Introduction:

Target, a leading retail company, has expanded its e-commerce operations in United States to cater and is ready to the growing online shopping trend in Brazil. To ensure success in this competitive market, it is crucial for Target to understand the dynamics of e-commerce in Brazil and leverage data-driven insights to enhance its operations. In this project we will analyse Target's e-commerce dataset using structured query language (SQL) power queries to provide actionable and upcoming 100,000 orders placed between 2016 and 2018 recommendations for improving their operations in Brazil.

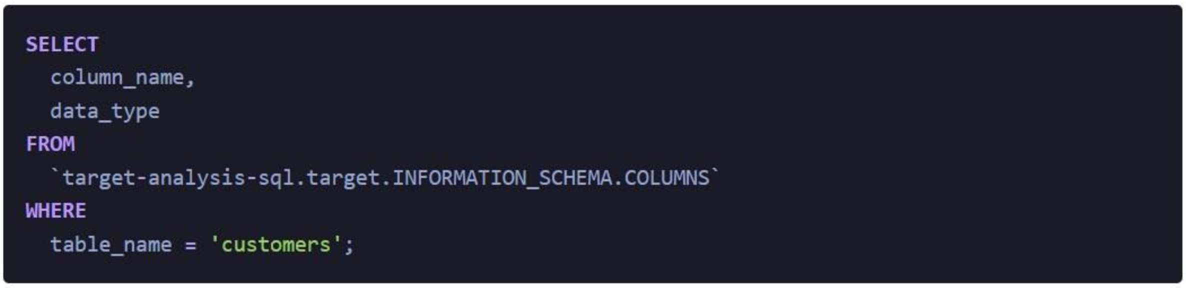
Objectives:

1. Gain proficiency in SQL by analysing Target's Brazil e-commerce dataset.
2. Learn how to perform initial exploration of a dataset, including data cleaning and preparation using SQL queries.
3. Understand how to analyse and interpret e-commerce trends in Brazil using SQL queries.
4. Develop skills in identifying and analysing seasonality patterns in e-commerce data using SQL queries.
5. Learn to extract valuable insights about customer buying patterns and preferences through SQL analysis.
6. Develop the ability to make data-driven recommendations and actionable insights for improving e-commerce operations based on SQL analysis of the dataset.

By understanding the data types of each & every table, we ensure accurate analysis and interpretation of the dataset.

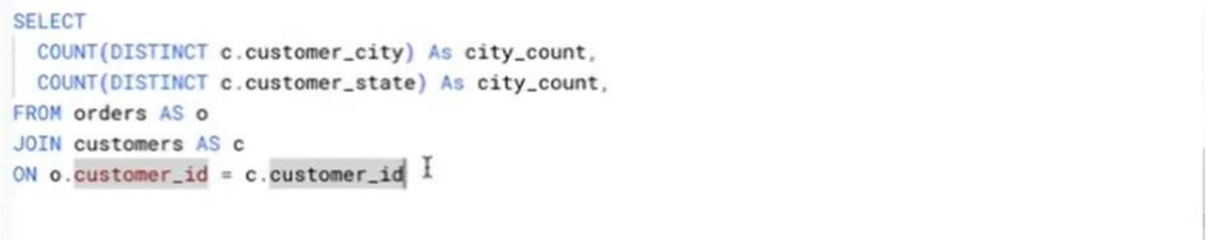
This allowed us to determine the start and end dates of the data i.e. from 4th September 2016 to 17th October 2018.

Q] Data type of all columns in the "customers" table.



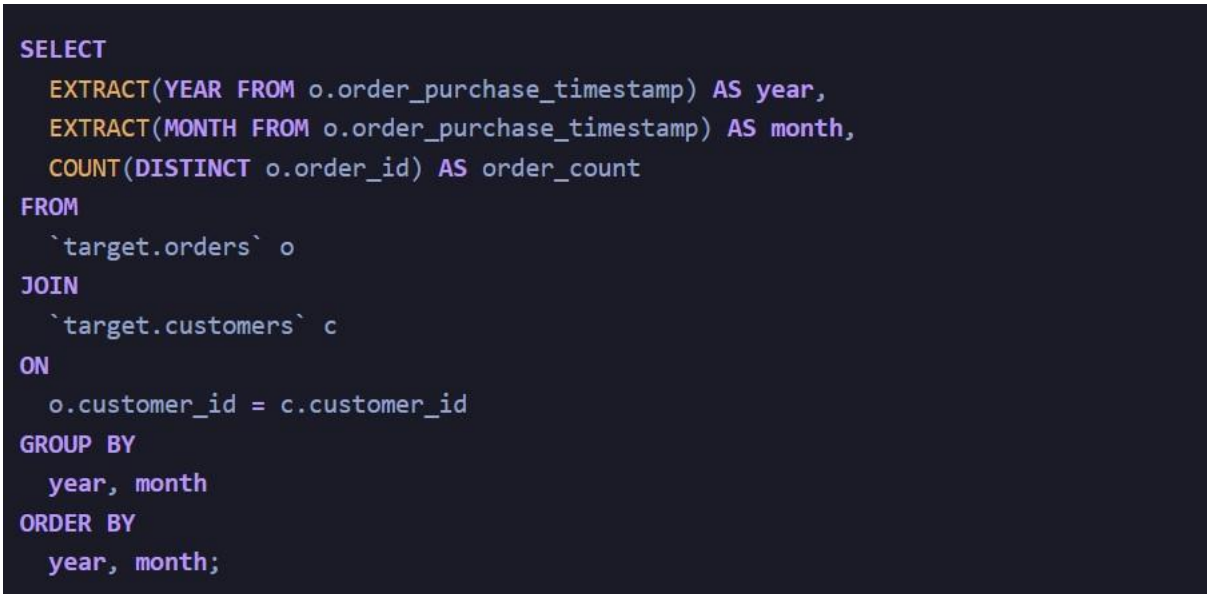
```
SELECT
  column_name,
  data_type
FROM
  `target-analysis-sql.target.INFORMATION_SCHEMA.COLUMNS`
WHERE
  table_name = 'customers';
```

Q] Get the time range between which the orders were placed.



