3/20/2023

PROJECT 1 - Business Case: Target SQL

Shailendra Pratap Singh
SCALER ACADAMY

Table of Contents

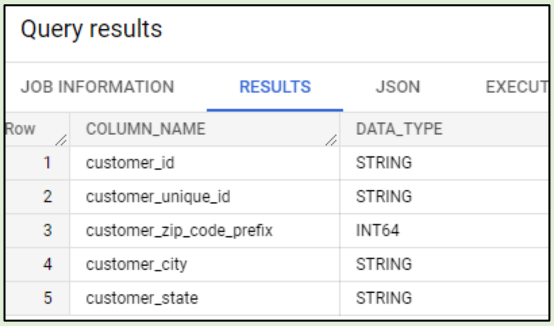
1 Usual exploratory analysis

1.1 Data type of columns in a table

- **Approach:**
  - Exploration of Data and Data type using 'INFORMATION.SCHEMA'
  - As per the question, extracted the data type for all the table's columns.

✓ **Table Name:** customers

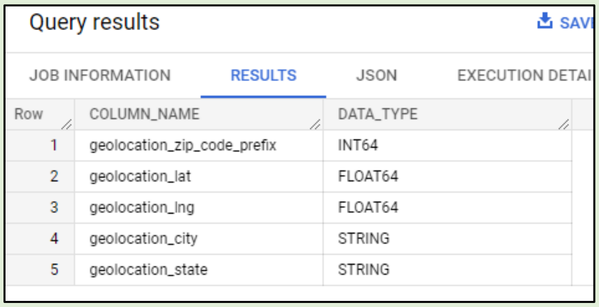
```
SELECT
  COLUMN_NAME, DATA_TYPE
FROM
  `target`.INFORMATION_SCHEMA.COLUMNS
WHERE
  table_name = 'customers';
```



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

✓ **Table Name:** geolocation

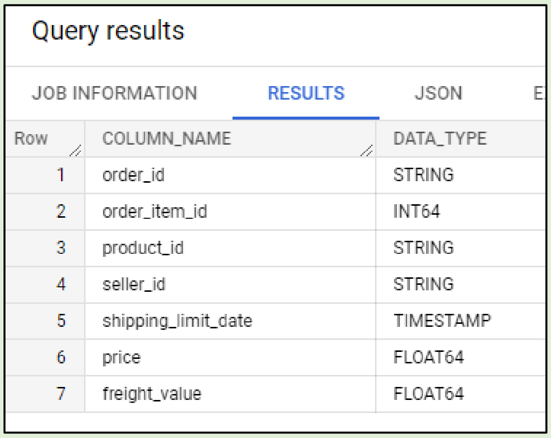
```
SELECT
  COLUMN_NAME, DATA_TYPE
FROM
  `target`.INFORMATION_SCHEMA.COLUMNS
WHERE
  table_name = 'geolocation';
```



Row	COLUMN_NAME	DATA_TYPE
1	geolocation_zip_code_prefix	INT64
2	geolocation_lat	FLOAT64
3	geolocation_lng	FLOAT64
4	geolocation_city	STRING
5	geolocation_state	STRING

✓ **Table Name:** order_items

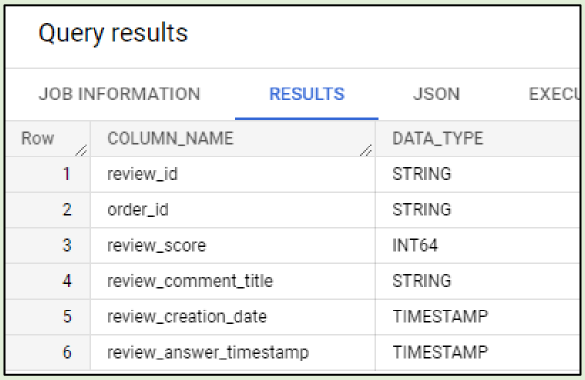
```
SELECT
  COLUMN_NAME, DATA_TYPE
FROM
  `target`.INFORMATION_SCHEMA.COLUMNS
WHERE
  table_name = 'order_items';
```



Row	COLUMN_NAME	DATA_TYPE
1	order_id	STRING
2	order_item_id	INT64
3	product_id	STRING
4	seller_id	STRING
5	shipping_limit_date	TIMESTAMP
6	price	FLOAT64
7	freight_value	FLOAT64

✓ **Table Name:** order_reviews

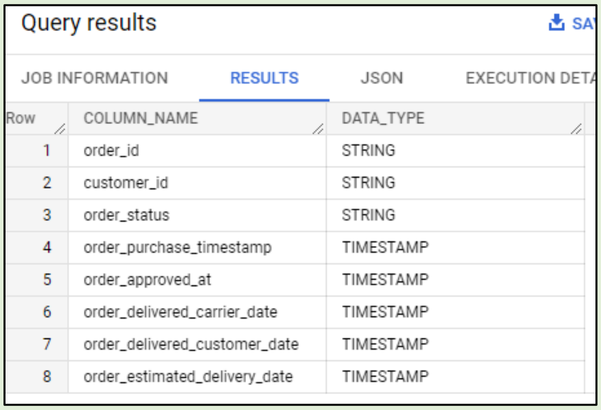
```
SELECT
  COLUMN_NAME, DATA_TYPE
FROM
  `target`.INFORMATION_SCHEMA.COLUMNS
WHERE
  table_name = ' order_reviews';
```



Row	COLUMN_NAME	DATA_TYPE
1	review_id	STRING
2	order_id	STRING
3	review_score	INT64
4	review_comment_title	STRING
5	review_creation_date	TIMESTAMP
6	review_answer_timestamp	TIMESTAMP

✓ **Table Name:** orders

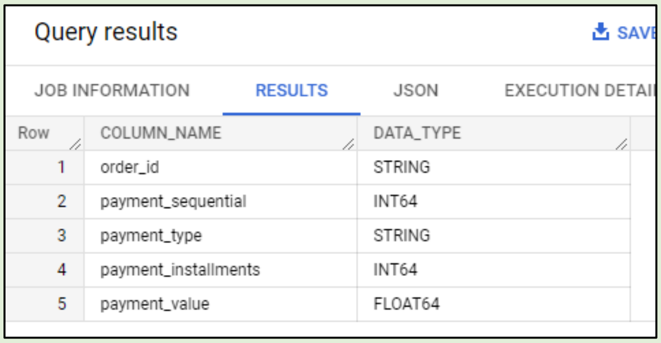
```
SELECT
  COLUMN_NAME, DATA_TYPE
FROM
  `target`.INFORMATION_SCHEMA.COLUMNS
WHERE
  table_name = ' orders';
```



Row	COLUMN_NAME	DATA_TYPE
1	order_id	STRING
2	customer_id	STRING
3	order_status	STRING
4	order_purchase_timestamp	TIMESTAMP
5	order_approved_at	TIMESTAMP
6	order_delivered_carrier_date	TIMESTAMP
7	order_delivered_customer_date	TIMESTAMP
8	order_estimated_delivery_date	TIMESTAMP

✓ **Table Name:** payments

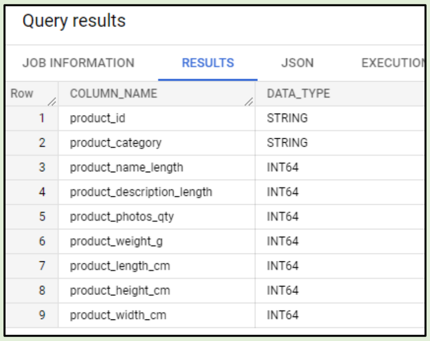
```
SELECT
  COLUMN_NAME, DATA_TYPE
FROM
  `target`.INFORMATION_SCHEMA.COLUMNS
WHERE
  table_name = 'payments’;
```



