TARGET-SQL BUSINESS CASE STUDY

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?

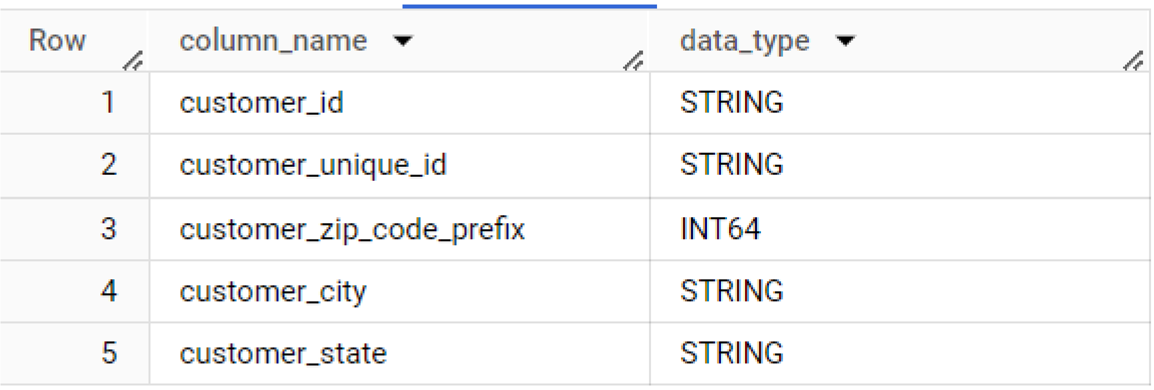
1. Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset:

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

QUERY:

```
select column_name, data_type
from target_sql.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

INSIGHTS: All the columns are of ‘STRING’ data type except customer_zip_code_prefix which is of ‘INT’ data type.

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

QUERY:

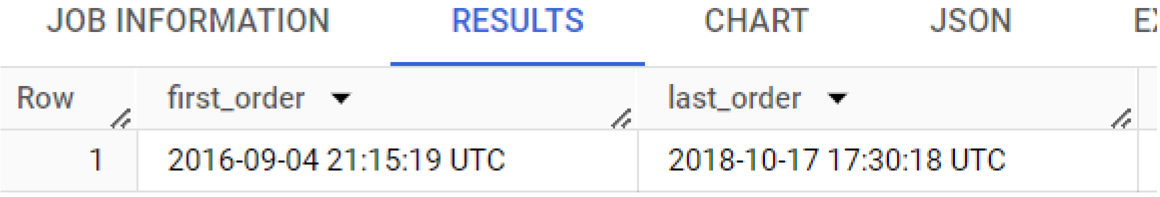
```
select

min(order_purchase_timestamp) as first_order,

max(order_purchase_timestamp) as last_order

from target_sql.orders;
```

OUTPUT:



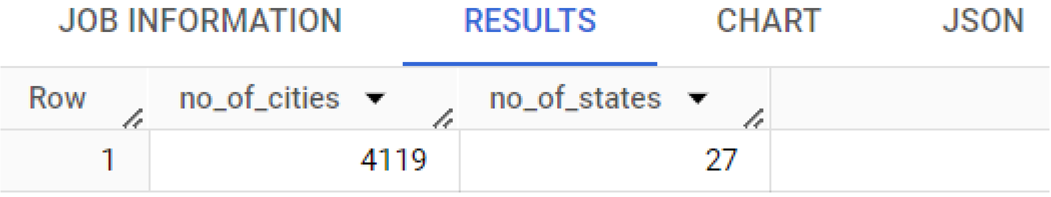
Row	first_order	last_order
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC

INSIGHTS: The first order was placed on 4th Sep 2016 and the last order was placed on 17th Oct 2018.

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

```
select
count(distinct customer_city) as no_of_cities,
count(distinct customer_state) as no_of_states
from target_sql.customers as c
inner join target_sql.orders as o
on o.customer_id = c.customer_id;
```

OUTPUT:



Row	no_of_cities	no_of_states
1	4119	27

INSIGHTS:

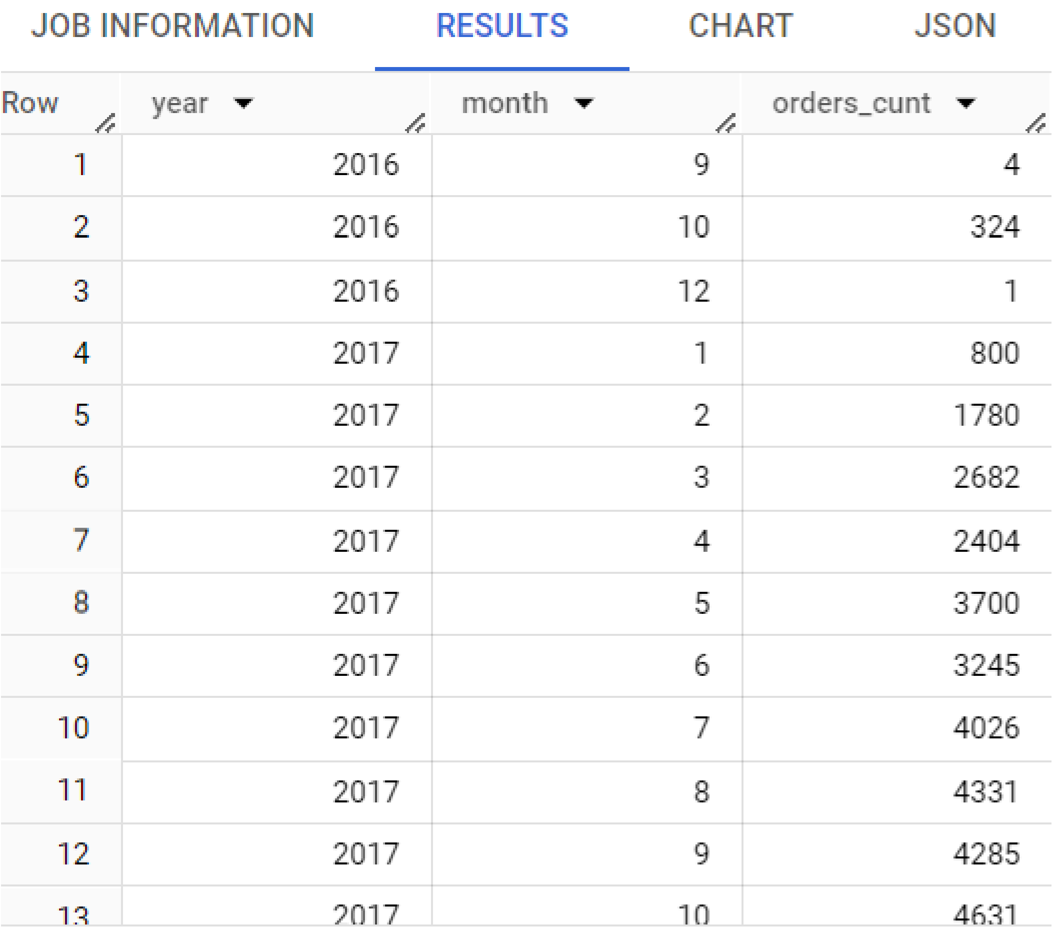
There are 4119 different cities and 27 different states of customers who ordered during the given period.

2. In-depth Exploration:

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

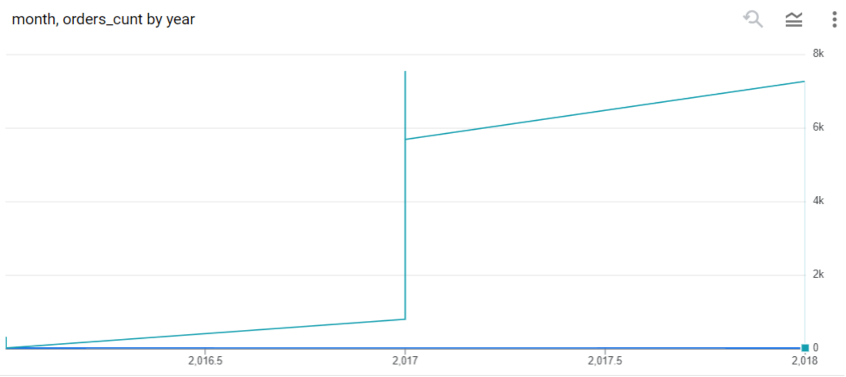
ANS:

