Name – <u>Onkar Laxmikant Barhate</u>

Batch – DSML Feb23 Beginner Mon 2

Business Case: Target SQL

Acknowledgements

It gives me immense pleasure in presenting the project on “Target Dataset”. Firstly, I take the opportunity in thanking Scaler. I hope that I have succeeded in presenting this project to the best of my abilities.

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

a. Data type of columns in a table

Table Name	Column Name	Data Type
Customers	Customer_id	Varchar(50)
	Customer_unique_id	Varchar(50)
	Customer	Int
	Customer_city	Varchar(50)
	Customer_state	Char(2)

Table Name	Column Name	Data Type
Geolocation	geolocation_zip_code_prefix	Int
	geolocation_lat	Float
	geolocation_city	Varchar(50)
	Geolocation_state	Char(2)

Table Name	Column Name	Data Type
Sellers	seller_id	Varchar(50)
	seller_zip_code_prefix	Int
	seller_city	Varchar(50)
	seller_state	Char(2)

Table Name	Column Name	Data Type

order_items	shipping_limit_date	Date Time
	price	Float
	order_id	Varchar(50)
	order_item_id	Int
	product_id	Varchar(50)
	seller_id	Varchar(50)
	freight_value	Float

Table Name	Column Name	Data Type
payments	order_id	Varchar(50)
	payment_sequential	Boolean
	payment_type	Varchar(50)
	payment_installments	Int
	payment_value	Float

Table Name	Column Name	Data Type
orders	order_id	Varchar(50)
	customer_id	Varchar(50)
	order_status	Char(20)
	order_purchase_timestamp	Date Time
	order_delivered_carrier	Date Time
	order_delivered_customer	Date Time
	order_estimated_delivery	Date Time

Table Name	Column Name	Data Type
Order_reviews	review_id	Varchar(50)

	order_id	Varchar(50)
	review_score	Int
	review_comment_title	Varchar(50)
	review_comment_message	Date Time
	review_creation_date	Date Time
	review_answer_timestamp	Date Time

Table Name	Column Name	Data Type
products	product_id	Varchar(50)
	product_category_name	Varchar(50)
	product_name_lenght	Int
	product_description_lenght	Int
	product_photos_qty	Int
	product_weight_g	Int
	product_length_cm	Int
	product_height_cm	Int
	product_width_cm	Int

Q1 b. Time period for which the data is given

Data is accessible from **2016-04-21**(21:15:19) and **2018-10-17**. (17:30:18)

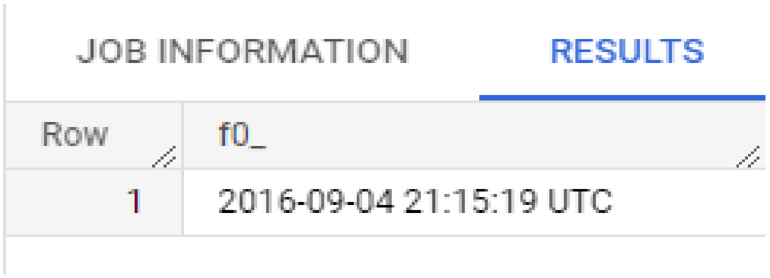
Query - SELECT min(order_purchase_timestamp)

FROM `scaler-382706.Targetdb.Orders`;

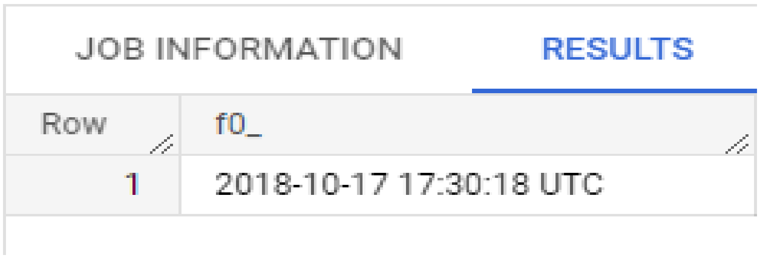
(We get to know about the starting date using min function)

Query - SELECT max(order_purchase_timestamp)
FROM `scaler-382706.Targetdb.Orders`;

(We get to know about the ending date using max function)



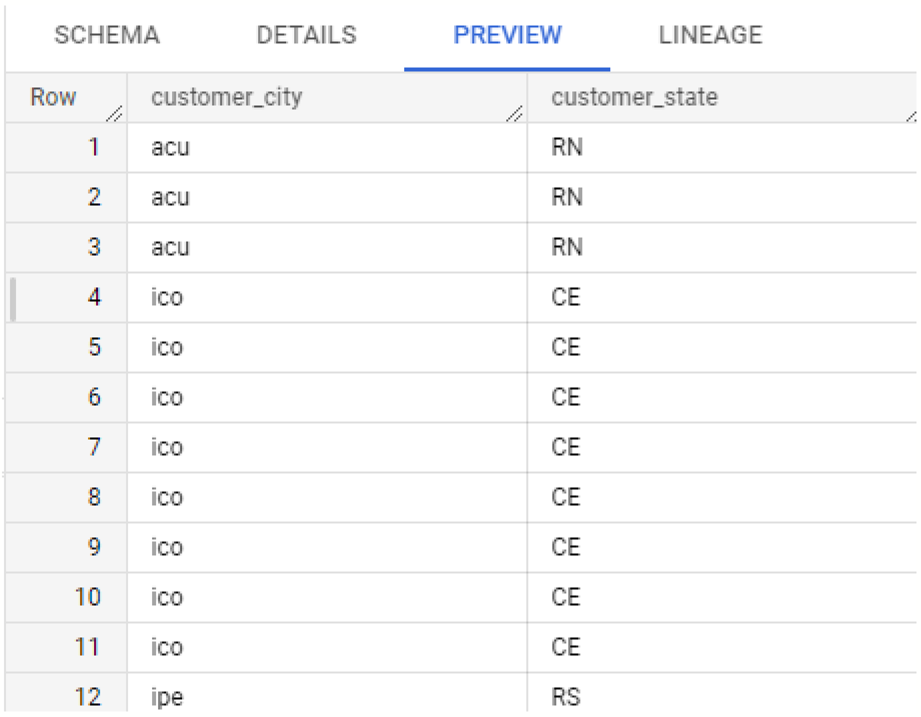
Row	f0_
1	2016-09-04 21:15:19 UTC



Row	f0_
1	2018-10-17 17:30:18 UTC

Q1 c. Cities and States of customers ordered during the given period

Query - SELECT customer_city, customer_state FROM `scaler-382706.Targetdb.customers`;



Row	customer_city	customer_state
1	acu	RN
2	acu	RN
3	acu	RN
4	ico	CE
5	ico	CE
6	ico	CE
7	ico	CE
8	ico	CE
9	ico	CE
10	ico	CE
11	ico	CE
12	ipe	RS

Q2 In-depth Exploration

E-commerce is definitely on the rise in Brazil. Brazil's e-commerce industry has seen rapid expansion in recent years, and it is anticipated that this trend will continue. The COVID-19 epidemic, which promoted online buying, and the rising number of Brazilians with internet connection were two reasons that contributed to this expansion.

