TARGET SQL BUSINESS CASE STUDY

Name: Sukanya Devi B

Problem Statement:

Assuming you are a data analyst/ scientist at Target, you have been assigned the task of analyzing the given dataset to extract valuable insights and provide actionable recommendations.

What does 'good' look like?

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

1) 1. Data type of all columns in the "customers" table.

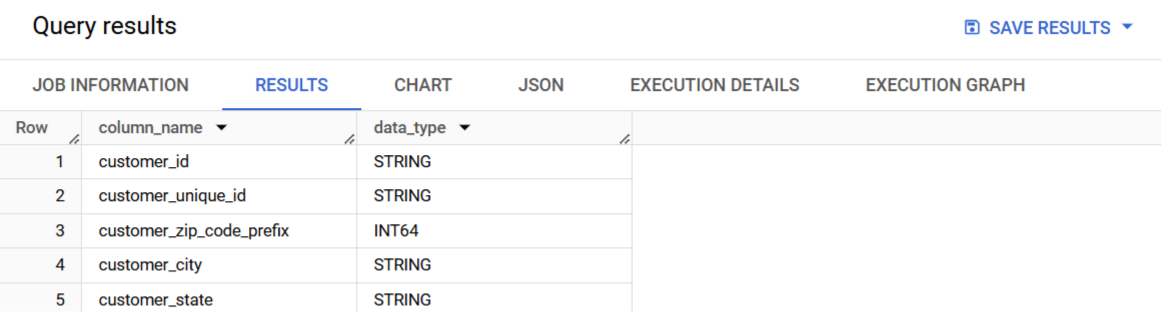
Query:

```
select column_name , data_type

from scaler-dsml-sql-444307.Target.INFORMATION_SCHEMA.COLUMNS

where table_name = 'customers'
```

Output:



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

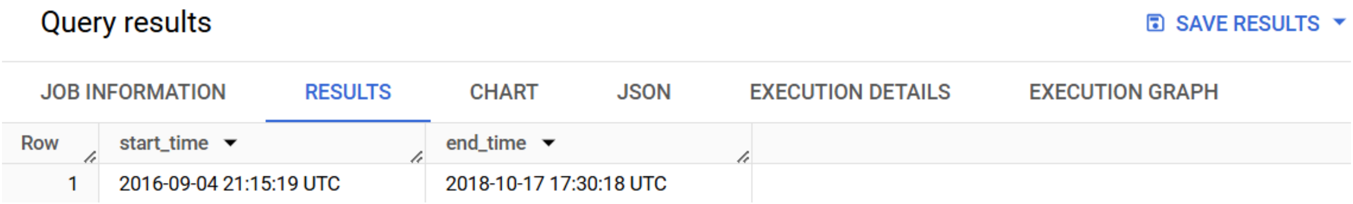
1) 2. Get the time range between which the orders were placed.

Query:

```
select min(order_purchase_timestamp) as start_time, max(order_purchase_timestamp) as end_time

from `scaler-dsml-sql-444307.Target.orders`
```

Output:



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

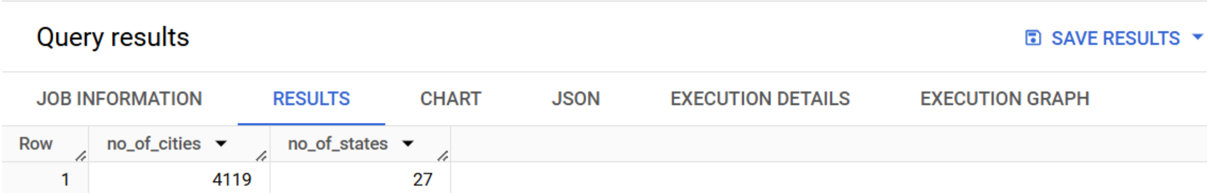
1) 3.Count the Cities & States of customers who ordered during the given period.

Query:

```
select count(distinct customer_city) as no_of_cities, count(distinct customer_state) as no_of_states

from `scaler-dsml-sql-444307.Target.customers`
```

Output:



Row	no_of_cities	no_of_states
1	4119	27

2) In-depth Exploration:

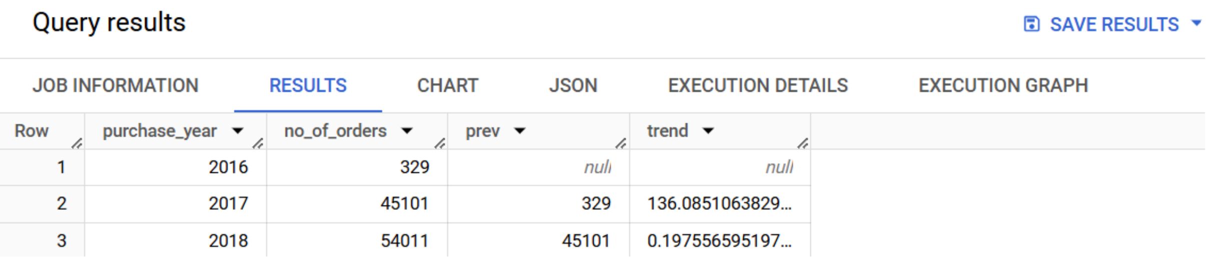
2) 1. Is there a growing trend in the no. of orders placed over the past years?

Query:

```
with final as
(select distinct extract(year from order_purchase_timestamp)as purchase_year,
count(order_id)as no_of_orders
from `scaler-dsml-sql-444307.Target.orders`
group by 1
order by 1),
previous_data as
(select * ,lag(no_of_orders,1)over(order by purchase_year)as prev
from final)

select purchase_year, no_of_orders, prev, ((no_of_orders - prev)/prev)as trend
from previous_data
order by 1
```

Output:



Row	purchase_year	no_of_orders	prev	trend
1	2016	329	*null*	*null*
2	2017	45101	329	136.0851063829...
3	2018	54011	45101	0.197556595197...

