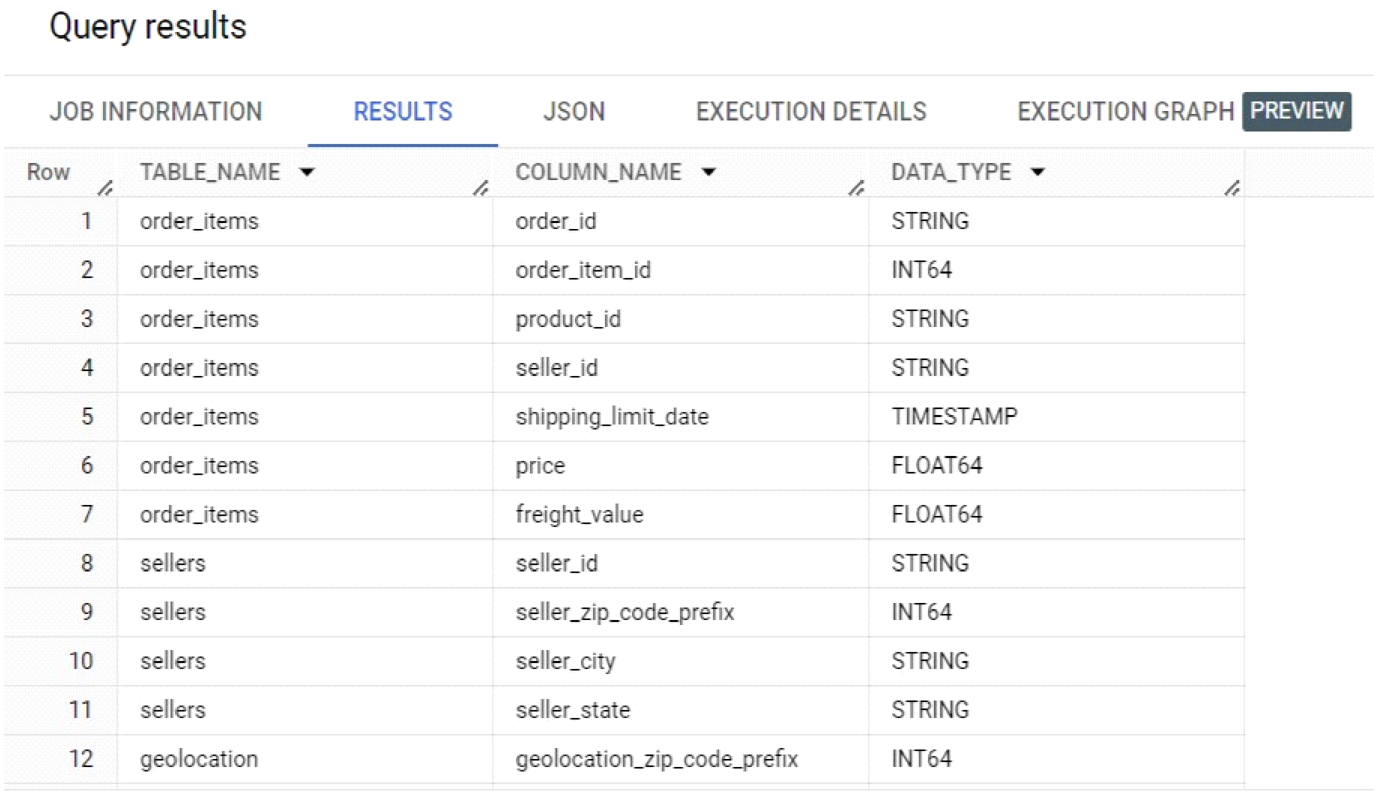
BUSINESS CASE : TARGET SQL

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

1) Data type of columns in a table -

QUERY :

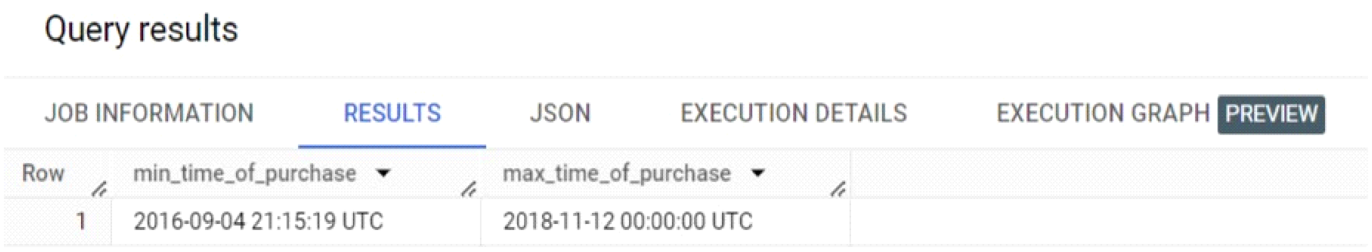
```
SELECT    TABLE_NAME, COLUMN_NAME, DATA_TYPE
FROM target_sql.INFORMATION_SCHEMA.COLUMNS;
```



Row	TABLE_NAME	COLUMN_NAME	DATA_TYPE
1	order_items	order_id	STRING
2	order_items	order_item_id	INT64
3	order_items	product_id	STRING
4	order_items	seller_id	STRING
5	order_items	shipping_limit_date	TIMESTAMP
6	order_items	price	FLOAT64
7	order_items	freight_value	FLOAT64
8	sellers	seller_id	STRING
9	sellers	seller_zip_code_prefix	INT64
10	sellers	seller_city	STRING
11	sellers	seller_state	STRING
12	geolocation	geolocation_zip_code_prefix	INT64

2) Time period for which the data is given -

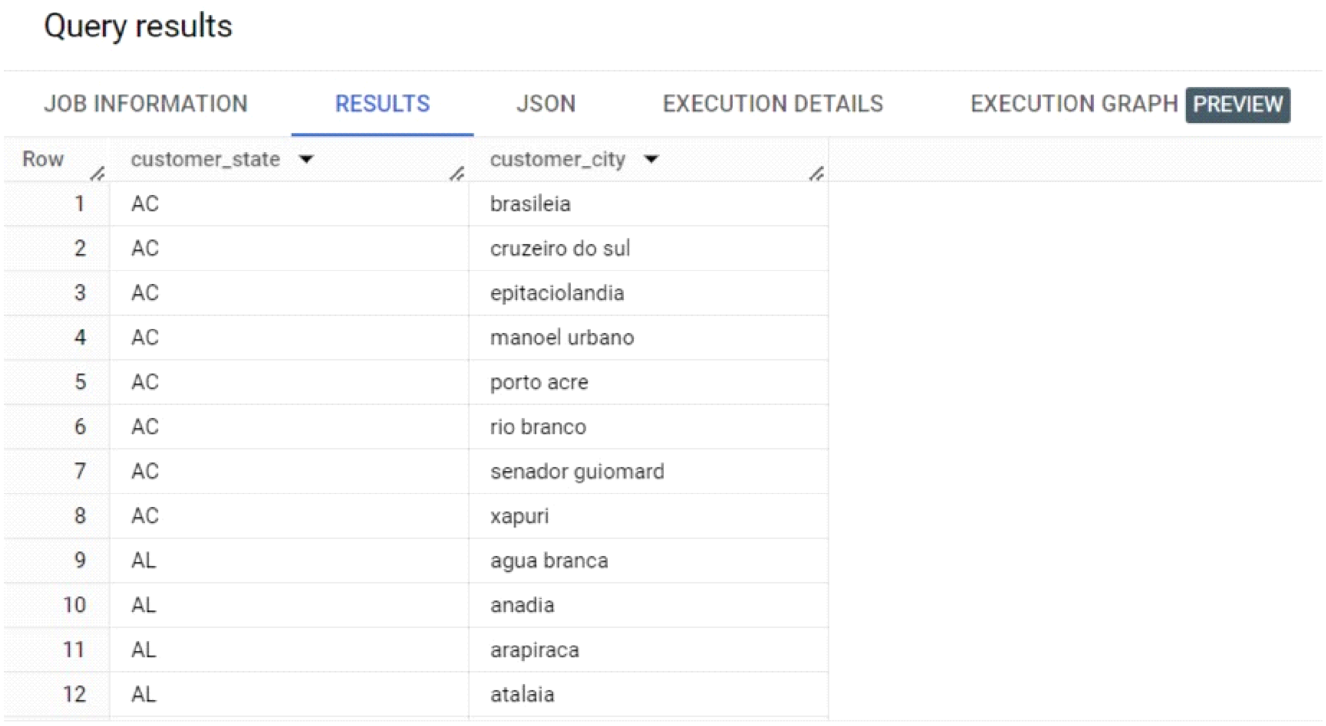
QUERY : SELECT MIN(order_purchase_timestamp) AS min_time_of_purchase,
MAX(order_estimated_delivery_date) AS
max_time_of_purchase
FROM `target_sql.orders`, `target_sql.order_items`;



