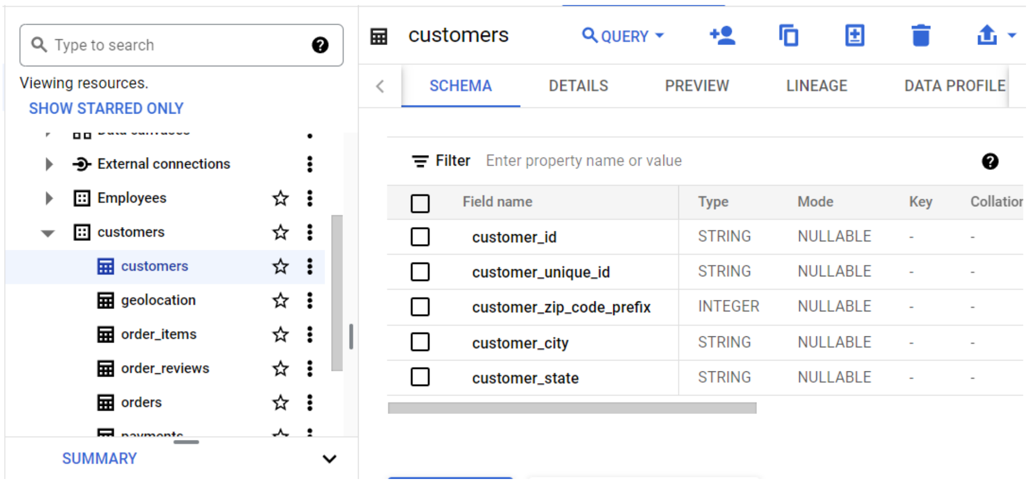
BUSINESS CASE

Hey, I am providing some insights based on the dataset provided to me by Target. And here are my insights -

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

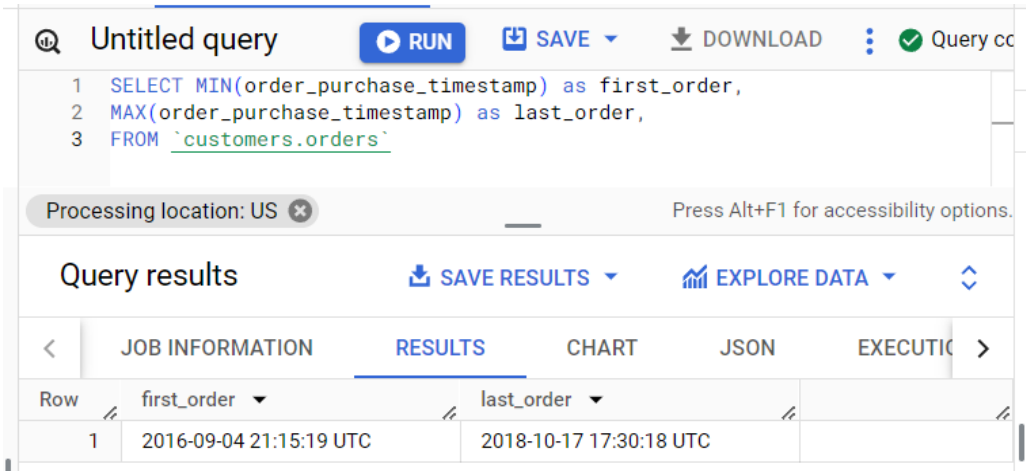
A) Data type of the columns contained in the 'customers' table are –



Field name	Type	Mode	Key	Collation
customer_id	STRING	NULLABLE	-	-
customer_unique_id	STRING	NULLABLE	-	-
customer_zip_code_prefix	INTEGER	NULLABLE	-	-
customer_city	STRING	NULLABLE	-	-
customer_state	STRING	NULLABLE	-	-

B) The time range between which the orders were placed is -

```
SELECT MIN(order_purchase_timestamp) as first_date,
MAX(order_purchase_timestamp) as last_date
FROM `customers.orders`
LIMIT 10
```



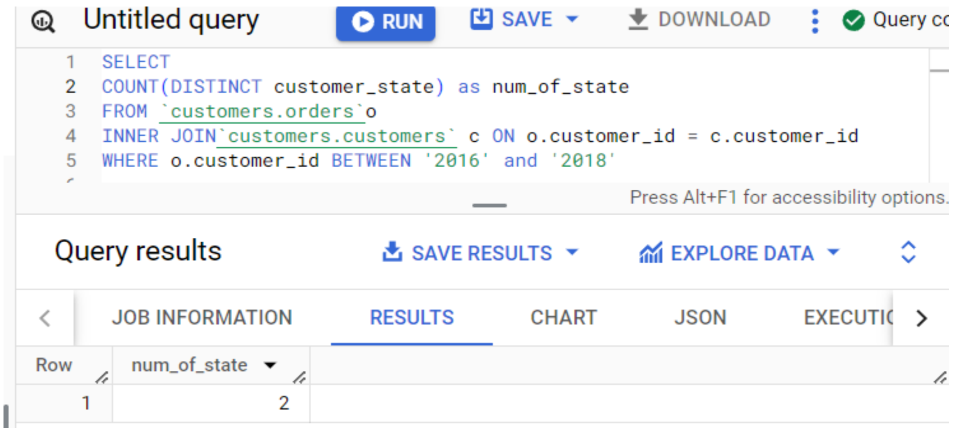
```
SELECT MIN(order_purchase_timestamp) as first_order,
MAX(order_purchase_timestamp) as last_order,
FROM `customers.orders`
```

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

Interpretation – Orders were placed between 2016 and 2018 .

C) There are 2 cities and 2 states of customers who ordered during the given period

```
SELECT COUNT(DISTINCT customer_city) as num_of_city,
COUNT(DISTINCT customer_state) as num_of_state
FROM `customers.orders`o
INNER JOIN`customers.customers` c ON o.customer_id = c.customer_id
WHERE o.customer_id BETWEEN '2016' and '2018'
```



