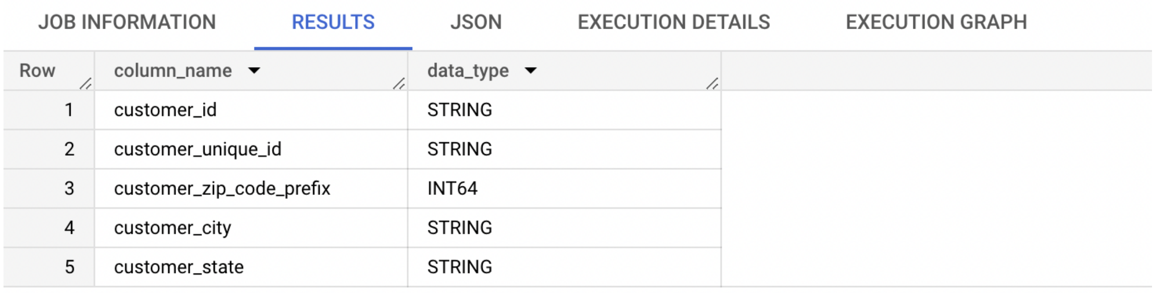
Business Case: Target SQL

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

```
SELECT
  column_name,
  data_type
FROM
  `Target.INFORMATION_SCHEMA.COLUMNS`
WHERE
  table_name = 'customers';
```

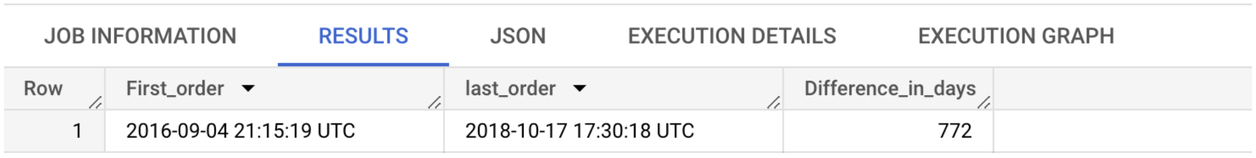


JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH

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

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

```
SELECT
 min(order_purchase_timestamp) as First_order,
 max(order_purchase_timestamp ) as last_order,
 date_diff(max(order_purchase_timestamp ),
 min(order_purchase_timestamp), day) as Difference in days'
 FROM `Target.orders`
```

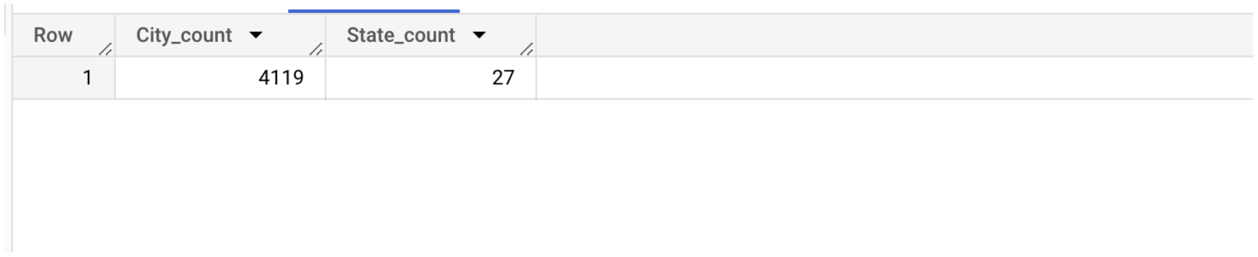


JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	First_order	last_order	Difference_in_days
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC	772

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

```
SELECT
count(distinct customer_city) as City_count,
count(distinct customer_state) as State_count
FROM
`Target.customers` as c JOIN `Target.orders` as o
 ON c.customer_id = o.customer_id
```



Row	City_count	State_count
1	4119	27

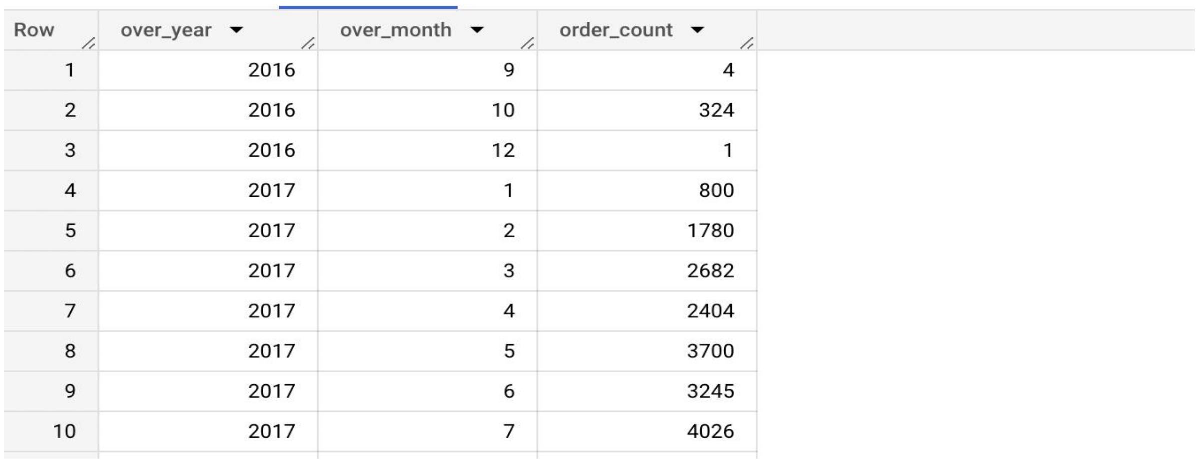
2. In-depth Exploration:

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

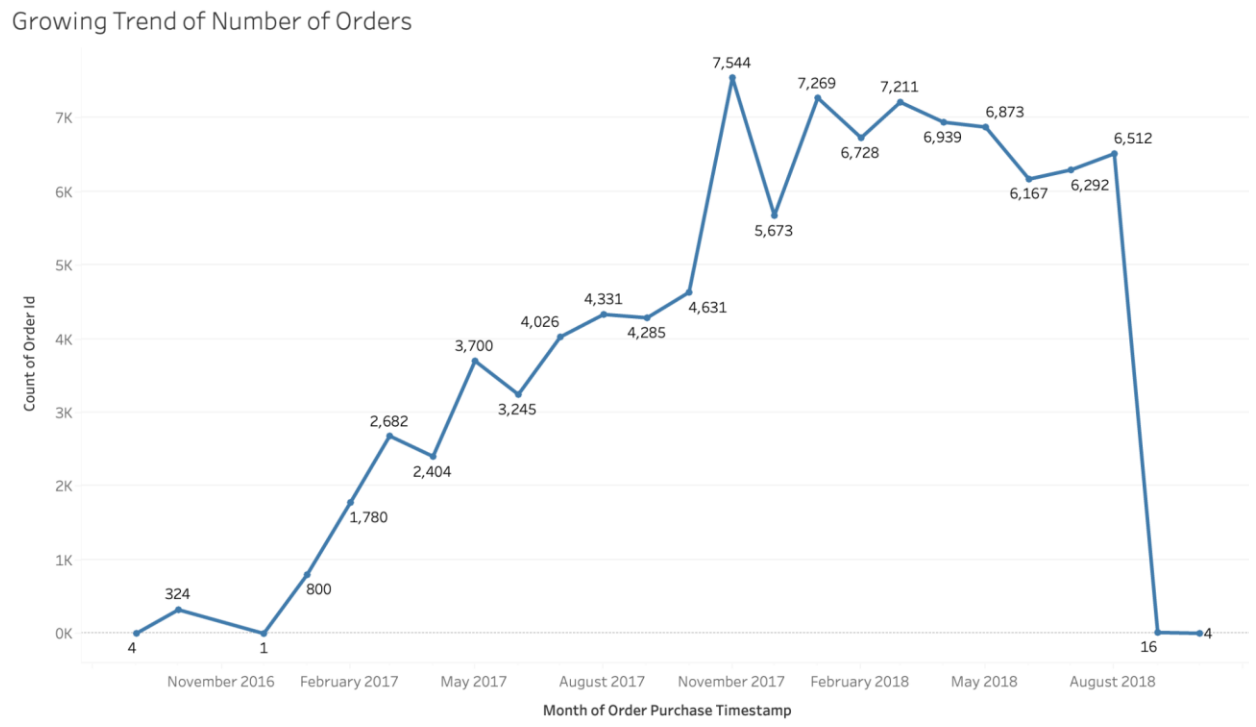
Growing trends considering year-on-year analysis for 2016, 2017, and 2018 isn't much insightful here, assuming that the question is asking month-on-month analysis.

```
 SELECT  EXTRACT( Year from order_purchase_timestamp) AS order_year,EXTRACT( month
from order_purchase_timestamp) AS  order_month, COUNT(*) AS order_count

FROM `Target.orders`
GROUP BY
order_year, order_month
ORDER BY
 order_year, order_month;
```



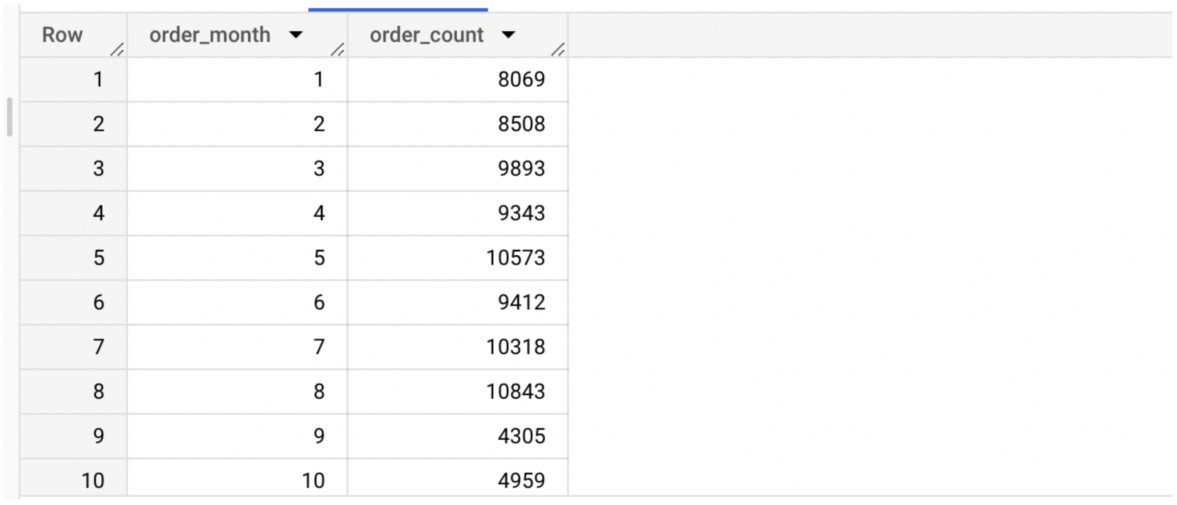
Row	over_year	over_month	order_count
1	2016	9	4
2	2016	10	324
3	2016	12	1
4	2017	1	800
5	2017	2	1780
6	2017	3	2682
7	2017	4	2404
8	2017	5	3700
9	2017	6	3245
10	2017	7	4026