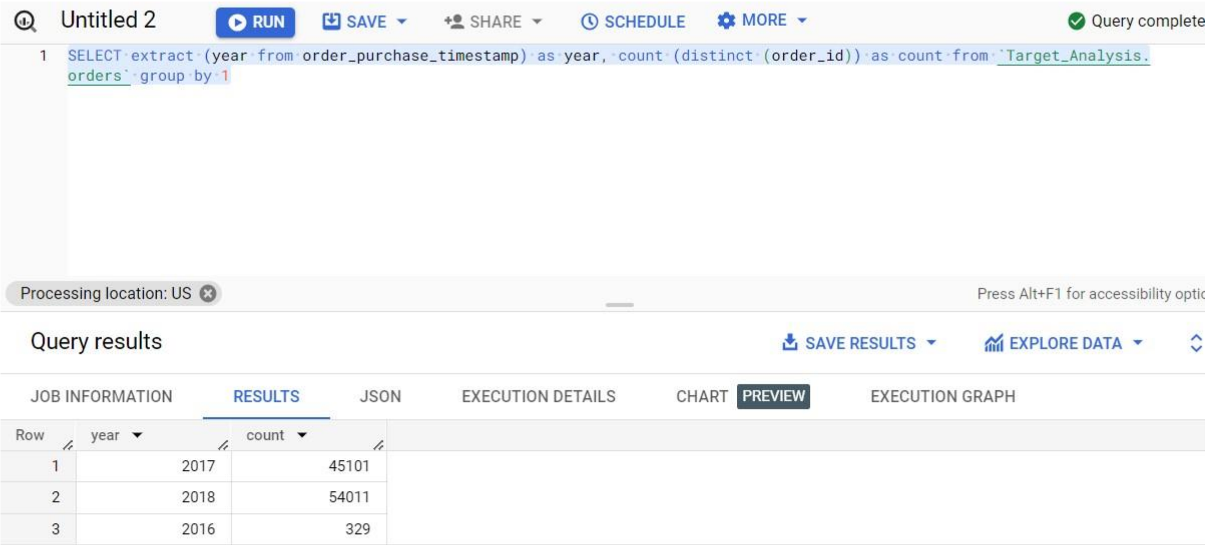
```
SELECT
  COUNT(DISTINCT c.customer_city) As city_count,
  COUNT(DISTINCT c.customer_state) As city_count,
FROM orders AS o
JOIN customers AS c
ON o.customer_id = c.customer_id
```

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



```
SELECT
  EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
  EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
  COUNT(DISTINCT o.order_id) AS order_count
FROM
  `target.orders` o
JOIN
  `target.customers` c
ON
  o.customer_id = c.customer_id
GROUP BY
  year, month
ORDER BY
  year, month;
```

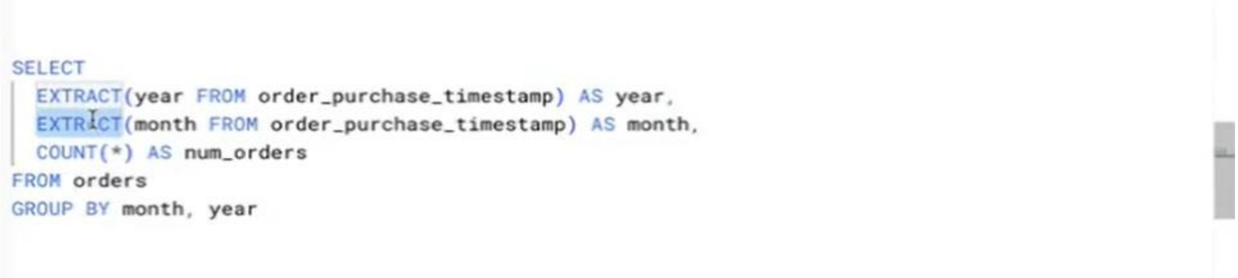
Q] Is there a growing trend in the no. of orders placed over the past years?



```
SELECT extract (year from order_purchase_timestamp) as year, count (distinct (order_id)) as count from `Target_Analysis.
orders` group by 1
```