Row	min_time_of_purchase	max_time_of_purchase
1	2016-09-04 21:15:19 UTC	2018-11-12 00:00:00 UTC

3) Cities and States of customers ordered during the given period-

QUERY : SELECT customer_state, customer_city
FROM `target_sql.customers`
GROUP BY customer_state, customer_city
ORDER BY customer_state, customer_city;


Query results

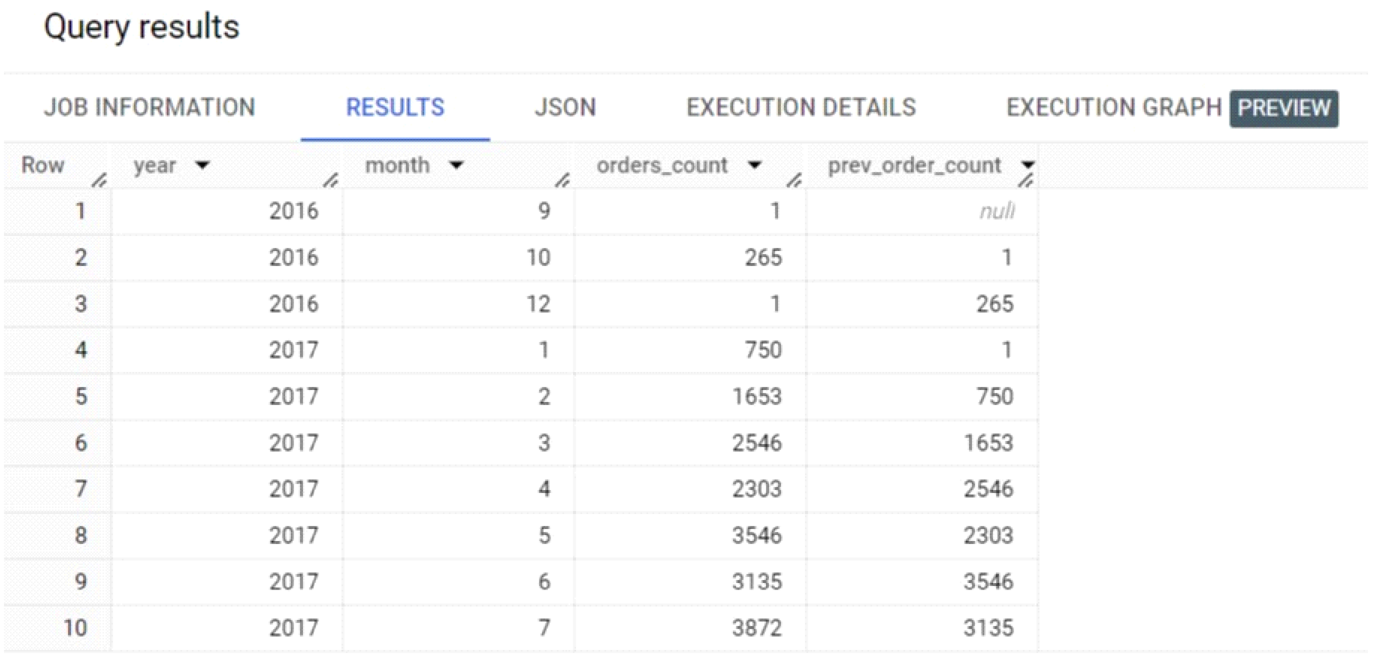
Row	customer_state	customer_city
1	AC	brasileia
2	AC	cruzeiro do sul
3	AC	epitaciolandia
4	AC	manoel urbano
5	AC	porto acre
6	AC	rio branco
7	AC	senador guiomard
8	AC	xapuri
9	AL	agua branca
10	AL	anadia
11	AL	arapiraca
12	AL	atalaia

2. In-depth Exploration :

1) Is there a growing trend on e-commerce in Brazil? How can we describe a complete scenario? Can we see some seasonality with peaks at specific months?

QUERY :

```
SELECT *,
       LAG(orders_count) OVER(ORDER BY year, month) AS prev_order_count
FROM(SELECT EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
            EXTRACT(MONTH FROM order_purchase_timestamp) AS month,
            COUNT(*) AS orders_count
     FROM `target_sql.orders`
     WHERE order_status ='delivered'
     GROUP BY 1,2)

ORDER BY year, month;
```



Row	year	month	orders_count	prev_order_count
1	2016	9	1	null
2	2016	10	265	1
3	2016	12	1	265
4	2017	1	750	1
5	2017	2	1653	750
6	2017	3	2546	1653
7	2017	4	2303	2546
8	2017	5	3546	2303
9	2017	6	3135	3546
10	2017	7	3872	3135

INSGHITS - i) There is a growing trend on e-commerce in Brazil from the year 2016 to 2018

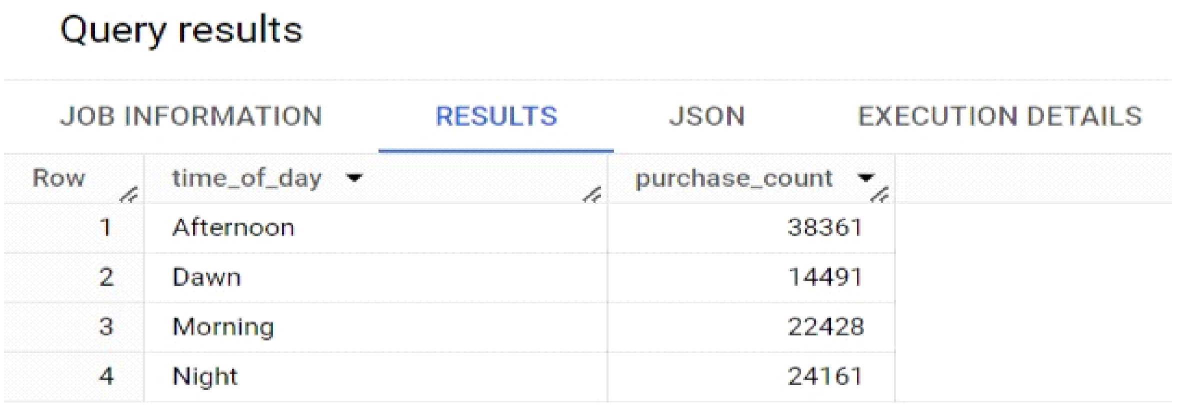
ii) Peak of the orders count occurred in November month of 2018 while there was a huge decrease in the orders count from August 2018 to September 2018.