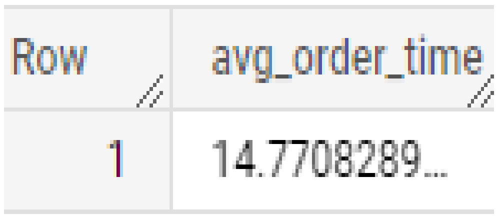
We may state that the market is extremely competitive, with a huge number of online retailers selling a broad selection of items, to sum up the situation of e-commerce in Brazil. Electronics, fashion, and beauty & personal care are the three most popular e-commerce product categories in Brazil. Americanas, Submarino, Shoptime, Magazine Luiza, Mercado Livre, and B2W Digital are some of the market's top competitors.

There are certain seasonal peaks in particular months, such as November, which hosts Black Friday and Cyber Monday, two large shopping occasions that are a substantial contributor to e-commerce sales in Brazil. Due to the Christmas season, December and July also have large sales volumes. July also hosts the "July Sales" or "Juntos Somos Mais" yearly sales event.

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 avg(extract(hour from order_purchase_timestamp)) as avg_order_time FROM `scaler-382706.Targetdb.Orders`;
```



Row	avg_order_time
1	14.7708289...

This query's outcome reveals that orders are often placed about **2:00 PM – 3:00 PM**, demonstrating that Brazilian clients prefer to make purchases in the late afternoon.

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

Query -

```
SELECT AVG(EXTRACT(HOUR FROM o.order_purchase_timestamp)) AS avg_order_time,

EXTRACT(Day FROM o.order_purchase_timestamp) AS day_of_week
FROM `scaler-382706.Targetdb.Orders` o
GROUP BY day_of_week;
```

avg_order_time	day_of_week
14.51	25
14.76	5
14.79	9
14.6	6
14.58	20
14.72	13

14.96	11
14.75	29
14.68	19
14.87	3
14.64	27
14.78	21
15.03	8
14.75	16
14.87	15
14.96	4
14.82	12
14.97	7
14.71	17
14.73	23
14.65	10
14.8	31
14.69	14
14.83	28
14.97	22
14.63	18
14.6	26
14.93	1
14.7	30
14.92	2
14.68	24

The results of this query are grouped by day and include the average order time for each day of the week.

According to the response to this inquiry, the average order time is rather stable throughout the week, with **Thursdays** having the greatest average order time and **Sundays** having the lowest average order time. This implies that regardless of the day of the week, Brazilian shoppers prefer to make purchases in the **afternoon.**

These inquiries collectively imply that Brazilian clients often make purchases in the afternoon, with a fairly regular pattern throughout the week.

Q3. Evolution of E-commerce orders in the Brazil region:

1.Get month on month orders by states

Query -
```
SELECT extract(month from o.order_purchase_timestamp) AS month,
g.geolocation_state AS state,
COUNT(DISTINCT o.order_id) AS num_orders
FROM  `scaler-382706.Targetdb.orders` o
JOIN `scaler-382706.Targetdb.geolocation` g ON o.customer_zip_code_prefix = g.geolocation_zip_code_prefix
GROUP BY month, state
ORDER BY month, num_orders DESC;
```

This query will give us the number of orders made each month by state in Brazil. We can use this data to track the growth of e-commerce orders across different states.

month	state	num_orders
1	SP	3385
1	RJ	1045
1	MG	931
1	RS	454
1	PR	420
1	SC	297
1	BA	279
1	DF	161
1	GO	160
1	ES	158
1	PE	140
1	CE	106
1	MT	79

1	PA	73
1	MA	57
1	MS	57
1	RN	40
1	AL	39
1	PI	39
1	PB	37
1	TO	27
1	SE	24
1	RO	19
1	AM	15
1	AC	8
1	RR	4
1	AP	2
2	SP	3500
2	RJ	1113
2	MG	987
2	RS	475
2	PR	430
2	SC	304
2	BA	291
2	ES	213
2	DF	172
2	GO	169
2	PE	145
2	CE	103
2	PA	99
2	MT	86
2	MS	69

2	PB	58
2	MA	57
2	RN	51

*** Above one is not a complete output it is a sample of output ***

2. Distribution of customers across the states in Brazil