```
select
extract(year from order_purchase_timestamp) as year,
extract(month from order_purchase_timestamp) as month,
count(*) as orders_cunt
from target_sql.orders
group by 1,2
order by 1,2;
```



Row	year	month	orders_cunt
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
11	2017	8	4331
12	2017	9	4285
13	2017	10	4631

INSIGHTS:

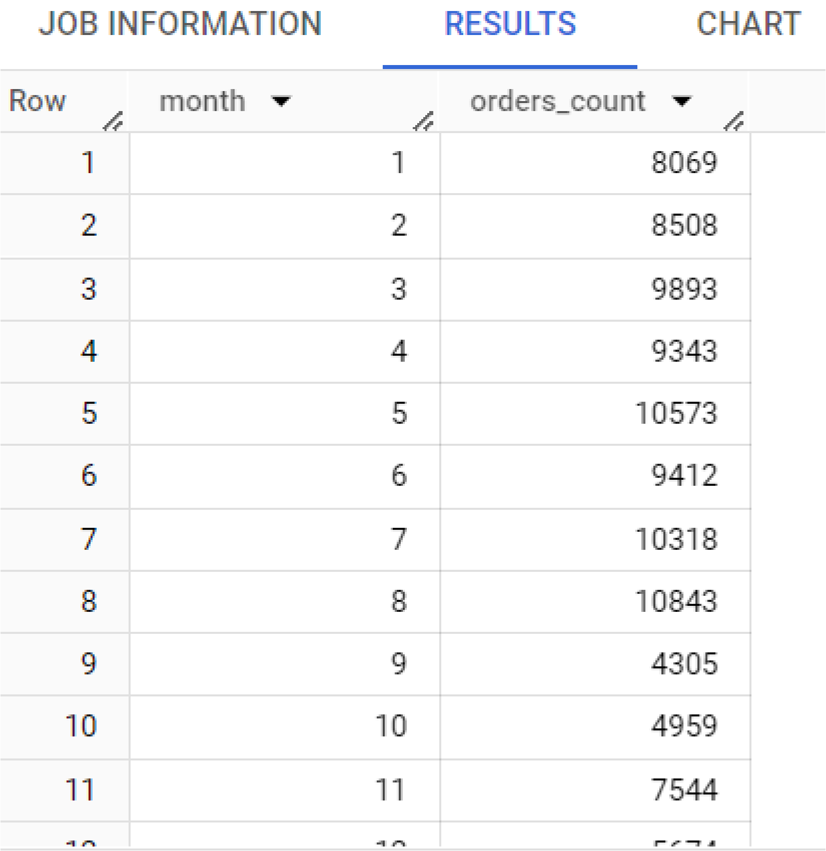


From the order_year 2016 to 2017 the order count has been increased rapidly in a very huge number i.e from 329 to 45k. Whereas from the order_year 2017 to 2018 also we can see an increase but not as huge as the previous year's increment.

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

ANS:

```
select
extract(month from order_purchase_timestamp) as month,
count(*) as orders_count
from target_sql.orders
group by 1
order by 1;
```



Row	month	orders_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
11	11	7544
12	12	5674

INSIGHTS:

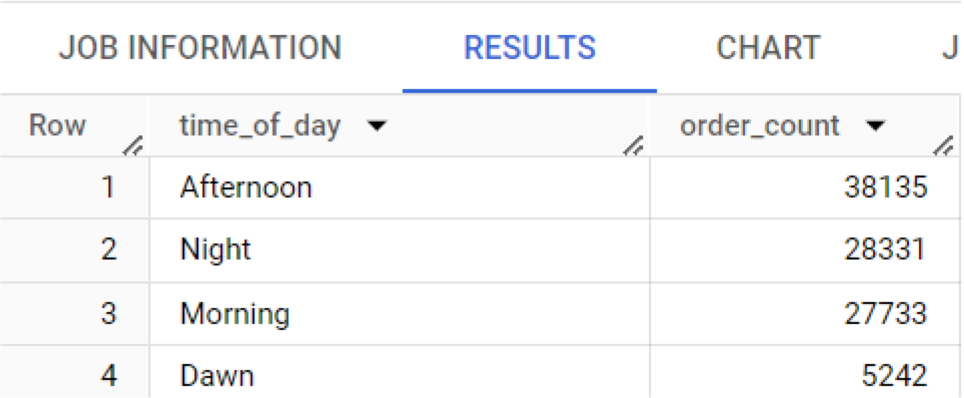
The order_count is changing year by year which is not linear but overall we can say that there are more number of orders taken place in the year 2017 from the 5th month to 2018 8th month.

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**

ANS:

```
WITH order_hours AS (
  SELECT
    EXTRACT(HOUR FROM order_purchase_timestamp) AS order_hour
  FROM
    target_sql.orders
)
SELECT
  CASE
    WHEN order_hour >= 0 AND order_hour < 7 THEN 'Dawn'
    WHEN order_hour >= 7 AND order_hour < 13 THEN 'Morning'
    WHEN order_hour >= 13 AND order_hour < 19 THEN 'Afternoon'
    ELSE 'Night'
  END AS time_of_day,
  COUNT(*) AS order_count
FROM
  order_hours
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

INSIGHTS:

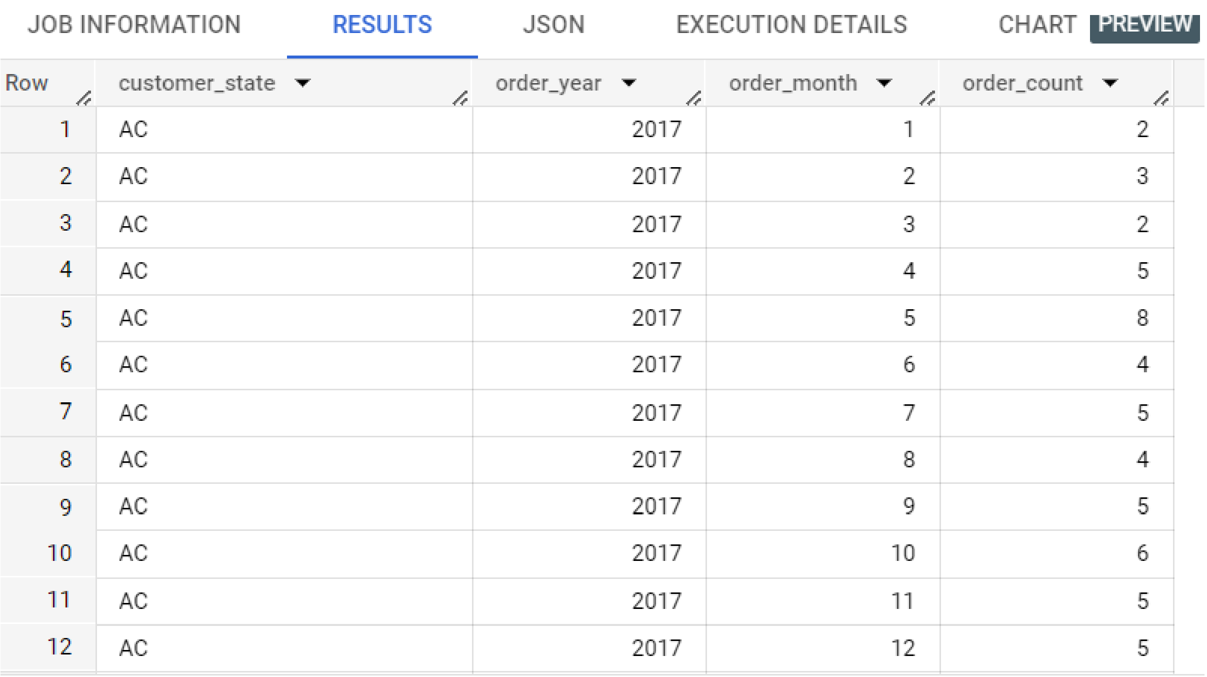
We can clearly observe that during Afternoon time the order count is quite high as compared to other timings.
Early hours of the day is having very less amount of orders.

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

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

ANS:

```
WITH order_monthly AS (
  SELECT
    EXTRACT(YEAR FROM o.order_purchase_timestamp) AS order_year,
    EXTRACT(MONTH FROM o.order_purchase_timestamp) AS order_month,
    c.customer_state,
    COUNT(*) AS order_count
  FROM
    `target_sql.orders` AS o
  JOIN
    `target_sql.customers` AS c
  ON
    o.customer_id = c.customer_id
  GROUP BY
    order_year, order_month, customer_state
)
SELECT
  customer_state,
  order_year,
  order_month,
  order_count
FROM
  order_monthly
ORDER BY
  customer_state, order_year, order_month;
```



Row	customer_state	order_year	order_month	order_count
1	AC	2017	1	2
2	AC	2017	2	3
3	AC	2017	3	2
4	AC	2017	4	5
5	AC	2017	5	8
6	AC	2017	6	4
7	AC	2017	7	5
8	AC	2017	8	4
9	AC	2017	9	5
10	AC	2017	10	6
11	AC	2017	11	5
12	AC	2017	12	5

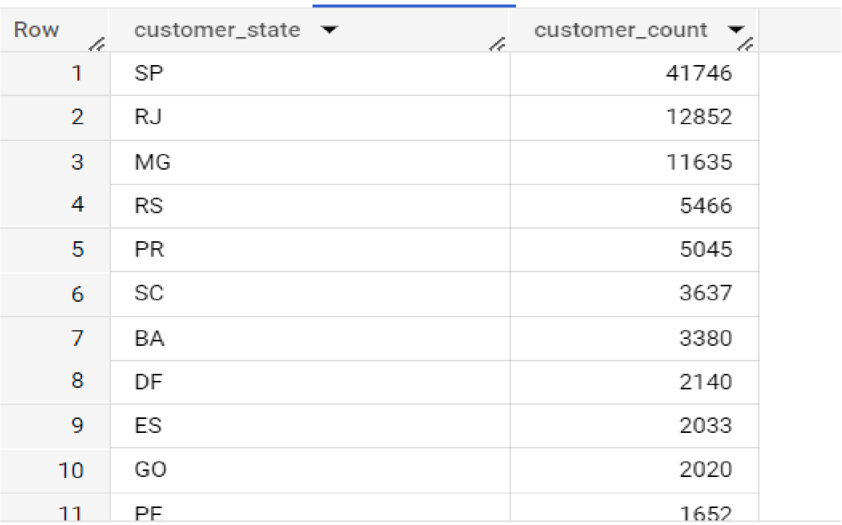
INSIGHTS:

The state AL is having more number of orders placed compared to other states with respect to month on month and also year on year.

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

ANS:

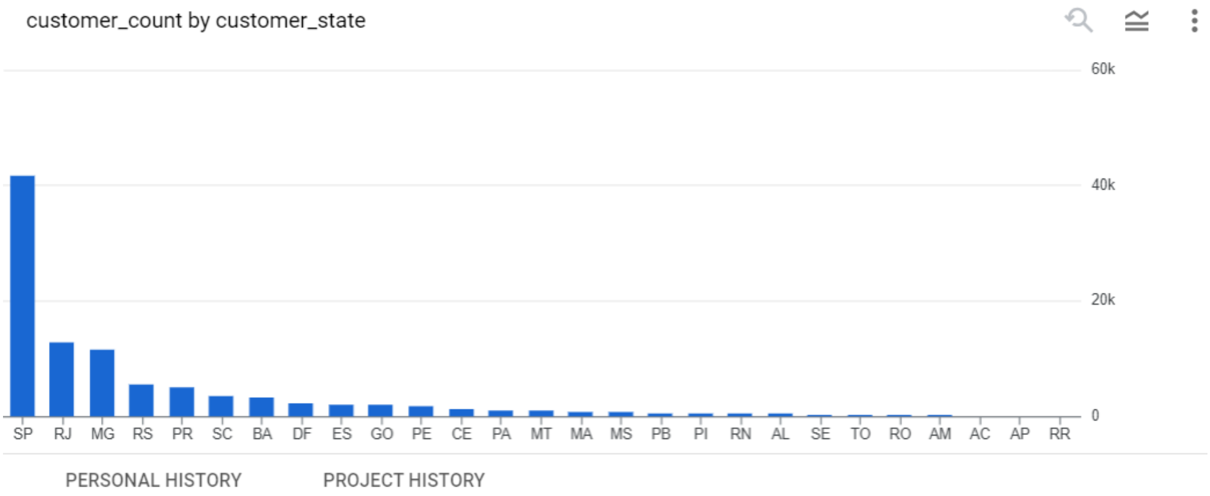
```
SELECT
  customer_state,
  COUNT(DISTINCT customer_id) AS customer_count
FROM
  `target_sql.customers`
GROUP BY
  customer_state
ORDER BY
  customer_count DESC;
```



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
11	PE	1652

INSIGHTS:

As we can see each state having different number of customers. Highest number of customers are from the states ‘SP’, ‘RJ’, and ‘MG’ in descending order respectively.



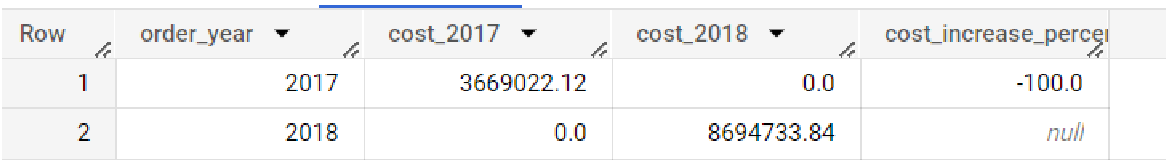
4. Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight and others.

