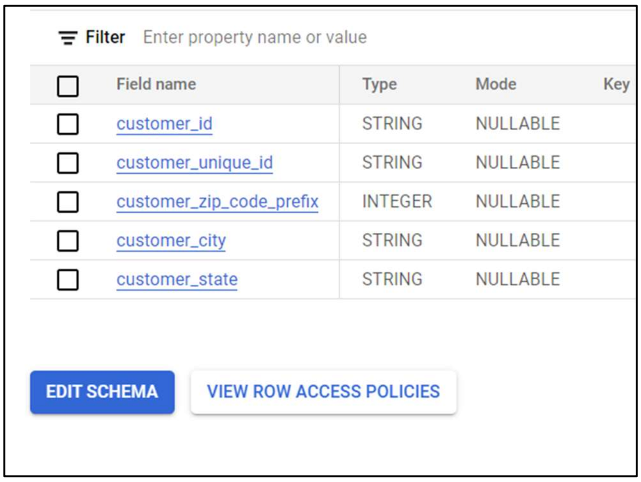
**REPORT - Business Case: Target SQL**

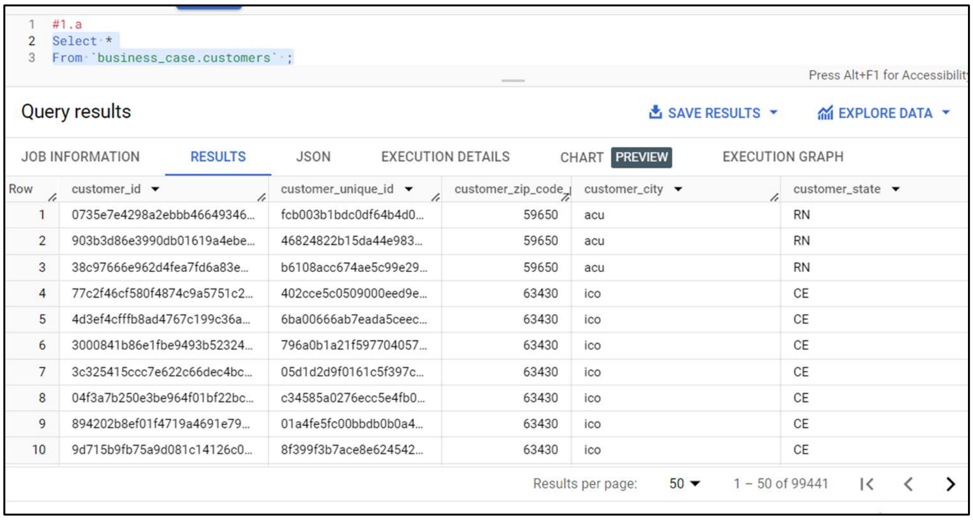
**By Mansi Sinha**

1. **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**
   **Query:** Select *
   From `business_case.customers` ;

   **Screenshot:**



| □ | 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.a
Select *
From `business_case.customers` ;
```

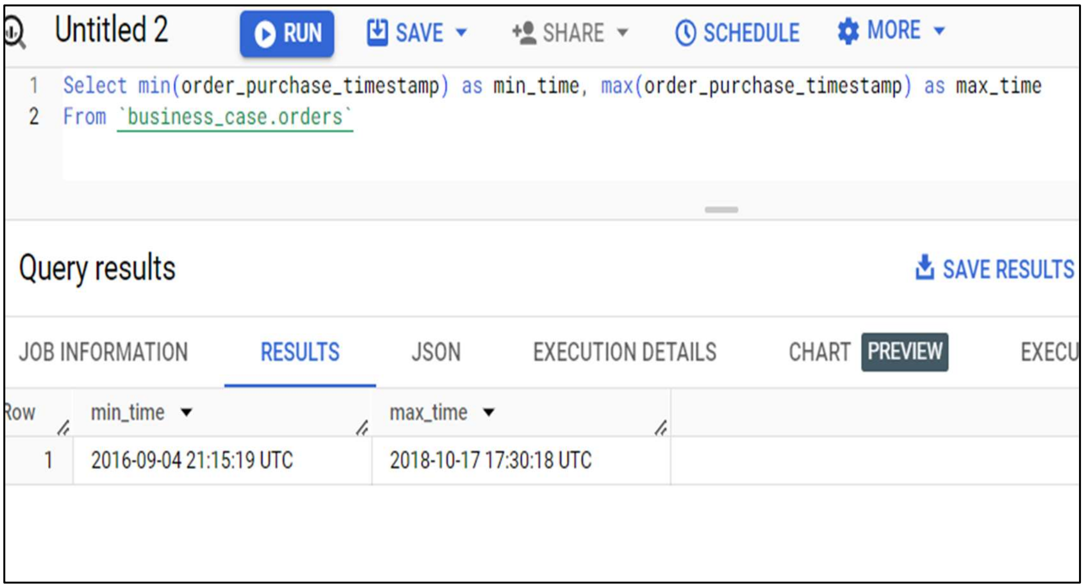
| Row | customer_id | customer_unique_id | customer_zip_code_ | customer_city | customer_state |
|---|---|---|---|---|---|
| 1 | 0735e7e4298a2ebbb46649346... | fcb003b1bdc0df64b4d0... | 59650 | acu | RN |
| 2 | 903b3d86e3990db01619a4ebe... | 46824822b15da44e983... | 59650 | acu | RN |
| 3 | 38c97666e962d4fea7fd6a83e... | b6108acc674ae5c99e29... | 59650 | acu | RN |
| 4 | 77c2f46cf580f4874c9a5751c2... | 402cce5c0509000eed9e... | 63430 | ico | CE |
| 5 | 4d3ef4cfffb8ad4767c199c36a... | 6ba00666ab7eada5ceec... | 63430 | ico | CE |
| 6 | 3000841b86e1fbe9493b52324... | 796a0b1a21f597704057... | 63430 | ico | CE |
| 7 | 3c325415ccc7e622c66dec4bc... | 05d1d2d9f0161c5f397c... | 63430 | ico | CE |
| 8 | 04f3a7b250e3be964f01bf22bc... | c34585a0276ecc5e4fb0... | 63430 | ico | CE |
| 9 | 894202b8ef01f4719a4691e79... | 01a4fe5fc00bbdb0b0a4... | 63430 | ico | CE |
| 10 | 9d715b9fb75a9d081c14126c0... | 8f399f3b7ace8e624542... | 63430 | ico | CE |

Results per page: 50 1 – 50 of 99441

**INSIGHT:** From our customers table we've identified, that table contained five attributes 'customer_id', 'customer_unique_id', 'customer_zip_code_prefix', 'customer_city', 'customer_state'. Data type of these attributes are string except customer zip code that is integer.

   b. **Get the time range between which the orders were placed**
   **QUERY:**
   Select min(order_purchase_timestamp) as min_time, max(order_purchase_timestamp) as max_time
   From `business_case.orders`

**SCREENSHOT:**