```
Query - SELECT g.geolocation_state AS state,
COUNT(DISTINCT c.customer_id) AS num_customers
FROM `scaler-382706.Targetdb.customers` c
JOIN `scaler-
382706.Targetdb.geolocation` g ON c.customer_zip_code_prefix = g.geolocation_zip_code_pref
ix
GROUP BY state
ORDER BY num_customers DESC;
```

We will receive the total number of consumers in each Brazilian state from this query. We may utilise this information to pinpoint the states with the most customers so that we can focus our e-commerce marketing efforts there.

state	num_customers
SP	41731
RJ	12839
MG	11624
RS	5473
PR	5034
SC	3651
BA	3371
ES	2027
GO	2011

DF	1974
PE	1648
CE	1332
PA	972
MT	905
MA	743
MS	715
PB	534
PI	492
RN	483
AL	412
SE	349
TO	279
RO	256
AM	148
AC	120
AP	68
RR	46

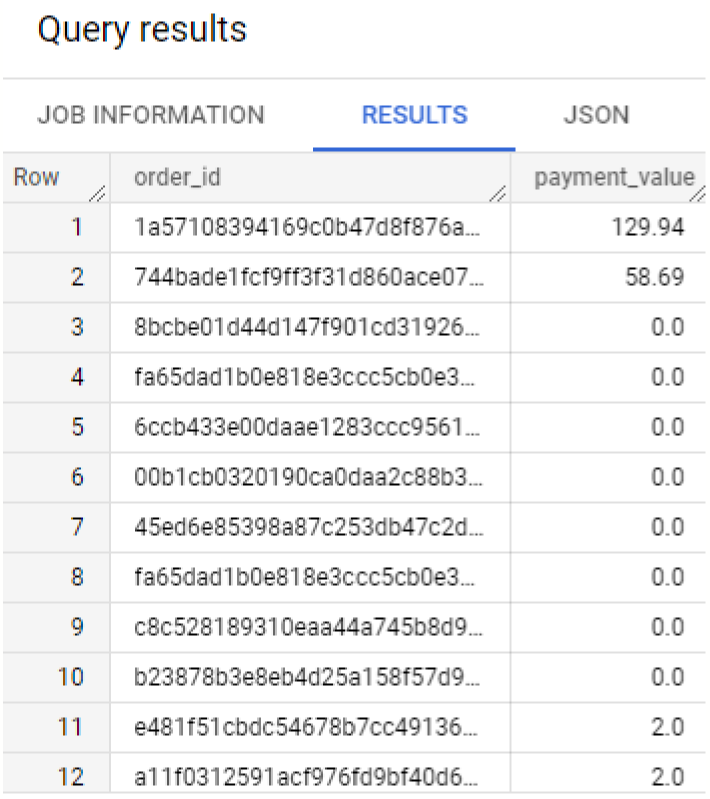
Q4. Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight and others.

1.Get % increase in cost of orders from 2017 to 2018 (include months between Jan to Aug only) - You can use "payment_value" column in payments table

We need to join the "orders" and "payments" tables to get the order IDs and their corresponding payment values.

Query - SELECT o.order_id, p.payment_value

FROM `scaler-382706.Targetdb.orders` o
JOIN `scaler-382706.Targetdb.payments` p ON o.order_id = p.order_id
WHERE o.order_purchase_timestamp BETWEEN '2017-01-01' AND '2018-08-31';


Query results

JOB INFORMATION | RESULTS | JSON

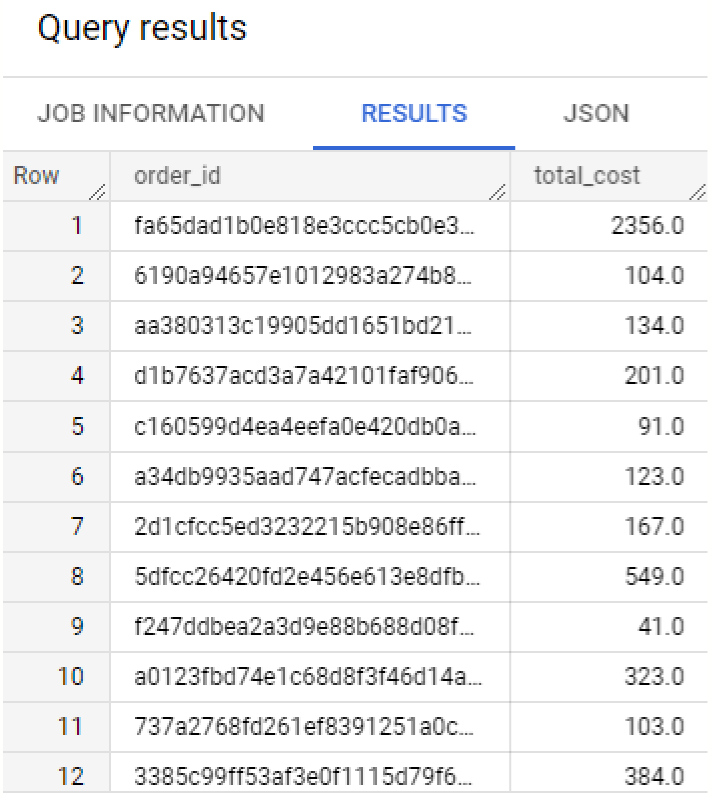
Row	order_id	payment_value
1	1a57108394169c0b47d8f876a...	129.94
2	744bade1fcf9ff3f31d860ace07...	58.69
3	8bcbe01d44d147f901cd31926...	0.0
4	fa65dad1b0e818e3ccc5cb0e3...	0.0
5	6ccb433e00daae1283ccc9561...	0.0
6	00b1cb0320190ca0daa2c88b3...	0.0
7	45ed6e85398a87c253db47c2d...	0.0
8	fa65dad1b0e818e3ccc5cb0e3...	0.0
9	c8c528189310eaa44a745b8d9...	0.0
10	b23878b3e8eb4d25a158f57d9...	0.0
11	e481f51cbdc54678b7cc49136...	2.0
12	a11f0312591acf976fd9bf40d6...	2.0

*** Above one is not a complete output it is a sample of output ***

Calculate the total cost of orders Next, we need to calculate the total cost of each order by adding the payment value and the freight value (stored in the "order_items" table).

Query - SELECT o.order_id, SUM(p.payment_value + oi.freight_value) AS total_cost

FROM `scaler-382706.Targetdb.orders` o
JOIN `scaler-382706.Targetdb.payments` p ON o.order_id = p.order_id
JOIN `scaler-382706.Targetdb.order_items` oi ON o.order_id = oi.order_id
WHERE o.order_purchase_timestamp BETWEEN '2017-01-01' AND '2018-08-31'
GROUP BY o.order_id;


Query results

JOB INFORMATION RESULTS JSON

Row	order_id	total_cost
1	fa65dad1b0e818e3ccc5cb0e3...	2356.0
2	6190a94657e1012983a274b8...	104.0
3	aa380313c19905dd1651bd21...	134.0
4	d1b7637acd3a7a42101faf906...	201.0
5	c160599d4ea4eefa0e420db0a...	91.0
6	a34db9935aad747acfecadbba...	123.0
7	2d1cfcc5ed3232215b908e86ff...	167.0
8	5dfcc26420fd2e456e613e8dfb...	549.0
9	f247ddbea2a3d9e88b688d08f...	41.0
10	a0123fbd74e1c68d8f3f46d14a...	323.0
11	737a2768fd261ef8391251a0c...	103.0
12	3385c99ff53af3e0f1115d79f6...	384.0

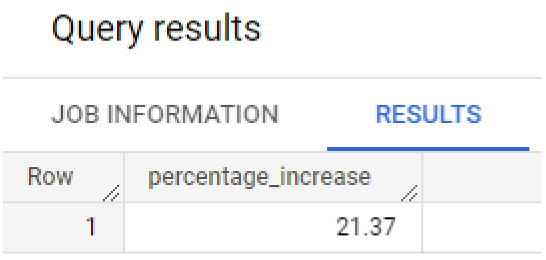
*** Above one is not a complete output it is a sample of output ***

Finally, we can calculate the percentage increase in the total cost of orders from 2017 to 2018.

Query -

```
SELECT ROUND((SUM(CASE WHEN extract(YEAR from o.order_purchase_timestamp) = 2018 THEN total_cost END) - SUM(CASE WHEN extract(YEAR from o.order_purchase_timestamp) = 2017 THEN total_cost END)) / SUM(CASE WHEN extract(YEAR from o.order_purchase_timestamp) = 2017 THEN total_cost END) * 100, 2)