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.

ANS:

```
WITH order_costs AS (
  SELECT
```

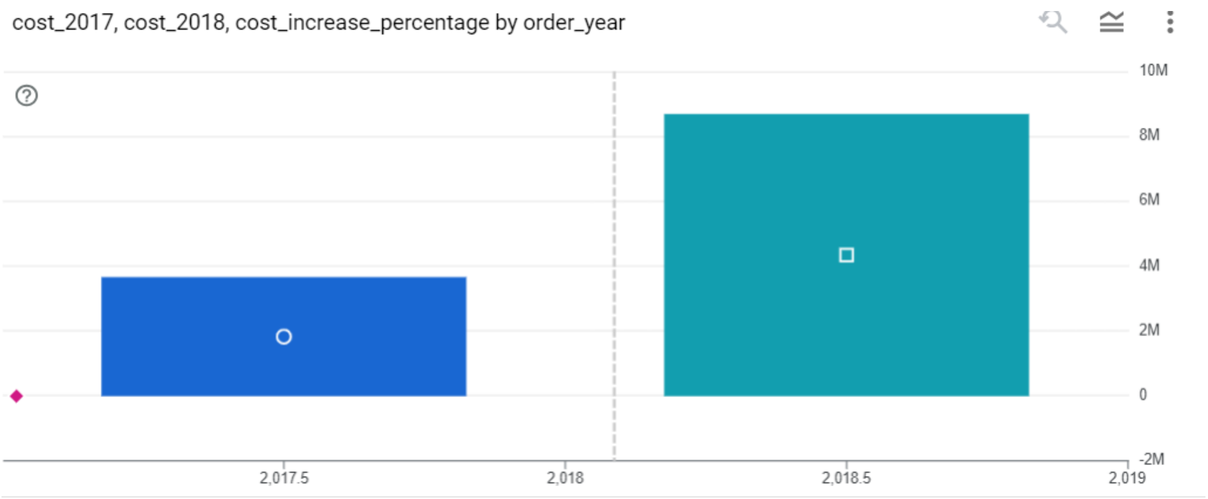
```
    EXTRACT(YEAR FROM o.order_purchase_timestamp) AS order_year,
    EXTRACT(MONTH FROM o.order_purchase_timestamp) AS order_month,
    p.payment_value AS order_cost
  FROM
    `target_sql.orders` AS o
  JOIN
    `target_sql.payments` AS p
  ON
    o.order_id = p.order_id
  WHERE
    (EXTRACT(YEAR FROM o.order_purchase_timestamp) = 2017 OR EXTRACT(YEAR FROM
o.order_purchase_timestamp) = 2018)
    AND
    EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8
)
SELECT
  order_year,
  ROUND(SUM(CASE WHEN order_year = 2017 THEN order_cost ELSE 0 END), 2) AS
cost_2017,
  ROUND(SUM(CASE WHEN order_year = 2018 THEN order_cost ELSE 0 END), 2) AS
cost_2018,
  CASE
    WHEN SUM(CASE WHEN order_year = 2017 THEN order_cost ELSE 0 END) = 0 THEN NULL
    ELSE ROUND(((SUM(CASE WHEN order_year = 2018 THEN order_cost ELSE 0 END) -
SUM(CASE WHEN order_year = 2017 THEN order_cost ELSE 0 END)) / SUM(CASE WHEN
order_year = 2017 THEN order_cost ELSE 0 END)) * 100, 2)
  END AS cost_increase_percentage
FROM
  order_costs
GROUP BY
  order_year
ORDER BY
  order_year;
```



Row	order_year	cost_2017	cost_2018	cost_increase_perce
1	2017	3669022.12	0.0	-100.0
2	2018	0.0	8694733.84	null

INSIGHTS:

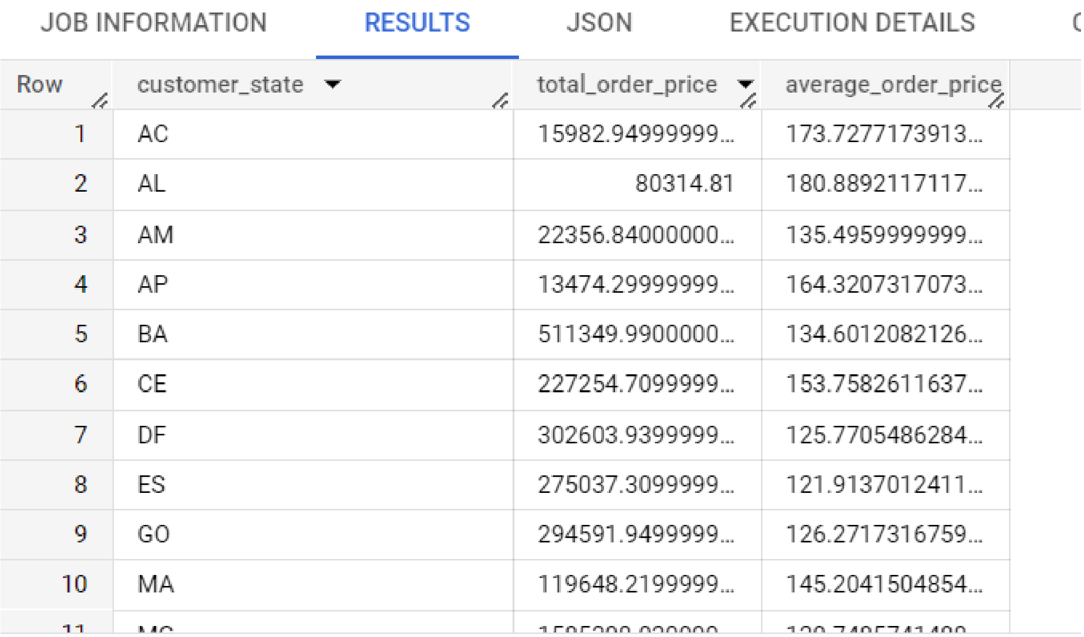
There is a increase of 42.2% in the cost of orders from the year 2017 to 2018.