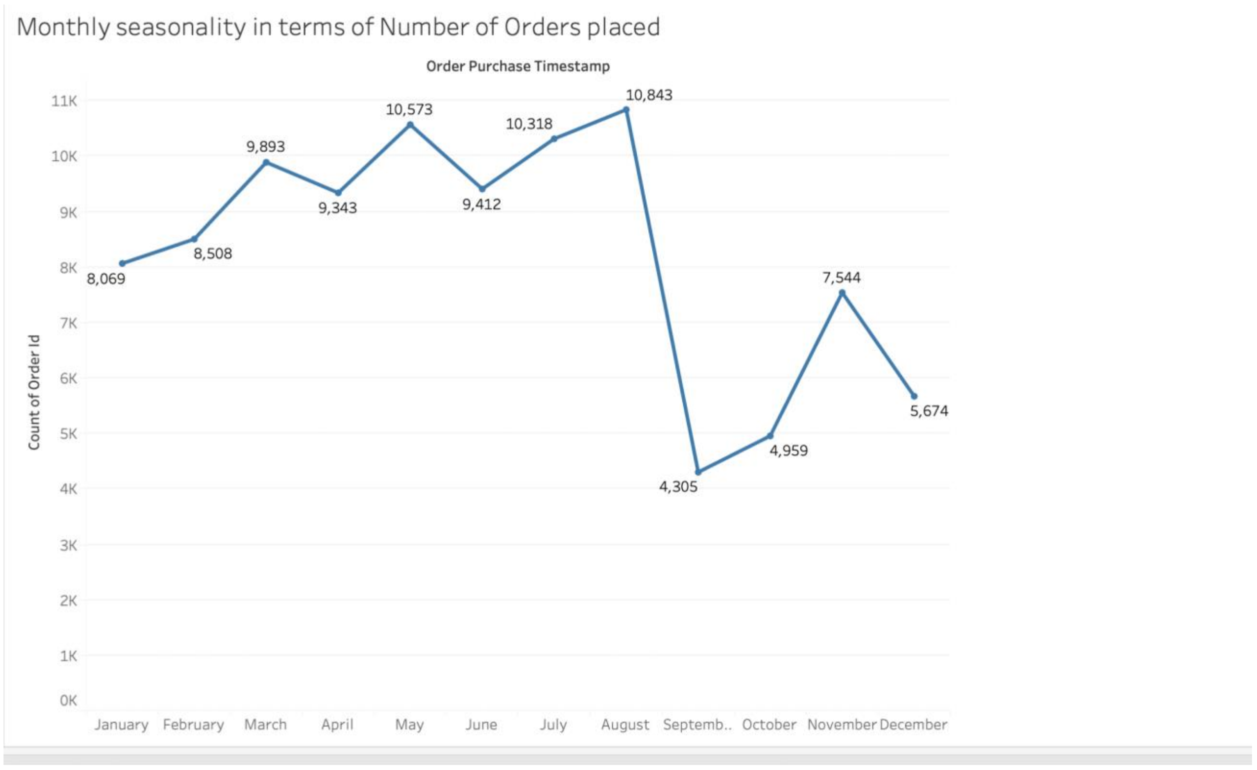
From September 2016 to November 2017, there is a general upward trend in order counts, indicating overall growth in orders placed. However, this growth trend was disrupted in December 2017, and the orders count dropped, recovered back from January 2018 and was stable and dropped drastically again in September 2018.

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

```
SELECT
EXTRACT( month from order_purchase_timestamp) AS order_month,
COUNT(*) AS order_count
FROM `Target.orders`
GROUP BY  order_month
ORDER BY
Order_month;
```



Row	order_month	order_count
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

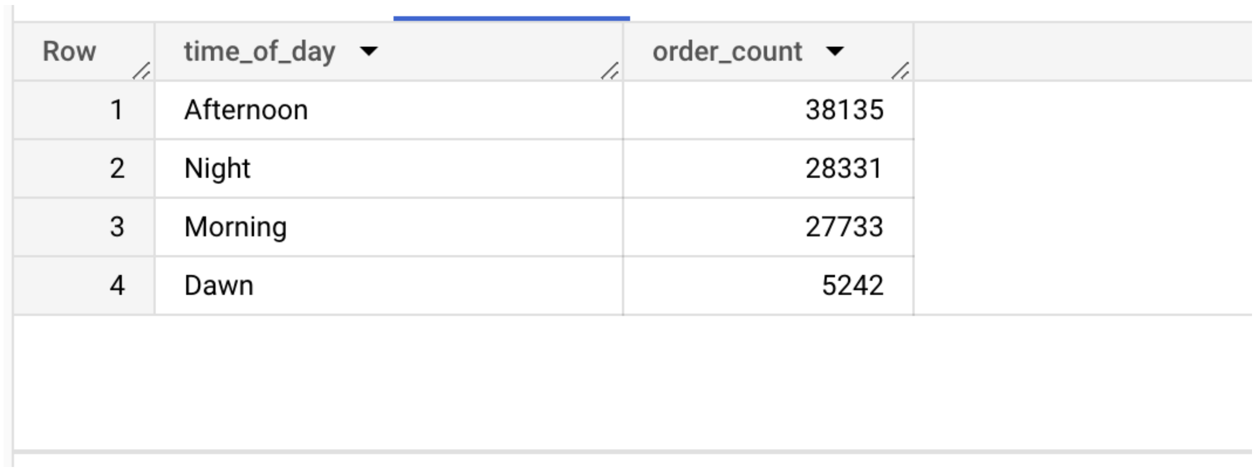


There is a general trend of higher order counts in the middle months of the year (May, June, July, August), possibly due to summer holiday shopping, indicating that these months might be popular for shopping.

September has the lowest order count in this dataset. It might be worth investigating potential reasons for the sudden and significant drop in orders during this month.

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

```
    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
'Afternoon'
         ELSE 'Night'
    END AS time_of_day,
    COUNT(*) AS order_count
FROM
    `Target.orders`
GROUP BY
    time_of_day
ORDER BY
    order_count DESC;
```



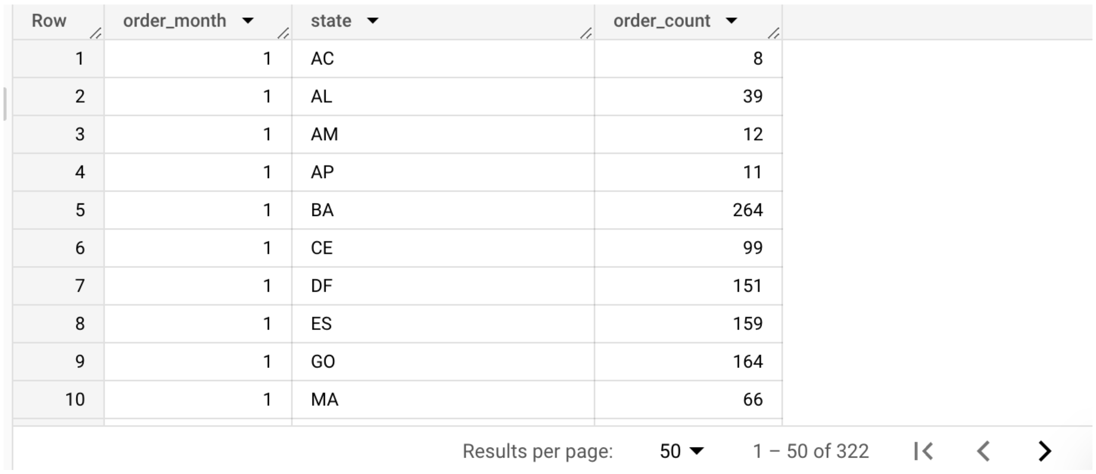
Row	time_of_day	order_count
1	Afternoon	38135
2	Night	28331
3	Morning	27733
4	Dawn	5242

Most orders were placed during the afternoon followed by Night.

