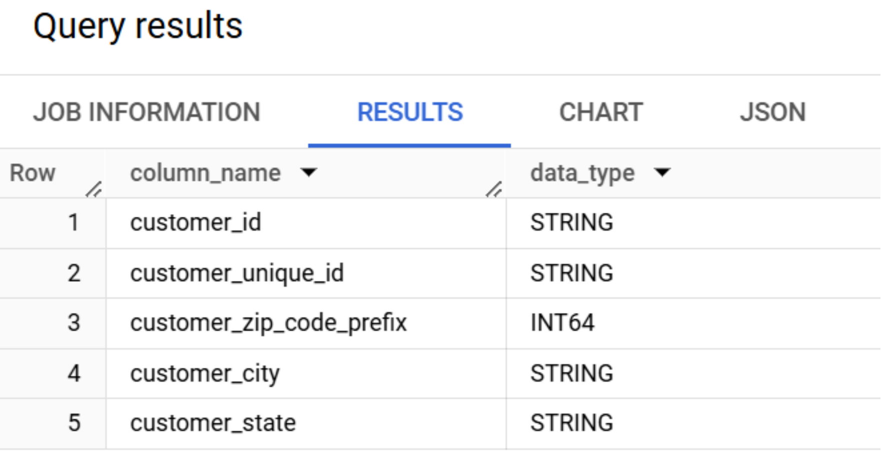
Business Case: Target SQL

1. **Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset:**
    a. Data type of all columns in the "customers" table.

Query:-

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

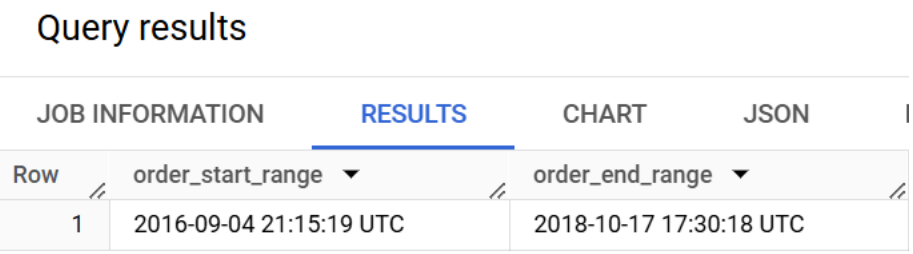


Row	column_name	data_type
1	customer_id	STRING
2	customer_unique_id	STRING
3	customer_zip_code_prefix	INT64
4	customer_city	STRING
5	customer_state	STRING

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

Query:-

```
SELECT min(order_purchase_timestamp) order_start_range,
max(order_purchase_timestamp) order_end_range FROM `target.orders`;
```

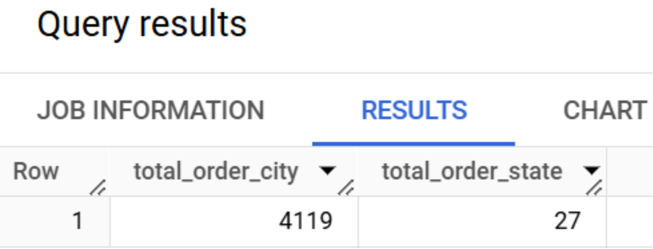


Row	order_start_range	order_end_range
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC

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

Query:-

```
SELECT count(distinct customer_city) total_order_city, count(distinct
customer_state) total_order_state
FROM `target.customers`;
```



Row	total_order_city	total_order_state
1	4119	27

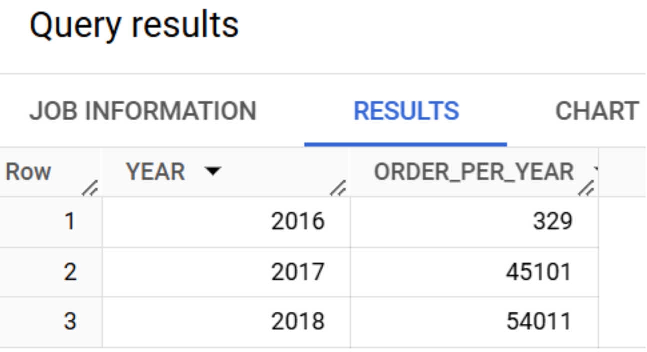
2. In-depth Exploration:

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

Yes, based on the query, year-on-year there is increase in the no. of orders.

Query:-

```
SELECT EXTRACT(YEAR FROM ORDER_PURCHASE_TIMESTAMP) AS YEAR,
COUNT(1) ORDER_PER_YEAR
FROM `target.orders`
GROUP BY YEAR
ORDER BY YEAR;
```



Row	YEAR	ORDER_PER_YEAR
1	2016	329
2	2017	45101
3	2018	54011

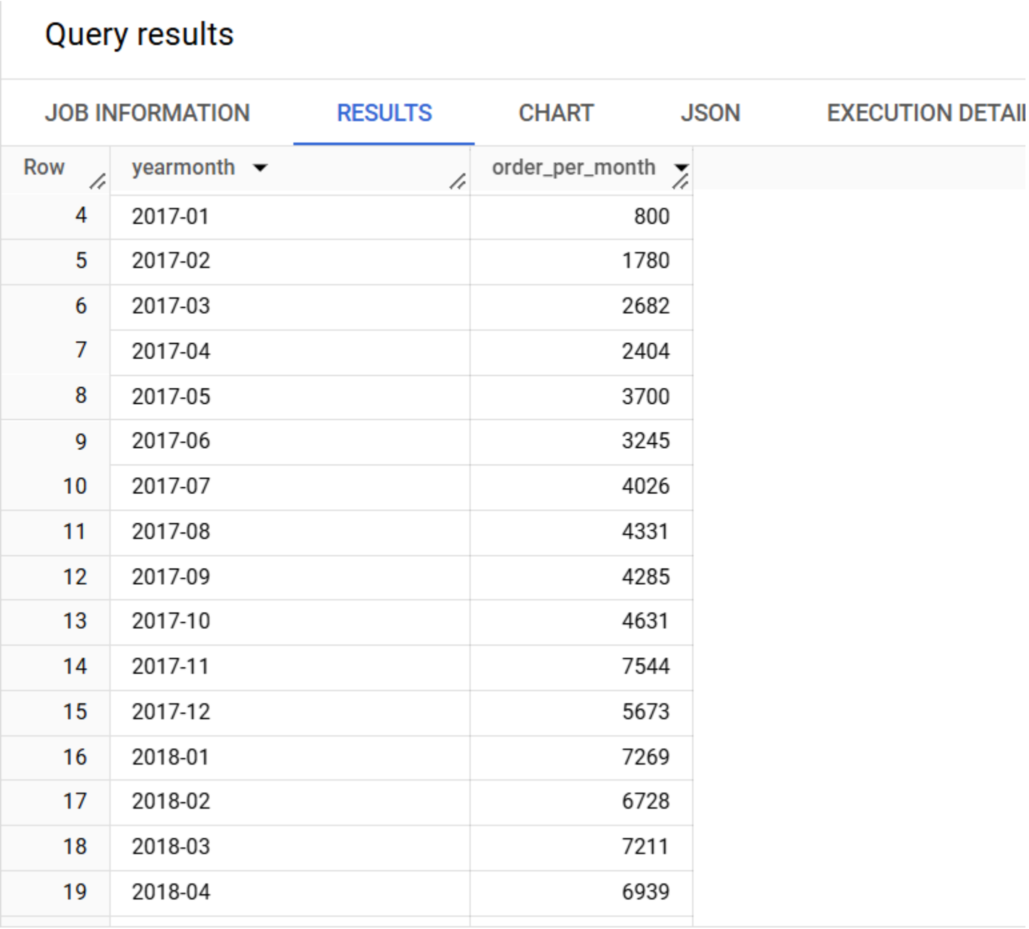
b. Can we see some kind of monthly seasonality in terms of the no. of orders being placed?

