| | e of all columns in the "custor customers.csv | inera table. | | |
|-----------|---|--------------|----------|--|
| | Field name | Type | Mode | |
| | customer_id | STRING | NULLABLE | |
| | customer_unique_id | STRING | NULLABLE | |
| | customer_zip_code_prefix | INTEGER | NULLABLE | |
| | customer_city | STRING | NULLABLE | |
| | customer_state | STRING | NULLABLE | |
| Table 2 - | - sellers.csv Field name | Туре | Mode | |
| | seller_id | STRING | NULLABLE | |
| | seller_zip_code_prefix | INTEGER | NULLABLE | |
| | seller_city | STRING | NULLABLE | |
| | seller_state | STRING | NULLABLE | |

03. Table 3 - order_items.csv

| Field name | Туре | Mode |
|---------------------|-----------|----------|
| order_id | STRING | NULLABLE |
| order_item_id | INTEGER | NULLABLE |
| product_id | STRING | NULLABLE |
| seller_id | STRING | NULLABLE |
| shipping_limit_date | TIMESTAMP | NULLABLE |
| price | FLOAT | NULLABLE |
| freight_value | FLOAT | NULLABLE |

04. Table 4 - geolocation.csv

| Field name | Туре | Mode |
|-----------------------------|---------|----------|
| geolocation_zip_code_prefix | INTEGER | NULLABLE |
| geolocation_lat | FLOAT | NULLABLE |
| geolocation_lng | FLOAT | NULLABLE |
| geolocation_city | STRING | NULLABLE |
| geolocation_state | STRING | NULLABLE |

05. Table 5 - payments.csv

| Field name | Туре | Mode |
|----------------------|---------|----------|
| order_id | STRING | NULLABLE |
| payment_sequential | INTEGER | NULLABLE |
| payment_type | STRING | NULLABLE |
| payment_installments | INTEGER | NULLABLE |
| payment_value | FLOAT | NULLABLE |

06. Table 6 - order_reviews.csv

| Field name | Туре | Mode |
|-------------------------|-----------|----------|
| review_id | STRING | NULLABLE |
| order_id | STRING | NULLABLE |
| review_score | INTEGER | NULLABLE |
| review_comment_title | STRING | NULLABLE |
| review_creation_date | TIMESTAMP | NULLABLE |
| review_answer_timestamp | TIMESTAMP | NULLABLE |

07. Table 7 - orders.csv

| Field name | Туре | Mode |
|-------------------------------|-----------|----------|
| order_id | STRING | NULLABLE |
| customer_id | STRING | NULLABLE |
| order_status | STRING | NULLABLE |
| order_purchase_timestamp | TIMESTAMP | NULLABLE |
| order_approved_at | TIMESTAMP | NULLABLE |
| order_delivered_carrier_date | TIMESTAMP | NULLABLE |
| order_delivered_customer_date | TIMESTAMP | NULLABLE |
| order_estimated_delivery_date | TIMESTAMP | NULLABLE |

ŀ

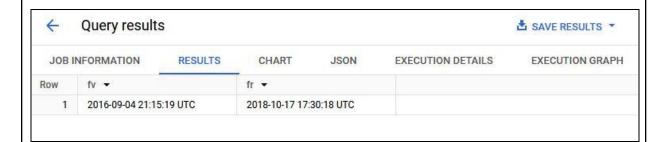
08. Table 8 - products.csv

| Field name | Type | Mode |
|----------------------------|---------|----------|
| product_id | STRING | NULLABLE |
| product_category | STRING | NULLABLE |
| product_name_length | INTEGER | NULLABLE |
| product_description_length | INTEGER | NULLABLE |
| product_photos_qty | INTEGER | NULLABLE |
| product_weight_g | INTEGER | NULLABLE |
| product_length_cm | INTEGER | NULLABLE |
| product_height_cm | INTEGER | NULLABLE |
| product_width_cm | INTEGER | NULLABLE |

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

QUERY:

```
SELECT
  MIN(order_purchase_timestamp) AS fv,
  MAX(order_purchase_timestamp) AS fr
FROM
  customer.orders;
```



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

QUERY:

SELECT

```
COUNT(DISTINCT customer_city) AS count_city,
COUNT(DISTINCT customer_state) AS count_state
```



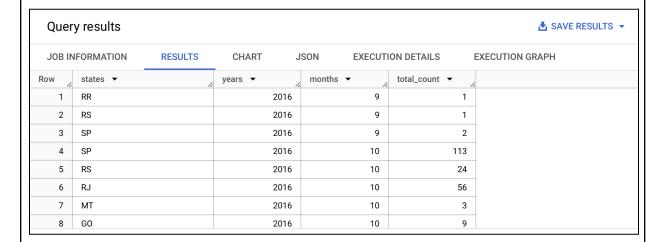
4. Is there a growing trend in the no. of orders placed over the past years?

```
SELECT
  EXTRACT(YEAR
  FROM
    order_purchase_timestamp) AS year,
  COUNT(order_id) AS count
FROM
  customer.orders
WHERE
  order_status = "delivered"
GROUP BY
  1
ORDER BY
  1;
```