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

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

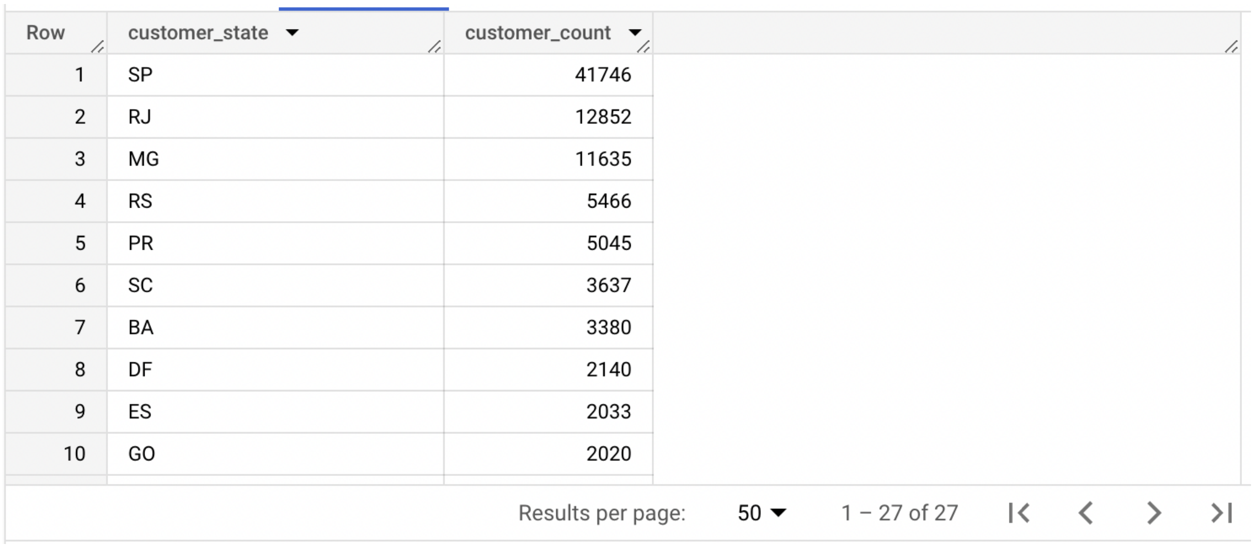
```
SELECT
   EXTRACT(MONTH FROM o.order_purchase_timestamp) AS order_month,
   c.customer_state as state,
   COUNT(*) AS order_count
FROM
   `Target.orders`  as o  JOIN `Target.customers` as c
   ON o.customer_id = c.customer_id
GROUP BY
   order_month,
   state
ORDER BY
   order_month,
   state,
   Order_count;
```



Row	order_month	state	order_count
1	1	AC	8
2	1	AL	39
3	1	AM	12
4	1	AP	11
5	1	BA	264
6	1	CE	99
7	1	DF	151
8	1	ES	159
9	1	GO	164
10	1	MA	66

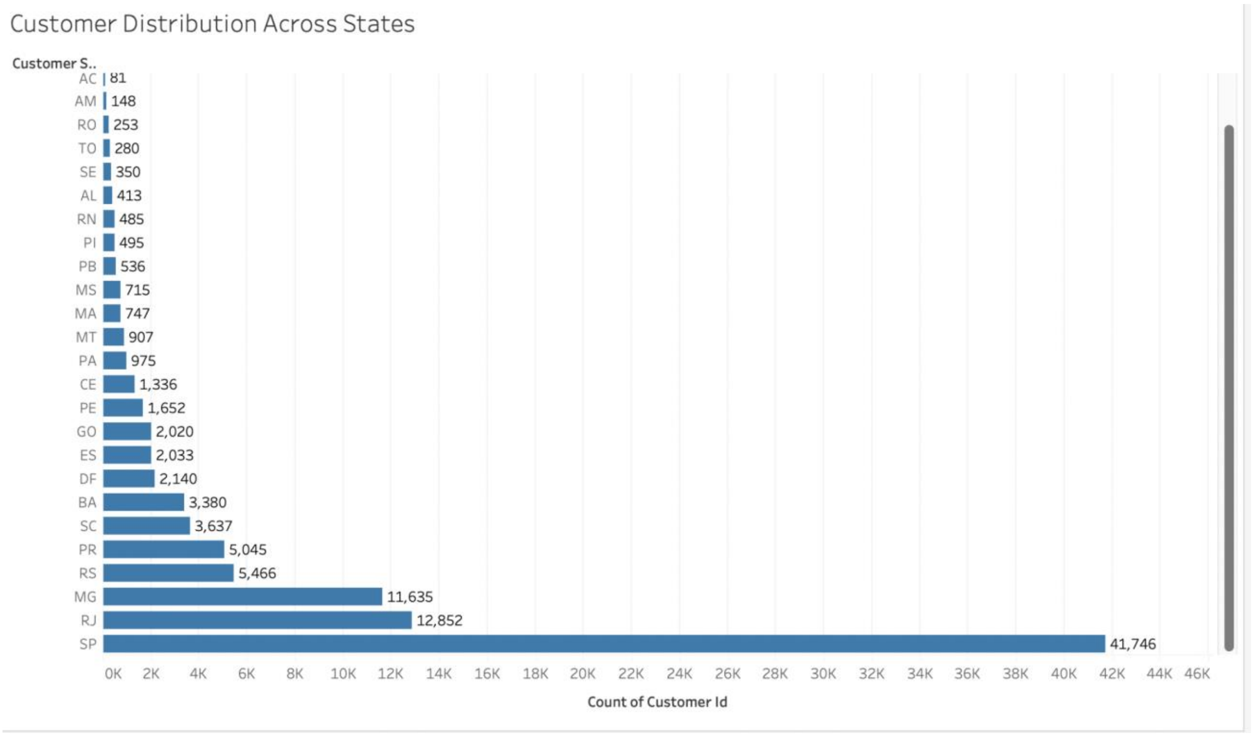
2. How are the customers distributed across all the states?

```
SELECT customer_state, count(*) as customer_count
FROM `Target.customers`
GROUP BY customer_state
ORDER BY  customer_count desc
```



Row	customer_state	customer_count
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

Results per page: 50 1 – 27 of 27



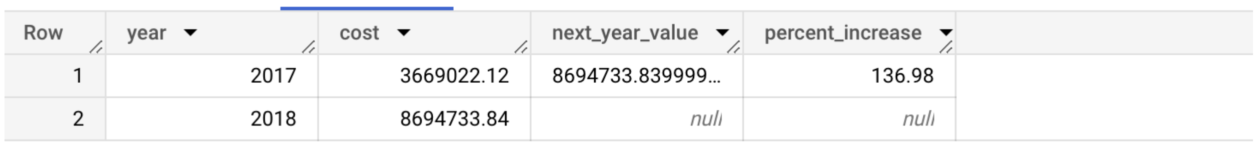
4. Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight and others.

