Business Case: Target SQL

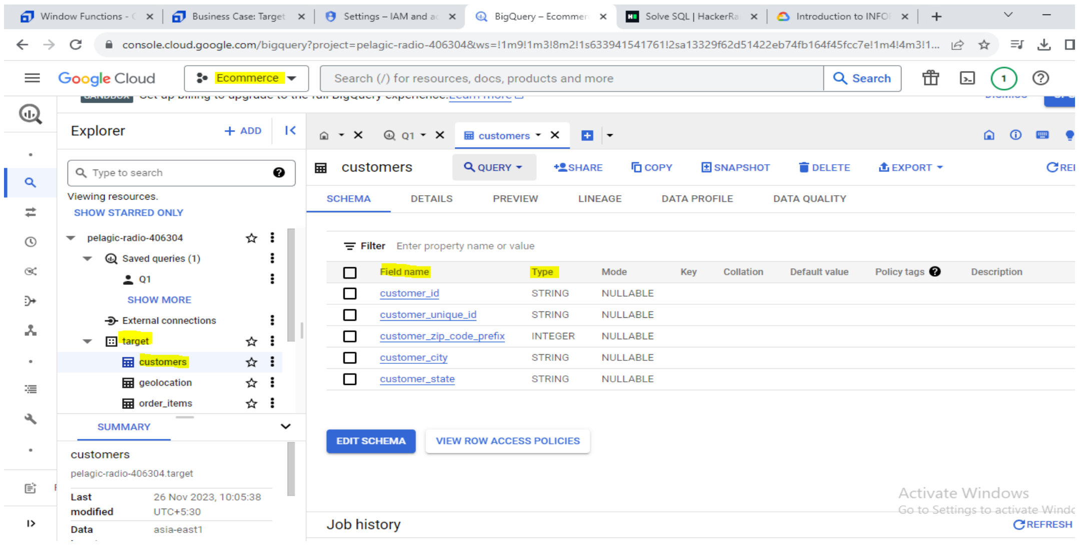
1. Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset:

1. Data type of all columns in the "customers" table.

Approach: In BigQuery we can get in 2 ways:

a. After uploading tables in the dataset, when we click on table, under schema, we were able to see the data type of all columns.

Screen shot:



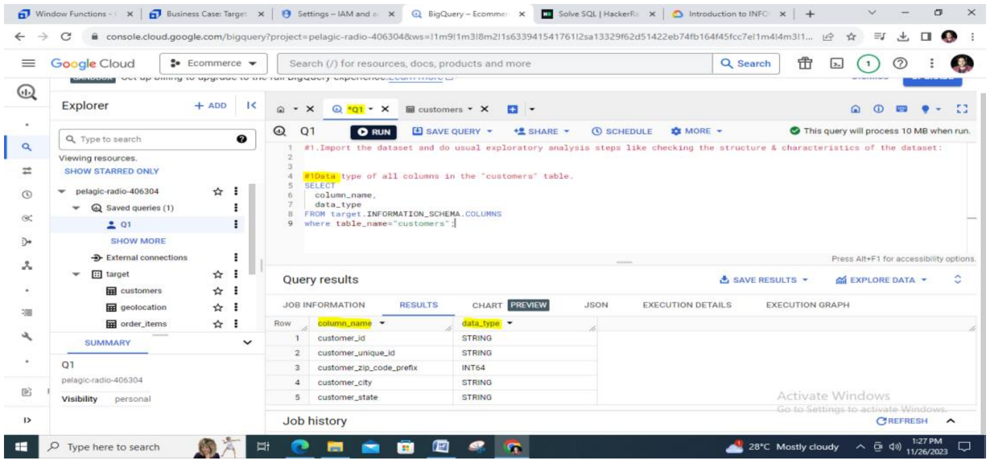
Field name	Type	Mode	Key	Collation	Default value	Policy tags	Description
customer_id	STRING	NULLABLE					
customer_unique_id	STRING	NULLABLE					
customer_zip_code_prefix	INTEGER	NULLABLE					
customer_city	STRING	NULLABLE					
customer_state	STRING	NULLABLE					

b. By writing query using information_schema

Query:

```
SELECT

        column_name,
        data_type
FROM target.INFORMATION_SCHEMA.COLUMNS
where table_name="customers";
```

Screen shot:



```
#1.Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset:


#1Data 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

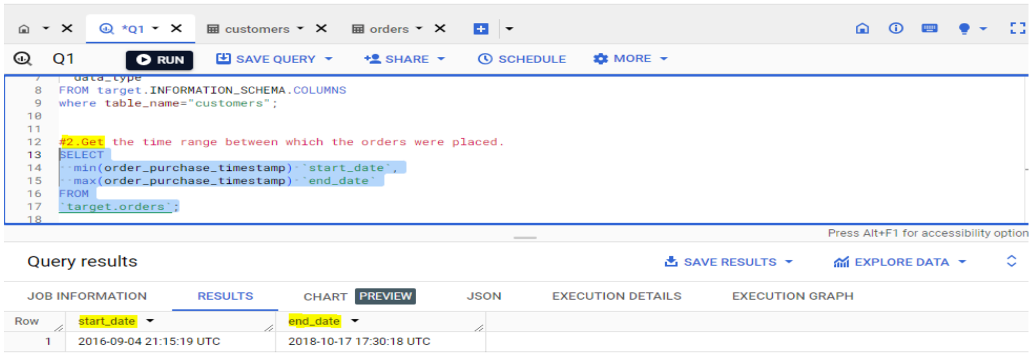
Insights: In customer table of Target we have 5 columns in which **customer_zip_code_prefix** is of integer data type remaining all are of string data type.
Recommendation: Not required

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

Query:

```
SELECT

        min(order_purchase_timestamp) `start_date`,
        max(order_purchase_timestamp) `end_date`
FROM
`target.orders`;
```

Screenshot:



```
  data_type
FROM target.INFORMATION_SCHEMA.COLUMNS
where table_name="customers";


#2.Get the time range between which the orders were placed.
SELECT
  min(order_purchase_timestamp) `start_date`,
  max(order_purchase_timestamp) `end_date`
FROM
`target.orders`;
```

Row	start_date	end_date
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC

Activate Windows

Insights: The time range were orders placed between **'2016-09-04' and '2018-10-17'.**

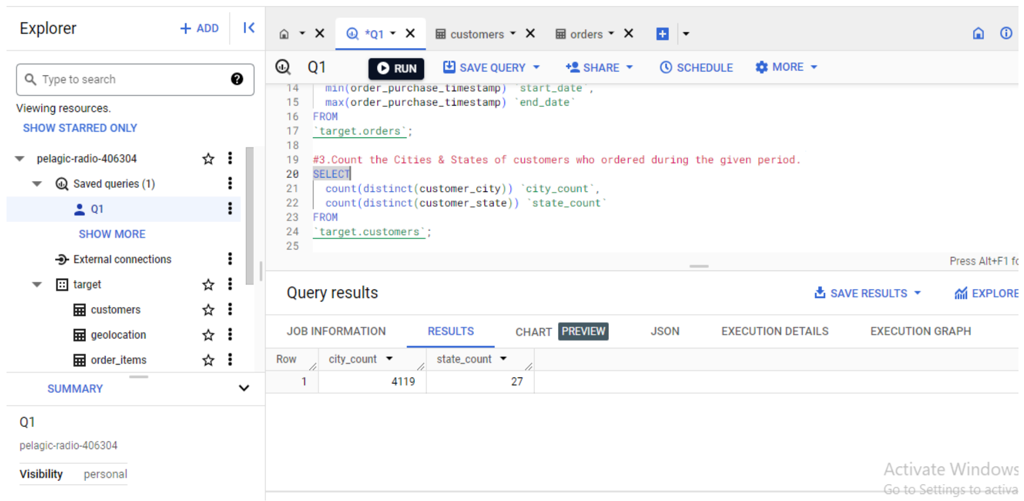
Recommendation: Not required

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

Query:

```
SELECT
        count(distinct(customer_city)) `city_count`,
        count(distinct(customer_state)) `state_count`
FROM `target.customers`;
```

Screen Shot:



```
  min(order_purchase_timestamp) `start_date`,
  max(order_purchase_timestamp) `end_date`
FROM
`target.orders`;

