The screenshot shows the Google Cloud BigQuery console interface. The query results are displayed in a table with the following data:

| Row | Part_Of_Day | No_Of_Orders |
|-----|-------------|--------------|
| 1   | Afternoon   | 32366        |
| 2   | Morning     | 28423        |
| 3   | Evening     | 24161        |
| 4   | Night       | 14491        |

The interface also includes tabs for 'JOB INFORMATION', 'RESULTS', 'JSON', 'EXECUTION DETAILS', and 'EXECUTION GRAPH'. The 'RESULTS' tab is currently selected, showing the table data. At the bottom, there are sections for 'PERSONAL HISTORY' and 'PROJECT HISTORY', and a 'REFRESH' button.

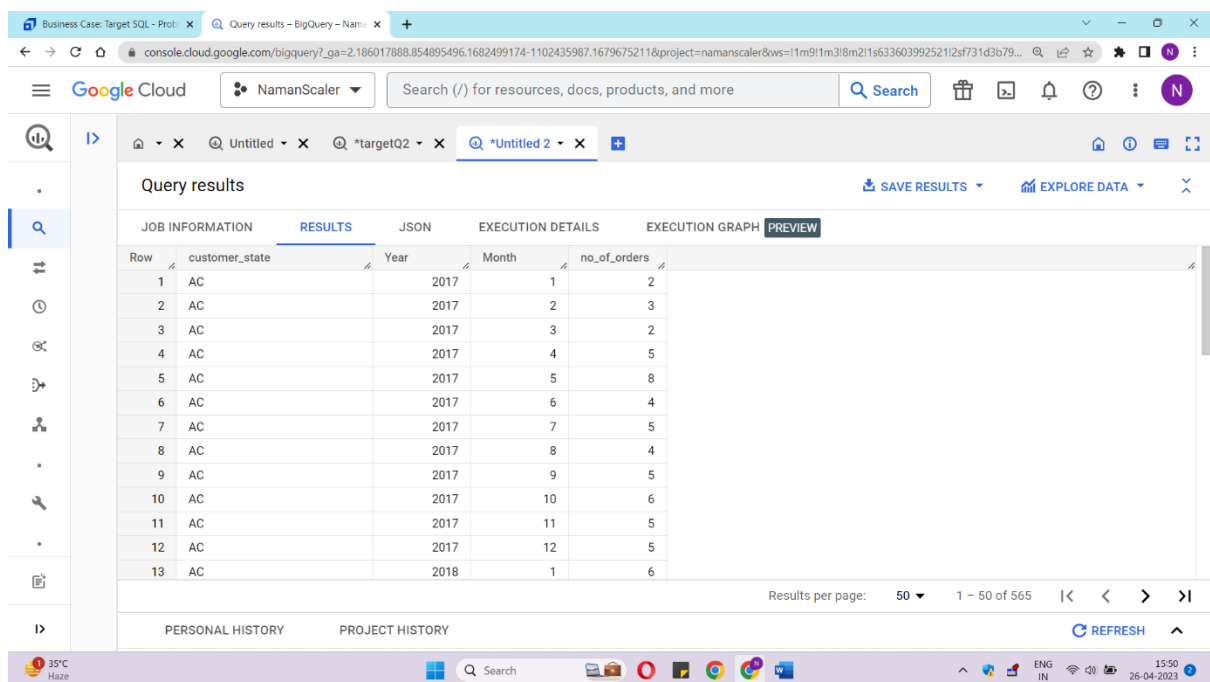
### Insight:

From this we infer that people mostly place orders in afternoon and morning that is between 5 am to 5pm and very less people place orders at night that is from 9pm to 4am.

### Q3.1 Get month on month orders by states.

**Ans-** I have taken out count of orders state wise and month wise for the entire time period of data.

```
select customer_state, extract (year from order_purchase_timestamp) as Year,  
extract (month from order_purchase_timestamp) as Month,  
count (distinct ord.order_id) as no_of_orders  
from `Target_Database.orders` ord  
join `Target_Database.customers` cust on ord.customer_id=cust.customer_id  
group by customer_state, Year, Month  
order by customer_state, Year, Month
```



The screenshot shows the Google Cloud BigQuery console interface. The query results are displayed in a table with the following columns: Row, customer\_state, Year, Month, and no\_of\_orders. The data shows a steady increase in the number of orders from January to December 2017, with a slight dip in December 2017 compared to November 2017. The state 'AC' is consistently the top state for orders.