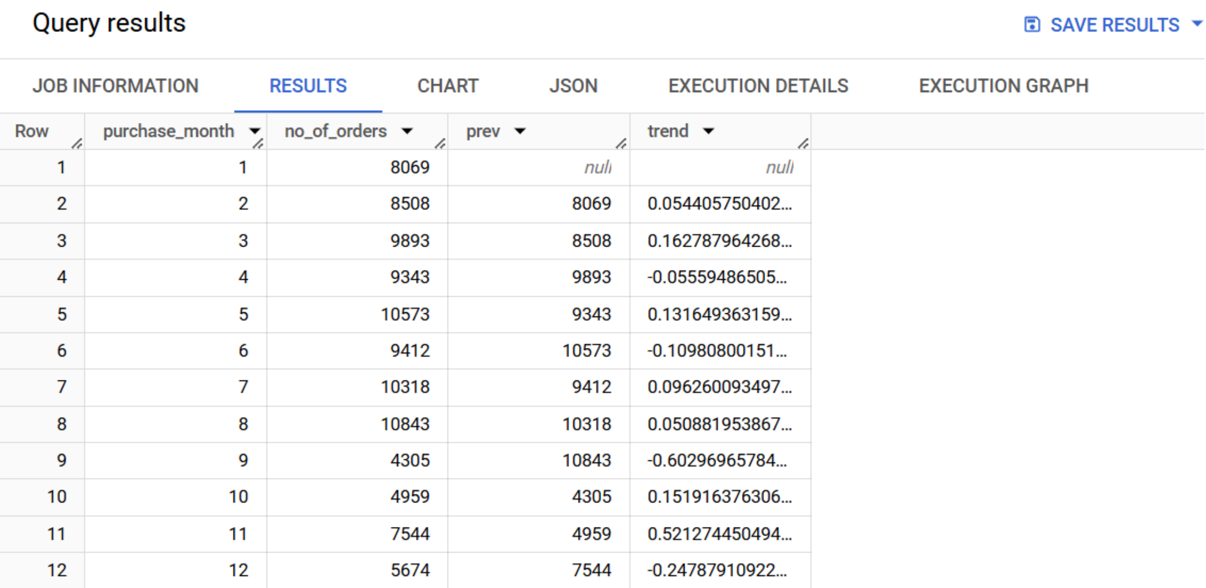
2) 2.Can we see some kind of monthly seasonality in terms of the no. of orders being placed

Query:

```
with final as
(select distinct extract(month from order_purchase_timestamp)as purchase_month,
count(order_id)as no_of_orders
from `scaler-dsml-sql-444307.Target.orders`
group by 1
order by 1),
previous_data as
(select * ,lag(no_of_orders,1)over(order by purchase_month)as prev
from final)

select purchase_month, no_of_orders, prev, ((no_of_orders - prev)/prev)as trend
from previous_data
order by 1
```

Output:



Row	purchase_month	no_of_orders	prev	trend
1	1	8069	*null*	*null*
2	2	8508	8069	0.054405750402...
3	3	9893	8508	0.162787964268...
4	4	9343	9893	-0.05559486505...
5	5	10573	9343	0.131649363159...
6	6	9412	10573	-0.10980800151...
7	7	10318	9412	0.096260093497...
8	8	10843	10318	0.050881953867...
9	9	4305	10843	-0.60296965784...
10	10	4959	4305	0.151916376306...
11	11	7544	4959	0.521274450494...
12	12	5674	7544	-0.24787910922...

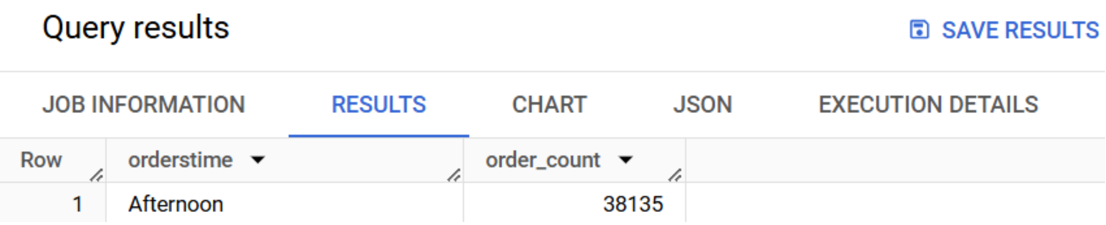
2) 3.During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)

- **0-6 hrs : Dawn**
- **7-12 hrs : Mornings**
- **13-18 hrs : Afternoon**
- **19-23 hrs : Night**

Query:

```
select * from
(select case when extract(hour from order_purchase_timestamp) between 0 and 6
then "Dawn"
when extract(hour from order_purchase_timestamp) between 7 and 12
then "Mornings"
when extract(hour from order_purchase_timestamp) between 13 and
18 then "Afternoon"
when extract(hour from order_purchase_timestamp) between 19 and 23
then "Night"
end as orderstime,
count(distinct order_id) as order_count
from `scaler-dsml-sql-444307.Target.orders`
group by 1
order by 2 desc
limit 1)
```

Output:



Row	orderstime	order_count
1	Afternoon	38135

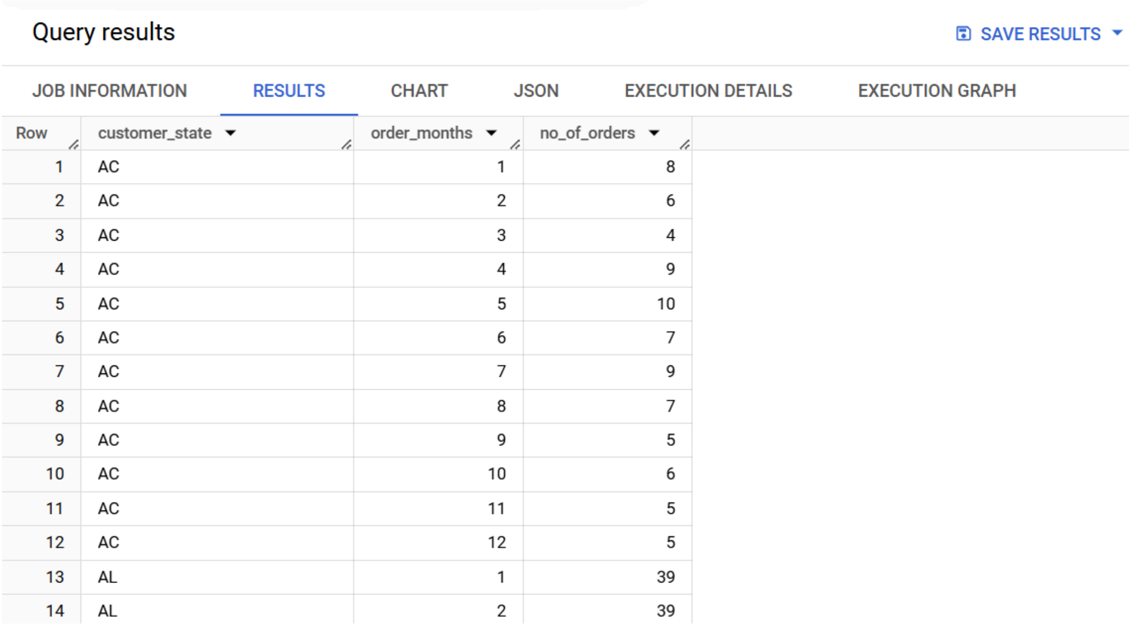
3) Evolution of E-commerce orders in the Brazil region

3) 1.Get the month on month no. of orders placed in each state.

Query:

```
select c.customer_state, extract (month from o.order_purchase_timestamp)as order_months,
 count(distinct o.order_id) as no_of_orders
from `scaler-dsml-sql-444307.Target.orders` o
inner join `scaler-dsml-sql-444307.Target.customers` c
on o.customer_id =c.customer_id
group by 1,2
order by 1,2
```

Output:



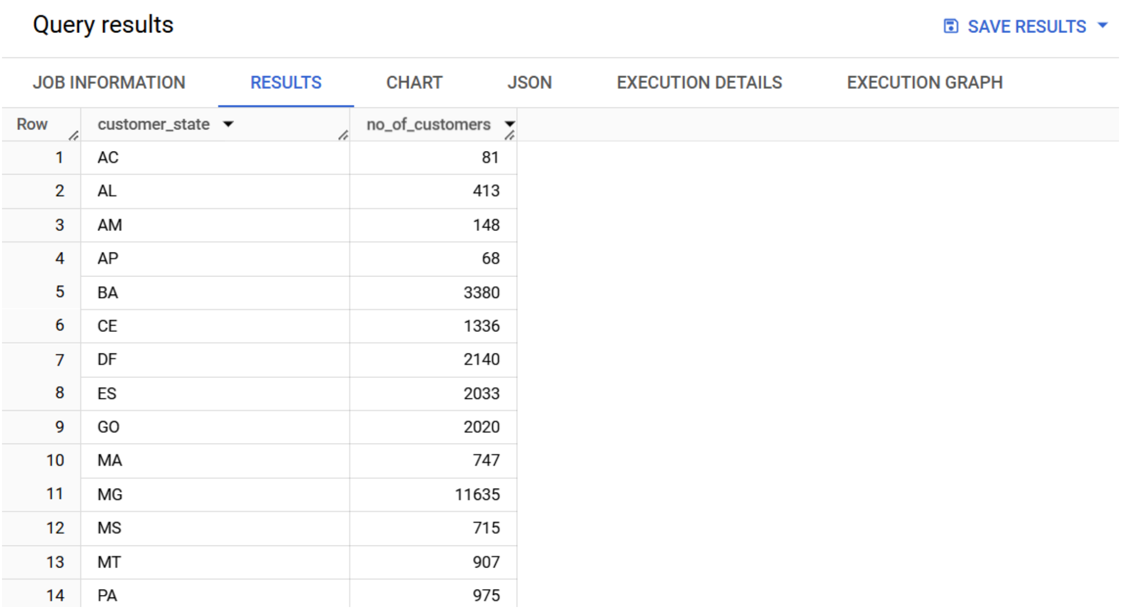
Row	customer_state	order_months	no_of_orders
1	AC	1	8
2	AC	2	6
3	AC	3	4
4	AC	4	9
5	AC	5	10
6	AC	6	7
7	AC	7	9
8	AC	8	7
9	AC	9	5
10	AC	10	6
11	AC	11	5
12	AC	12	5
13	AL	1	39
14	AL	2	39

3) 2.How are the customers distributed across all the states?

Query:

```
select customer_state, count(distinct customer_id)as no_of_customers
from `scaler-dsml-sql-444307.Target.customers`
group by 1
order by 1
```

Output:



Row	customer_state	no_of_customers
1	AC	81
2	AL	413
3	AM	148
4	AP	68
5	BA	3380
6	CE	1336
7	DF	2140
8	ES	2033
9	GO	2020
10	MA	747
11	MG	11635
12	MS	715
13	MT	907
14	PA	975

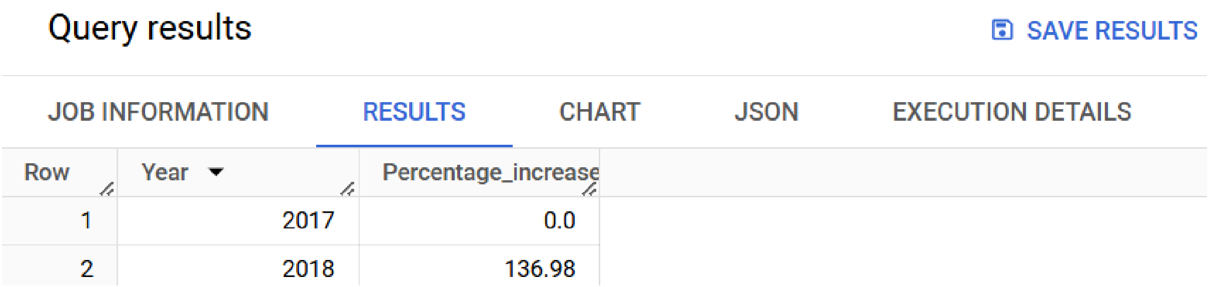
4) Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight and others.