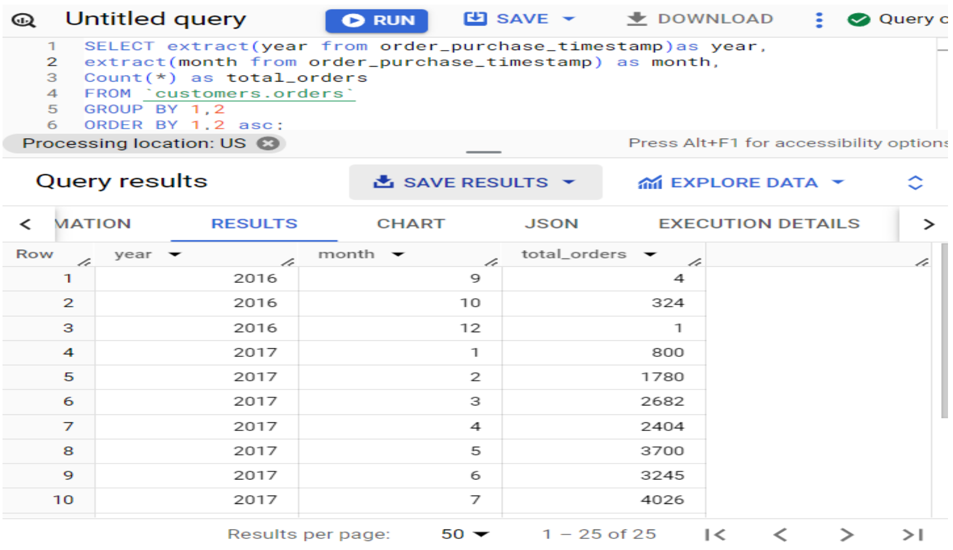
```
SELECT
COUNT(DISTINCT customer_state) as num_of_state
FROM `customers.orders`o
INNER JOIN`customers.customers` c ON o.customer_id = c.customer_id
WHERE o.customer_id BETWEEN '2016' and '2018'
```

Row	num_of_state
1	2

Interpretation – There are total of 2 cities and 2 states of customers who has made purchases during 2016 to 2018.

II) In-depth Exploration:

A) We can see month on month growing trend in the year 2017. Company can follow the strategy



```
SELECT extract(year from order_purchase_timestamp)as year,
extract(month from order_purchase_timestamp) as month,
Count(*) as total_orders
FROM `customers.orders`
GROUP BY 1,2
ORDER BY 1,2 asc;
```

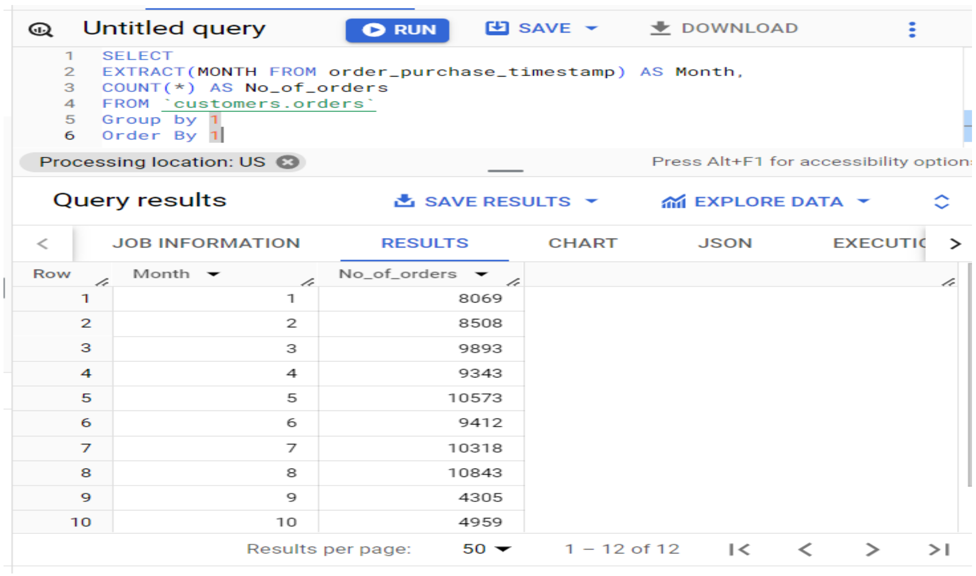
Row	year	month	total_orders
1	2016	9	4
2	2016	10	324
3	2016	12	1
4	2017	1	800
5	2017	2	1780
6	2017	3	2682
7	2017	4	2404
8	2017	5	3700
9	2017	6	3245
10	2017	7	4026

Results per page: 50 1 – 25 of 25

Interpretation - We can see month on month growing trend in the year 2017. Company can follow the strategy of the year 2017 in the following years to increase its sales and grow.

B) Monthly seasonality

```
SELECT
EXTRACT(MONTH FROM order_purchase_timestamp) AS Month,
COUNT(*) AS No_of_orders
FROM `customers.orders`
Group by 1
Order By 1
```



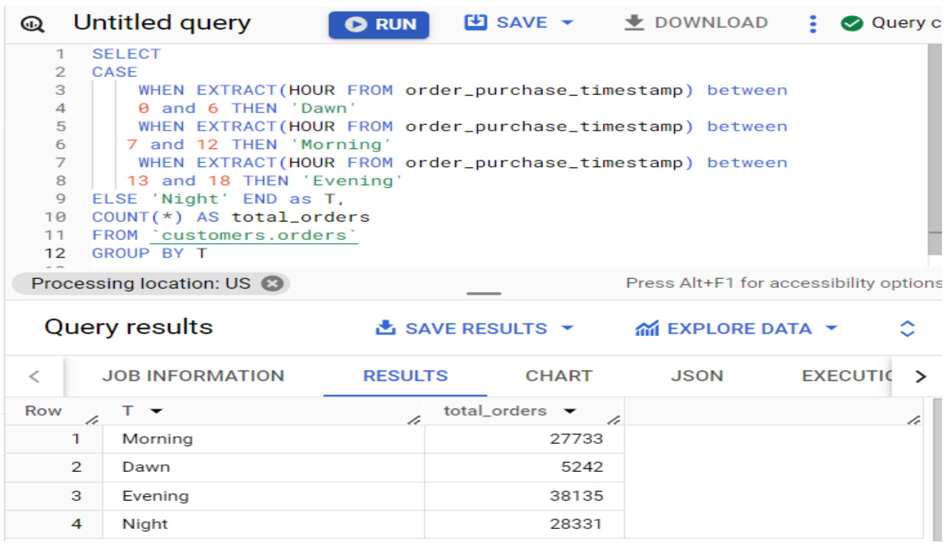
```
SELECT
EXTRACT(MONTH FROM order_purchase_timestamp) AS Month,
COUNT(*) AS No_of_orders
FROM `customers.orders`
Group by 1
Order By 1
```

Row	Month	No_of_orders
1	1	8069
2	2	8508
3	3	9893
4	4	9343
5	5	10573
6	6	9412
7	7	10318
8	8	10843
9	9	4305
10	10	4959

Interpretation – We can see that in the month of May, July and August the sales is comparatively high. The number of orders sold in these months are higher than the number of orders in other months. So, we can clearly see seasonality in these 3 months. From company's perspective we can follow the strategy used in these months to increase its sales.