#3.Count the Cities & States of customers who ordered during the given period.
SELECT
  count(distinct(customer_city)) `city_count`,
  count(distinct(customer_state)) `state_count`
FROM
`target.customers`;
```

Query results

Row	city_count	state_count
1	4119	27

Insight: During the given period, the customers orderd from 4119 different cities and 27 states.

Recommendation: We can get highest number of customers of state and city where we can increase productivity, customer services better.
We can find least number of customers of state and city we can increase order from them by promoting ads and coupon, discounts.

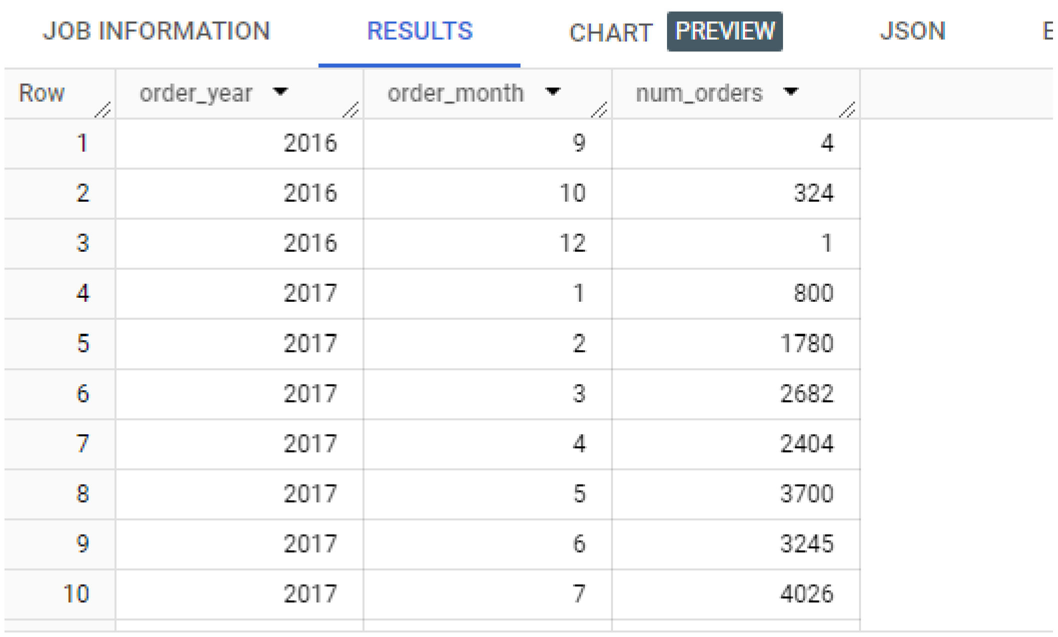
2. In-depth Exploration:

1. 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,
    EXTRACT(MONTH FROM order_purchase_timestamp) AS order_month,
    COUNT(*) AS num_orders
FROM
    `target.orders`
GROUP BY
    order_year, order_month
ORDER BY
    order_year, order_month;
```

Output:


JOB INFORMATION RESULTS CHART PREVIEW JSON E

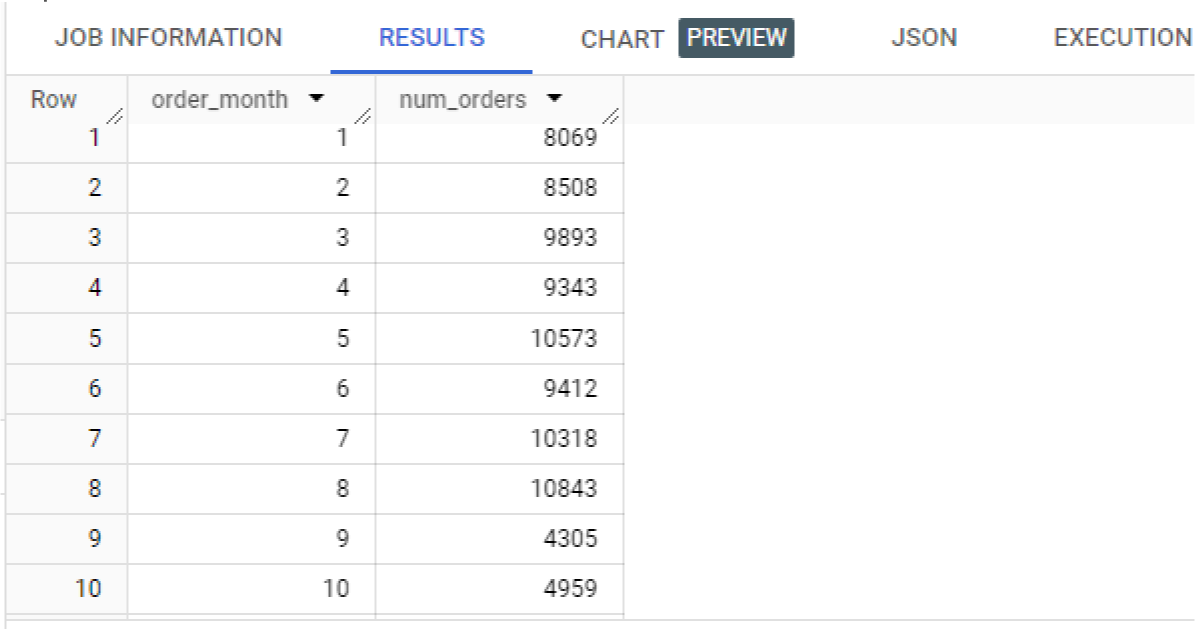
Row	order_year	order_month	num_orders
1	2016	9	4
2	2016	10	324
3	2016	12	1
4	2017	1	800
5	2017	2	1780
6	2017	3	2682
7	2017	4	2404
8	2017	5	3700
9	2017	6	3245
10	2017	7	4026

Insights: In 2016, the number of orders placed in months are highly fluctuating and also we missed the data of 11 th month, From 2017 from month to month there slight increase and decrease in no. of orders placed.
From year to year there is an increase trend in place of orders.
Recommendation:
To know more precisely we can visualize the above data and get where there is high change in trend w.r.t decrease orders so that we can find which month it is and it's reason and we can improve by finding root cause.

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

```
SELECT
    EXTRACT(MONTH FROM order_purchase_timestamp) AS order_month,
    COUNT(*) AS num_orders
FROM
    `target.orders`
GROUP BY
    order_month
ORDER BY
    order_month;
```

output:



Row	order_month	num_orders
1	1	8069
2	2	8508
3	3	9893
4	4	9343
5	5	10573
6	6	9412
7	7	10318
8	8	10843
9	9	4305
10	10	4959

Insights: From the data we have, we can say 5th , 7th and 8th month of all years having high number of orders placed and 9th , 10th and 11th months are having low number of orders.
Recommendation: We can say that for the highest months there is festival season so the orders are high we can promote items with respect to the seasons in the areas and also provide coupons, discounts on other targeted items and also need to provide efficient delivery.
For lowest months we can provide some extra offers on items.

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

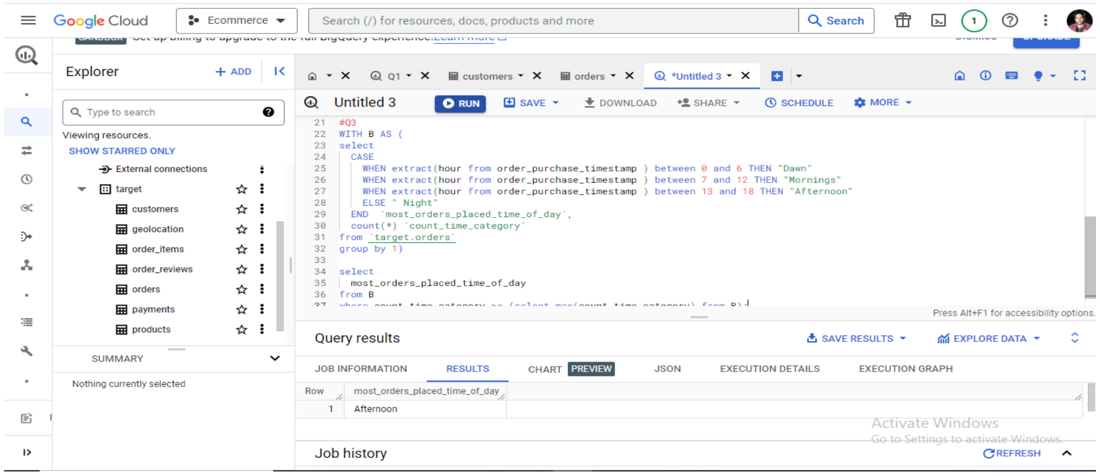
- 0-6 hrs : Dawn
- 7-12 hrs : Mornings
- 13-18 hrs : Afternoon
- 19-23 hrs : Night