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

Query:

```
WITH time_btw as
(SELECT EXTRACT(YEAR FROM o.order_purchase_timestamp) as Year,
sum(p.payment_value) as cost
FROM `scaler-dsml-sql-444307.Target.orders` AS o
INNER JOIN `scaler-dsml-sql-444307.Target.payments` AS p
ON o.order_id = p.order_id
WHERE EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8 AND
EXTRACT(YEAR FROM o.order_purchase_timestamp) IN (2017,2018)
GROUP BY Year
ORDER BY Year),
lag_btw AS
(SELECT *, LAG(cost) OVER(ORDER BY Year) as lagg
FROM time_btw)
SELECT Year,
ROUND(ifnull(((cost-lagg)/lagg)*100,0),2) as Percentage_increase
FROM lag_btw
ORDER BY Year
```

Output:



Query results

Row	Year	Percentage_increase
1	2017	0.0
2	2018	136.98

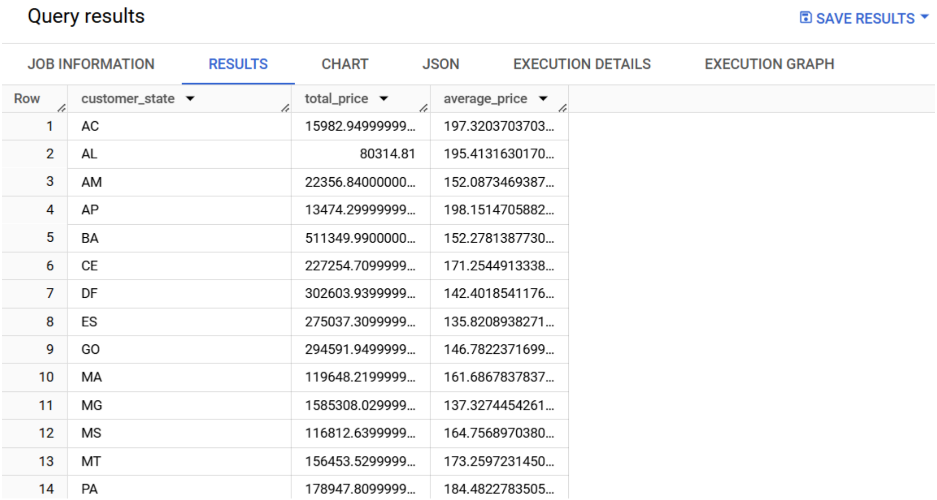
4) 2.Calculate the Total & Average value of order price for each state.

Query:

```
select c.customer_state ,
sum(oi.price)as total_price,
sum(oi.price)/count(distinct(o.order_id))as average_price
from `scaler-dsml-sql-444307.Target.order_items` oi
inner join `scaler-dsml-sql-444307.Target.orders` o
on oi.order_id = o.order_id
inner join `scaler-dsml-sql-444307.Target.customers` c
on o.customer_id = c.customer_id

group by c.customer_state
order by c.customer_state
```

Output:



Query results

Row	customer_state	total_price	average_price
1	AC	15982.94999999...	197.3203703703...
2	AL	80314.81	195.4131630170...
3	AM	22356.84000000...	152.0873469387...
4	AP	13474.29999999...	198.1514705882...
5	BA	511349.9900000...	152.2781387730...
6	CE	227254.7099999...	171.2544913338...
7	DF	302603.9399999...	142.4018541176...
8	ES	275037.3099999...	135.8208938271...
9	GO	294591.9499999...	146.7822371699...
10	MA	119648.2199999...	161.6867837837...
11	MG	1585308.029999...	137.3274454261...
12	MS	116812.6399999...	164.7568970380...
13	MT	156453.5299999...	173.2597231450...
14	PA	178947.8099999...	184.4822783505...

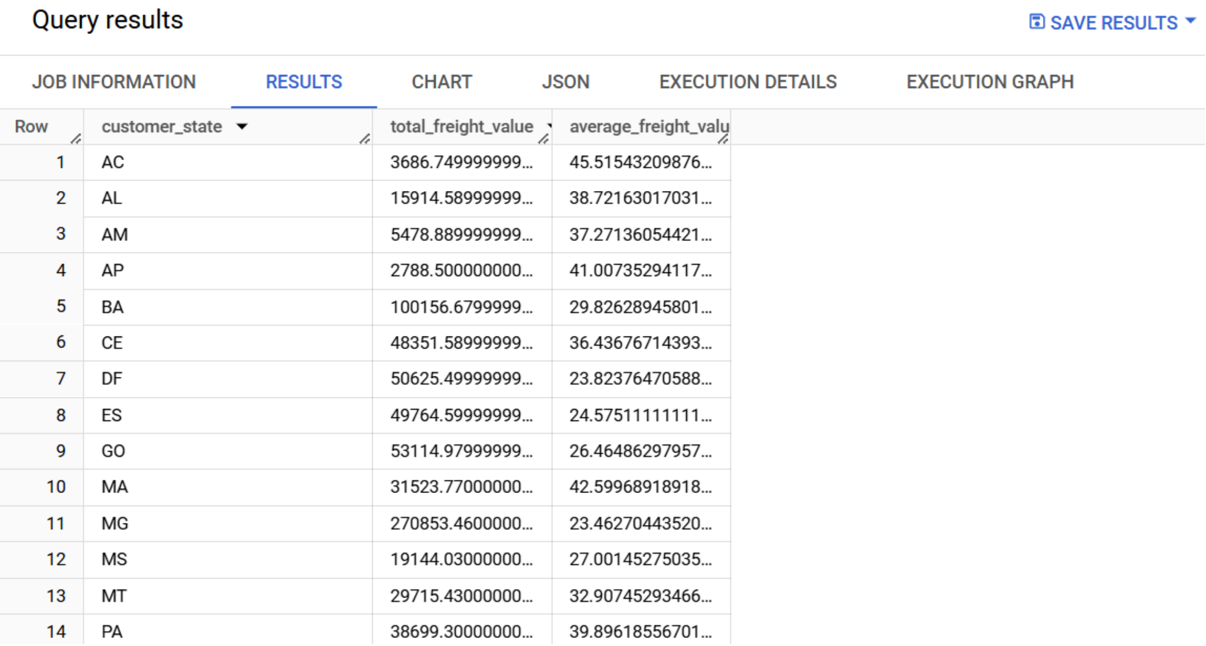
4) 3.Calculate the Total & Average value of order freight for each state.