1. 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.

```
with CTE as
(SELECT *
    FROM
   `Target.payments` AS p JOIN `Target.orders` as o
   ON p.order_id = o.order_id
   WHERE EXTRACT(month from o.order_purchase_timestamp) BETWEEN 1 AND 8
   AND EXTRACT(year from o.order_purchase_timestamp) BETWEEN 2017 AND 2018

 ), CTE2 AS(SELECT EXTRACT(year from order_purchase_timestamp) AS year,
   SUM(payment_value) as cost,
   FROM CTE
   GROUP BY year
   ORDER BY year
 )

 SELECT year, round(cost,2) as cost, LEAD(cost) over(order by year) as next_year_value,
 round(((LEAD(cost) over(order by year)-cost)/cost)*100,2) as percent_increase from CTE2
```



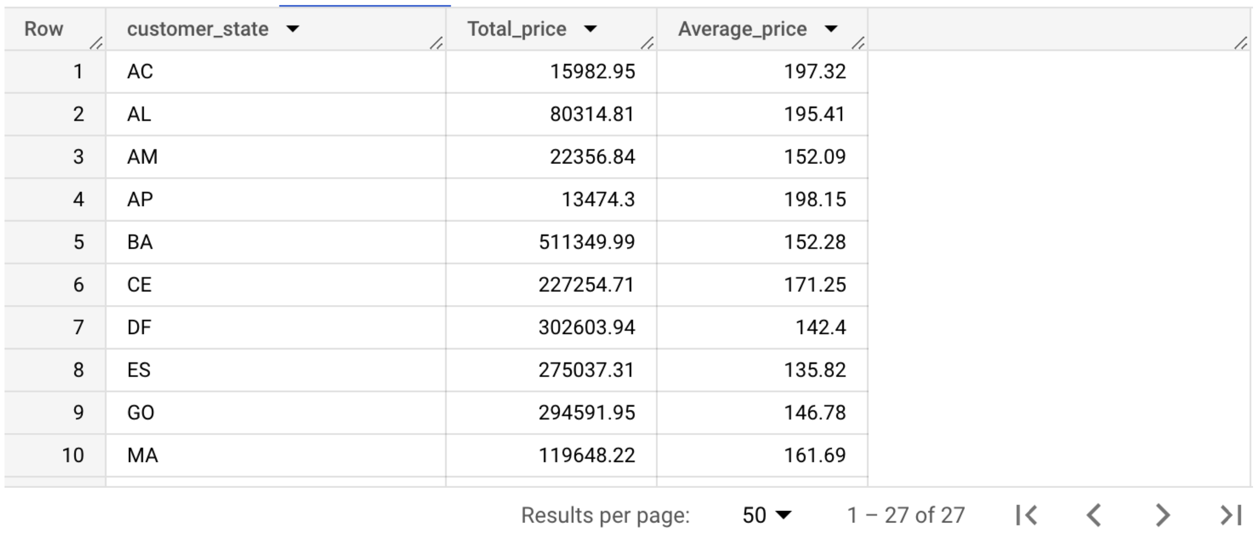
Row	year	cost	next_year_value	percent_increase
1	2017	3669022.12	8694733.839999...	136.98
2	2018	8694733.84	null	null

There is a 136.98 % increase in the cost of orders from 2017 to 2018.

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

```
SELECT
 c.customer_state,
 round(SUM(oil.price),2) as Total_price,
  round(sum(oil.price)/count(distinct oil.order_id),2) as Average_price
  FROM `
  Target.customers` as c JOIN `Target.orders` as o
  ON c.customer_id = o.customer_id
  JOIN `Target.order_items` as oil
  ON oil.order_id = o.order_id
  GROUP BY
    c.customer_state
  ORDER BY
```

```
c.customer_state;
```

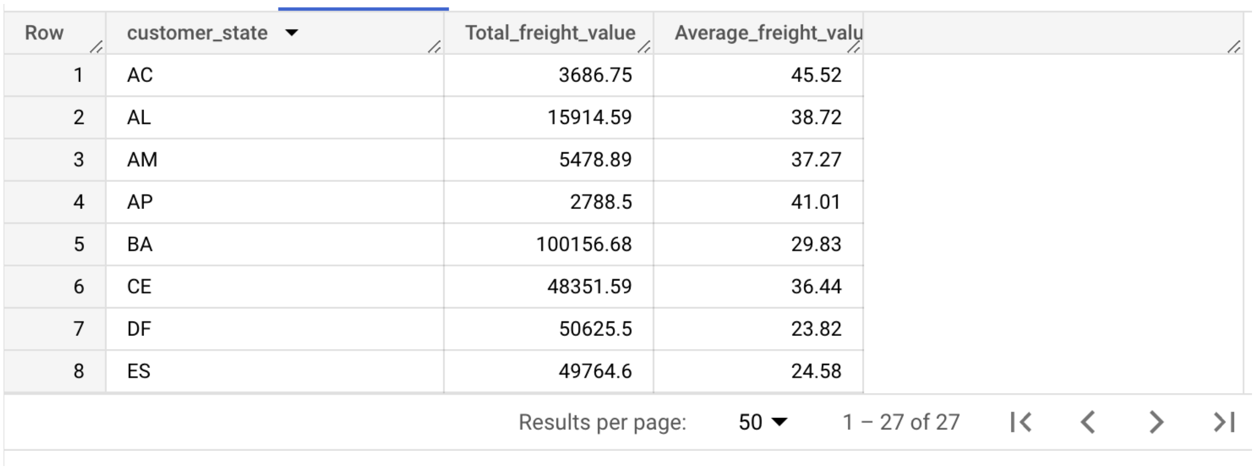


Row	customer_state	Total_price	Average_price
1	AC	15982.95	197.32
2	AL	80314.81	195.41
3	AM	22356.84	152.09
4	AP	13474.3	198.15
5	BA	511349.99	152.28
6	CE	227254.71	171.25
7	DF	302603.94	142.4
8	ES	275037.31	135.82
9	GO	294591.95	146.78
10	MA	119648.22	161.69