Query:

```
WITH B AS (
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
                "Mornings"
                WHEN extract(hour from order_purchase_timestamp ) between 13 and 18 THEN
                "Afternoon"
                ELSE " Night"
        END  `most_orders_placed_time_of_day`,
        count(*) `count_time_category`
from `target.orders`
group by 1)

select
        most_orders_placed_time_of_day
from B
where count_time_category >= (select max(count_time_category) from B);
```

Screenshot:



```
#Q3
WITH B AS (
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 "Mornings"
    WHEN extract(hour from order_purchase_timestamp ) between 13 and 18 THEN "Afternoon"
    ELSE " Night"
  END  `most_orders_placed_time_of_day`,
  count(*) `count_time_category`
from `target.orders`
group by 1)

select
  most_orders_placed_time_of_day
from B
where count_time_category >= (select max(count_time_category) from B);
```

Query results

Row	most_orders_placed_time_of_day
1	Afternoon

Insight: During Afternoon(13-18 hrs) of the day, the Brazilian customers mostly place their orders
Recommendation: We can provide best ways to handle the traffic of orders placed in After noon and also increase the best support teams to handle if customers placing orders are having any issues**.**

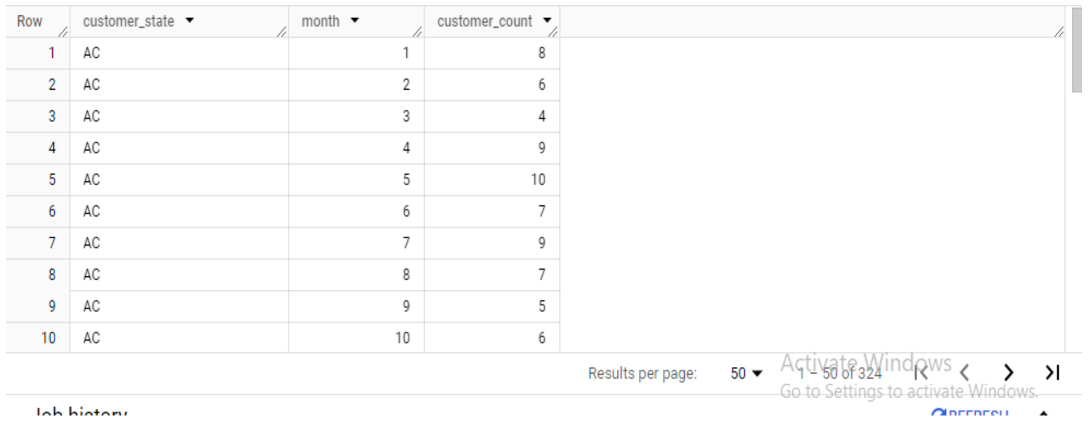
3.Evolution of E-commerce orders in the Brazil region:

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

Query:

```
#CTE
WITH
months AS (
select distinct(extract(month from order_purchase_timestamp)) `month` from `target.orders`),

filter_table AS(
select
  c.customer_id,
  c.customer_state,
  extract(month from o.order_purchase_timestamp) `month`
from `target.customers` c
left join `target.orders` o
on c.customer_id = o.customer_id)

select
  cs.`customer_state`,
  m.month,
  coalesce(count(distinct ft.customer_id),0) as customer_count
from (
  (select distinct(customer_state) `customer_state` from `target.customers`) cs
  cross join
  months `m`
  left join filter_table `ft` on ft.`customer_state` = cs.`customer_state` and ft.`month` =
m.month)
group by
cs.`customer_state`,`m`.month
order by
cs.`customer_state`,`m`.month;
```

ScreenShot:



Row	customer_state	month	customer_count
1	AC	1	8
2	AC	2	6
3	AC	3	4
4	AC	4	9
5	AC	5	10
6	AC	6	7
7	AC	7	9
8	AC	8	7
9	AC	9	5
10	AC	10	6

Results per page: 50 1 – 50 of 324

Activate Windows
Go to Settings to activate Windows.

Insights: We are getting all months order placed for each state, so we can use this data to conclude, on which month the orders were not placed or least number of orders placed and also we can get which months for which states we are getting highest number of orders.

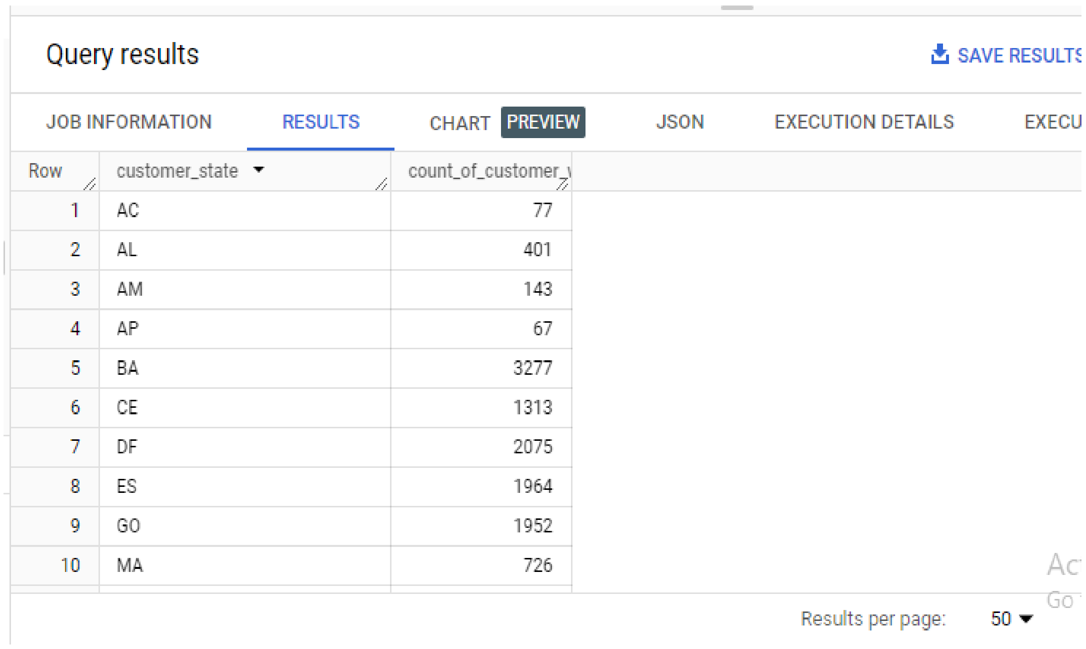
Recommendation: By using this data we can provide delivery efficiency in the state and month with highest orders and also from application p.o.v we need to be more efficient by maintaining best user experience for the traffic of orders.
For lower and zero orders in the state and particular month we need to check the data is not recorded or really the orders were not placed and need to find the root cause analysis by providing details to market or sales executives to get the correct information then we can look into the issue and make decisions accordingly.

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

Query:

```
select

        customer_state,
        count(distinct(customer_unique_id)) `count_of_customer_wrt_state`
from `target.customers`
group by customer_state
order by customer_state;
```

ScreenShot:



Row	customer_state	count_of_customer_\
1	AC	77
2	AL	401
3	AM	143
4	AP	67
5	BA	3277
6	CE	1313
7	DF	2075
8	ES	1964
9	GO	1952
10	MA	726

Results per page: 50

Insights:State SP has highest number of customers and RR has the lowest number of customers

Recommendation: So from the data we can say that for the highest number of customers of that state we can provide best customer experiences by providing some reward points and offers.

In addition we can add items which are least selling by providing certain offers and increase the selling of least selling items.

For the lowest customers of states we can increase offers and discounts and target the customer with items which are frequently buy. Need to make promotions by mentioning maximum offers.