C) Time of the day

```
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 'Morning'
    WHEN EXTRACT(HOUR FROM order_purchase_timestamp) between
   13 and 18 THEN 'Evening'
ELSE 'Night' END as T,
COUNT(*) AS total_orders
FROM `customers.orders`
GROUP BY T
```



```
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 'Morning'
    WHEN EXTRACT(HOUR FROM order_purchase_timestamp) between
   13 and 18 THEN 'Evening'
ELSE 'Night' END as T,
COUNT(*) AS total_orders
FROM `customers.orders`
GROUP BY T
```

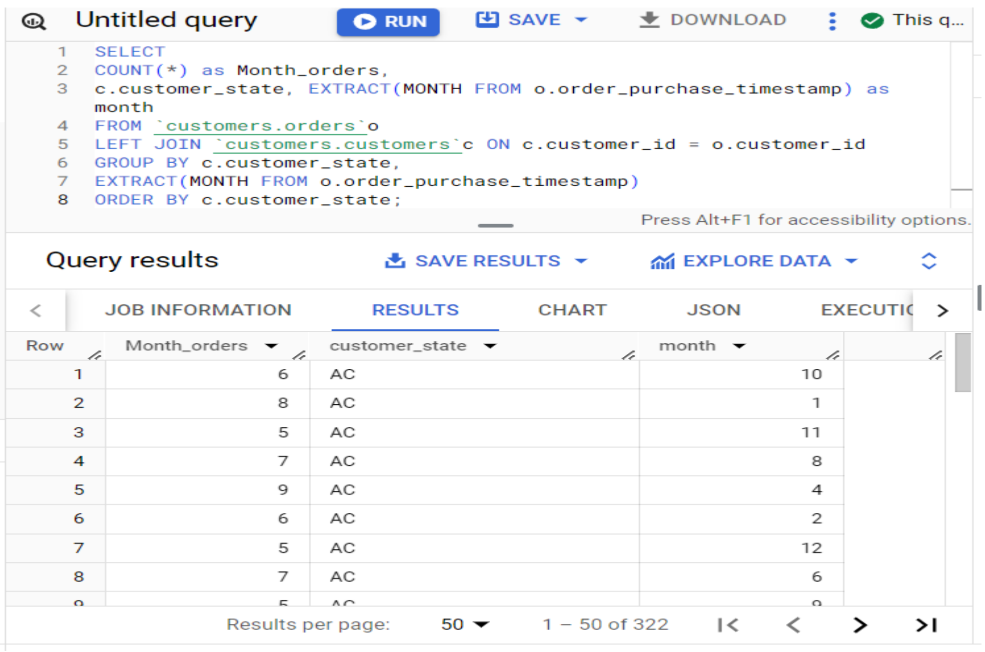
Row	T	total_orders
1	Morning	27733
2	Dawn	5242
3	Evening	38135
4	Night	28331

Interpretation – The total orders placed by the Brazilian customers is during evening time. We can see that the orders placed in the evening time is 38,135 which is comparatively higher than other time slots. The second most preferred time after evening for purchase is night.

III) Evolution of E-commerce orders in the Brazil region:

A) No. of orders placed in each month

```
SELECT
COUNT(*) as Month_orders,
c.customer_state, EXTRACT(MONTH FROM o.order_purchase_timestamp) as month
FROM `customers.orders`o
LEFT JOIN `customers.customers`c ON c.customer_id = o.customer_id
GROUP BY c.customer_state,
EXTRACT(MONTH FROM o.order_purchase_timestamp)
ORDER BY c.customer_state;
```



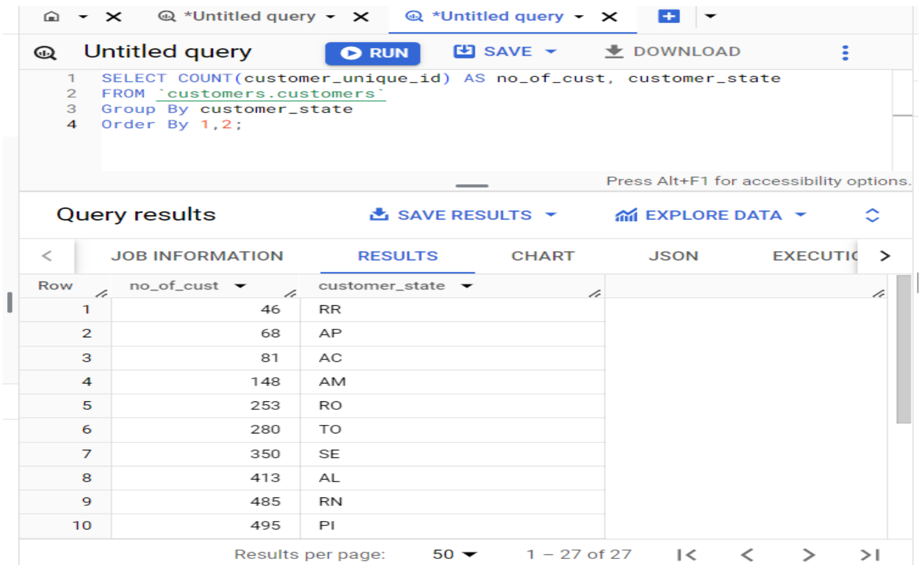
```
SELECT
COUNT(*) as Month_orders,
c.customer_state, EXTRACT(MONTH FROM o.order_purchase_timestamp) as
month
FROM `customers.orders`o
LEFT JOIN `customers.customers`c ON c.customer_id = o.customer_id
GROUP BY c.customer_state,
EXTRACT(MONTH FROM o.order_purchase_timestamp)
ORDER BY c.customer_state;
```

Row	Month_orders	customer_state	month
1	6	AC	10
2	8	AC	1
3	5	AC	11
4	7	AC	8
5	9	AC	4
6	6	AC	2
7	5	AC	12
8	7	AC	6
9	5	AC	9

Interpretation – Here is the data for the month on month order placed by the Brazilian customers between year 2016 to 2018.