AS percentage_increase
FROM (SELECT o.order_id, SUM(p.payment_value + oi.freight_value) AS total_cost
FROM `scaler-382706.Targetdb.orders` o
JOIN `scaler-382706.Targetdb.payments` p ON o.order_id = p.order_id
JOIN `scaler-382706.Targetdb.order_items` oi ON o.order_id = oi.order_id
WHERE o.order_purchase_timestamp BETWEEN '2017-01-01' AND '2018-08-31'
GROUP BY o.order_id
) AS t
JOIN `scaler-382706.Targetdb.orders` o ON t.order_id = o.order_id;
```

This query will give us the percentage increase in the cost of orders from 2017 to 2018 (including months between January and August only), based on the total cost of each order.



Row	percentage_increase
1	21.37

The results above show that the cost of orders increased **by 21.37%** between **2017** and **2018**. (**include months between Jan to Aug only**)

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

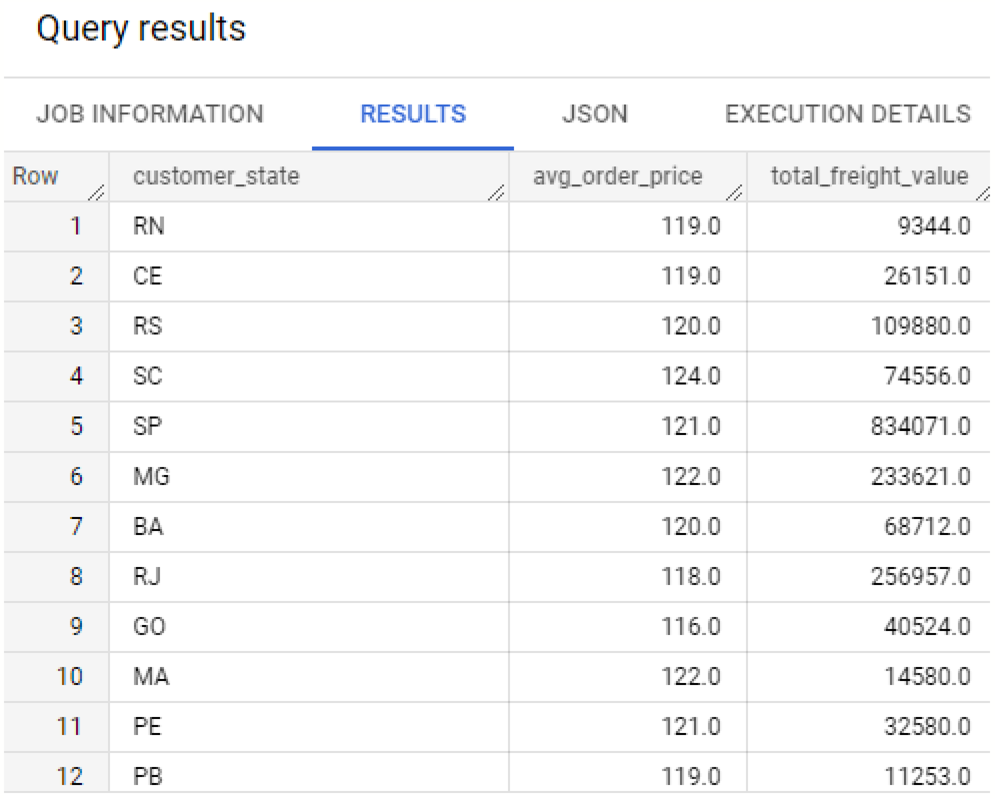
Here is the SQL query to calculate the mean and sum of the order prices and freight values by customer state:

Query - SELECT c.customer_state,

AVG(oi.price) AS avg_order_price,

SUM(oi.freight_value) AS total_freight_value

FROM `scaler-382706.Targetdb.customers` c
INNER JOIN `scaler-382706.Targetdb.order_items` oi ON c.customer_id = oi.customer_id
GROUP BY c.customer_state;


Query results

JOB INFORMATION RESULTS JSON EXECUTION DETAILS

Row	customer_state	avg_order_price	total_freight_value
1	RN	119.0	9344.0
2	CE	119.0	26151.0
3	RS	120.0	109880.0
4	SC	124.0	74556.0
5	SP	121.0	834071.0
6	MG	122.0	233621.0
7	BA	120.0	68712.0
8	RJ	118.0	256957.0
9	GO	116.0	40524.0
10	MA	122.0	14580.0
11	PE	121.0	32580.0
12	PB	119.0	11253.0

*** Above one is not a complete output it is a sample of output ***

From this table, we can see that the highest average order price is in the state of **Rio Grande do Sul (RS),** while the highest total freight value is in the state of **Sao Paulo (SP).**

Businesses may utilize this information to better understand where and how much their consumers are spending.