Here we are trying to find order pattern as per season.

- In Brazil, November to March is considered as warm months and we can see rise in orders during these months.

Query:-

```
SELECT yearmonth, order_per_month
from (SELECT format_datetime('%Y-%m', order_purchase_timestamp)
yearmonth,  count(*) order_per_month
FROM `target.orders`
GROUP BY yearmonth) tbl
ORDER BY yearmonth
```



Row	yearmonth	order_per_month
4	2017-01	800
5	2017-02	1780
6	2017-03	2682
7	2017-04	2404
8	2017-05	3700
9	2017-06	3245
10	2017-07	4026
11	2017-08	4331
12	2017-09	4285
13	2017-10	4631
14	2017-11	7544
15	2017-12	5673
16	2018-01	7269
17	2018-02	6728
18	2018-03	7211
19	2018-04	6939

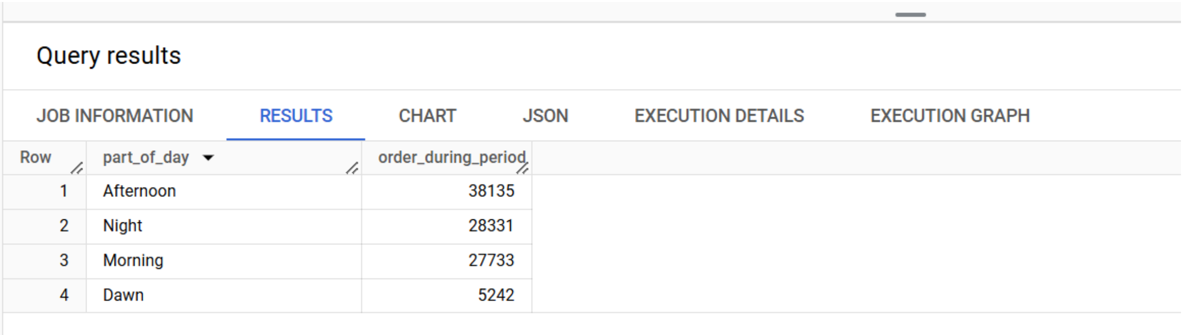
Created by **Vikas Jain**

c. During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)
   i. 0-6 hrs : Dawn
   ii. 7-12 hrs : Mornings
   iii. 13-18 hrs : Afternoon
   iv. 19-23 hrs : Night

- Brazilian customers mostly place their order during Afternoon, followed by Night, Morning and Dawn.

Query:-

```
SELECT CASE WHEN num_hour >= 0 and num_hour <=6 THEN 'Dawn'
    WHEN num_hour > 6 and num_hour <= 12 THEN 'Morning'
    WHEN num_hour > 12 and num_hour <=18 THEN 'Afternoon'
    WHEN num_hour > 18 and num_hour <= 23 THEN 'Night'
    END as part_of_day, sum(order_per_hour) order_during_period
FROM (
SELECT CAST(FORMAT_DATETIME('%H', order_purchase_timestamp) as NUMERIC)
as num_hour, count(*) order_per_hour
FROM `target.orders`
GROUP BY num_hour) tbl
GROUP BY part_of_day
ORDER BY order_during_period desc
```



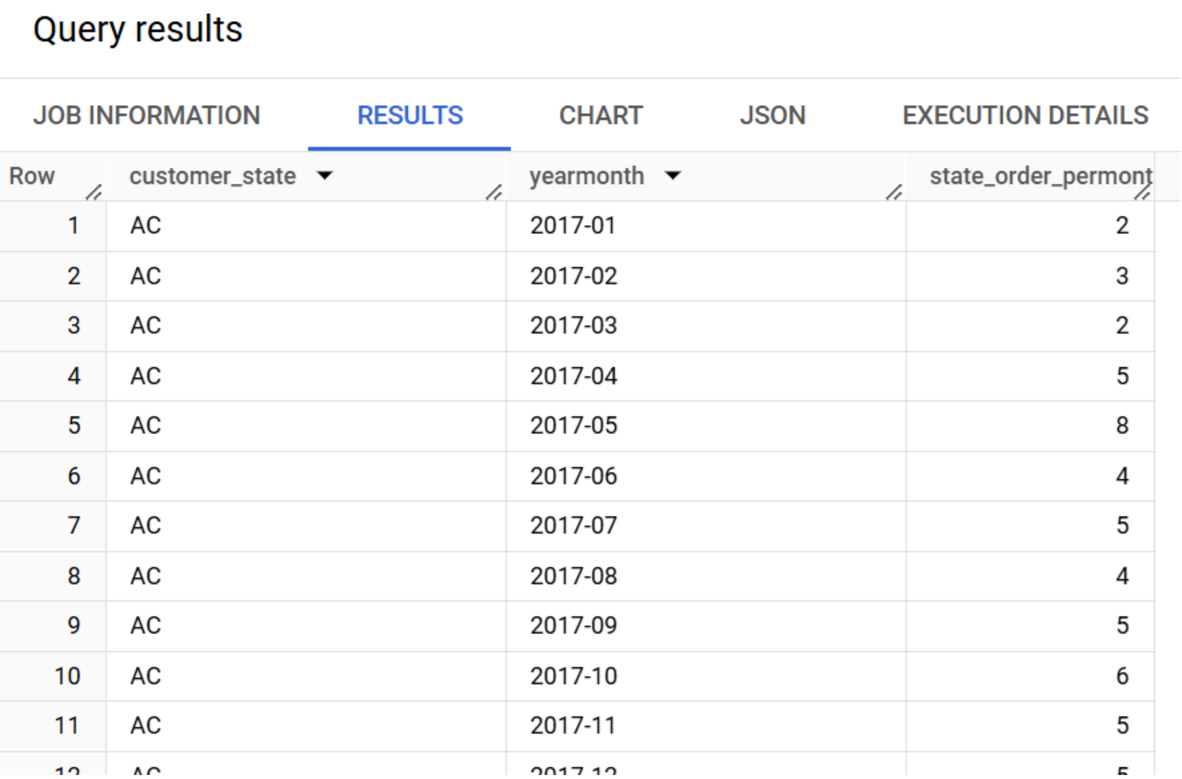
Row	part_of_day	order_during_period
1	Afternoon	38135
2	Night	28331
3	Morning	27733
4	Dawn	5242