B) Customer distribution

```
SELECT COUNT(customer_unique_id) AS no_of_cust, customer_state
FROM `customers.customers`
Group By customer_state
Order By 1,2;
```



```
SELECT COUNT(customer_unique_id) AS no_of_cust, customer_state
FROM `customers.customers`
Group By customer_state
Order By 1,2;
```

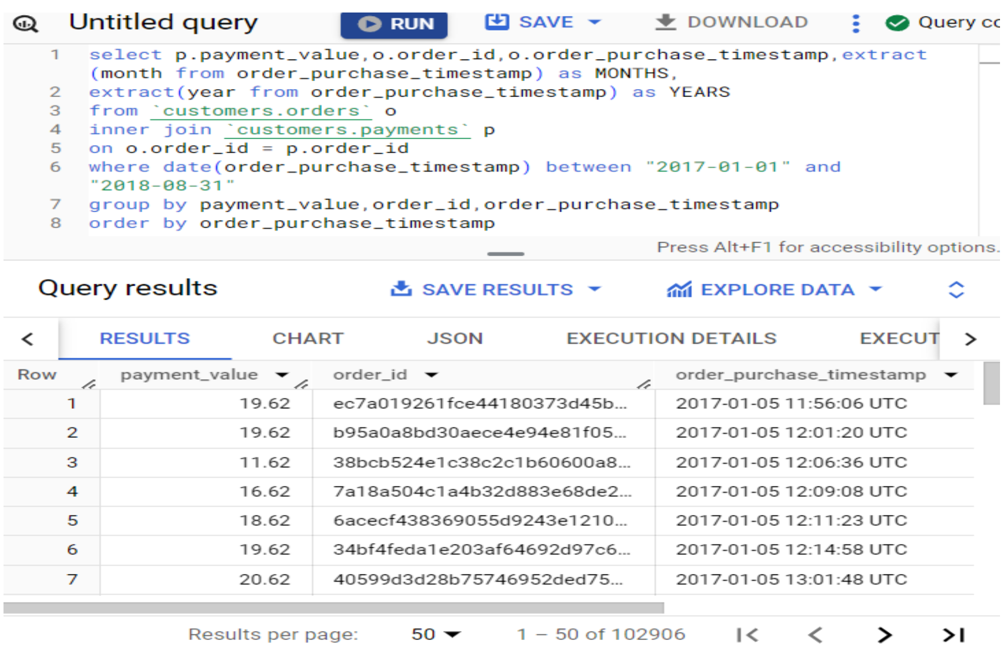
Row	no_of_cust	customer_state
1	46	RR
2	68	AP
3	81	AC
4	148	AM
5	253	RO
6	280	TO
7	350	SE
8	413	AL
9	485	RN
10	495	PI

Interpretation – We can see here state wise unique number of customers who have placed the order. It means that the table contain only the data of those customers who have at least made a purchase and if they have placed multiple orders then they would be count as only one.

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

A) Percentage increase

```
select p.payment_value,o.order_id,o.order_purchase_timestamp,extract(month from
order_purchase_timestamp) as MONTHS,
extract(year from order_purchase_timestamp) as YEARS
from `customers.orders` o
inner join `customers.payments` p
on o.order_id = p.order_id
where date(order_purchase_timestamp) between "2017-01-01" and "2018-08-31"
group by payment_value,order_id,order_purchase_timestamp
order by order_purchase_timestamp
```



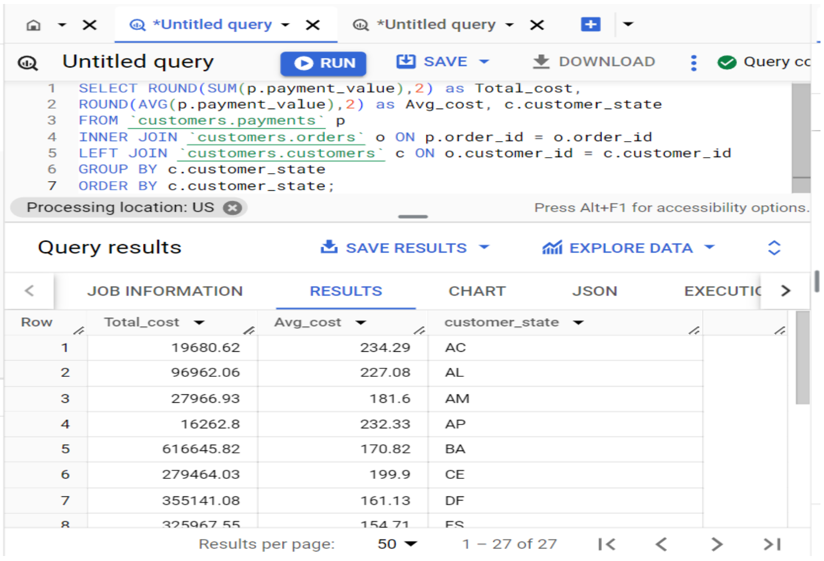
```
select p.payment_value,o.order_id,o.order_purchase_timestamp,extract
(month from order_purchase_timestamp) as MONTHS,
extract(year from order_purchase_timestamp) as YEARS
from `customers.orders` o
inner join `customers.payments` p
on o.order_id = p.order_id
where date(order_purchase_timestamp) between "2017-01-01" and
"2018-08-31"
group by payment_value,order_id,order_purchase_timestamp
order by order_purchase_timestamp
```

Row	payment_value	order_id	order_purchase_timestamp
1	19.62	ec7a019261fce44180373d45b...	2017-01-05 11:56:06 UTC
2	19.62	b95a0a8bd30aece4e94e81f05...	2017-01-05 12:01:20 UTC
3	11.62	38bcb524e1c38c2c1b60600a8...	2017-01-05 12:06:36 UTC
4	16.62	7a18a504c1a4b32d883e68de2...	2017-01-05 12:09:08 UTC
5	18.62	6acecf438369055d9243e1210...	2017-01-05 12:11:23 UTC
6	19.62	34bf4feda1e203af64692d97c6...	2017-01-05 12:14:58 UTC
7	20.62	40599d3d28b75746952ded75...	2017-01-05 13:01:48 UTC

Interpretation – We can see certain percentage of change in the number of orders placed in the year 2018.

B) Total and Average value of order price

```
SELECT ROUND(SUM(p.payment_value),2) as Total_cost,
ROUND(AVG(p.payment_value),2) as Avg_cost, c.customer_state
FROM `customers.payments` p
INNER JOIN `customers.orders` o ON p.order_id = o.order_id
LEFT JOIN `customers.customers` c ON o.customer_id = c.customer_id
GROUP BY c.customer_state
ORDER BY c.customer_state;
```