Query:

```
select c.customer_state ,
sum(oi.freight_value)as total_freight_value,
sum(oi.freight_value)/count(distinct(o.order_id))as average_freight_value
from `scaler-dsml-sql-444307.Target.order_items` oi
inner join `scaler-dsml-sql-444307.Target.orders` o
on oi.order_id = o.order_id
inner join `scaler-dsml-sql-444307.Target.customers` c
on o.customer_id = c.customer_id

group by c.customer_state
order by c.customer_state
```

Output:



Row	customer_state	total_freight_value	average_freight_valu
1	AC	3686.749999999...	45.51543209876...
2	AL	15914.58999999...	38.72163017031...
3	AM	5478.889999999...	37.27136054421...
4	AP	2788.500000000...	41.00735294117...
5	BA	100156.6799999...	29.82628945801...
6	CE	48351.58999999...	36.43676714393...
7	DF	50625.49999999...	23.82376470588...
8	ES	49764.59999999...	24.57511111111...
9	GO	53114.97999999...	26.46486297957...
10	MA	31523.77000000...	42.59968918918...
11	MG	270853.4600000...	23.46270443520...
12	MS	19144.03000000...	27.00145275035...
13	MT	29715.43000000...	32.90745293466...
14	PA	38699.30000000...	39.89618556701...

5) Analysis based on sales, freight and delivery time.

5) 1.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.

Query:

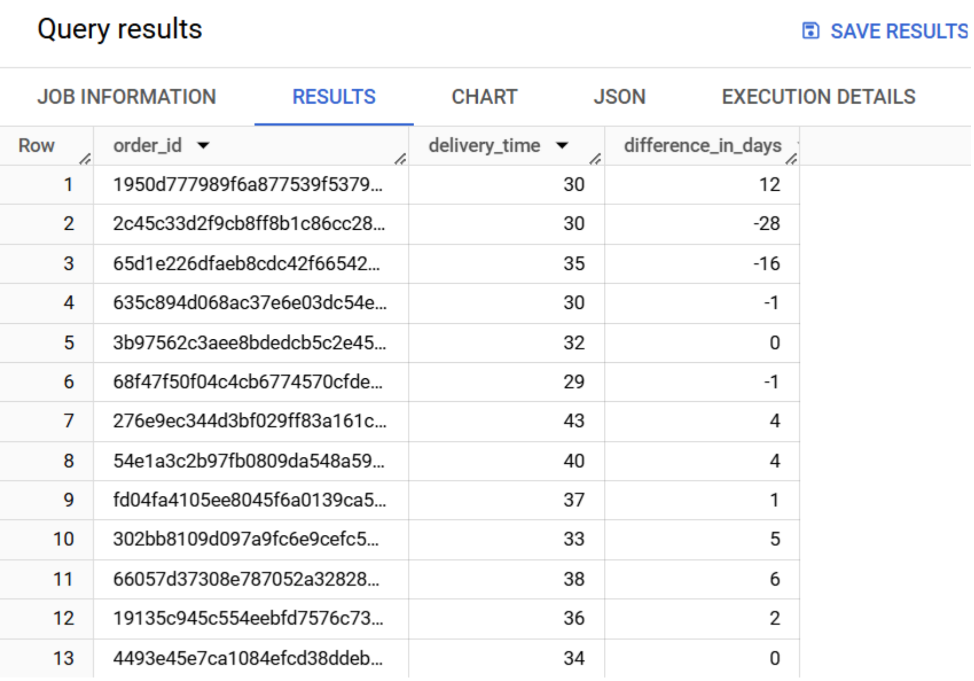
```
select order_id ,

DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp,day)as delivery_time,

date_diff(order_delivered_customer_date,order_estimated_delivery_date,day)as difference_in_days

from `scaler-dsml-sql-444307.Target.orders`
```

Output:



Row	order_id	delivery_time	difference_in_days
1	1950d777989f6a877539f5379...	30	12
2	2c45c33d2f9cb8ff8b1c86cc28...	30	-28
3	65d1e226dfaeb8cdc42f66542...	35	-16
4	635c894d068ac37e6e03dc54e...	30	-1
5	3b97562c3aee8bdedcb5c2e45...	32	0
6	68f47f50f04c4cb6774570cfde...	29	-1
7	276e9ec344d3bf029ff83a161c...	43	4
8	54e1a3c2b97fb0809da548a59...	40	4
9	fd04fa4105ee8045f6a0139ca5...	37	1
10	302bb8109d097a9fc6e9cefc5...	33	5
11	66057d37308e787052a32828...	38	6
12	19135c945c554eebfd7576c73...	36	2
13	4493e45e7ca1084efcd38ddeb...	34	0

