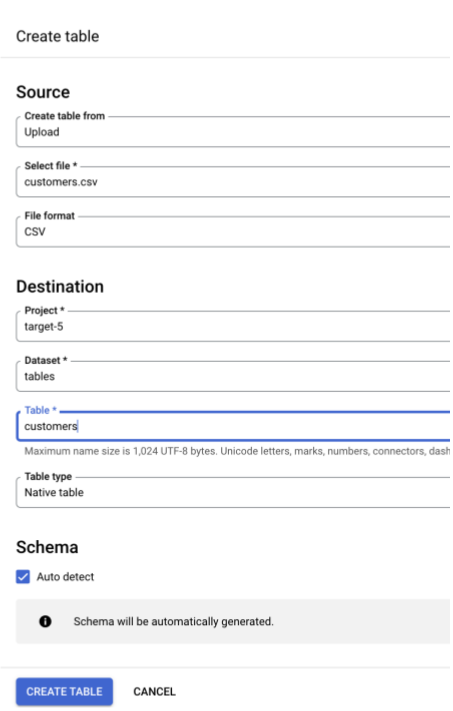
SQL Business Case Study on Target sales data

Table of Contents

Importing the data into big query.



Create table

Source

Create table from: Upload

Select file *: customers.csv

File format: CSV

Destination

Project *: target-5

Dataset *: tables

Table *: customers

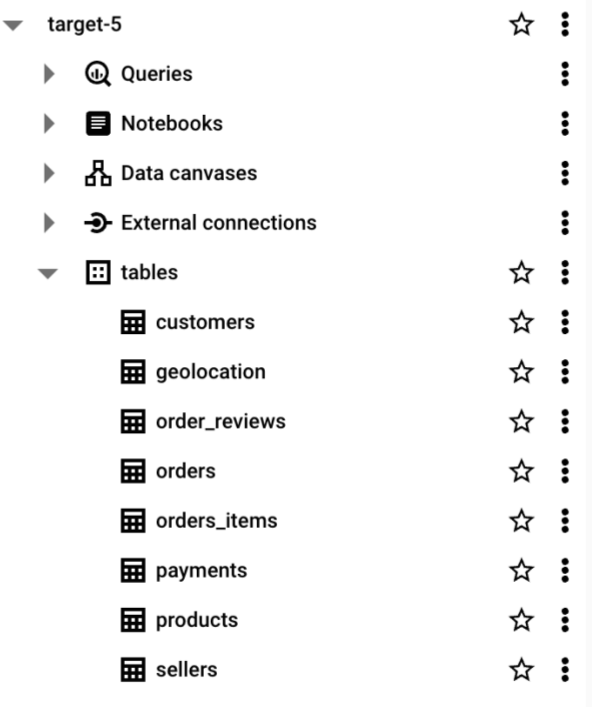
Maximum name size is 1,024 UTF-8 bytes. Unicode letters, marks, numbers, connectors, dash

Table type: Native table

Schema

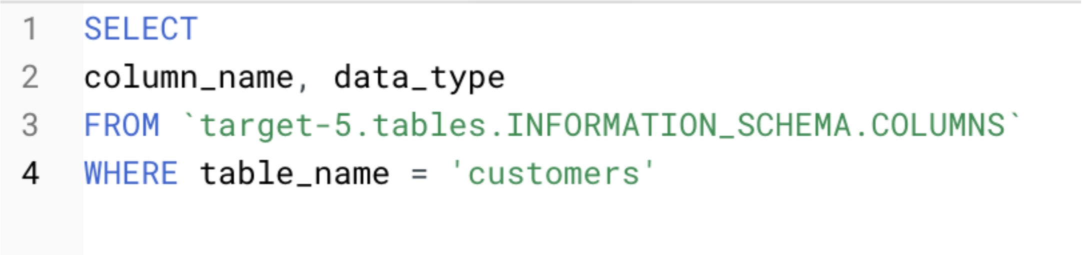
☑ Auto detect

Schema will be automatically generated.



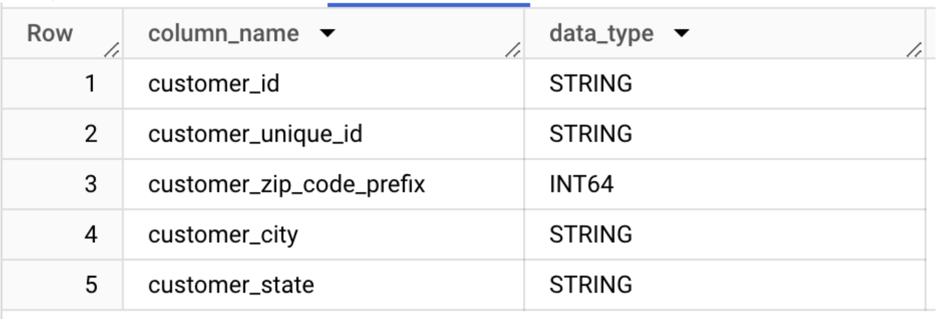
Exploring the data set.

Data type of all columns in the "customers" table.
Query:



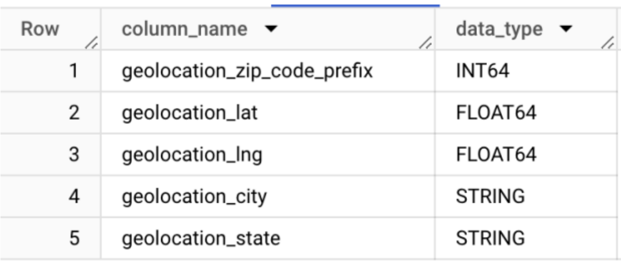
```
SELECT
column_name, data_type
FROM `target-5.tables.INFORMATION_SCHEMA.COLUMNS`
WHERE table_name = 'customers'
```

Output:

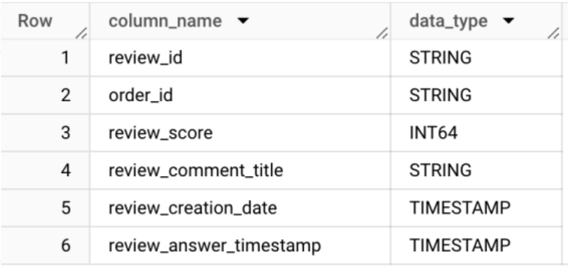


Row	column_name	data_type
1	customer_id	STRING
2	customer_unique_id	STRING
3	customer_zip_code_prefix	INT64
4	customer_city	STRING
5	customer_state	STRING

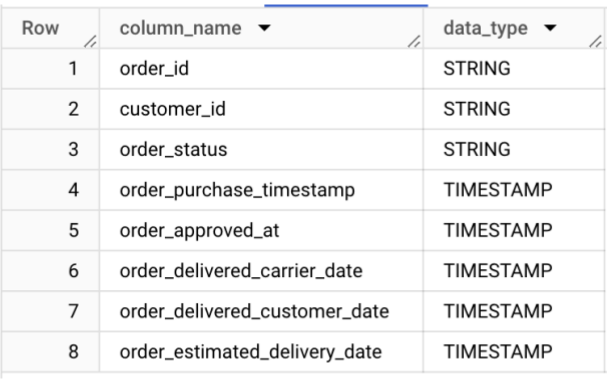
Likewise we can get the datatype for all other tables.