Row	year	count
1	2017	45101
2	2018	54011
3	2016	329

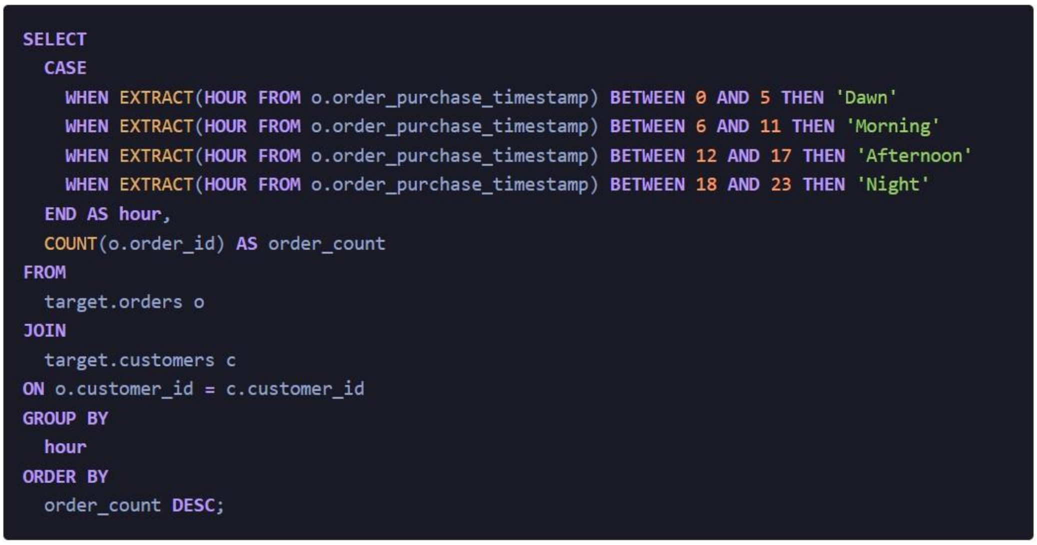
Q] Can we see some kind of monthly seasonality in terms of the no. of orders being placed?



```
SELECT
  EXTRACT(year FROM order_purchase_timestamp) AS year,
  EXTRACT(month FROM order_purchase_timestamp) AS month,
  COUNT(*) AS num_orders
FROM orders
GROUP BY month, year
```

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

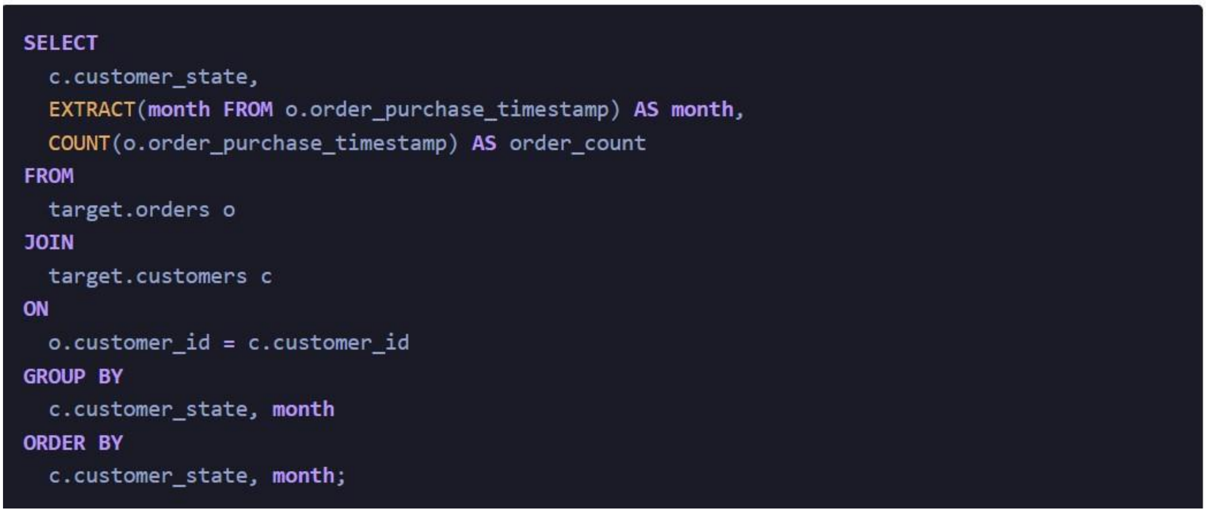
- 0-6 hrs : Dawn
- 7-12 hrs : Mornings
- 13-18 hrs : Afternoon
- 19-23 hrs : Night



```
SELECT
  CASE
    WHEN EXTRACT(HOUR FROM o.order_purchase_timestamp) BETWEEN 0 AND 5 THEN 'Dawn'
    WHEN EXTRACT(HOUR FROM o.order_purchase_timestamp) BETWEEN 6 AND 11 THEN 'Morning'
    WHEN EXTRACT(HOUR FROM o.order_purchase_timestamp) BETWEEN 12 AND 17 THEN 'Afternoon'
    WHEN EXTRACT(HOUR FROM o.order_purchase_timestamp) BETWEEN 18 AND 23 THEN 'Night'
  END AS hour,
  COUNT(o.order_id) AS order_count
FROM
  target.orders o
JOIN
  target.customers c
ON o.customer_id = c.customer_id
GROUP BY
  hour
ORDER BY
  order_count DESC;
```