```sql
Select min(order_purchase_timestamp) as min_time, max(order_purchase_timestamp) as max_time
From `business_case.orders`
```

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

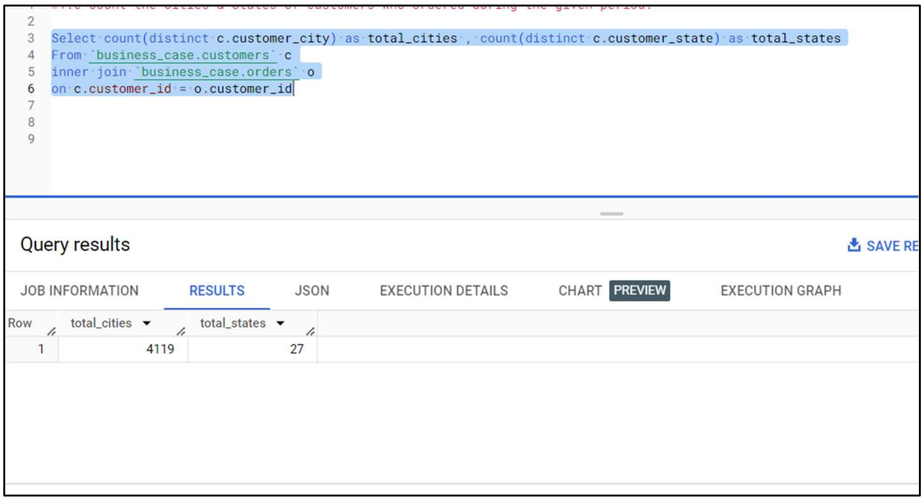
**INSIGHT:** From above order table we can show time range where our order placed. Where min() function helping us to know when were the 1st day of order was placed and max() function helping us to know last day order day.

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

**QUERY:** Select count(distinct c.customer_city) as total_cities , count(distinct c.customer_state) as total_states

From `business_case.customers` c
inner join `business_case.orders` o
on c.customer_id = o.customer_id

**SCREENSHOT:**



```sql
Select count(distinct c.customer_city) as total_cities , count(distinct c.customer_state) as total_states
From `business_case.customers` c
inner join `business_case.orders` o
on c.customer_id = o.customer_id
```

| Row | total_cities | total_states |
|---|---|---|
| 1 | 4119 | 27 |

**INSIGHT:** The aims to count the number of distinct cities and states based on customers who have made orders. The query joins the customers table with the orders table on the customer_id and retrieves a count of distinct cities and states.

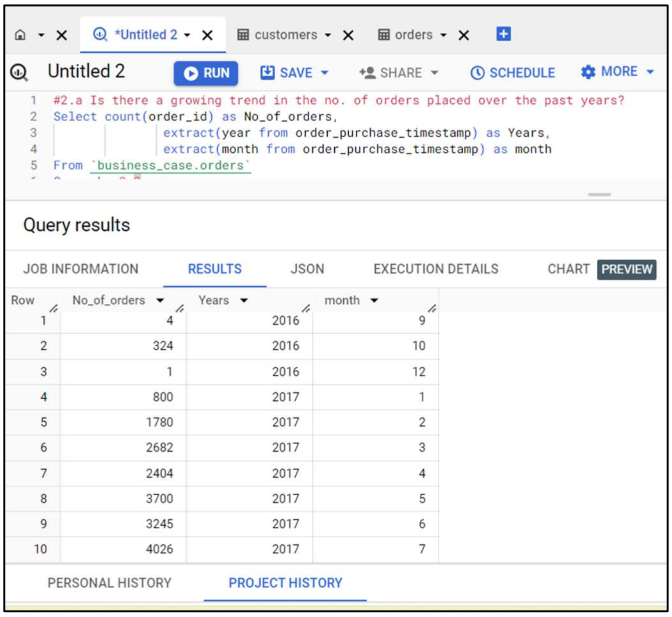
## 2. In-depth Exploration:

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

**Query:**

```
Select count(order_id) as No_of_orders,
               extract(year from order_purchase_timestamp) as Years,
               extract(month from order_purchase_timestamp) as month
From `business_case.orders`
Group by 2,3
Order by 2,3
Limit 10
```

**Screenshot:**



```
#2.a Is there a growing trend in the no. of orders placed over the past years?
Select count(order_id) as No_of_orders,
              extract(year from order_purchase_timestamp) as Years,
              extract(month from order_purchase_timestamp) as month
From `business_case.orders`
```

| Row | No_of_orders | Years | month |
|---|---|---|---|
| 1 | 4 | 2016 | 9 |
| 2 | 324 | 2016 | 10 |
| 3 | 1 | 2016 | 12 |
| 4 | 800 | 2017 | 1 |
| 5 | 1780 | 2017 | 2 |
| 6 | 2682 | 2017 | 3 |
| 7 | 2404 | 2017 | 4 |
| 8 | 3700 | 2017 | 5 |
| 9 | 3245 | 2017 | 6 |
| 10 | 4026 | 2017 | 7 |

**Insight:**

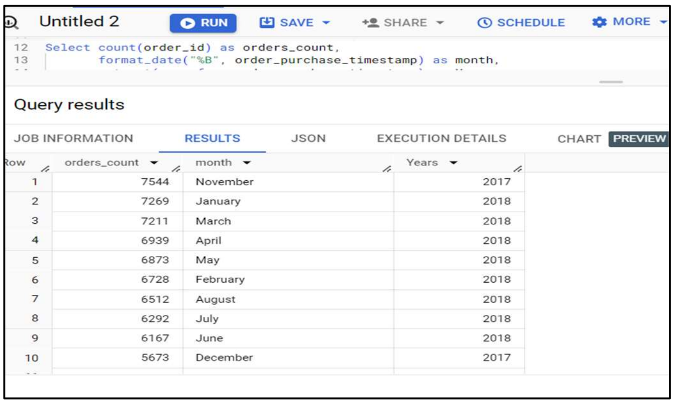
It counts the number of orders from the orders table. It extracts the year and month from the order_purchase_timestamp column. The results are grouped by the extracted year and month. Finally, the results are sorted by year and then month

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

**Query:**

