Business Case – Target SQL

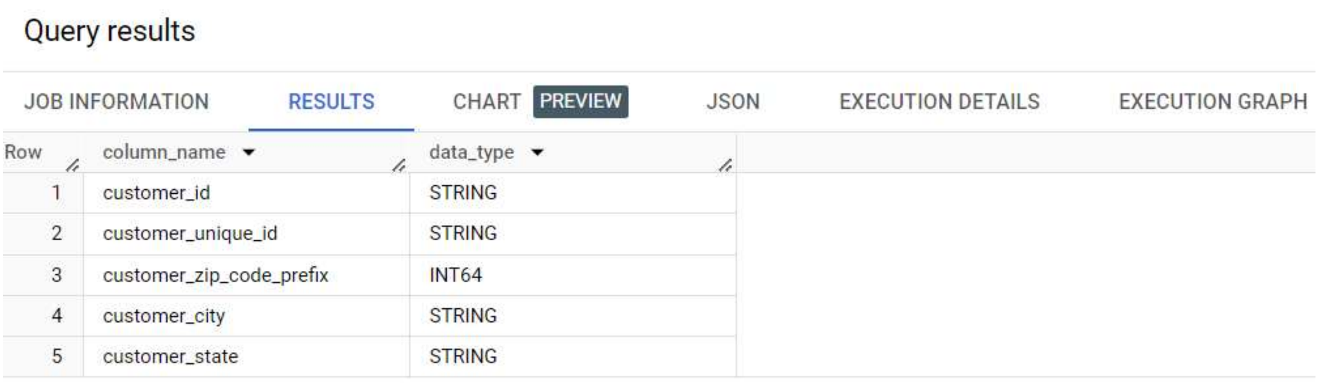
I. Initial exploration like checking the structure & characteristics of the data

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

SQL Code

```
SELECT column_name, data_type
FROM `target`.INFORMATION_SCHEMA.COLUMNS
WHERE table_name='customers';
```

Output



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

Discussion

- The `INFORMATION_SCHEMA.COLUMNS` provides the metadata of columns present in a data set.

Observations

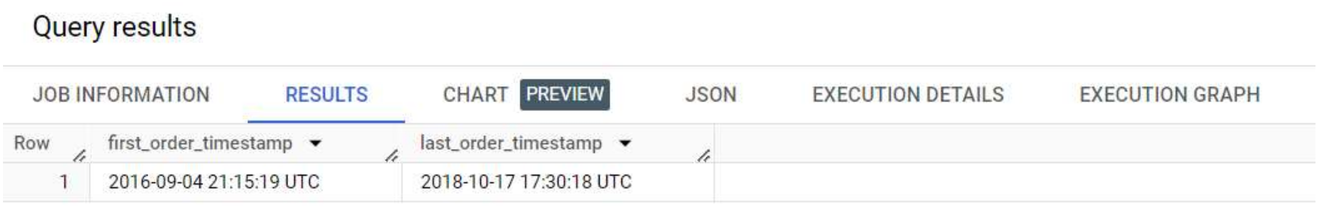
- From the output, we see that there are 5 columns in the `customers` table :
  - `customer_id` (data type : `STRING`)
  - `customer_unique_id` (data type : `STRING`)
  - `customer_zip_code_prefix` (data type : `INT64`)
  - `customer_city` (data type : `STRING`)
  - `customer_state` (data type : `STRING`)

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

SQL Code

```
SELECT MIN(order_purchase_timestamp) AS first_order_timestamp,
MAX(order_purchase_timestamp) AS last_order_timestamp
FROM `target.orders`;
```

Output



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

Discussion

- The date and time information at which each order was placed is present in the `orders` table.

Observations

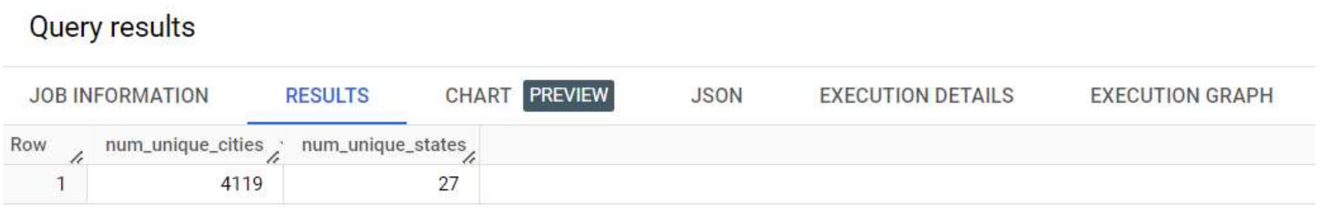
- From the output, we see that the time range between which the orders were placed extends from 4th September, 2016 at 21:15:19 UTC (first order placed) to 17th October, 2018 at 17:30:18 UTC (last order placed).

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

SQL Code

```
SELECT COUNT(DISTINCT C.customer_city) AS num_unique_cities, COUNT(DISTINCT C.customer_state) AS
num_unique_states
FROM `target.orders` AS O
INNER JOIN `target.customers` AS C
ON O.customer_id=C.customer_id;
```

Output



Row	num_unique_cities	num_unique_states
1	4119	27

Discussion

- The orders table consists of important information related to each order placed. The customers table consists of the details of all customers present within the company's database, including the city and state in which the customer is based in.
- From the ER diagram, we see that the customer_id column in the orders table is the foreign key which can be used to join the orders table with the customers table. We can perform an inner join because we expect every order placed to have an associated customer_id value that is NOT NULL (this has been verified). We are not interested in the information about customers who are registered with the company but who have not placed any orders during the time frame obtained previously. By joining these two tables, we are able to obtain the names of the cities and states where orders were placed during the given time period.

Observations

- From the output, we see that there are 4119 unique cities and 27 unique states across Brazil from which the orders were placed during the given time period.
- Since Brazil has 26 states and one federal district, we can conclude that orders were placed from areas throughout the country.