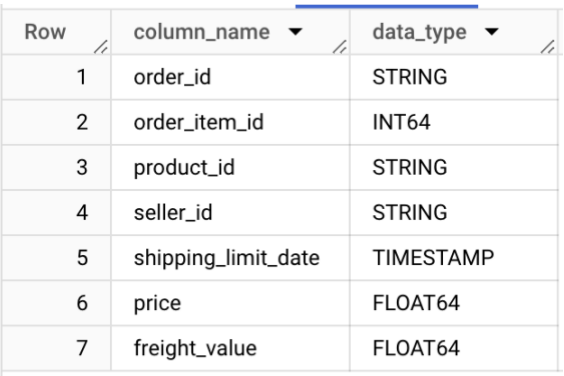
Row	column_name	data_type
1	geolocation_zip_code_prefix	INT64
2	geolocation_lat	FLOAT64
3	geolocation_lng	FLOAT64
4	geolocation_city	STRING
5	geolocation_state	STRING



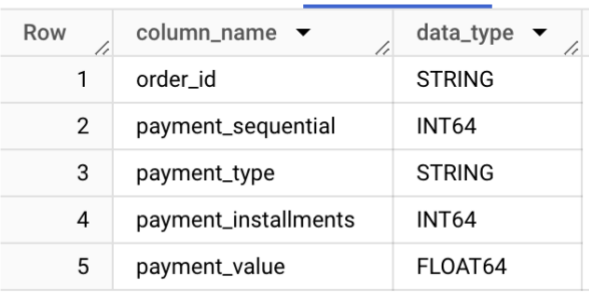
Row	column_name	data_type
1	review_id	STRING
2	order_id	STRING
3	review_score	INT64
4	review_comment_title	STRING
5	review_creation_date	TIMESTAMP
6	review_answer_timestamp	TIMESTAMP



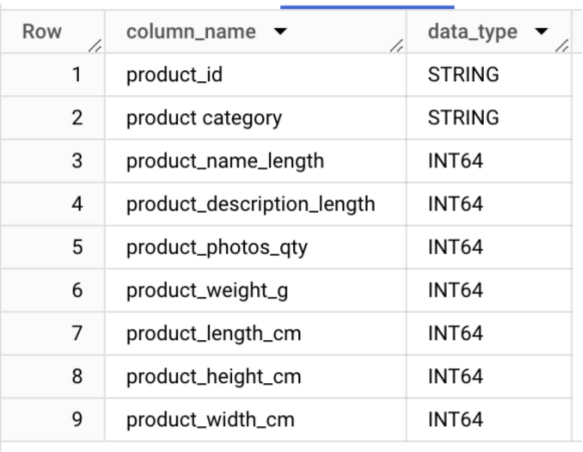
Row	column_name	data_type
1	order_id	STRING
2	customer_id	STRING
3	order_status	STRING
4	order_purchase_timestamp	TIMESTAMP
5	order_approved_at	TIMESTAMP
6	order_delivered_carrier_date	TIMESTAMP
7	order_delivered_customer_date	TIMESTAMP
8	order_estimated_delivery_date	TIMESTAMP



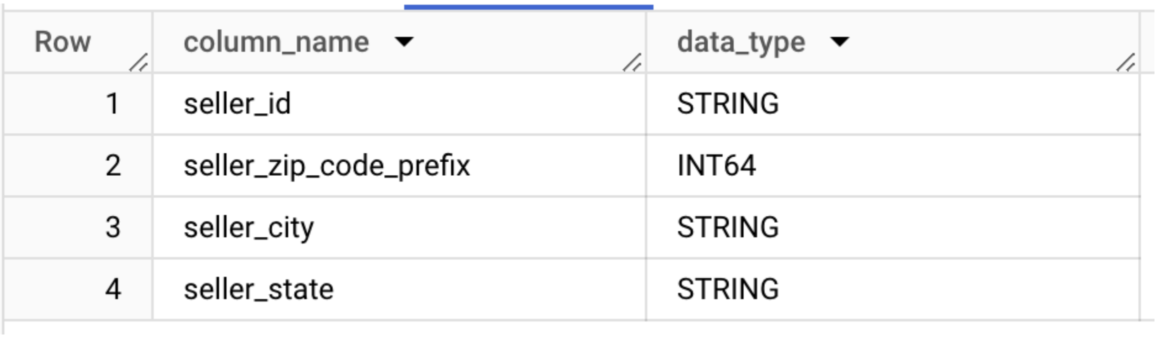
Row	column_name	data_type
1	order_id	STRING
2	order_item_id	INT64
3	product_id	STRING
4	seller_id	STRING
5	shipping_limit_date	TIMESTAMP
6	price	FLOAT64
7	freight_value	FLOAT64



Row	column_name	data_type
1	order_id	STRING
2	payment_sequential	INT64
3	payment_type	STRING
4	payment_installments	INT64
5	payment_value	FLOAT64



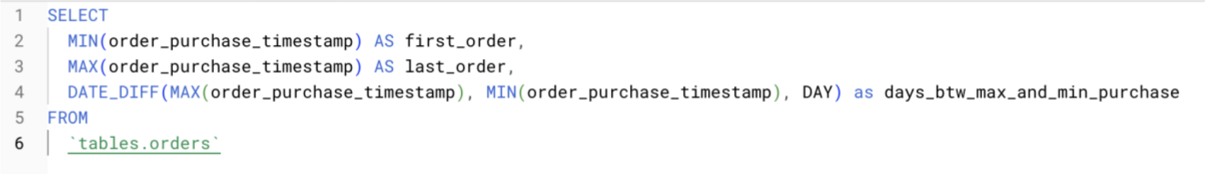
Row	column_name	data_type
1	product_id	STRING
2	product category	STRING
3	product_name_length	INT64
4	product_description_length	INT64
5	product_photos_qty	INT64
6	product_weight_g	INT64
7	product_length_cm	INT64
8	product_height_cm	INT64
9	product_width_cm	INT64



Row	column_name	data_type
1	seller_id	STRING
2	seller_zip_code_prefix	INT64
3	seller_city	STRING
4	seller_state	STRING

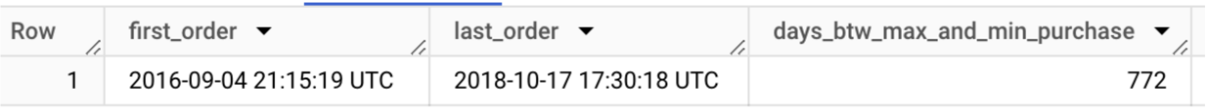
Get the time range between which the orders were placed.

Query:



```
SELECT
  MIN(order_purchase_timestamp) AS first_order,
  MAX(order_purchase_timestamp) AS last_order,
  DATE_DIFF(MAX(order_purchase_timestamp), MIN(order_purchase_timestamp), DAY) as days_btw_max_and_min_purchase
FROM
  `tables.orders`
```

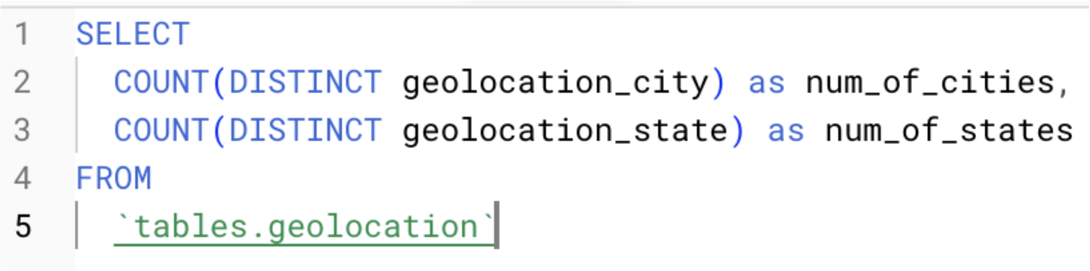
Output:



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

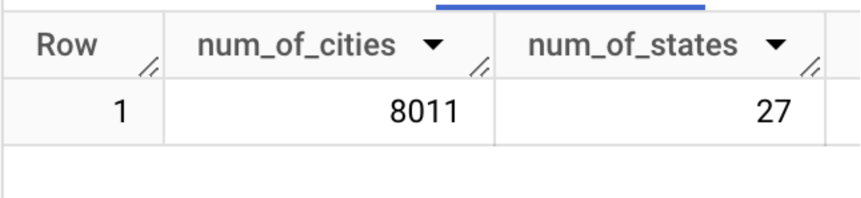
Count the Cities & States of customers who ordered during the given period.