Row	COLUMN_NAME	DATA_TYPE
1	order_id	STRING
2	payment_sequential	INT64
3	payment_type	STRING
4	payment_installments	INT64
5	payment_value	FLOAT64

✓ **Table Name:** products

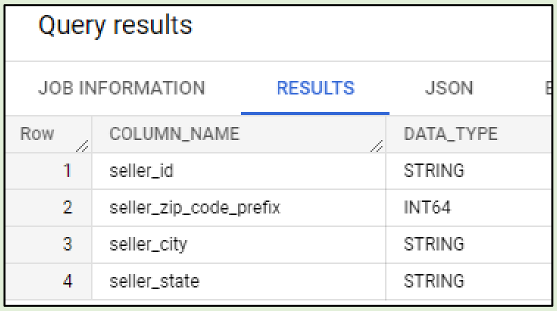
```
SELECT
  COLUMN_NAME, DATA_TYPE
FROM
  `target`.INFORMATION_SCHEMA.COLUMNS
WHERE
  table_name = 'products’;
```



Row	COLUMN_NAME	DATA_TYPE
1	product_id	STRING
2	product_category	STRING
3	product_name_length	INT64
4	product_description_length	INT64
5	product_photos_qty	INT64
6	product_weight_g	INT64
7	product_length_cm	INT64
8	product_height_cm	INT64
9	product_width_cm	INT64

✓ **Table Name:** sellers

```
SELECT
  COLUMN_NAME,
  DATA_TYPE
FROM
  `target`.INFORMATION_SCHEMA.COLUMNS
WHERE
  table_name = 'sellers’;
```



Row	COLUMN_NAME	DATA_TYPE
1	seller_id	STRING
2	seller_zip_code_prefix	INT64
3	seller_city	STRING
4	seller_state	STRING

✓ **Observations**

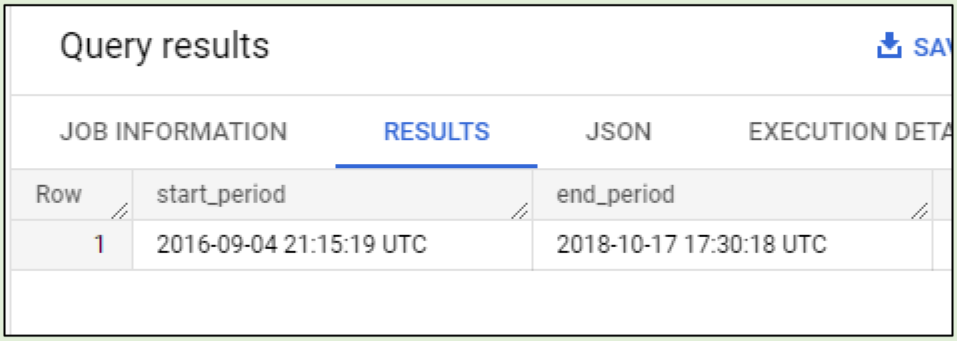
- It Looks data is in a correct format
- There are null values present and missing data in some columns that need to be taken care

1.2 Time period for which the data is given

Approach:

Minimum date of purchase from the orders table as a Starting Period, and the Maximum date of purchase as Ending Period.

```
SELECT
  MIN(order_purchase_timestamp) start_period,
  MAX(order_purchase_timestamp) end_period
FROM
  `target.orders`;
```



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

✓ <u>Observations</u>

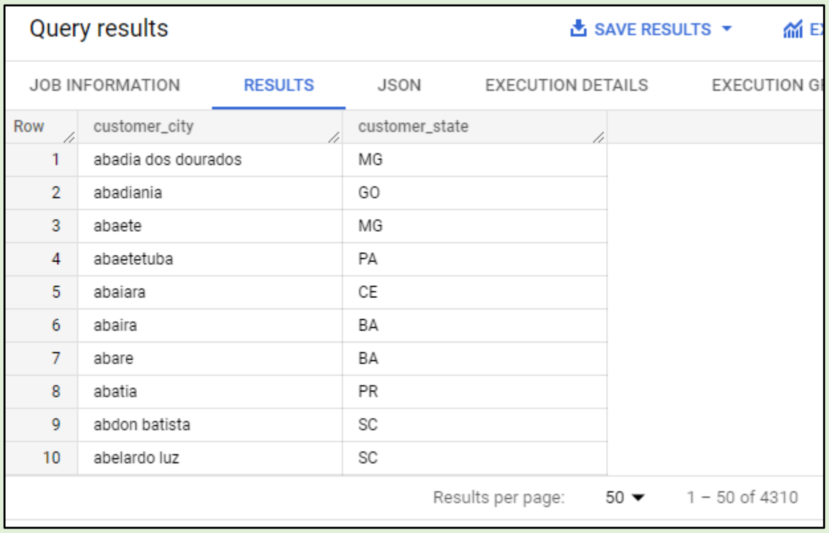
- The data shows we have around two years of data starting from Sep’16 to Oct’18.
- For 2016 and 2018, we don’t have full year’s data.

1.3 Cities and States of customers ordered during the given period

Approach:

The ‘orders’ table has the details of customers who ordered during the period. As the ‘customers’ table contains the details of the states and cities of customers, I have joined ‘customers’ with that to get the cities and states of the same customers.

```
SELECT
  DISTINCT c.customer_city,
  c.customer_state
FROM
  `target.orders` o
JOIN
  `target.customers` c
ON
  o.customer_id = c.customer_id
ORDER BY
  1;
```



Row	customer_city	customer_state
1	abadia dos dourados	MG
2	abadiania	GO
3	abaete	MG
4	abaetetuba	PA
5	abaiara	CE
6	abaira	BA
7	abare	BA
8	abatia	PR
9	abdon batista	SC
10	abelardo luz	SC

Results per page: 50   1 – 50 of 4310

✓ Observations

- There are 4,310 cities in 27 States from where the customers have placed an order during the given period.

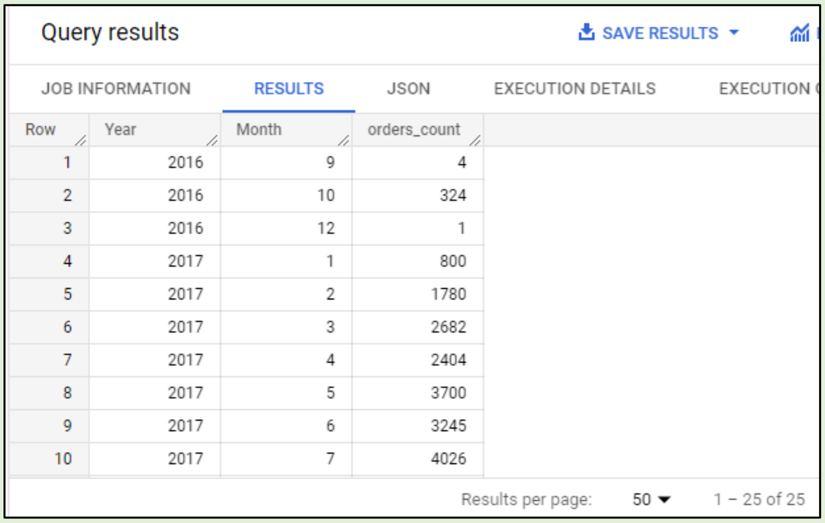
2 In-depth Exploration

2.1 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?