| 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            |
| 13  | AC             | 2018 | 1     | 6            |

### Insight:

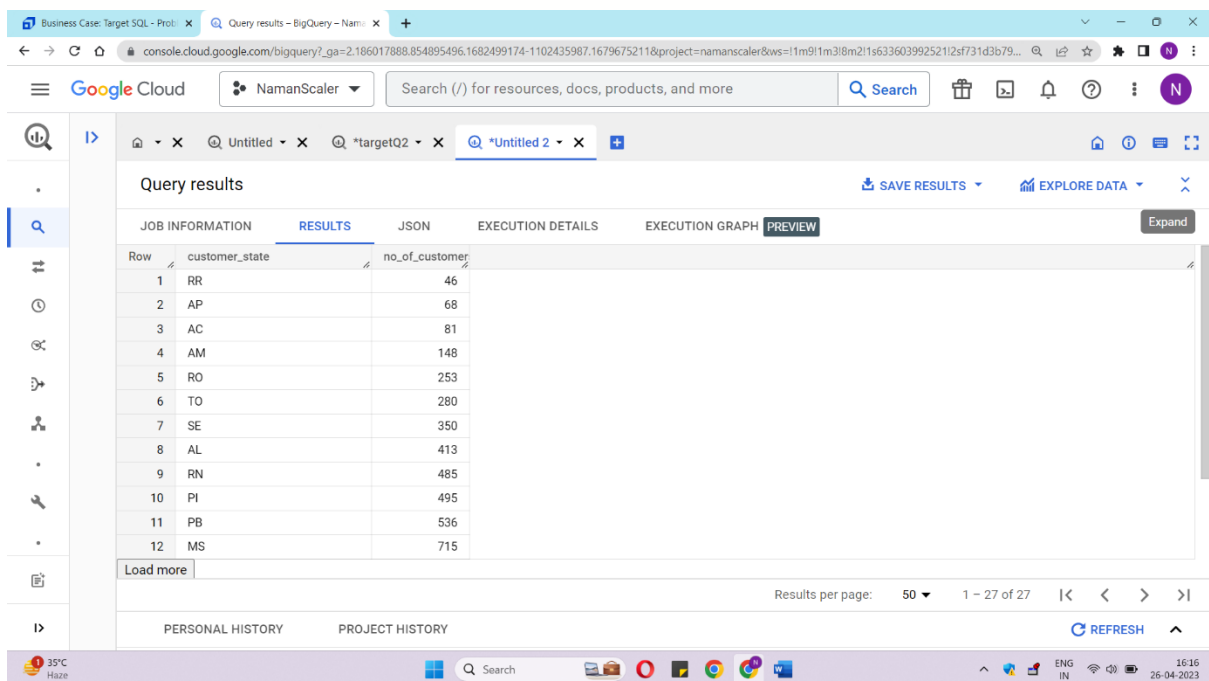
From this we can see growth of business state wise. There is not much growth in business.



### Q3.2 Distribution of customers across the states in Brazil

Ans- For Distribution data we have counted number of customers from each state in Brazil.

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



The screenshot shows the Google Cloud BigQuery console interface. The query results are displayed in a table with the following data:

| Row | customer_state | no_of_customer |
|-----|----------------|----------------|
| 1   | RR             | 46             |
| 2   | AP             | 68             |
| 3   | AC             | 81             |
| 4   | AM             | 148            |
| 5   | RO             | 253            |
| 6   | TO             | 280            |
| 7   | SE             | 350            |
| 8   | AL             | 413            |
| 9   | RN             | 485            |
| 10  | PI             | 495            |
| 11  | PB             | 536            |
| 12  | MS             | 715            |

The interface includes tabs for JOB INFORMATION, RESULTS, JSON, EXECUTION DETAILS, and EXECUTION GRAPH. The RESULTS tab is active, showing the table data. A 'Load more' button is visible at the bottom of the table. The console also shows a search bar, project name 'NamanScaler', and a sidebar with navigation icons.

### Insight:

From this we can get total number of customers from each state and thus find out which state needs more attention. Thus, from above we can see that top 5 customer states needs more attention.