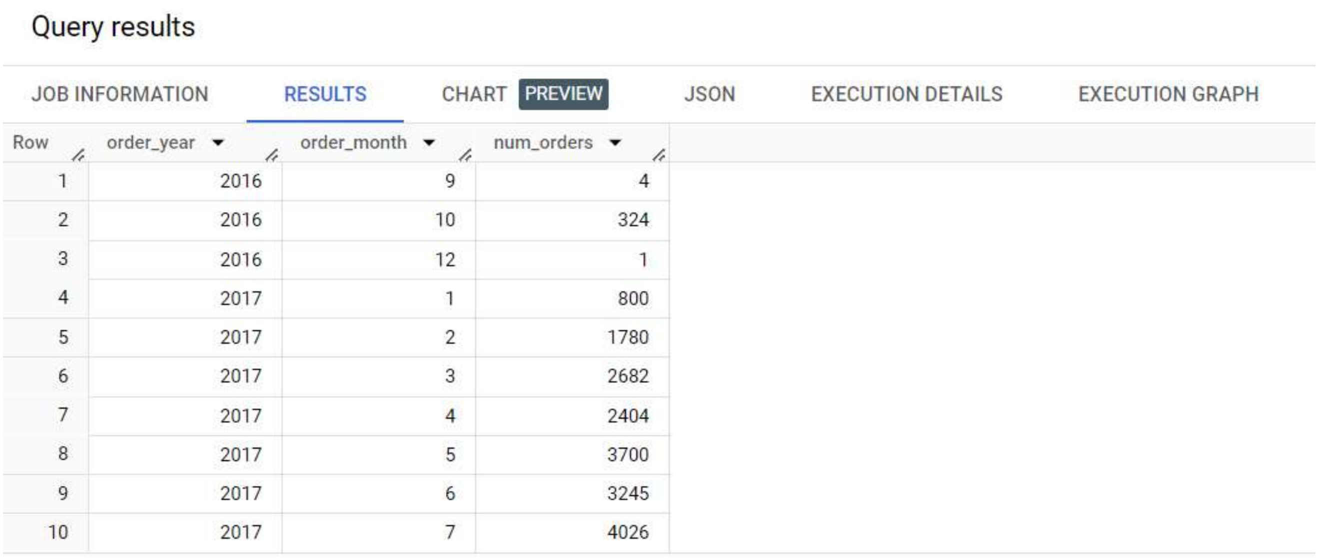
II. In-depth Exploration

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

SQL Code

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

Output



Query results

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAILS | EXECUTION GRAPH

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

Discussion

- To understand the trend in the number of orders placed over the past years, we must count the number of orders placed in each month during the time period of interest for which this data is provided.
- In order to this, we have to apply the `COUNT()` aggregate function to count the number of orders placed grouped by year and month.

Observations

- From the output, we see that there are only a few orders in the initial months of operation (in 2016). Starting 2017, we observe a gradual and sustained growth in the number of orders placed per month, which peaks in November 2017. Post this peak, there appears to be a sudden drop in the number of orders, which eventually picks up and stabilizes till August 2018, followed by a drastic fall.

Recommendations/Actionable Insights

- The causes for the sudden drop in sales in September 2018 must be determined and corresponding action plans must be carried out to ensure a swift recovery.

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

Observations

- We observe that the number of orders placed on the Target Ecommerce site peaked at November 2017.
- This peak in November may be attributed to the string of holidays extending from Thanksgiving & Black Friday and in anticipation of the approaching holiday season at the end of the year.

Recommendations/Actionable Insights

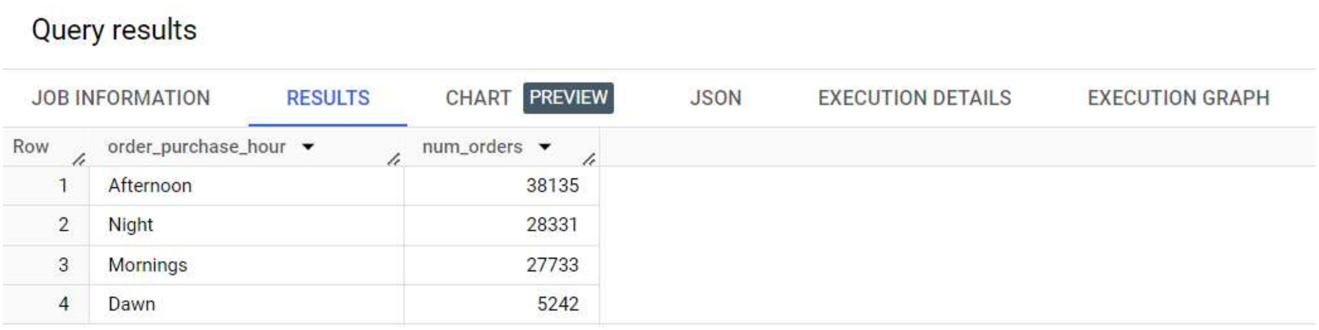
- To keep up with the peak in demand for products during this season, we can recommend the company to maintain additional stock of items in their inventory to ensure short delivery times.
- Further, to keep up with the increased number of orders during this period, it is recommended to hire a greater number of delivery partners and other workforce.
- A higher number of orders placed also means that there is a proportional increase in the traffic on the ecommerce site (not all searches will result in orders). The company must ensure that its website is equipped with additional servers to ensure smooth searching and shopping experience for its customers.
- The company can conduct marketing campaigns and offer special discounts during this season to boost sales.

C. During what time of the day, do the Brazilian customers mostly place their orders?

SQL Code

```
SELECT order_purchase_hour, COUNT(order_purchase_hour) AS num_orders
FROM
(SELECT
CASE
WHEN EXTRACT(HOUR FROM order_purchase_timestamp)<=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"
WHEN EXTRACT(HOUR FROM order_purchase_timestamp)>=19 THEN "Night"
END AS order_purchase_hour
FROM `target.orders`) AS hour_details
GROUP BY order_purchase_hour
ORDER BY num_orders DESC;
```

Output



Row	order_purchase_hour	num_orders
1	Afternoon	38135
2	Night	28331
3	Mornings	27733
4	Dawn	5242

Discussion

- The time of the day (Dawn, Morning, Afternoon or Night) during which the order was placed is found based on the hour at which the order was placed extracted from the `order_purchase_timestamp` column and the number of orders placed in each time slot is computed.

Observations

- From the output, we see that Brazilian customers placed the highest number of orders in the afternoon, followed by the night time, and then mornings, and the least number of orders were placed during dawn period.
- The number of orders placed is significantly lower in the “Dawn” time slot.

Recommendations/Actionable Insights

- In order to maximize the reach and impact of promotional campaigns and advertisements, it is recommended to synchronize them with the peak hours of traffic and orders on the ecommerce website.
- Since maximum orders are placed during the afternoon and nigh time slots, it is advisable to increase the number of customer support executives working during these hours in order to assist customers with queries or other issues.
- It is recommended that website maintenance activities be scheduled during the “Dawn” phase to minimize disruption in service to customers.

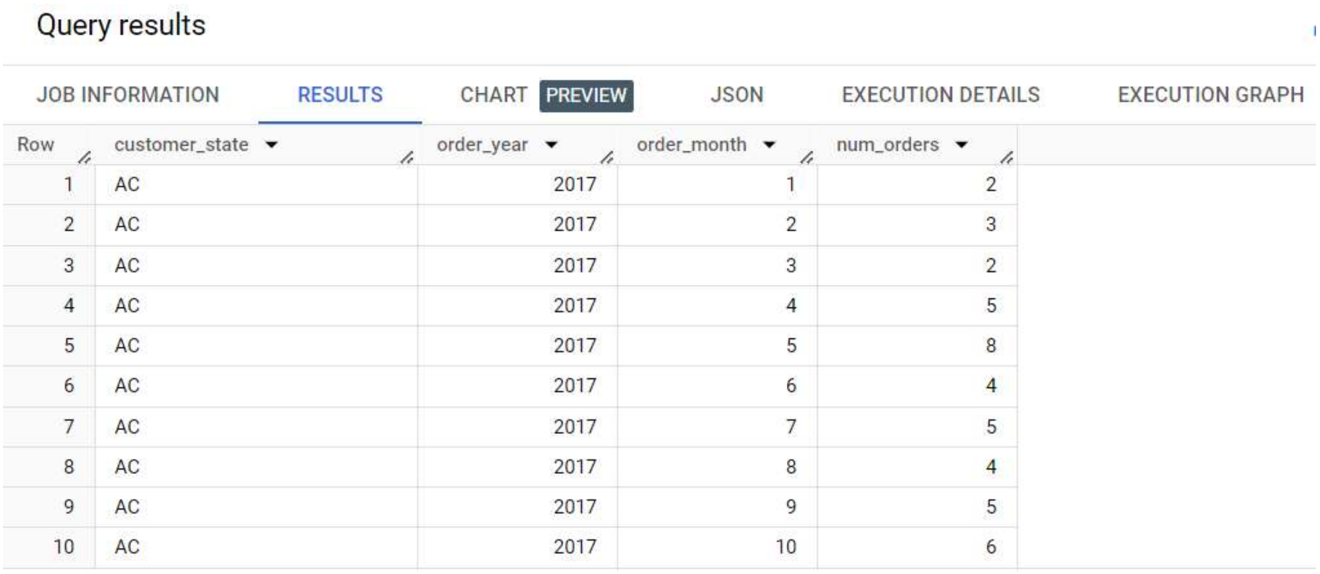
III. Evolution of E-commerce orders in the Brazil region