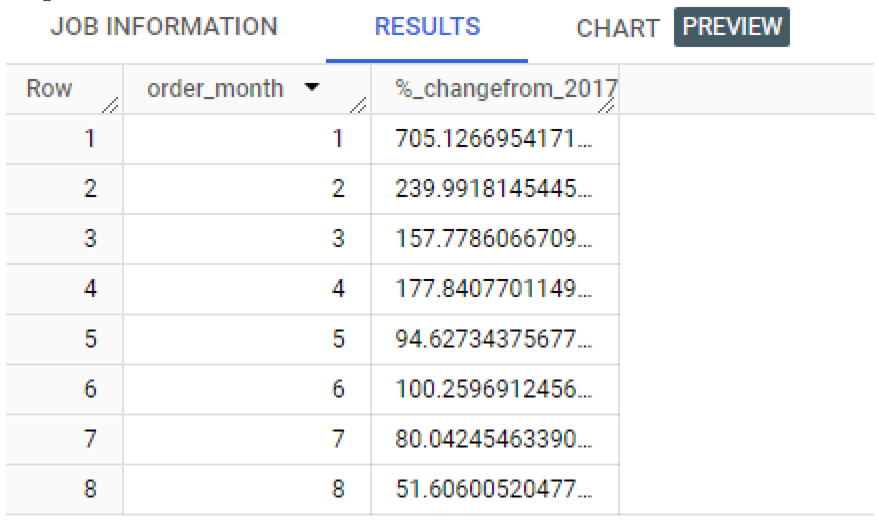
4. Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight and others.

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
  `data_2017` AS (
    select
      extract(month from `orders`.order_purchase_timestamp) as `order_month`,
      sum(`payments`.payment_value) as `cost_of_orders_2017`
    from `target.orders` `orders`
    left join `target.payments` `payments`
    on `orders`.order_id = `payments`.order_id
    where extract(month from `orders`.order_purchase_timestamp) IN (1,2,3,4,5,6,7,8)
    and
    extract(year from `orders`.order_purchase_timestamp)=2017
    group by `order_month`
    order by `order_month`
  ),

  `data_2018` AS (
    select
      extract(month from `orders`.order_purchase_timestamp) as `order_month`,
      sum(`payments`.payment_value) as `cost_of_orders_2018`
    from `target.orders` `orders`
    left join `target.payments` `payments`
    on `orders`.order_id = `payments`.order_id
    where extract(month from `orders`.order_purchase_timestamp) IN (1,2,3,4,5,6,7,8)
    and
    extract(year from `orders`.order_purchase_timestamp)=2018
    group by `order_month`
    order by `order_month`
  ),

  `data_2017_2018` AS (
    select
     `data_2017`.order_month,
      `cost_of_orders_2017`,
      `cost_of_orders_2018`
    from `data_2017`
    inner join `data_2018`
    on `data_2017`.order_month =`data_2018`.order_month
  )



select
    order_month,
    ((`cost_of_orders_2018`-`cost_of_orders_2017`)/`cost_of_orders_2017`)*100 as
`%_changefrom_2017_to_2018`
from `data_2017_2018`
order by order_month asc;
```

output:



Row	order_month	%_changefrom_2017
1	1	705.1266954171...
2	2	239.9918145445...
3	3	157.7786066709...
4	4	177.8407701149...
5	5	94.62734375677...
6	6	100.2596912456...
7	7	80.04245463390...
8	8	51.60600520477...

Insight: From January to August, from 2017 to 2018, January has high change in terms of percentage and August has least change.

Recommendation: Identify Trends, Provide Promotions and Discounts.

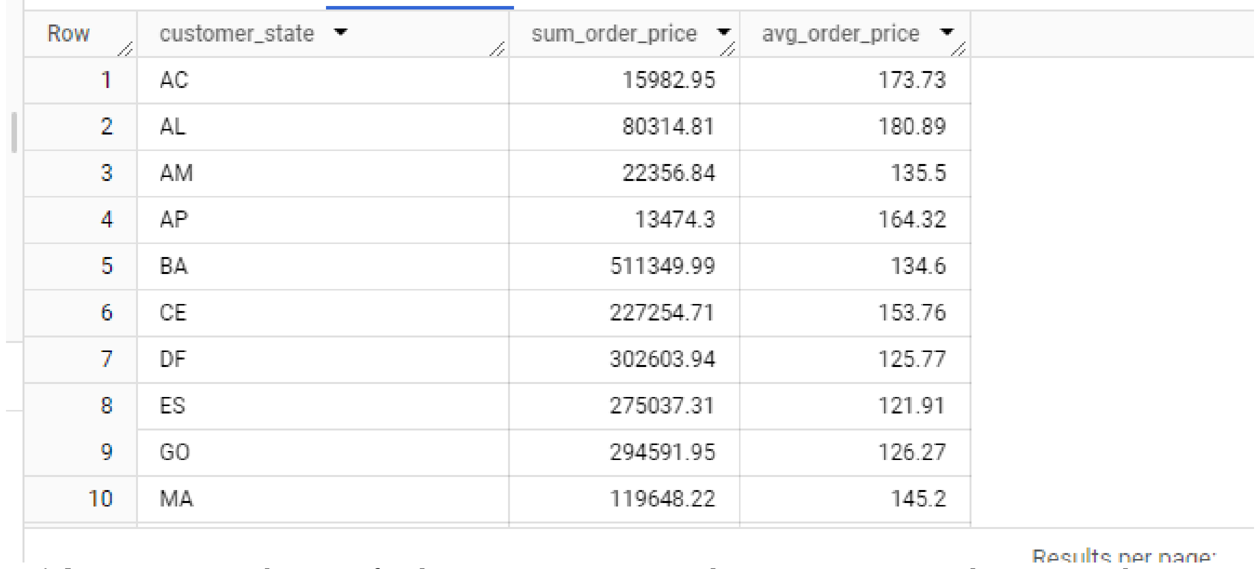
Evaluate the factors contributing to increased order costs. Assess whether it's due to higher product prices, increased shipping costs, or other factors.

Consider adjusting product prices or shipping fees based on the analysis. Ensure that pricing strategies align with customer expectations and market conditions.

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

```
select `tc`.customer_state,
  round(SUM(`oi`.price),2) as `sum_order_price`,
  round(AVG(`oi`.price),2) as `avg_order_price`
from `target.customers` `tc`
left join `target.orders` `to`
on `tc`.customer_id = `to`.customer_id
left join `target.order_items` `oi`
on `oi`.order_id = `to`.order_id
group by `tc`.customer_state
order by `tc`.customer_state;
```

output:



Row	customer_state	sum_order_price	avg_order_price
1	AC	15982.95	173.73
2	AL	80314.81	180.89
3	AM	22356.84	135.5
4	AP	13474.3	164.32
5	BA	511349.99	134.6
6	CE	227254.71	153.76
7	DF	302603.94	125.77
8	ES	275037.31	121.91
9	GO	294591.95	126.27
10	MA	119648.22	145.2

Results per page:

Insight: For state SP the sum of order price is maximum, value is 5202955.05 and average_order price is max for state PE.

For state AP the sum of orders price is minimum, value is 13474.3 and average_order_price is minimum for state SP

Recommendation: Analyze the states with the highest total order prices to identify states with strong sales performance.

Segment customers based on states to understand preferences and behaviors. Adjust marketing and product strategies accordingly.

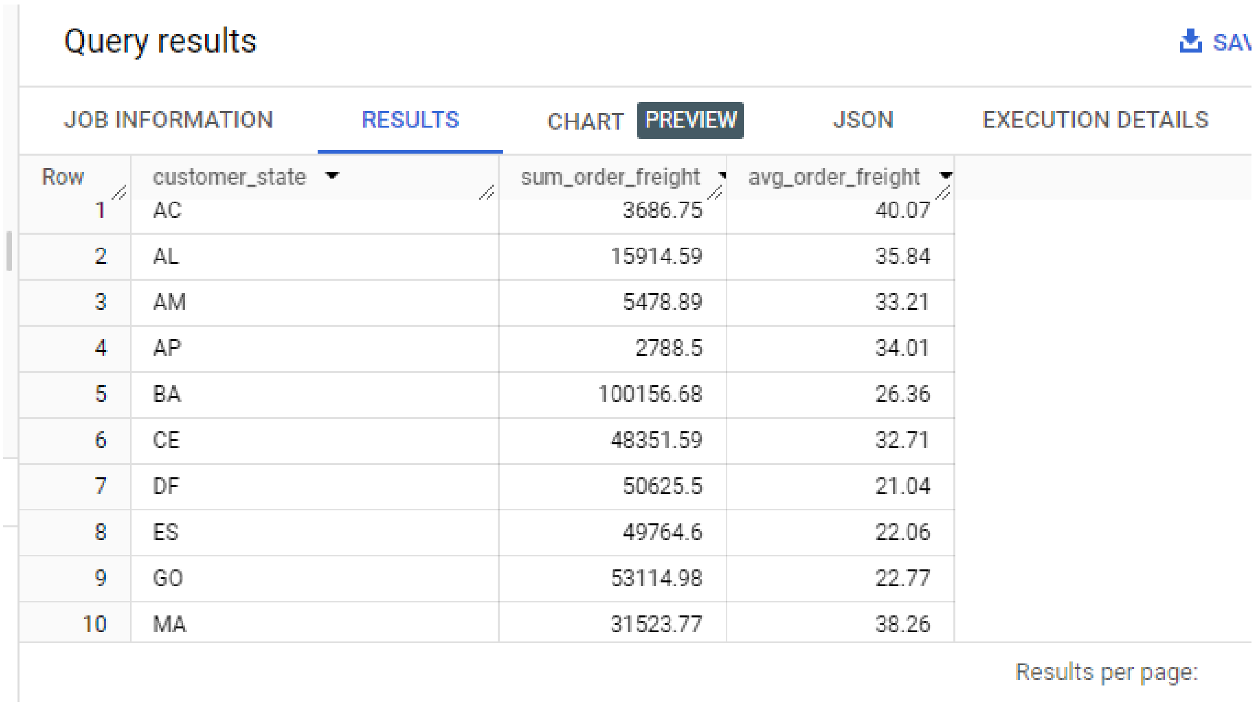
Analyze the types of products that contribute significantly to total order prices. Consider expanding or promoting these product categories.

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

Query:

```
select `tc`.customer_state,
  round(SUM(`oi`.freight_value),2) as `sum_order_freight`,
  round(AVG(`oi`.freight_value),2) as `avg_order_freight`
from `target.customers` `tc`
left join `target.orders` `to`
on `tc`.customer_id = `to`.customer_id
left join `target.order_items` `oi`
on `oi`.order_id = `to`.order_id
group by `tc`.customer_state
order by `tc`.customer_state;
```

output:



Row	customer_state	sum_order_freight	avg_order_freight
1	AC	3686.75	40.07
2	AL	15914.59	35.84
3	AM	5478.89	33.21
4	AP	2788.5	34.01
5	BA	100156.68	26.36
6	CE	48351.59	32.71
7	DF	50625.5	21.04
8	ES	49764.6	22.06
9	GO	53114.98	22.77
10	MA	31523.77	38.26

Insight:

Total value of order freight for each state, indicating the overall shipping cost volume.

Average value of order freight for each state, providing insights into customer shipping cost patterns.

Recommendation: Analyze the states with the highest total order freight to identify regions with higher shipping costs and vice versa.

Explore opportunities to optimize shipping costs, especially in states with high shipping expenses. Negotiate with logistics partners for better rates.

Clearly communicate shipping costs to customers, especially in states with higher average order freight. Manage customer expectations regarding shipping expenses.

Consider adjusting product prices or introducing shipping promotions in states with higher shipping costs to remain competitive.

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

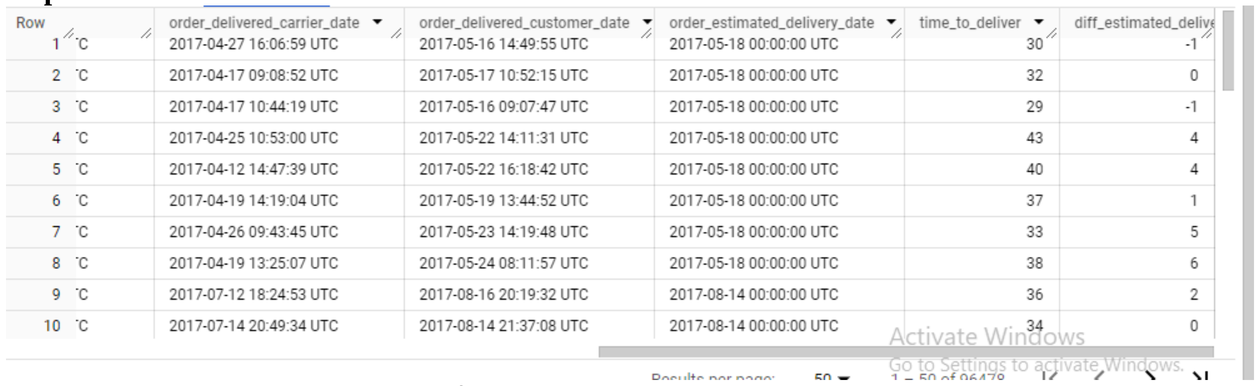
1. Find the no. of days taken to deliver each order from the order's purchase date as delivery time. Also, calculate the difference (in days) between the estimated & actual delivery date of an order. Do this in a single query.

   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_timestamp
   - **diff_estimated_delivery** = order_estimated_delivery_date - order_delivered_customer_date

Query:

```
select
  *,
  DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp,day) as `time_to_deliver`,
  DATE_DIFF(order_delivered_customer_date,order_estimated_delivery_date,day) as
`diff_estimated_delivery` #negative number indicates no. of days before estimated delivered,
postive indicates delivered after estimated date
from `target.orders`
where order_status="delivered";
```

output:



Row		order_delivered_carrier_date	order_delivered_customer_date	order_estimated_delivery_date	time_to_deliver	diff_estimated_delive
1	C	2017-04-27 16:06:59 UTC	2017-05-16 14:49:55 UTC	2017-05-18 00:00:00 UTC	30	-1
2	C	2017-04-17 09:08:52 UTC	2017-05-17 10:52:15 UTC	2017-05-18 00:00:00 UTC	32	0
3	C	2017-04-17 10:44:19 UTC	2017-05-16 09:07:47 UTC	2017-05-18 00:00:00 UTC	29	-1
4	C	2017-04-25 10:53:00 UTC	2017-05-22 14:11:31 UTC	2017-05-18 00:00:00 UTC	43	4
5	C	2017-04-12 14:47:39 UTC	2017-05-22 16:18:42 UTC	2017-05-18 00:00:00 UTC	40	4
6	C	2017-04-19 14:19:04 UTC	2017-05-19 13:44:52 UTC	2017-05-18 00:00:00 UTC	37	1
7	C	2017-04-26 09:43:45 UTC	2017-05-23 14:19:48 UTC	2017-05-18 00:00:00 UTC	33	5
8	C	2017-04-19 13:25:07 UTC	2017-05-24 08:11:57 UTC	2017-05-18 00:00:00 UTC	38	6
9	C	2017-07-12 18:24:53 UTC	2017-08-16 20:19:32 UTC	2017-08-14 00:00:00 UTC	36	2
10	C	2017-07-14 20:49:34 UTC	2017-08-14 21:37:08 UTC	2017-08-14 00:00:00 UTC	34	0

Activate Windows
Go to Settings to activate Windows.

Insight: We can analyze the distribution of delivery times to understand the average and variation in the time taken to deliver orders.

Identify orders that were delivered within the estimated delivery time. Determine the percentage of orders that are delivered on time.

Analyze whether the geo location of customers has an impact on delivery times. Identify regions with consistently longer or shorter delivery times and Seasonal Patterns and Delivery Delays.

Recommendation:

If there are consistent delays, evaluate and optimize the logistics and order fulfillment processes to improve operational efficiency.

Implement effective communication strategies to inform customers about estimated delivery times and any potential delays. Transparency can enhance customer satisfaction.

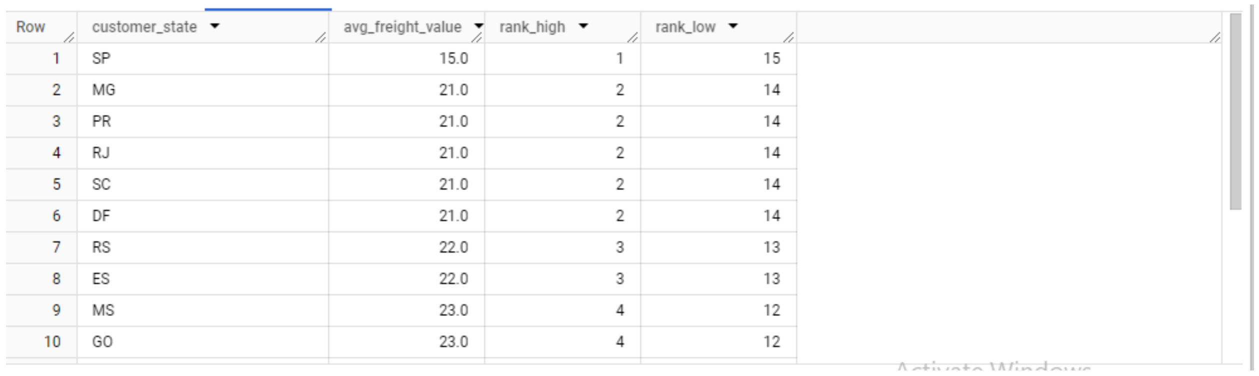
Implement predictive analytics to forecast delivery times more accurately. Use historical data to identify patterns and improve delivery estimates.

Collaboration with Logistics Partners and focus on Customer Feedback Analysis.

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