Approach:

1. To analyze the trend, I have calculated the total count of orders monthly.
2. First, I have extracted the year and months from the orders table.
3. For each extracted period, I have aggregated the count of orders for that period
4. Finally, I have arranged it like a time-series data to see the trend and seasonality.

```sql
SELECT
    EXTRACT(YEAR FROM order_purchase_timestamp) AS Year,
    EXTRACT(MONTH FROM order_purchase_timestamp) AS Month,
    COUNT(order_id) AS orders_count
FROM
    `target.orders`
GROUP BY 1 , 2
ORDER BY 1 , 2;
```

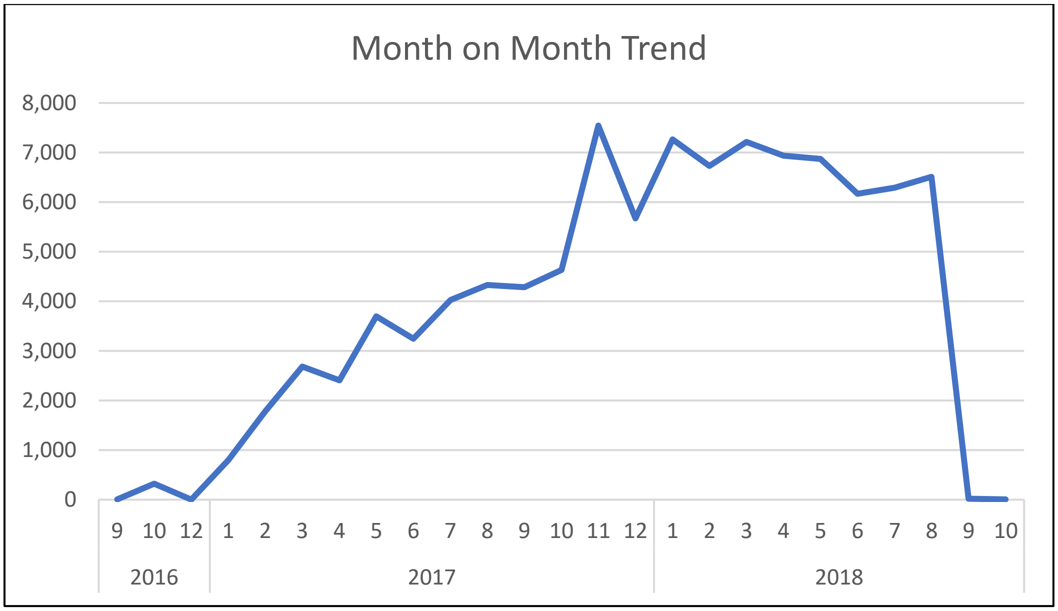


Row	Year	Month	orders_count
1	2016	9	4
2	2016	10	324
3	2016	12	1
4	2017	1	800
5	2017	2	1780
6	2017	3	2682
7	2017	4	2404
8	2017	5	3700
9	2017	6	3245
10	2017	7	4026

Results per page: 50 ▾ 1 – 25 of 25

Analysis:

1. The data shows a trend during the 2016-17, which flattened during 2018.
2. We have a seasonality present (peaks available) during November, but it can't be concluded due to insufficient data.
3. Overall Scenario is not going as it should be. The orders are dropping, which needs to be analysed and resolved.

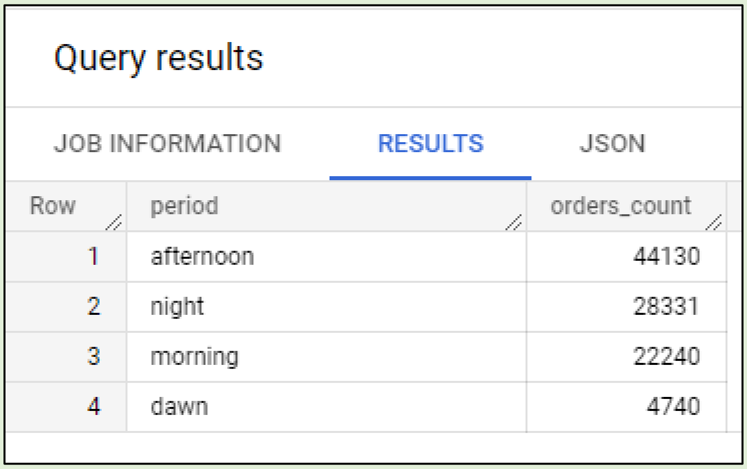


2.2 What time do Brazilian customers tend to buy (Dawn, Morning, Afternoon or Night)?

Approach

Created cases on purchase_time and counted total orders during those time frame, considering Dawa as 12:00 AM to 06:00 AM, Morning as 06:00 AM to 12:00 PM, Afternoon as 12:00 PM to 06:00 PM, and Night as 06:00 PM to 12:00 AM.

```
SELECT
    CASE
        WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 0 AND 5 THEN 'dawn'
        WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 6 AND 11 THEN 'morning'
        WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 12 AND 18 THEN 'afternoon'
        ELSE 'night'
    END AS period,
    COUNT(order_id) AS orders_count
FROM
    `target.orders`
GROUP BY 1
ORDER BY 2 DESC;
```



Query results

Row	period	orders_count
1	afternoon	44130
2	night	28331
3	morning	22240
4	dawn	4740

Analysis:

1. Majority of the orders placed during Afternoon (12 PM- 6 PM) and Night (6 PM – 12 AM)

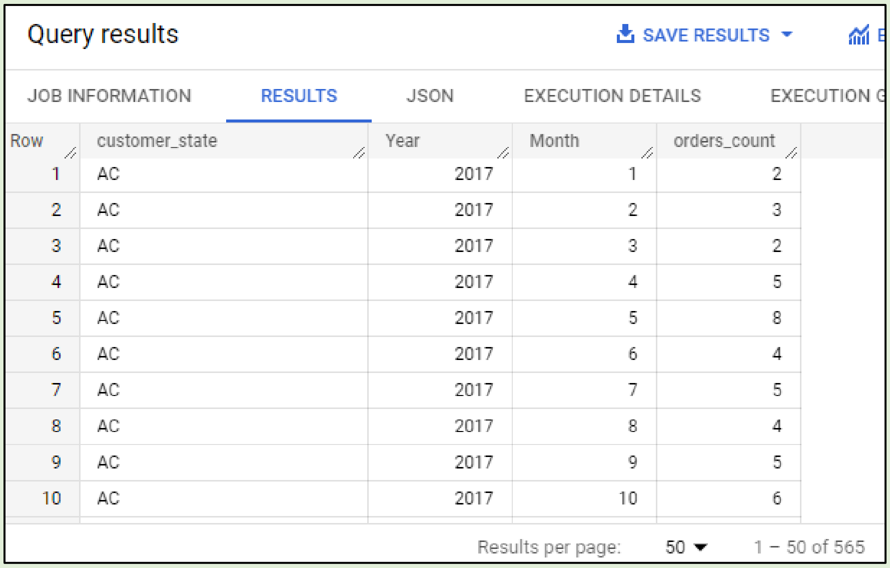
3 Evolution of E-commerce orders in the Brazil region:

3.1 Get month on month orders by states

Approach

Extracted Year and month and Aggregared (Count) the Order_id and grouped by Year and Month to get the Month on Month orders data for different states

```sql
SELECT
    EXTRACT(YEAR FROM order_purchase_timestamp) AS Year,
    EXTRACT(MONTH FROM order_purchase_timestamp) AS Month,
    COUNT(order_id) AS orders_count
FROM
    `target.orders`
GROUP BY 1 , 2
ORDER BY 1 , 2;
```



Row	customer_state	Year	Month	orders_count
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