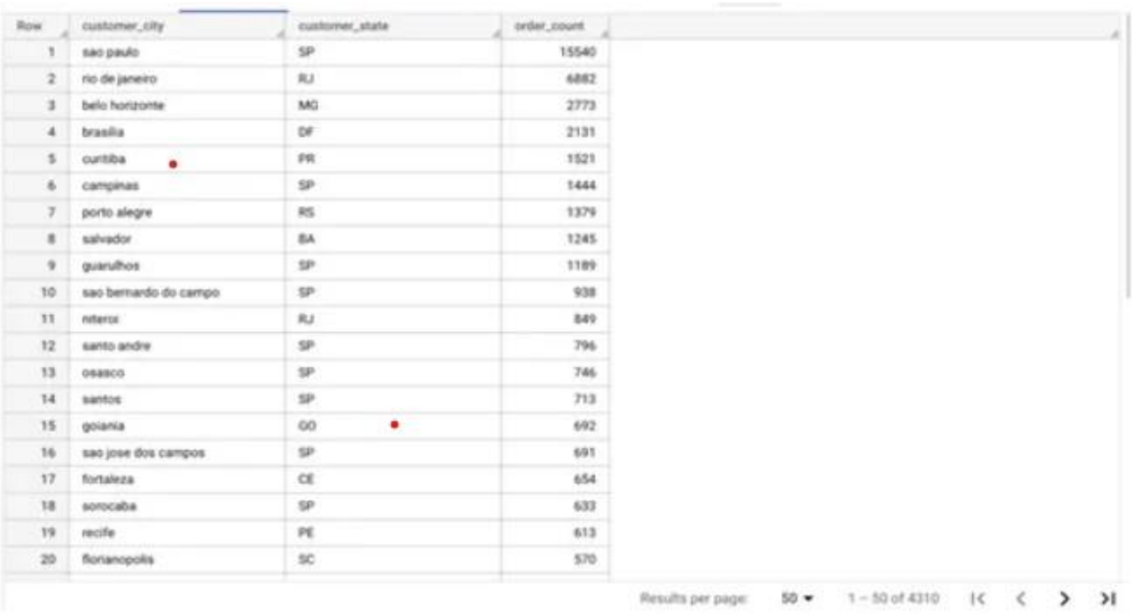
In conclusion, analysing the buying patterns of Brazilian customers reveals the growing trend of e-commerce in the country, highlights the importance of considering various factors for a complete understanding of the e-commerce scenario, and sheds light on the preferred time periods for online shopping. Armed with these insights, Target and other e-commerce businesses can make data-driven decisions to enhance their operations and improve customer satisfaction.

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



```
SELECT
  c.customer_state,
  EXTRACT(month FROM o.order_purchase_timestamp) AS month,
  COUNT(o.order_purchase_timestamp) AS order_count
FROM
  target.orders o
JOIN
  target.customers c
ON
  o.customer_id = c.customer_id
GROUP BY
  c.customer_state, month
ORDER BY
  c.customer_state, month;
```

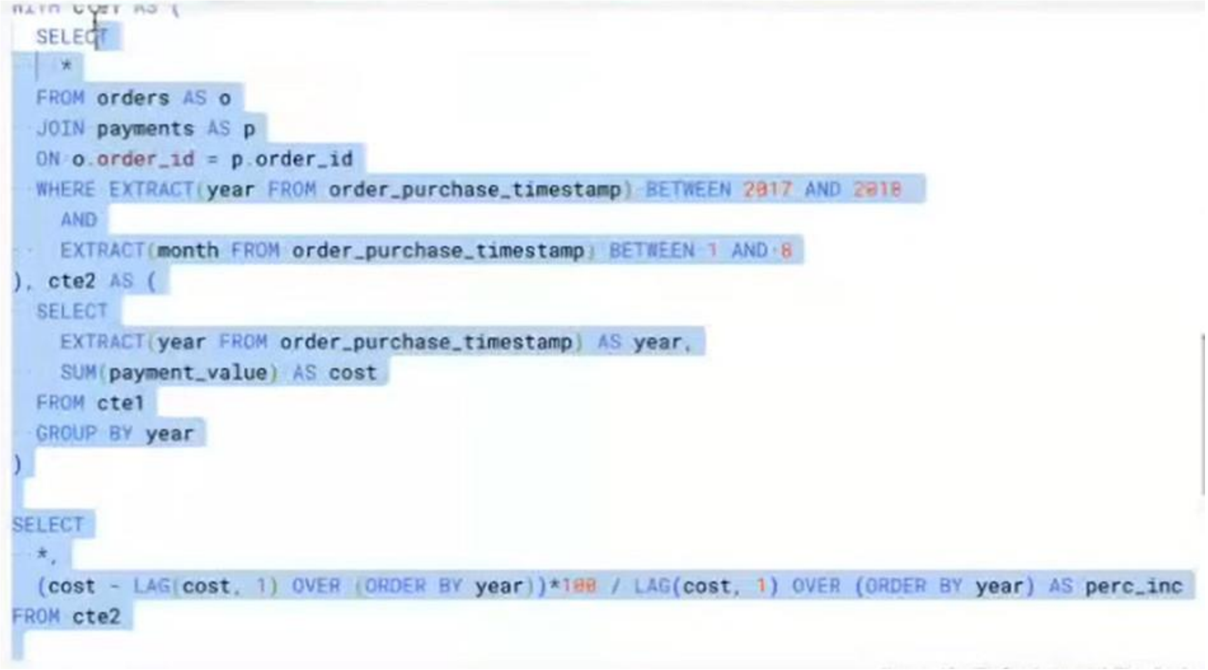
Q] How are the customers distributed across all the states?



