**PROJECT:**TARGET SQL BUSINESS CASE

**NAME:**BHUVANESWARAN.S

**BATCH:**DSML OCT24(1)BEGINNER2

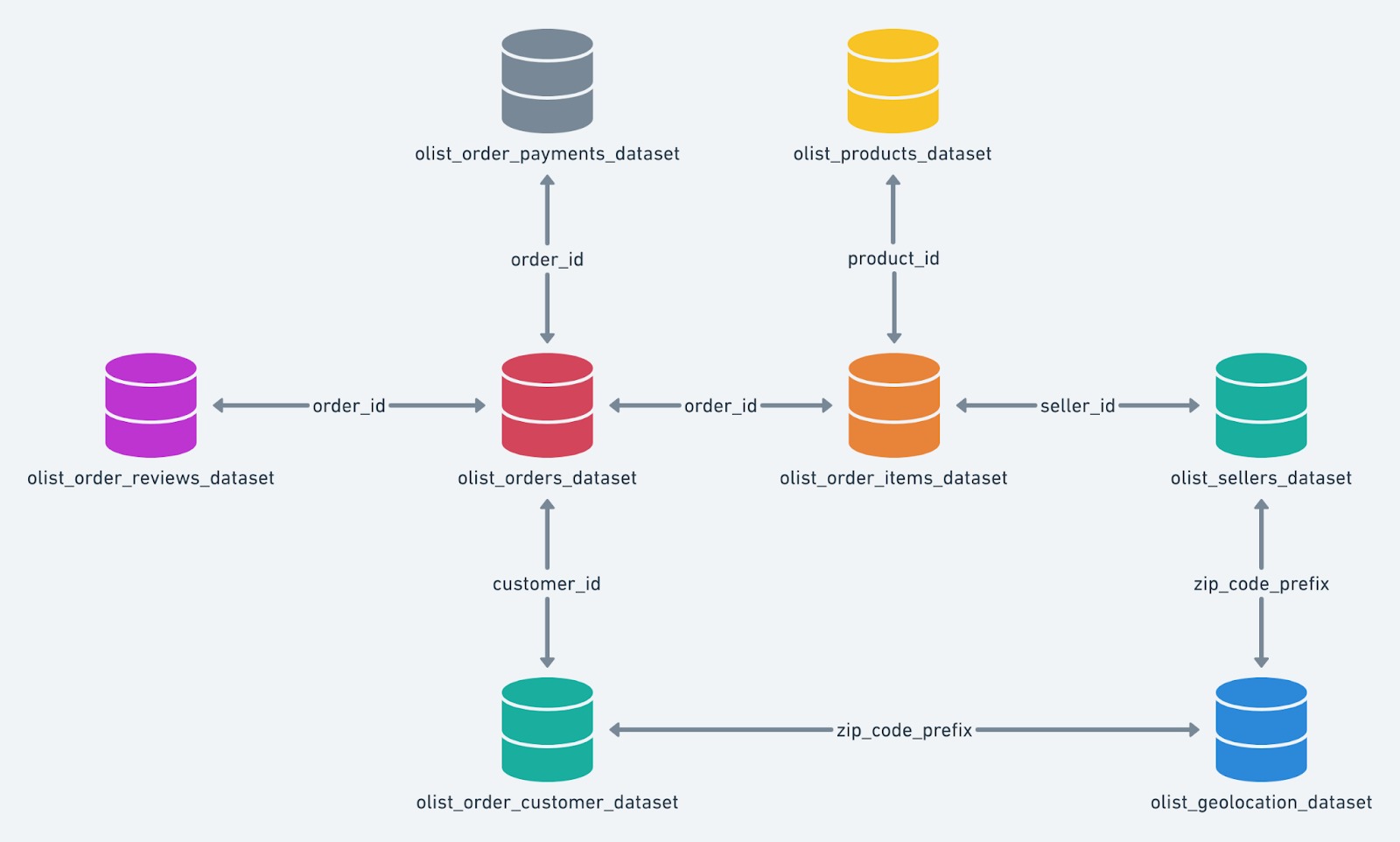
**CONTEXT:**

Target is a globally renowned brand and a prominent retailer in the United States. Target makes itself a preferred shopping destination by offering outstanding value, inspiration, innovation and an exceptional guest experience that no other retailer can deliver.

This particular business case focuses on the operations of Target in Brazil and provides insightful information about 100,000 orders placed between 2016 and 2018. The dataset offers a comprehensive view of various dimensions including the order status, price, payment and freight performance, customer location, product attributes, and customer reviews.

By analyzing this extensive dataset, it becomes possible to gain valuable insights into Target's operations in Brazil. The information can shed light on various aspects of the business, such as order processing, pricing strategies, payment and shipping efficiency, customer demographics, product characteristics, and customer satisfaction levels.

**DATASET SCHEMA:**

****

1. **Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset:**
2. Data type of all columns in the "customers" table?