```
Select count(order_id) as orders_count,
       format_date("%B", order_purchase_timestamp) as month,
       extract(year from order_purchase_timestamp) as Years
From `business_case.orders`
Group by 2,3
Order by 1 desc
```

**Screenshot:**



```
Select count(order_id) as orders_count,
       format_date("%B", order_purchase_timestamp) as month,
```

| Row | orders_count | month | Years |
|---|---|---|---|
| 1 | 7544 | November | 2017 |
| 2 | 7269 | January | 2018 |
| 3 | 7211 | March | 2018 |
| 4 | 6939 | April | 2018 |
| 5 | 6873 | May | 2018 |
| 6 | 6728 | February | 2018 |
| 7 | 6512 | August | 2018 |
| 8 | 6292 | July | 2018 |
| 9 | 6167 | June | 2018 |
| 10 | 5673 | December | 2017 |

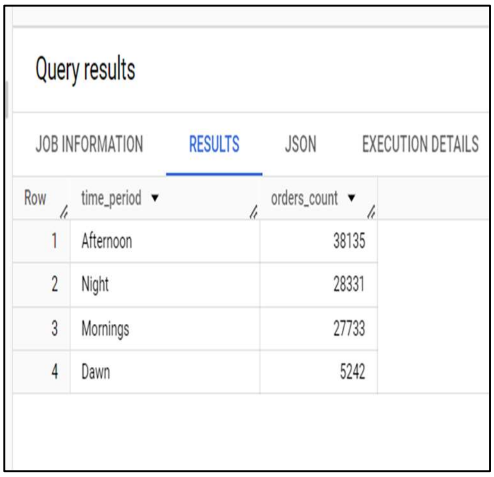
**Insight:**
The query results in a list of months and years along with their corresponding order counts. The data is sorted with the month-year combinations having the highest order counts at the top. This allows for a quick identification of peak order periods of certain month of the year.

c. **During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)**
   a. **0-6 hrs : Dawn**
   b. **7-12 hrs : Mornings**
   c. **13-18 hrs : Afternoon**
   d. **19-23 hrs : Night**

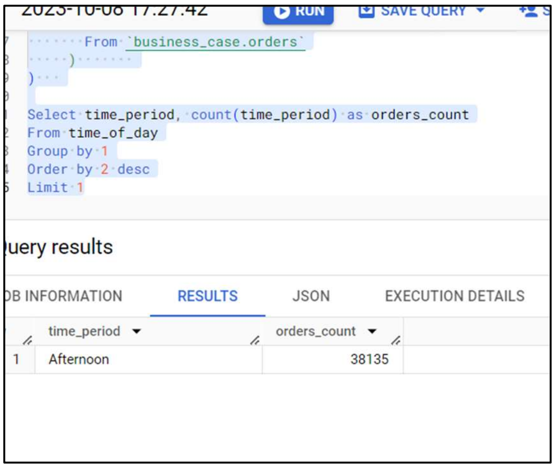
**Query:**

```
with time_of_day as
(
Select
        case
           when hours >=0 and hours <= 6 then 'Dawn'
           when hours >=7 and hours <= 12 then 'Mornings'
           when hours >=13 and hours <= 18 then 'Afternoon'
           when hours >=19 and hours <= 23 then 'Night'
        end as time_period
From (
        Select
        extract(hour from order_purchase_timestamp) as hours
        From `business_case.orders`
        )
)
Select time_period, count(time_period) as orders_count
From time_of_day
Group by 1
Order by 2 desc
Limit 1
```

**Screenshot:**



| Row | time_period | orders_count |
|---|---|---|
| 1 | Afternoon | 38135 |
| 2 | Night | 28331 |
| 3 | Mornings | 27733 |
| 4 | Dawn | 5242 |



```
      From `business_case.orders`
    )
)

Select time_period, count(time_period) as orders_count
From time_of_day
Group by 1
Order by 2 desc
Limit 1
```

| | time_period | orders_count |
|---|---|---|
| 1 | Afternoon | 38135 |

**Insight:**

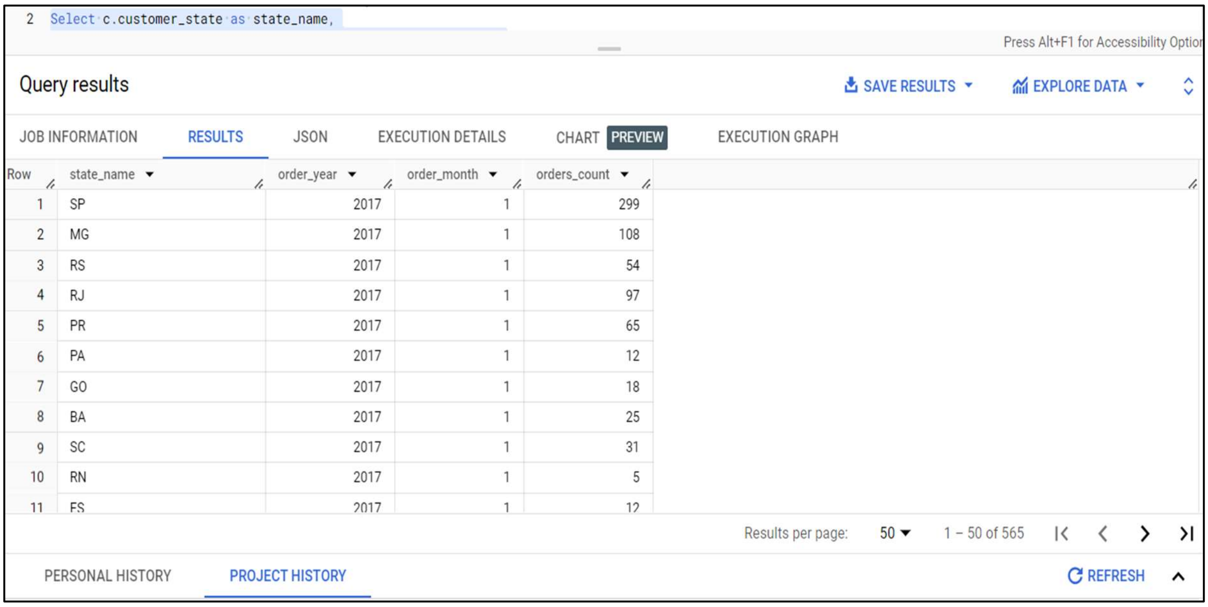
The CASE statement in the WITH clause (a Common Table Expression) assigns a time of day (Dawn, Morning, Afternoon, Night) based on the hour the order was placed. The main query then counts the number of orders for each time of day, and orders the result in descending order by count. The LIMIT 1 ensures that only the time of day with the most orders is returned, that is in **Afternoon**.

3. **Evolution of E-commerce orders in the Brazil region:**
   a. **Get the month-on-month no. of orders placed in each state.**

**Query:**

Select c.customer_state as state_name,
extract(year from o.order_purchase_timestamp) as order_year,
extract(month from o.order_purchase_timestamp) as order_month,
count(o.order_id) as orders_count
From `business_case.customers` c
join `business_case.orders` o
on c.customer_id = o.customer_id
Group By 1,2,3
Order By 3,2

**Screenshot:**