2)What time do Brazilian customers tend to buy (Dawn, Morning, Afternoon or Night)?

QUERY : SELECT

```
CASE
  WHEN EXTRACT(HOUR FROM order_purchase_timestamp) >= 5 AND
  EXTRACT(HOUR FROM order_purchase_timestamp) < 12 THEN  'Morning'

  WHEN EXTRACT(HOUR FROM order_purchase_timestamp) >= 12 AND
  EXTRACT(HOUR FROM order_purchase_timestamp) < 18 THEN  'Afternoon'

       WHEN EXTRACT(HOUR FROM order_purchase_timestamp) >= 18 AND
       EXTRACT(HOUR FROM order_purchase_timestamp) < 22 THEN  'Night'
```

```
                    ELSE 'Dawn'
                END AS time_of_day,

                  COUNT(*) AS purchase_count
        FROM    `target_sql.orders`
        GROUP BY    time_of_day
        ORDER BY    time_of_day;
```



Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	time_of_day	purchase_count
1	Afternoon	38361
2	Dawn	14491
3	Morning	22428
4	Night	24161

INSIGHTS- i) It can be seen that the customers prefer making orders at night and afternoon rather than at dawn and morning.
ii) More than three fourth of the total orders are made during these two times of the day.

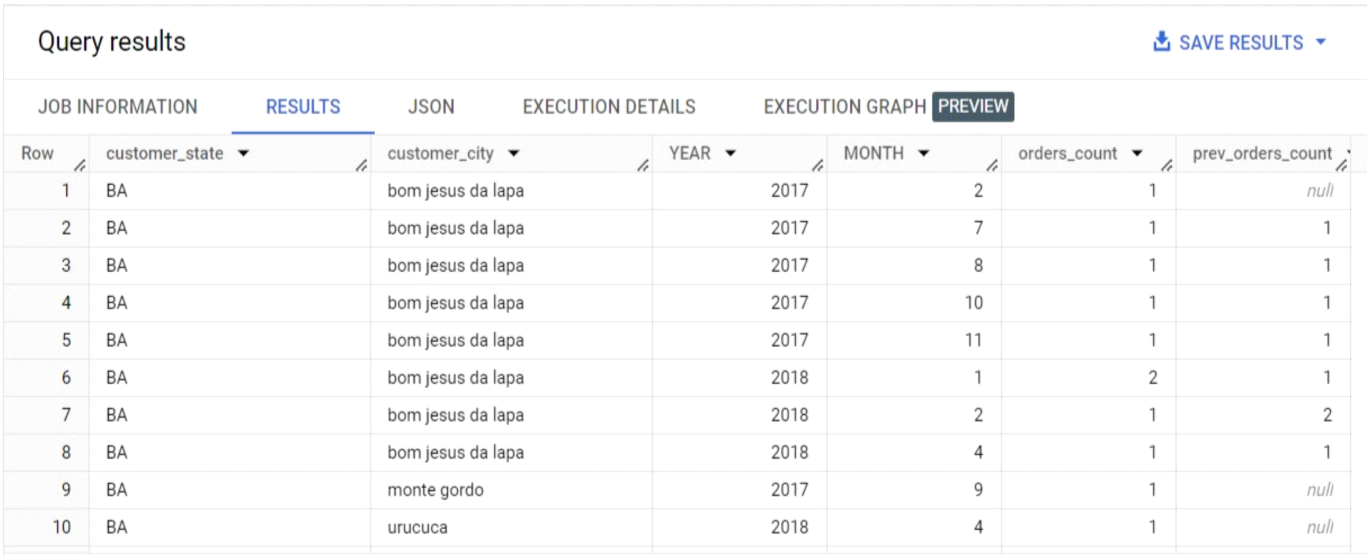
3. Evolution of E-commerce orders in the Brazil region:

1) Get month on month orders by states

QUERY :

```
SELECT    *,
          LAG(orders_count) OVER (PARTITION BY customer_state,
customer_city            ORDER BY YEAR, MONTH) AS prev_orders_count
        FROM    (SELECT
                    C.customer_state,
                    C.customer_city,
                    X1.YEAR,
                    X1.MONTH,
                    COUNT(*) AS orders_count
```

```
                        FROM `target_sql.customers` AS C
                              JOIN
                                   ( SELECT *,
                                   EXTRACT(MONTH FROM order_purchase_timestamp) AS
MONTH,
                                   EXTRACT(YEAR FROM order_purchase_timestamp) AS
YEAR
                                   FROM `target_sql.orders`
                                   WHERE order_status = 'delivered'    ) AS X1

                                   ON C.customer_id = X1.customer_id
                                   GROUP BY 1, 2, 3, 4
                                    ) AS X2;
```