**Q4.1 Get % increase in cost of orders from 2017 to 2018 (include months between Jan to Aug only) - You can use “payment\_value” column in payments table.**

**Ans-**

```

Select *,
((Total_Sales - lag(Total_Sales, 1)
over(order by Year_Month))/(lag(Total_Sales,1) over(order by Year_Month))*100) as
Percentage_Difference_Cost_Of_Orders
From (
select FORMAT_DATE ('%Y-%m', DateTime (order_purchase_timestamp)) as Year_Month,
sum(payment_value) as Total_Sales
from `Target_Database.orders` ord
join `Target_Database.payments` pay on ord.order_id = pay.order_id
where (order_purchase_timestamp>='2017-01-01' and order_purchase_timestamp<='2017-
08-31') or (order_purchase_timestamp>='2018-01-
01' and order_purchase_timestamp<='2018-08-31')
group by Year_Month
order by Year_Month) as temp
order by Year_Month

```

The screenshot shows the Google Cloud BigQuery console interface. The query results are displayed in a table with the following data:

| Row | Year_Month | Total_Sales   | Percentage_Diff |
|-----|------------|---------------|-----------------|
| 1   | 2017-01    | 138488.039... | null            |
| 2   | 2017-02    | 291908.009... | 110.782107...   |
| 3   | 2017-03    | 449863.600... | 54.1114270...   |
| 4   | 2017-04    | 417788.030... | -7.13006564...  |
| 5   | 2017-05    | 592918.820... | 41.9185753...   |
| 6   | 2017-06    | 511276.380... | -13.7695814...  |
| 7   | 2017-07    | 592382.920... | 15.8635413...   |
| 8   | 2017-08    | 650481.470... | 9.80760046...   |
| 9   | 2018-01    | 1115004.18... | 71.4121356...   |
| 10  | 2018-02    | 992463.340... | -10.9901686...  |
| 11  | 2018-03    | 1159652.11... | 16.8458393...   |

The table shows a significant increase in total sales from 2017 to 2018, with a corresponding increase in the percentage difference. The interface also includes a sidebar with navigation options and a bottom status bar showing the current time and date.

**Insight:**

From this you can see the % increase in cost of orders from 2017 to 2018 month-wise, negative percentage indicates that there is decrease in cost of orders. We can see that % difference is decreasing which means that total sales is decreasing which is not good.

## Q4.2 Mean & Sum of price and freight value by customer state.

Ans-

```
select customer_state, sum(price) as sum_price_item, avg(price) as avg_price_item,
sum(freight_value) as sum_freight_cost, avg(freight_value) as avg_freight_cost
from `Target_Database.orders` ord
join `Target_Database.order_items` ord_i on ord.order_id=ord_i.order_id
join `Target_Database.customers` cust on cust.customer_id=ord.customer_id
group by customer_state
order by customer_state
```

| Row | customer_state | sum_price_item | avg_price_item | sum_freight_cost | avg_freight_cost |
|-----|----------------|----------------|----------------|------------------|------------------|
| 1   | AC             | 15982.9499...  | 173.727717...  | 3686.74999...    | 40.0733695...    |
| 2   | AL             | 80314.81       | 180.889211...  | 15914.5899...    | 35.8436711...    |
| 3   | AM             | 22356.8400...  | 135.495999...  | 5478.88999...    | 33.2053939...    |
| 4   | AP             | 13474.2999...  | 164.320731...  | 2788.50000...    | 34.0060975...    |
| 5   | BA             | 511349.990...  | 134.601208...  | 100156.679...    | 26.3639589...    |
| 6   | CE             | 227254.709...  | 153.758261...  | 48351.5899...    | 32.7142016...    |
| 7   | DF             | 302603.939...  | 125.770548...  | 50625.4999...    | 21.0413549...    |
| 8   | ES             | 275037.309...  | 121.913701...  | 49764.5999...    | 22.0587765...    |
| 9   | GO             | 294591.949...  | 126.271731...  | 53114.9799...    | 22.7668152...    |
| 10  | MA             | 119648.219...  | 145.204150...  | 31523.7700...    | 38.2570024...    |
| 11  | MG             | 1585308.02...  | 120.748574...  | 270853.460...    | 20.6301668...    |
| 12  | MS             | 116812.639...  | 142.628376...  | 19144.0300...    | 23.3748840...    |
| 13  | MT             | 156453.529...  | 148.297184...  | 29715.4300...    | 28.1662843...    |

## Insight:

From this we can see average order value and average freight cost contribution by different states and total sales contribution.