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

SQL Code

```
SELECT C.customer_state,
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 num_orders
FROM `target.orders` AS O
INNER JOIN `target.customers` AS C
ON O.customer_id=C.customer_id
GROUP BY C.customer_state,order_year,order_month
ORDER BY C.customer_state,order_year,order_month;
```

Output



Query results

Row	customer_state	order_year	order_month	num_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

Discussion

- The `orders` and `customers` tables are joined together using the `customer_id` column.
- The orders are then grouped by state and the year and month in which they were placed.
- The number of orders placed in each of the 12 months across the time period of this data set across all 27 states is counted to get the month on month number of orders placed in each state.

Observations

- A preliminary scan of the output indicates monthly variation in the orders for each state.
- In most cases, we observe a consistent increase in the number of orders from the start of the year to the end, with a peak around November.

Recommendations/Actionable Insights

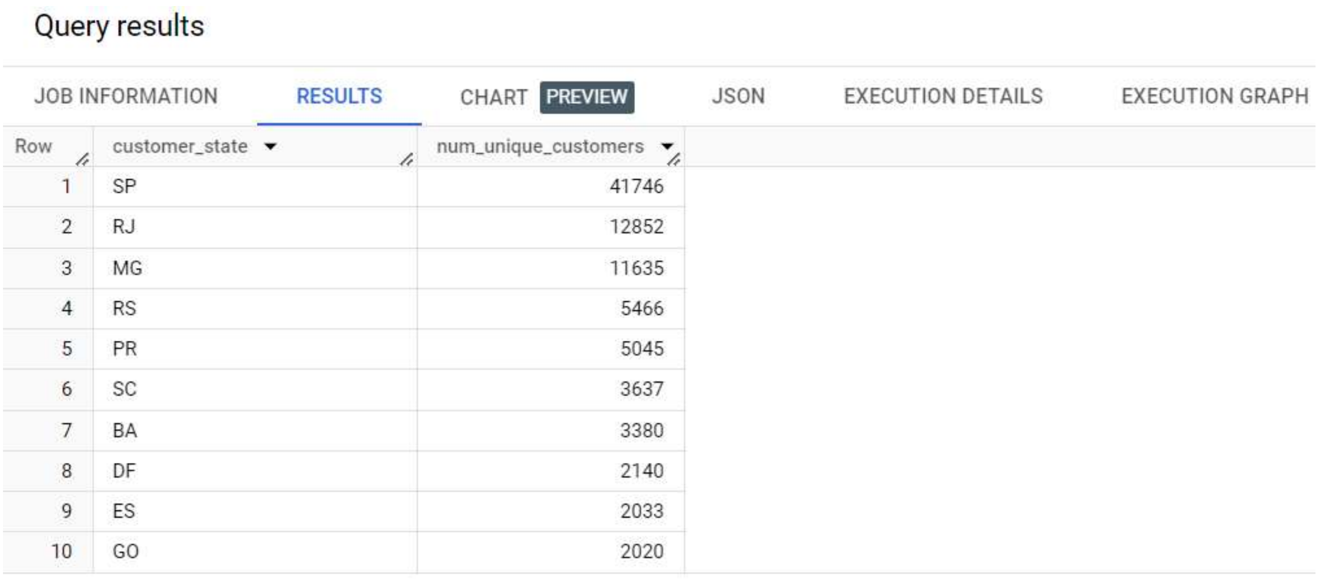
- It is recommended that the company maintain sufficient stocks of products in its warehouses throughout the country to facilitate quick delivery of items ordered in anticipation of the peak season.
- Items must be stocked in warehouses in such a way that states with higher magnitudes of order numbers receive greater volumes to meet their expected demand.
- Increase recruitment of workforce and boost logistical efficiency in states with higher orders as well as during expected peak seasons.

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

SQL Code

```
SELECT customer_state, COUNT(DISTINCT customer_id) AS num_unique_customers
FROM `target.customers`
GROUP BY customer_state
ORDER BY num_unique_customers DESC;
```

Output



Row	customer_state	num_unique_customers
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

Discussion

- To understand the distribution of customers across the different states of Brazil, we have to apply the `COUNT()` aggregate function to count the number of customers present in each state based on the data in the `customers` table.

Observations

- From the output, we observe that state "SP" accounts for an unusually large number of customers, followed by "RJ" and "MG". Thereafter, there is a drastic drop in the number of customers in the other states.

Recommendations/Actionable Insights

- In anticipation of higher order volumes in states with more customers, inventories must be kept sufficiently stocked with products to meet the demand.
- Measures must be taken to increase customers in states with low customer counts through promotional activities and discounts as these regions may represent untapped markets.
- The large customer base in certain states may be due to the higher population of these states. A large market also brings the threat of competitors. Hence, customer feedback must be collected efforts must be taken to retain existing customers in these regions.