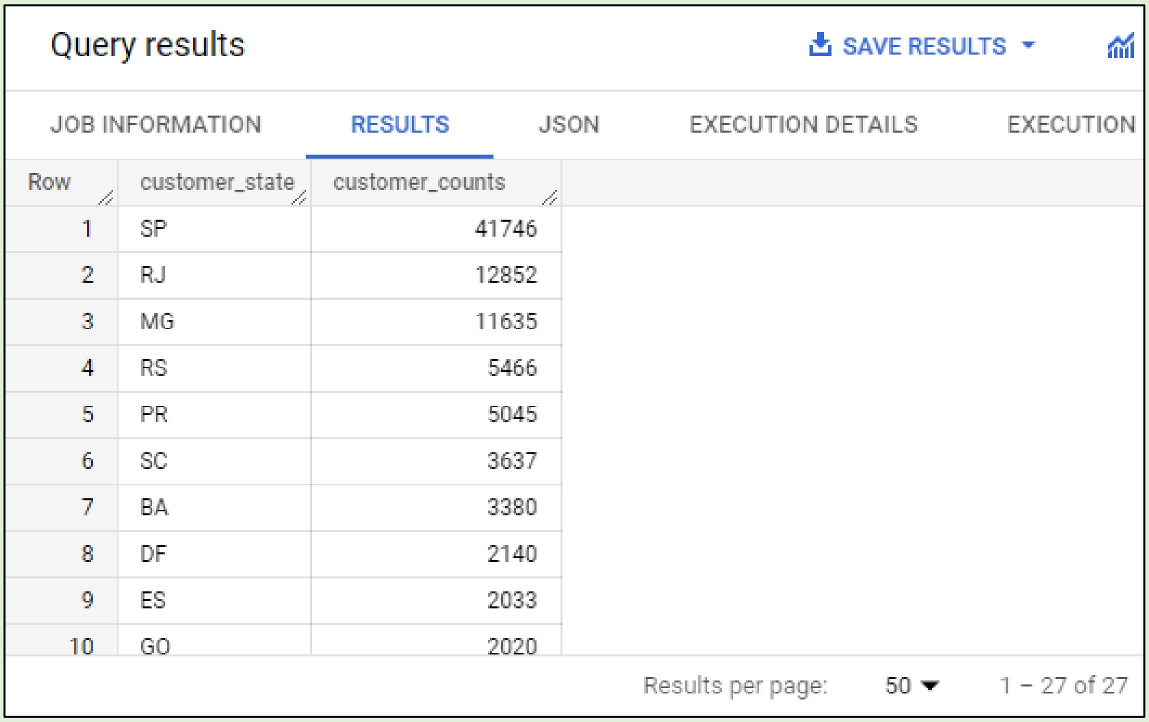
Results per page: 50 1 – 50 of 565

3.2 Distribution of customers across the states in Brazil

Approach:

To get the distribution, count aggregated customer_id and grouped by state.

```
SELECT
    customer_state, COUNT(customer_id) as customer_counts
FROM
    `target.customers`
GROUP BY 1
ORDER BY 2 DESC;
```



Row	customer_state	customer_counts
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

✓ **Observations**

- The distribution of customer is fairly skewed.
- SP is the state with highest customer’s base and it has more than 3x customer base to the next highest (RJ)
- Top 3 states have more than 65% of customer’s base. This shows the scope of expansion

4 Impact on Economy

- Analyze the money movement by e-commerce by looking at order prices, freight and others.

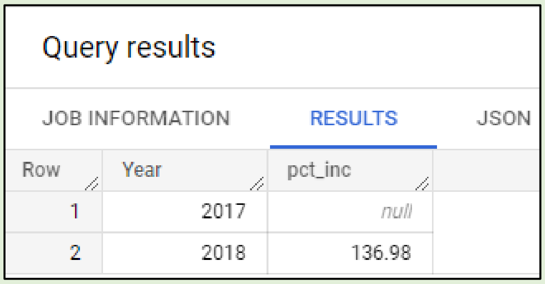
4.1 Get % increase in cost of orders from 2017 to 2018.

- (include months between Jan to Aug only) - You can use “payment_value” column in payments table

Approach:

Firstly, Using CTE, get the table of MoM payment value, then filter it for month 1 to 8 (Y2016 excluded). Finaly calculated % increase using Lag window function.

```
With CTE as
(SELECT o.order_id, p.payment_value,
    EXTRACT(YEAR FROM o.order_purchase_timestamp) AS Year,
    EXTRACT(MONTH FROM o.order_purchase_timestamp) AS Month
from `target.payments` p
JOIN `target.orders` o
ON p.order_id = o.order_id
order by 3,4)
SELECT Year, ROUND((SUM(payment_value)/ LAG(SUM(payment_value),1) over(order by Year)
  -1)*100, 2) as pct_inc
FROM CTE
WHERE Month BETWEEN 1 AND 8
GROUP BY 1
ORDER BY 1;
```



Row	Year	pct_inc
1	2017	*null*
2	2018	136.98

Analysis:

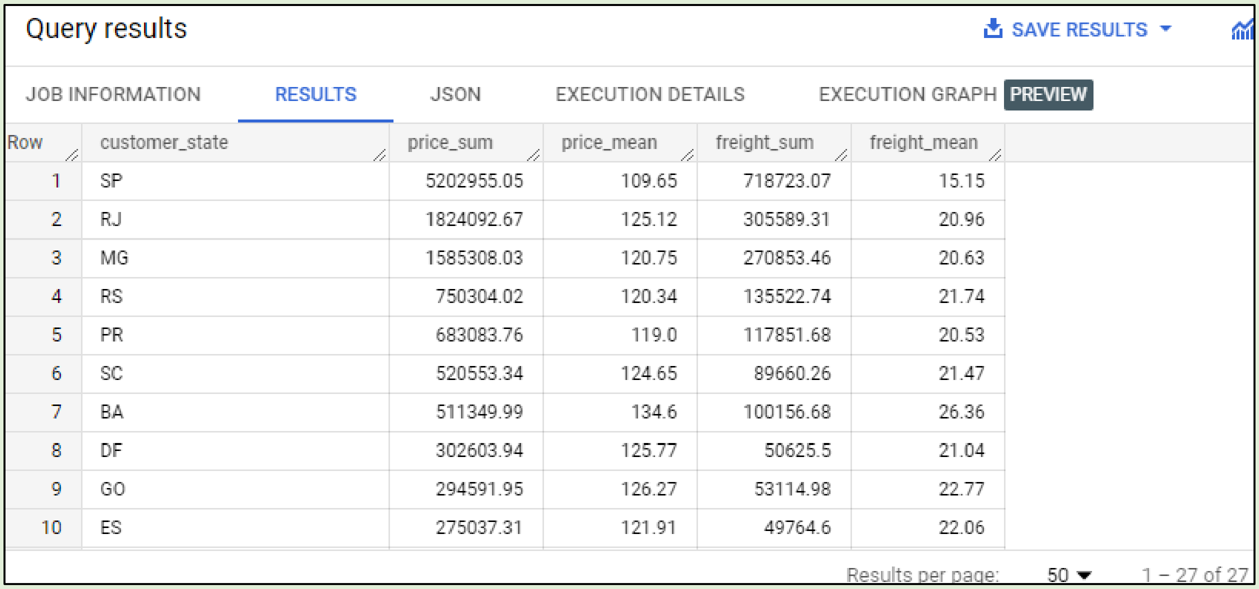
1. The order cost has seen a huge increase of 137% YoY during the first 8 months. This shows considerable growth.