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

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

Query:-

```
SELECT c.customer_state, format_datetime('%Y-%m',
o.order_purchase_timestamp) yearmonth, count(1) state_order_permonth
FROM `target.customers` c INNER JOIN `target.orders` o
ON c.customer_id = o.customer_id
GROUP BY customer_state, yearmonth
ORDER BY customer_state, yearmonth
```



Query results

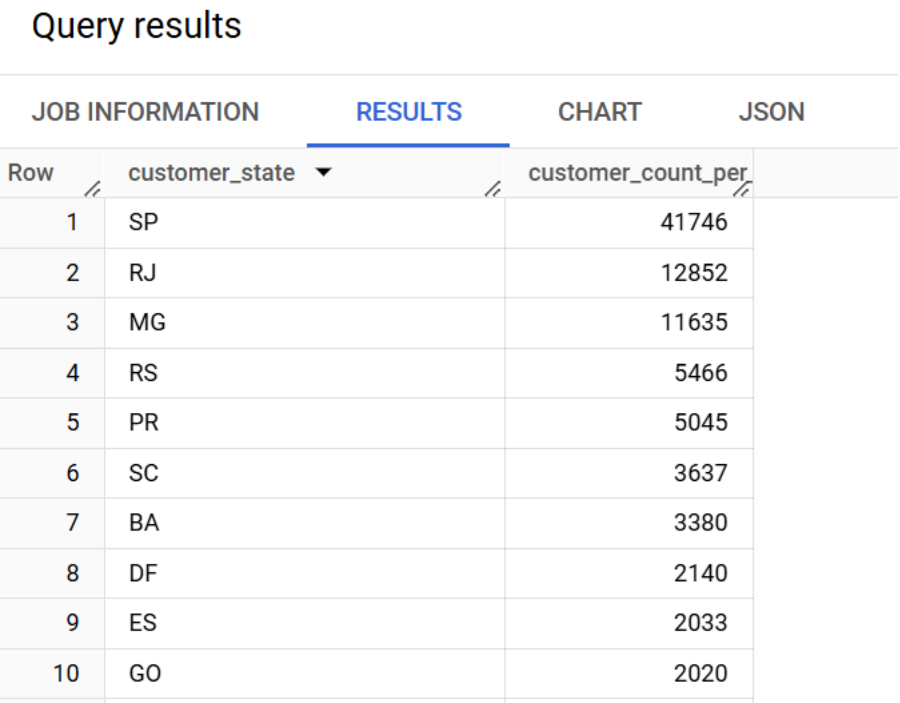
Row	customer_state	yearmonth	state_order_permont
1	AC	2017-01	2
2	AC	2017-02	3
3	AC	2017-03	2
4	AC	2017-04	5
5	AC	2017-05	8
6	AC	2017-06	4
7	AC	2017-07	5
8	AC	2017-08	4
9	AC	2017-09	5
10	AC	2017-10	6
11	AC	2017-11	5
12	AC	2017-12	5

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

- Most of the customer's are from SP, RJ & MG state.
- As customer_unique_id represents unique id for each customer, have taken that to identify the count of customer per state.

Query:-

```
SELECT customer_state, count(customer_unique_id) customer_count_per_state
FROM `target.customers`
GROUP BY customer_state
ORDER BY customer_count_per_state desc
```



Query results

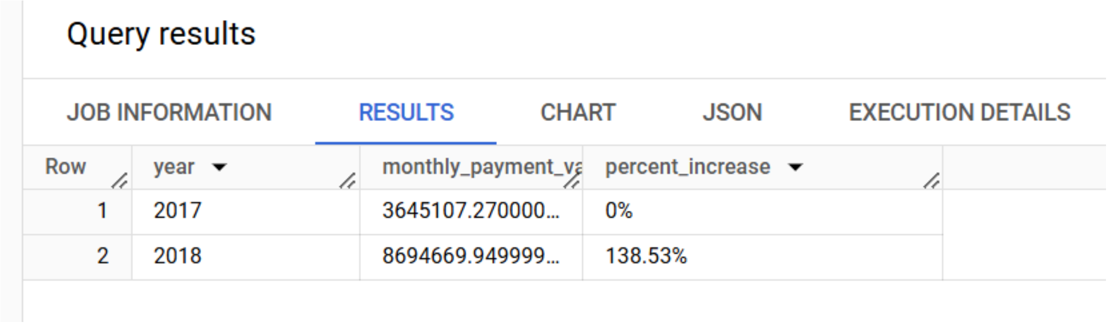
Row	customer_state	customer_count_per
1	SP	41746
2	RJ	12852
3	MG	11635
4	RS	5466
5	PR	5045
6	SC	3637
7	BA	3380
8	DF	2140
9	ES	2033
10	GO	2020

4. **Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight and others.**
    a. Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only).
    You can use the "payment_value" column in the payments table to get the cost of orders.

There is **138.53 %** increase in payment value from 2017 to 2018 (b/w Jan to Aug only).

Query :-

```
SELECT year, monthly_payment_value,
concat(round(ifnull((monthly_payment_value - lag(monthly_payment_value) over(order by
year asc))*100/lag(monthly_payment_value) over(order by year asc),0), 2), '%')
percent_increase
FROM (select format_datetime('%Y', o.order_purchase_timestamp) year,
sum(p.payment_value) monthly_payment_value,
from `target.orders` o join `target.payments` p
on o.order_id = p.order_id
where order_purchase_timestamp between '2017-01-01' and '2017-08-31' or
order_purchase_timestamp between '2018-01-01' and '2018-08-31'
group by year) tbl
ORDER BY year
```