```
Select c.customer_state as state_name,
```

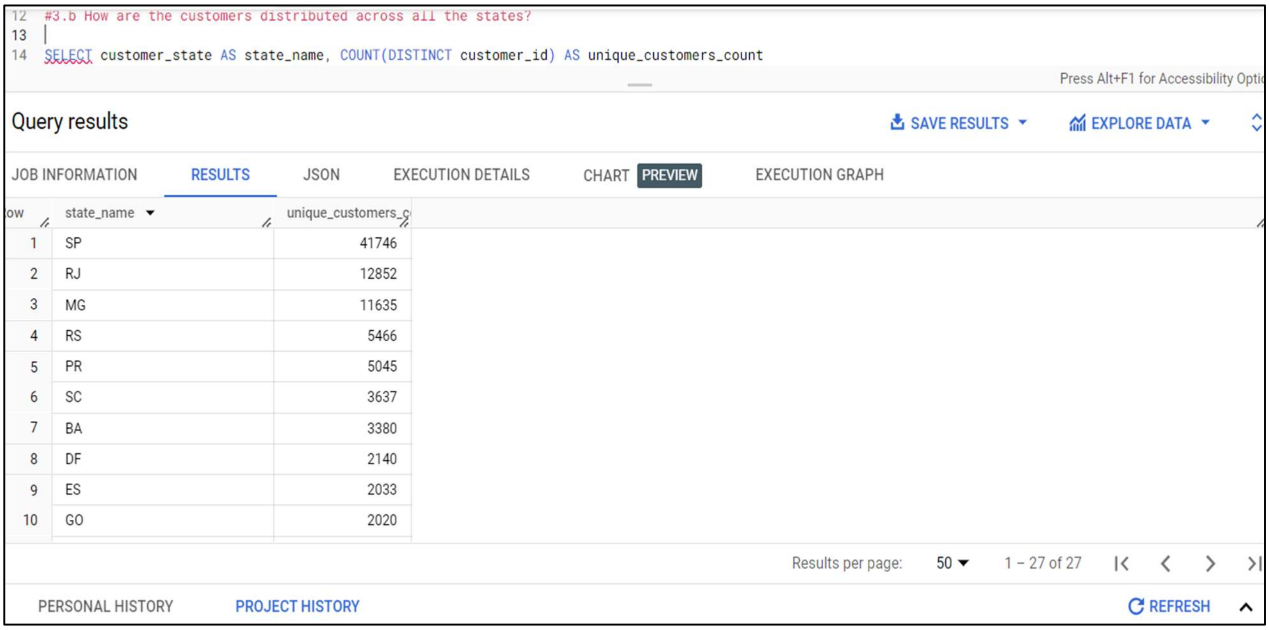
| Row | state_name | order_year | order_month | orders_count |
|---|---|---|---|---|
| 1 | SP | 2017 | 1 | 299 |
| 2 | MG | 2017 | 1 | 108 |
| 3 | RS | 2017 | 1 | 54 |
| 4 | RJ | 2017 | 1 | 97 |
| 5 | PR | 2017 | 1 | 65 |
| 6 | PA | 2017 | 1 | 12 |
| 7 | GO | 2017 | 1 | 18 |
| 8 | BA | 2017 | 1 | 25 |
| 9 | SC | 2017 | 1 | 31 |
| 10 | RN | 2017 | 1 | 5 |
| 11 | ES | 2017 | 1 | 12 |

**Insight:**

From the output of this query, we can observe the distribution of orders for every state on a month-by-month basis for each year. However, since the result is primarily ordered by month and then year, you'll see patterns like all January results for every state first, then all February results, and so on.

b. **How are the customers distributed across all the states?**
**Query:**
select customer_state as state_name, count(distinct customer_id) as unique_customers_count
from `business_case.customers`
group by customer_state
order by unique_customers_count desc

**Screenshot:**



```
#3.b How are the customers distributed across all the states?

SELECT customer_state AS state_name, COUNT(DISTINCT customer_id) AS unique_customers_count
```

| ow | state_name | unique_customers_c |
|---|---|---|
| 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 |

**Insight**:
The SQL query provided aggregates data from customers table to determine the number of unique customers present in each state. The results are sorted in descending order based on the count of unique customers, ensuring states with the highest counts are presented first.

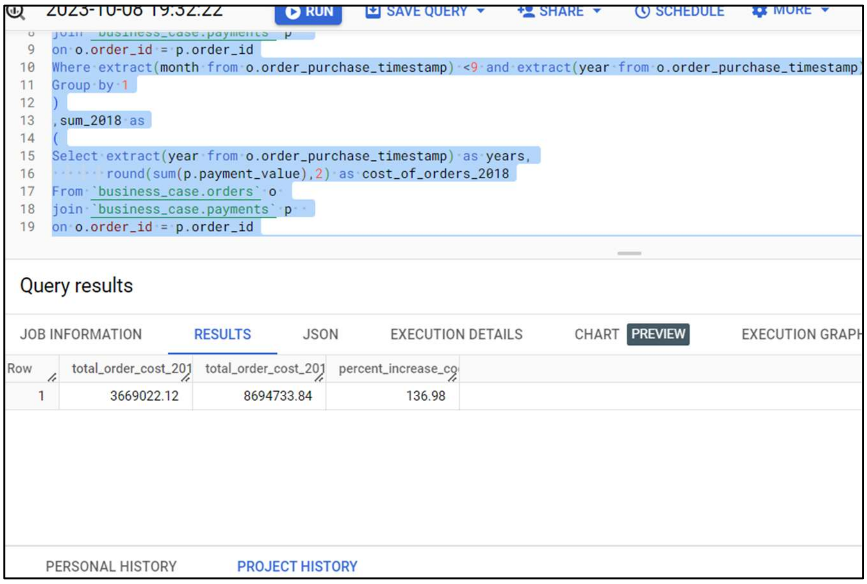
4. **Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight and others.**

a. **Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only).**
**Query:**
With sum_2017 as
(
Select extract(year from o.order_purchase_timestamp) as years,
    round(sum(p.payment_value),2) as cost_of_orders_2017
From `business_case.orders` o
join `business_case.payments` p
on o.order_id = p.order_id
Where extract(month from o.order_purchase_timestamp) <9 and extract(year from o.order_purchase_timestamp) = 2017
Group by 1
)
,sum_2018 as
(
Select extract(year from o.order_purchase_timestamp) as years,
    round(sum(p.payment_value),2) as cost_of_orders_2018
From `business_case.orders` o
join `business_case.payments` p
on o.order_id = p.order_id
Where extract(month from o.order_purchase_timestamp) <9 and extract(year from o.order_purchase_timestamp) = 2018

Group by 1
)

Select
    round((Select cost_of_orders_2017 from sum_2017),2) as total_order_cost_2017,
    round((Select cost_of_orders_2018 from sum_2018),2) as total_order_cost_2018,
    round(((Select cost_of_orders_2018 from sum_2018)/(Select cost_of_orders_2017 from sum_2017)-1)*100,2) as percent_increase_cod
From sum_2017

**Screenshot:**