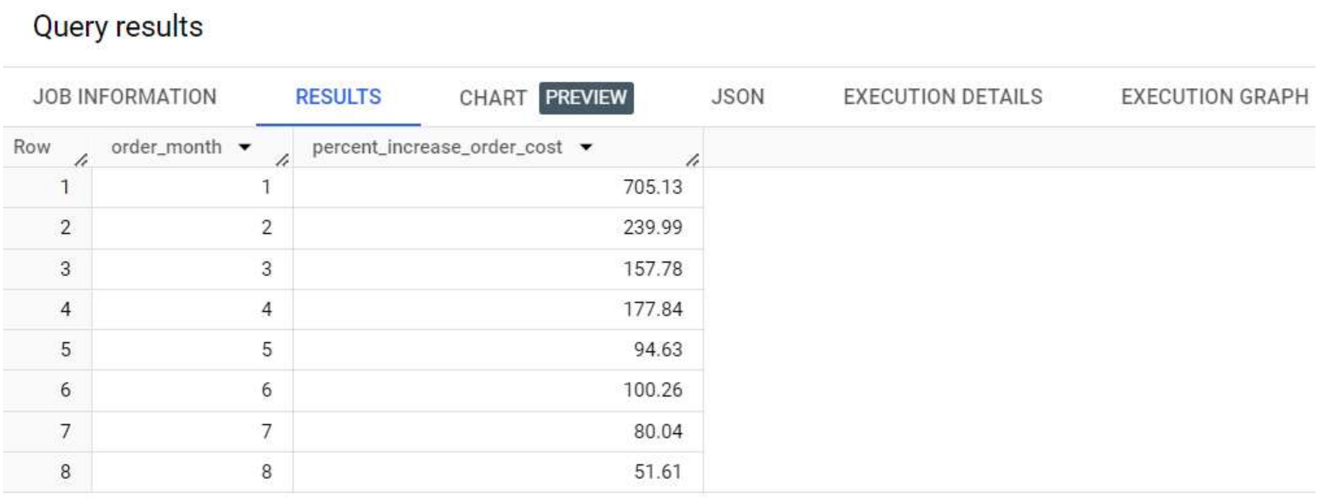
IV. Impact on Economy

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

SQL Code

```
SELECT order_month, ROUND((next_year_same_month_total_cost - total_cost)/total_cost*100,2) AS
percent_increase_order_cost
FROM
(SELECT *,
LEAD (total_cost, 1, 0) OVER (PARTITION BY order_month ORDER BY order_year) AS
next_year_same_month_total_cost
FROM
(SELECT EXTRACT(YEAR FROM O.order_purchase_timestamp) AS order_year,
EXTRACT(MONTH FROM O.order_purchase_timestamp) AS order_month,
SUM(P.payment_value) AS total_cost
FROM `target.orders` AS O
INNER JOIN `target.payments` AS P
ON O.order_id=P.order_id
WHERE EXTRACT(YEAR FROM O.order_purchase_timestamp) IN (2017,2018) AND EXTRACT(MONTH FROM
O.order_purchase_timestamp) BETWEEN 1 AND 8
GROUP BY 1,2
ORDER BY 1,2) AS dt1
ORDER BY order_year,order_month) AS dt2
WHERE order_year=2017
ORDER BY order_month;
```

Output



Row	order_month	percent_increase_order_cost
1	1	705.13
2	2	239.99
3	3	157.78
4	4	177.84
5	5	94.63
6	6	100.26
7	7	80.04
8	8	51.61

Discussion

- The `orders` and `payments` tables are joined together using the order_`id` column.
- The data is filtered to only include orders placed in 2017 and 2018 for the months between Jan to August only. The filtered records are stored in a derived table.
- The total cost of orders by year and month are calculated and stored in a derived table.
- The `LEAD()` is used to obtain the corresponding total cost for the same month in the next year.
- The percentage increase in cost is calculated for each month from January to August from the year 2017 to 2018.

Observations

- We observe that the largest percentage of increase in the order cost from 2017 to 2018 was observed in January with a drop in the increase in subsequent months.

Recommendations/Actionable Insights

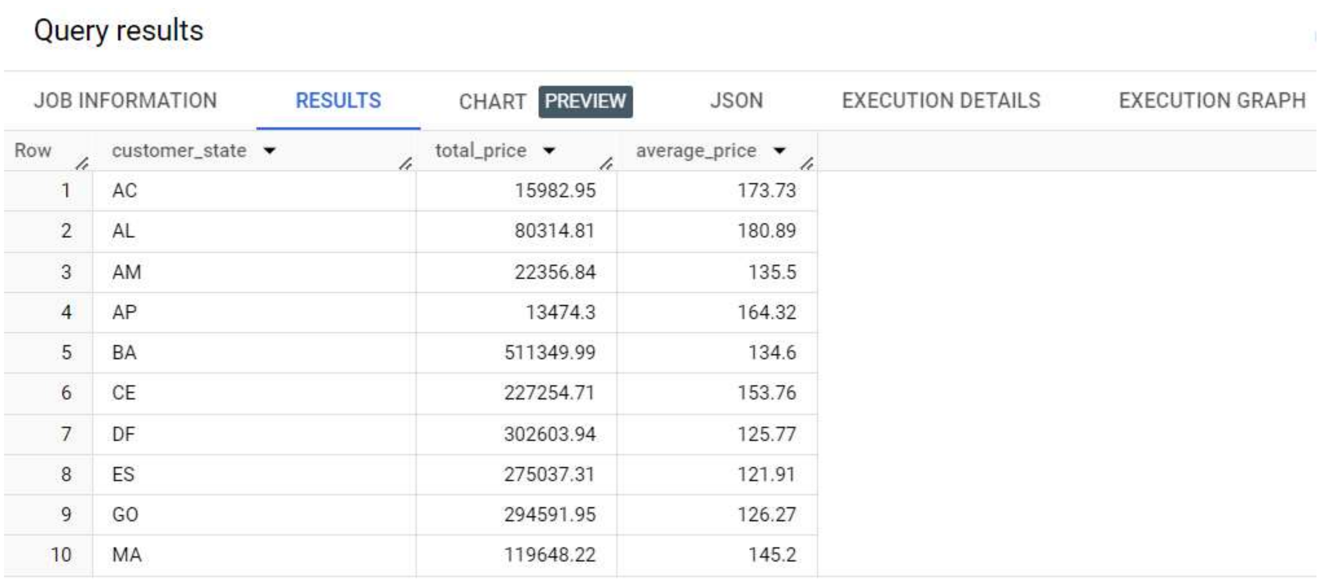
- The unusually large values of increase can be ignored as they are commonly observed during the initial phase of growth of a company. However, the company should aim to main a growth rate of at least 15-20% from one year to the next.
- In order to maintain a consistent level of growth, it is import to ensure a smooth customer experience on the website and conduct promotional programs as well as offer seasonal.

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

SQL Code

```
SELECT C.customer_state,
ROUND(SUM(OI.price),2) AS total_price,
ROUND(AVG(OI.price),2) AS average_price
FROM `target.orders` AS O
INNER JOIN `target.order_items` AS OI
ON O.order_id=OI.order_id
INNER JOIN `target.customers` AS C
ON O.customer_id=C.customer_id
GROUP BY C.customer_state
ORDER BY C.customer_state;
```

Output


Query results

Row	customer_state	total_price	average_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

Discussion

- The orders and order_items tables are joined together using the order_id column.
- The resultant table is joined with the customers tables are joined together using the customer_id column.
- The records are grouped by the customer_state column and the aggregate functions SUM() and AVG() is used to calculate the total price and the average price of orders for each state.

Observations

- There are noticeable variations in the total price and the average price of orders among the different states of Brazil.
- As expected, the states with larger order volumes (based on previous analysis) such as SP, RJ, and MG have larger total order prices. However, these states tend to have a lower average price of sales. This may be due to the reason that a large number of smaller orders are placed in these states which bring down the average price of orders.