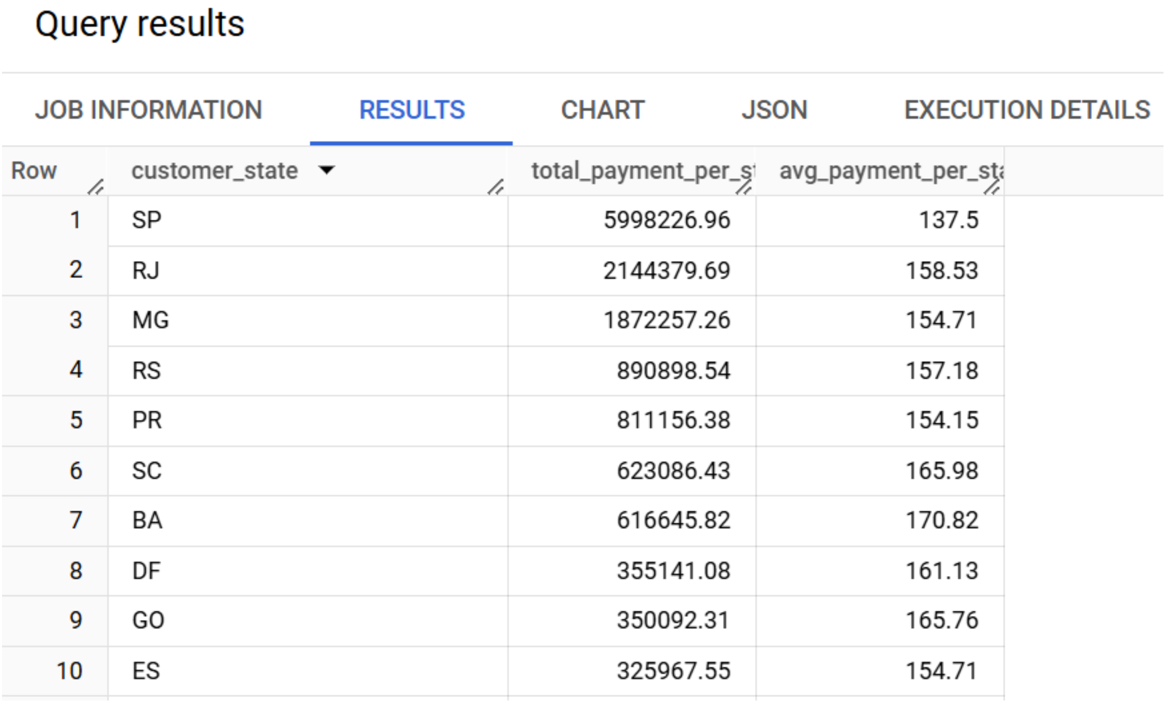


Row	year	monthly_payment_va	percent_increase
1	2017	3645107.270000...	0%
2	2018	8694669.949999...	138.53%

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

Query:-

```
SELECT c.customer_state,
round(sum(p.payment_value),2) total_payment_per_state,
round(avg(p.payment_value),2) avg_payment_per_state
FROM `target.customers` c inner join `target.orders` o
on c.customer_id = o.customer_id
inner join `target.payments` p
on o.order_id = p.order_id
GROUP BY c.customer_state
ORDER BY total_payment_per_state desc
```

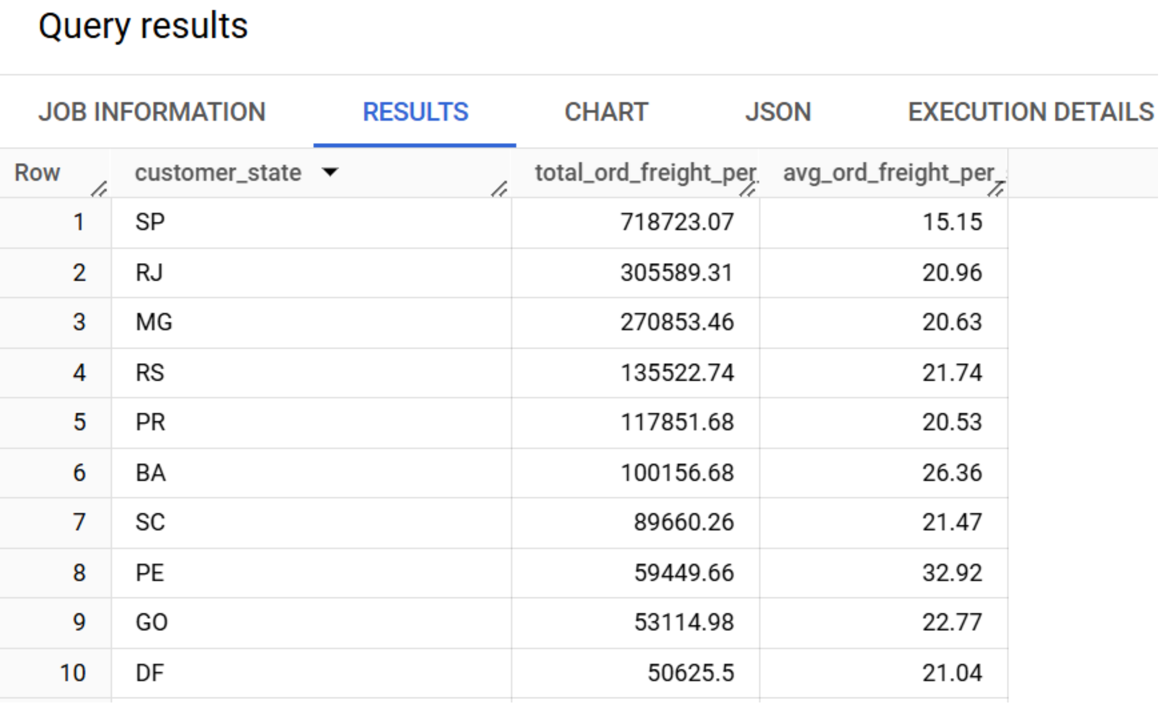


Row	customer_state	total_payment_per_s	avg_payment_per_st
1	SP	5998226.96	137.5
2	RJ	2144379.69	158.53
3	MG	1872257.26	154.71
4	RS	890898.54	157.18
5	PR	811156.38	154.15
6	SC	623086.43	165.98
7	BA	616645.82	170.82
8	DF	355141.08	161.13
9	GO	350092.31	165.76
10	ES	325967.55	154.71

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

Query:-

```
SELECT c.customer_state,
round(sum(oi.freight_value),2) total_ord_freight_per_state,
round(avg(oi.freight_value),2) avg_ord_freight_per_state
FROM `target.customers` c inner join `target.orders` o
on c.customer_id = o.customer_id
inner join `target.order_items` oi
on o.order_id = oi.order_id
GROUP BY c.customer_state
ORDER BY total_ord_freight_per_state desc
```



Query results

Row	customer_state	total_ord_freight_per	avg_ord_freight_per_
1	SP	718723.07	15.15
2	RJ	305589.31	20.96
3	MG	270853.46	20.63
4	RS	135522.74	21.74
5	PR	117851.68	20.53
6	BA	100156.68	26.36
7	SC	89660.26	21.47
8	PE	59449.66	32.92
9	GO	53114.98	22.77
10	DF	50625.5	21.04

5. **Analysis based 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.