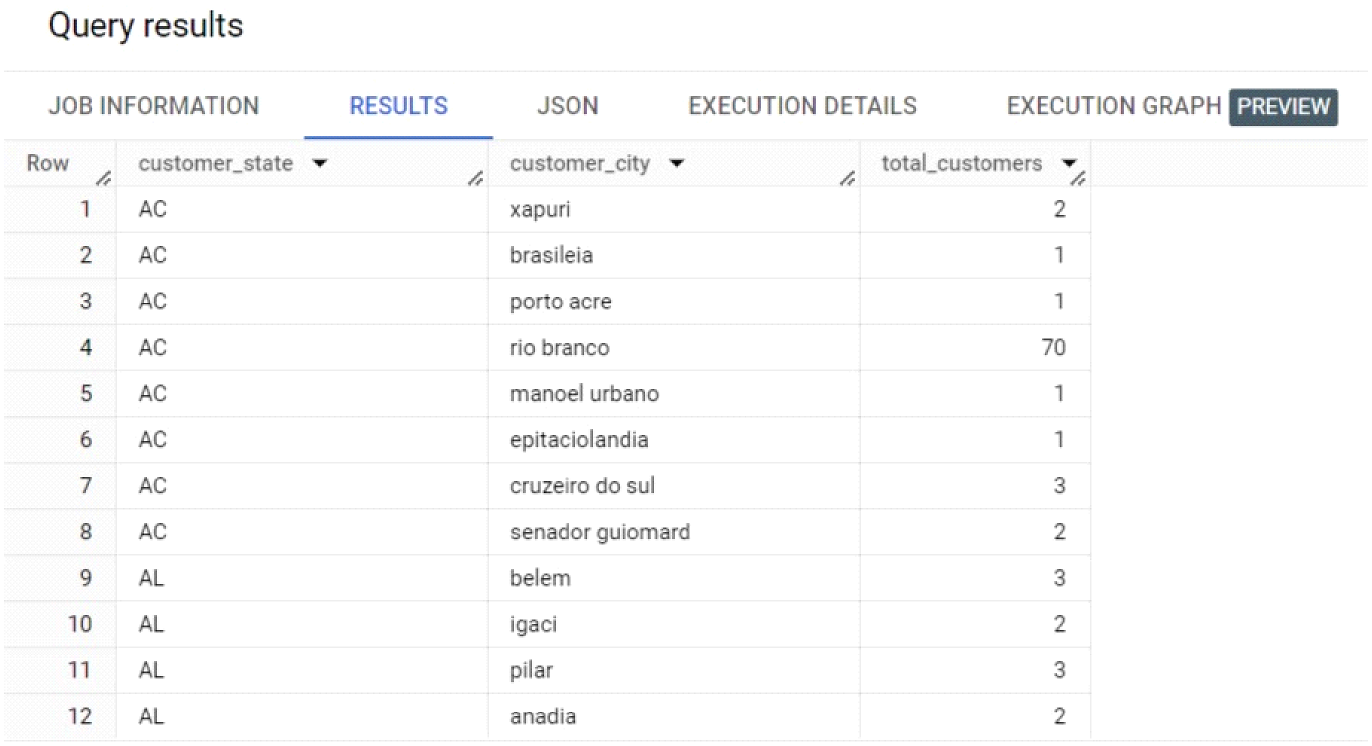


Query results

Row	customer_state	customer_city	YEAR	MONTH	orders_count	prev_orders_count
1	BA	bom jesus da lapa	2017	2	1	null
2	BA	bom jesus da lapa	2017	7	1	1
3	BA	bom jesus da lapa	2017	8	1	1
4	BA	bom jesus da lapa	2017	10	1	1
5	BA	bom jesus da lapa	2017	11	1	1
6	BA	bom jesus da lapa	2018	1	2	1
7	BA	bom jesus da lapa	2018	2	1	2
8	BA	bom jesus da lapa	2018	4	1	1
9	BA	monte gordo	2017	9	1	null
10	BA	urucuca	2018	4	1	null

2) Distribution of customers across the states in Brazil

QUERY :
```
SELECT    customer_state, customer_city,
               count(*) AS total_customers
FROM `target_sql.customers`
GROUP BY customer_state, customer_city
ORDER BY customer_state, customer_city;
```


Query results

JOB INFORMATION RESULTS JSON EXECUTION DETAILS EXECUTION GRAPH PREVIEW

Row	customer_state	customer_city	total_customers
1	AC	xapuri	2
2	AC	brasileia	1
3	AC	porto acre	1
4	AC	rio branco	70
5	AC	manoel urbano	1
6	AC	epitaciolandia	1
7	AC	cruzeiro do sul	3
8	AC	senador guiomard	2
9	AL	belem	3
10	AL	igaci	2
11	AL	pilar	3
12	AL	anadia	2

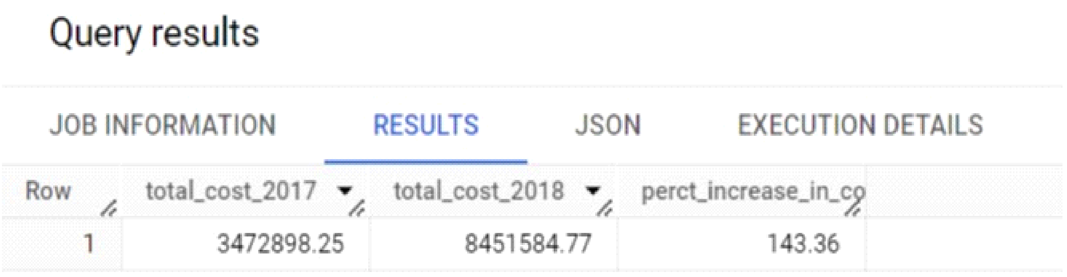
INSIGHTS - The distribution of customers over 27 states in Brazil. SP has the highest number of customers followed by Rio D and MG.

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

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

QUERY :

```
WITH CTE1 AS (
 SELECT ROUND(SUM(price + freight_value), 2) AS total_cost_2017
 FROM (SELECT *
         FROM target_sql.orders AS O JOIN target_sql.order_items AS
OI
 ON O.order_id = OI.order_id
         WHERE O.order_status = 'delivered'
         AND (EXTRACT(YEAR FROM O.order_purchase_timestamp) =
2017)
         AND EXTRACT(MONTH FROM O.order_purchase_timestamp)
BETWEEN 1 AND 8)),
CTE2 AS (
   SELECT ROUND(SUM(price + freight_value), 2) AS total_cost_2018
   FROM (
            SELECT *
            FROM target_sql.orders AS O JOIN target_sql.order_items AS
OI

   ON O.order_id = OI.order_id
            WHERE O.order_status = 'delivered'
            AND (EXTRACT(YEAR FROM O.order_purchase_timestamp) =
2018)

            AND EXTRACT(MONTH FROM O.order_purchase_timestamp)
                  BETWEEN 1 AND 8))

SELECT
    c1.total_cost_2017,
    c2.total_cost_2018,
    ROUND(((c2.total_cost_2018 - c1.total_cost_2017) / c1.total_cost_
          2017) * 100, 2) AS perct_increase_in_cost
   FROM
   CTE1 AS c1 CROSS JOIN CTE2 AS c2;
```


Query results

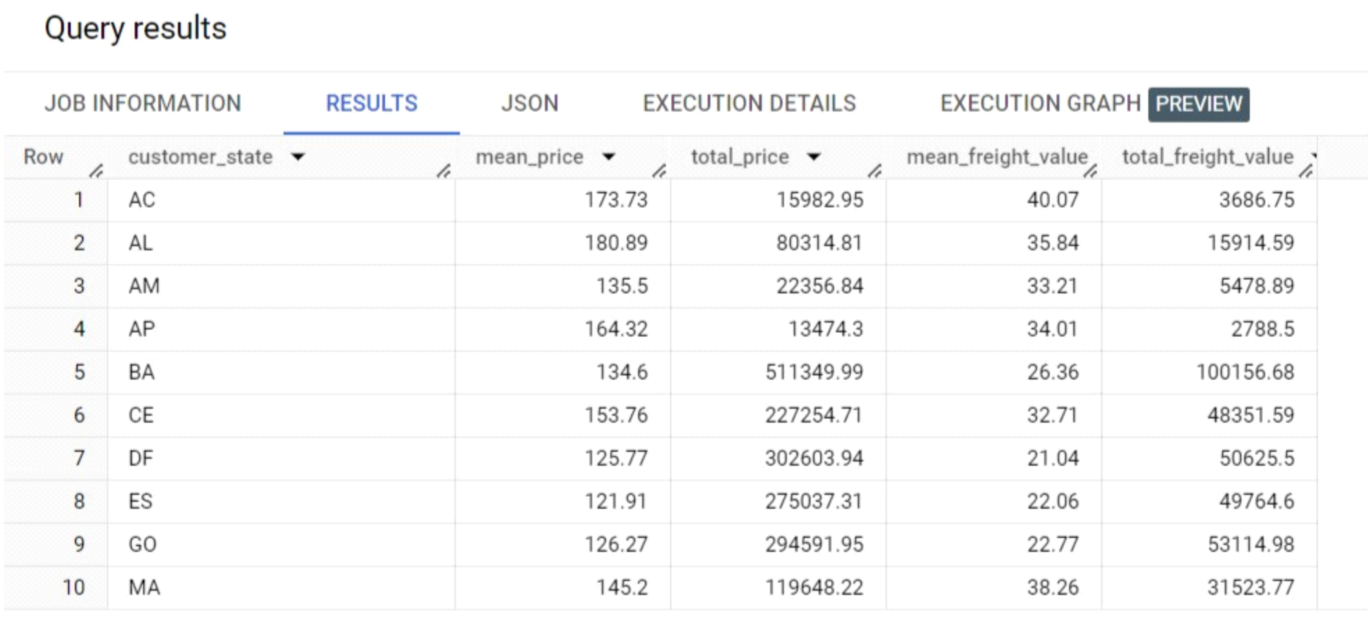
JOB INFORMATION RESULTS JSON EXECUTION DETAILS