Row	customer_city	customer_state	order_count
1	sao paulo	SP	15540
2	rio de janeiro	RJ	6882
3	belo horizonte	MG	2773
4	brasilia	DF	2131
5	curitiba	PR	1521
6	campinas	SP	1444
7	porto alegre	RS	1379
8	salvador	BA	1245
9	guarulhos	SP	1189
10	sao bernardo do campo	SP	938
11	niteroi	RJ	849
12	santo andre	SP	796
13	osasco	SP	746
14	santos	SP	713
15	goiania	GO	692
16	sao jose dos campos	SP	691
17	fortaleza	CE	654
18	sorocaba	SP	633
19	recife	PE	613
20	florianopolis	SC	570

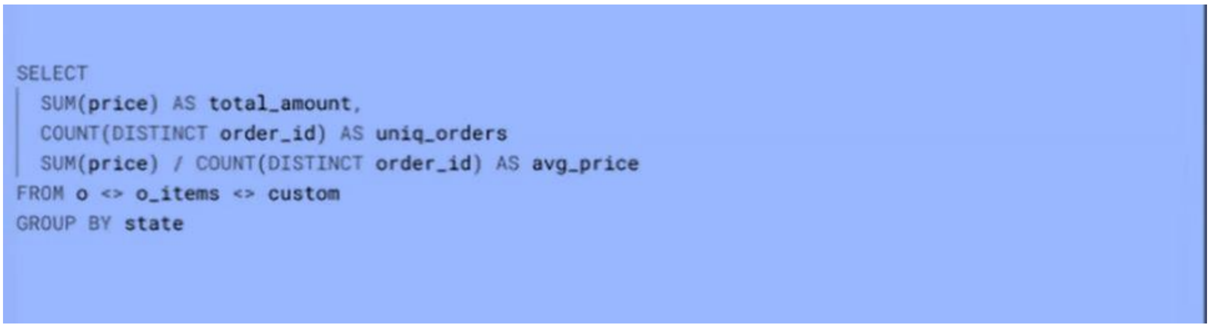
Furthermore, we examined the cities and states of customers who placed orders during the specified time period. The following SQL query helped us identify the customer distribution.

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



```
WITH cte1 AS (
  SELECT
  | *
  FROM orders AS o
  JOIN payments AS p
  ON o.order_id = p.order_id
  WHERE EXTRACT(year FROM order_purchase_timestamp) BETWEEN 2017 AND 2018
    AND
    EXTRACT(month FROM order_purchase_timestamp) BETWEEN 1 AND 8
), cte2 AS (
  SELECT
    EXTRACT(year FROM order_purchase_timestamp) AS year,
    SUM(payment_value) AS cost
  FROM cte1
  GROUP BY year
)

SELECT
  *,
  (cost - LAG(cost, 1) OVER (ORDER BY year))*100 / LAG(cost, 1) OVER (ORDER BY year) AS perc_inc
FROM cte2
```

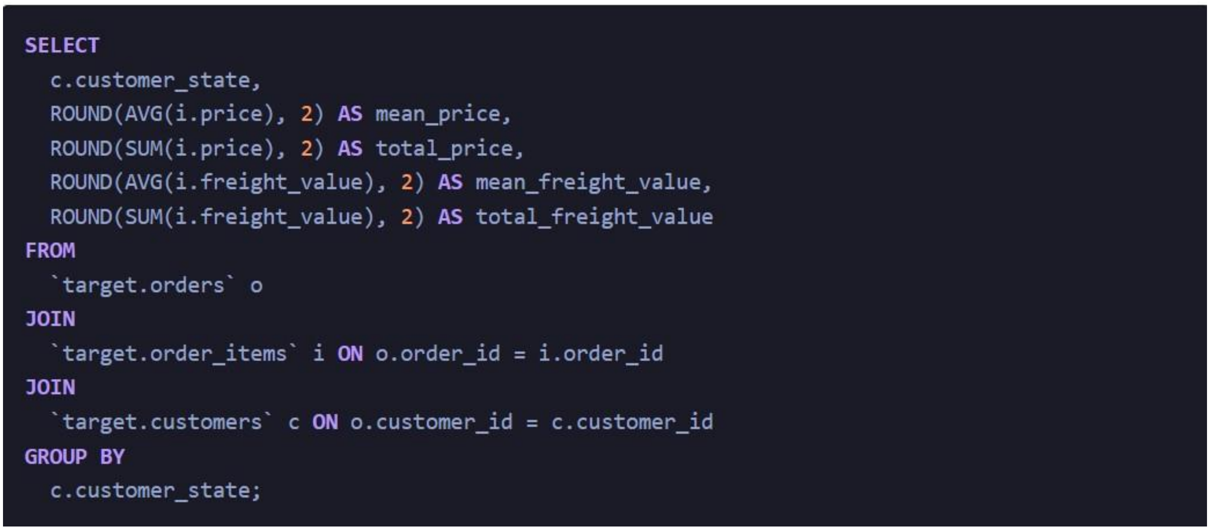
Q] Calculate the Total & Average value of order price for each state.



```
SELECT
| SUM(price) AS total_amount,
| COUNT(DISTINCT order_id) AS uniq_orders
| SUM(price) / COUNT(DISTINCT order_id) AS avg_price
FROM o <> o_items <> custom
GROUP BY state
```

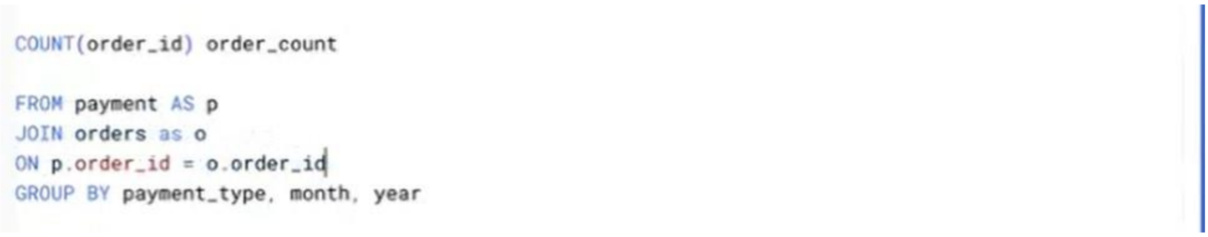
Q] Calculate the Total & Average value of order freight for each state.