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

ANS:

```
SELECT
  c.customer_state,
  SUM(oi.price) AS total_order_price,
  AVG(oi.price) AS average_order_price
FROM
  `target_sql.customers` AS c
JOIN
  `target_sql.orders` AS o
ON
  c.customer_id = o.customer_id
JOIN
  `target_sql.order_items` AS oi
ON
  o.order_id = oi.order_id
GROUP BY
  c.customer_state
ORDER BY
  c.customer_state;
```



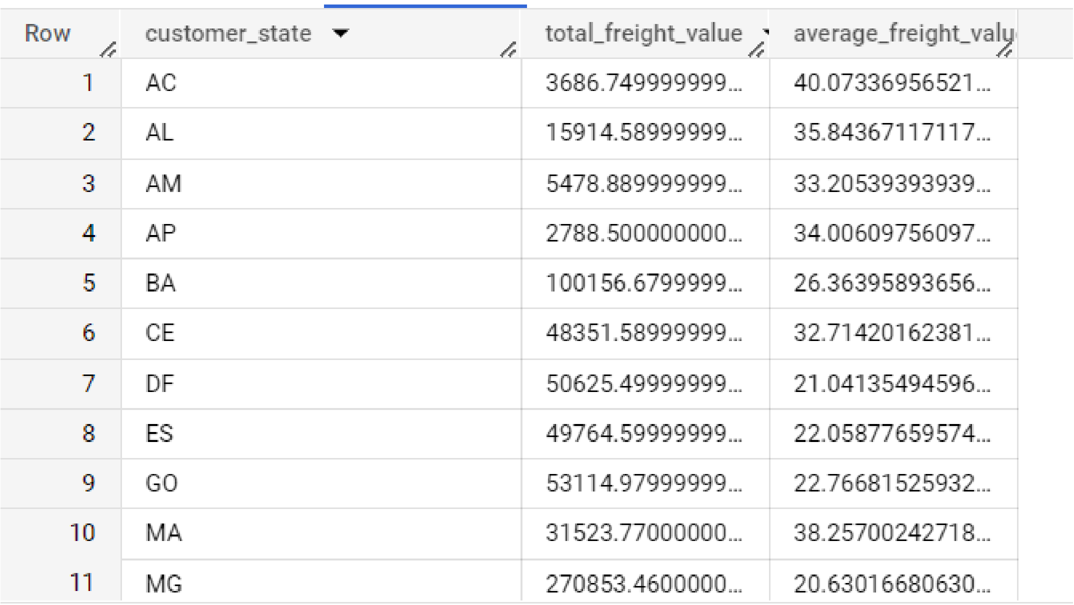
Row	customer_state	total_order_price	average_order_price
1	AC	15982.94999999...	173.7277173913...
2	AL	80314.81	180.8892117117...
3	AM	22356.84000000...	135.4959999999...
4	AP	13474.29999999...	164.3207317073...
5	BA	511349.9900000...	134.6012082126...
6	CE	227254.7099999...	153.7582611637...
7	DF	302603.9399999...	125.7705486284...
8	ES	275037.3099999...	121.9137012411...
9	GO	294591.9499999...	126.2717316759...
10	MA	119648.2199999...	145.2041504854...
11	MG	1585308.029999...	120.7485741488...

INSIGHTS:
Here we have seen the total order price as well as average order price of each state respectively.

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

ANS:

```
SELECT
  c.customer_state,
  SUM(oi.freight_value) AS total_freight_value,
  AVG(oi.freight_value) AS average_freight_value
FROM
  `target_sql.customers` AS c
JOIN
  `target_sql.orders` AS o
ON
  c.customer_id = o.customer_id
JOIN
  `target_sql.order_items` AS oi
ON
  o.order_id = oi.order_id
GROUP BY
  c.customer_state
ORDER BY
  c.customer_state;
```



Row	customer_state	total_freight_value	average_freight_valu
1	AC	3686.749999999...	40.07336956521...
2	AL	15914.58999999...	35.84367117117...
3	AM	5478.889999999...	33.20539393939...
4	AP	2788.500000000...	34.00609756097...
5	BA	100156.6799999...	26.36395893656...
6	CE	48351.58999999...	32.71420162381...
7	DF	50625.49999999...	21.04135494596...
8	ES	49764.59999999...	22.05877659574...
9	GO	53114.97999999...	22.76681525932...
10	MA	31523.77000000...	38.25700242718...
11	MG	270853.4600000...	20.63016680630...

INSIGHTS:

Here we have seen the total freight value and average freight value for each state respectively.

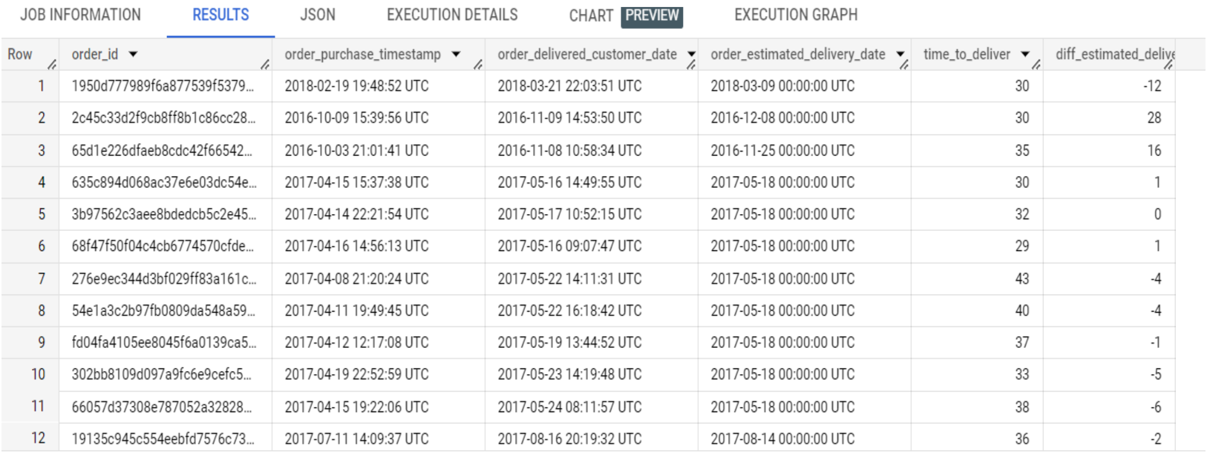
5. Analysis based on sales, freight and delivery time.

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**

ANS:

```
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_estimated_delivery_date, order_delivered_customer_date, DAY)
AS diff_estimated_delivery
FROM
  `target_sql.orders`;
```



Row	order_id	order_purchase_timestamp	order_delivered_customer_date	order_estimated_delivery_date	time_to_deliver	diff_estimated_delive
1	1950d777989f6a877539f5379...	2018-02-19 19:48:52 UTC	2018-03-21 22:03:51 UTC	2018-03-09 00:00:00 UTC	30	-12
2	2c45c33d2f9cb8ff8b1c86cc28...	2016-10-09 15:39:56 UTC	2016-11-09 14:53:50 UTC	2016-12-08 00:00:00 UTC	30	28
3	65d1e226dfaeb8cdc42f66542...	2016-10-03 21:01:41 UTC	2016-11-08 10:58:34 UTC	2016-11-25 00:00:00 UTC	35	16
4	635c894d068ac37e6e03dc54e...	2017-04-15 15:37:38 UTC	2017-05-16 14:49:55 UTC	2017-05-18 00:00:00 UTC	30	1
5	3b97562c3aee8bdedcb5c2e45...	2017-04-14 22:21:54 UTC	2017-05-17 10:52:15 UTC	2017-05-18 00:00:00 UTC	32	0
6	68f47f50f04c4cb6774570cfde...	2017-04-16 14:56:13 UTC	2017-05-16 09:07:47 UTC	2017-05-18 00:00:00 UTC	29	1
7	276e9ec344d3bf029ff83a161c...	2017-04-08 21:20:24 UTC	2017-05-22 14:11:31 UTC	2017-05-18 00:00:00 UTC	43	-4
8	54e1a3c2b97fb0809da548a59...	2017-04-11 19:49:45 UTC	2017-05-22 16:18:42 UTC	2017-05-18 00:00:00 UTC	40	-4
9	fd04fa4105ee8045f6a0139ca5...	2017-04-12 12:17:08 UTC	2017-05-19 13:44:52 UTC	2017-05-18 00:00:00 UTC	37	-1
10	302bb8109d097a9fc6e9cefc5...	2017-04-19 22:52:59 UTC	2017-05-23 14:19:48 UTC	2017-05-18 00:00:00 UTC	33	-5
11	66057d37308e787052a32828...	2017-04-15 19:22:06 UTC	2017-05-24 08:11:57 UTC	2017-05-18 00:00:00 UTC	38	-6
12	19135c945c554eebfd7576c73...	2017-07-11 14:09:37 UTC	2017-08-16 20:19:32 UTC	2017-08-14 00:00:00 UTC	36	-2