```
CREATE VIEW target.`ranks_cities` AS
select
  `tc`.customer_state,
  round(AVG(freight_value)) as `avg_freight_value`,
  dense_rank() over(order by round(AVG(freight_value))) as `rank_high`,
  dense_rank() over(order by round(AVG(freight_value)) desc) as `rank_low`
from `target.customers` `tc`
left join `target.orders` `to`
on `tc`.customer_id = `to`.customer_id
left join `target.order_items` `oi`
on `oi`.order_id = `to`.order_id
group by `tc`.customer_state
order by `avg_freight_value`;
```

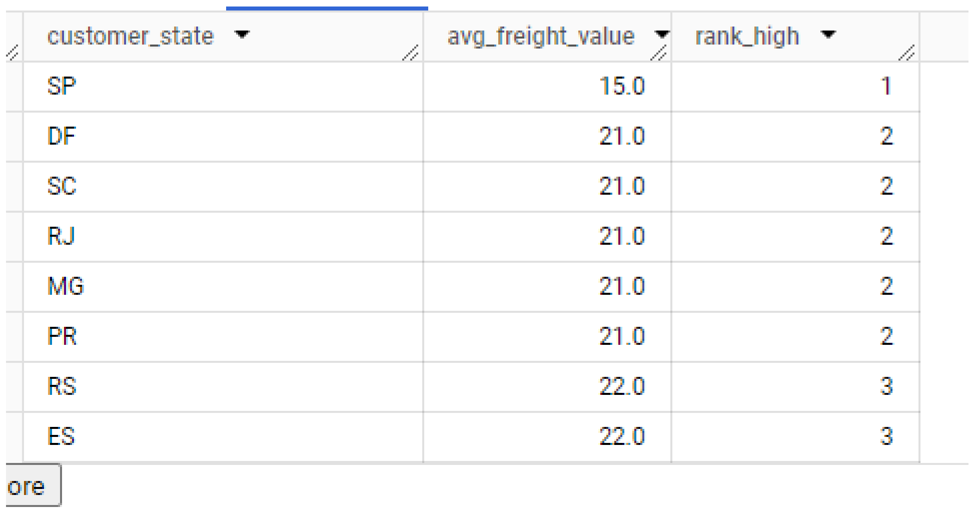
screenshot:



Row	customer_state	avg_freight_value	rank_high	rank_low
1	SP	15.0	1	15
2	MG	21.0	2	14
3	PR	21.0	2	14
4	RJ	21.0	2	14
5	SC	21.0	2	14
6	DF	21.0	2	14
7	RS	22.0	3	13
8	ES	22.0	3	13
9	MS	23.0	4	12
10	GO	23.0	4	12

```
#Top 5 state with the lowest average freight value
select
  `ranks_cities`.customer_state,
  avg_freight_value,
  rank_high
from target.`ranks_cities`
where `rank_high`<=5
order by avg_freight_value;
```

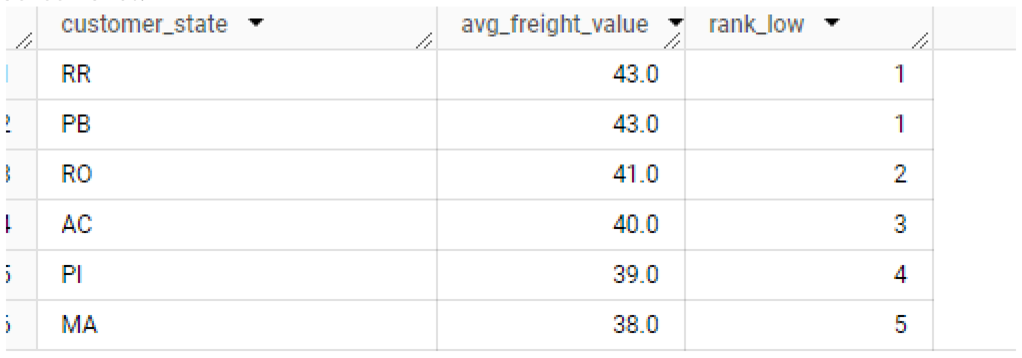
screenshot:



customer_state	avg_freight_value	rank_high
SP	15.0	1
DF	21.0	2
SC	21.0	2
RJ	21.0	2
MG	21.0	2
PR	21.0	2
RS	22.0	3
ES	22.0	3

```
#Top 5states with the highest average freight value
select
  ranks_cities.customer_state,
  avg_freight_value,
  rank_low
from target.`ranks_cities`
where `rank_low`<=5
order by rank_low ;
```

screen shot:



customer_state	avg_freight_value	rank_low
RR	43.0	1
PB	43.0	1
RO	41.0	2
AC	40.0	3
PI	39.0	4
MA	38.0	5

Insight:

Top 5 States with Highest Average Freight:

This information can help understand states where shipping costs are generally higher.

Top 5 States with Lowest Average Freight:

This can be indicative of regions with lower shipping costs or efficient logistics.

Recommendations:

For states with the highest average freight, explore opportunities to optimize shipping routes, carrier negotiations, or packaging strategies to reduce costs.

Consider adjusting product prices or introducing promotions for states with higher average freight values to offset shipping costs and remain competitive.

Evaluate and negotiate partnerships with logistics providers to secure more favorable rates, especially in regions with higher freight costs.

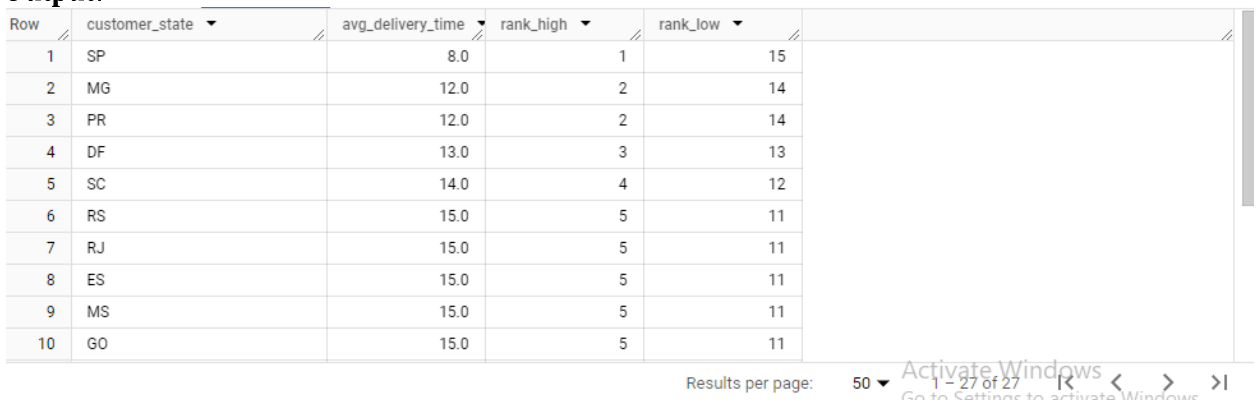
Explore the possibility of establishing regional warehouses or fulfillment centers in states with higher average freight costs to minimize shipping distances.

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

Query:

```
CREATE VIEW target.`avg_delivery_states_table` AS
SELECT
    customer_state,
    ROUND(AVG(DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp,day))) AS
avg_delivery_time,
    DENSE_RANK() OVER (ORDER BY ROUND(AVG(DATE_DIFF(order_delivered_customer_date,
order_purchase_timestamp,day)))) AS rank_high,
    DENSE_RANK() OVER (ORDER BY ROUND(AVG(DATE_DIFF(order_delivered_customer_date,
order_purchase_timestamp,day)))DESC) AS rank_low
FROM
    `target.orders` AS `to`
LEFT JOIN `target.customers` AS `tc`
ON `to`.customer_id = `tc`.customer_id
WHERE
    order_status ="delivered"
GROUP BY
    customer_state
ORDER BY
    avg_delivery_time ;
```

Output:



Row	customer_state	avg_delivery_time	rank_high	rank_low
1	SP	8.0	1	15
2	MG	12.0	2	14
3	PR	12.0	2	14
4	DF	13.0	3	13
5	SC	14.0	4	12
6	RS	15.0	5	11
7	RJ	15.0	5	11
8	ES	15.0	5	11
9	MS	15.0	5	11
10	GO	15.0	5	11

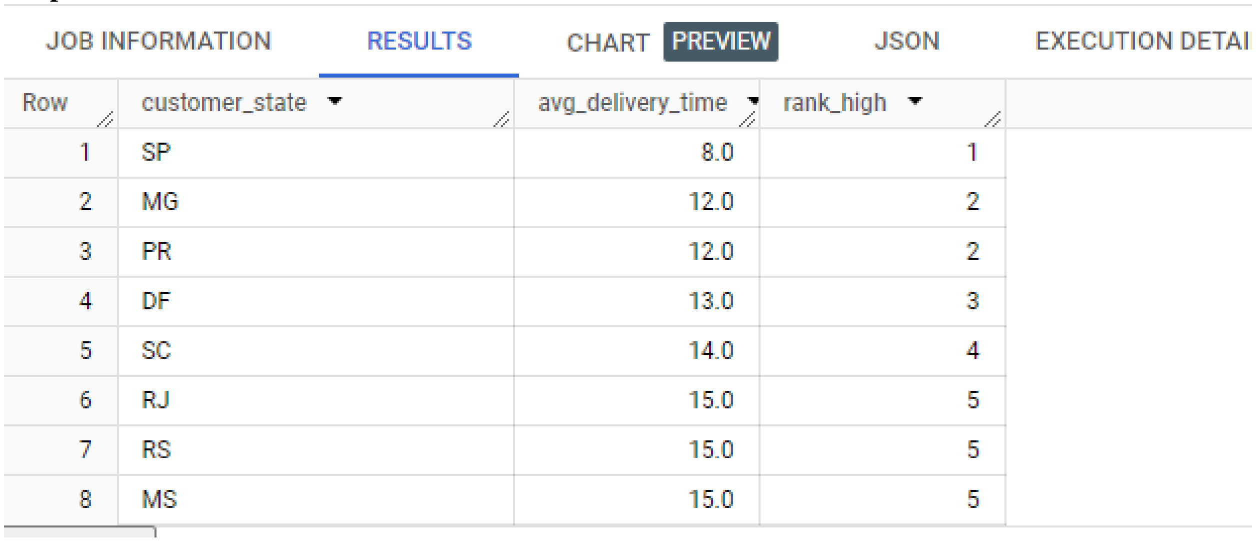
Results per page: 50 1 – 27 of 27

#top 5 states with lowest avg_delivery_time

Query:

```
SELECT
    customer_state,
    avg_delivery_time,
    rank_high
FROM
target.`avg_delivery_states_table`
WHERE `rank_high`<=5;
```

Output:



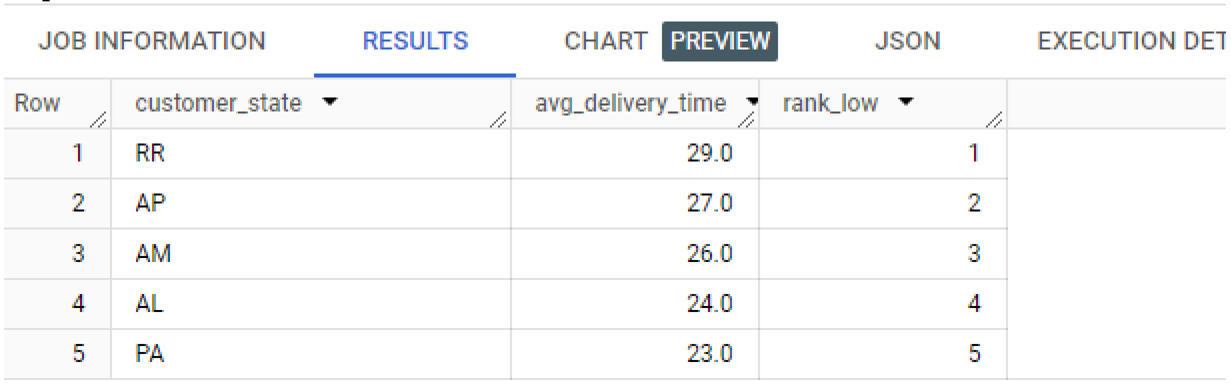
Row	customer_state	avg_delivery_time	rank_high
1	SP	8.0	1
2	MG	12.0	2
3	PR	12.0	2
4	DF	13.0	3
5	SC	14.0	4
6	RJ	15.0	5
7	RS	15.0	5
8	MS	15.0	5

# top 5 states with highest avg_time

Query:

```
SELECT
    customer_state,
    avg_delivery_time,
    rank_low
FROM
target.`avg_delivery_states_table`
WHERE `rank_low`<=5
order by `rank_low`;
```

output:



Row	customer_state	avg_delivery_time	rank_low
1	RR	29.0	1
2	AP	27.0	2
3	AM	26.0	3
4	AL	24.0	4
5	PA	23.0	5

Insights:

Top 5 States with Highest Average Delivery Time:

Identify the states where customers, on average, experience the longest delivery times. This information can highlight states where delivery logistics may need improvement.

Top 5 States with Lowest Average Delivery Time:

Identify the states where customers, on average, experience the shortest delivery times. This can indicate states with efficient delivery logistics.

Recommendations:

For states with the highest average delivery time, analyze and optimize the logistics and transportation processes to reduce delivery times.

Explore the possibility of regional warehousing or fulfillment centers in states with longer average delivery times to shorten shipping distances.

Enhance the overall customer experience by ensuring timely deliveries, particularly in regions with longer average delivery times.

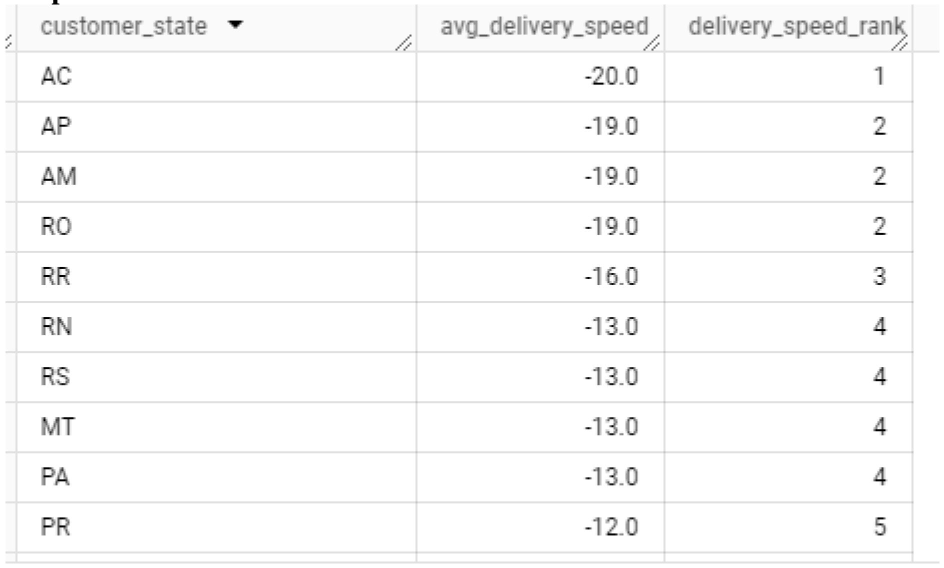
Work on improving the overall efficiency of the supply chain, from order processing to delivery, to reduce lead times

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 DeliverySpeeds AS (
    SELECT
        `tc`.customer_state,
        ROUND(AVG(DATE_DIFF(order_delivered_customer_date,
order_estimated_delivery_date,day))) AS `avg_delivery_speed`,
        DENSE_RANK() OVER (ORDER BY ROUND(AVG(DATE_DIFF(order_delivered_customer_date,
order_estimated_delivery_date,       day)))ASC) as `delivery_speed_rank`
    FROM
    `target.orders` AS `to`
    LEFT JOIN `target.customers` AS `tc`
    ON `to`.customer_id = `tc`.customer_id
    WHERE
        order_status = 'delivered'
    GROUP BY
        customer_state
    ORDER BY
        `avg_delivery_speed`
)

SELECT
    customer_state,
    avg_delivery_speed,
    delivery_speed_rank
FROM
    DeliverySpeeds
WHERE
    `delivery_speed_rank` <= 5;
```