To answer this simply we need to do the same actions like previous. Just use we should by sum and count distinct order.



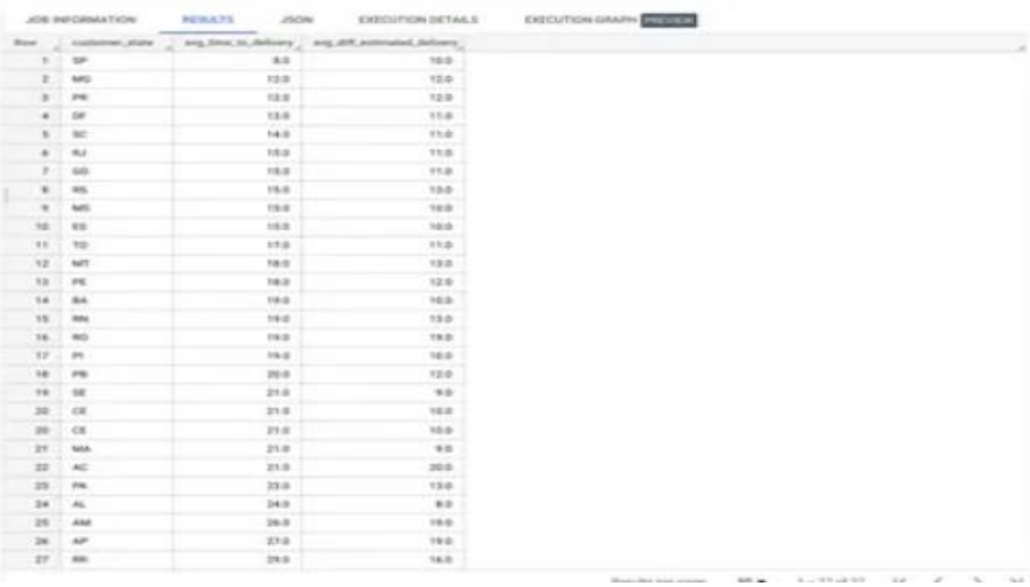
```
SELECT
  c.customer_state,
  ROUND(AVG(i.price), 2) AS mean_price,
  ROUND(SUM(i.price), 2) AS total_price,
  ROUND(AVG(i.freight_value), 2) AS mean_freight_value,
  ROUND(SUM(i.freight_value), 2) AS total_freight_value
FROM
  `target.orders` o
JOIN
  `target.order_items` i ON o.order_id = i.order_id
JOIN
  `target.customers` c ON o.customer_id = c.customer_id
GROUP BY
  c.customer_state;
```

Q] Find the month on month no. of orders placed using different payment types.



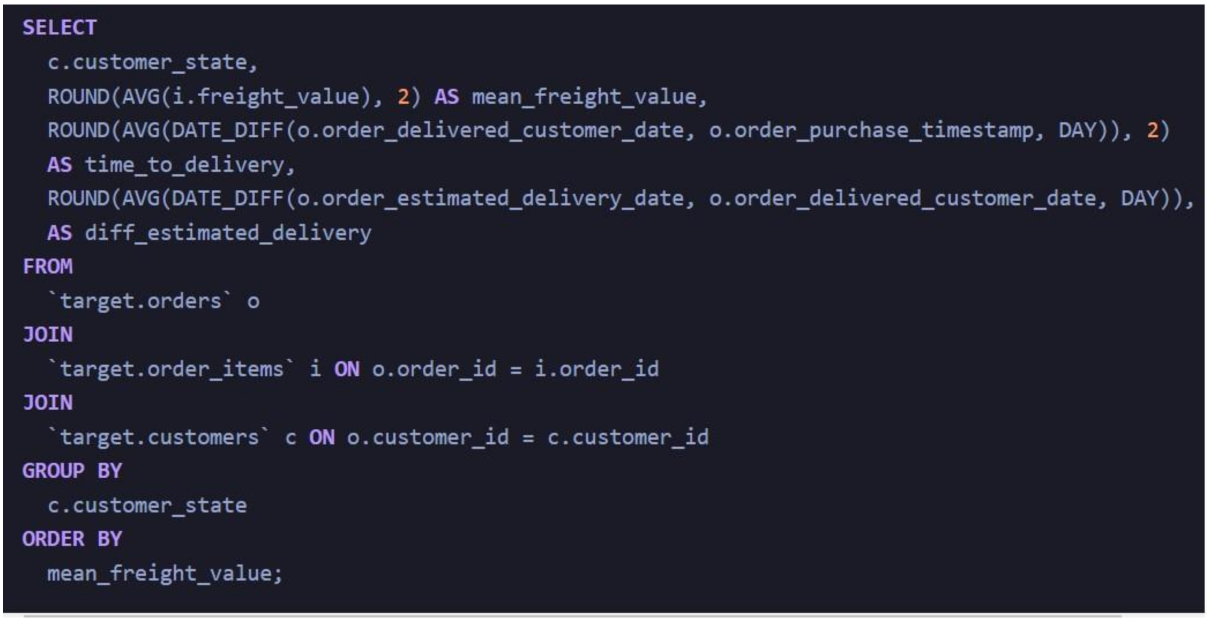
```
COUNT(order_id) order_count

FROM payment AS p
JOIN orders as o
ON p.order_id = o.order_id
GROUP BY payment_type, month, year
```