Row	total_cost_2017	total_cost_2018	perct_increase_in_co
1	3472898.25	8451584.77	143.36

2) Mean & Sum of price and freight value by customer state

QUERY :

```
SELECT c.customer_state,
              ROUND(AVG(price),2) AS mean_price,
              ROUND(SUM(price),2) AS total_price,
              ROUND(AVG(freight_value),2) AS mean_freight_value,
                              ROUND(SUM(freight_value),2) AS total_freight_value

                    FROM `target_sql.customers` c JOIN `target_sql.orders` o
                         ON c.customer_id = o.customer_id
                         JOIN `target_sql.order_items` p
                         ON o.order_id = p.order_id

                    GROUP BY c.customer_state
                    ORDER BY c.customer_state;
```



Query results

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

Row	customer_state	mean_price	total_price	mean_freight_value	total_freight_value
1	AC	173.73	15982.95	40.07	3686.75
2	AL	180.89	80314.81	35.84	15914.59
3	AM	135.5	22356.84	33.21	5478.89
4	AP	164.32	13474.3	34.01	2788.5
5	BA	134.6	511349.99	26.36	100156.68
6	CE	153.76	227254.71	32.71	48351.59
7	DF	125.77	302603.94	21.04	50625.5
8	ES	121.91	275037.31	22.06	49764.6
9	GO	126.27	294591.95	22.77	53114.98
10	MA	145.2	119648.22	38.26	31523.77

5. Analysis on sales, freight and delivery time

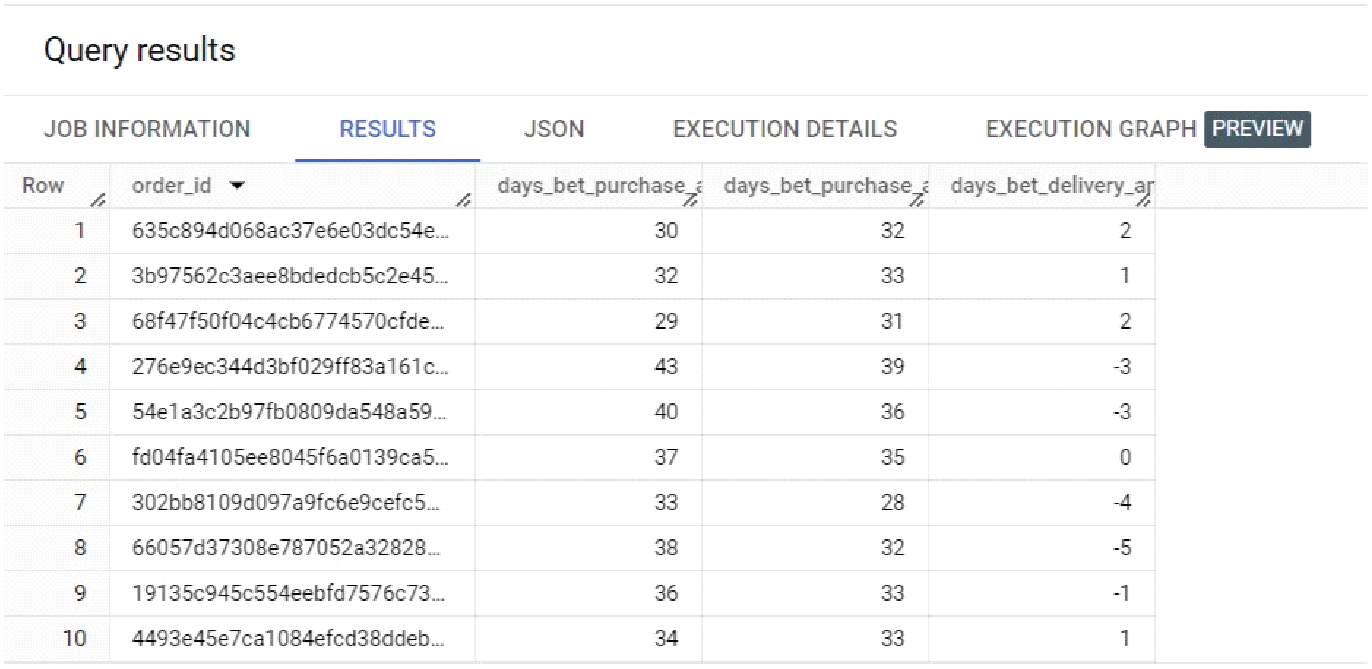
1) Calculate days between purchasing, delivering and estimated delivery

QUERY : SELECT

```
order_id,
     TIMESTAMP_DIFF(order_delivered_customer_date,
     order_purchase_timestamp,Day) AS days_bet_purchase_and_delivery,

     TIMESTAMP_DIFF(order_estimated_delivery_date,
     order_purchase_timestamp,Day) AS days_bet_purchase_and_estmitatedD,
```

TIMESTAMP_DIFF(order_estimated_delivery_date,
order_delivered_customer_date,Day) AS days_bet_delivery_and_estimatedD

FROM `target_sql.orders`
WHERE order_status = 'delivered';


Query results

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

Row	order_id	days_bet_purchase_a	days_bet_purchase_a	days_bet_delivery_ar
1	635c894d068ac37e6e03dc54e...	30	32	2
2	3b97562c3aee8bdedcb5c2e45...	32	33	1
3	68f47f50f04c4cb6774570cfde...	29	31	2
4	276e9ec344d3bf029ff83a161c...	43	39	-3
5	54e1a3c2b97fb0809da548a59...	40	36	-3
6	fd04fa4105ee8045f6a0139ca5...	37	35	0
7	302bb8109d097a9fc6e9cefc5...	33	28	-4
8	66057d37308e787052a32828...	38	32	-5
9	19135c945c554eebfd7576c73...	36	33	-1
10	4493e45e7ca1084efcd38ddeb...	34	33	1