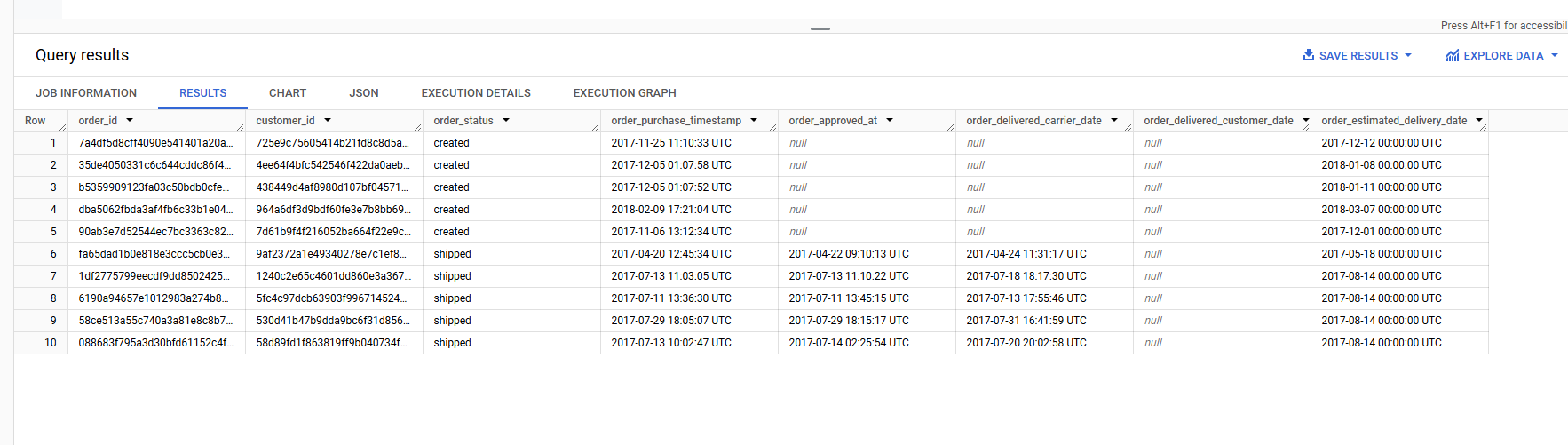
**QUERY:**

select\*

from  Ecommerce.orders

limit 10;

**OUTPUT:**



**INSIGHT:**

* These are the all columns of customers table within the “limit of 10”