4.2 Mean & Sum of price and freight value by a customer state

Approach:

Joined the customers, orders, and order_items table and get the required aggregations.

```
SELECT
    c.customer_state,
    ROUND(SUM(oi.price), 2) price_sum,
    ROUND(AVG(oi.price), 2) price_mean,
    ROUND(SUM(oi.freight_value), 2) freight_sum,
    ROUND(AVG(oi.freight_value), 2) freight_mean
FROM
    `target.customers` c
        JOIN
    `target.orders` o ON c.customer_id = o.customer_id
        JOIN
    `target.order_items` oi ON o.order_id = oi.order_id
GROUP BY 1
ORDER BY 2 DESC;
```



Row	customer_state	price_sum	price_mean	freight_sum	freight_mean
1	SP	5202955.05	109.65	718723.07	15.15
2	RJ	1824092.67	125.12	305589.31	20.96
3	MG	1585308.03	120.75	270853.46	20.63
4	RS	750304.02	120.34	135522.74	21.74
5	PR	683083.76	119.0	117851.68	20.53
6	SC	520553.34	124.65	89660.26	21.47
7	BA	511349.99	134.6	100156.68	26.36
8	DF	302603.94	125.77	50625.5	21.04
9	GO	294591.95	126.27	53114.98	22.77
10	ES	275037.31	121.91	49764.6	22.06

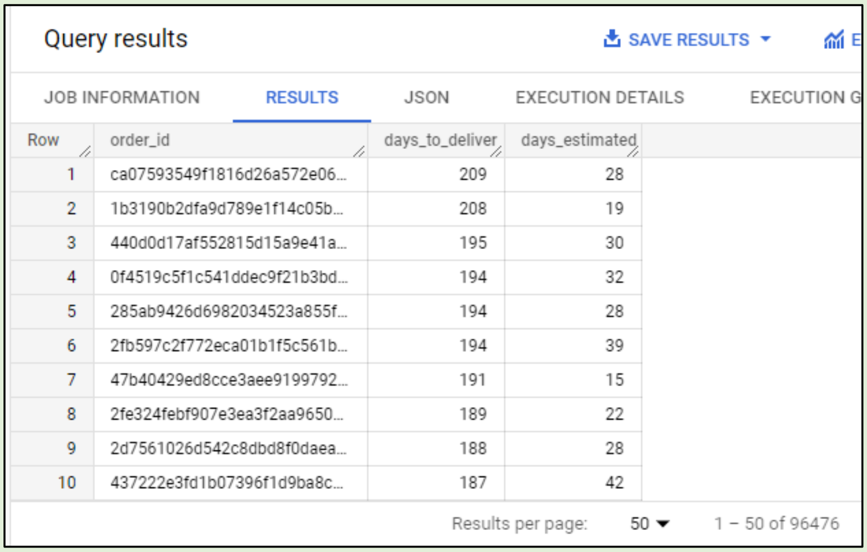
5 Analysis on sales, freight, and delivery time

5.1 Calculate days between purchasing, delivering, and estimated delivery

Approach:

- By analysing the data, there are 2,965 transactions in the orders table where order_delivery_customer_date is NULL. We need to eliminate these transactions for our calculations to avoid misrepresentation.
- There are 8 transactions where order status is delivered but order_delivery_customer_date is NULL shows missing data
-

```
SELECT
    order_id,
    DATE_DIFF(order_delivered_customer_date,
            order_purchase_timestamp,
            day) AS days_to_deliver,
    DATE_DIFF(order_estimated_delivery_date,
            order_purchase_timestamp,
            day) AS days_estimated
FROM
    `target.orders`
WHERE
    order_delivered_customer_date IS NOT NULL
ORDER BY 2 DESC;
```



Row	order_id	days_to_deliver	days_estimated
1	ca07593549f1816d26a572e06...	209	28
2	1b3190b2dfa9d789e1f14c05b...	208	19
3	440d0d17af552815d15a9e41a...	195	30
4	0f4519c5f1c541ddec9f21b3bd...	194	32
5	285ab9426d6982034523a855f...	194	28
6	2fb597c2f772eca01b1f5c561b...	194	39
7	47b40429ed8cce3aee9199792...	191	15
8	2fe324febf907e3ea3f2aa9650...	189	22
9	2d7561026d542c8dbd8f0daea...	188	28
10	437222e3fd1b07396f1d9ba8c...	187	42

Results per page: 50 1 – 50 of 96476

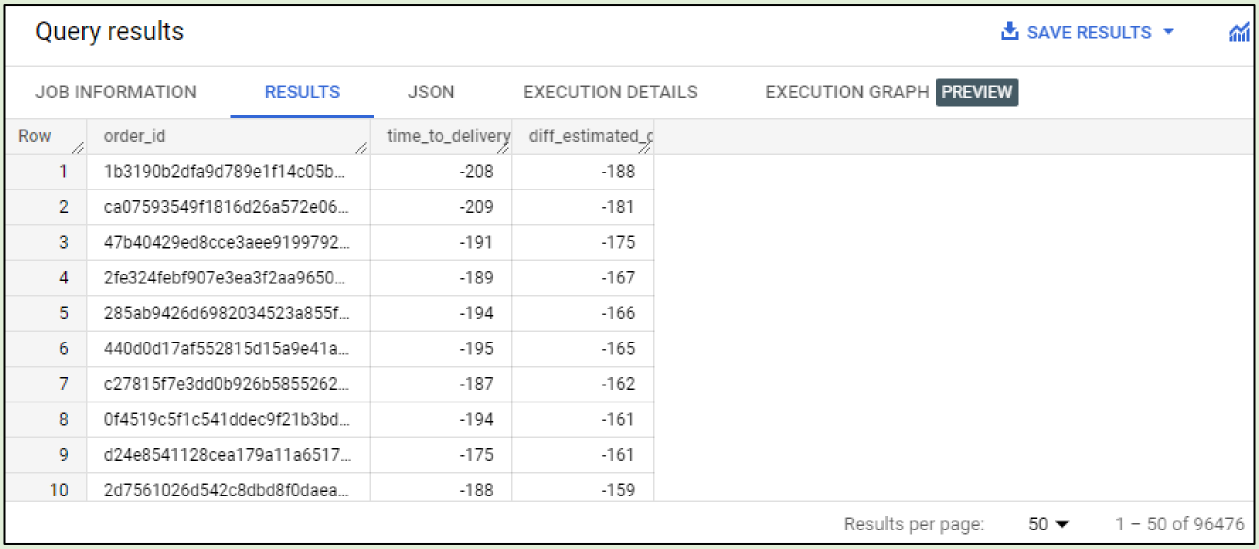
5.2 Find time_to_delivery & diff_estimated_delivery.