2) Find time_to_delivery & diff_estimated_delivery. Formula for the same given below:

i) time_to_delivery = order_delivered_customer_date-order_purchase_timestamp

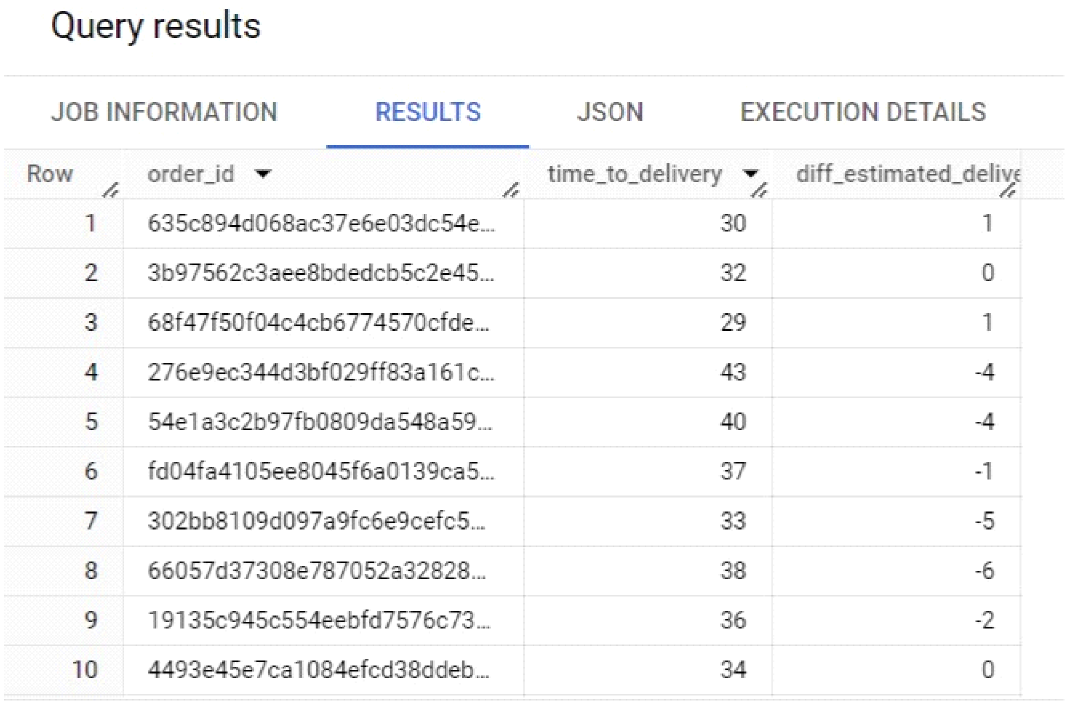
ii) diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date

QUERY :

SELECT order_id,

TIMESTAMP_DIFF(order_delivered_customer_date,order_purchase_timestamp, DAY) AS time_to_delivery,

```
TIMESTAMP_DIFF(order_estimated_delivery_date,
order_delivered_customer_date, DAY) AS diff_estimated_delivery

FROM `target_sql.orders`
WHERE order_status = 'delivered';
```

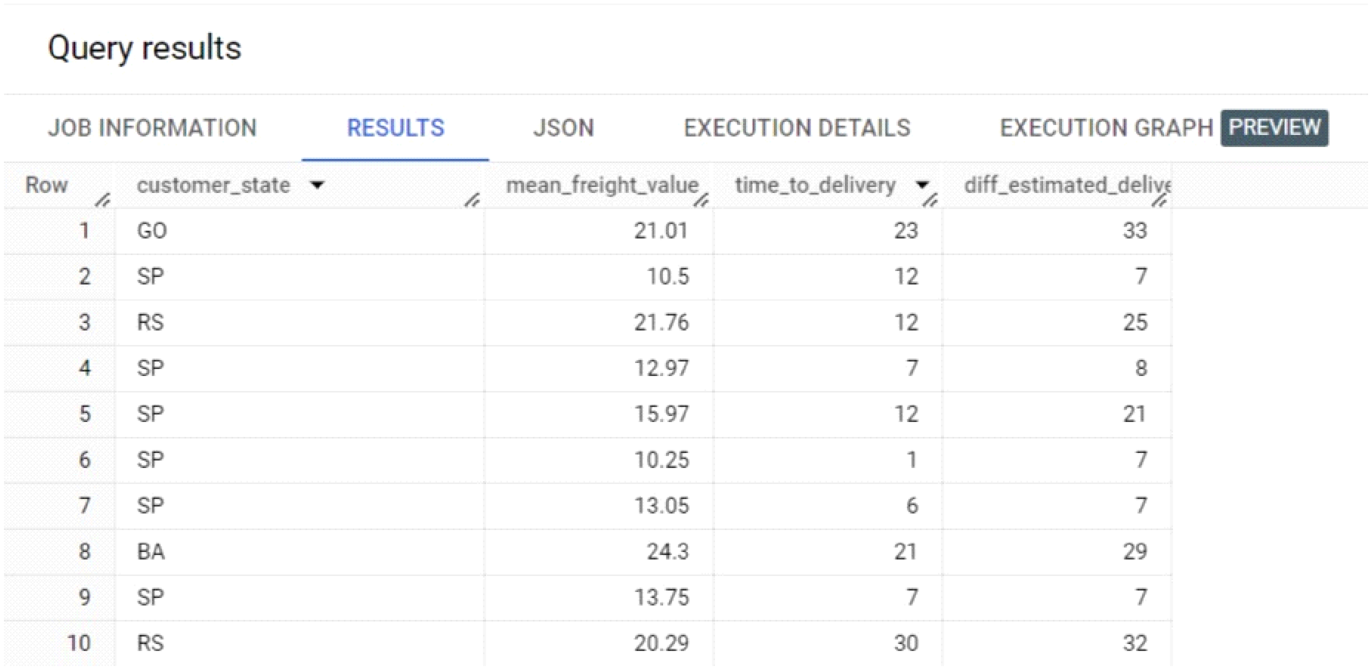

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	order_id	time_to_delivery	diff_estimated_delive
1	635c894d068ac37e6e03dc54e...	30	1
2	3b97562c3aee8bdedcb5c2e45...	32	0
3	68f47f50f04c4cb6774570cfde...	29	1
4	276e9ec344d3bf029ff83a161c...	43	-4
5	54e1a3c2b97fb0809da548a59...	40	-4
6	fd04fa4105ee8045f6a0139ca5...	37	-1
7	302bb8109d097a9fc6e9cefc5...	33	-5
8	66057d37308e787052a32828...	38	-6
9	19135c945c554eebfd7576c73...	36	-2
10	4493e45e7ca1084efcd38ddeb...	34	0

3) Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery

QUERY :
```
SELECT   c.customer_state,
                ROUND(AVG(p.freight_value),2) AS mean_freight_value,
                x.time_to_delivery,
                x.diff_estimated_delivery
FROM `target_sql.customers` AS c
JOIN
(SELECT *,
       TIMESTAMP_DIFF(order_delivered_customer_date,
       order_purchase_timestamp, DAY) AS time_to_delivery,

       TIMESTAMP_DIFF(order_estimated_delivery_date,
       order_purchase_timestamp, DAY) AS diff_estimated_delivery,
                    FROM `target_sql.orders`
                    WHERE order_status = 'delivered')       AS    x

                    ON c.customer_id = x.customer_id
                    JOIN
                    `target_sql.order_items` AS    p
                    x.order_id = p.order_id

            GROUP BY c.customer_state,    x.time_to_delivery,
x.diff_estimated_delivery ;
```


Query results

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

Row	customer_state	mean_freight_value	time_to_delivery	diff_estimated_delive
1	GO	21.01	23	33
2	SP	10.5	12	7
3	RS	21.76	12	25
4	SP	12.97	7	8
5	SP	15.97	12	21
6	SP	10.25	1	7
7	SP	13.05	6	7
8	BA	24.3	21	29
9	SP	13.75	7	7
10	RS	20.29	30	32