|  |  |  |
| --- | --- | --- |
|  |  |  |

1. **Get the time range between which the orders were placed?**

**QUERY:**

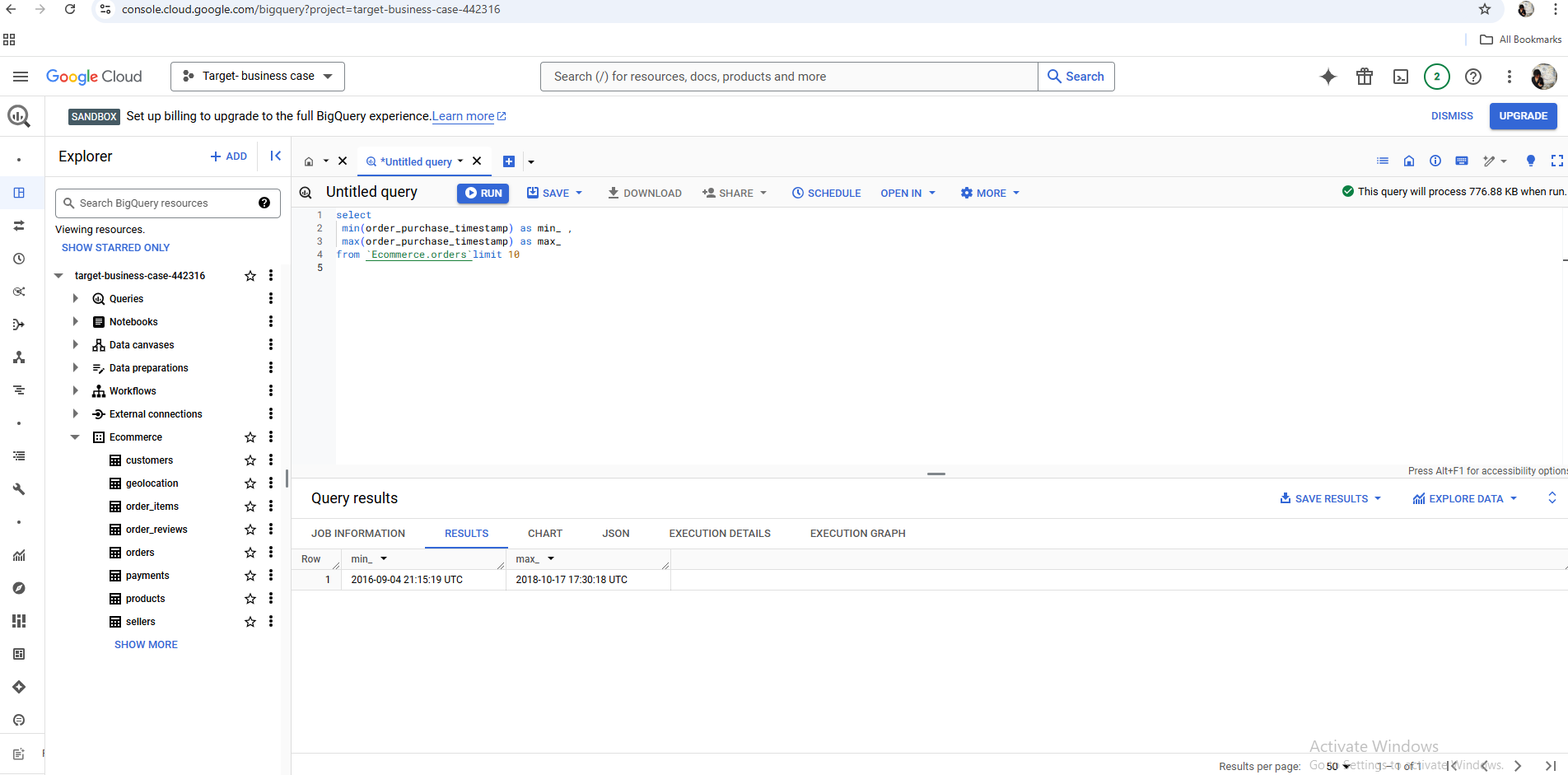
select

 min(order\_purchase\_timestamp) as min\_ ,

 max(order\_purchase\_timestamp) as max\_

from `Ecommerce.orders`limit 10

**OUTPUT:**

****

**INSIGHT:**

* Minumum time range of the order is = 2016-09-04 21:15:19 UTC
* Maximum time range of the order is= 2018-10-17 17:30:18 UTC

1. **Count the Cities & States of customers who ordered during the given period?**

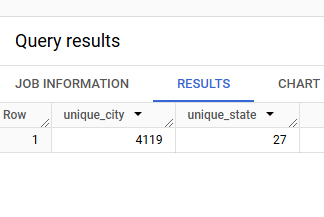
**QUERY:**

select count(distinct customer\_city)as unique\_city,

 count(distinct customer\_state)as unique\_state,

 from `Ecommerce.customers`;

**OUTPUT:**

****

1. **In-depth Exploration:**
2. Is there a growing trend in the no. of orders placed over the past years?

**QUERY:**

SELECT

EXTRACT(YEAR FROM order\_purchase\_timestamp) AS order\_year,

    COUNT(\*) AS total\_orders

FROM

   `Ecommerce.orders`

WHERE

    order\_purchase\_timestamp BETWEEN '2016-01-01' AND '2018-12-31'

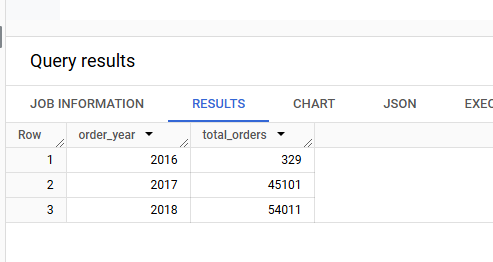
GROUP BY

    EXTRACT(YEAR FROM order\_purchase\_timestamp)

ORDER BY

    order\_year;

**OUTPUT:**



**INSIGHT:**

* This would indicate that the number of orders increased from 329 in 2016 to 54011 in 2018

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

**QUERY:**

SELECT

    EXTRACT(YEAR FROM order\_purchase\_timestamp) AS order\_year,

    EXTRACT(MONTH FROM order\_purchase\_timestamp) AS order\_month,

    COUNT(\*) AS total\_orders

FROM

    `Ecommerce.orders`

WHERE

    order\_purchase\_timestamp BETWEEN '2016-01-01' AND '2018-12-31'

GROUP BY

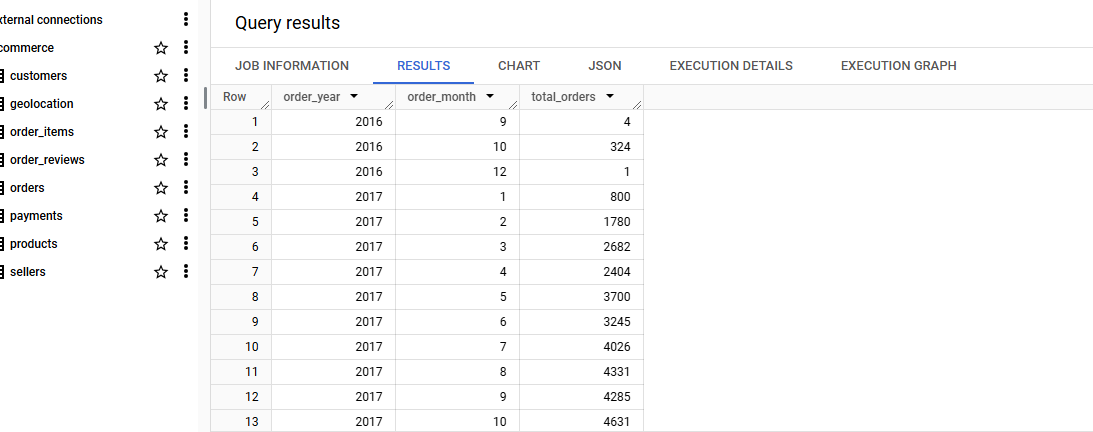
    EXTRACT(YEAR FROM order\_purchase\_timestamp),