### Q5.1 Calculate days between purchasing, delivering and estimated delivery.

$\text{time\_to\_delivery} = \text{order\_purchase\_timestamp} - \text{order\_delivered\_customer\_date}$

$\text{diff\_estimated\_delivery} = \text{order\_estimated\_delivery\_date} - \text{order\_delivered\_customer\_date}$

Group data by state, take mean of freight\_value, time\_to\_delivery, diff\_estimated\_delivery.

Ans-

with ctedelivery as (

select customer\_state, avg(freight\_value) as avg\_freight\_value,

avg (round (date\_diff (order\_delivered\_customer\_date, order\_purchase\_timestamp, hour)/24,1)) as avg\_time\_to\_delivery,

avg (round (date\_diff (order\_estimated\_delivery\_date, order\_delivered\_customer\_date, hour)/24,1)) as avg\_diff\_estimated\_delivery

from `Target\_Database.orders` ord

join `Target\_Database.customers` cust on ord.customer\_id=cust.customer\_id

join `Target\_Database.order\_items` oitem on oitem.order\_id=ord.order\_id

group by customer\_state)

select \* from ctedelivery

order by customer\_state

The screenshot shows the Google Cloud BigQuery console interface. At the top, there's a navigation bar with the Google Cloud logo, a search bar, and user information. Below this, a 'SANDBOX' banner indicates a trial version. The main area displays 'Query results' for a query named 'targetQ2'. The results are shown in a table with columns: Row, customer\_state, avg\_freight\_value, avg\_time\_to\_delivery, and avg\_diff\_estimated\_delivery. The table contains 11 rows of data for different states: AC, AL, AM, AP, BA, CE, DF, ES, GO, MA, and MG. The bottom of the interface shows a taskbar with various application icons and system information like 'USDINR -0.23%' and the date '26-04-2023'.

| Row | customer_state | avg_freight_value | avg_time_to_delivery | avg_diff_estimated_delivery |
|-----|----------------|-------------------|----------------------|-----------------------------|
| 1   | AC             | 40.0733695...     | 20.6967032...        | 20.3142857...               |
| 2   | AL             | 35.8436711...     | 24.4756440...        | 8.05245901...               |
| 3   | AM             | 33.2053939...     | 26.3730061...        | 19.2116564...               |
| 4   | AP             | 34.0060975...     | 28.1814814...        | 17.7555555...               |
| 5   | BA             | 26.3639589...     | 19.2302742...        | 10.2768123...               |
| 6   | CE             | 32.7142016...     | 20.9708976...        | 10.3990182...               |
| 7   | DF             | 21.0413549...     | 12.9405520...        | 11.4788959...               |
| 8   | ES             | 22.0587765...     | 15.6311460...        | 9.93779775...               |
| 9   | GO             | 22.7668152...     | 15.3857268...        | 11.5827843...               |
| 10  | MA             | 38.2570024...     | 21.6304999...        | 9.21525                     |
| 11  | MG             | 20.6301668...     | 11.9673376...        | 12.6252845...               |

## Q5.1 Top 5 states with highest/lowest average freight value - sort in desc/asc limit 5

Ans-

```
select customer_state, avg_freight_value
from ctedelivery
order by avg_freight_value desc
limit 5
```

The screenshot shows the Google Cloud BigQuery console interface. At the top, there's a navigation bar with the Google Cloud logo, a search bar, and user profile information. Below this, a sidebar on the left contains icons for navigation. The main area displays the 'Query results' for a query named '\*Untitled 2'. The results are shown in a table with two columns: 'customer\_state' and 'avg\_freight\_valu'. The table lists the top 5 states based on average freight value in descending order.