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

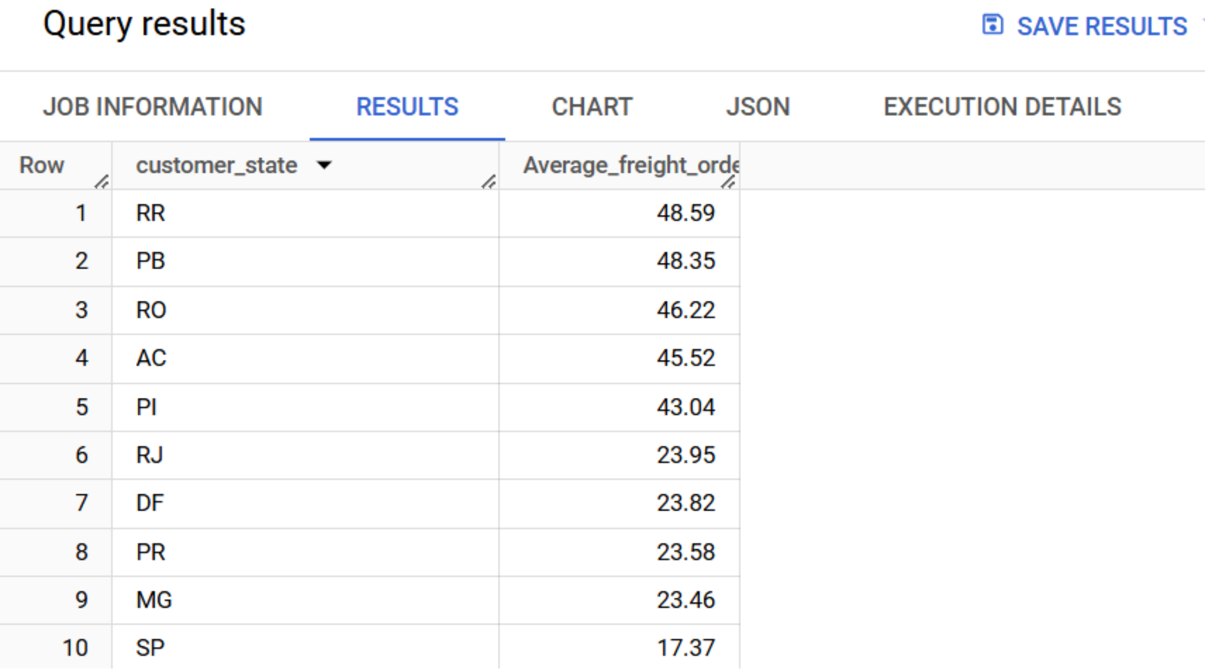
Query:

```
with cte as
(select c.customer_state,
round((sum(i.freight_value)/count(distinct i.order_id)),2) as
Average_freight_order_price
from  `Target.order_items` i
join `Target.orders` o
on o.order_id=i.order_id
join Target.customers c
on c.customer_id=o.customer_id
group by 1
order by 1),

rank1 as
(select *,
dense_rank() over( order by Average_freight_order_price desc) as
highest_drnk,
dense_rank() over( order by Average_freight_order_price asc) as lowest_drnk
from cte)

select customer_state, Average_freight_order_price
from rank1 where lowest_drnk <=5
union all
select customer_state, Average_freight_order_price
from rank1 where highest_drnk <=5
order by Average_freight_order_price desc
```

Output:



Row	customer_state	Average_freight_orde
1	RR	48.59
2	PB	48.35
3	RO	46.22
4	AC	45.52
5	PI	43.04
6	RJ	23.95
7	DF	23.82
8	PR	23.58
9	MG	23.46
10	SP	17.37

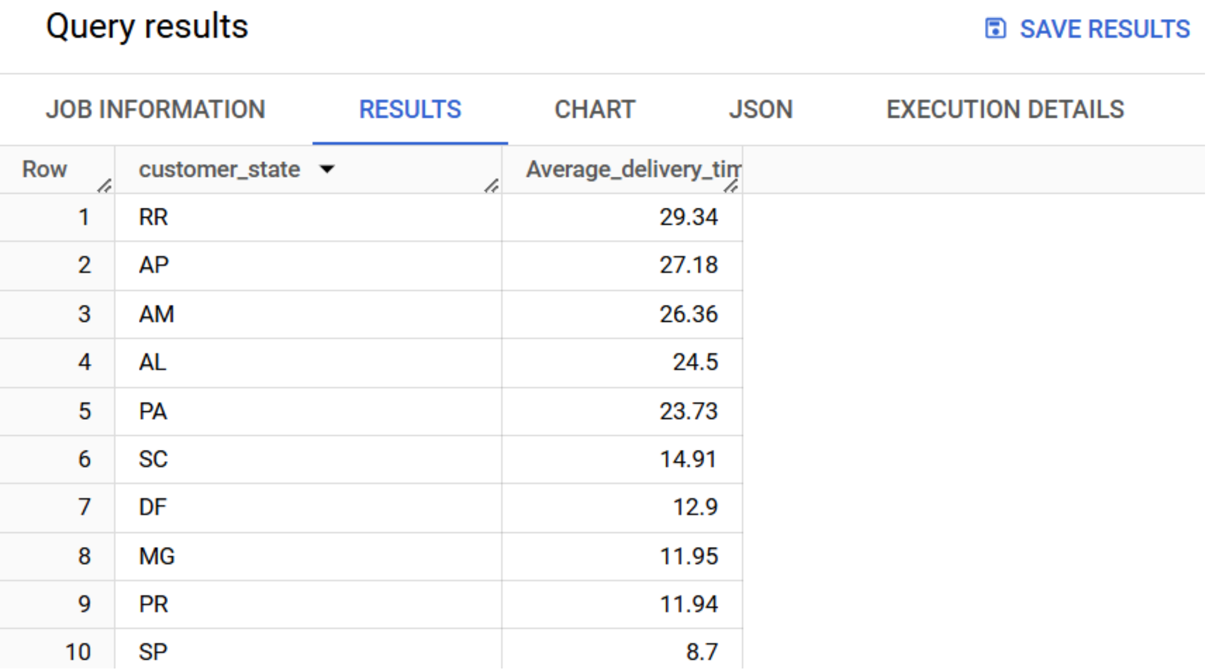
5) 3.Find out the top 5 states with the highest & lowest average delivery time