    EXTRACT(MONTH FROM order\_purchase\_timestamp)

ORDER BY

    order\_year, order\_month;

**OUTPUT:**

****

**INSIGHT:**

* This will shows the number of orders placed months of each years 2016 to 2018

**3** .During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)

0-6 hrs : Dawn

7-12 hrs : Mornings

13-18 hrs : Afternoon

19-23 hrs : Night

**QUERY:** SELECT

    CASE

        WHEN EXTRACT(HOUR FROM order\_purchase\_timestamp) BETWEEN 0 AND 6 THEN 'Dawn'

        WHEN EXTRACT(HOUR FROM order\_purchase\_timestamp) BETWEEN 7 AND 12 THEN 'Morning'

        WHEN EXTRACT(HOUR FROM order\_purchase\_timestamp) BETWEEN 13 AND 18 THEN 'Afternoon'

        WHEN EXTRACT(HOUR FROM order\_purchase\_timestamp) BETWEEN 19 AND 23 THEN 'Night'

    END AS time\_of\_day,

    COUNT(\*) AS total\_orders

FROM

    `Ecommerce.orders`

WHERE

    order\_purchase\_timestamp BETWEEN '2016-01-01' AND '2018-12-31'

GROUP BY

    time\_of\_day

ORDER BY

    CASE

        WHEN time\_of\_day = 'Dawn' THEN 1

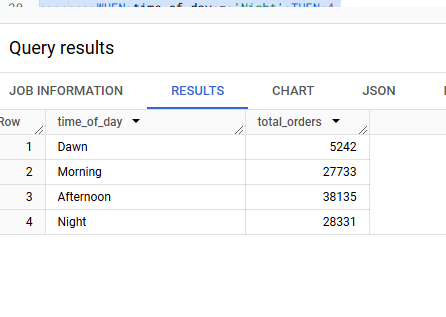
        WHEN time\_of\_day = 'Morning' THEN 2

        WHEN time\_of\_day = 'Afternoon' THEN 3

        WHEN time\_of\_day = 'Night' THEN 4

    END;

**OUTPUT:**

****

**INSIGHT:**

* This indicate the total no of orders (drawn,morning,afternoon,and night)

3).**Evolution of E-commerce orders in the Brazil region:**

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

**QUERY:**

SELECT

    EXTRACT(YEAR FROM o.order\_purchase\_timestamp) AS order\_year,

    EXTRACT(MONTH FROM o.order\_purchase\_timestamp) AS order\_month,

    c.customer\_state AS state,

    COUNT(o.order\_id) AS total\_orders

FROM

    Ecommerce.orders o

JOIN

    Ecommerce.customers c

ON

    o.customer\_id = c.customer\_id

WHERE

    o.order\_purchase\_timestamp BETWEEN '2016-01-01' AND '2018-12-31'  -- Adjust the date range as needed

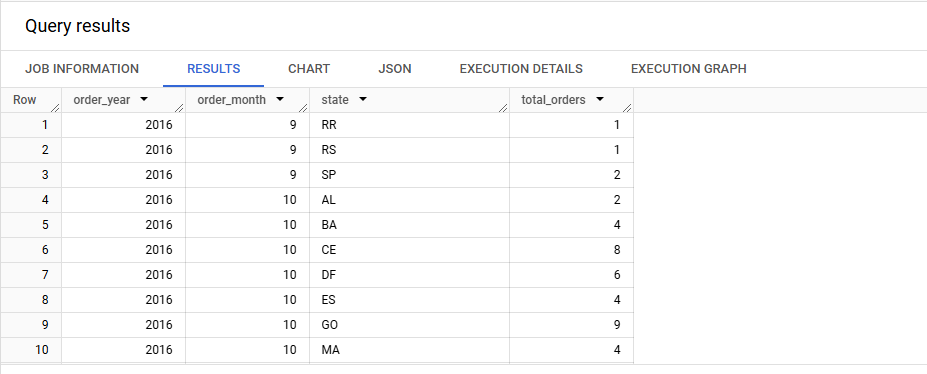
GROUP BY

    order\_year, order\_month, c.customer\_state

ORDER BY

    order\_year, order\_month, c.customer\_state;

**OUTPUT:**

****

**INSIGHT:**

* You can modify the date range in the WHERE clause to analyze a different period.
* If you want to further analyze the data, you can add filters such as order\_status or payment\_type to focus on specific types of orders