Query:



```
SELECT
  COUNT(DISTINCT geolocation_city) as num_of_cities,
  COUNT(DISTINCT geolocation_state) as num_of_states
FROM
  `tables.geolocation`
```

Output:

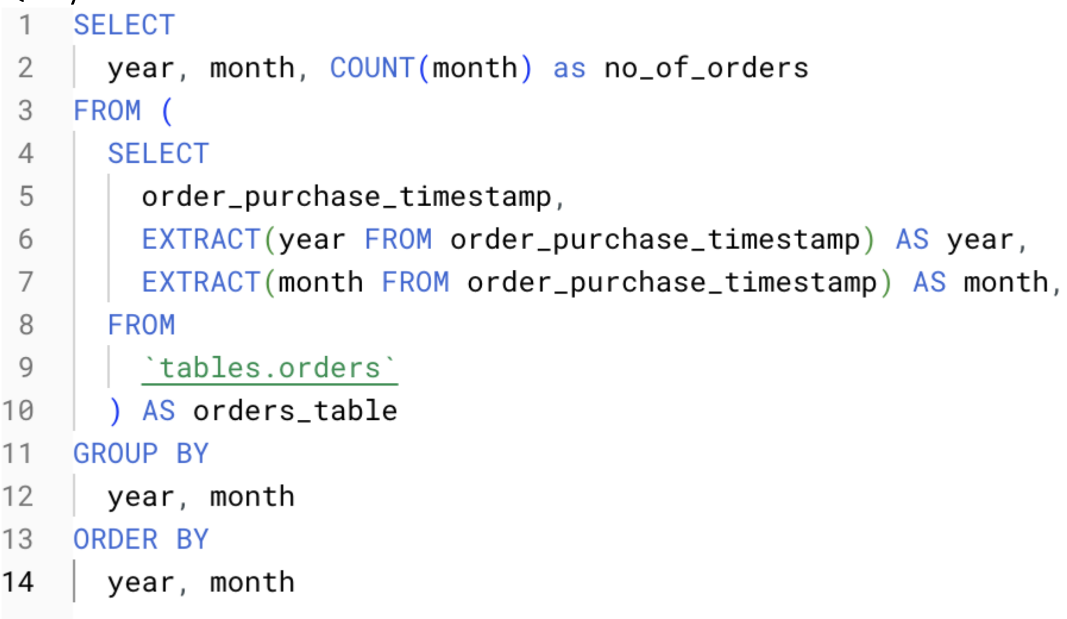


Row	num_of_cities	num_of_states
1	8011	27

In-depth Exploration:

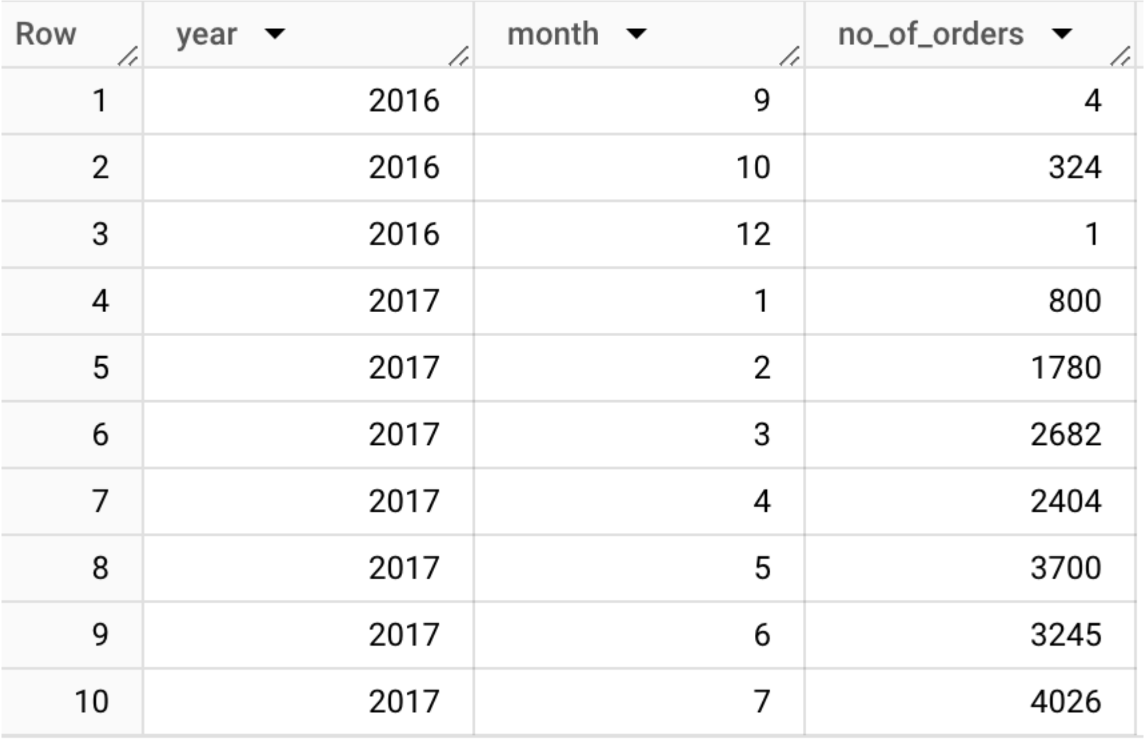
Is there a growing trend in the no. of orders placed over the past years?

Query:



```sql
SELECT
  year, month, COUNT(month) as no_of_orders
FROM (
  SELECT
    order_purchase_timestamp,
    EXTRACT(year FROM order_purchase_timestamp) AS year,
    EXTRACT(month FROM order_purchase_timestamp) AS month,
  FROM
    `tables.orders`
  ) AS orders_table
GROUP BY
  year, month
ORDER BY
  year, month
```

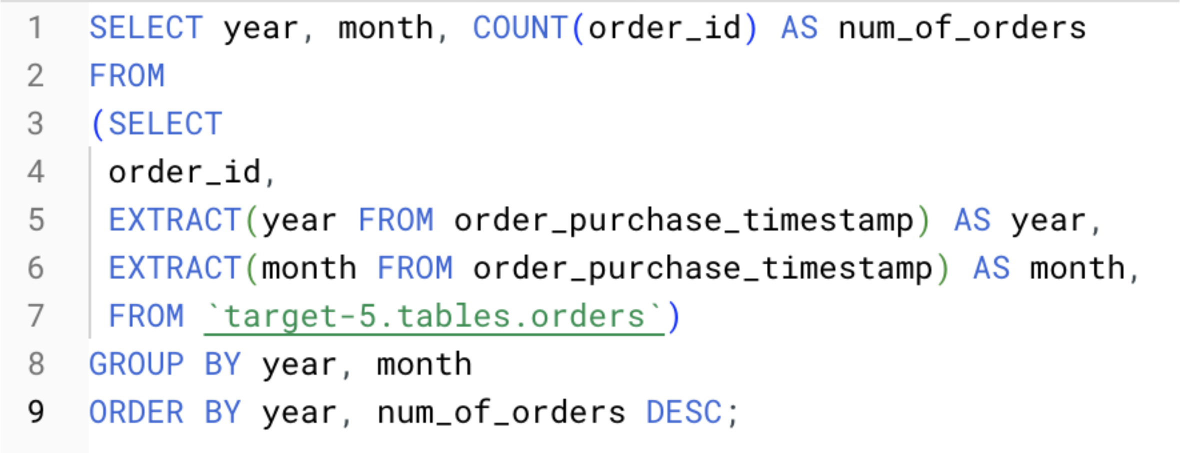
Output:



Row	year	month	no_of_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

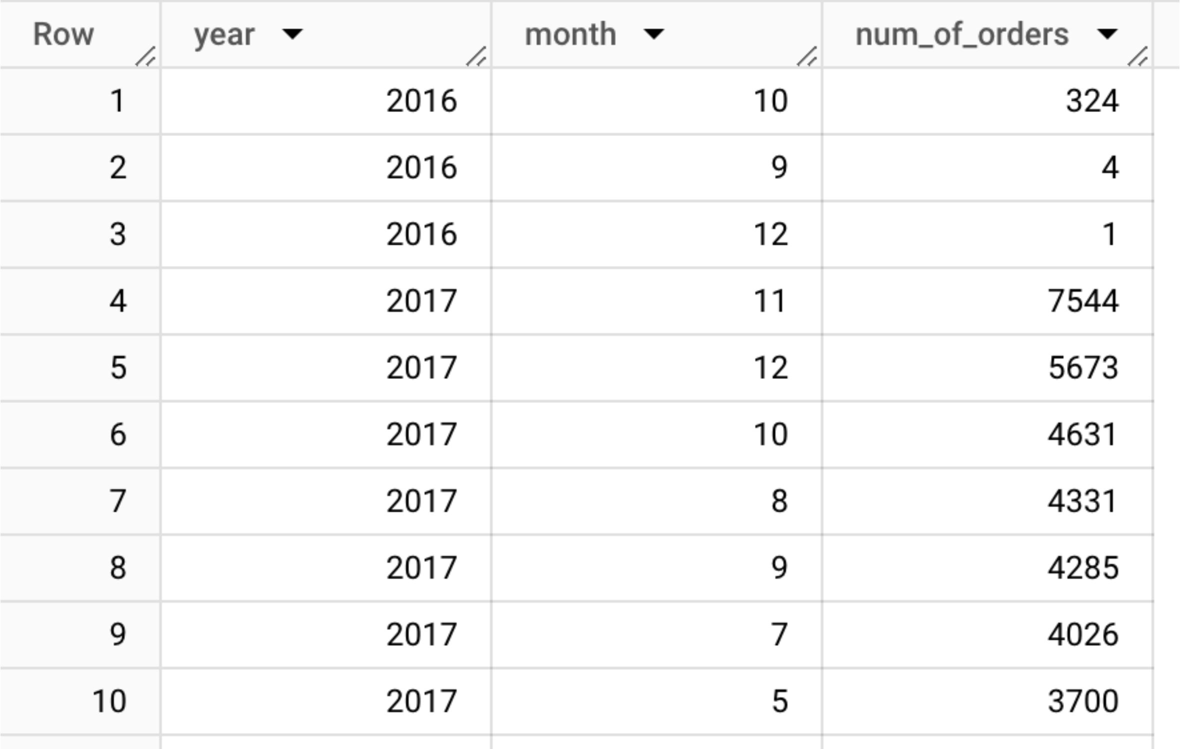
Can we see some kind of monthly seasonality in terms of the no. of orders being placed?

Query:



```
SELECT year, month, COUNT(order_id) AS num_of_orders
FROM
(SELECT
 order_id,
 EXTRACT(year FROM order_purchase_timestamp) AS year,
 EXTRACT(month FROM order_purchase_timestamp) AS month,
 FROM `target-5.tables.orders`)
GROUP BY year, month
ORDER BY year, num_of_orders DESC;
```

Result:



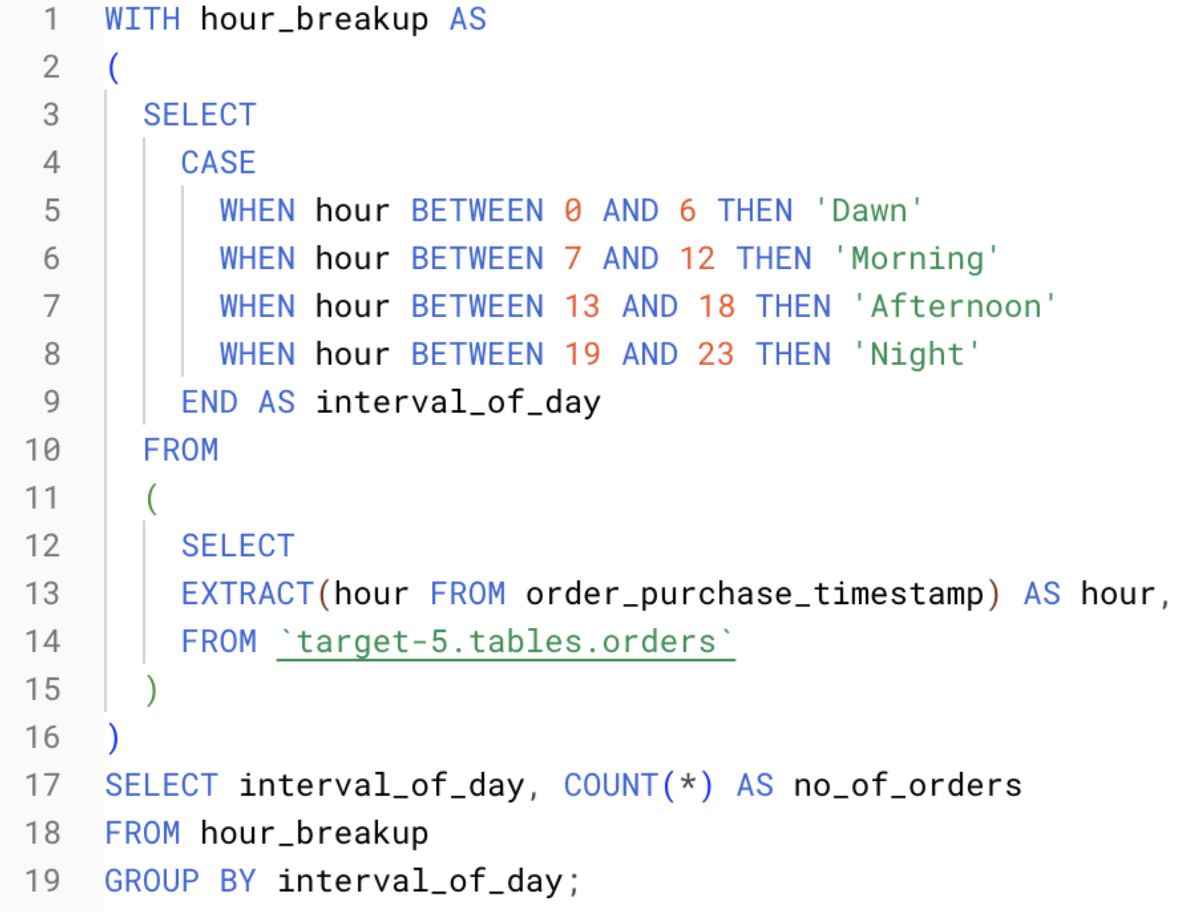
Row	year	month	num_of_orders
1	2016	10	324
2	2016	9	4
3	2016	12	1
4	2017	11	7544
5	2017	12	5673
6	2017	10	4631
7	2017	8	4331
8	2017	9	4285
9	2017	7	4026
10	2017	5	3700

Insights: With just 3 months data available in 2016, the number of orders peak in October. In 2017, the top 3 months are November, December and October. In 2018, January, March and April are the top 3 months. With the limited data available I don't see any monthly seasonality in terms of number of orders being placed.
Recommendations: NA
Assumptions: NA