```
join `business_case.payments` p
on o.order_id = p.order_id
Where extract(month from o.order_purchase_timestamp) <9 and extract(year from o.order_purchase_timestamp)
Group by 1
)
,sum_2018 as
(
Select extract(year from o.order_purchase_timestamp) as years,
      round(sum(p.payment_value),2) as cost_of_orders_2018
From `business_case.orders` o
join `business_case.payments` p
on o.order_id = p.order_id
```

| Row | total_order_cost_201 | total_order_cost_201 | percent_increase_co |
|---|---|---|---|
| 1 | 3669022.12 | 8694733.84 | 136.98 |

**Insight:**

Query aimed at calculating the total order costs for the years 2017 and 2018 (from January to August) and then computing the percentage increase in the cost of orders from 2017 to 2018.

**sum_2017 CTE:** This subquery extracts orders for the year 2017 and only considers the months from January to August (extract (month from o.order_purchase_timestamp) < 9). The subquery joins the orders table with the payments table to accumulate the total payment value for each order. The result is the rounded total payment value (cost of orders) for 2017.

**sum_2018 CTE:** Similarly, this subquery extracts orders for the year 2018, again considering only the months from January to August. Like the previous subquery, it joins the orders table with the payments table and provides the rounded total payment value (cost of orders) for 2018.

Main Query: It selects the total order costs for 2017 and 2018.

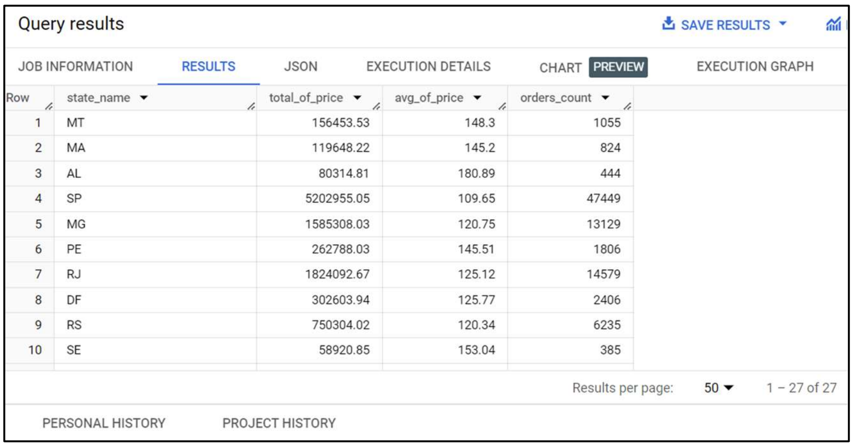
- The percentage increase in the cost of orders from 2017 to 2018 is calculated using the formula: ((cost of 2018 - cost of 2017) / cost of 2017) * 100

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

**Query:**

Select c.customer_state as state_name,
    round(sum(oi.price),2) as total_of_price,
    round(avg(oi.price),2) as avg_of_price,
    count(o.order_id) as orders_count
From `business_case.order_items` oi join `business_case.orders` o
on oi.order_id = o.order_id
join `business_case.customers` c
on c.customer_id = o.customer_id
GROUP BY 1

**Screenshot:**



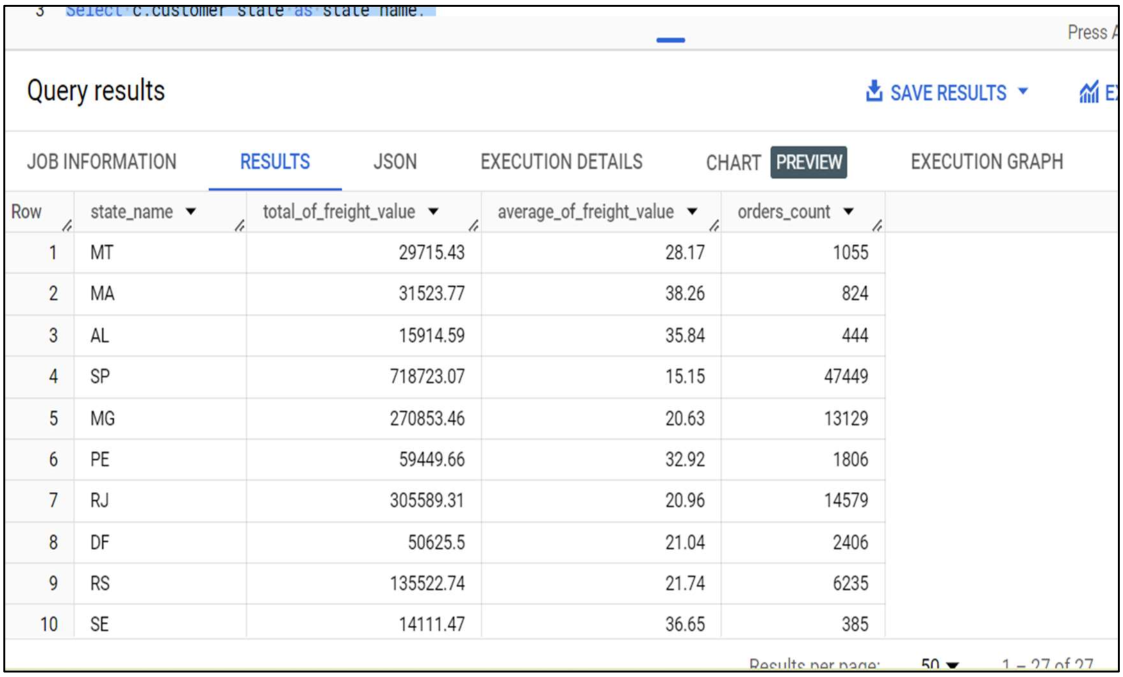
| Row | state_name | total_of_price | avg_of_price | orders_count |
|---|---|---|---|---|
| 1 | MT | 156453.53 | 148.3 | 1055 |
| 2 | MA | 119648.22 | 145.2 | 824 |
| 3 | AL | 80314.81 | 180.89 | 444 |
| 4 | SP | 5202955.05 | 109.65 | 47449 |
| 5 | MG | 1585308.03 | 120.75 | 13129 |
| 6 | PE | 262788.03 | 145.51 | 1806 |
| 7 | RJ | 1824092.67 | 125.12 | 14579 |
| 8 | DF | 302603.94 | 125.77 | 2406 |
| 9 | RS | 750304.02 | 120.34 | 6235 |
| 10 | SE | 58920.85 | 153.04 | 385 |

**Insight:**
The provided query joins the order_items, orders, and customers tables to aggregate data on the price field based on each customer state. For each state, it computes the total and average price of order items, both values rounded to two decimal places. Additionally, the query counts the number of orders associated with each state. The results are grouped by the customer's state.

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

**Query:**Select c.customer_state as state_name,
round(sum(oi.freight_value),2) as total_of_freight_value,
round(avg(oi.freight_value),2) as average_of_freight_value,
count(o.order_id) as orders_count
From `business_case.order_items` oi join `business_case.orders` o on oi.order_id = o.order_id
join `business_case.customers` c on c.customer_id = o.customer_id
GROUP BY 1

**Screenshot:**