**2 How are the customers distributed across all the states?**

**QUERY:**

SELECT

  customer\_state,

  COUNT(DISTINCT customer\_id) AS customer\_count,

FROM

  `Ecommerce.customers`

  where customer\_state is not null

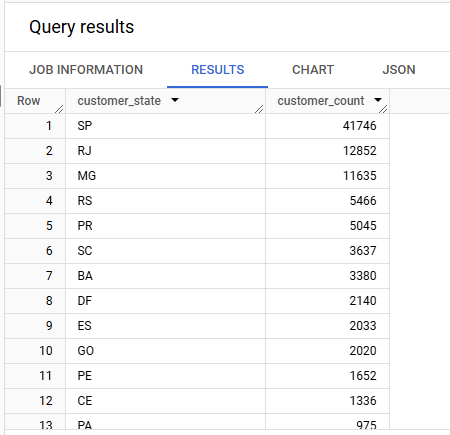
GROUP BY

  customer\_state

ORDER BY

  customer\_count DESC;

**OUTPUT:**



**INSIGHT:**

* the states where customer located (sp and rj)

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

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

**You can use the "payment\_value" column in the payments table to get the cost of orders.**

**QUERY:**

WITH OrderCostByYear AS (

  -- Calculate the total payment value for each year, for months Jan-Aug

  SELECT

    EXTRACT(YEAR FROM o.order\_purchase\_timestamp) AS order\_year,

    SUM(p.payment\_value) AS total\_payment\_value

  FROM

    Ecommerce.orders o

  JOIN

    Ecommerce.payments p

  ON

    o.order\_id = p.order\_id

  WHERE

    EXTRACT(MONTH FROM o.order\_purchase\_timestamp) BETWEEN 1 AND 8  -- Months Jan-Aug

    AND EXTRACT(YEAR FROM o.order\_purchase\_timestamp) IN (2017, 2018)  -- Filter for 2017 and 2018

  GROUP BY

    order\_year

)

SELECT

  t2017.total\_payment\_value AS total\_payment\_value\_2017,

  t2018.total\_payment\_value AS total\_payment\_value\_2018,

  ((t2018.total\_payment\_value - t2017.total\_payment\_value) / t2017.total\_payment\_value) \* 100 AS percentage\_increase

FROM

  OrderCostByYear t2017

JOIN

  OrderCostByYear t2018

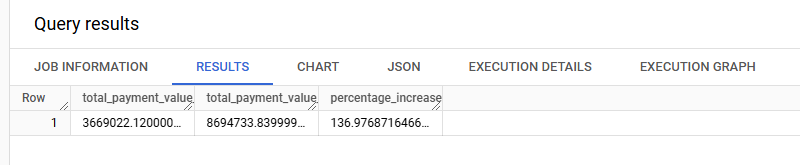
ON

  t2017.order\_year = 2017

AND

  t2018.order\_year = 2018;

**OUTPUT:**

****

**INSIGHT:**

* Ensure that the payment\_value column in the payments table is numeric and represents the total cost of the orders.
* The time range in the WHERE clause is specifically restricted to the months from January to August (months 1 through 8) for both years.

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

**QUERY:**

SELECT c.customer\_state AS state,

    SUM(p.payment\_value) AS total\_order\_value,

    AVG(p.payment\_value) AS avg\_order\_value

FROM

    Ecommerce.orders o

JOIN

    Ecommerce.payments p

ON

    o.order\_id = p.order\_id

JOIN

    Ecommerce.customers c

ON

    o.customer\_id = c.customer\_id

WHERE

    o.order\_purchase\_timestamp BETWEEN '2016-01-01' AND '2018-12-31'  -- Filter by date range if needed

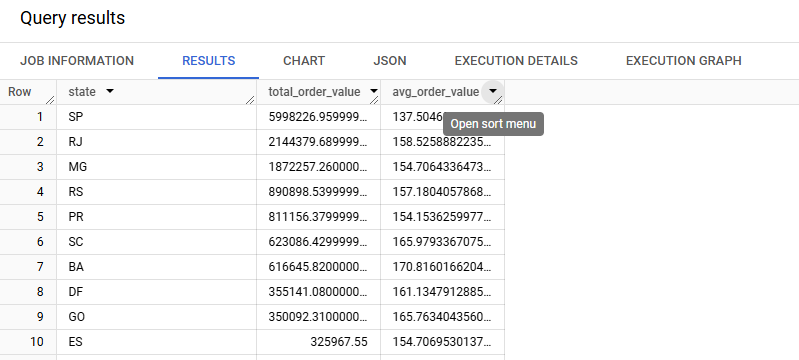
GROUP BY

    c.customer\_state

ORDER BY

    total\_order\_value DESC;

**OUTPUT:**



**INSIGHT:**

* total\_order\_value gives the sum of the payment\_value for all orders in each state.
* avg\_order\_value calculates the average price of an order in each state.
* This analysis helps understand how each state contributes to the overall revenue and provides insights into pricing trends across different regions.