Recommendations/Actionable Insights

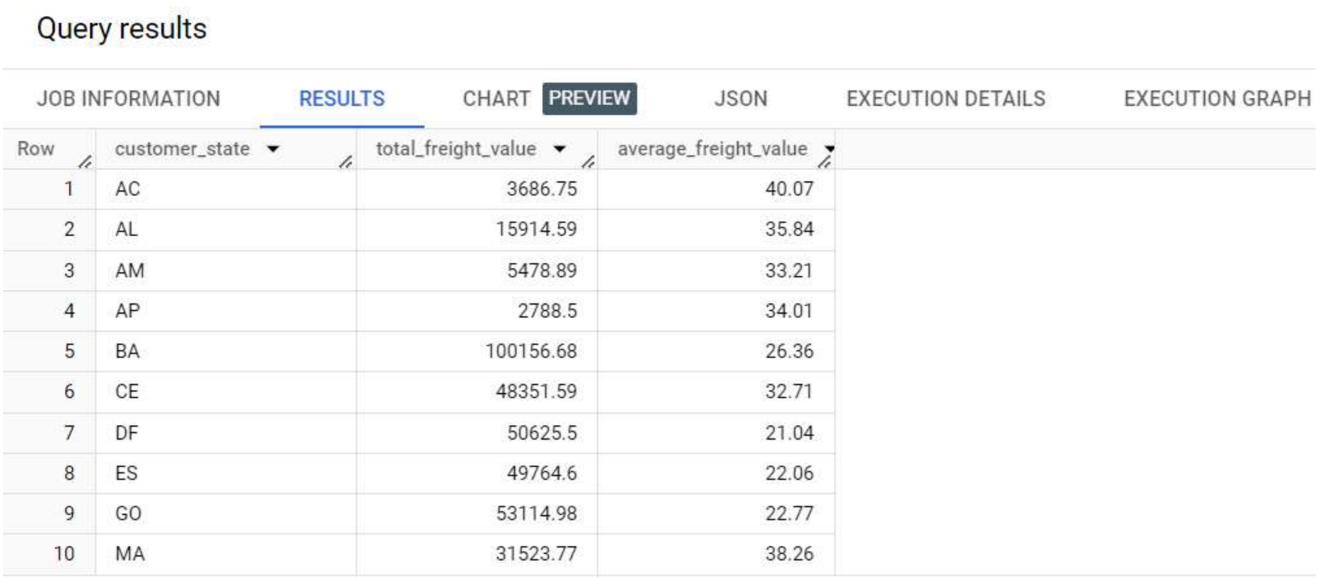
- Market segmentation strategies can be implemented to tailor pricing and product offerings to specific customer segments within each state.
- Introduce dynamic pricing for products based on the state, demand, seasonality, and the perceived willingness of customers to pay a premium for products.

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

SQL Code

```
SELECT C.customer_state,
ROUND(SUM(OI.freight_value),2) AS total_freight_value,
ROUND(AVG(OI.freight_value),2) AS average_freight_value
FROM `target.orders` AS O
INNER JOIN `target.order_items` AS OI
ON O.order_id=OI.order_id
INNER JOIN `target.customers` AS C
ON O.customer_id=C.customer_id
GROUP BY C.customer_state
ORDER BY C.customer_state;
```

Output



Row	customer_state	total_freight_value	average_freight_value
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

Discussion

- The `orders` and `order_items` tables are joined together using the order_`id` column.
- The resultant table is joined with the `customers` tables are joined together using the `customer_id` column.
- The records are grouped by the `customer_state` column and the aggregate functions `SUM()` and `AVG()` is used to calculate the total freight value and the average freight value of orders for each state.

Observations

- There are noticeable variations in the total freight value and the average freight value of orders among the different states of Brazil.
- As expected, the states with larger order volumes (based on previous analysis) such as SP, RJ, and MG have larger total freight values. However, these states tend to have a lower average freight values. This may be due to the reason that shipping to these regions is cheaper due to the continuous demand.

Recommendations/Actionable Insights

- In general, we see that boosting sales in any state, leads to reduction in the freight values. Hence, measures such as promotional activities and seasonal discounts on certain products can be offered which has the potential to not only increase the demand for products during certain periods but also makes the demand more predictable, thereby making it cheaper to ship these products from seller to buyer.
- Improve the logistics and warehouse efficiency to reduce the freight value for states with higher averages.

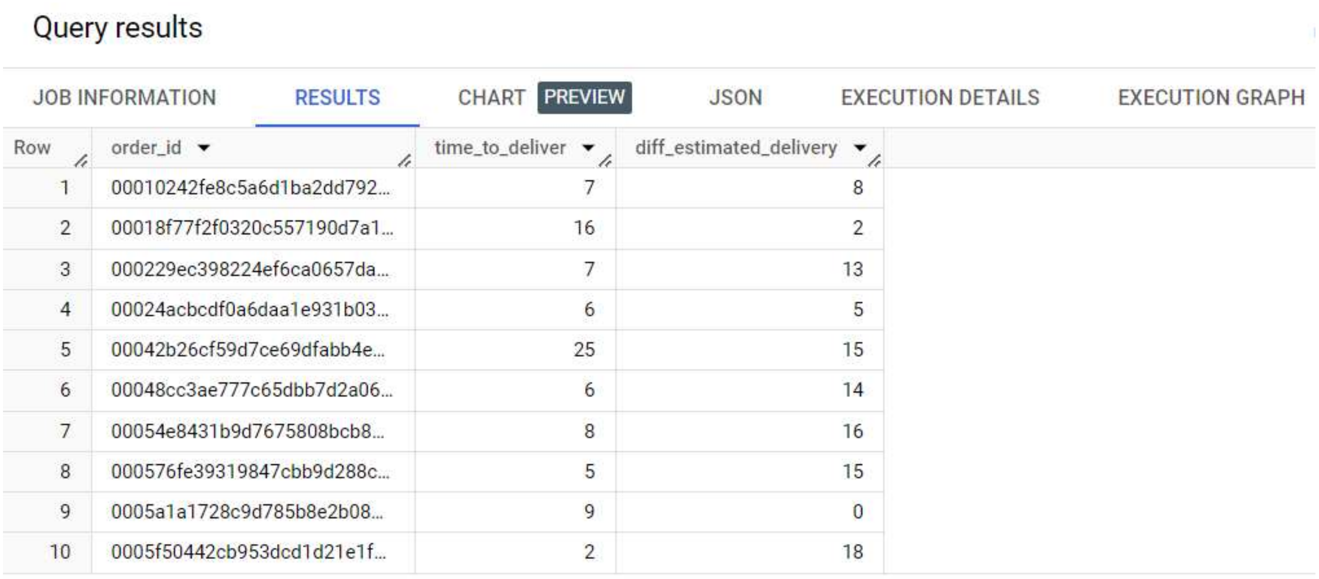
V. Analysis on sales, freight and delivery time

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.

SQL Code

```
SELECT order_id,
DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp,day) AS time_to_deliver,
DATE_DIFF(order_estimated_delivery_date,order_delivered_customer_date,day) AS diff_estimated_delivery
FROM `target.orders`
WHERE order_delivered_customer_date IS NOT NULL
ORDER BY order_id;
```

Output



Row	order_id	time_to_deliver	diff_estimated_delivery
1	00010242fe8c5a6d1ba2dd792...	7	8
2	00018f77f2f0320c557190d7a1...	16	2
3	000229ec398224ef6ca0657da...	7	13
4	00024acbcdf0a6daa1e931b03...	6	5
5	00042b26cf59d7ce69dfabb4e...	25	15
6	00048cc3ae777c65dbb7d2a06...	6	14
7	00054e8431b9d7675808bcb8...	8	16
8	000576fe39319847cbb9d288c...	5	15
9	0005a1a1728c9d785b8e2b08...	9	0
10	0005f50442cb953dcd1d21e1f...	2	18

Discussion

- There are some orders for which the `order_delivered_customer_date` contains `NULL` values. This may be due to the fact that the order was not delivered to the customer. We can ignore these records.
- The `DATE_DIFF()` function is used to find the number of days taken to deliver an order and the difference between the estimated & actual delivery date of an order.

Observations

- An inspection of the `time_to_deliver` column reveals that the delivery time varies from 0 days (same day/less than 24 hours delivery) to several days.
- The `diff_estimated_delivery` column contains both positive and negative values. Positive values indicate that the order was delivered to the customer ahead of the estimated date, while negative values indicate a delay in order delivery.