```
select c.customer_state as state_name,
```

| Row | state_name | total_of_freight_value | average_of_freight_value | orders_count |
|---|---|---|---|---|
| 1 | MT | 29715.43 | 28.17 | 1055 |
| 2 | MA | 31523.77 | 38.26 | 824 |
| 3 | AL | 15914.59 | 35.84 | 444 |
| 4 | SP | 718723.07 | 15.15 | 47449 |
| 5 | MG | 270853.46 | 20.63 | 13129 |
| 6 | PE | 59449.66 | 32.92 | 1806 |
| 7 | RJ | 305589.31 | 20.96 | 14579 |
| 8 | DF | 50625.5 | 21.04 | 2406 |
| 9 | RS | 135522.74 | 21.74 | 6235 |
| 10 | SE | 14111.47 | 36.65 | 385 |

**Insight:**
The query joins the order_items, orders, and customers tables using the respective order_id and customer_id fields. It then calculates the total and average freight value (both rounded to two decimal places) for each customer state. Additionally, it counts the number of orders for each state. The results are grouped by the customer's state, providing insights into shipping costs and order counts per state.

5. **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.**

**Query:**

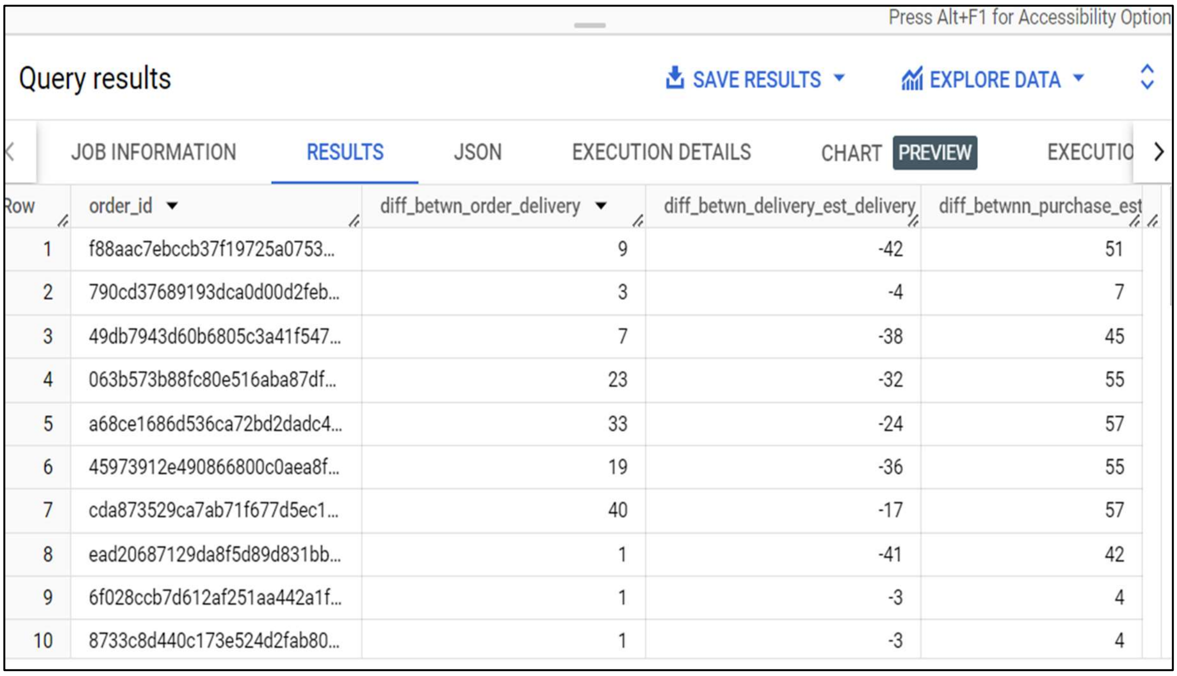
```
SELECT
o.order_id,
DATETIME_DIFF(EXTRACT(date FROM o.order_delivered_carrier_date),
              EXTRACT(date FROM o.order_purchase_timestamp),
              day) AS diff_betwn_order_delivery,

DATETIME_DIFF(EXTRACT(date FROM o.order_delivered_carrier_date),
          EXTRACT(date FROM o.order_estimated_delivery_date),
          day) AS diff_betwn_delivery_est_delivery,

DATETIME_DIFF(EXTRACT(date FROM o.order_estimated_delivery_date),
          EXTRACT(date FROM o.order_purchase_timestamp),
          day) AS diff_betwnn_purchase_est_delivery