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

**QUERY:**

SELECT

    c.customer\_state AS state,

    SUM(oi.freight\_value) AS total\_freight\_value,

    AVG(oi.freight\_value) AS avg\_freight\_value

FROM

    Ecommerce.orders o

JOIN

    Ecommerce.order\_items oi

ON

    o.order\_id = oi.order\_id

JOIN

    Ecommerce.customers c

ON

    o.customer\_id = c.customer\_id

WHERE

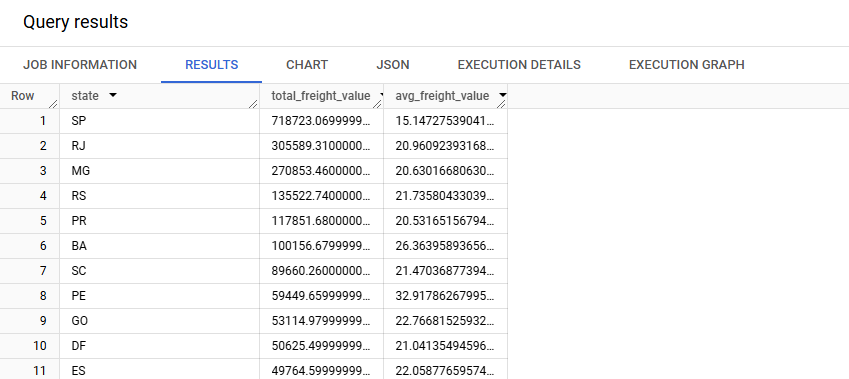
    o.order\_purchase\_timestamp BETWEEN '2016-01-01' AND '2018-12-31'  -- Filter by date range if needed

GROUP BY

    c.customer\_state

ORDER BY

    total\_freight\_value DESC;

**OUTPUT:**

**INSIGHT:**

* Freight costs may vary depending on distance, order size, and other logistics factors. Analyzing freight costs by state can help identify regions where shipping costs are disproportionately high.
* If needed, you can also further break down the freight analysis by month or year to see trends in shipping costs over time.

**5).Analysis based on sales, freight and delivery time.**

**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.You can calculate the delivery time and the difference between the estimated & actual delivery date using the given formula:time\_to\_deliver = order\_delivered\_customer\_date - order\_purchase\_timestampdiff\_estimated\_delivery = order\_delivered\_customer\_date - order\_estimated\_delivery\_date.**

**QUERY:**

SELECT

    o.order\_id,

    -- Calculate the delivery time (time taken to deliver the order)

    DATE\_DIFF(o.order\_delivered\_customer\_date, o.order\_purchase\_timestamp, DAY) AS time\_to\_deliver,

    -- Calculate the difference between the estimated and actual delivery date

    DATE\_DIFF(o.order\_delivered\_customer\_date, o.order\_estimated\_delivery\_date, DAY) AS diff\_estimated\_delivery

FROM

    Ecommerce.orders o

WHERE

    o.order\_delivered\_customer\_date IS NOT NULL  -- Ensure that delivered orders are considered

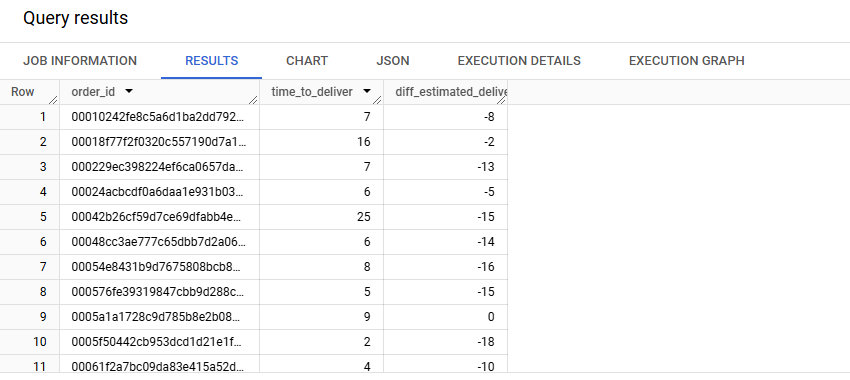
    AND o.order\_purchase\_timestamp IS NOT NULL  -- Ensure that purchase timestamp is available

    AND o.order\_estimated\_delivery\_date IS NOT NULL  -- Ensure estimated delivery date is available

ORDER BY

    o.order\_id;

**OUTPUT:**

****

**INSIGHT:**

* If there are any NULL values in the date fields (like order\_delivered\_customer\_date, order\_purchase\_timestamp, or order\_estimated\_delivery\_date), the WHERE clause will filter those orders out, ensuring we only calculate the differences for orders with complete date information.
* You can adjust the sorting or add more filters (e.g., filtering by specific years or states) depending on the level of detail you want in your analysis.

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

**QUERY:**

WITH FreightValuesByState AS (

  -- Calculate the average freight value for each state

  SELECT

    c.customer\_state AS state,

    AVG(oi.freight\_value) AS avg\_freight\_value

  FROM

    Ecommerce.orders o

  JOIN

    Ecommerce.order\_items oi

  ON

    o.order\_id = oi.order\_id

  JOIN

    Ecommerce.customers c

  ON

    o.customer\_id = c.customer\_id

  WHERE

    o.order\_purchase\_timestamp BETWEEN '2016-01-01' AND '2018-12-31'  -- You can adjust the date range as needed

  GROUP BY

    c.customer\_state

)

-- Top 5 states with highest average freight value

(SELECT state, avg\_freight\_value

FROM FreightValuesByState

ORDER BY avg\_freight\_value DESC

LIMIT 5)