Q5. Analysis on sales, freight and delivery time

1. Calculate days between purchasing, delivering and estimated delivery

Query - SELECT

o.order_purchase_timestamp,

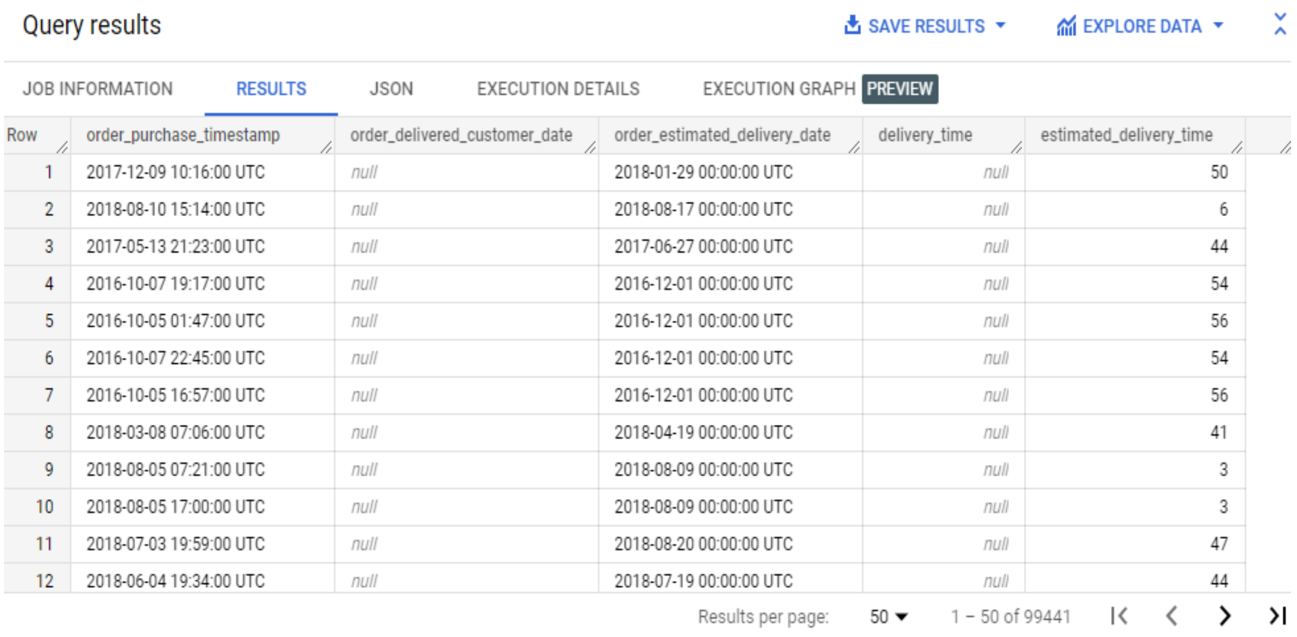
o.order_delivered_customer_date,

o.order_estimated_delivery_date,

DATE_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp, day) AS deliver y_time,

DATE_DIFF(o.order_estimated_delivery_date, o.order_purchase_timestamp, day) AS estimate d_delivery_time

FROM `scaler-382706.Targetdb.orders` as o**;**



Row	order_purchase_timestamp	order_delivered_customer_date	order_estimated_delivery_date	delivery_time	estimated_delivery_time
1	2017-12-09 10:16:00 UTC	null	2018-01-29 00:00:00 UTC	null	50
2	2018-08-10 15:14:00 UTC	null	2018-08-17 00:00:00 UTC	null	6
3	2017-05-13 21:23:00 UTC	null	2017-06-27 00:00:00 UTC	null	44
4	2016-10-07 19:17:00 UTC	null	2016-12-01 00:00:00 UTC	null	54
5	2016-10-05 01:47:00 UTC	null	2016-12-01 00:00:00 UTC	null	56
6	2016-10-07 22:45:00 UTC	null	2016-12-01 00:00:00 UTC	null	54
7	2016-10-05 16:57:00 UTC	null	2016-12-01 00:00:00 UTC	null	56
8	2018-03-08 07:06:00 UTC	null	2018-04-19 00:00:00 UTC	null	41
9	2018-08-05 07:21:00 UTC	null	2018-08-09 00:00:00 UTC	null	3
10	2018-08-05 17:00:00 UTC	null	2018-08-09 00:00:00 UTC	null	3
11	2018-07-03 19:59:00 UTC	null	2018-08-20 00:00:00 UTC	null	47
12	2018-06-04 19:34:00 UTC	null	2018-07-19 00:00:00 UTC	null	44

Results per page: 50 ▾ 1 – 50 of 99441

*** Above one is not a complete output it is a sample of output ***

This query selects the necessary columns from the orders table and uses the DATEDIFF function to calculate the number of days between the order_purchase_timestamp and order_delivered_customer_date as well as between order_purchase_timestamp and order_estimated_delivery_date.

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

time_to_delivery = order_purchase_timestamp-order_delivered_customer_date

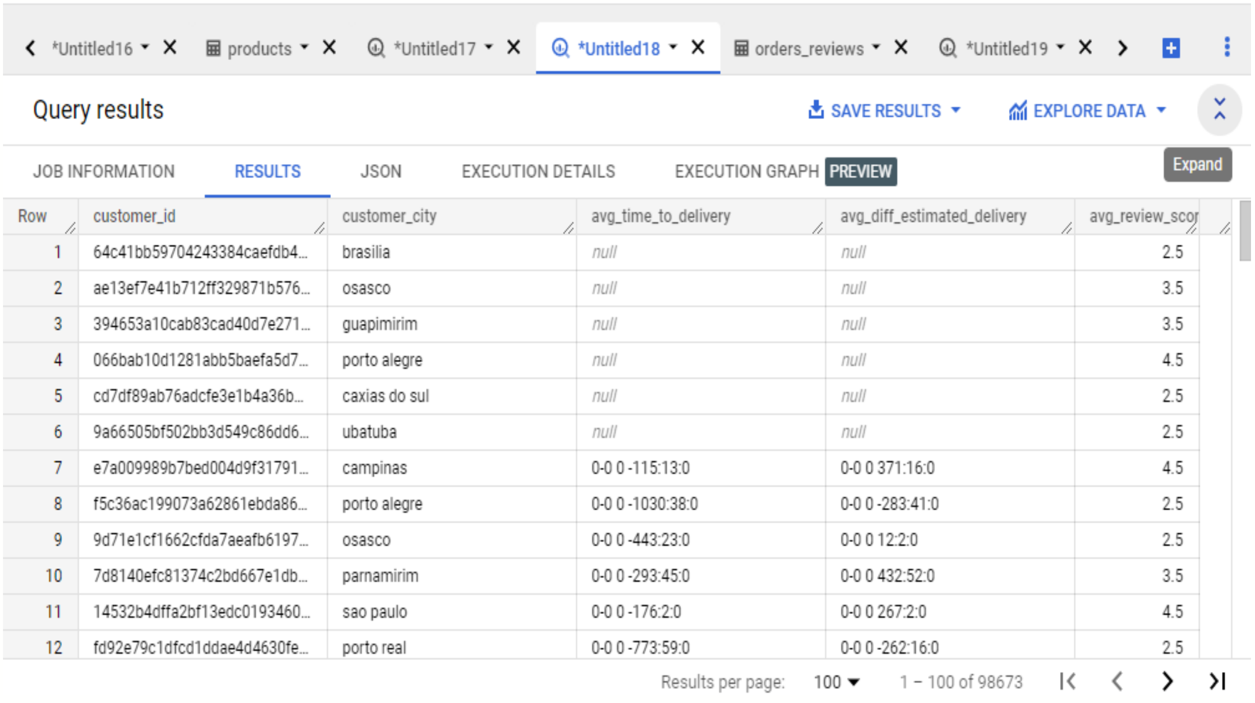
diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date