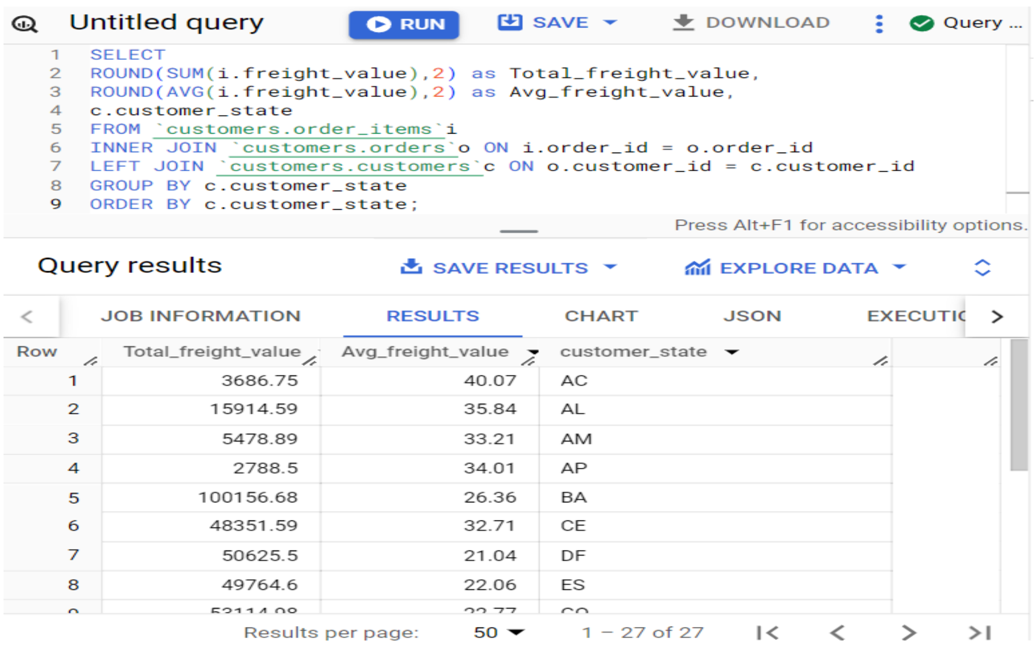
```
SELECT ROUND(SUM(p.payment_value),2) as Total_cost,
ROUND(AVG(p.payment_value),2) as Avg_cost, c.customer_state
FROM `customers.payments` p
INNER JOIN `customers.orders` o ON p.order_id = o.order_id
LEFT JOIN `customers.customers` c ON o.customer_id = c.customer_id
GROUP BY c.customer_state
ORDER BY c.customer_state;
```

Row	Total_cost	Avg_cost	customer_state
1	19680.62	234.29	AC
2	96962.06	227.08	AL
3	27966.93	181.6	AM
4	16262.8	232.33	AP
5	616645.82	170.82	BA
6	279464.03	199.9	CE
7	355141.08	161.13	DF
8	325967.55	154.71	ES

Interpretation – The above table contains the total and average cost of the orders made in each state. We have gathered the data from multiple tables and so we have joined payments, orders and customer table to get the expected output.

C) Total & Average value of order freight for each state

```sql
SELECT
ROUND(SUM(i.freight_value),2) as Total_freight_value,
ROUND(AVG(i.freight_value),2) as Avg_freight_value,
c.customer_state
FROM `customers.order_items`i
INNER JOIN `customers.orders`o ON i.order_id = o.order_id
LEFT JOIN `customers.customers`c ON o.customer_id = c.customer_id
GROUP BY c.customer_state
ORDER BY c.customer_state;
```



Untitled query

```sql
SELECT
ROUND(SUM(i.freight_value),2) as Total_freight_value,
ROUND(AVG(i.freight_value),2) as Avg_freight_value,
c.customer_state
FROM `customers.order_items`i
INNER JOIN `customers.orders`o ON i.order_id = o.order_id
LEFT JOIN `customers.customers`c ON o.customer_id = c.customer_id
GROUP BY c.customer_state
ORDER BY c.customer_state;
```

Query results

Row	Total_freight_value	Avg_freight_value	customer_state
1	3686.75	40.07	AC
2	15914.59	35.84	AL
3	5478.89	33.21	AM
4	2788.5	34.01	AP
5	100156.68	26.36	BA
6	48351.59	32.71	CE
7	50625.5	21.04	DF
8	49764.6	22.06	ES
9	[illegible]	[illegible]	[illegible]

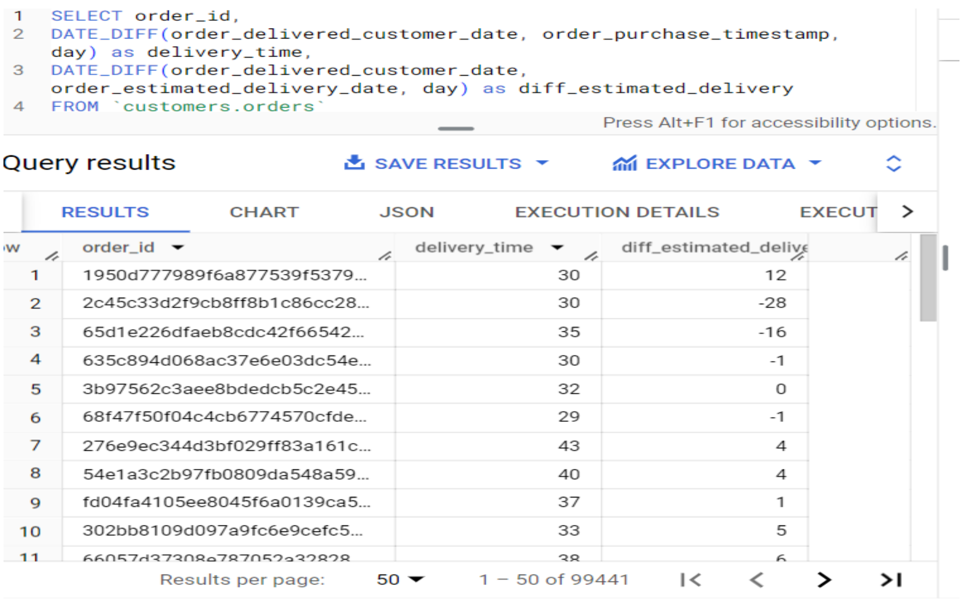
Results per page: 50 1 – 27 of 27

Interpretation – Total freight value is the additional amount that has been spent on the delivery of the order. Higher the freight value of a state higher the quantity of items sold in that state.

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

A) No. of days taken to deliver

```sql
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
diff_estimated_delivery
FROM `customers.orders`
```



```
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 diff_estimated_delivery
FROM `customers.orders`
```

Press Alt+F1 for accessibility options.