UNION ALL

-- Top 5 states with lowest average freight value

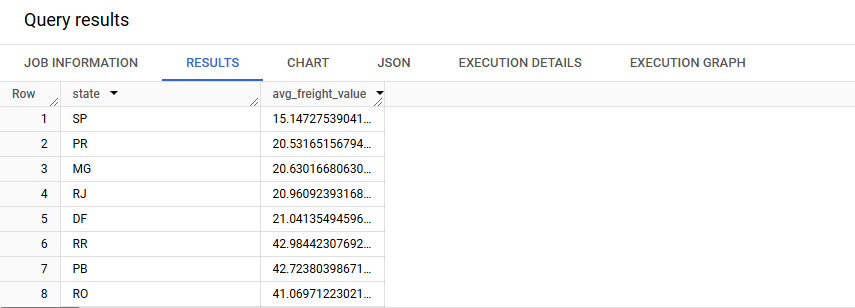
(SELECT state, avg\_freight\_value

FROM FreightValuesByState

ORDER BY avg\_freight\_value ASC

LIMIT 5);

**OUTPUT:**



**INSIGHT:**

* Freight value may vary due to factors like distance, order size, or logistics complexities. Understanding these factors can help optimize shipping strategies.
* The date range used here (2016–2018) can be modified as per the analysis period.
* If there are any NULL or missing values in the freight data, they would not be included in the calculations as the query excludes such records by default.**3.Find out the top 5 states with the highest & lowest average delivery time.**

**QUERY:**

WITH DeliveryTimesByState AS (

  -- Calculate the average delivery time for each state

  SELECT

    c.customer\_state AS state,

    AVG(DATE\_DIFF(o.order\_delivered\_customer\_date, o.order\_purchase\_timestamp, DAY)) AS avg\_delivery\_time

  FROM

    Ecommerce.orders o

  JOIN

    Ecommerce.customers c

  ON

    o.customer\_id = c.customer\_id

  WHERE

    o.order\_delivered\_customer\_date IS NOT NULL  -- Ensure only delivered orders are considered

    AND o.order\_purchase\_timestamp IS NOT NULL  -- Ensure purchase timestamp is available

    AND o.order\_delivered\_customer\_date IS NOT NULL  -- Ensure delivery date is available

  GROUP BY

    c.customer\_state

)

-- Top 5 states with highest average delivery time

(SELECT state, avg\_delivery\_time

FROM DeliveryTimesByState

ORDER BY avg\_delivery\_time DESC

LIMIT 5

)

UNION ALL

-- Top 5 states with lowest average delivery time

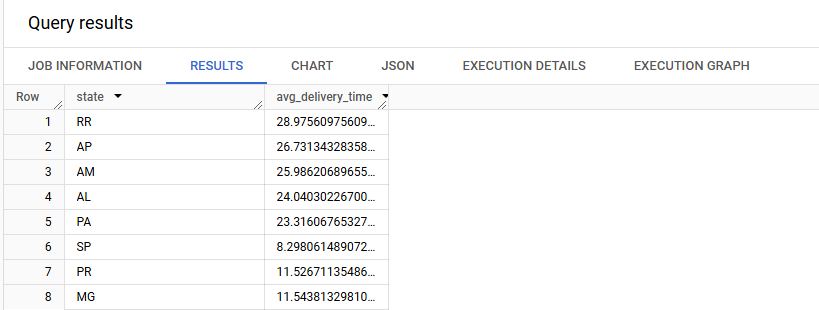
(SELECT state, avg\_delivery\_time

FROM DeliveryTimesByState

ORDER BY avg\_delivery\_time ASC

LIMIT 5)

**OUTPUT:**



**INSIGHT:**

* The delivery time is only calculated for orders that have a valid purchase timestamp and delivery date.
* The estimated delivery date is not used in the calculation of the average delivery time, but the diff\_estimated\_delivery can be calculated separately if needed.
* You can modify the date range in the WHERE clause if you want to focus on a different period of time.

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

**You can use the difference between the averages of actual & estimated delivery date to figure out how fast the delivery was for each state.**

**QUERY:**

WITH DeliverySpeedByState AS (

  -- Calculate the average difference between actual and estimated delivery dates

  SELECT

    c.customer\_state AS state,

    AVG(DATE\_DIFF(o.order\_delivered\_customer\_date, o.order\_estimated\_delivery\_date, DAY)) AS avg\_delivery\_difference

  FROM

    Ecommerce.orders o

  JOIN

    Ecommerce.customers c

  ON

    o.customer\_id = c.customer\_id

  WHERE

    o.order\_delivered\_customer\_date IS NOT NULL

    AND o.order\_estimated\_delivery\_date IS NOT NULL

    AND o.order\_purchase\_timestamp IS NOT NULL

  GROUP BY

    c.customer\_state

)

-- Top 5 states where the order delivery is faster (i.e., negative difference between actual and estimated delivery dates)