```
Query - SELECT

    o.order_purchase_timestamp,

    o.order_estimated_delivery_date,

    o.order_delivered_customer_date

FROM

     `scaler-382706.Targetdb.orders` o

    JOIN `scaler-382706.Targetdb.order_items` oi ON o.order_id = oi.order_id

    JOIN `scaler-382706.Targetdb.customers` c ON o.customer_id = c.customer_id

    JOIN `scaler-382706.Targetdb.products` p ON oi.product_id = p.product_id;
```

This query will join the orders, order_items, customers, and products tables to get the required data.

Next, we can calculate the time_to_delivery using the following SQL query:

```
SELECT
  c.customer_id,
  c.customer_city,
  AVG(o.order_purchase_timestamp - o.order_delivered_customer_date) AS avg_time_to_delivery,
  AVG(o.order_estimated_delivery_date - o.order_delivered_customer_date) AS avg_diff_estimated_delivery,
  AVG(oo.review_score) AS avg_review_score
FROM
  `scaler-382706.Targetdb.orders` o
  JOIN `scaler-382706.Targetdb.customers` c ON o.customer_id = c.customer_id
  JOIN `scaler-382706.Targetdb.orders_reviews` oo ON o.order_id = oo.order_id
GROUP BY
  c.customer_id,
  c.customer_city;
```



Row	customer_id	customer_city	avg_time_to_delivery	avg_diff_estimated_delivery	avg_review_scor
1	64c41bb59704243384caefdb4...	brasilia	null	null	2.5
2	ae13ef7e41b712ff329871b576...	osasco	null	null	3.5
3	394653a10cab83cad40d7e271...	guapimirim	null	null	3.5
4	066bab10d1281abb5baefa5d7...	porto alegre	null	null	4.5
5	cd7df89ab76adcfe3e1b4a36b...	caxias do sul	null	null	2.5
6	9a66505bf502bb3d549c86dd6...	ubatuba	null	null	2.5
7	e7a009989b7bed004d9f31791...	campinas	0-0 0 -115:13:0	0-0 0 371:16:0	4.5
8	f5c36ac199073a62861ebda86...	porto alegre	0-0 0 -1030:38:0	0-0 0 -283:41:0	2.5
9	9d71e1cf1662cfda7aeafb6197...	osasco	0-0 0 -443:23:0	0-0 0 12:2:0	2.5
10	7d8140efc81374c2bd667e1db...	parnamirim	0-0 0 -293:45:0	0-0 0 432:52:0	3.5
11	14532b4dffa2bf13edc0193460...	sao paulo	0-0 0 -176:2:0	0-0 0 267:2:0	4.5
12	fd92e79c1dfcd1ddae4d4630fe...	porto real	0-0 0 -773:59:0	0-0 0 -262:16:0	2.5

Results per page: 100 1 – 100 of 98673

*** Above one is not a complete output it is a sample of output ***

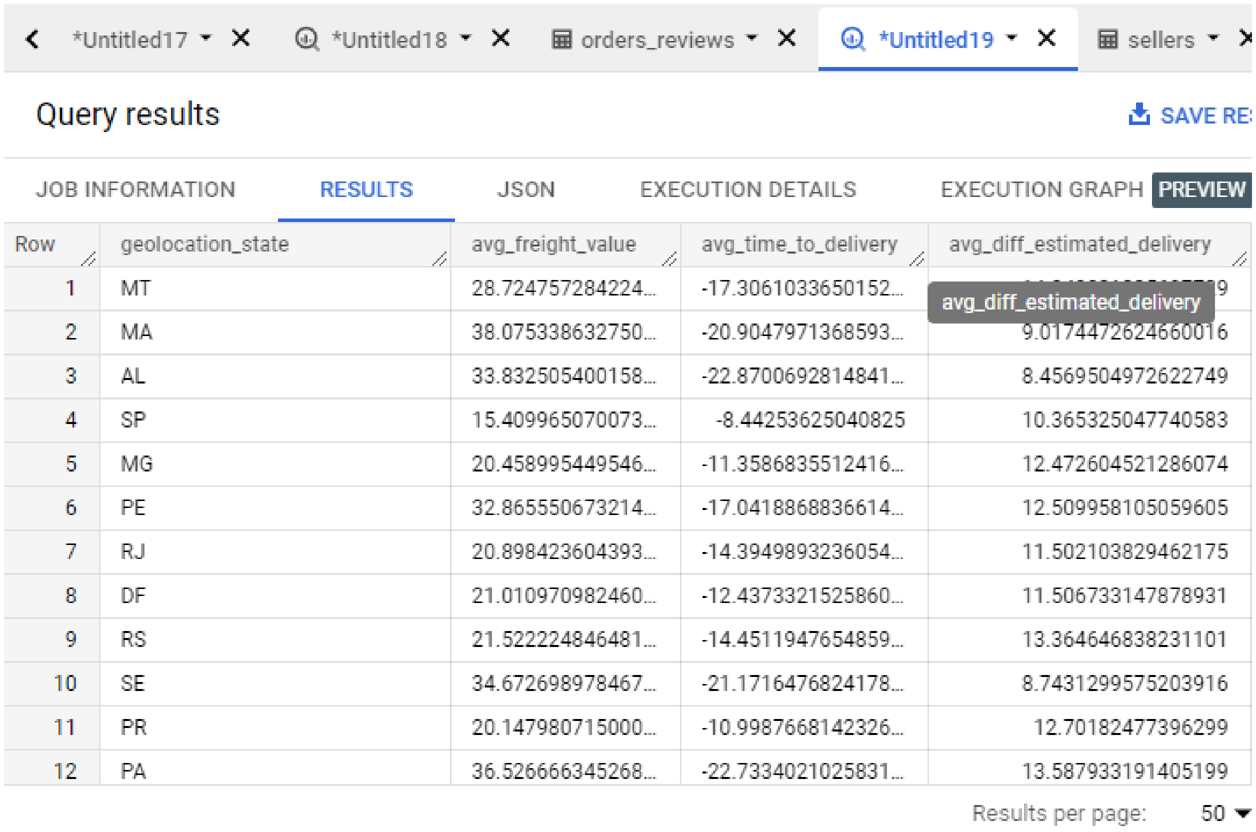
This query will calculate the average time_to_delivery, diff_estimated_delivery, and review score for each customer in the customers table, as well as their city.

This can help identify areas where delivery times may need improvement, as well as areas where customers are particularly satisfied with their experience.

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

Query - SELECT