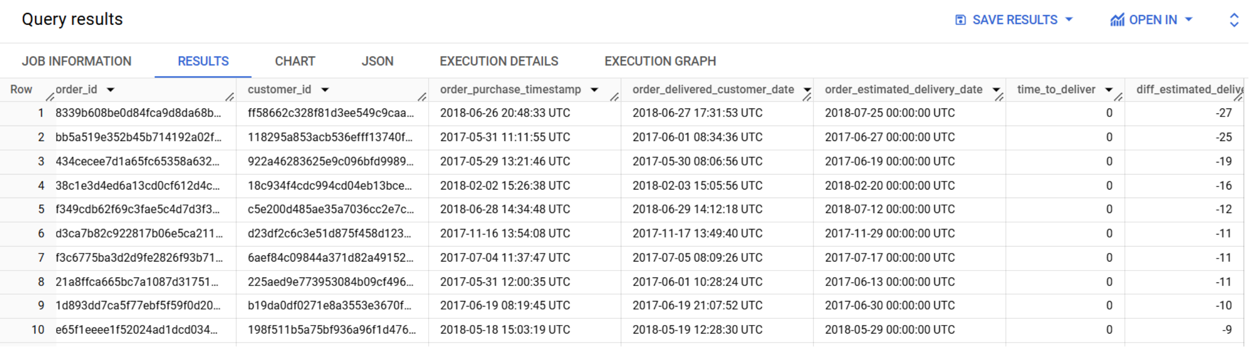
You can calculate the delivery time and the difference between the estimated & actual delivery date using the given formula:

i. **time_to_deliver** = order_delivered_customer_date - order_purchase_timestamp

ii. **diff_estimated_delivery** = order_delivered_customer_date - order_estimated_delivery_date

Query:-

```
SELECT order_id, customer_id, order_purchase_timestamp,
order_delivered_customer_date,  order_estimated_delivery_date,
date_diff(order_delivered_customer_date,order_purchase_timestamp, Day) as
time_to_deliver,
date_diff(order_delivered_customer_date, order_estimated_delivery_date, Day) as
diff_estimated_delivery
FROM `target.orders`
WHERE order_delivered_customer_date is not null
ORDER BY time_to_deliver, diff_estimated_delivery
```



Row	order_id	customer_id	order_purchase_timestamp	order_delivered_customer_date	order_estimated_delivery_date	time_to_deliver	diff_estimated_deliv
1	8339b608be0d84fca9d8da68b...	ff58662c328f81d3ee549c9caa...	2018-06-26 20:48:33 UTC	2018-06-27 17:31:53 UTC	2018-07-25 00:00:00 UTC	0	-27
2	bb5a519e352b45b714192a02f...	118295a853acb536efff13740f...	2017-05-31 11:11:55 UTC	2017-06-01 08:34:36 UTC	2017-06-27 00:00:00 UTC	0	-25
3	434cecee7d1a65fc65358a632...	922a46283625e9c096bfd9989...	2017-05-29 13:21:46 UTC	2017-05-30 08:06:56 UTC	2017-06-19 00:00:00 UTC	0	-19
4	38c1e3d4ed6a13cd0cf612d4c...	18c934f4cdc994cd04eb13bce...	2018-02-02 15:26:38 UTC	2018-02-03 15:05:56 UTC	2018-02-20 00:00:00 UTC	0	-16
5	f349cdb62f69c3fae5c4d7d3f3...	c5e200d485ae35a7036cc2e7c...	2018-06-28 14:34:48 UTC	2018-06-29 14:12:18 UTC	2018-07-12 00:00:00 UTC	0	-12
6	d3ca7b82c922817b06e5ca211...	d23df2c6c3e51d875f458d123...	2017-11-16 13:54:08 UTC	2017-11-17 13:49:40 UTC	2017-11-29 00:00:00 UTC	0	-11
7	f3c6775ba3d2d9fe2826f93b71...	6aef84c09844a371d82a49152...	2017-07-04 11:37:47 UTC	2017-07-05 08:09:26 UTC	2017-07-17 00:00:00 UTC	0	-11
8	21a8ffca665bc7a1087d31751...	225aed9e773953084b09cf496...	2017-05-31 12:00:35 UTC	2017-06-01 10:28:24 UTC	2017-06-13 00:00:00 UTC	0	-11
9	1d893dd7ca5f77ebf5f59f0d20...	b19da0df0271e8a3553e3670f...	2017-06-19 08:19:45 UTC	2017-06-19 21:07:52 UTC	2017-06-30 00:00:00 UTC	0	-10
10	e65f1eeee1f52024ad1dcd034...	198f511b5a75bf936a96f1d476...	2018-05-18 15:03:19 UTC	2018-05-19 12:28:30 UTC	2018-05-29 00:00:00 UTC	0	-9

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

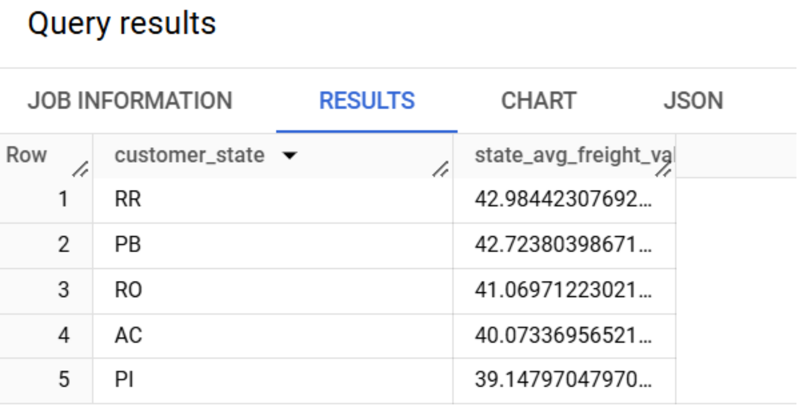
As it is not clear if we need to perform both these operation in same query or different query, hence have provided 2 different solutions.