Formula for the same given below:

time_to_delivery = order_purchase_timestamp-order_delivered_customer_date

diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date

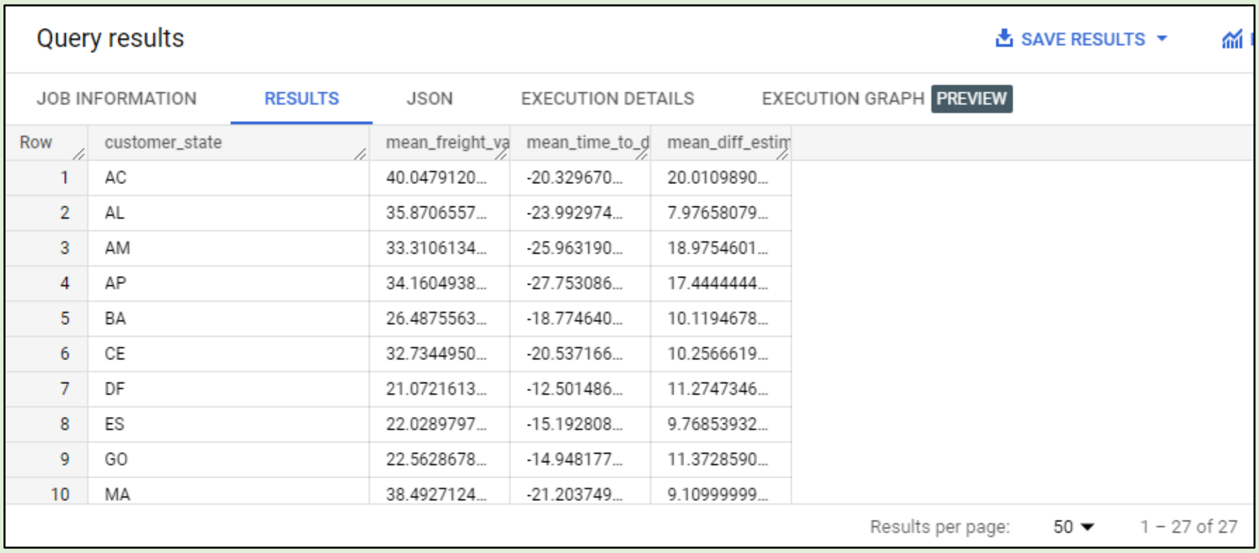
```sql
SELECT
    order_id,
    DATE_DIFF(order_purchase_timestamp,
            order_delivered_customer_date,
            day) AS time_to_delivery,
    DATE_DIFF(order_estimated_delivery_date,
            order_delivered_customer_date,
            day) AS diff_estimated_delivery
FROM
    `target.orders`
WHERE
    order_delivered_customer_date IS NOT NULL
ORDER BY 3;
```



Row	order_id	time_to_delivery	diff_estimated_c
1	1b3190b2dfa9d789e1f14c05b...	-208	-188
2	ca07593549f1816d26a572e06...	-209	-181
3	47b40429ed8cce3aee9199792...	-191	-175
4	2fe324febf907e3ea3f2aa9650...	-189	-167
5	285ab9426d6982034523a855f...	-194	-166
6	440d0d17af552815d15a9e41a...	-195	-165
7	c27815f7e3dd0b926b5855262...	-187	-162
8	0f4519c5f1c541ddec9f21b3bd...	-194	-161
9	d24e8541128cea179a11a6517...	-175	-161
10	2d7561026d542c8dbd8f0daea...	-188	-159

5.3 Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery

```sql
SELECT
    c.customer_state,
    AVG(oi.freight_value) mean_freight_value,
    AVG(DATE_DIFF(o.order_purchase_timestamp,
            o.order_delivered_customer_date,
            day)) AS mean_time_to_delivery,
    AVG(DATE_DIFF(o.order_estimated_delivery_date,
            o.order_delivered_customer_date,
            day)) AS mean_diff_estimated_delivery
FROM
    `target.orders` o
        JOIN
    `target.customers` c ON o.customer_id = c.customer_id
        JOIN
    `target.order_items` oi ON o.order_id = oi.order_id
WHERE
    o.order_delivered_customer_date IS NOT NULL
GROUP BY 1
ORDER BY 1;
```



Row	customer_state	mean_freight_va	mean_time_to_d	mean_diff_estim
1	AC	40.0479120...	-20.329670...	20.0109890...
2	AL	35.8706557...	-23.992974...	7.97658079...
3	AM	33.3106134...	-25.963190...	18.9754601...
4	AP	34.1604938...	-27.753086...	17.4444444...
5	BA	26.4875563...	-18.774640...	10.1194678...
6	CE	32.7344950...	-20.537166...	10.2566619...
7	DF	21.0721613...	-12.501486...	11.2747346...
8	ES	22.0289797...	-15.192808...	9.76853932...
9	GO	22.5628678...	-14.948177...	11.3728590...
10	MA	38.4927124...	-21.203749...	9.10999999...

Results per page: 50 ▾ 1 – 27 of 27

5.4 Sort the data to get the following:

Approach:

- Created a view with the required parameters which will be used in subsequent questions to sort the data.

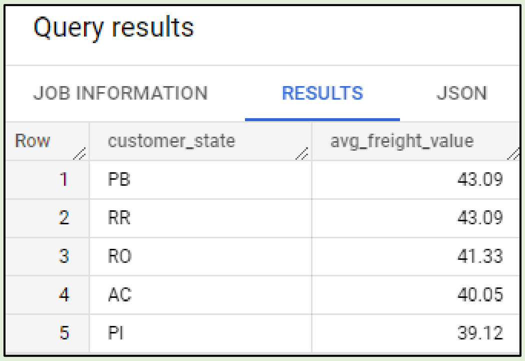
```sql
CREATE VIEW `target.states_wise_data` as
(SELECT
    c.customer_state as cust_states,
    ROUND(AVG(oi.freight_value), 2) AS avg_freight_value,
    ROUND(AVG(DATE_DIFF(o.order_purchase_timestamp,
            o.order_delivered_customer_date,
            day)),2) AS mean_time_to_delivery,
    ROUND(AVG(DATE_DIFF(o.order_estimated_delivery_date,
            o.order_delivered_customer_date,
            day)),2) AS mean_diff_estimated_delivery
FROM
    `target.customers` c
        JOIN
    `target.orders` o ON c.customer_id = o.customer_id
        JOIN
    `target.order_items` oi ON o.order_id = oi.order_id
WHERE
    o.order_delivered_customer_date IS NOT NULL
GROUP BY 1);
```

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

Note: Calculation is done on orders delivered to customers (where cust. delivery date is not null)

✓ **Top 5 states with highest average freight values:**

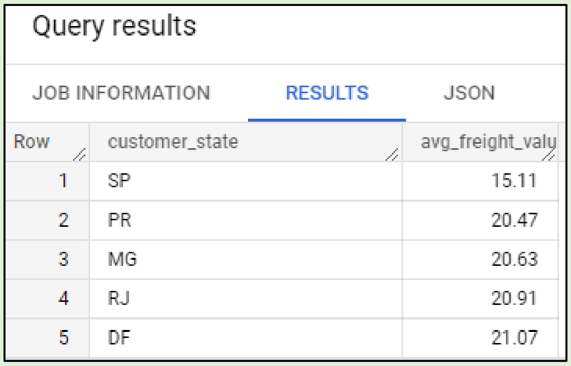
```
SELECT
  s.customer_state, s.avg_freight_value
FROM
  `target.states_wise_data` s
ORDER BY 2 DESC
LIMIT 5;
```


Query results

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

✓ **Top 5 states with lowest average freight values:**

```
SELECT
  s.customer_state, s.avg_freight_value
FROM
  `target.states_wise_data` s
ORDER BY 2 DESC
LIMIT 5;
```

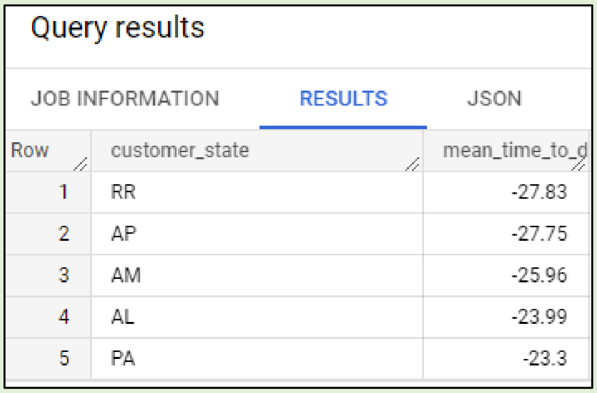

Query results

Row	customer_state	avg_freight_valu
1	SP	15.11
2	PR	20.47
3	MG	20.63
4	RJ	20.91
5	DF	21.07