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

```
QUERY:
```



6. 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
  cte AS (
  SELECT
    EXTRACT(YEAR
    FROM
       orders.order_purchase_timestamp) AS year_,
    SUM(payments.payment_value) AS revenue
FROM
  customer.orders AS orders
```

```
INNER JOIN
    customer.payments AS payments
  ON
    orders.order_id = payments.order_id
  WHERE
    EXTRACT (MONTH
    FROM
      orders.order_purchase_timestamp) BETWEEN 0
    AND 8
  GROUP BY
    Year_
  ORDER BY
    Year_)
SELECT
  LAG(revenue, 1) OVER(ORDER BY year_) AS pre_revenue,
  (revenue - (LAG(revenue, 1) OVER(ORDER BY year_)))/(LAG(revenue, 1) OVER(ORDER BY
year_))*100 AS increse_per
FROM
  Cte
  Query results

▲ SAVE RESULTS ▼

                                                                                           EXPLO
  JOB INFORMATION
                   RESULTS
                              CHART
                                               EXECUTION DETAILS
                                                                 EXECUTION GRAPH
                                       JSON
 Row
       year_ ▼
                    revenue -
                                 pre_revenue ▼
                                              increse per ▼
    1
               2017
                    3669022.120000...
```

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

136.9768716466...

0-6 hrs : Dawn
7-12 hrs : Mornings
13-18 hrs : Afternoon
19-23 hrs : Night

2018

8694733.839999...

3669022.120000...

Hint: We want you to categorize the hours of a day into the given time brackets/ intervals and find out during which intervals the Brazilian customers usually order the most.

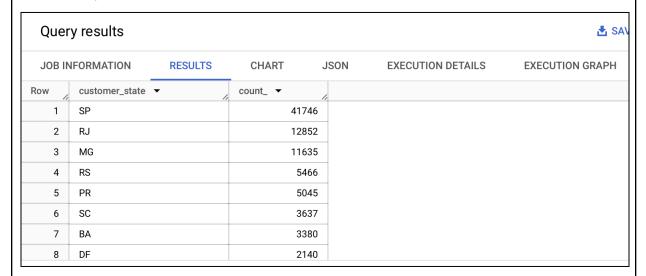
```
SELECT
   CASE
     WHEN (EXTRACT(HOUR FROM order_purchase_timestamp) >= 0) AND (EXTRACT(HOUR FROM
order_purchase_timestamp) <= 6) THEN "Dawn"
     WHEN (EXTRACT(HOUR
     FROM
       order_purchase_timestamp) >= 7)
   AND (EXTRACT(HOUR
     FROM
       order_purchase_timestamp) <= 12) THEN "Morning"</pre>
     WHEN (EXTRACT(HOUR FROM order_purchase_timestamp) >= 13) AND (EXTRACT(HOUR FROM
order_purchase_timestamp) <= 18) THEN "Afternoon"</pre>
   ELSE
   "Night"
 END
   ) AS tod,
 COUNT(order_id) AS total_count
FROM
customer.orders
GROUP BY
ORDER BY
2 DESC;
        Query results

♣ SAV

   JOB INFORMATION
                      RESULTS
                                   CHART
                                             JSON
                                                       EXECUTION DETAILS
                                                                            EXECUTION GRAPH
        tod ▼
                                 total_count ▼
 Row
                                         38135
        Afternoon
    1
     2
        Night
                                         28331
                                         27733
     3
        Mornings
        Dawn
                                          5242
```

8. How are the customers distributed across all the states?
Hint: We want you to get the no. of unique customers present in each state.

```
customer_state,
count(customer_id) as count_
FROM
customer.customers
GROUP BY
1
ORDER BY
2 DESC;
```

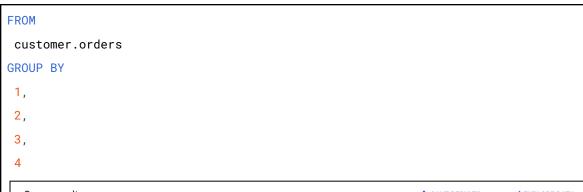


9. Calculate the Total & Average value of order price for each state.

Hint: We want you to fetch the total price and the average price of orders for each state.