Solution 1 (listing in different query) :-

Top 5 states with highest average freight value: -

Query:-

```
SELECT c.customer_state, avg(oi.freight_value) state_avg_freight_value
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 customer_state
ORDER BY state_avg_freight_value desc
LIMIT 5
```



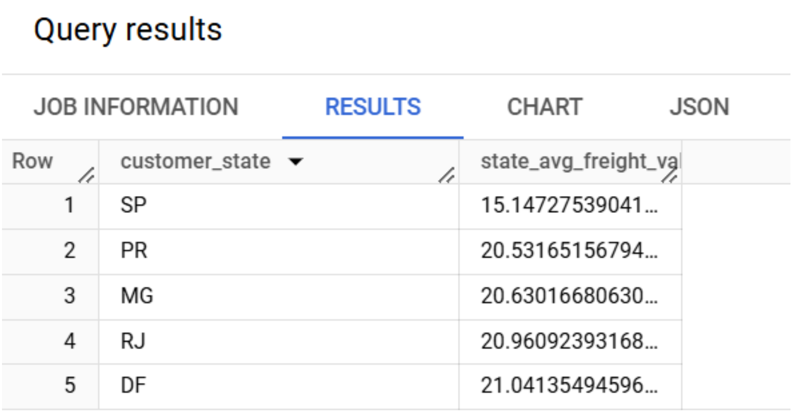
Query results

Row	customer_state	state_avg_freight_val
1	RR	42.98442307692...
2	PB	42.72380398671...
3	RO	41.06971223021...
4	AC	40.07336956521...
5	PI	39.14797047970...

Top 5 states with lowest average freight value: -

Query:-

```
SELECT c.customer_state, avg(oi.freight_value) state_avg_freight_value
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 customer_state
ORDER BY state_avg_freight_value
LIMIT 5
```



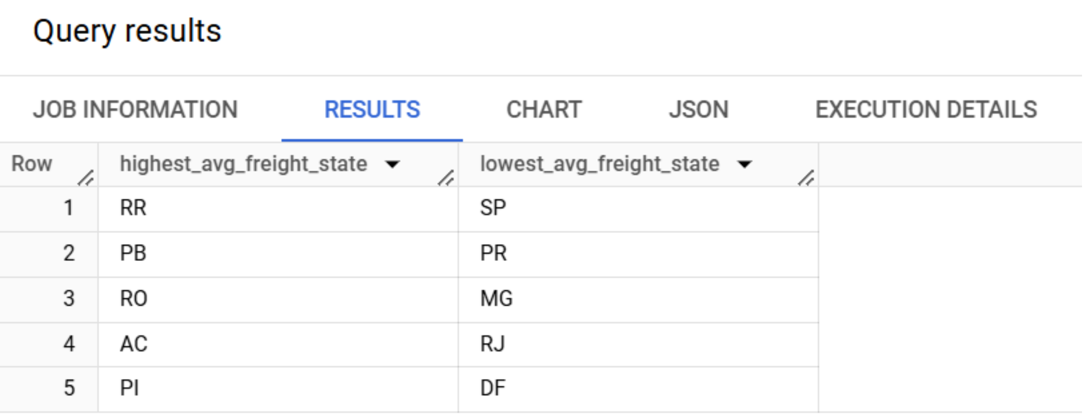
Row	customer_state	state_avg_freight_val
1	SP	15.14727539041...
2	PR	20.53165156794...
3	MG	20.63016680630...
4	RJ	20.96092393168...
5	DF	21.04135494596...

Solution 2 (listing in same query)

Query: -

```
With cte AS
(SELECT c.customer_state, avg(oi.freight_value) state_avg_freight_value,
row_number() over(order by avg(oi.freight_value) desc)
highest_freight_state_num,
row_number() over(order by avg(oi.freight_value) asc)
lowest_freight_state_num
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 customer_state
ORDER BY state_avg_freight_value)

SELECT c1.customer_state as highest_avg_freight_state, c2.customer_state
as lowest_avg_freight_state
from cte c1 inner join cte c2
on c1.highest_freight_state_num = c2.lowest_freight_state_num
where c1.highest_freight_state_num <= 5
order by c1.highest_freight_state_num;
```



Row	highest_avg_freight_state	lowest_avg_freight_state
1	RR	SP
2	PB	PR
3	RO	MG
4	AC	RJ
5	PI	DF

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

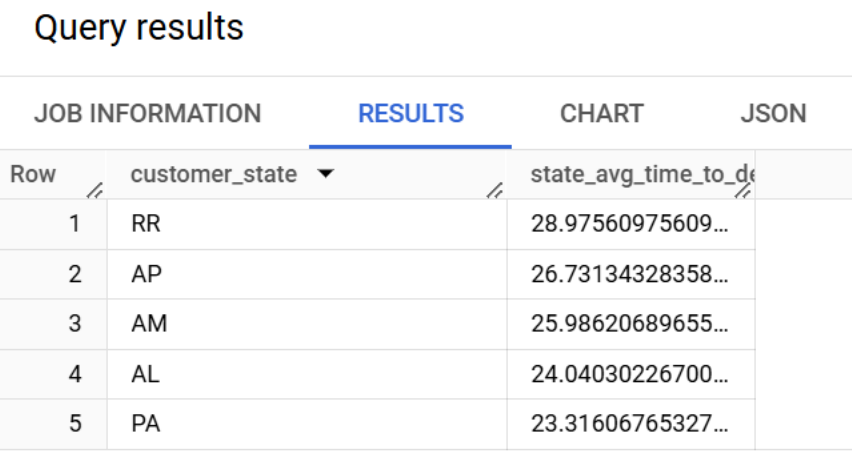
As it is not clear if we need to perform both these operation in same query or different query, hence have provided 2 different solutions. Also, to derive appropriate info, have considered order_status as delivered.

Solution 1 (listing in different query) :-

Top 5 states with highest average delivery time: -

Query:-

```
SELECT c.customer_state,
avg(date_diff(o.order_delivered_customer_date,o.order_purchase_timestamp,
Day)) as state_avg_time_to_deliver
FROM `target.customers` c join `target.orders` o
on c.customer_id = o.customer_id
WHERE order_status = 'delivered'
GROUP BY c.customer_state
ORDER BY state_avg_time_to_deliver DESC
LIMIT 5
```



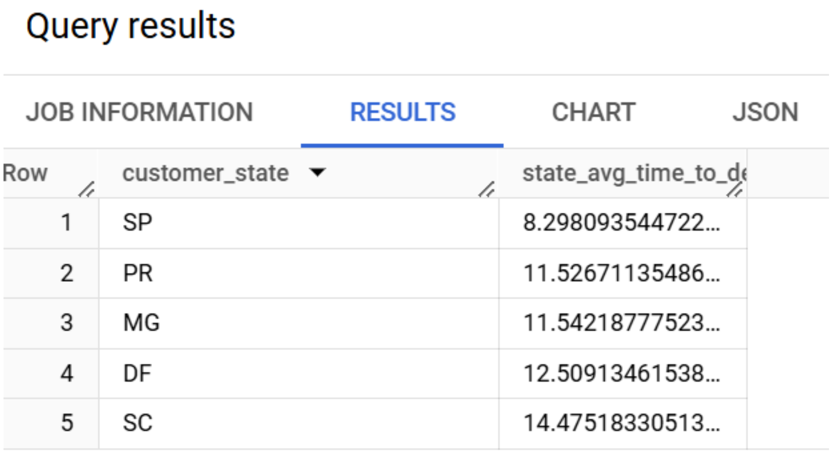
Query results

Row	customer_state	state_avg_time_to_de
1	RR	28.97560975609...
2	AP	26.73134328358...
3	AM	25.98620689655...
4	AL	24.04030226700...
5	PA	23.31606765327...