Query:

```
with AverageDelivertime as
(SELECT
c.customer_state,ROUND(AVG(DATE_DIFF(DATE(o.order_delivered_customer_date),
DATE(o.order_purchase_timestamp),Day)),2) AS Average_delivery_time
from `Target.orders` o
 join `Target.customers` c
  on c.customer_id=o.customer_id
 group by 1
 order by 2),
 rank1 as
 (select customer_state,Average_delivery_time,
        dense_rank() over(order by Average_delivery_time desc) as
Highest_Average_delivery_time,
        dense_rank() over(order by Average_delivery_time asc) as
Lowest_Average_delivery_time
        from AverageDelivertime)
select customer_state,Average_delivery_time
      from rank1
```

```
    where Highest_Average_delivery_time <=5 or
          Lowest_Average_delivery_time <=5
  order by Average_delivery_time desc
```

Output:



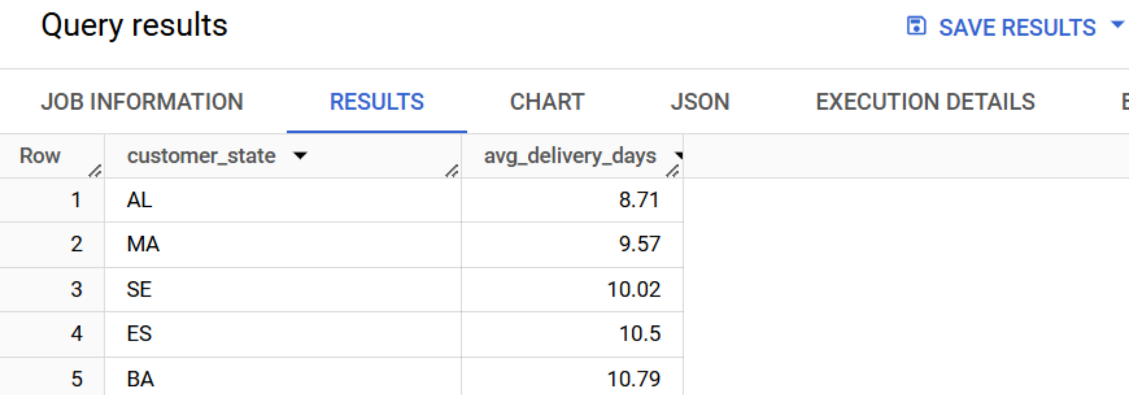
Row	customer_state	Average_delivery_tim
1	RR	29.34
2	AP	27.18
3	AM	26.36
4	AL	24.5
5	PA	23.73
6	SC	14.91
7	DF	12.9
8	MG	11.95
9	PR	11.94
10	SP	8.7

5) 4. 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.

Query:

```
Select
c.customer_state,ROUND(AVG(DATE_DIFF(DATE(order_estimated_delivery_date),DATE(order_delivered_customer_date),Day)),2) AS avg_delivery_days
FROM `scaler-dsml-sql-444307.Target.orders`as o
INNER JOIN `scaler-dsml-sql-444307.Target.customers` AS c
ON o.customer_id = c.customer_id
GROUP BY c.customer_state
ORDER BY avg_delivery_days
LIMIT 5
```

Output:



Row	customer_state	avg_delivery_days
1	AL	8.71
2	MA	9.57
3	SE	10.02
4	ES	10.5
5	BA	10.79

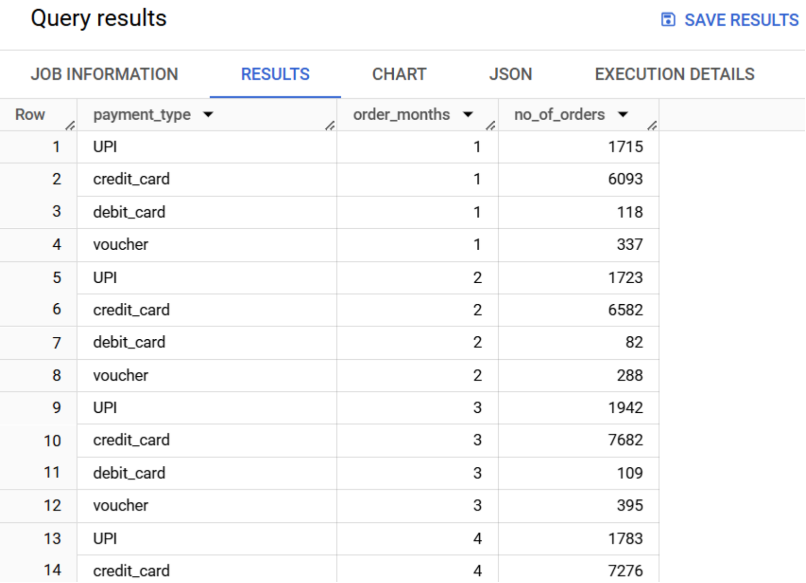
6) Analysis based on the payments

6) 1. Find the month on month no. of orders placed using different payment types.

Query:

```
select p.payment_type, extract (month from o.order_purchase_timestamp)as order_months,
 count(distinct o.order_id) as no_of_orders
from `scaler-dsml-sql-444307.Target.orders` o
inner join `scaler-dsml-sql-444307.Target.payments` p
on o.order_id = p.order_id
group by 1,2
order by 2,1
```

Output:



Row	payment_type	order_months	no_of_orders
1	UPI	1	1715
2	credit_card	1	6093
3	debit_card	1	118
4	voucher	1	337
5	UPI	2	1723
6	credit_card	2	6582
7	debit_card	2	82
8	voucher	2	288
9	UPI	3	1942
10	credit_card	3	7682
11	debit_card	3	109
12	voucher	3	395
13	UPI	4	1783
14	credit_card	4	7276

6) 2. Find the no. of orders placed on the basis of the payment installments that have been paid.