Query results — SAVE RESULTS — EXPLORE DATA

RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUT

w	order_id	delivery_time	diff_estimated_delive
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

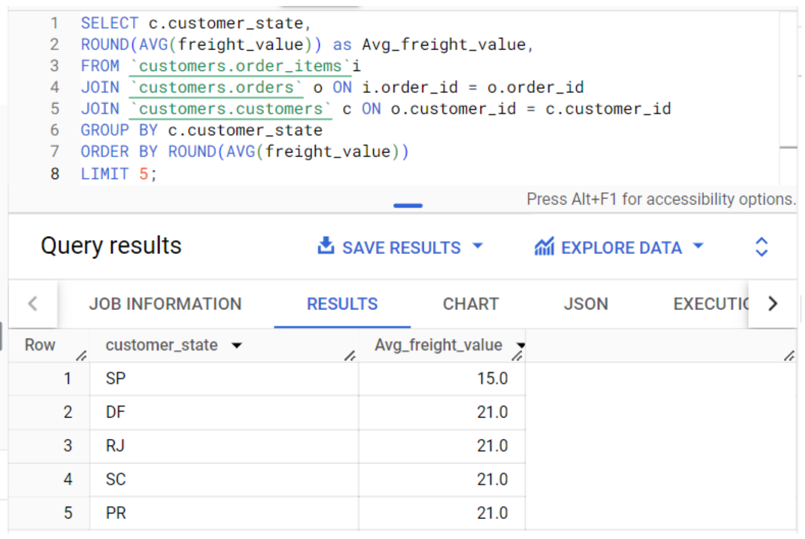
Results per page: 50 1 – 50 of 99441

Interpretation – As we can see that the delivery time is between 30-40 days for each order. It means that customers got their orders within 30-40 days of making the purchase. The company is efficient in terms of delivering its order as there are negative values in the table showing that it has delivered earlier than expected. Orders reached their customer before the expected delivery date.

B)

a) Lowest average freight value

```
SELECT c.customer_state,
ROUND(AVG(freight_value)) as Avg_freight_value,
FROM `customers.order_items`i
JOIN `customers.orders` o ON i.order_id = o.order_id
JOIN `customers.customers` c ON o.customer_id = c.customer_id
GROUP BY c.customer_state
ORDER BY ROUND(AVG(freight_value))
LIMIT 5;
```



```
SELECT c.customer_state,
ROUND(AVG(freight_value)) as Avg_freight_value,
FROM `customers.order_items`i
JOIN `customers.orders` o ON i.order_id = o.order_id
JOIN `customers.customers` c ON o.customer_id = c.customer_id
GROUP BY c.customer_state
ORDER BY ROUND(AVG(freight_value))
LIMIT 5;
```

Press Alt+F1 for accessibility options.

Query results — SAVE RESULTS — EXPLORE DATA

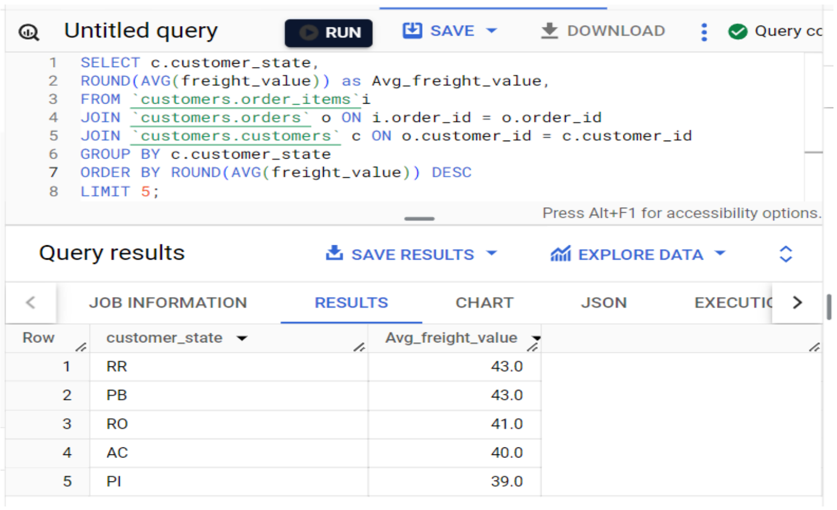
JOB INFORMATION | RESULTS | CHART | JSON | EXECUTIC

Row	customer_state	Avg_freight_value
1	SP	15.0
2	DF	21.0
3	RJ	21.0
4	SC	21.0
5	PR	21.0

Interpretation – SP,DF,RJ,SC and PR are the top states that have the lowest average freight value.

b) Highest average freight value

```
SELECT c.customer_state,
ROUND(AVG(freight_value)) as Avg_freight_value,
FROM `customers.order_items`i
JOIN `customers.orders` o ON i.order_id = o.order_id
JOIN `customers.customers` c ON o.customer_id = c.customer_id
GROUP BY c.customer_state
ORDER BY ROUND(AVG(freight_value)) DESC
LIMIT 5;
```



```
SELECT c.customer_state,
ROUND(AVG(freight_value)) as Avg_freight_value,
FROM `customers.order_items`i
JOIN `customers.orders` o ON i.order_id = o.order_id
JOIN `customers.customers` c ON o.customer_id = c.customer_id
GROUP BY c.customer_state
ORDER BY ROUND(AVG(freight_value)) DESC
LIMIT 5;
```

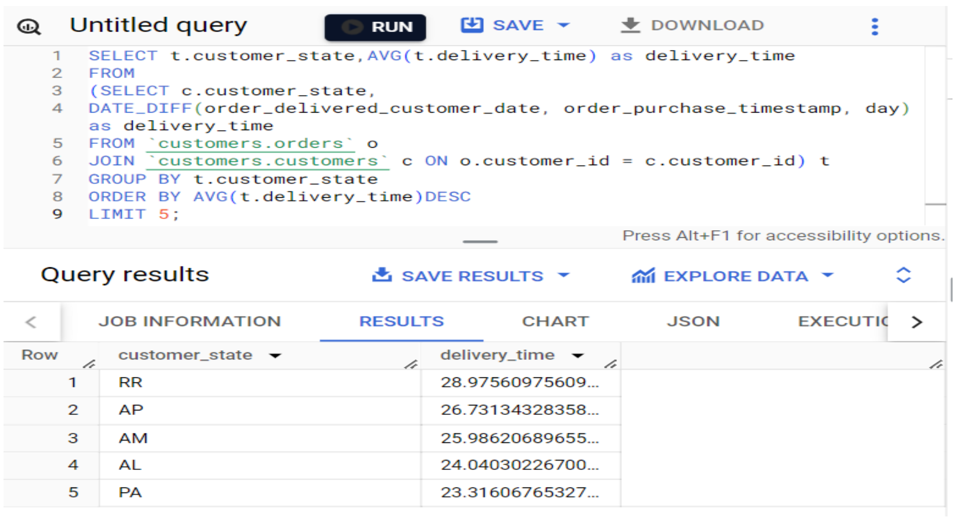
Row	customer_state	Avg_freight_value
1	RR	43.0
2	PB	43.0
3	RO	41.0
4	AC	40.0
5	PI	39.0

Interpretation – RR,PB,RO,AC and PI are the top states that have the highest average freight value.