Top 5 states with lowest average delivery time: -

Query:-

```
SELECT c.customer_state,
avg(date_diff(o.order_delivered_customer_date,o.order_purchase_timestamp,
Day)) as state_avg_time_to_deliver
FROM `target.customers` c join `target.orders` o
on c.customer_id = o.customer_id
WHERE order_status = 'delivered'
GROUP BY c.customer_state
ORDER BY state_avg_time_to_deliver
LIMIT 5
```



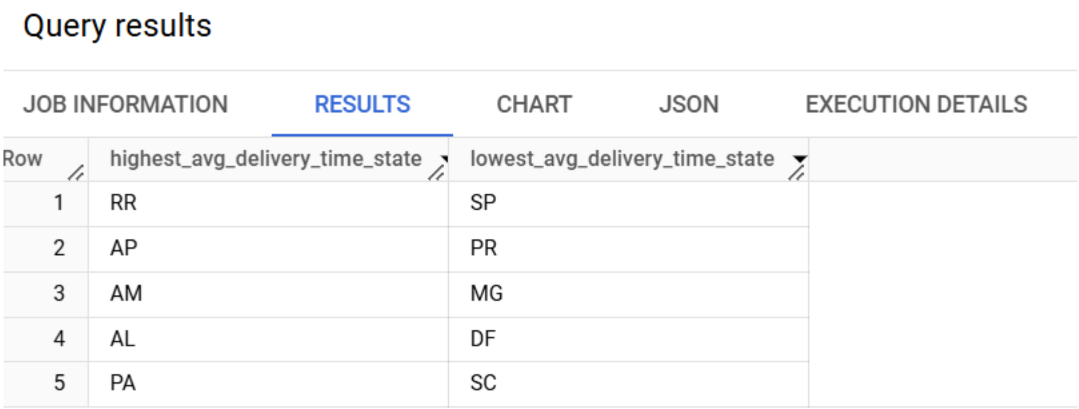
Query results

Row	customer_state	state_avg_time_to_de
1	SP	8.298093544722...
2	PR	11.52671135486...
3	MG	11.54218777523...
4	DF	12.50913461538...
5	SC	14.47518330513...

Solution 2 (listing in same query):-

```
With cte AS
(SELECT c.customer_state,
avg(date_diff(o.order_delivered_customer_date,o.order_purchase_timestamp,Day)
) as state_avg_time_to_deliver,
row_number() over(order by
avg(date_diff(o.order_delivered_customer_date,o.order_purchase_timestamp,Day)
) desc) highest_avg_delivery_time_num,
row_number() over(order by
avg(date_diff(o.order_delivered_customer_date,o.order_purchase_timestamp,Day)
) asc) lowest_avg_delivery_time_num
FROM `target.customers` c join `target.orders` o
on c.customer_id = o.customer_id
WHERE order_status = 'delivered'
GROUP BY c.customer_state
ORDER BY state_avg_time_to_deliver)

SELECT c1.customer_state as highest_avg_delivery_time_state, c2.customer_state
as lowest_avg_delivery_time_state
from cte c1 inner join cte c2
on c1.highest_avg_delivery_time_num = c2.lowest_avg_delivery_time_num
where c1.highest_avg_delivery_time_num <= 5
order by c1.highest_avg_delivery_time_num
```



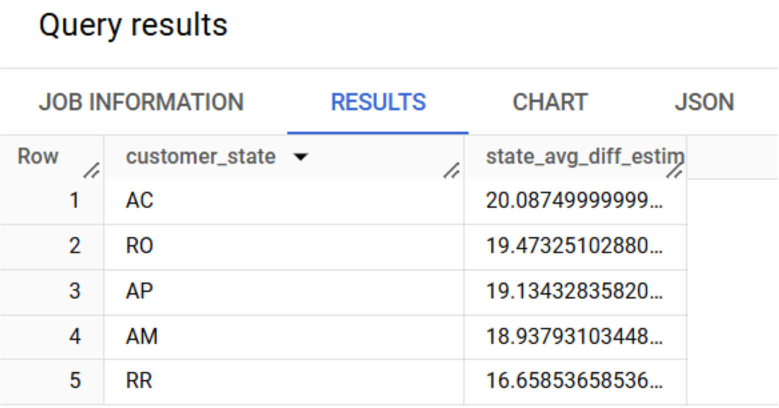
Row	highest_avg_delivery_time_state	lowest_avg_delivery_time_state
1	RR	SP
2	AP	PR
3	AM	MG
4	AL	DF
5	PA	SC

d. Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery.
You can use the difference between the averages of actual & estimated delivery date to figure out how fast the delivery was for each state.

To derive fastest delivery, have subtracted average time to deliver from average estimated delivery time. Positive value indicate product delivered before estimated time and negative indicate after estimated time.

Query:-

```
SELECT customer_state, avg(state_avg_estimated_delivery - state_avg_time_to_deliver) as
state_avg_diff_estimate_to_deliver
FROM (SELECT c.customer_state,
avg(date_diff(o.order_delivered_customer_date,o.order_purchase_timestamp, Day)) as
state_avg_time_to_deliver,
avg(date_diff(o.order_estimated_delivery_date, o.order_purchase_timestamp, Day)) as
state_avg_estimated_delivery
FROM `target.customers` c join `target.orders` o
on c.customer_id = o.customer_id
WHERE o.order_status = 'delivered'
group by c.customer_state) tbl
GROUP BY customer_state
ORDER BY state_avg_diff_estimate_to_deliver desc
LIMIT 5
```