FROM `business_case.orders` o
```

**Screenshot:**



| Row | order_id | diff_betwn_order_delivery | diff_betwn_delivery_est_delivery | diff_betwnn_purchase_est |
|---|---|---|---|---|
| 1 | f88aac7ebccb37f19725a0753... | 9 | -42 | 51 |
| 2 | 790cd37689193dca0d00d2feb... | 3 | -4 | 7 |
| 3 | 49db7943d60b6805c3a41f547... | 7 | -38 | 45 |
| 4 | 063b573b88fc80e516aba87df... | 23 | -32 | 55 |
| 5 | a68ce1686d536ca72bd2dadc4... | 33 | -24 | 57 |
| 6 | 45973912e490866800c0aea8f... | 19 | -36 | 55 |
| 7 | cda873529ca7ab71f677d5ec1... | 40 | -17 | 57 |
| 8 | ead20687129da8f5d89d831bb... | 1 | -41 | 42 |
| 9 | 6f028ccb7d612af251aa442a1f... | 1 | -3 | 4 |
| 10 | 8733c8d440c173e524d2fab80... | 1 | -3 | 4 |

**Insight:**

The difference between the date the order was delivered to the carrier and the date the order was purchased. This gives an understanding of how long it took for the order to be processed and handed over to the carrier after being placed.

The difference between the date the order was delivered to the carrier and the order's estimated delivery date. This helps assess how the actual delivery date by the carrier compared with the estimated delivery date.

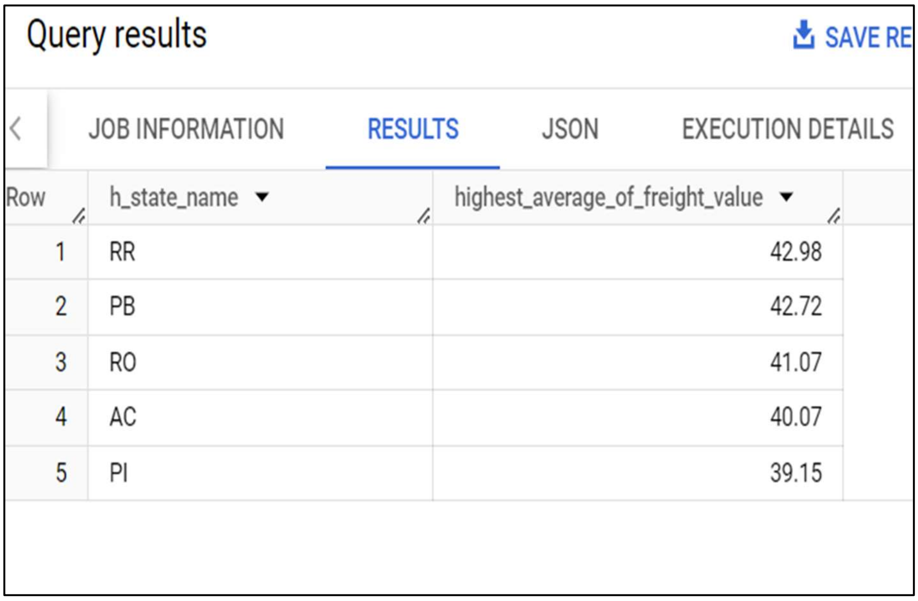
The difference between the order's estimated delivery date and the date of purchase. This provides insight into the initially anticipated waiting period for the customer, from when they purchased the product to when they expected to receive it.

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

**Query:**

```
Select c.customer_state as h_state_name,
    round(avg(oi.freight_value),2) as highest_average_of_freight_value
From `business_case.order_items` oi join `business_case.orders` o on oi.order_id =
o.order_id
                    join `business_case.customers` c on c.customer_id = o.customer_id
Group by h_state_name
Order by highest_average_of_freight_value desc
Limit 5
```

**Screenshot:**



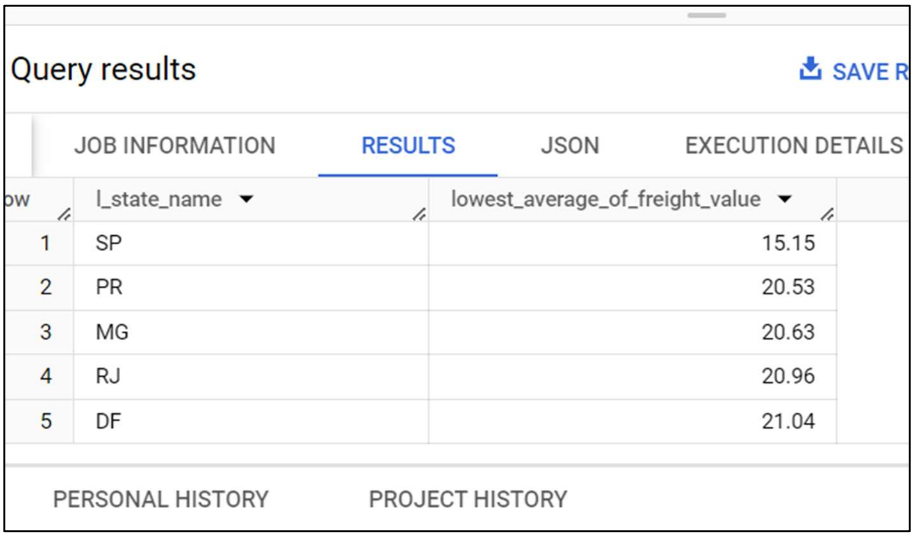
Query results

| Row | h_state_name | highest_average_of_freight_value |
|---|---|---|
| 1 | RR | 42.98 |
| 2 | PB | 42.72 |
| 3 | RO | 41.07 |
| 4 | AC | 40.07 |
| 5 | PI | 39.15 |

**Query:**