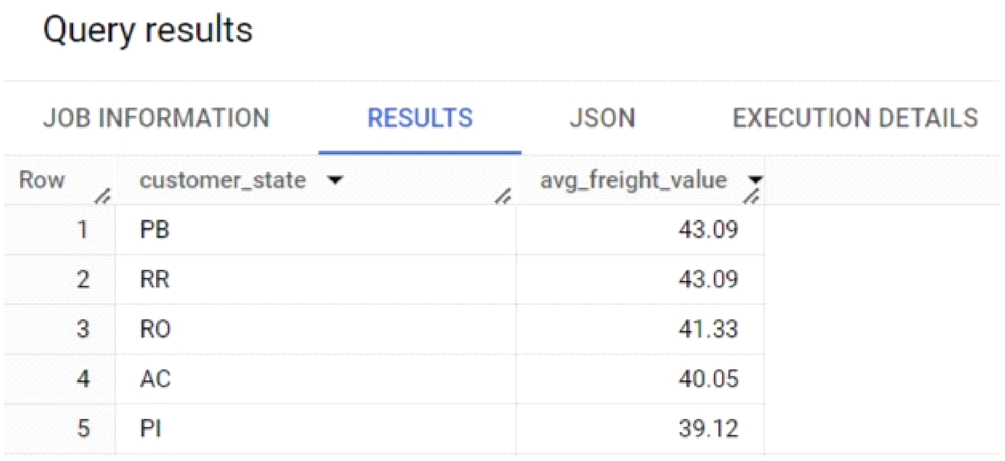
INSIGHTS - The customers of the northern states have to wait longer to receive the shipment as compared to the customers in the southern states.

4) Sort the data to get the following:

i) Top 5 states with highest/lowest average freight value - sort in desc/asc limit 5

QUERY :

```
          SELECT      c.customer_state,
                            ROUND(AVG(p.freight_value),2) AS avg_freight_value,
                FROM `target_sql.customers` AS c

                JOIN

                      (SELECT *, FROM `target_sql.orders`
                 WHERE order_status = 'delivered') AS     x

                 ON c.customer_id = x.customer_id

                JOIN
               `target_sql.order_items` AS p
                ON x.order_id = p.order_id

                GROUP BY c.customer_state
                      ORDER BY AVG(p.freight_value) DESC
                      LIMIT 5;
```



Query results

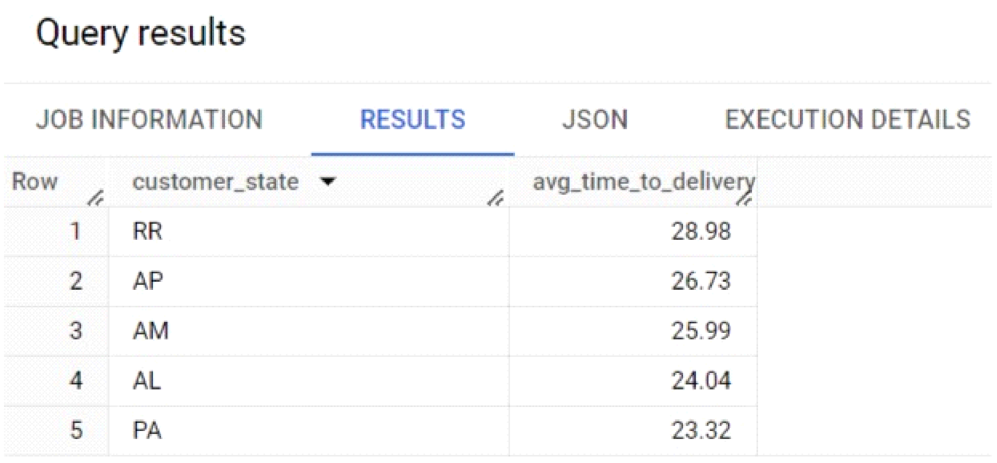
JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	avg_freight_value
1	PB	43.09
2	RR	43.09
3	RO	41.33
4	AC	40.05
5	PI	39.12

ii) Top 5 states with highest/lowest average time to delivery

QUERY : SELECT c.customer_state,

```
                              ROUND(AVG(x.time_to_delivery), 2) AS
avg_time_to_delivery,
                      FROM `target_sql.customers` AS    c
              JOIN
                 (SELECT *,
                      TIMESTAMP_DIFF(order_delivered_customer_date,
                            order_purchase_timestamp, DAY) AS
time_to_delivery,
                                             FROM `target_sql.orders`
                          WHERE order_status = 'delivered') AS    x

                                        ON c.customer_id = x.customer_id

                                        GROUP BY c.customer_state
                                ORDER BY AVG(x.time_to_delivery) DESC
                        LIMIT 5;
```



Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

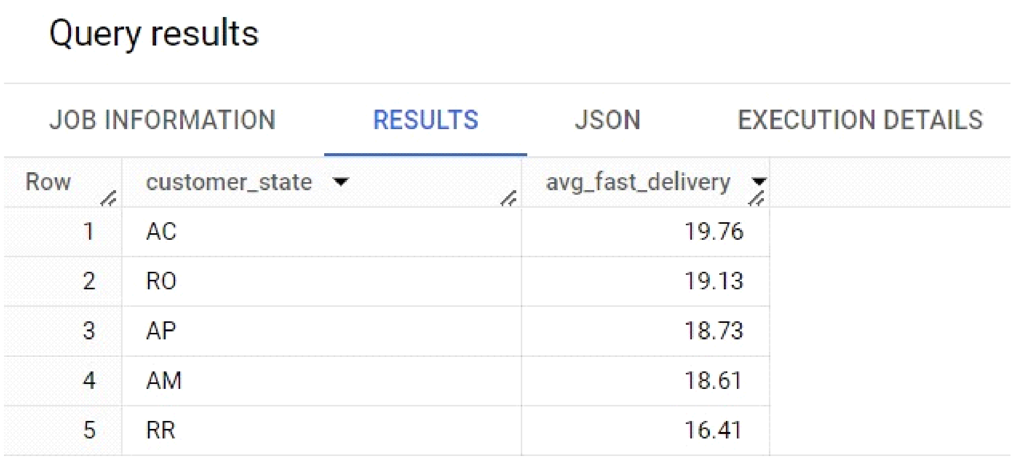
Row	customer_state	avg_time_to_delivery
1	RR	28.98
2	AP	26.73
3	AM	25.99
4	AL	24.04
5	PA	23.32

iii) Top 5 states where delivery is really fast/ not so fast compared to estimated date

QUERY :

```
SELECT      c.customer_state,
            ROUND(AVG(x.diff_estimate_del ),2) AS avg_fast_delivery
            FROM `target_sql.customers` AS c
            JOIN
            (SELECT *,
            TIMESTAMP_DIFF(order_estimated_delivery_date,
                    order_delivered_customer_date, DAY) AS diff_estimate_del
             FROM `target_sql.orders`
             WHERE order_status = 'delivered')    AS    x

            ON c.customer_id = x.customer_id

      GROUP BY c..customer_state
```

```
ORDER BY AVG(x.diff_estimate_del) DESC
LIMIT 5;
```



Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	avg_fast_delivery
1	AC	19.76
2	RO	19.13
3	AP	18.73
4	AM	18.61
5	RR	16.41