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



Row	customer_state	avg_time_to_delivery	avg_diff_estimated_delivery
1	SP	8.0	10.0
2	MG	12.0	12.0
3	PR	12.0	12.0
4	DF	13.0	11.0
5	SC	14.0	11.0
6	RJ	15.0	11.0
7	GO	15.0	11.0
8	RS	15.0	13.0
9	MS	15.0	10.0
10	ES	15.0	10.0
11	TO	17.0	11.0
12	MT	18.0	13.0
13	PE	18.0	12.0
14	BA	19.0	10.0
15	RN	19.0	13.0
16	RO	19.0	19.0
17	PI	19.0	10.0
18	PB	20.0	12.0
19	SE	21.0	9.0
20	CE	21.0	10.0
20	CE	21.0	10.0
21	MA	21.0	9.0
22	AC	21.0	20.0
23	PA	23.0	13.0
24	AL	24.0	8.0
25	AM	26.0	19.0
26	AP	27.0	19.0
27	RR	29.0	16.0

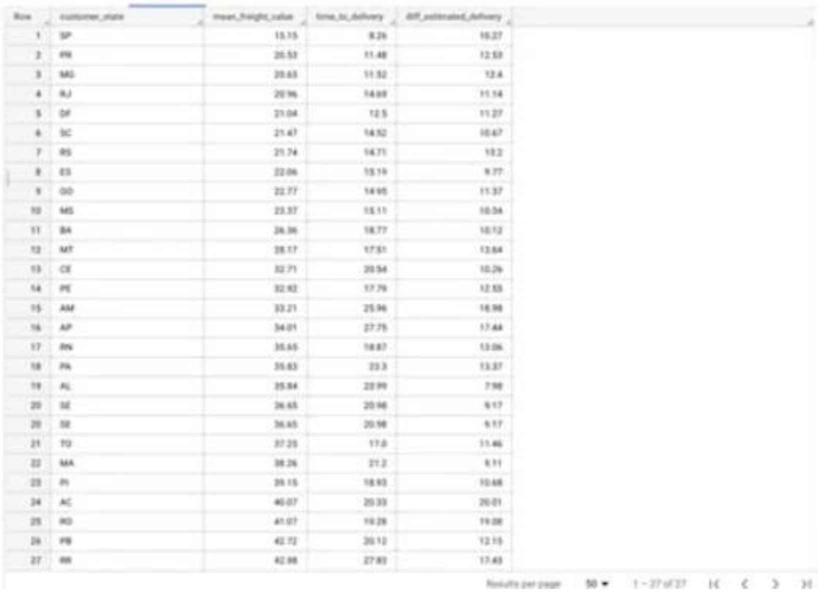
Q] Find out the top 5 states with the highest & lowest average freight value.



```
SELECT
  c.customer_state,
  ROUND(AVG(i.freight_value), 2) AS mean_freight_value,
  ROUND(AVG(DATE_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp, DAY)), 2)
  AS time_to_delivery,
  ROUND(AVG(DATE_DIFF(o.order_estimated_delivery_date, o.order_delivered_customer_date, DAY)),
  AS diff_estimated_delivery
FROM
  `target.orders` o
JOIN
  `target.order_items` i ON o.order_id = i.order_id
JOIN
  `target.customers` c ON o.customer_id = c.customer_id
GROUP BY
  c.customer_state
ORDER BY
  mean_freight_value;
```

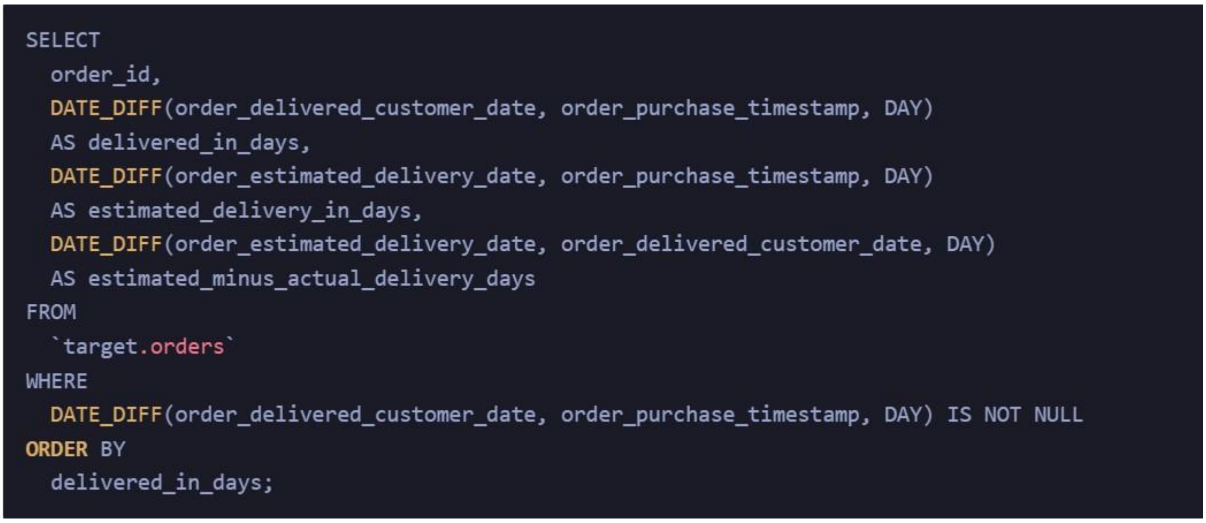
The analysis reveals a weak positive correlation between mean freight value and time to delivery. São Paulo (SP) has the lowest mean freight value, while Roraima (RR) has the highest mean freight value.