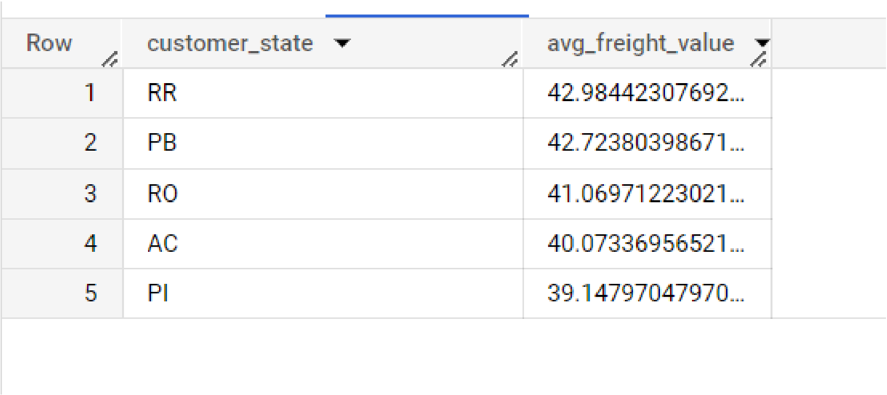
INSIGHTS:
For some of the orders there are negative values for diff_estimated_date which means there was a delay in delivery. And this has happened for many orders here. Even for some of the orders there was a delay of more than a month than the estimated delivery date.

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

ANS:

```
SELECT
  c.customer_state,
  AVG(oi.freight_value) AS avg_freight_value
FROM
  `target_sql.customers` AS c
JOIN
  `target_sql.orders` AS o
ON
  c.customer_id = o.customer_id
JOIN
  `target_sql.order_items` AS oi
ON
  o.order_id = oi.order_id
GROUP BY
  c.customer_state
ORDER BY
  avg_freight_value DESC
LIMIT 5;
```

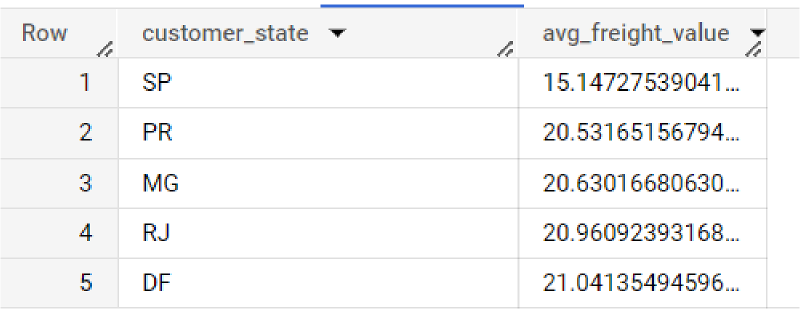
Top 5 states with highest average freight values:



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

```
SELECT
  c.customer_state,
  AVG(oi.freight_value) AS avg_freight_value
FROM
  `target_sql.customers` AS c
JOIN
  `target_sql.orders` AS o
ON
  c.customer_id = o.customer_id
JOIN
  `target_sql.order_items` AS oi
ON
  o.order_id = oi.order_id
GROUP BY
  c.customer_state
ORDER BY
  avg_freight_value ASC
LIMIT 5;
```

Top 5 states with lowest average freight values:



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

INSIGHTS:

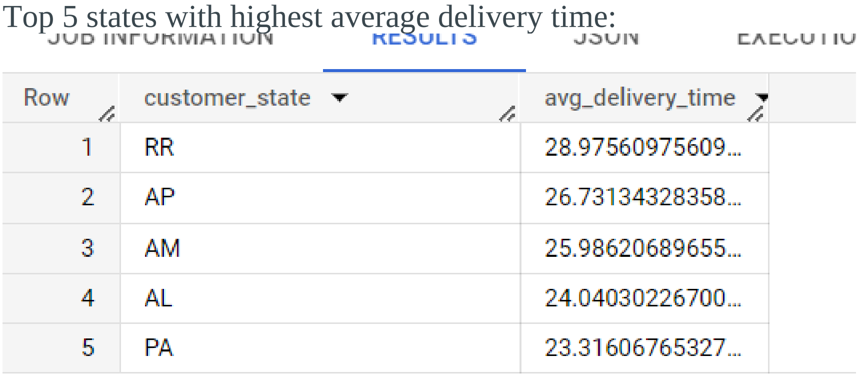
The top 5 states with highest average freight values are RR, PB, RO, AC, PI which are having the average around 40 and the top 5 states with lowest average freight values are SP, PR, MG, RJ, DF which are having the average around 19.

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

ANS:

```
SELECT
  c.customer_state,
  AVG(DATE_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp, DAY))
AS avg_delivery_time
FROM
  `target_sql.customers` AS c
JOIN
  `target_sql.orders` AS o
ON
  c.customer_id = o.customer_id
GROUP BY
  c.customer_state
ORDER BY
  avg_delivery_time DESC
LIMIT 5;
```

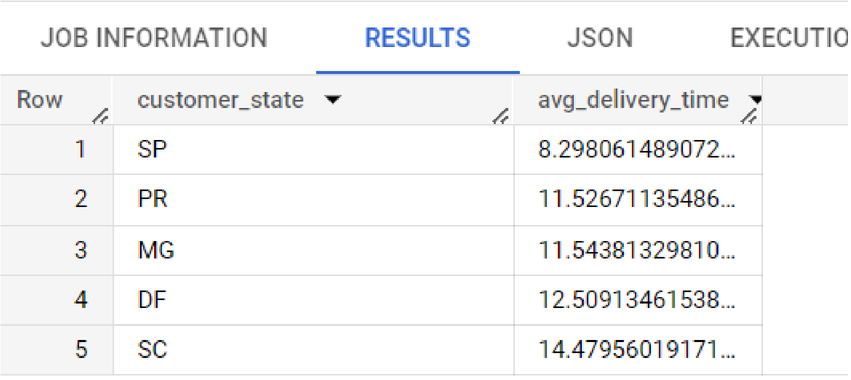
Top 5 states with highest average delivery time:



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

```
SELECT
  c.customer_state,
  AVG(DATE_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp, DAY))
AS avg_delivery_time
FROM
  `target_sql.customers` AS c
JOIN
  `target_sql.orders` AS o
ON
  c.customer_id = o.customer_id
GROUP BY
  c.customer_state
ORDER BY
  avg_delivery_time ASC
LIMIT 5;
```

Top 5 states with lowest average delivery time:



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

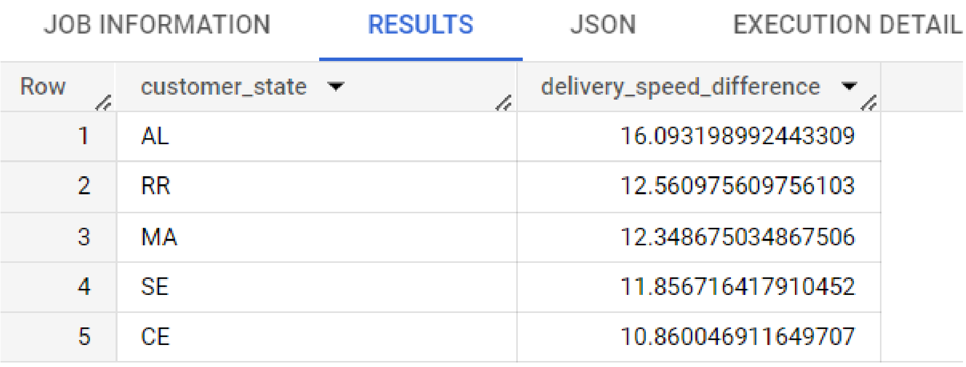
INSIGHTS:

The top 5 states with highest average delivery time are RR, AP, AM, AL, PA which are having the average around 25 and the top 5 states with lowest average delivery time are SP, PR, MG, DF, SC which are having the average around 11.

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.

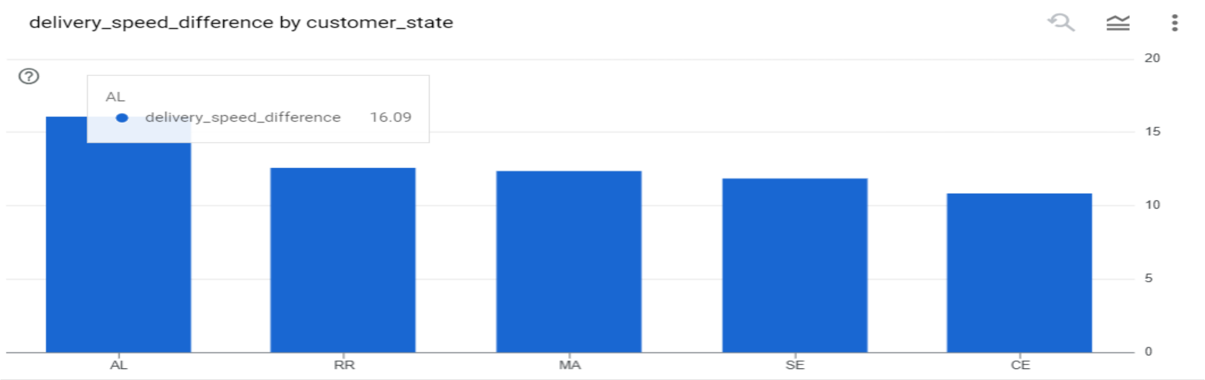
ANS:

```
WITH delivery_time_diff AS (
  SELECT
    c.customer_state,
    AVG(DATE_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp,
DAY)) AS avg_actual_delivery_time,
    AVG(DATE_DIFF(o.order_estimated_delivery_date, o.order_delivered_customer_date,
DAY)) AS avg_estimated_delivery_diff
  FROM
    `target_sql.customers` AS c
  JOIN
    `target_sql.orders` AS o
  ON
    c.customer_id = o.customer_id
  GROUP BY
    c.customer_state
)
SELECT
  customer_state,
  (avg_actual_delivery_time - avg_estimated_delivery_diff) AS
delivery_speed_difference
FROM
  delivery_time_diff
ORDER BY
  delivery_speed_difference DESC
LIMIT 5;
```



Row	customer_state	delivery_speed_difference
1	AL	16.093198992443309
2	RR	12.560975609756103
3	MA	12.348675034867506
4	SE	11.856716417910452
5	CE	10.860046911649707

INSIGHTS:



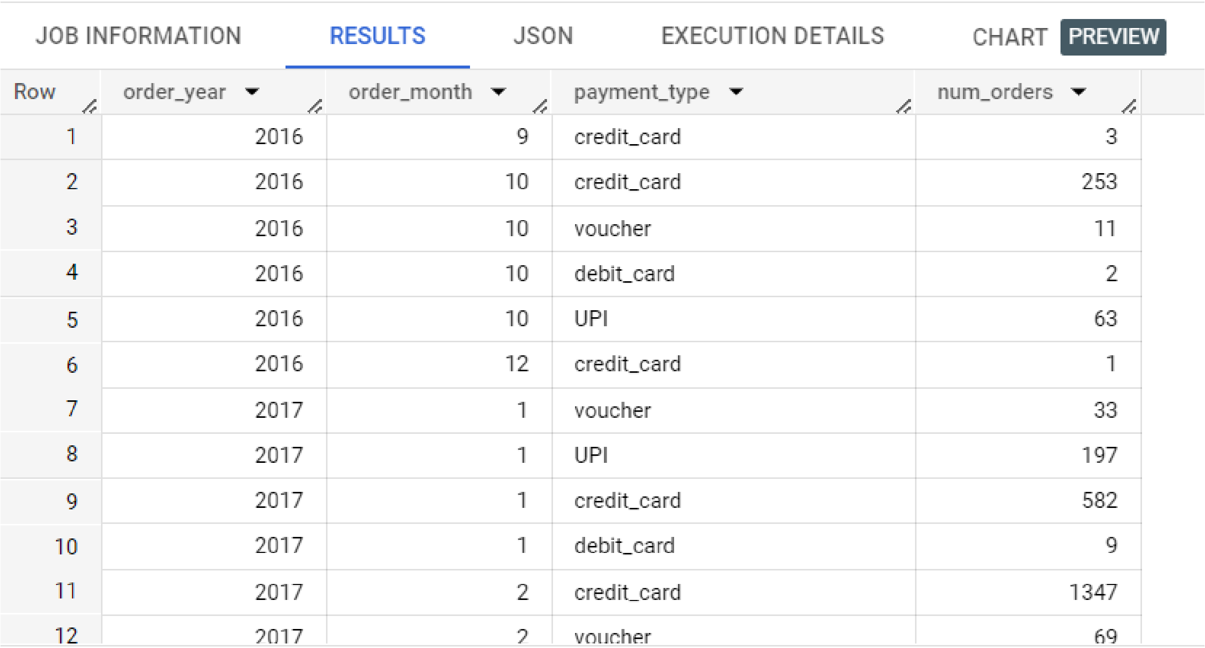
The top 5 states where the order delivery date is really fast as compared to the estimated date of delivery are AL, RR, MA, SE and CE. Among these AL is having the fastest delivery speed.