SELECT state, avg\_delivery\_difference

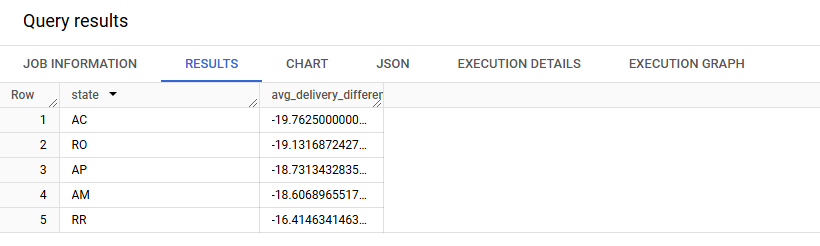
FROM DeliverySpeedByState

WHERE avg\_delivery\_difference < 0  -- Negative values indicate faster delivery

ORDER BY avg\_delivery\_difference ASC  -- Sort by the most negative values (fastest delivery)

LIMIT 5;

**OUTPUT:**



**INSIGHT:**

* This query only considers orders with both valid actual and estimated delivery dates.
* The date range used can be adjusted if needed, and additional filters (such as product category or shipping method) can be added to focus on specific types of orders.

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

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

**QUERY:**

SELECT

    EXTRACT(YEAR FROM o.order\_purchase\_timestamp) AS order\_year,

    EXTRACT(MONTH FROM o.order\_purchase\_timestamp) AS order\_month,

    p.payment\_type AS payment\_type,

    COUNT(o.order\_id) AS total\_orders

FROM

    Ecommerce.orders o

JOIN

    Ecommerce.payments p

ON

    o.order\_id = p.order\_id

WHERE

    o.order\_purchase\_timestamp BETWEEN '2016-01-01' AND '2018-12-31'  -- You can adjust the date range as needed

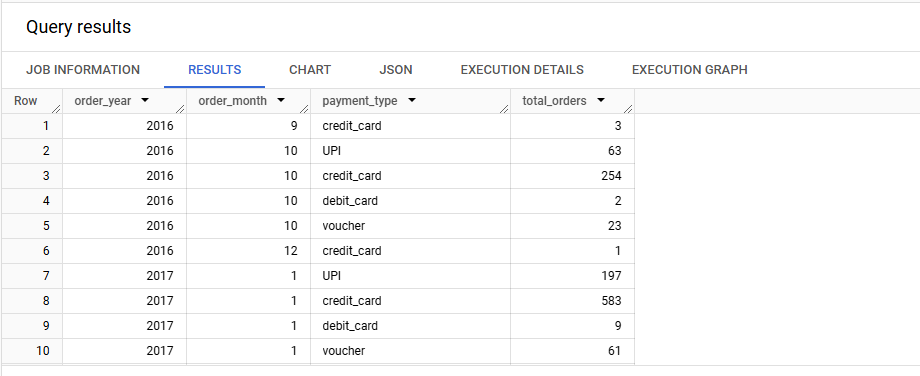
GROUP BY

    order\_year, order\_month, payment\_type

ORDER BY

    order\_year, order\_month, payment\_type;

**OUTPUT:**

****

**INSIGHT:**

* If there are any null or missing payment types, they won't be included in the results, so ensure the data is clean.
* You can modify the date range in the WHERE clause to suit your needs.

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

**QUERY:**

SELECT

    p.payment\_installments AS installment\_count,

    COUNT(o.order\_id) AS total\_orders

FROM

    Ecommerce.orders o

JOIN

    Ecommerce.payments p

ON

    o.order\_id = p.order\_id

WHERE

    p.payment\_installments IS NOT NULL  -- Ensure there are installment details

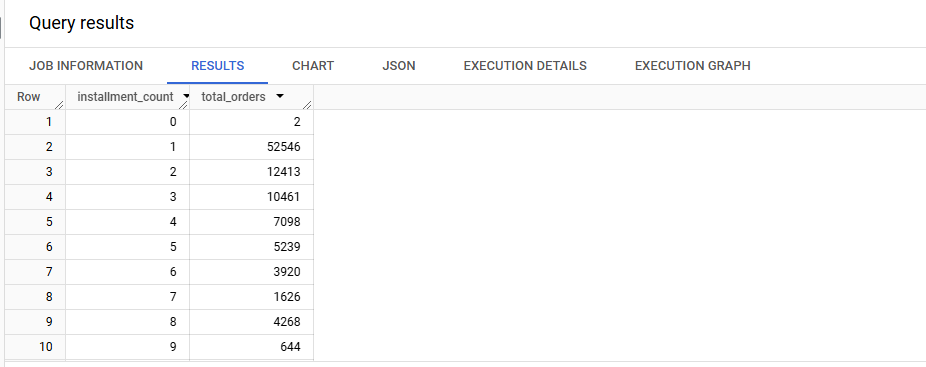
GROUP BY

    p.payment\_installments

ORDER BY

    installment\_count;

**OUTPUT:**

****

**INSIGHT:**

* Orders with NULL in the payment\_installments field will be excluded from the results.
* If a customer selects EMI (Equated Monthly Installments), the number of installments will typically be greater than 1. A single payment (no installments) will be recorded as 1 in the payment\_installments field.