C)

a) Top 5 States with highest delivery time

```
SELECT t.customer_state,AVG(t.delivery_time) as delivery_time
FROM
(SELECT c.customer_state,
DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, day)as delivery_time
FROM `customers.orders` o
JOIN `customers.customers` c ON o.customer_id = c.customer_id) t
GROUP BY t.customer_state
ORDER BY AVG(t.delivery_time)DESC
LIMIT 5;
```

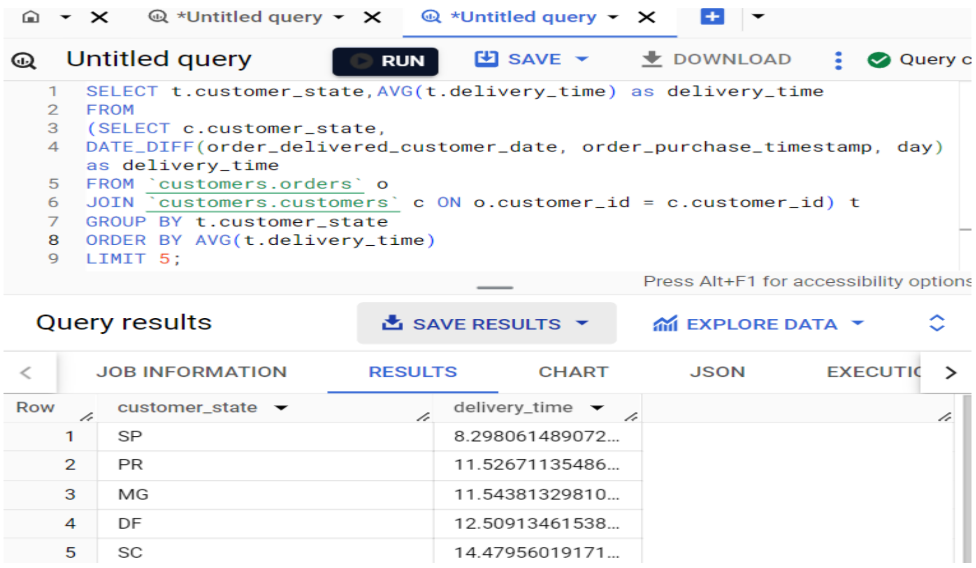


```
SELECT t.customer_state,AVG(t.delivery_time) as delivery_time
FROM
(SELECT c.customer_state,
DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, day)
as delivery_time
FROM `customers.orders` o
JOIN `customers.customers` c ON o.customer_id = c.customer_id) t
GROUP BY t.customer_state
ORDER BY AVG(t.delivery_time)DESC
LIMIT 5;
```

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

b) Top 5 States with lowest delivery time

```
SELECT t.customer_state,AVG(t.delivery_time) as delivery_time
FROM
(SELECT c.customer_state,
DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, day)as delivery_time
FROM `customers.orders` o
JOIN `customers.customers` c ON o.customer_id = c.customer_id) t
GROUP BY t.customer_state
ORDER BY AVG(t.delivery_time)
LIMIT 5;
```

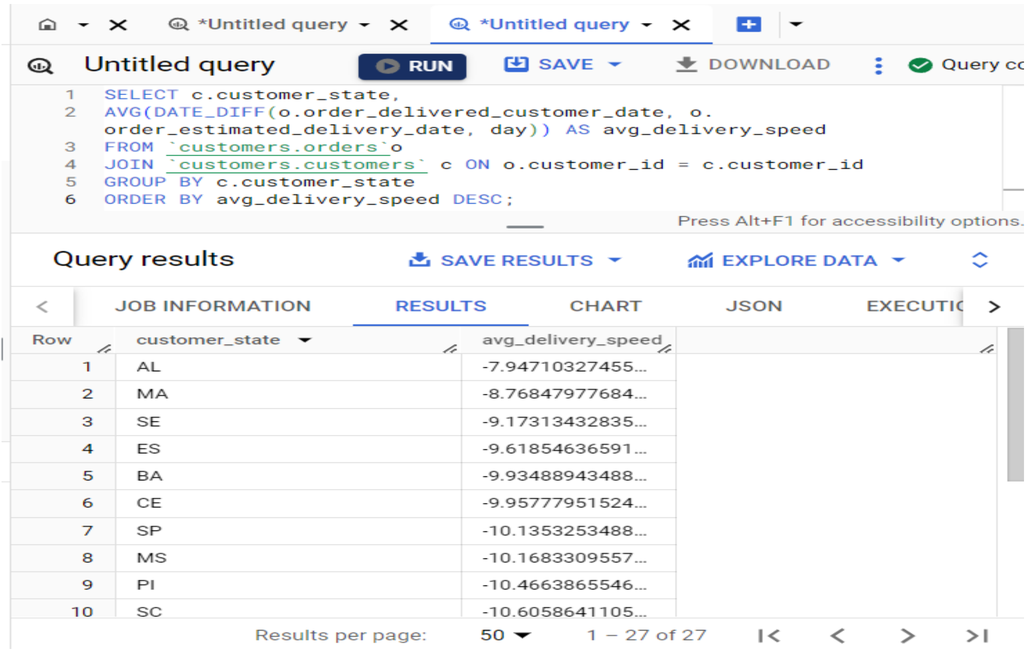


```
SELECT t.customer_state,AVG(t.delivery_time) as delivery_time
FROM
(SELECT c.customer_state,
DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, day)
as delivery_time
FROM `customers.orders` o
JOIN `customers.customers` c ON o.customer_id = c.customer_id) t
GROUP BY t.customer_state
ORDER BY AVG(t.delivery_time)
LIMIT 5;
```

Row	customer_state	delivery_time
1	SP	8.298061489072...
2	PR	11.52671135486...
3	MG	11.54381329810...
4	DF	12.50913461538...
5	SC	14.47956019171...

D) Top 5 states where the order delivery is really fast

```
SELECT c.customer_state,
AVG(DATE_DIFF(o.order_delivered_customer_date, o.order_estimated_delivery_date, day)) AS
avg_delivery_speed
FROM `customers.orders`o
JOIN `customers.customers` c ON o.customer_id = c.customer_id
GROUP BY c.customer_state
ORDER BY avg_delivery_speed DESC;
```



```
SELECT c.customer_state,
AVG(DATE_DIFF(o.order_delivered_customer_date, o.
order_estimated_delivery_date, day)) AS avg_delivery_speed
FROM `customers.orders`o
JOIN `customers.customers` c ON o.customer_id = c.customer_id
GROUP BY c.customer_state
ORDER BY avg_delivery_speed DESC;
```