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

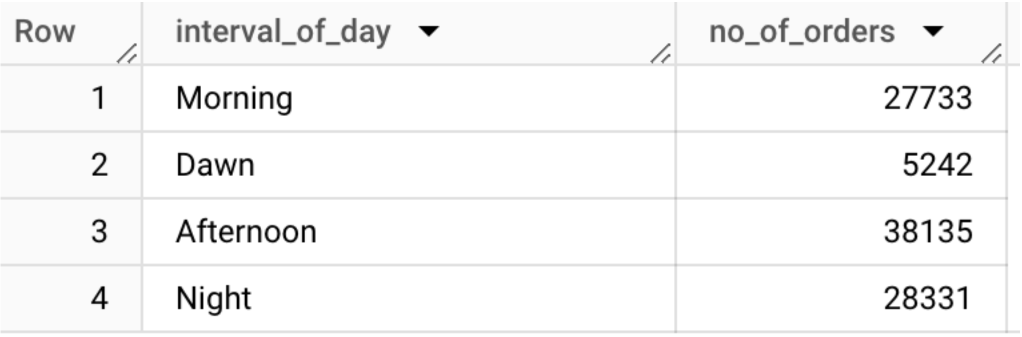
- 0-6 hrs : Dawn
- 7-12 hrs : Mornings
- 13-18 hrs : Afternoon
- 19-23 hrs : Night

Query:



```
WITH hour_breakup AS
(
  SELECT
    CASE
      WHEN hour BETWEEN 0 AND 6 THEN 'Dawn'
      WHEN hour BETWEEN 7 AND 12 THEN 'Morning'
      WHEN hour BETWEEN 13 AND 18 THEN 'Afternoon'
      WHEN hour BETWEEN 19 AND 23 THEN 'Night'
    END AS interval_of_day
  FROM
  (
    SELECT
    EXTRACT(hour FROM order_purchase_timestamp) AS hour,
    FROM `target-5.tables.orders`
  )
)
SELECT interval_of_day, COUNT(*) AS no_of_orders
FROM hour_breakup
GROUP BY interval_of_day;
```

Output:



Row	interval_of_day	no_of_orders
1	Morning	27733
2	Dawn	5242
3	Afternoon	38135
4	Night	28331

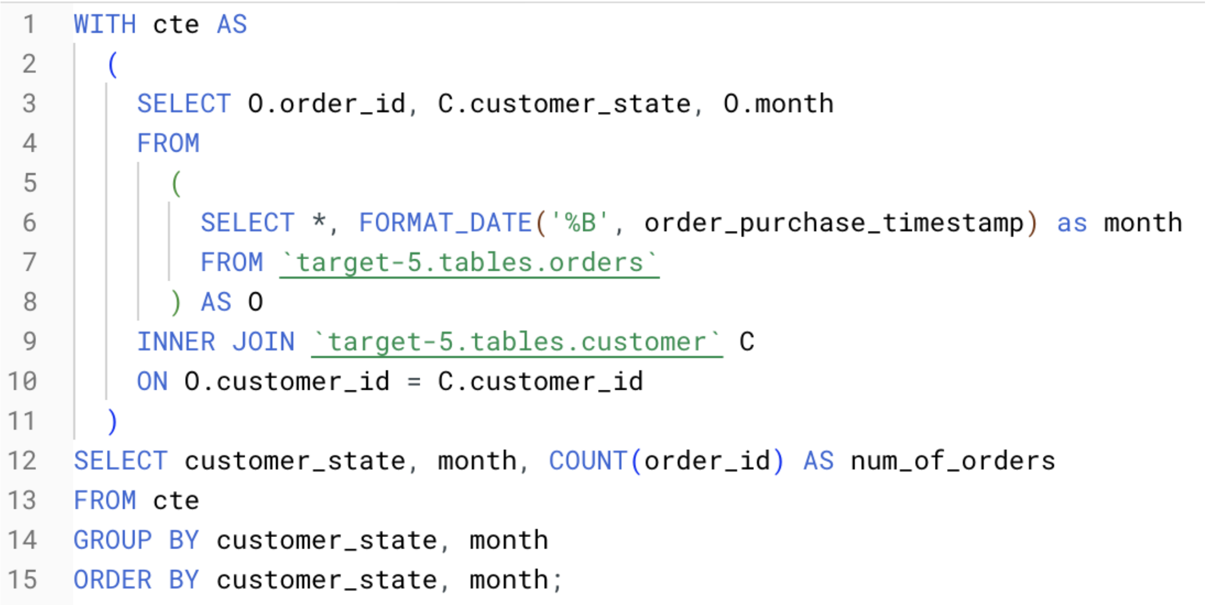
Insights: We can see that that the customers order the most during afternoon and least during dawn with mornings and night, being almost same, in the middle.

Recommendations: NA
Assumptions: NA

Evolution of E-commerce orders in the Brazil region

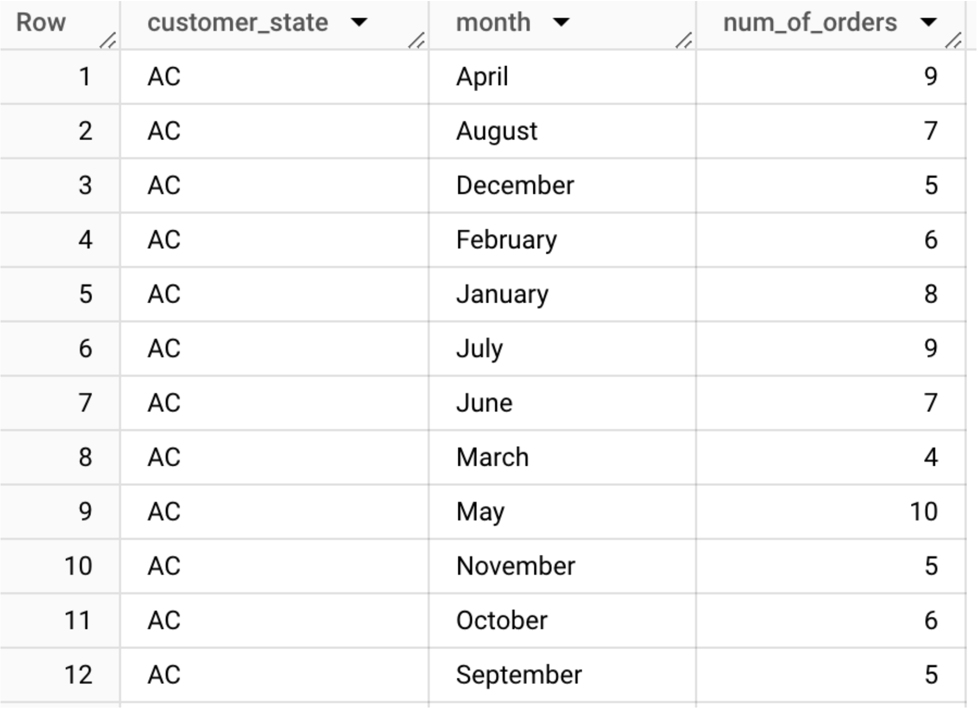
Get the month on month number of orders placed in each state.

Query:



```
WITH cte AS
  (
    SELECT O.order_id, C.customer_state, O.month
    FROM
      (
        SELECT *, FORMAT_DATE('%B', order_purchase_timestamp) as month
        FROM `target-5.tables.orders`
      ) AS O
    INNER JOIN `target-5.tables.customer` C
    ON O.customer_id = C.customer_id
  )
SELECT customer_state, month, COUNT(order_id) AS num_of_orders
FROM cte
GROUP BY customer_state, month
ORDER BY customer_state, month;
```

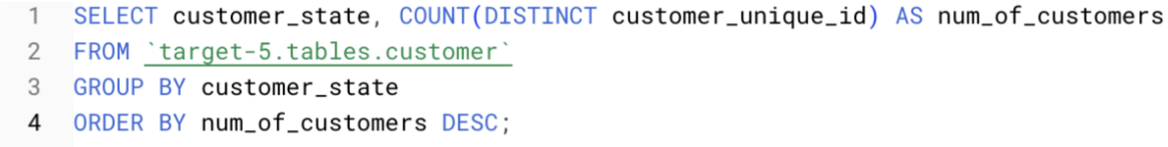
Output:



Row	customer_state	month	num_of_orders
1	AC	April	9
2	AC	August	7
3	AC	December	5
4	AC	February	6
5	AC	January	8
6	AC	July	9
7	AC	June	7
8	AC	March	4
9	AC	May	10
10	AC	November	5
11	AC	October	6
12	AC	September	5