5.6 Top 5 states with highest/lowest average time to delivery

✓ **Top 5 states with highest average time to delivery:**

```
SELECT
  s.customer_state, s.mean_time_to_delivery
FROM
  `target.states_wise_data` s
ORDER BY 2 DESC
LIMIT 5;
```

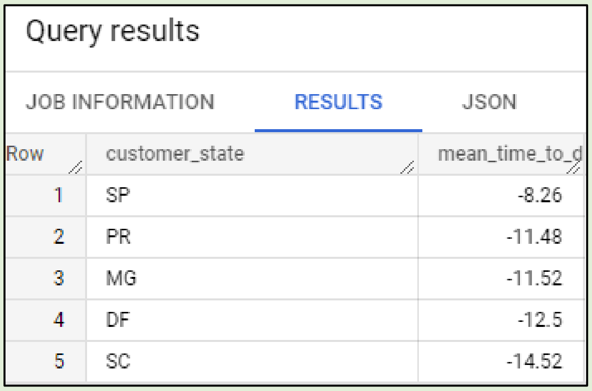


Query results

Row	customer_state	mean_time_to_d
1	RR	-27.83
2	AP	-27.75
3	AM	-25.96
4	AL	-23.99
5	PA	-23.3

✓ **Top 5 states with lowest average time to delivery:**

```
SELECT
  s.customer_state, s.mean_time_to_delivery
FROM
  `target.states_wise_data` s
ORDER BY 2 DESC
LIMIT 5;
```



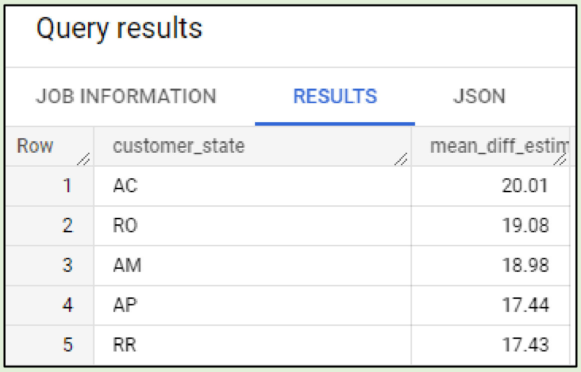
Query results

Row	customer_state	mean_time_to_d
1	SP	-8.26
2	PR	-11.48
3	MG	-11.52
4	DF	-12.5
5	SC	-14.52

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

✓ **Top 5 states with fastest delivery:**

```
SELECT
  s.customer_state, s. mean_diff_estimated_delivery
FROM
  `target.states_wise_data` s
ORDER BY 2 DESC
LIMIT 5;
```

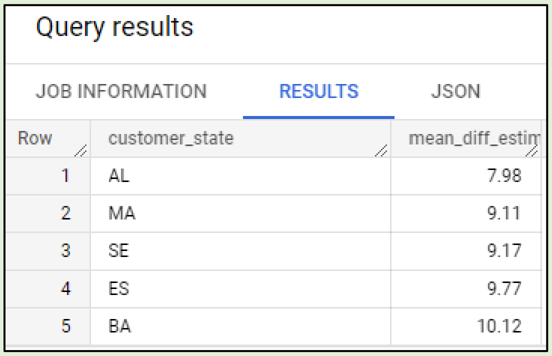


Query results

Row	customer_state	mean_diff_estim
1	AC	20.01
2	RO	19.08
3	AM	18.98
4	AP	17.44
5	RR	17.43

✓ **Top 5 states with slowest delivery:**

```
SELECT
  s.customer_state, s. mean_diff_estimated_delivery
FROM
  `target.states_wise_data` s
ORDER BY 2
LIMIT 5;
```



Query results

Row	customer_state	mean_diff_estim
1	AL	7.98
2	MA	9.11
3	SE	9.17
4	ES	9.77
5	BA	10.12

	Avg. Freight	Avg. Time to Delivery	Avg. Diff. Estimated Delivery
Top 5	PB, RR, RO, AC, PI (Higher Freight Cost)	RR,AP,AM,AL,PA (Higher Avg time to deliver)	AC, RO, AM, AP, RR (Fastest compared to Estimated date)
Bottom 5	SP, PR,MG, RJ, DF (Lower Fright Cost)	SP, PR, MG, DF, SC (Lower time to deliver)	AL, MA, SE, ES, BA (Not so fast compared to Estimated date)

Analysis:

- There is a major gap between the estimated delivery time and the actual delivery time.
- Data represents and suggests scope of improvements in delivery estimation and time to delivery, which can be improved by data analytics algorithms and models.
- Company could think of strenthening its position in the cities were they have better freight value, lesser time to delivery, and predictable delivery time. But should strongly focus on the cities where the order delivery service is worst. There are many instances were actual delivery date is much higher than estimated delivey date. These instanses affects customer loyality and satisfaction.

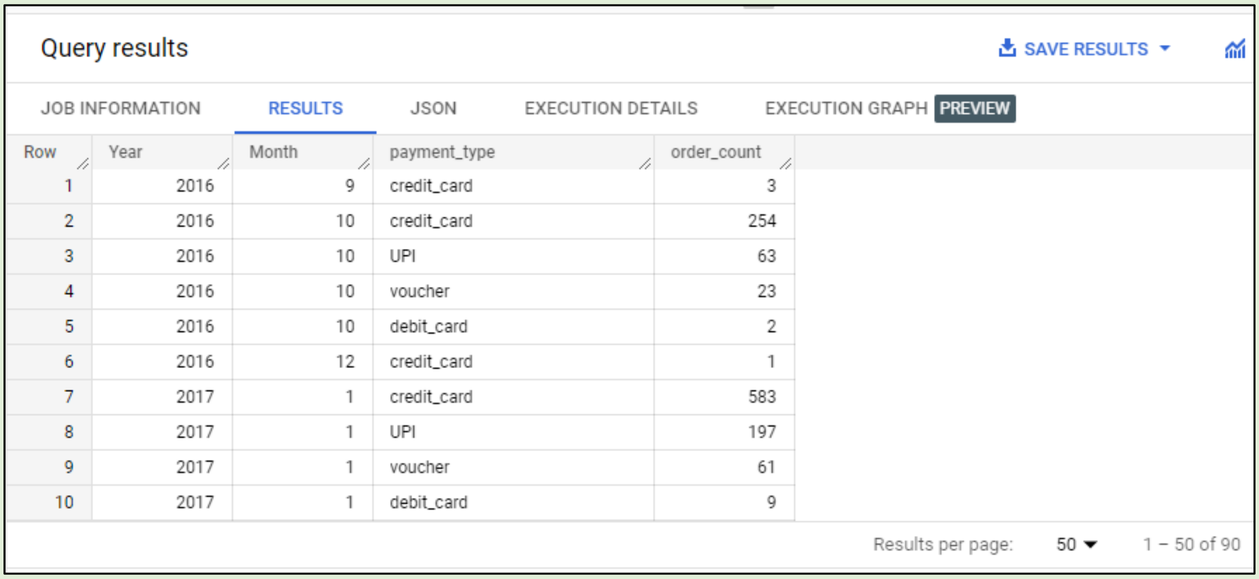
6 Payment type analysis:

6.1 Month over Month count of orders for different payment types

Approach:

- Extracted Year and Month from purchase timestamp and count aggregate on order_id to get Month on Month order counts.