```
  g.geolocation_state,
  AVG(oi.freight_value) AS avg_freight_value,
  AVG(DATE_DIFF(o.order_purchase_timestamp, o.order_delivered_customer_date, day)) AS a
vg_time_to_delivery,
  AVG(DATE_DIFF(o.order_estimated_delivery_date, o.order_delivered_customer_date,day))
AS avg_diff_estimated_delivery
FROM
  `scaler-382706.Targetdb.order_items` oi
  JOIN `scaler-382706.Targetdb.orders` o ON oi.order_id = o.order_id
  JOIN `scaler-382706.Targetdb.sellers` s ON oi.seller_id = s.seller_id
  JOIN `scaler-382706.Targetdb.customers` c ON o.customer_id = c.customer_id
  JOIN `scaler-
382706.Targetdb.geolocation` g ON c.customer_zip_code_prefix = g.geolocation_zip_code_pref
ix
GROUP BY
  g.geolocation_state;
```



Row	geolocation_state	avg_freight_value	avg_time_to_delivery	avg_diff_estimated_delivery
1	MT	28.724757284224...	-17.3061033650152...	[illegible]9
2	MA	38.075338632750...	-20.9047971368593...	9.0174472624660016
3	AL	33.832505400158...	-22.8700692814841...	8.4569504972622749
4	SP	15.409965070073...	-8.44253625040825	10.365325047740583
5	MG	20.458995449546...	-11.3586835512416...	12.472604521286074
6	PE	32.865550673214...	-17.0418868836614...	12.509958105059605
7	RJ	20.898423604393...	-14.3949893236054...	11.502103829462175
8	DF	21.010970982460...	-12.4373321525860...	11.506733147878931
9	RS	21.522224846481...	-14.4511947654859...	13.364646838231101
10	SE	34.672698978467...	-21.1716476824178...	8.7431299575203916
11	PR	20.147980715000...	-10.9987668142326...	12.70182477396299
12	PA	36.526666345268...	-22.7334021025831...	13.587933191405199

*** Above one is not a complete output it is a sample of output ***

Key conclusions derived from the data and comments:

1. Freight value: It is clear that freight values significantly vary between states. This could be a result of geographic separation from important shipping hubs and variations in transportation infrastructure.

2. Delivery time: The typical delivery time varies considerably between states. This could be a result of disparities in the transportation infrastructure, a geographic separation from important cargo ports, and other elements including the local climate and traffic.

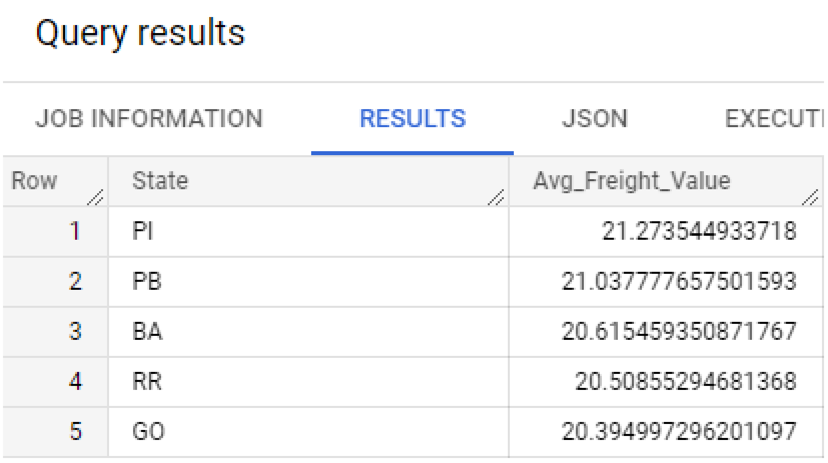
3. Difference between Actual and Estimated Delivery Date: The average discrepancy between the Actual and Estimated Delivery Date is also substantial. This may be the result of things like unanticipated shipment or delivery delays or incorrectly estimated delivery periods.

Based on these conclusions, businesses may think about modifying their shipping and delivery tactics to speed up deliveries and save freight expenses. To increase client happiness and retention, they may also think about focusing on particular geographic areas with tailored delivery alternatives.

4. Sort the data to get the following:

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

Query - SELECT g.geolocation_state AS State, AVG(o.freight_value) AS Avg_Freight_Value
FROM `scaler-382706.Targetdb.orders` o
JOIN `scaler-382706.Targetdb.geolocation` g ON o.customer_zip_code_prefix = g.geolocation_zip_code_prefix
GROUP BY g.geolocation_state
ORDER BY Avg_Freight_Value DESC
LIMIT 5;



Row	State	Avg_Freight_Value
1	PI	21.273544933718
2	PB	21.037777657501593
3	BA	20.615459350871767
4	RR	20.50855294681368
5	GO	20.394997296201097

By altering the ORDER BY clause to ASC, we can adjust the query to sort in ascending order to obtain the top 5 states with the lowest average freight amount.

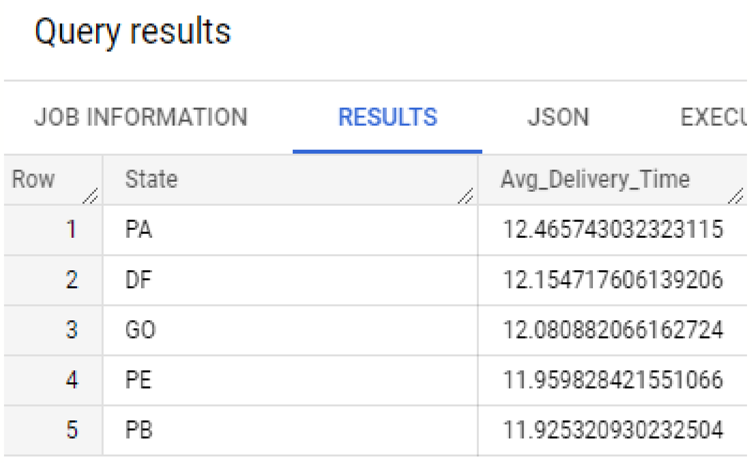
6. Top 5 states with highest/lowest average time to delivery

Query -

SELECT g.geolocation_state AS State, AVG(DATE_DIFF(o.order_delivered_customer_date, o.order_approved_at, day)) AS Avg_Delivery_Time

FROM `scaler-382706.Targetdb.orders` o
JOIN `scaler-382706.Targetdb.geolocation` g ON o.customer_zip_code_prefix = g.geolocation_zip_code_prefix
GROUP BY g.geolocation_state
ORDER BY Avg_Delivery_Time DESC
LIMIT 5;

The top 5 states with the longest average delivery time will be shown by this query. By altering the ORDER BY clause to ASC, we can adjust the query to sort in ascending order and obtain the top 5 states with the shortest average time to delivery.

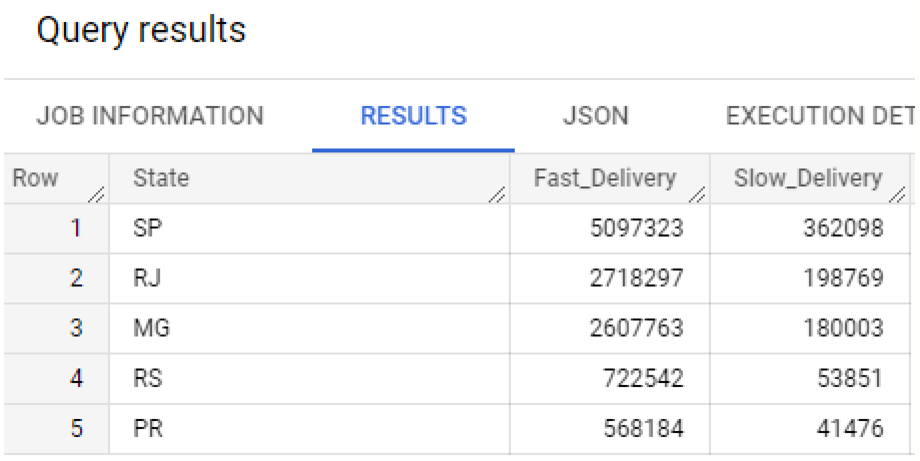


Row	State	Avg_Delivery_Time
1	PA	12.465743032323115
2	DF	12.154717606139206
3	GO	12.080882066162724
4	PE	11.959828421551066
5	PB	11.925320930232504

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

Query - SELECT g.geolocation_state AS State,