Output:



customer_state	avg_delivery_speed	delivery_speed_rank
AC	-20.0	1
AP	-19.0	2
AM	-19.0	2
RO	-19.0	2
RR	-16.0	3
RN	-13.0	4
RS	-13.0	4
MT	-13.0	4
PA	-13.0	4
PR	-12.0	5

Insights: This information highlights states with efficient and expedited delivery services. The state with highest average delivery speed is ‘AC’.

Recommendations: Analyze the processes and logistics in states with the fastest delivery speeds. Identify factors contributing to the efficiency and replicate successful strategies in other regions.

Share best practices and optimize logistics processes across the entire delivery network to ensure consistent and fast delivery times.

Implement or enhance real-time tracking features to keep customers informed about the status and location of their orders, contributing to a positive delivery experience.

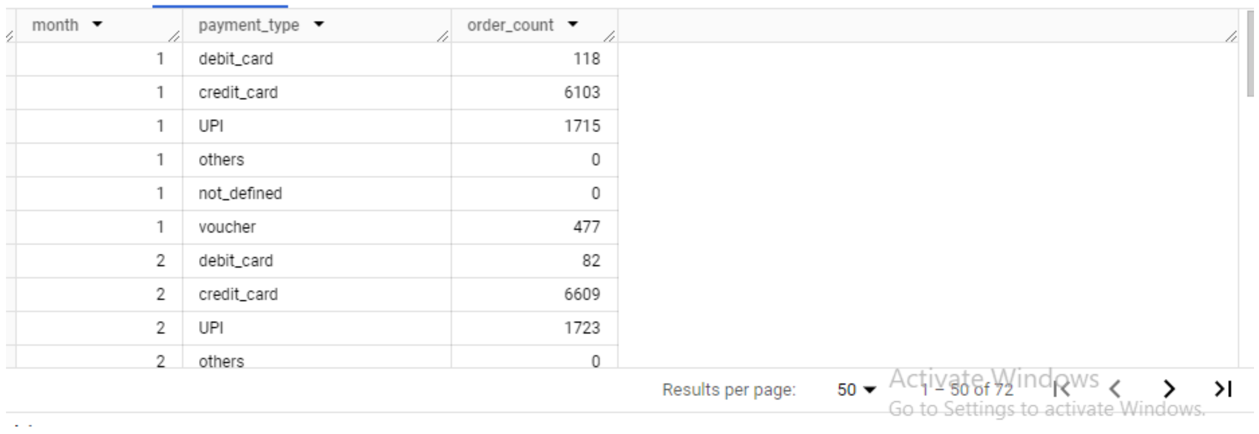
6.Analysis based on the payments:

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

Query:

```
WITH
  `payment_mode` AS (
      select
          IF (payment_type IS NULL,"others",payment_type) AS payment_type
      from `target.orders`
      LEFT JOIN `target.payments`
      ON `target.orders`.order_id = `target.payments`.order_id
      group by payment_type),

  `month` AS(
      select
          extract(month from order_purchase_timestamp) as `month`
      from `target.orders`
      group by `month`),

  `filtered_date_orders_count` AS (
      SELECT
          extract(month from order_purchase_timestamp) AS `month`,
          IF (payment_type IS NULL,"others",payment_type) AS payment_type,
          count(*) AS `total_orders_placed`
      FROM `target.orders`
      LEFT JOIN `target.payments`
      ON `target.orders`.order_id = `target.payments`.order_id
      group by `month`,payment_type)


SELECT
        `month`.month,
        `payment_mode`.payment_type,
        COALESCE(`filtered_date_orders_count`.`total_orders_placed`,0) AS `order_count`
    FROM `month`
    CROSS JOIN `payment_mode`
    LEFT JOIN `filtered_date_orders_count`
    ON`month`.month = `filtered_date_orders_count`.month and `payment_mode`.payment_type=
`filtered_date_orders_count`.   payment_type
ORDER BY month;
```

ScreenShot:



month	payment_type	order_count
1	debit_card	118
1	credit_card	6103
1	UPI	1715
1	others	0
1	not_defined	0
1	voucher	477
2	debit_card	82
2	credit_card	6609
2	UPI	1723
2	others	0

Results per page: 50 1 – 50 of 72

Activate Windows
Go to Settings to activate Windows.

Insight: It is observed that for few order payments are not done using any of the payment methods , so for that I added as others as payment _type for each month. order_count.

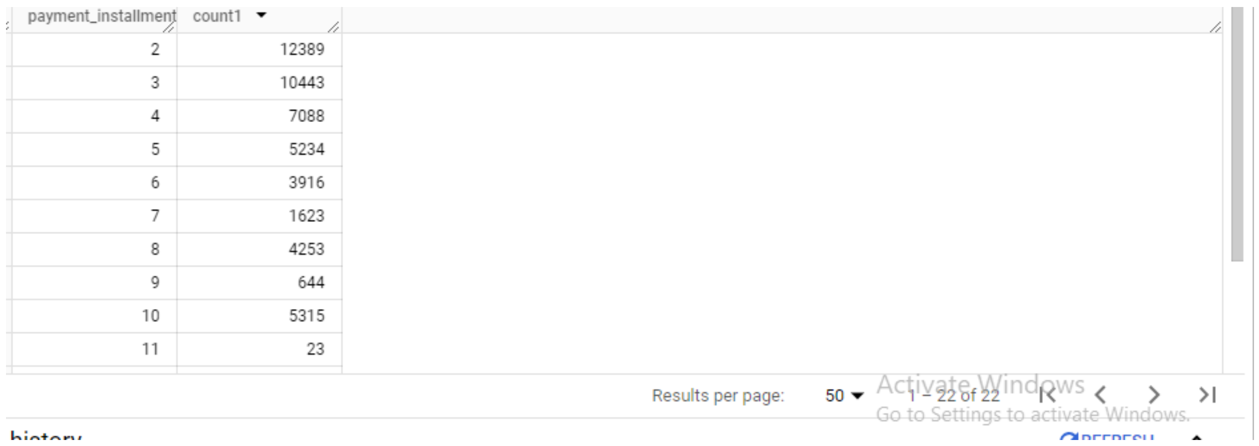
We have almost 6 types of payment types, we are considering: debit_card, credit_card, UPI, not_defined, voucher and others.

So for 12 months- overall we have 72 rows, we can analyze as required.

Recommendation: we can do the following things- Payment Method Analysis, Seasonal Patterns, Customer Preferences, Payment Processing Efficiency, User Experience

```
#2.Find the no. of orders placed on the basis of the payment installments that have been paid.
select
  payment_installments,
  count(distinct(order_id)) AS `count1`
from `target.payments`
where payment_installments>1
group by payment_installments
order by payment_installments;
```

output:



payment_installment	count1
2	12389
3	10443
4	7088
5	5234
6	3916
7	1623
8	4253
9	644
10	5315
11	23

Results per page: 50 1 – 22 of 22

Activate Windows
Go to Settings to activate Windows.

Insight: We can identify which payment installment plans are most commonly chosen by customers,

We can assess whether customers tend to choose a specific number of installments or if there is variability in their choices

Recommendations: If certain payment installment options are more popular, consider offering flexible payment plans to attract a broader customer base.

Increase marketing strategies to highlight the availability and benefits of installment payment options. Consider promoting installment plans during peak shopping periods.

Understand customer behavior regarding installment payments. Determine if there are preferred installment plans or if customers tend to pay in full.