Recommendations/Actionable Insights

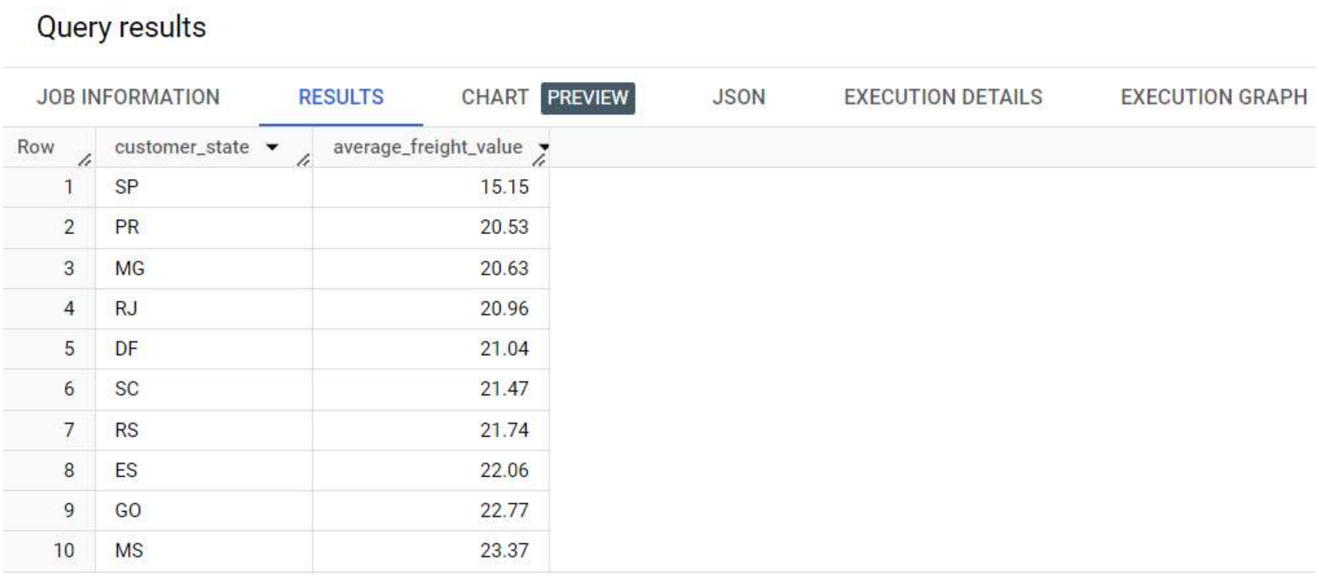
- Investigations must be carried out to determine the reason for delay in delivery of some orders.

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

SQL Code

```
SELECT C.customer_state, ROUND(AVG(freight_value),2) AS average_freight_value
FROM `target.orders` AS O
INNER JOIN `target.order_items` AS OI
ON O.order_id=OI.order_id
INNER JOIN `target.customers` AS C
ON O.customer_id=C.customer_id
GROUP BY C.customer_state
ORDER BY average_freight_value;
```

Output



Query results

Row	customer_state	average_freight_value
1	SP	15.15
2	PR	20.53
3	MG	20.63
4	RJ	20.96
5	DF	21.04
6	SC	21.47
7	RS	21.74
8	ES	22.06
9	GO	22.77
10	MS	23.37

Discussion

- The `orders` and `order_items` tables are joined together using the order_`id` column.
- The resultant table is joined with the `customers` tables are joined together using the `customer_id` column.
- The records are grouped by the `customer_state` column and the aggregate function `AVG()` is used to calculate the average freight value for each state.

Observations

- There is disparity in the freight values across the different states of Brazil.
- The 5 states with the highest freight value (in descending order) are : RR, PB, RO, AC, and PI.
- The 5 states with the lowest freight value (in ascending order) are : SP, PR, MG, RJ, and DF.

Recommendations/Actionable Insights

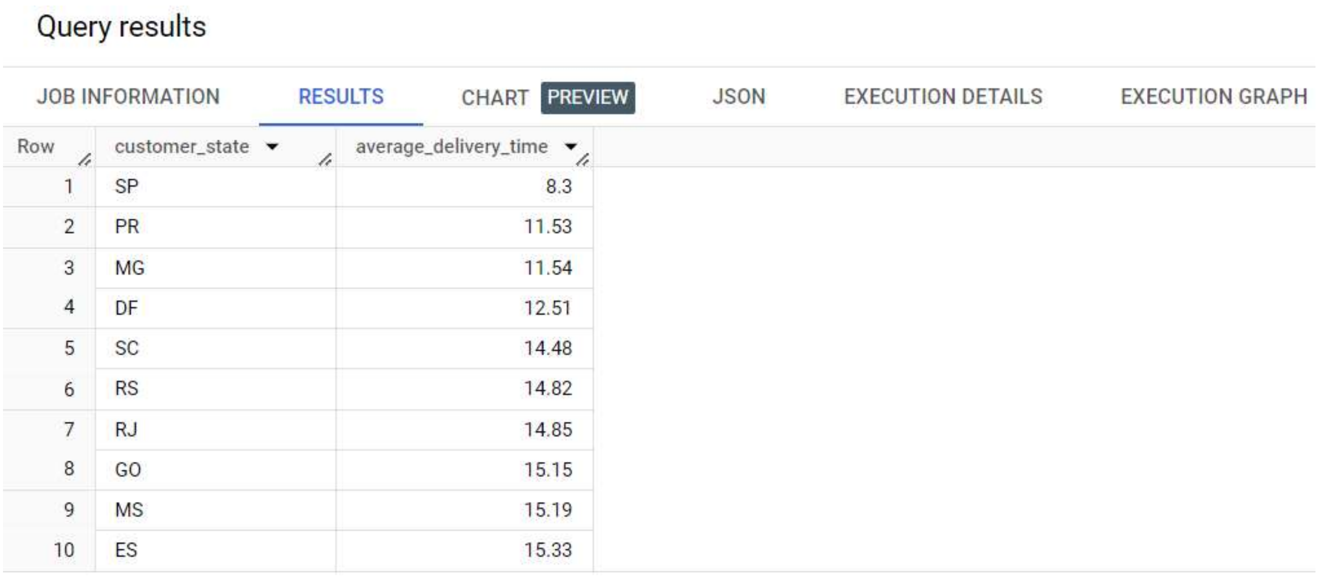
- Try to find out the reasons for higher freight charges in certain states and explore options such as tying up with local logistics services to reduce these costs. Some likely reasons include low population concentration of a state as well as challenging topography or inaccessibility.
- Pass on the benefits of lower freight charges to customers in certain states so that the price of items is more attractive to customers in these regions.
- Introduce loyalty programs to provide lower shipping rates for customers in states where the freight values are on the higher side.

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

SQL Code

```
SELECT C.customer_state, ROUND(AVG(dt.time_to_deliver),2) AS average_delivery_time
FROM
(SELECT order_id,
customer_id,
DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp,day) AS time_to_deliver,
FROM `target.orders`
WHERE order_delivered_customer_date IS NOT NULL
ORDER BY order_id) AS dt
INNER JOIN `target.customers` AS C
ON dt.customer_id=C.customer_id
GROUP BY C.customer_state
ORDER BY average_delivery_time;
```

Output



Row	customer_state	average_delivery_time
1	SP	8.3
2	PR	11.53
3	MG	11.54
4	DF	12.51
5	SC	14.48
6	RS	14.82
7	RJ	14.85
8	GO	15.15
9	MS	15.19
10	ES	15.33

Discussion

- The `DATE_DIFF()` function is used to find the delivery time (in days) for each order.
- The derived table contain the `order_id`, `customer_id`, and `time_to_deliver` information for each order is joined together with the `customers` column using the `customer_id` column.
- The records are grouped by the `customer_state` column and the aggregate function `AVG()` is used to calculate the average delivery time for each state.