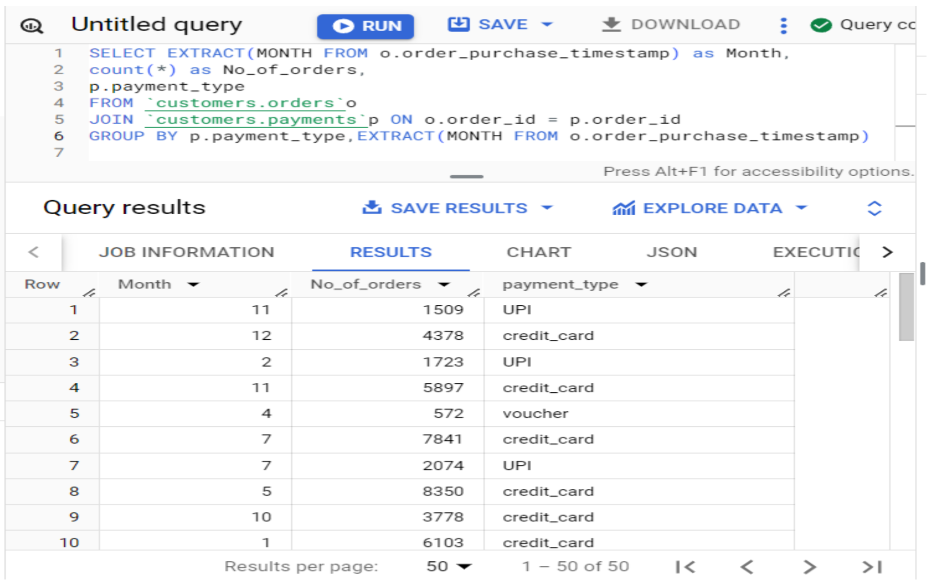
Row	customer_state	avg_delivery_speed
1	AL	-7.94710327455...
2	MA	-8.76847977684...
3	SE	-9.17313432835...
4	ES	-9.61854636591...
5	BA	-9.93488943488...
6	CE	-9.95777951524...
7	SP	-10.1353253488...
8	MS	-10.1683309557...
9	PI	-10.4663865546...
10	SC	-10.6058641105...

Interpretation – Average delivery speed is negative it means that delivery took less time than expected delivery date. All the states are efficient in achieving its delivery targets or speed. Target should follow the same delivery pattern and sustain in it.

VI) Analysis based on the payments:

A) month on month no. of orders placed using different payment types

```
SELECT EXTRACT(MONTH FROM o.order_purchase_timestamp) as Month,
count(*) as No_of_orders,
p.payment_type
FROM `customers.orders`o
JOIN `customers.payments`p ON o.order_id = p.order_id
GROUP BY p.payment_type,EXTRACT(MONTH FROM o.order_purchase_timestamp)
```



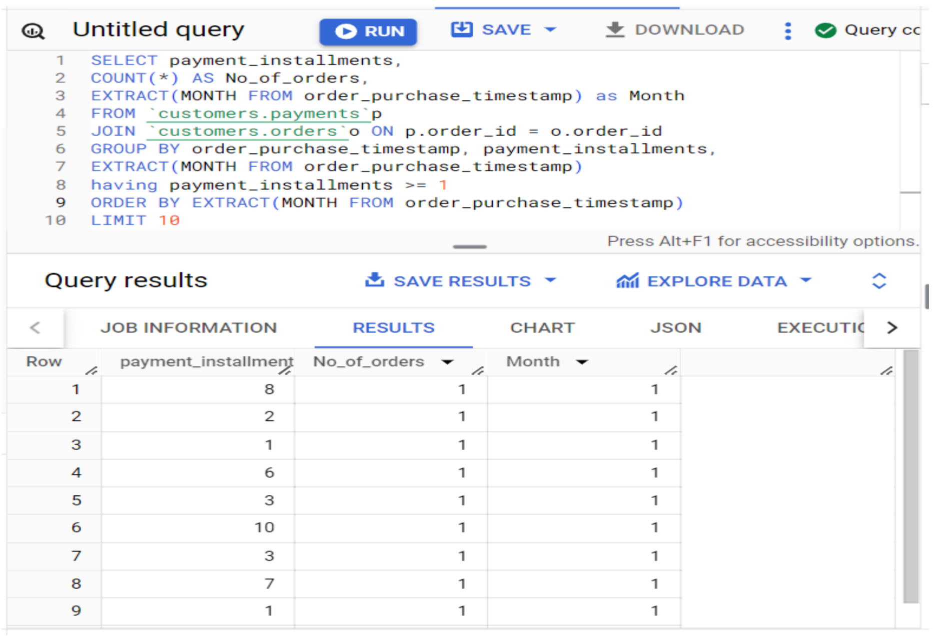
```
SELECT EXTRACT(MONTH FROM o.order_purchase_timestamp) as Month,
count(*) as No_of_orders,
p.payment_type
FROM `customers.orders`o
JOIN `customers.payments`p ON o.order_id = p.order_id
GROUP BY p.payment_type,EXTRACT(MONTH FROM o.order_purchase_timestamp)
```

Row	Month	No_of_orders	payment_type
1	11	1509	UPI
2	12	4378	credit_card
3	2	1723	UPI
4	11	5897	credit_card
5	4	572	voucher
6	7	7841	credit_card
7	7	2074	UPI
8	5	8350	credit_card
9	10	3778	credit_card
10	1	6103	credit_card

Interpretation - Customers has placed orders using multiple payment modes like UPI, Credit card, voucher, debit card, etc. Mostly Credit card and UPI is used for payment in different months.

B) no. of orders placed on the basis of the payment instalments that have been paid.

```
SELECT payment_installments,
COUNT(*) AS No_of_orders,
EXTRACT(MONTH FROM order_purchase_timestamp) as Month
FROM `customers.payments`p
JOIN `customers.orders`o ON p.order_id = o.order_id
GROUP BY order_purchase_timestamp, payment_installments,
EXTRACT(MONTH FROM order_purchase_timestamp)
having payment_installments >= 1
ORDER BY EXTRACT(MONTH FROM order_purchase_timestamp)
LIMIT 10
```



```
SELECT payment_installments,
COUNT(*) AS No_of_orders,
EXTRACT(MONTH FROM order_purchase_timestamp) as Month
FROM `customers.payments`p
JOIN `customers.orders`o ON p.order_id = o.order_id
GROUP BY order_purchase_timestamp, payment_installments,
EXTRACT(MONTH FROM order_purchase_timestamp)
having payment_installments >= 1
ORDER BY EXTRACT(MONTH FROM order_purchase_timestamp)
LIMIT 10
```

Row	payment_installment	No_of_orders	Month
1	8	1	1
2	2	1	1
3	1	1	1
4	6	1	1
5	3	1	1
6	10	1	1
7	3	1	1
8	7	1	1
9	1	1	1

Interpretation – Here is the list of the orders where at least one instalment is paid by the customer. In the first month itself the count of installments paid is till 10 that means customers are paying it since last 10 months.