6. Analysis based on the payments:

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

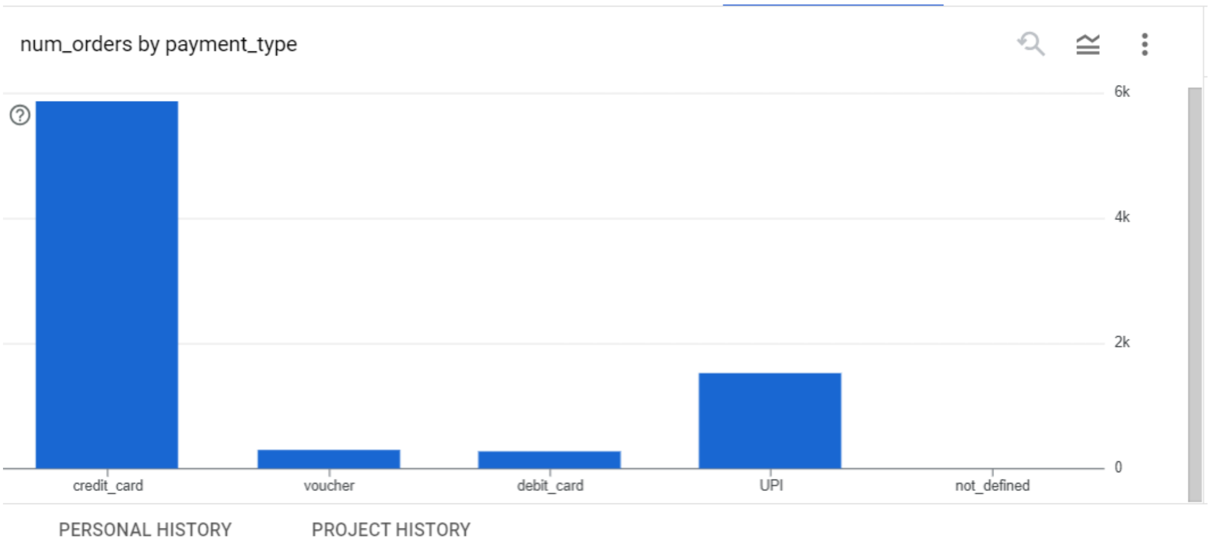
ANS:

```
SELECT
  EXTRACT(YEAR FROM o.order_purchase_timestamp) AS order_year,
  EXTRACT(MONTH FROM o.order_purchase_timestamp) AS order_month,
  p.payment_type,
  COUNT(DISTINCT o.order_id) AS num_orders
FROM
  `target_sql.orders` AS o
JOIN
  `target_sql.payments` AS p
ON
  o.order_id = p.order_id
GROUP BY
  order_year,
  order_month,
  payment_type
ORDER BY
  order_year,
  order_month;
```



Row	order_year	order_month	payment_type	num_orders
1	2016	9	credit_card	3
2	2016	10	credit_card	253
3	2016	10	voucher	11
4	2016	10	debit_card	2
5	2016	10	UPI	63
6	2016	12	credit_card	1
7	2017	1	voucher	33
8	2017	1	UPI	197
9	2017	1	credit_card	582
10	2017	1	debit_card	9
11	2017	2	credit_card	1347
12	2017	2	voucher	69

INSIGHTS:

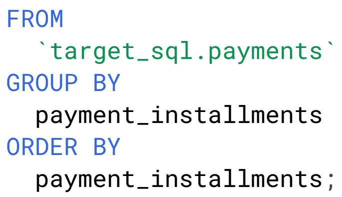


Mostly credit_card is used for making payments and also number of order is very higher when credit card is used.

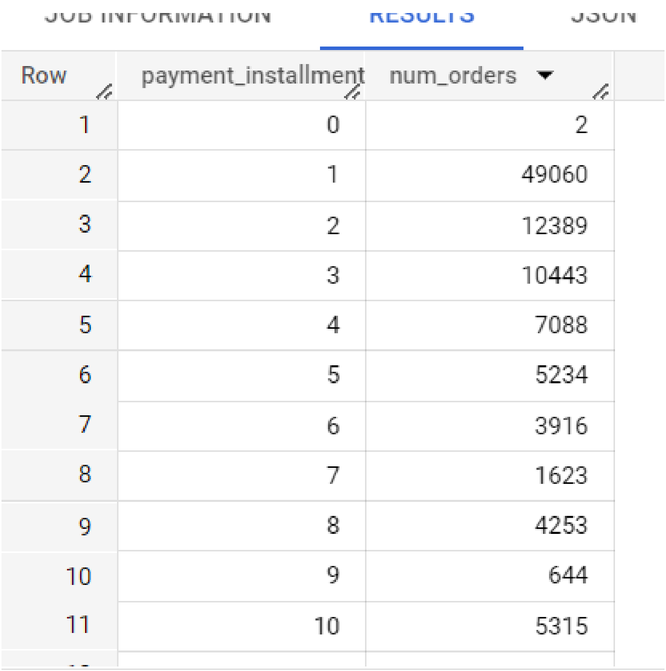
2. **Find the no. of orders placed on the basis of the payment installments that have been paid.**

ANS:

```
SELECT
  payment_installments,
  COUNT(DISTINCT order_id) AS num_orders
```



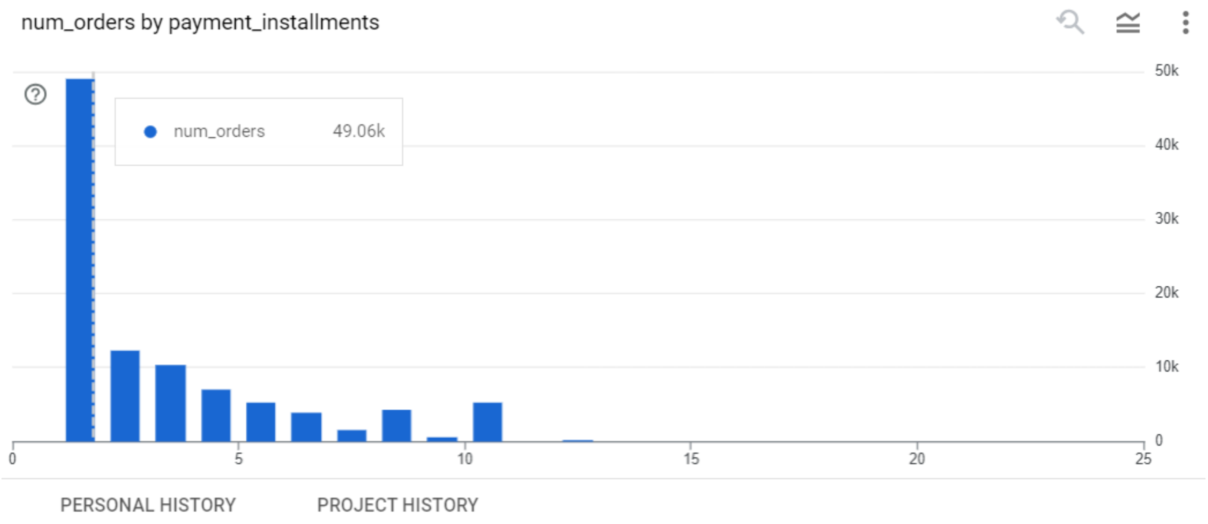
```
FROM
  `target_sql.payments`
GROUP BY
  payment_installments
ORDER BY
  payment_installments;
```



JOB INFORMATION
RESULTS
JSON

Row	payment_installment	num_orders
1	0	2
2	1	49060
3	2	12389
4	3	10443
5	4	7088
6	5	5234
7	6	3916
8	7	1623
9	8	4253
10	9	644
11	10	5315

INSIGHTS:



The payment installment with highest number of orders is 1 which is having 49060 orders then follows 2 and 3 as second highest and third highest respectively, and rest all are having below 10000 orders.