Observations

- There is disparity in the delivery time of orders across the different states of Brazil.
- The 5 states with the highest freight value (in descending order) are : RR, AP, AM, AL, and PA.
- The 5 states with the lowest freight value (in ascending order) are : SP, PR, MG, RJ, and SC.

Recommendations/Actionable Insights

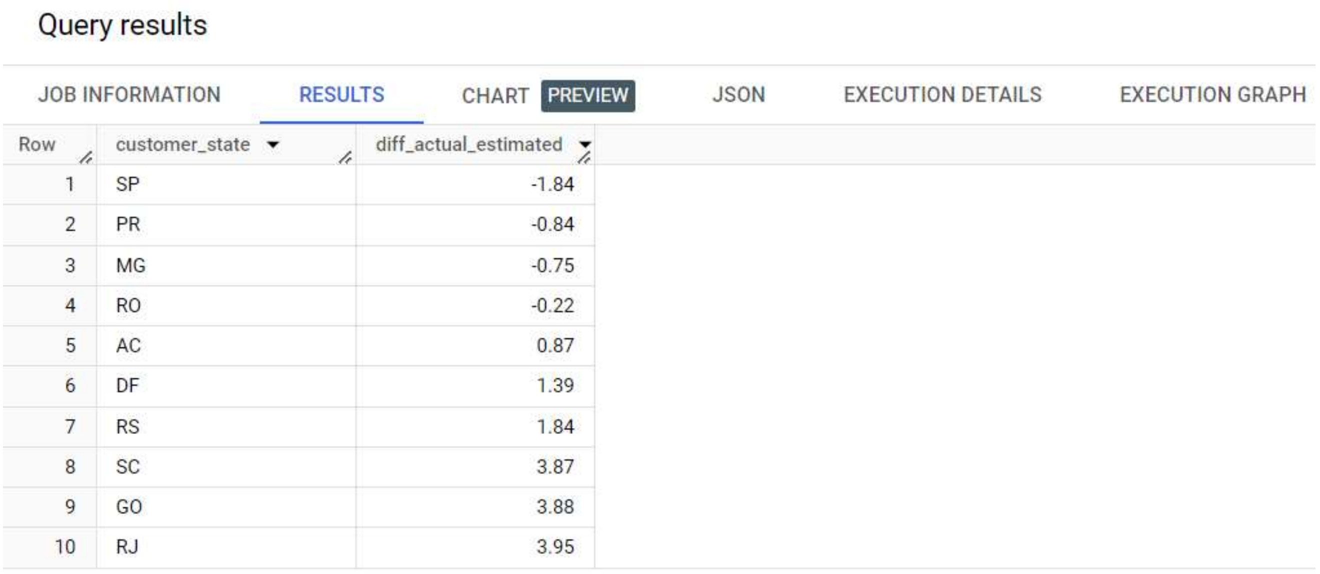
- Try to find out the reasons for higher delivery times in certain states. Potential reasons include difficulty in accessing cities in certain states due to their remoteness or challenging topography as well as low population density which implies that there are lesser travel options available to such places.
- In states with lesser delivery times, use it as a marketing tool to attract customers.
- Build warehouses in strategic locations to store non-perishable items in regions with larger average delivery times to speed up the delivery process.

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

SQL Code

```
SELECT C.customer_state,
ROUND(AVG(dt.time_to_deliver)-AVG(dt.diff_estimated_delivery),2) AS diff_actual_estimated,
FROM
(SELECT order_id,customer_id,
DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp,day) AS time_to_deliver,
DATE_DIFF(order_estimated_delivery_date,order_delivered_customer_date,day) AS diff_estimated_delivery
FROM `target.orders`
WHERE order_delivered_customer_date IS NOT NULL
ORDER BY order_id) AS dt
INNER JOIN `target.customers` AS C
ON dt.customer_id=C.customer_id
GROUP BY C.customer_state
ORDER BY diff_actual_estimated;
```

Output



Row	customer_state	diff_actual_estimated
1	SP	-1.84
2	PR	-0.84
3	MG	-0.75
4	RO	-0.22
5	AC	0.87
6	DF	1.39
7	RS	1.84
8	SC	3.87
9	GO	3.88
10	RJ	3.95

Discussion

- We must only include orders that are already delivered to the customer.
- The `DATE_DIFF()` function is used to find the number of days taken to deliver an order and the difference between the estimated & actual delivery date of an order.
- The derived table contain the `order_id`, `customer_id`, `time_to_deliver`, and `diff_estimated_delivery` information for each order is joined together with the `customers` column using the `customer_id` column.
- The records are grouped by the `customer_state` column and the difference between the averages of the actual and estimate delivery times for each state are calculated. The data is ordered in increasing order of these differences.

Observations

- A preliminary scan of the output indicates that there is disparity in the values of these differences across the different states in Brazil.
- We observe that some values are negative (indicating that orders in these states are delivered ahead of the estimated date, on average) and others are positive (indicating that orders in these states are delivered after the estimated date, on average).

- The top 5 states where the order delivery is really fast as compared to the estimated date of delivery are SP, PR, MG, RO, and AC.

<u>Recommendations/Actionable Insights</u>

- Since in all but 4 states, the average of these differences is positive, it means that in most states, the orders are delivered later than the estimated delivery date promised to the customer. To prevent dissatisfaction among customers, it is important to provide realistic delivery timelines to customers when they place the orders.
- Determine the causes for longer-than-estimated delivery times for states with large positive difference values in the above table. Focus on optimizing the controllable such as strategic building of warehouse locations and improving logistical efficiency.
- In states with faster-than-expected delivery, the speedy delivery service can be leveraged as a promotional tool to attract more customers.
- Brazil is a large country with diverse geographical regions ranging from the highly populated coastal regions to the dense Amazon forest. Seasonal factors may play a role in certain states. It is important to develop tailor-made logistics strategies for different states taking into account population, time of year and topography.