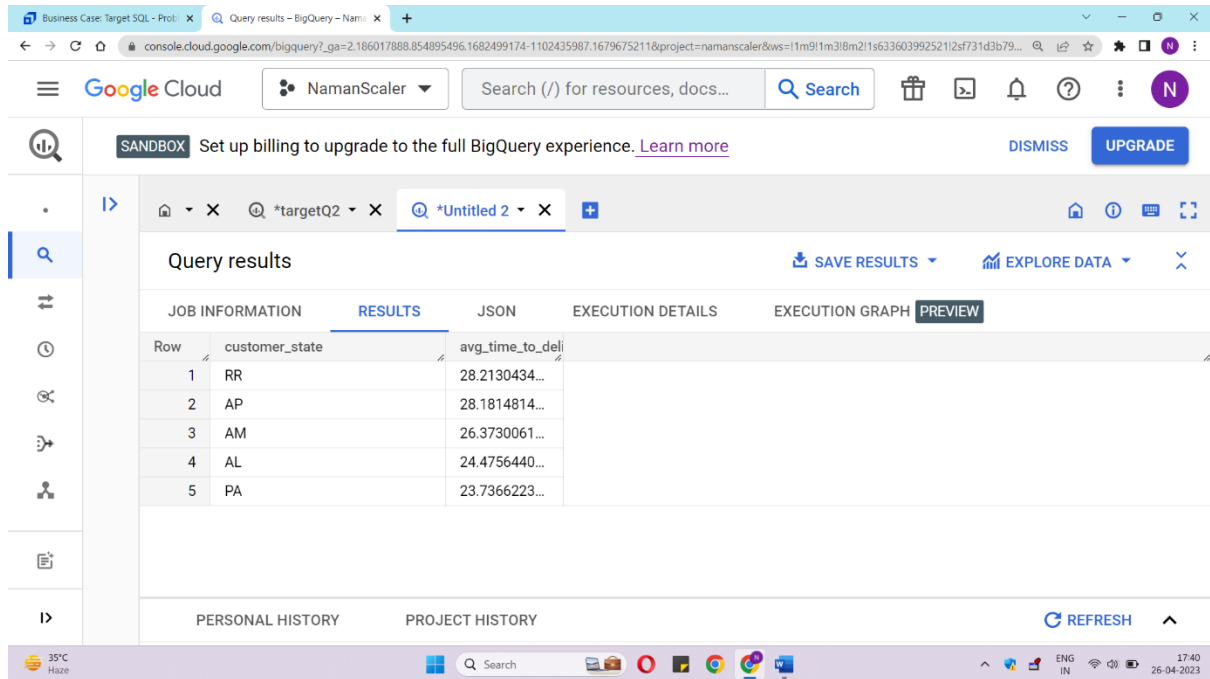
| Row | customer_state | avg_freight_valu |
|-----|----------------|------------------|
| 1   | RR             | 42.9844230...    |
| 2   | PB             | 42.7238039...    |
| 3   | RO             | 41.0697122...    |
| 4   | AC             | 40.0733695...    |
| 5   | PI             | 39.1479704...    |

At the bottom of the console, there's a status bar showing the current temperature (35°C Haze), a search bar, and system information (17:38, 26-04-2023).

## 5.2 Top 5 states with highest/lowest average time to delivery

Ans-

```
select customer_state, avg_time_to_delivery
from ctedelivery
order by avg_time_to_delivery desc
limit 5
```



The screenshot shows the Google Cloud BigQuery console interface. At the top, there's a navigation bar with the Google Cloud logo, a search bar, and a user profile icon. Below this, a sidebar on the left contains various icons for navigation. The main area displays the query results for a query named '\*Untitled 2'. The results are shown in a table with two columns: 'customer\_state' and 'avg\_time\_to\_delivery'. The table lists the top 5 states based on the average time to delivery in descending order.

| Row | customer_state | avg_time_to_delivery |
|-----|----------------|----------------------|
| 1   | RR             | 28.2130434...        |
| 2   | AP             | 28.1814814...        |
| 3   | AM             | 26.3730061...        |
| 4   | AL             | 24.4756440...        |
| 5   | PA             | 23.7366223...        |

### Q5.3 Top 5 states where delivery is really fast/ not so fast compared to estimated date.

Ans-

```
select customer_state, avg_diff_estimated_delivery
from ctedelivery
order by avg_diff_estimated_delivery desc
limit 5
```

The screenshot shows the Google Cloud BigQuery console interface. At the top, there's a navigation bar with the Google Cloud logo, a search bar, and user information. Below this, a banner for 'SANDBOX' is visible. The main area displays 'Query results' for a query named '\*Untitled 2'. The results are shown in a table with columns 'customer\_state' and 'avg\_diff\_estimated\_delivery'. The table lists the top 5 states: AC, RO, AM, AP, and RR, ordered by their average difference in descending order. The 'AC' state has the highest difference at 20.3142857... days.