Results per page: 50 1 – 27 of 27

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

```
SELECT
C.customer_state,
 round(SUM(oil.freight_value),2) as Total_freight_value,
 round(sum(oil.freight_value)/count(distinct oil.order_id),2) as Average_freight_value
 FROM
`Target.customers` as c JOIN `Target.orders` as o
  ON c.customer_id = o.customer_id
  JOIN `Target.order_items` as oil
  ON oil.order_id = o.order_id
  GROUP BY c.customer_state
  ORDER BY c.customer_state;
```



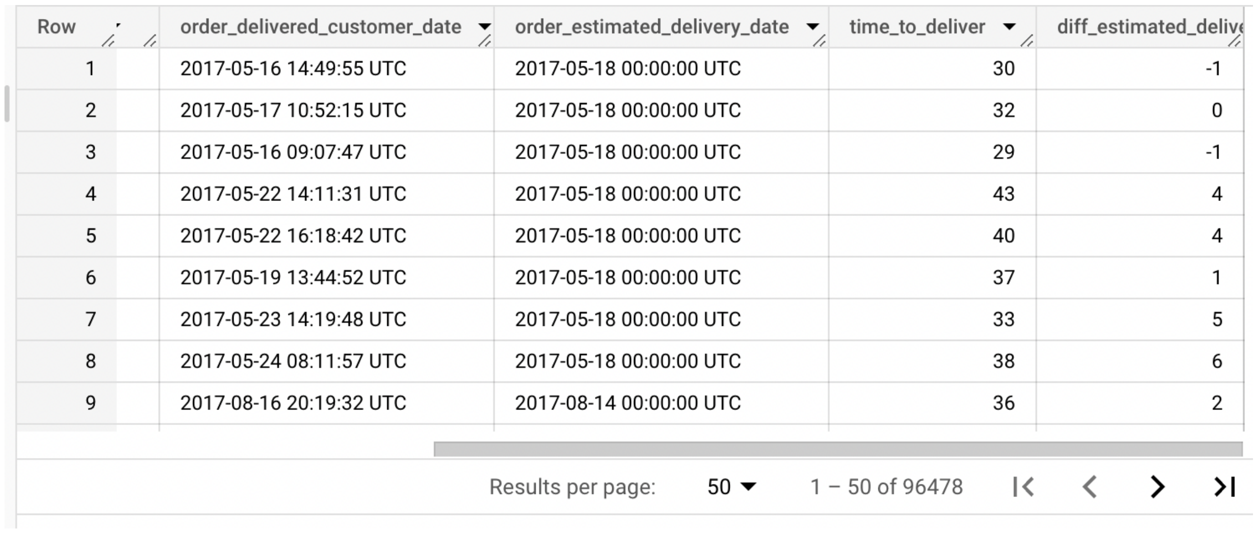
Row	customer_state	Total_freight_value	Average_freight_valu
1	AC	3686.75	45.52
2	AL	15914.59	38.72
3	AM	5478.89	37.27
4	AP	2788.5	41.01
5	BA	100156.68	29.83
6	CE	48351.59	36.44
7	DF	50625.5	23.82
8	ES	49764.6	24.58

Results per page: 50 1 – 27 of 27

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

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.
   You can calculate the delivery time and the difference between the estimated & actual delivery date using the given formula:
   - **time_to_deliver** = order_delivered_customer_date - order_purchase_timestamp
     - **diff_estimated_delivery** = order_estimated_delivery_date - order_delivered_customer_date

```
SELECT
order_id,
order_purchase_timestamp,
order_delivered_customer_date,
order_estimated_delivery_date,
timestamp_diff(order_delivered_customer_date, order_purchase_timestamp, day) as
time_to_deliver,
timestamp_diff( order_delivered_customer_date,order_estimated_delivery_date, day) as
diff_estimated_delivery
FROM `Target.orders`
where order_status = "delivered"
```



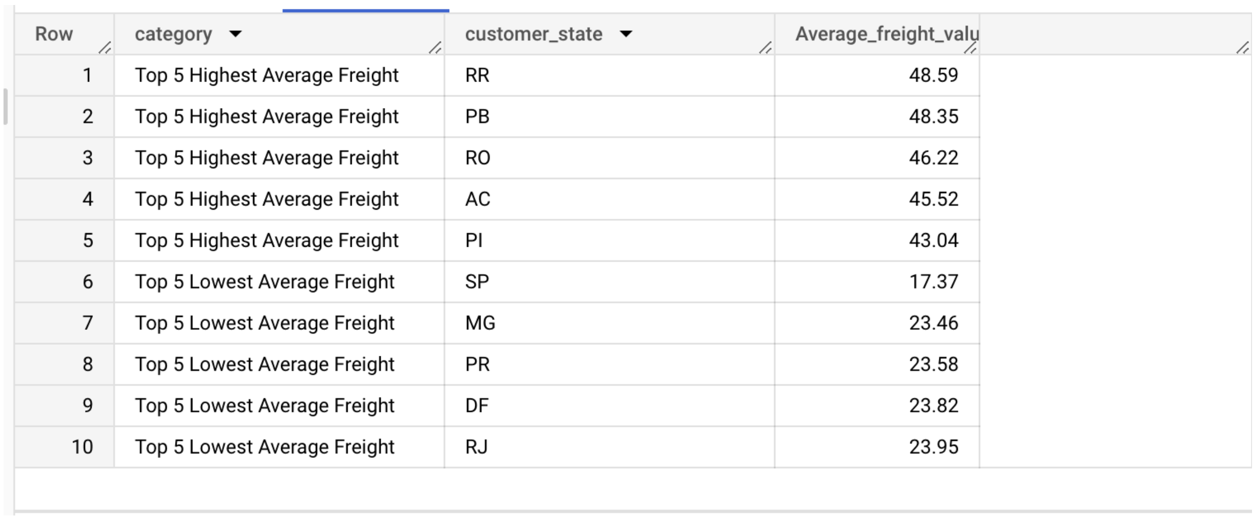
Row	order_delivered_customer_date	order_estimated_delivery_date	time_to_deliver	diff_estimated_delive
1	2017-05-16 14:49:55 UTC	2017-05-18 00:00:00 UTC	30	-1
2	2017-05-17 10:52:15 UTC	2017-05-18 00:00:00 UTC	32	0
3	2017-05-16 09:07:47 UTC	2017-05-18 00:00:00 UTC	29	-1
4	2017-05-22 14:11:31 UTC	2017-05-18 00:00:00 UTC	43	4
5	2017-05-22 16:18:42 UTC	2017-05-18 00:00:00 UTC	40	4
6	2017-05-19 13:44:52 UTC	2017-05-18 00:00:00 UTC	37	1
7	2017-05-23 14:19:48 UTC	2017-05-18 00:00:00 UTC	33	5
8	2017-05-24 08:11:57 UTC	2017-05-18 00:00:00 UTC	38	6
9	2017-08-16 20:19:32 UTC	2017-08-14 00:00:00 UTC	36	2

Results per page: 50 1 – 50 of 96478

87182 orders out of 96478 were delivered before the estimated delivery date.