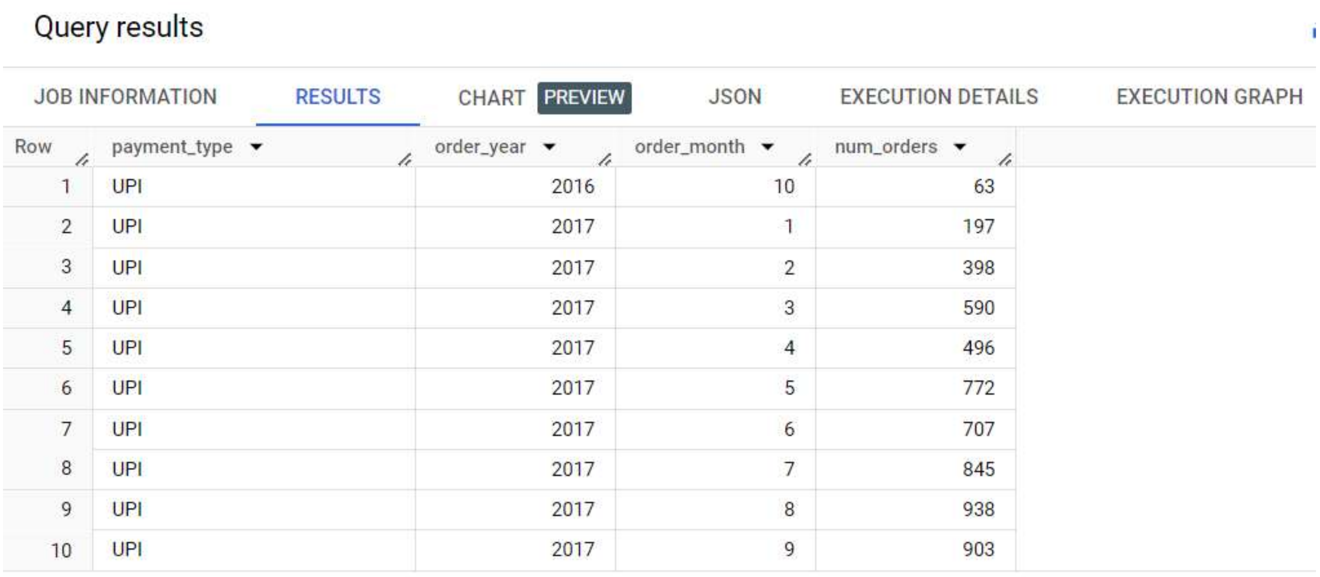
VI. Analysis based on the payments

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

SQL Code

```
SELECT P.payment_type,
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 num_orders
FROM `target.orders` AS O
INNER JOIN `target.payments` AS P
ON O.order_id=P.order_id
GROUP BY P.payment_type,order_year,order_month
ORDER BY P.payment_type,order_year,order_month;
```

Output


Query results

Row	payment_type	order_year	order_month	num_orders
1	UPI	2016	10	63
2	UPI	2017	1	197
3	UPI	2017	2	398
4	UPI	2017	3	590
5	UPI	2017	4	496
6	UPI	2017	5	772
7	UPI	2017	6	707
8	UPI	2017	7	845
9	UPI	2017	8	938
10	UPI	2017	9	903

Discussion

- The `orders` and `payments` tables are joined together using the `order_id` column.
- Basic exploratory analysis of the `orders` and `payments` tables indicates that while there are 99441 orders (all of them unique) present in the `orders` table, there are only 99440 unique orders present in the `payments` table. This means that there is no payment-related information for one of the orders placed.
- Further, the payments table contains a total of 103886 `order_id` values (of which 99440 are unique). This indicates that some orders were paid for using multiple payment types.
- The inner join or right join can be performed between the `orders` and `payments` tables
- The orders are then grouped by the payment type and the month in which they were placed.
- The number of orders placed in each of the 12 months using the 4 payment types is counted to get the month on month no. of orders placed using different payment types.

Observations

- A preliminary scan of the output indicates monthly fluctuations in the number of orders placed for each payment type (we can ignore the data where the `payment_type` is "not_defined" as there are very few such orders).
- We observe that the volumes of orders paid for using credit cards is of a higher magnitude as compared to the other payment types, followed by UPI, then voucher and debit card.

- A thorough analysis of this data reveals that the month on month number of orders placed using each payment type follows a similar pattern. In each case, there appears to be a consistent growth in the number of orders placed from the beginning of the year to the end, with a peak value around November, perhaps due to the festive season.

Recommendations/Actionable Insights

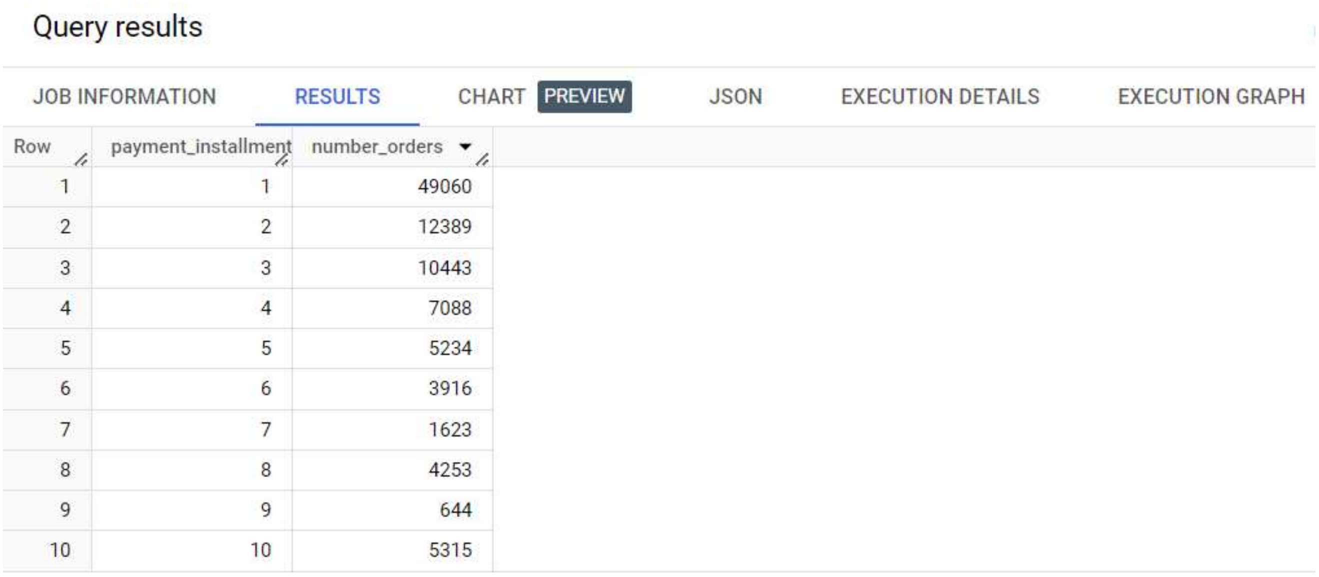
- Item discounts or EMI options linked to credit cards or other payment types can be offered to boost sales during the peak or festive seasons.
- Promotional offers and discounts linked to credit cards can also be offered during months of low order volumes to boost sales.
- To ensure data integrity, the reasons for instances of "not_defined" payment types must be investigated and any issues involved in the data collection must be resolved.

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

SQL Code

```
SELECT payment_installments, COUNT(DISTINCT order_id) AS number_orders
FROM `target.payments`
WHERE payment_installments>=1
GROUP BY payment_installments
ORDER BY payment_installments
```

Output



Row	payment_installment	number_orders
1	1	49060
2	2	12389
3	3	10443
4	4	7088
5	5	5234
6	6	3916
7	7	1623
8	8	4253
9	9	644
10	10	5315

Discussion

- There are some
- The `COUNT()` aggregate function to count the number of orders placed based on the number of payment installments where at least one installment has been successfully paid.

Observations

- From the output, we observe that a majority of orders are single-installment orders. These means that most customers prefer to pay for orders at once.
- There are also a significant number of orders paid for in a few monthly installments.

Recommendations/Actionable Insights

- Promotions and no-cost EMI schemes up to a few months can be offered to customers as there appears to be a significant amount of customers interested in paying for orders in this way.

****