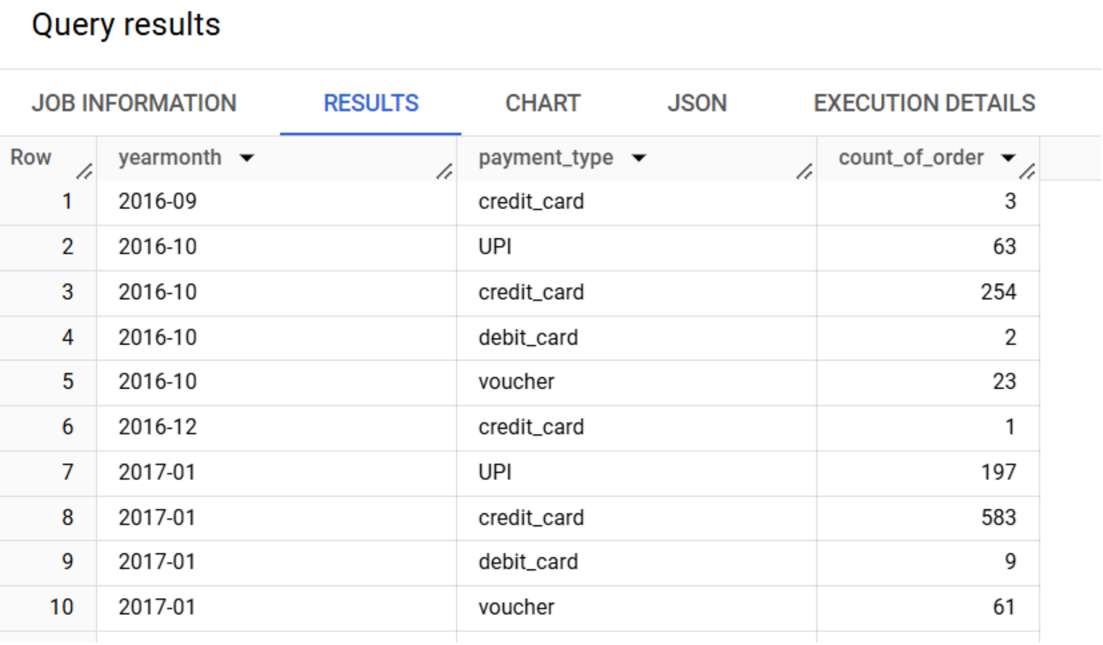


Row	customer_state	state_avg_diff_estim
1	AC	20.08749999999...
2	RO	19.47325102880...
3	AP	19.13432835820...
4	AM	18.93793103448...
5	RR	16.65853658536...

6. **Analysis based on the payments:**
    a. Find the month on month no. of orders placed using different payment types.

Query:-

```
SELECT format_datetime('%Y-%m', o.order_purchase_timestamp)
yearmonth, payment_type, count(p.order_id) count_of_order
FROM `target.orders` o join `target.payments` p
on o.order_id = p.order_id
GROUP BY yearmonth, payment_type
ORDER BY yearmonth, payment_type
```



Query results

Row	yearmonth	payment_type	count_of_order
1	2016-09	credit_card	3
2	2016-10	UPI	63
3	2016-10	credit_card	254
4	2016-10	debit_card	2
5	2016-10	voucher	23
6	2016-12	credit_card	1
7	2017-01	UPI	197
8	2017-01	credit_card	583
9	2017-01	debit_card	9
10	2017-01	voucher	61

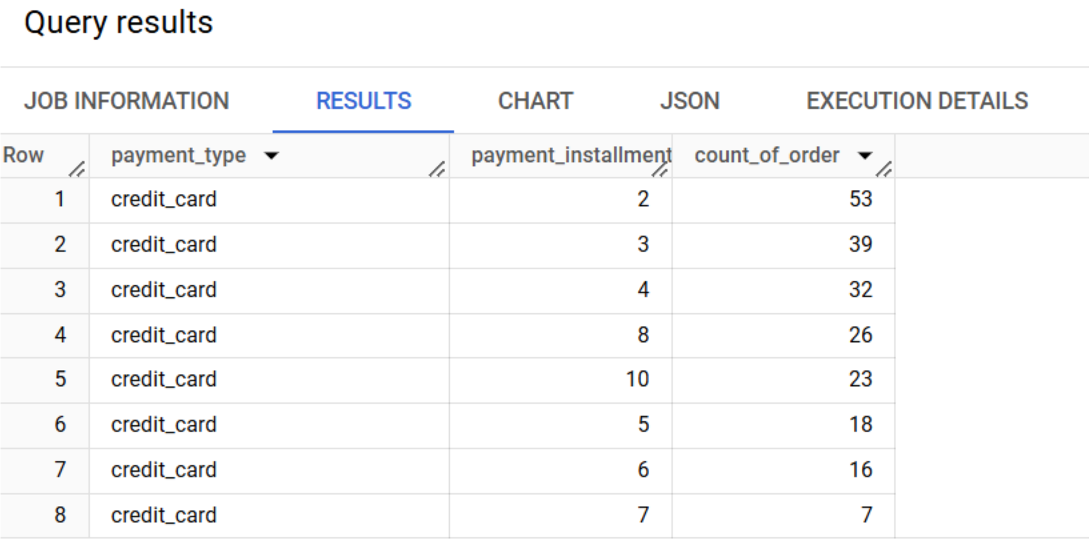
    b. Find the no. of orders placed on the basis of the payment installments that have been paid.

As per table definition, **payment_sequential** represents the sequences of the payments made in case of EMI and **payment_installments** represent the number of installments in case of EMI purchase.

Assuming, if number of installments > 1, then its EMI purchase, otherwise its full value purchase. Hence, added condition in where clause to filter only EMI transactions.

Query:-

```
SELECT p.payment_type, p.payment_installments, count(p.order_id)
count_of_order
FROM `target.orders` o join `target.payments` p
on o.order_id = p.order_id
where payment_sequential > 1 and payment_installments > 1
GROUP BY payment_type, payment_installments
ORDER BY count_of_order desc
```


Query results

Row	payment_type	payment_installment	count_of_order
1	credit_card	2	53
2	credit_card	3	39
3	credit_card	4	32
4	credit_card	8	26
5	credit_card	10	23
6	credit_card	5	18
7	credit_card	6	16
8	credit_card	7	7

Recommendations based on above analysis: -

1. There are less orders during Dawn (0 hrs to 6 hrs). Target can use this time as system downtime for enhancements.
2. During Nov to Mar, number of orders where high in numbers. This time can be utilized of new promotions and offers. This is considered as warm season in Brazil.
3. Target can plan to increase City coverage for 5 states having less than 10 cities currently. This may help to increase the orders from respective states, as most of them are currently having less than 150 orders.
4. Target can rollout promotions to increase customer base in 17 out of 27 states, where current customer count is less than 2000. This may help to increase customer base as well are number of orders.
5. Currently difference between Avg estimated delivery time and Avg Delivery time is quite high (approx. 10 days). Target can analyze this data and provide more appropriate estimated delivery date, as customers may not order in case estimated delivery time is high.
6. Around 50% of transactions are done via credit_card, Target may launch EMI based promotion to attract customers.
7. State PA has 1 seller, whereas approx. 1000 customers, whereas State AL has approx. 400 customers, but no seller. If they increase more sellers in these states, then it can help to reduce avg delivery time, which may in turn increase orders from these states.

8. Target received more than 75% review comments with 4+ ratings, so overall customers are satisfied. If they improve delivery time, then it may help to increase customer base.

Created by **Vikas Jain**