| Row | customer_state | avg_diff_estimated_delivery |
|-----|----------------|-----------------------------|
| 1   | AC             | 20.3142857...               |
| 2   | RO             | 19.3285714...               |
| 3   | AM             | 19.2116564...               |
| 4   | AP             | 17.7555555...               |
| 5   | RR             | 17.5978260...               |

### Insight:

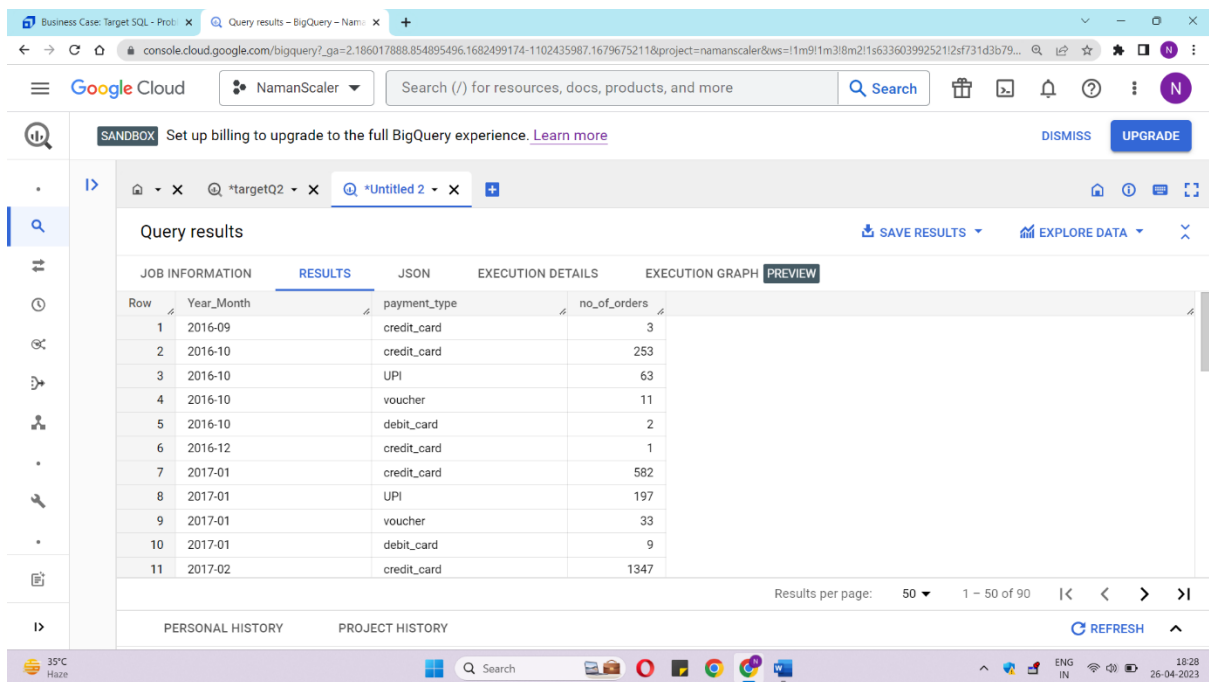
From above data we can analyse that as time of delivery increases freight value also increases. So there is a need to see why for some states freight are high, is it due to distance or something else.

Next we can see that for many states estimated delivery date is predicted wrong as you can see that for customer\_state AC the avg\_diff\_estimated\_Delivery is 20 days, this means that the product was delivered 20 days earlier than predicted so there is a need to optimize the estimated delivery date prediction system

## Q6.1 Month over Month count of orders for different payment types

Ans-

```
Select FORMAT_DATE ('%Y-%m', DateTime (order_purchase_timestamp)) as Year_Month,
payment_type, count (distinct pay.order_id) as no_of_orders
from `Target_Database.orders` ord
join `Target_Database.payments` pay on ord.order_id=pay.order_id
group by Year_Month, payment_type
order by Year_Month, no_of_orders desc
```



The screenshot shows the Google Cloud BigQuery console interface. The query results are displayed in a table with the following data:

| Row | Year_Month | payment_type | no_of_orders |
|-----|------------|--------------|--------------|
| 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         |