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

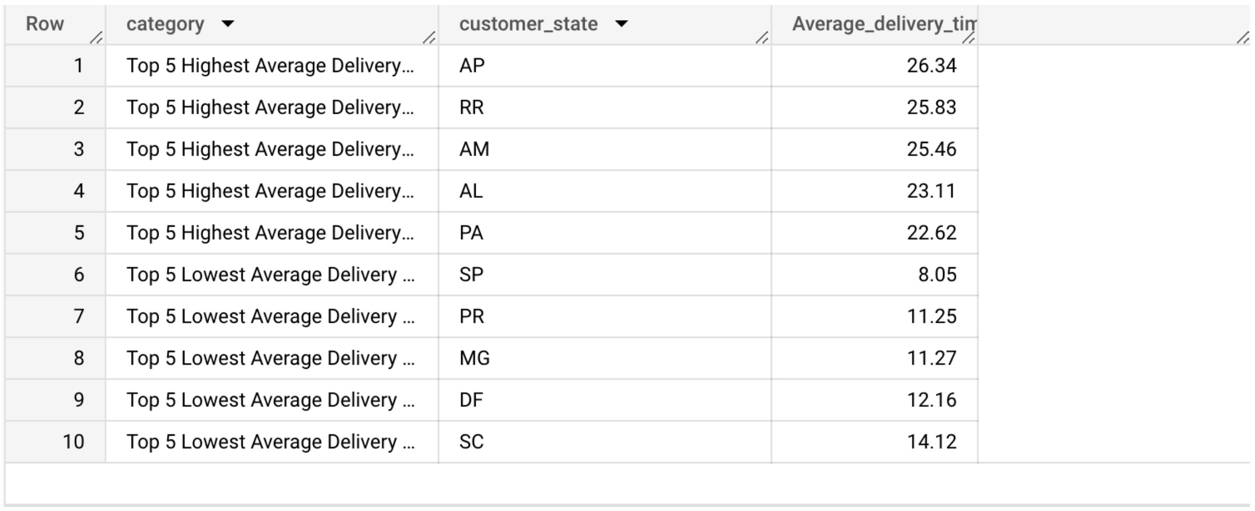
```
with CTE as
(SELECT c.customer_state,
round(sum(oil.freight_value)/count(distinct oil.order_id),2) as Average_freight_value
  FROM `Target.customers` as c JOIN `Target.orders` as o
  ON c.customer_id = o.customer_id
  JOIN `Target.order_items` as oil
  ON oil.order_id = o.order_id
  GROUP BY c.customer_state
 )
(SELECT 'Top 5 Lowest Average Freight' AS category, * from CTE
 order by average_freight_value
 limit 5)
UNION ALL
(SELECT 'Top 5 Highest Average Freight' AS category, * from CTE
 order by average_freight_value desc
 limit 5)
```



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

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

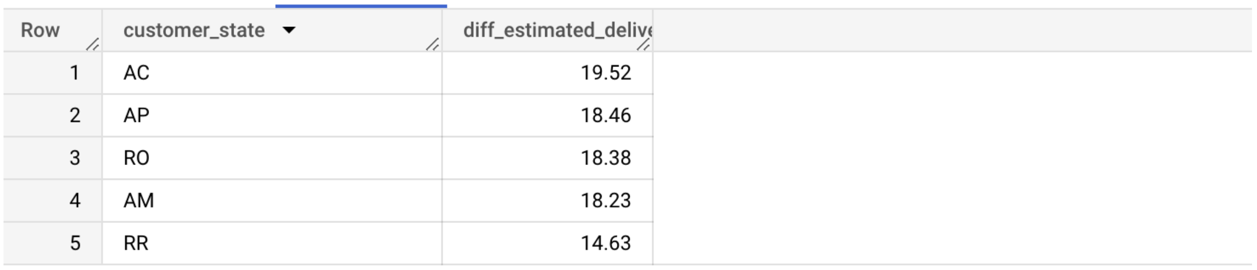
```
with CTE as
(SELECT c.customer_state, round(sum(timestamp_diff(order_delivered_customer_date ,
order_purchase_timestamp, day))/count(distinct c.customer_id),2)  as Average_delivery_time,
  FROM `Target.customers` as c JOIN `Target.orders` as o
  ON c.customer_id = o.customer_id
  GROUP BY c.customer_state)
  (SELECT 'Top 5 Lowest Average Delivery time' as category, * FROM CTE
  ORDER BY CTE.Average_delivery_time
  LIMIT 5)
  UNION ALL
  (SELECT 'Top 5 Highest Average Delivery time' as category, * FROM CTE
  ORDER BY CTE.Average_delivery_time  desc
  LIMIT 5)
```



Row	category	customer_state	Average_delivery_tim
1	Top 5 Highest Average Delivery...	AP	26.34
2	Top 5 Highest Average Delivery...	RR	25.83
3	Top 5 Highest Average Delivery...	AM	25.46
4	Top 5 Highest Average Delivery...	AL	23.11
5	Top 5 Highest Average Delivery...	PA	22.62
6	Top 5 Lowest Average Delivery ...	SP	8.05
7	Top 5 Lowest Average Delivery ...	PR	11.25
8	Top 5 Lowest Average Delivery ...	MG	11.27
9	Top 5 Lowest Average Delivery ...	DF	12.16
10	Top 5 Lowest Average Delivery ...	SC	14.12