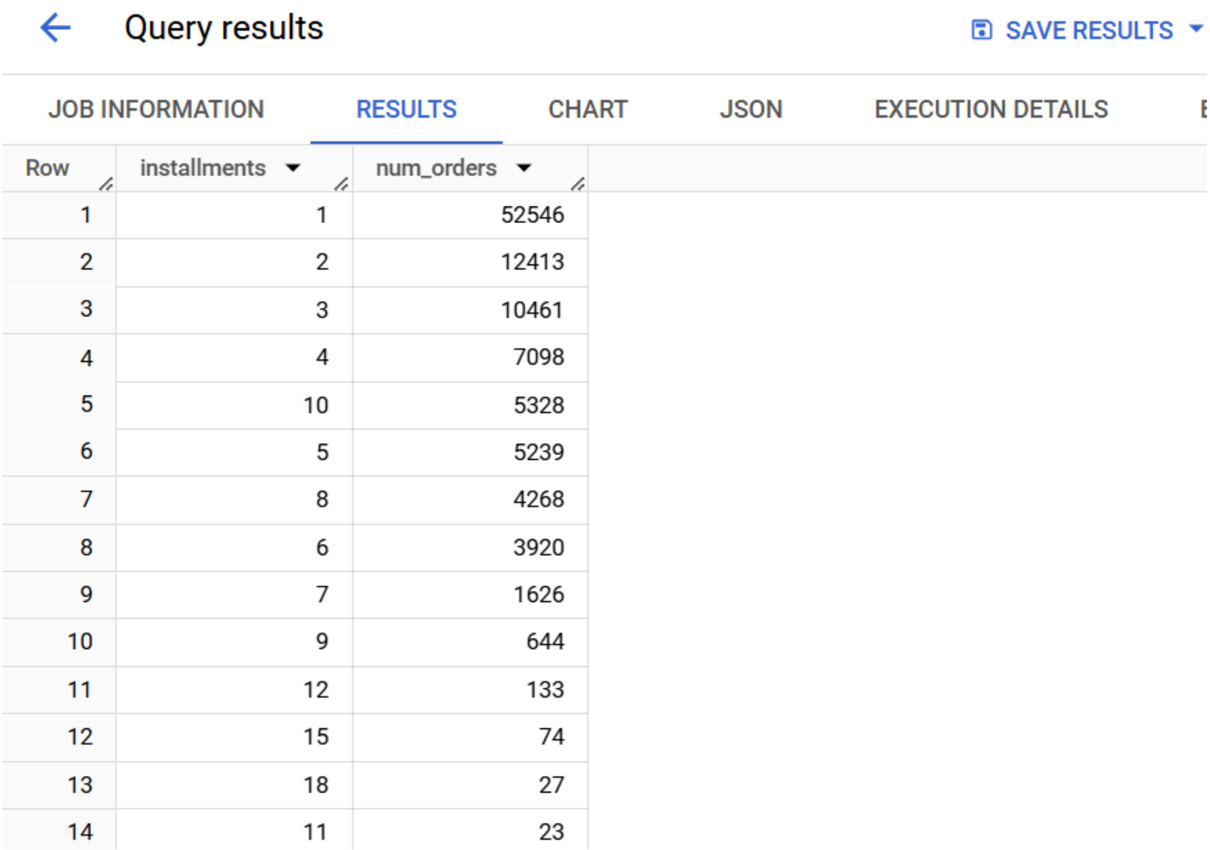
Query:

```
WITH cte_table AS
(SELECT
c.customer_state AS state,
SUM(price) AS total_price,
COUNT(DISTINCT(o.order_id)) AS num_orders
FROM `scaler-dsml-sql-444307.Target.orders` o
INNER JOIN `scaler-dsml-sql-444307.Target.order_items` i
ON o.order_id= i.order_id
INNER JOIN `scaler-dsml-sql-444307.Target.customers` c
ON o.customer_id=c.customer_id
GROUP BY state)

SELECT state,
total_price,
num_orders,
(total_price/num_orders) AS avg_price
FROM cte_table
ORDER BY total_price DESC;


SELECT payment_installments AS installments,
COUNT(order_id) AS num_orders
FROM `scaler-dsml-sql-444307.Target.payments`
WHERE payment_installments >= 1
GROUP BY payment_installments
ORDER BY num_orders DESC;
```

Output:



Row	installments	num_orders
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
11	12	133
12	15	74
13	18	27
14	11	23

Insights & Business recommendations

1. Orders are placed within the time range of 2 years and 1 month (September 4, 2016 - October 17, 2018) and customers from 4119 districts and 27 states have placed their order during the given time period.

2.Growing trend can be seen in no.of.orders placed over the past years and it's fluctuating if we consider the monthly basis. Brazilian customers mostly placed their orders in the Afternoon.

3. The percentage of increase in cost of orders from 2017 to 2018 between Jan to Aug is 136.98%

4. AL, MA, SE, ES & BA are the top 5 states where the delivery is fast compared to estimated delivery date.

Avg delivery time is quite high for most of those states from where the company is receiving quite less volume of orders, detailed study is needed further for checking the other reasons behind such a low volume of orders from the majority of states. Huge delivery time can be one of the reasons and we need to work on it.

5. Also only 3 states contribute to maximum volume, and the rest of the states need to be focused on improving the business.

From the analysis, We can see how the orders trajectory is showing a very abrupt increase in orders volume within a very short time. Looking at the overall trend, it is seen that business is picking up very fast in Brazil so the company has to be ready with an extra workforce. To avoid high risk, it can consider hiring contractual employees.