## Insight:

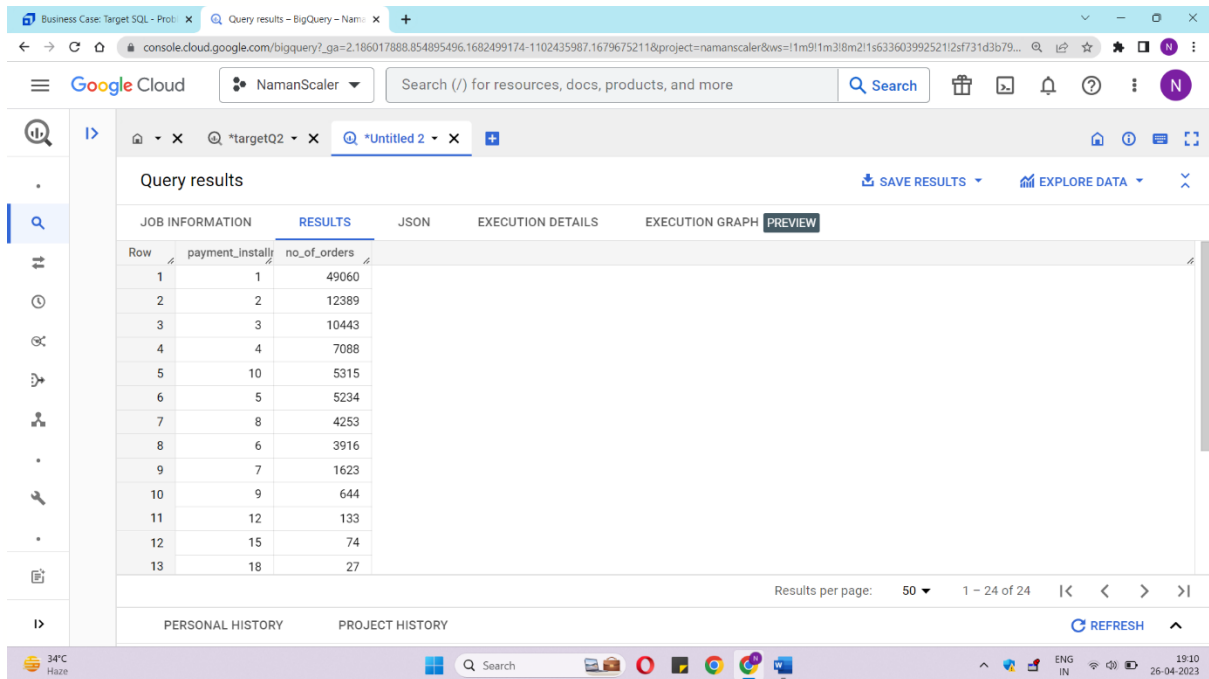
From above data we can clearly see monthwise count of orders for different payment types and then we can infer which mode of payment is growing and here we can see that credit card is mostly used as payment\_type for orders.



## Q6.2 Count of orders based on the no. of payment installments.

Ans-

```
select payment_installments, count (distinct order_id) as no_of_orders
from `Target_Database.payments`
group by payment_installments
order by no_of_orders desc
```



Query results

| Row | payment_installments | no_of_orders |
|-----|----------------------|--------------|
| 1   | 1                    | 49060        |
| 2   | 2                    | 12389        |
| 3   | 3                    | 10443        |
| 4   | 4                    | 7088         |
| 5   | 10                   | 5315         |
| 6   | 5                    | 5234         |
| 7   | 8                    | 4253         |
| 8   | 6                    | 3916         |
| 9   | 7                    | 1623         |
| 10  | 9                    | 644          |
| 11  | 12                   | 133          |
| 12  | 15                   | 74           |
| 13  | 18                   | 27           |

## Insight:

From above data we can see no of payment installments preferred by most of customers. And we can see that most payment\_installments preferred are of 1,2,3.

## **Recommendations-**

1. we should try to increase sales on months where no of orders are less by adopting an appropriate strategy.
2. We can also promote sales at night by offering midnight by offering exclusive deals so that more and more people order at night
3. For expanding our business we can focus on states where number of orders and customers are less
4. We should constantly focus on cost of orders monthwise and as it is decreasing so we need to adopt a strategy to increase cost of orders.
5. we should try to deliver the order within estimated delivery date,some states are facing delay in orders and also at some places order is delivered very early as compared to estimated delivery date ,so there is need to optimize the delivery prediction system.