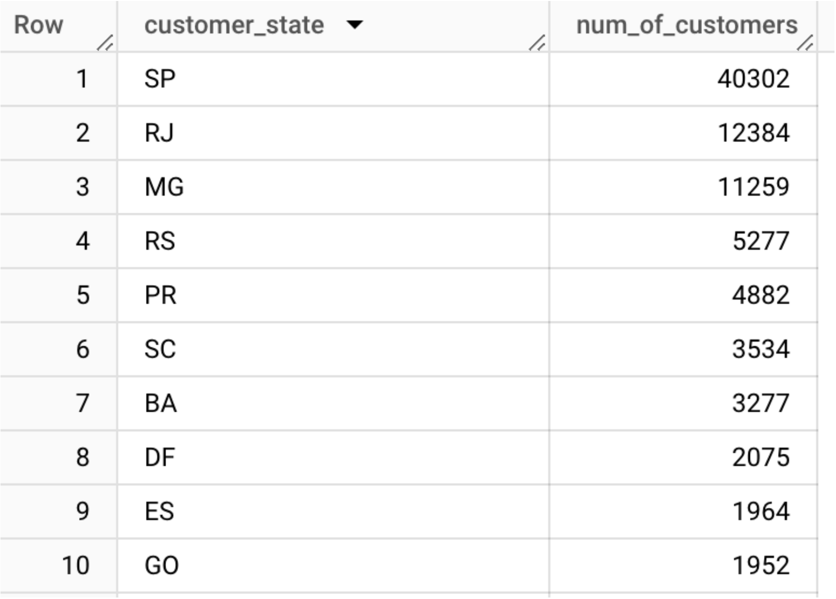
How are the customers distributed across all the states?

Query:



```sql
SELECT customer_state, COUNT(DISTINCT customer_unique_id) AS num_of_customers
FROM `target-5.tables.customer`
GROUP BY customer_state
ORDER BY num_of_customers DESC;
```

Output:



Row	customer_state	num_of_customers
1	SP	40302
2	RJ	12384
3	MG	11259
4	RS	5277
5	PR	4882
6	SC	3534
7	BA	3277
8	DF	2075
9	ES	1964
10	GO	1952

Insights: São Paulo has the largest number of customers followed not so closely by Rio de Janeiro and Minas Gerais. Roraima has the least number of customers.

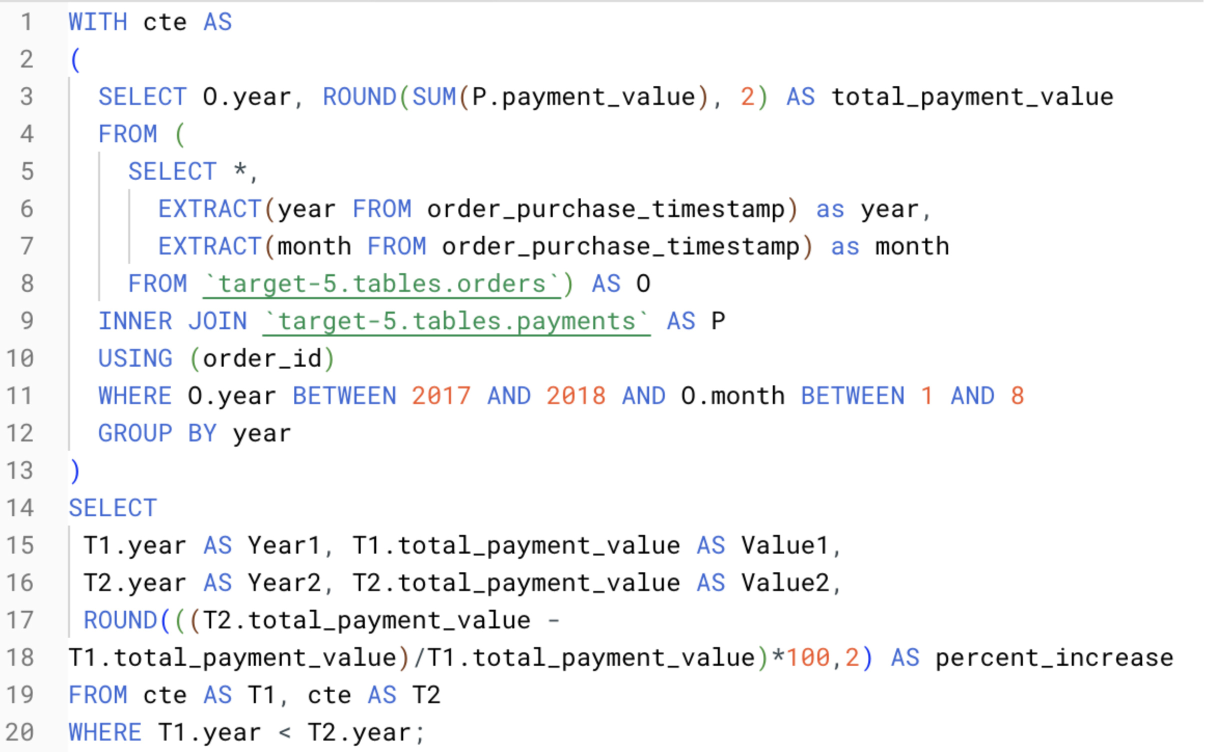
Recommendations: NA

Assumptions: NA

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

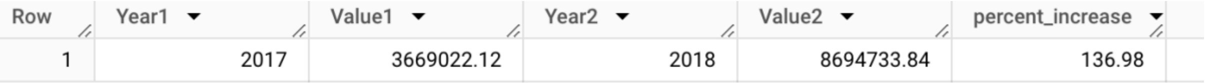
Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only).

Query:



```sql
WITH cte AS
(
  SELECT O.year, ROUND(SUM(P.payment_value), 2) AS total_payment_value
  FROM (
    SELECT *,
      EXTRACT(year FROM order_purchase_timestamp) as year,
      EXTRACT(month FROM order_purchase_timestamp) as month
    FROM `target-5.tables.orders`) AS O
  INNER JOIN `target-5.tables.payments` AS P
  USING (order_id)
  WHERE O.year BETWEEN 2017 AND 2018 AND O.month BETWEEN 1 AND 8
  GROUP BY year
)
SELECT
 T1.year AS Year1, T1.total_payment_value AS Value1,
 T2.year AS Year2, T2.total_payment_value AS Value2,
 ROUND(((T2.total_payment_value -
T1.total_payment_value)/T1.total_payment_value)*100,2) AS percent_increase
FROM cte AS T1, cte AS T2
WHERE T1.year < T2.year;
```

Output:

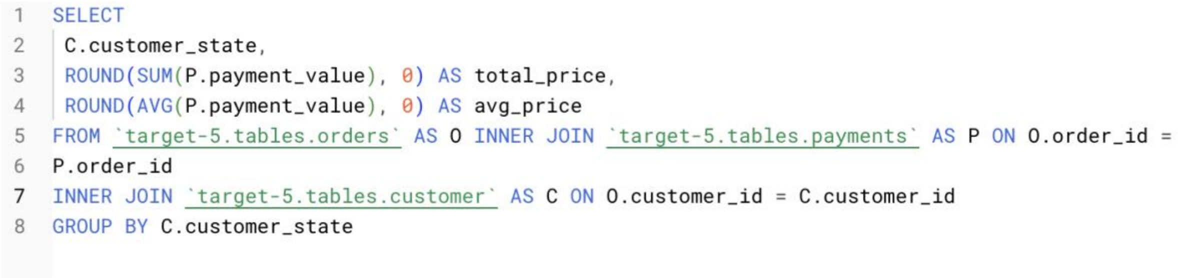


Row	Year1	Value1	Year2	Value2	percent_increase
1	2017	3669022.12	2018	8694733.84	136.98

Insights: The cost of orders has increased by around 137% from year 2017 to 2018, which is a huge increase, more than double.