```
orders.order_id,
orders.order_purchase_timestamp,
orders.order_delivered_customer_date,
orders.order_estimated_delivery_date,

TIMESTAMP_DIFF(orders.order_delivered_customer_date,orders.order_purchase_timestamp,
DAY) AS time_to_Deliver,
TIMESTAMP_DIFF(orders.order_estimated_delivery_date,
orders.order_delivered_customer_date, DAY) AS diff_estimated_delivery
```





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

Hint: We want you to find the top 5 & the bottom 5 states arranged in increasing order of the average freight value.

TOP 5 STATES WITH HIGHEST freight_value AVERAGE

```
QUERY:
```

```
SELECT
    c.customer_state,
    ROUND(AVG(oi.freight_value),2) AS AVERAGE_
FROM (customer.orders AS o
    INNER JOIN
        customer.customers AS c
    ON
        o.customer_id = c.customer_id
    INNER JOIN
        customer.order_items AS oi
    ON
        oi.order_id = o.order_id)
GROUP BY
1
```

```
ORDER BY

2 DESC

LIMIT

5
```

| Query results | | | | | | | ≛ SAVE RE |
|---------------|----------------|----------|----------|----------|-----|-------------------|------------------|
| JOB IN | IFORMATION | RESULTS | CHART | JS | SON | EXECUTION DETAILS | EXECUTION GRAPH |
| Row | customer_state | ▼ | AVERAGE_ | - | | | |
| 1 | RR | | | 42.98 | | | |
| 2 | PB | | | 42.72 | | | |
| 3 | RO | | | 41.07 | | | |
| 4 | AC | | | 40.07 | | | |
| 5 | PI | | | 39.15 | | | |

BOTTOM 5 STATES WITH LOWEST freight_value AVERAGE

QUERY:

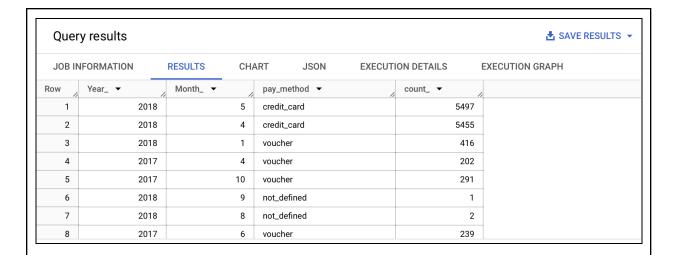
```
SELECT
    c.customer_state,
    ROUND(AVG(oi.freight_value),2) AS AVERAGE_
FROM (customer.orders AS o
    INNER JOIN
        customer.customers AS c
    ON
        o.customer_id = c.customer_id
    INNER JOIN
        customer.order_items AS oi
    ON
        oi.order_id = o.order_id)
GROUP BY
1
ORDER BY
2 asc
LIMIT
5
```



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

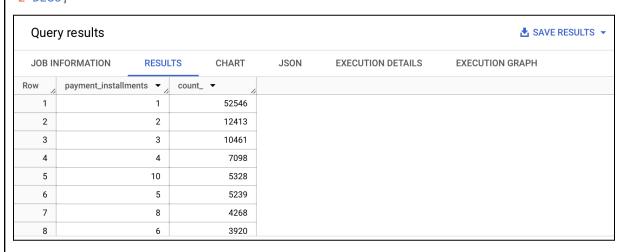
Hint: We want you to count the no. of orders placed using different payment methods in each month over the past years

```
SELECT
EXTRACT (YEAR
FROM
  o.order_purchase_timestamp) AS Year_,
EXTRACT (MONTH
FROM
  o.order_purchase_timestamp) AS Month_,
p.payment_type AS pay_method,
COUNT(p.order_id) AS count_
FROM
customer.payments AS p
INNER JOIN
customer.orders AS o
o.order_id = p.order_id
GROUP BY
1,
2,
3
ORDER BY
4 DESC;
```



12. Find the no. of orders placed on the basis of the payment installments that have been paid. Hint: We want you to count the no. of orders placed based on the no. of payment installments where at least one installment has been successfully paid.

```
payment_installments,
  COUNT(p.order_id) AS count_
FROM
  customer.payments AS p
WHERE
  p.payment_installments != 0
GROUP BY
1
ORDER BY
2 DESC;
```



Insights:

- Orders from the dataset were purchased from (2016-09-04) to (2018-10-17).
- Orders volume increased year by year i.e count of orders from 2016 increased from 267 to 52783 in 2018.
- Most of the orders placed in the afternoon, suggesting promotions for Target occur most in this time.
- Orders purchased across 4085 cities and 27 cities of Brazil. There's a
 variation in average order value across states. Target may explore optimizing
 pricing strategies based on location and customer segments.
- Understanding popular payment methods (credit card, debit card, installments)
 can help tailor the checkout process. A significant portion of customers use
 installments, indicating a preference for spreading purchase costs. Target
 might explore offering more attractive installment plans or partnering with
 financing companies.

Recommendations:

- Implement data-driven promotions based on seasonality, customer location, and order history.
- Implement data-driven promotions based on seasonality, customer location, and order history.
- Implement data-driven promotions based on seasonality, customer location, and order history.
- Optimize inventory levels in warehouses across Brazil to ensure faster delivery times.