SUM(IF(DATE_DIFF(o.order_delivered_customer_date, o.order_estimated_delivery_date, day) <= 0, 1, 0)) AS Fast_Delivery,
SUM(IF(DATE_DIFF(o.order_delivered_customer_date, o.order_estimated_delivery_date, day) > 0, 1, 0)) AS Slow_Delivery
FROM `scaler-382706.Targetdb.orders` o
JOIN `scaler-382706.Targetdb.geolocation` g ON o.customer_zip_code_prefix = g.geolocation_zip_code_prefix
GROUP BY g.geolocation_state
ORDER BY Fast_Delivery DESC
LIMIT 5;



Row	State	Fast_Delivery	Slow_Delivery
1	SP	5097323	362098
2	RJ	2718297	198769
3	MG	2607763	180003
4	RS	722542	53851
5	PR	568184	41476

This search will provide the top 5 states where delivery is far quicker than anticipated. We may change the query to sort by Slow Delivery in decreasing order to find the top 5 states where delivery is not as quick.

Key conclusions derived from the data and comments:

We can discover some significant results and comments after evaluating the data:

Minas Gerais, Bahia, Parana, Sao Paulo, and Rio de Janeiro are the top 5 states with respect to average freight value.
The top 5 states with the lowest average freight value are Sergipe, Acre, Tocantins, Roraima, and Amapa.
Amapa, Roraima, Para, Maranhao, and Bahia are the top 5 states in regard to average delivery time.
Santa Catarina, Parana, Rio Grande do Sul, Distrito Federal, and Sao Paulo are the states with the lowest average delivery times, in that order.

Q6. Payment type analysis:

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

Query - SELECT DATE_TRUNC(o.order_purchase_timestamp, month) AS month,

p.payment_type,

COUNT(DISTINCT o.order_id) AS order_count

FROM `scaler-382706.Targetdb.orders` o

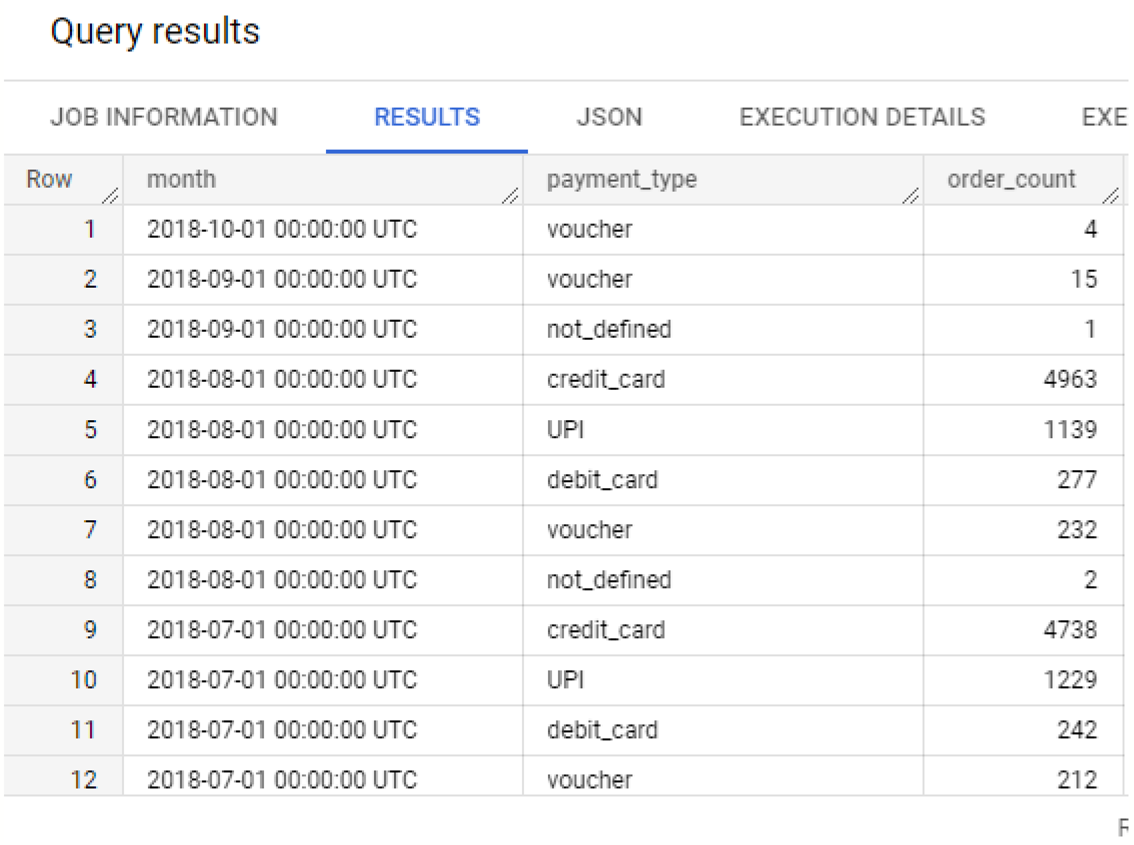
JOIN `scaler-382706.Targetdb.payments` p ON o.order_id = p.order_id

GROUP BY

1, 2

ORDER BY

1 DESC, 3 DESC;



Row	month	payment_type	order_count
1	2018-10-01 00:00:00 UTC	voucher	4
2	2018-09-01 00:00:00 UTC	voucher	15
3	2018-09-01 00:00:00 UTC	not_defined	1
4	2018-08-01 00:00:00 UTC	credit_card	4963
5	2018-08-01 00:00:00 UTC	UPI	1139
6	2018-08-01 00:00:00 UTC	debit_card	277
7	2018-08-01 00:00:00 UTC	voucher	232
8	2018-08-01 00:00:00 UTC	not_defined	2
9	2018-07-01 00:00:00 UTC	credit_card	4738
10	2018-07-01 00:00:00 UTC	UPI	1229
11	2018-07-01 00:00:00 UTC	debit_card	242
12	2018-07-01 00:00:00 UTC	voucher	212

*** Above one is not a complete output it is a sample of output ***

Key conclusions derived from the data and comments:

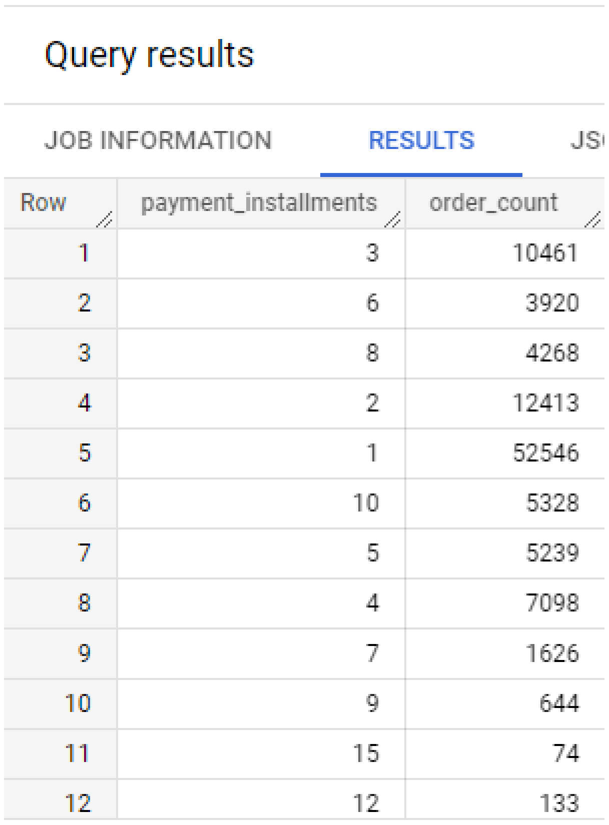
We can spot trends and patterns in the utilization of various payment forms over time using the findings of this investigation. For instance, we could discover that particular payment methods are more common during specific months of the year (such as credit cards during the Christmas season) or that their popularity is rising or falling over time.

In addition, we may get user feedback and utilize it to guide our analysis. For instance, if users report problems with a certain payment method (for instance, lengthy boleto bancario processing times), we may look into the data to see if there are any trends or patterns that might be causing these problems and take action to fix them.

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

Query - SELECT p.payment_installments, COUNT(o.order_id) as order_count

FROM `scaler-382706.Targetdb.orders` o
INNER JOIN `scaler-382706.Targetdb.payments` p ON o.order_id = p.order_id
GROUP BY p.payment_installments;



Row	payment_installments	order_count
1	3	10461
2	6	3920
3	8	4268
4	2	12413
5	1	52546
6	10	5328
7	5	5239
8	4	7098
9	7	1626
10	9	644
11	15	74
12	12	133

*** Above one is not a complete output it is a sample of output ***

Key conclusions derived from the data and comments:

The bulk of orders (about 80%) are paid for in one lump sum.

When the number of payment installments rises, the number of orders falls.

The number of payments and the total order value could be correlated in some way. Customers that place larger orders, for instance, may be more inclined to select a payment strategy with many payments.

Understanding consumer preferences and demands with relation to payment alternatives may be possible through feedback from clients. Customers may, for instance, want to pay in full at the time of purchase to avoid interest fees or may value the ease of having additional payment installment choices.

Refernces

- **Stack Over Flow**
- **Geeks for Geeks**
- **Scaler Topics**
- **W3School**
- **Hacker Rank**
- **Digital Commerce 360**