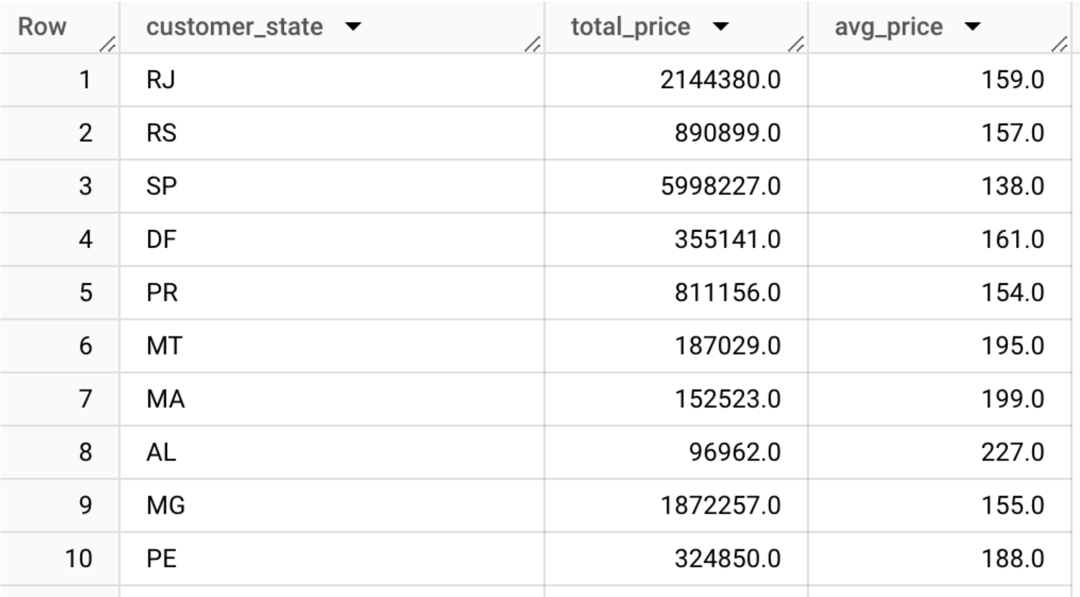
Calculate the Total & Average value of order price for each state.

Query:



```
SELECT
 C.customer_state,
 ROUND(SUM(P.payment_value), 0) AS total_price,
 ROUND(AVG(P.payment_value), 0) AS avg_price
FROM `target-5.tables.orders` AS O INNER JOIN `target-5.tables.payments` AS P ON O.order_id =
P.order_id
INNER JOIN `target-5.tables.customer` AS C ON O.customer_id = C.customer_id
GROUP BY C.customer_state
```

Output:

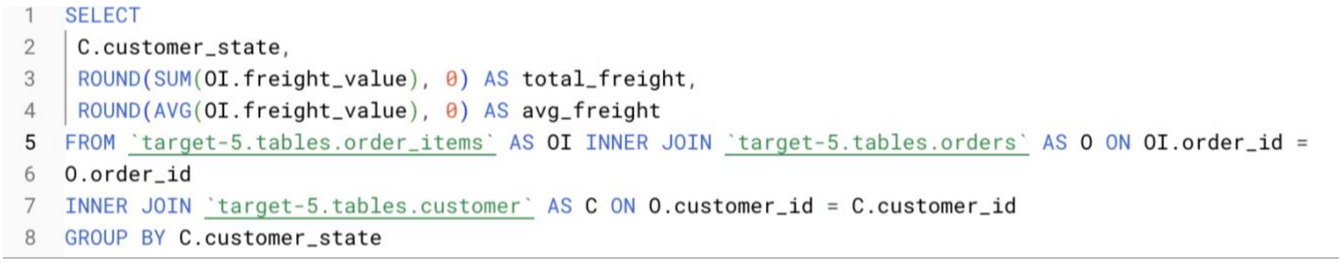


Row	customer_state	total_price	avg_price
1	RJ	2144380.0	159.0
2	RS	890899.0	157.0
3	SP	5998227.0	138.0
4	DF	355141.0	161.0
5	PR	811156.0	154.0
6	MT	187029.0	195.0
7	MA	152523.0	199.0
8	AL	96962.0	227.0
9	MG	1872257.0	155.0
10	PE	324850.0	188.0

Insights: The total price is highest for São Paulo and this is expected as the number of customers are also highest in São Paulo. The average price is around 200.

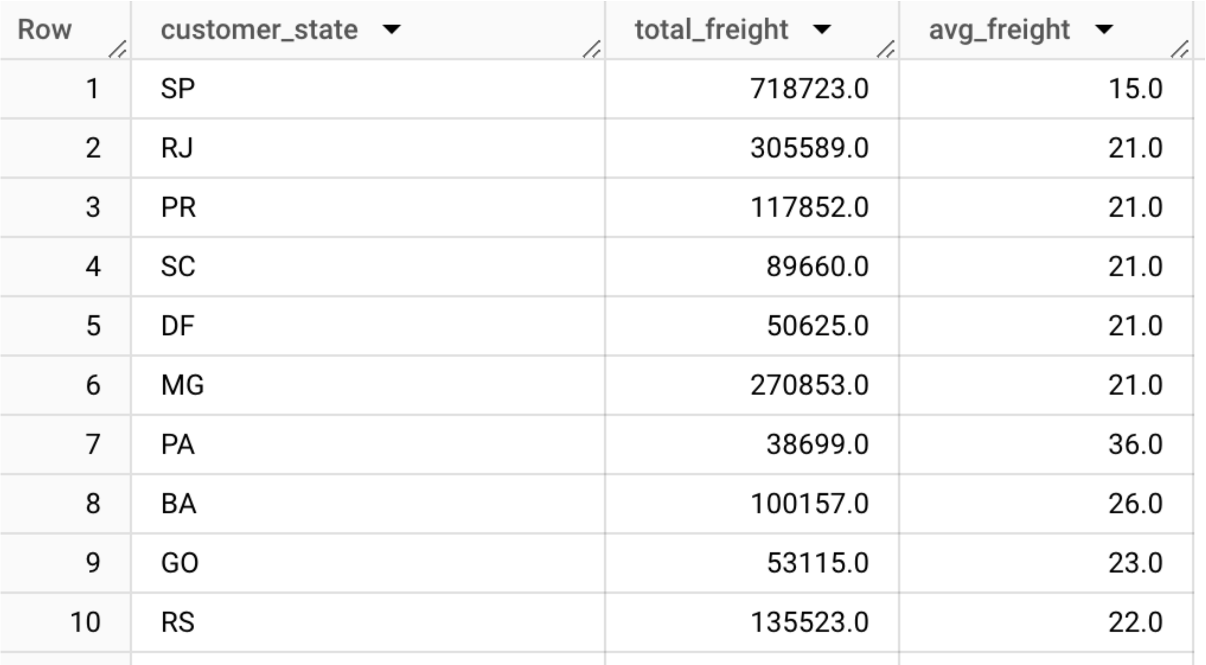
Calculate the Total & Average value of order freight for each state.