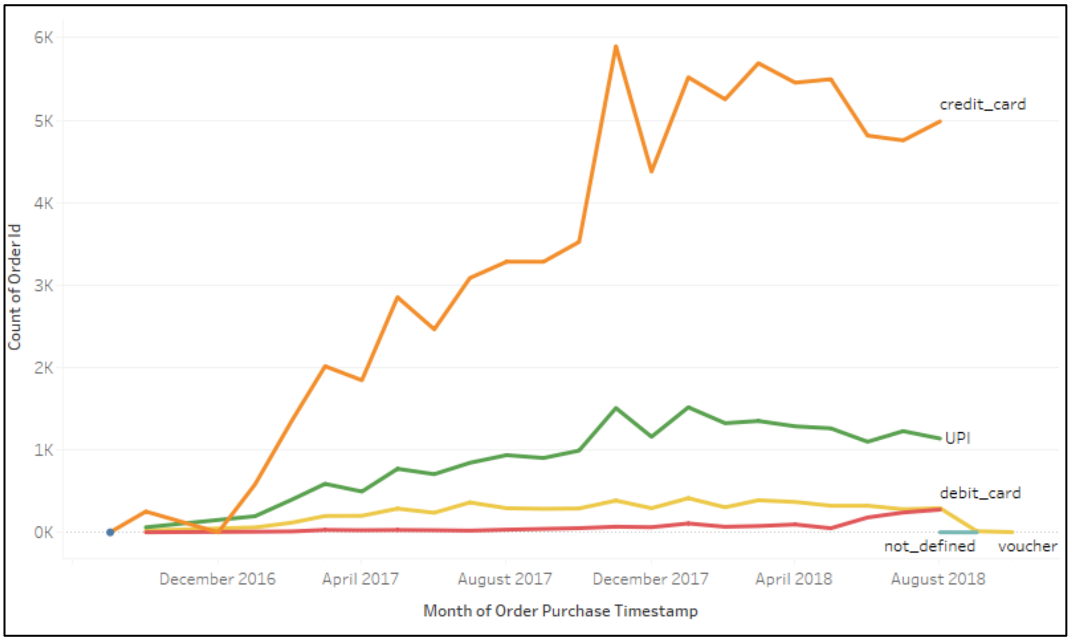
```
SELECT
    EXTRACT(YEAR FROM o.order_purchase_timestamp) AS Year,
    EXTRACT(MONTH FROM o.order_purchase_timestamp) AS Month,
    payment_type,
    COUNT(p.order_id) AS order_count
FROM
    `target.payments` p
        JOIN
    `target.orders` o ON p.order_id = o.order_id
GROUP BY 1 , 2 , 3
ORDER BY 1 , 2 , 4 DESC;
```



Row	Year	Month	payment_type	order_count
1	2016	9	credit_card	3
2	2016	10	credit_card	254
3	2016	10	UPI	63
4	2016	10	voucher	23
5	2016	10	debit_card	2
6	2016	12	credit_card	1
7	2017	1	credit_card	583
8	2017	1	UPI	197
9	2017	1	voucher	61
10	2017	1	debit_card	9

Analysis

- Credit Card is the most prefered payment method by the customers
- UPI adoption has increase over time along with orders

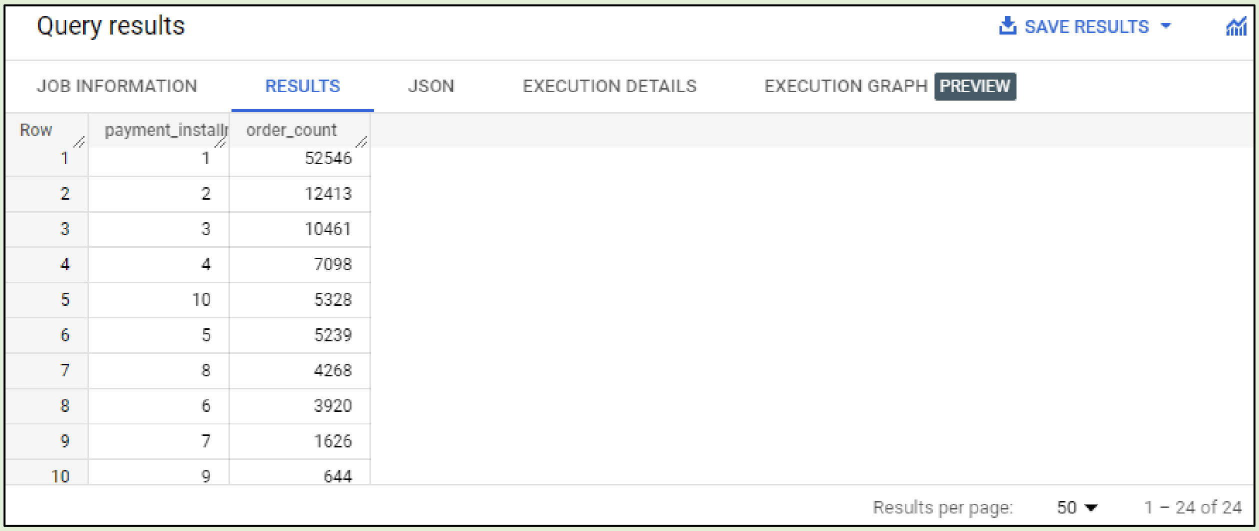


6.2 Count of orders based on the no. of payment installments

Approach:

- To get the desided data, I’ve grouped on payment_installments with count aggregation on order_id..

```
SELECT
    payment_installments, COUNT(order_id) order_count
FROM
    `target.payments`
GROUP BY 1
ORDER BY 2 DESC;
```



Row	payment_install	order_count
1	1	52546
2	2	12413
3	3	10461
4	4	7098
5	10	5328
6	5	5239
7	8	4268
8	6	3920
9	7	1626
10	9	644

Analysis

- Payment for majority of orders were made on one installments, which ensures cash flows
- Very less payments were done on installments > 12; again good for cash flows

7 Actionable Insights

1. The data shows a trend during the 2016-17 period that flattened during 2018.
2. We have a seasonality present (peaks available) during November, but it can’t be concluded due to insufficient data.
3. Overall Scenario regarding order counts looks like something should be going better. The orders are dropping, which needs to be analysed and resolved.
4. The majority of orders were placed in the Afternoon and Night.
5. Delivery time is much higher.
6. The distribution of customers is skewed.
7. SP is the state with the highest customer base, and it has more than 3x customer base to the next highest (RJ)
8. The top 3 states have more than 65% of the customer base, showing the scope of expansion.
9. The order cost has massive increased 137% YoY during the first eight months, showing considerable growth.
10. There is a significant gap between the estimated and actual delivery times.
11. Data represents and suggests the scope of improvements in delivery estimation and time to delivery, which data analytics algorithms and models can improve.
12. Credit Card is the most preferred payment method by customers.
13. UPI adoption has increased over time, along with orders
14. Payment for most orders was made in one instalment, ensuring cash flows.
15. Significantly fewer payments were made on instalments> 12; again suitable for cash flows, but it reflects people are less inclined to purchase higher ticket products.

8 Recommendations

1. The company could strengthen its position in the cities with better freight value, less delivery time, and predictable delivery time. But should intensely focus on the cities where the order delivery service is worst. There are many instances where the delivery date is much higher than the estimated delivery date. These instances affect customer loyalty and satisfaction.
2. The estimation of delivery time must be improved. Simple statistical models can enhance predictability.
3. The customer base is skewed toward some cities. The company has to look for opportunities to expand its customer base.
4. Despite lesser growth in order counts, the total order cost has seen a massive 137% increase. The company should maintain the same.
5. The company has to look into the flattening trend of order count in 2018. It should improve services in the cities where orders are declining
6. The company can plan a Marketing campaign to increase orders from the cities where orders count are declining.