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.

```
SELECT c.customer_state,
round(sum(timestamp_diff(order_estimated_delivery_date, order_delivered_customer_date, day))/count(distinct c.customer_id),2) as diff_estimated_delivery
FROM
`Target.customers` as c JOIN `Target.orders` as o
ON c.customer_id = o.customer_id
GROUP BY c.customer_state
ORDER BY diff_estimated_delivery DESC
LIMIT 5
```

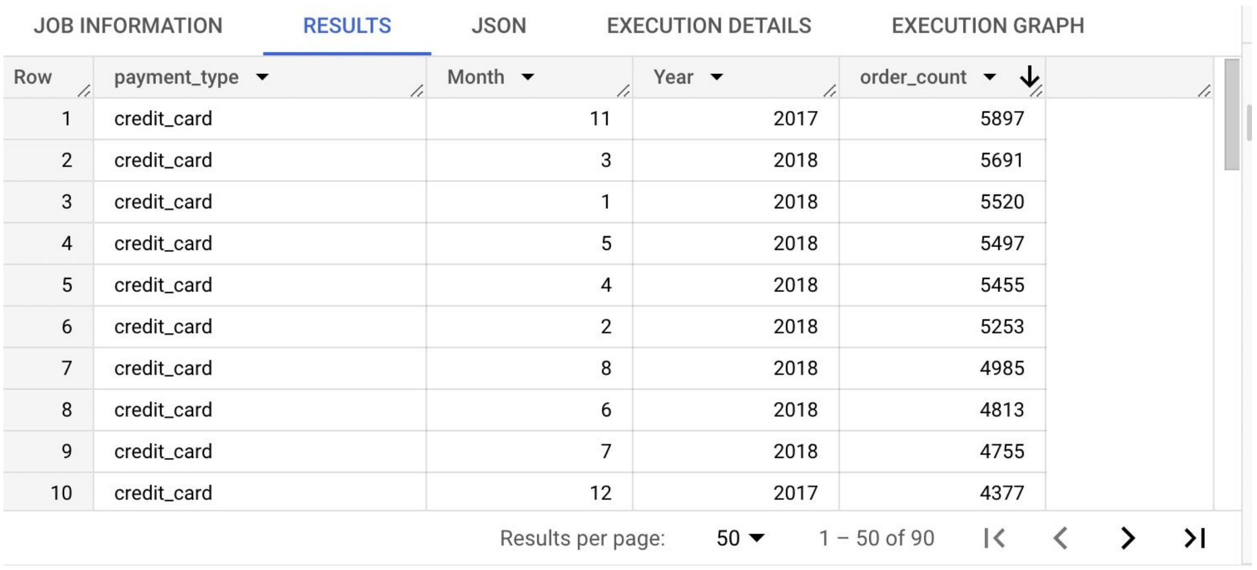


Row	customer_state	diff_estimated_delive
1	AC	19.52
2	AP	18.46
3	RO	18.38
4	AM	18.23
5	RR	14.63

6. Analysis based on the payments:

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

```
SELECT p.payment_type,
EXTRACT(Month from o.order_purchase_timestamp) as Month,
EXTRACT(Year from o.order_purchase_timestamp) as Year,
count(*) as order_count
from `Target.payments` as p JOIN `Target.orders` as o
ON p.order_id = o.order_id
group by p.payment_type, Year, Month
```


JOB INFORMATION RESULTS JSON EXECUTION DETAILS EXECUTION GRAPH

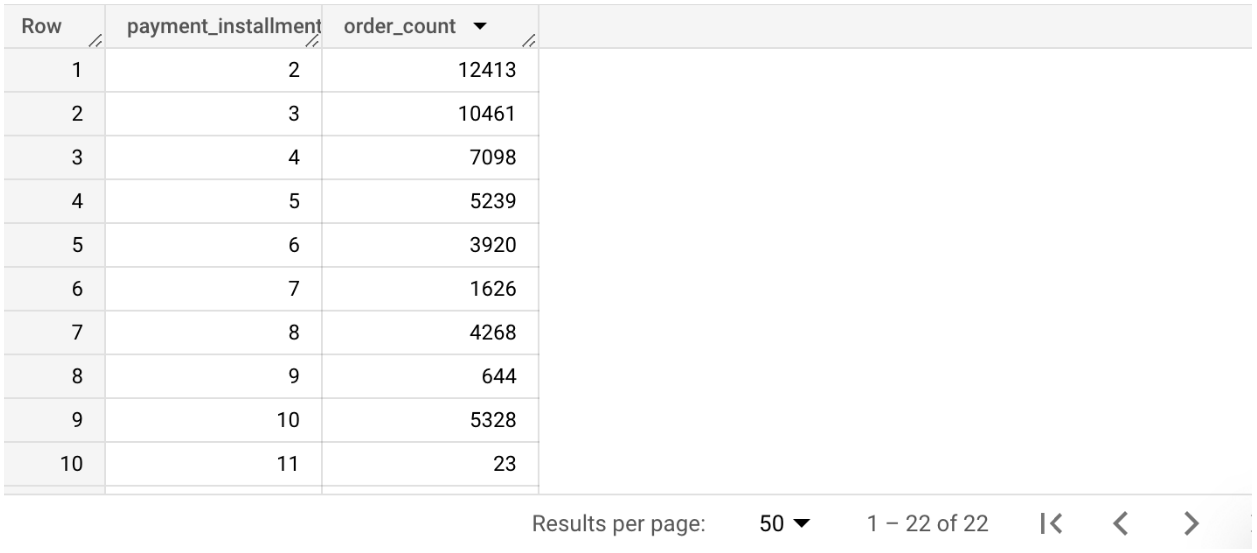
Row	payment_type	Month	Year	order_count
1	credit_card	11	2017	5897
2	credit_card	3	2018	5691
3	credit_card	1	2018	5520
4	credit_card	5	2018	5497
5	credit_card	4	2018	5455
6	credit_card	2	2018	5253
7	credit_card	8	2018	4985
8	credit_card	6	2018	4813
9	credit_card	7	2018	4755
10	credit_card	12	2017	4377

Results per page: 50 1 – 50 of 90

Most used payment method was credit_card followed by UPI.

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

```
SELECT payment_installments,
count(*) as order_count
from `Target.payments`
WHERE payment_installments > 1
GROUP BY payment_installments
```



Row	payment_installment	order_count
1	2	12413
2	3	10461
3	4	7098
4	5	5239
5	6	3920
6	7	1626
7	8	4268
8	9	644
9	10	5328
10	11	23

Results per page: 50 1 – 22 of 22

****End****