Q] Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery



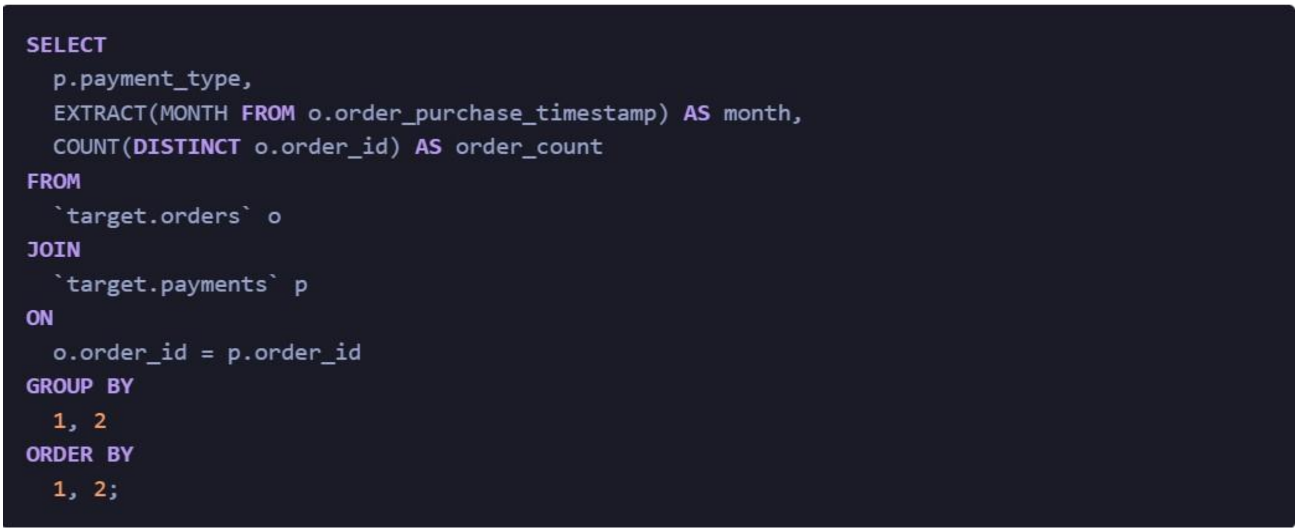
Row	customer_state	mean_freight_value	time_to_delivery	diff_estimated_delivery
1	SP	15.15	8.26	10.27
2	PR	20.53	11.48	12.53
3	MG	20.63	11.52	12.4
4	RJ	20.96	14.69	11.14
5	DF	21.04	12.5	11.27
6	SC	21.47	14.52	10.67
7	RS	21.74	14.71	13.2
8	ES	22.06	15.19	9.77
9	GO	22.77	14.95	11.37
10	MS	23.37	15.11	10.34
11	BA	26.36	18.77	10.12
12	MT	28.17	17.51	13.64
13	CE	32.71	20.54	10.26
14	PE	32.92	17.79	12.55
15	AM	33.21	25.96	18.98
16	AP	34.01	27.75	17.44
17	RN	35.65	18.87	13.06
18	PA	35.83	23.3	13.37
19	AL	35.84	23.99	7.98
20	SE	36.65	20.98	9.17
20	SE	36.65	20.98	9.17
21	TO	37.25	17.0	11.46
22	MA	38.26	21.2	9.11
23	PI	39.15	18.93	10.68
24	AC	40.07	20.33	20.01
25	RO	41.07	19.28	19.08
26	PB	42.72	20.12	12.15
27	RR	42.98	27.83	17.43

Q] Find the no. of days taken to deliver each order from the order’s purchase date as delivery time.



```
SELECT
  order_id,
  DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY)
  AS delivered_in_days,
  DATE_DIFF(order_estimated_delivery_date, order_purchase_timestamp, DAY)
  AS estimated_delivery_in_days,
  DATE_DIFF(order_estimated_delivery_date, order_delivered_customer_date, DAY)
  AS estimated_minus_actual_delivery_days
FROM
  `target.orders`
WHERE
  DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY) IS NOT NULL
ORDER BY
  delivered_in_days;
```

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



```
SELECT
  p.payment_type,
  EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
  COUNT(DISTINCT o.order_id) AS order_count
FROM
  `target.orders` o
JOIN
  `target.payments` p
ON
  o.order_id = p.order_id
GROUP BY
  1, 2
ORDER BY
  1, 2;
```

Credit card transactions are the most popular payment method, followed by UPI. Debit card transactions are the least preferred option. Notably, credit card transactions are rapidly increasing.