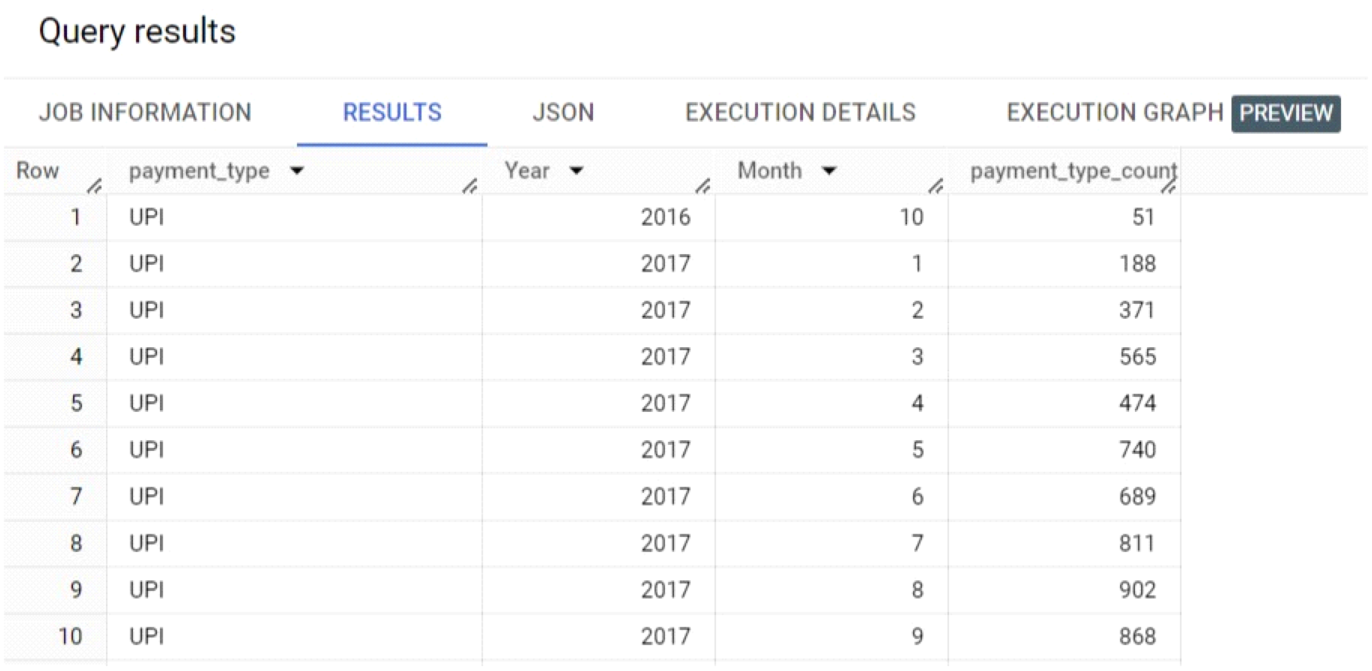
INSIGHTS - From the above the average days to deliver an item is 12 days and the orders are delivered at an average of 10 days before the estimated delivery date .

6. Payment type analysis:

1) Month over Month count of orders for different payment types

QUERY :

```
SELECT DISTINCT p.payment_type, o.year, o.month,
        COUNT(*) OVER (PARTITION BY p.payment_type, o.year,
            o.month ORDER BY o.year, o.month) AS
payment_type_count

FROM `target_sql.payments` AS p
    JOIN
    (SELECT order_id,
            EXTRACT(MONTH FROM
order_purchase_timestamp) AS month,
            EXTRACT(YEAR FROM
order_purchase_timestamp) AS year
    FROM `target_sql.orders`
            WHERE order_status = 'delivered') AS o
    ON p.order_id = o.order_id;
```



Query results

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

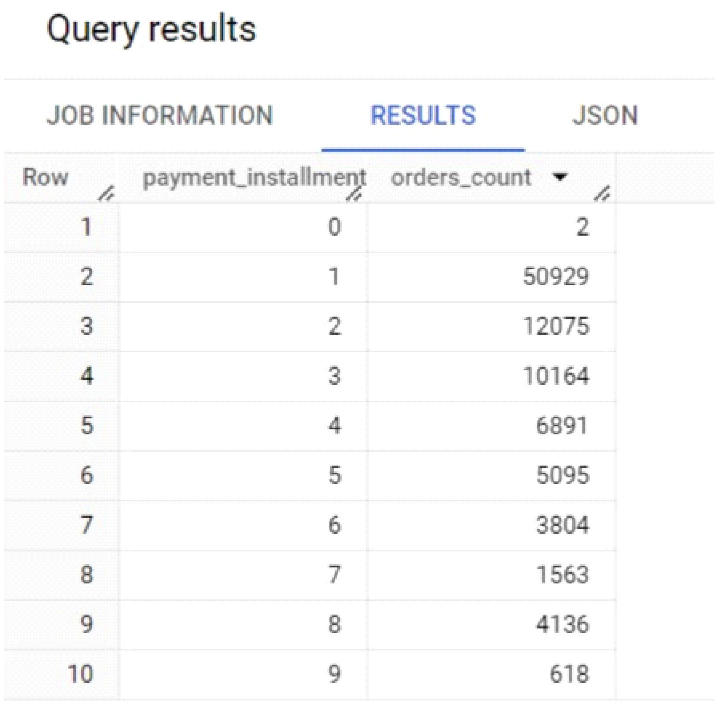
Row	payment_type	Year	Month	payment_type_count
1	UPI	2016	10	51
2	UPI	2017	1	188
3	UPI	2017	2	371
4	UPI	2017	3	565
5	UPI	2017	4	474
6	UPI	2017	5	740
7	UPI	2017	6	689
8	UPI	2017	7	811
9	UPI	2017	8	902
10	UPI	2017	9	868

INSIGHTS - It can be seen that the credit card is the most preferred payment type by customers followed by UPI and voucher.

2) Count of orders based on the no. of payment installment

QUERY :

```
SELECT    p.payment_installments,
                 COUNT(*) AS   orders_count
  FROM `target_sql.payments` AS p
  JOIN
  (SELECT *
     FROM `target_sql.orders`
     WHERE order_status = 'delivered')   AS   o

 ON p.order_id = o.order_id

 GROUP BY p.payment_installments
 ORDER BY p.payment_installments;
```


Query results

JOB INFORMATION RESULTS JSON

Row	payment_installment	orders_count
1	0	2
2	1	50929
3	2	12075
4	3	10164
5	4	6891
6	5	5095
7	6	3804
8	7	1563
9	8	4136
10	9	618

INSIGHTS- It can be seen that payment of a significant number of orders is made in small installments.

8. Recommendations :

1) The maximum orders are seen at night and afternoon (77% of total orders), target has to make sure that online portal runs smoothly during these times. So their online shopping experience will increase and also orders.

2) Monthly orders growth rate can be as like (Nov 2017) during the peak seasons. For that focus shuold be on inventory and stocks so can meet high demands.

3) If there is incrase in discount pricing strategies before the peak seasons so we can aquire new customers from the regions where the customer count is less. So it increase profit also.

4) For the new customer base in the region where low customer counts we can incrase discount pricing and also for old customer we can reduce average freight cost and average time to delivery. So it can aquire new customer base and also hold old customers by giving good experence.

5) Along with this we can build good social omnipresence by using all social media platfroms so we can attract new potential custmoers moslty in the region where customer count is very less.