```
Select c.customer_state as l_state_name,
    round(avg(oi.freight_value),2) as lowest_average_of_freight_value
From `business_case.order_items` oi join `business_case.orders` o on oi.order_id =
o.order_id
                    join `business_case.customers` c on c.customer_id = o.customer_id
Group by l_state_name
Order by lowest_average_of_freight_value asc
Limit 5
```

**Screenshot:**



| ow | l_state_name | lowest_average_of_freight_value |
|---|---|---|
| 1 | SP | 15.15 |
| 2 | PR | 20.53 |
| 3 | MG | 20.63 |
| 4 | RJ | 20.96 |
| 5 | DF | 21.04 |

**Insight:** aimed at retrieving the top 5 states with the highest average freight values and the top 5 states with the lowest average freight values.

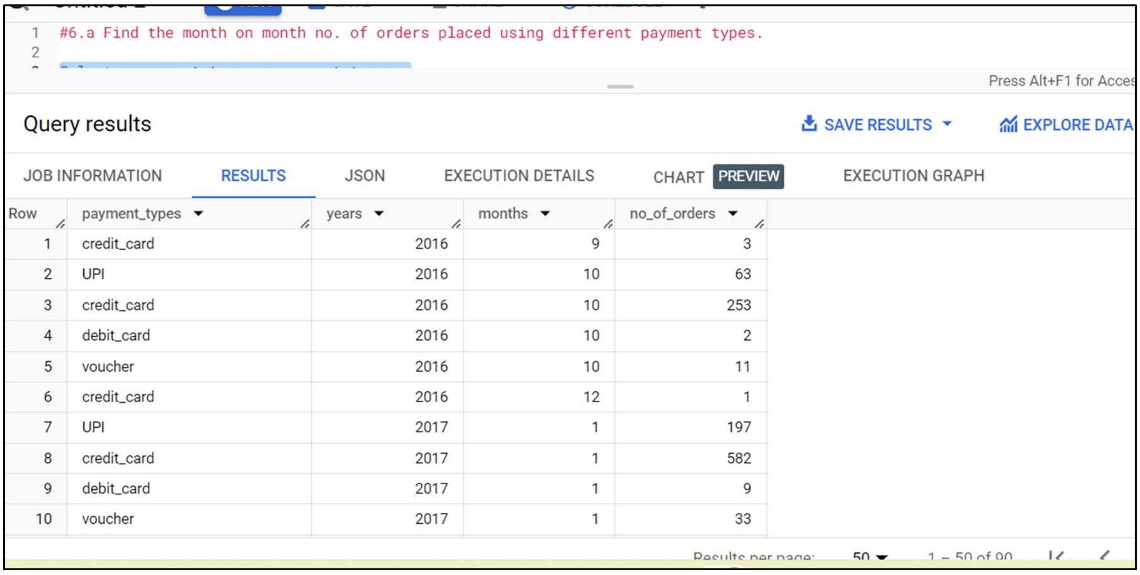
6. **Analysis based on the payments:**

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

**Query:**

```
Select p.payment_type as payment_types,
       extract(year from o.order_purchase_timestamp) as years,
       extract(month from o.order_purchase_timestamp) as months,
       count(distinct o.order_id) as no_of_orders
From `business_case.orders` o
join `business_case.payments` p
on o.order_id = p.order_id
Group by payment_types, months, years
Order by years, months, payment_types
```

**Screenshot:**



```
#6.a Find the month on month no. of orders placed using different payment types.
```

| Row | payment_types | years | months | no_of_orders |
|---|---|---|---|---|
| 1 | credit_card | 2016 | 9 | 3 |
| 2 | UPI | 2016 | 10 | 63 |
| 3 | credit_card | 2016 | 10 | 253 |
| 4 | debit_card | 2016 | 10 | 2 |
| 5 | voucher | 2016 | 10 | 11 |
| 6 | credit_card | 2016 | 12 | 1 |
| 7 | UPI | 2017 | 1 | 197 |
| 8 | credit_card | 2017 | 1 | 582 |
| 9 | debit_card | 2017 | 1 | 9 |
| 10 | voucher | 2017 | 1 | 33 |

**Insight:**

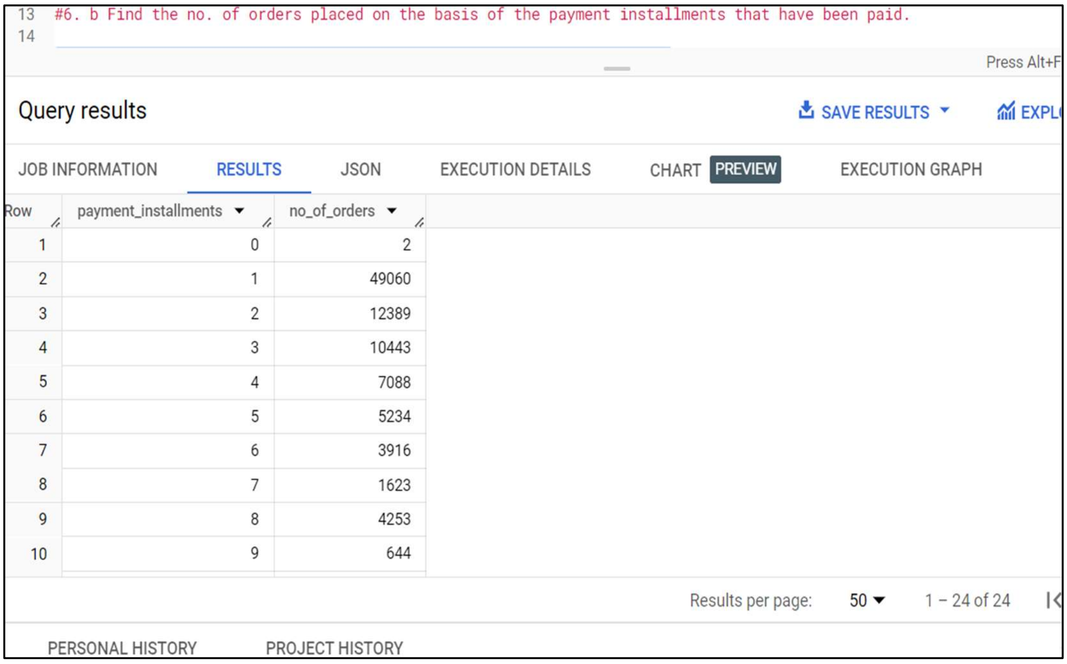
The query extracts and combines data from the orders and payments tables in the business_case database. Specifically, it categorizes and counts the number of orders based on their payment types and the month and year of the order purchase. The results are grouped by payment method, month, and year, then sorted chronologically by year and month, and further by payment type. This provides a structured overview of order counts by payment methods across different months and years.

b. **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 no_of_orders
From `business_case.payments`
Group By 1
Order By 1

**Screenshot:**



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

| Row | payment_installments | no_of_orders |
|---|---|---|
| 1 | 0 | 2 |
| 2 | 1 | 49060 |
| 3 | 2 | 12389 |
| 4 | 3 | 10443 |
| 5 | 4 | 7088 |
| 6 | 5 | 5234 |
| 7 | 6 | 3916 |
| 8 | 7 | 1623 |
| 9 | 8 | 4253 |
| 10 | 9 | 644 |

Results per page: 50 1 – 24 of 24

**Insight:**

The query examines the payments table in the business_case database to categorize and count the number of unique orders based on their payment installments. The results are grouped and sorted by the number of installments.