Query:



```
SELECT
 C.customer_state,
 ROUND(SUM(OI.freight_value), 0) AS total_freight,
 ROUND(AVG(OI.freight_value), 0) AS avg_freight
FROM `target-5.tables.order_items` AS OI INNER JOIN `target-5.tables.orders` AS O ON OI.order_id =
O.order_id
INNER JOIN `target-5.tables.customer` AS C ON O.customer_id = C.customer_id
GROUP BY C.customer_state
```

Output:



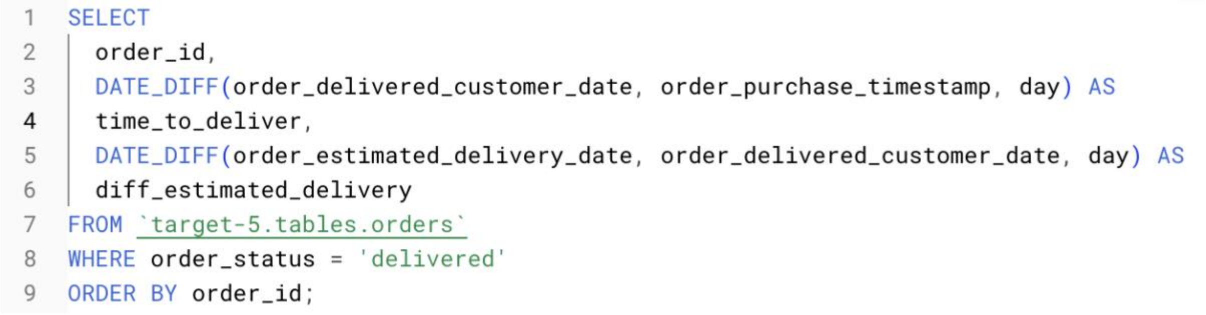
Row	customer_state	total_freight	avg_freight
1	SP	718723.0	15.0
2	RJ	305589.0	21.0
3	PR	117852.0	21.0
4	SC	89660.0	21.0
5	DF	50625.0	21.0
6	MG	270853.0	21.0
7	PA	38699.0	36.0
8	BA	100157.0	26.0
9	GO	53115.0	23.0
10	RS	135523.0	22.0

Insights: The total freight, following the similar pattern as total price, is highest for São Paulo. The average is around 30.

Analysis based on sales, freight and delivery time

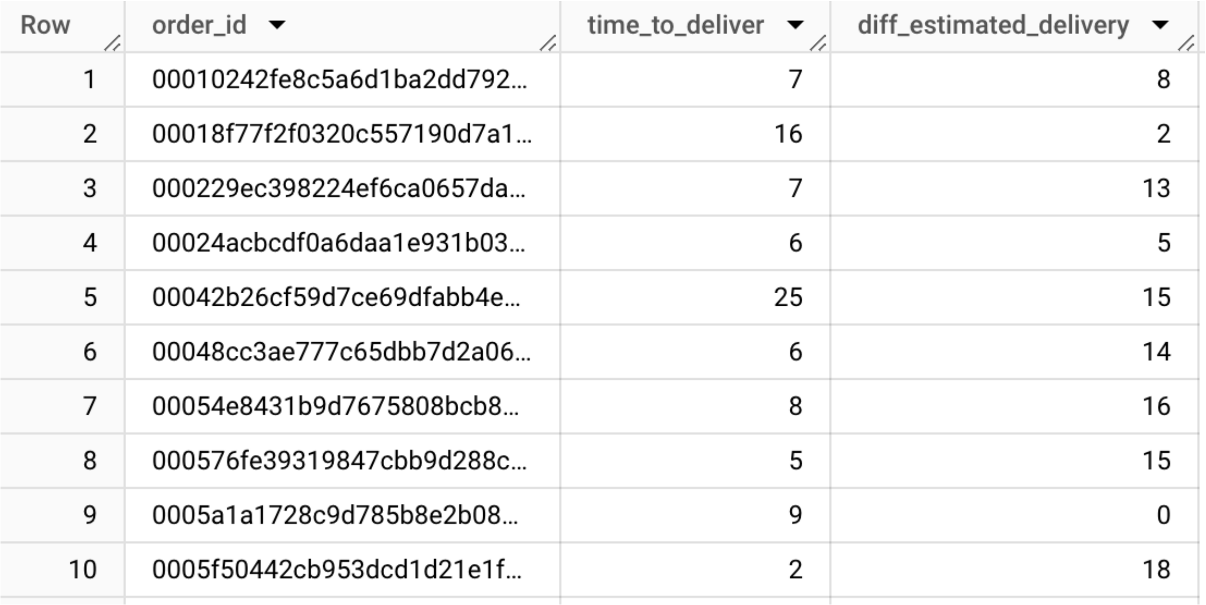
Find the no. of days taken to deliver each order from the order's purchase date as delivery time.

Query:



```
SELECT
  order_id,
  DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, day) AS
  time_to_deliver,
  DATE_DIFF(order_estimated_delivery_date, order_delivered_customer_date, day) AS
  diff_estimated_delivery
FROM `target-5.tables.orders`
WHERE order_status = 'delivered'
ORDER BY order_id;
```

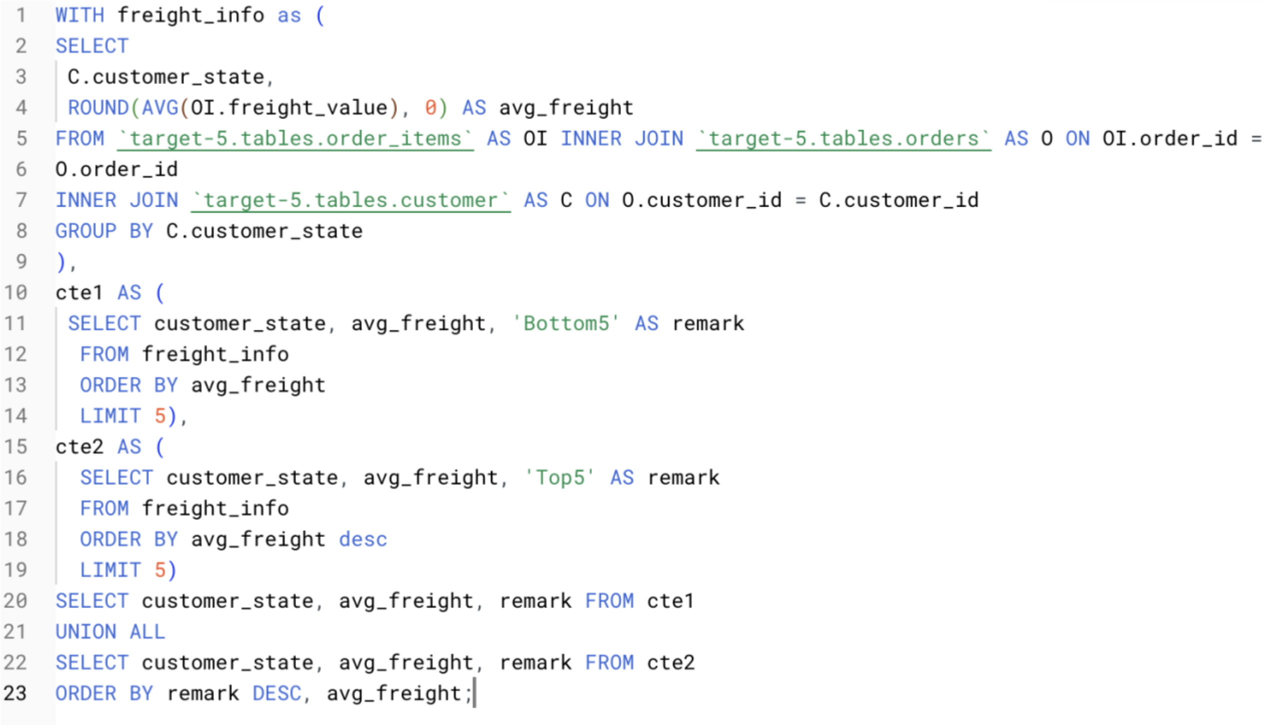
Output:



Row	order_id	time_to_deliver	diff_estimated_delivery
1	00010242fe8c5a6d1ba2dd792...	7	8
2	00018f77f2f0320c557190d7a1...	16	2
3	000229ec398224ef6ca0657da...	7	13
4	00024acbcdf0a6daa1e931b03...	6	5
5	00042b26cf59d7ce69dfabb4e...	25	15
6	00048cc3ae777c65dbb7d2a06...	6	14
7	00054e8431b9d7675808bcb8...	8	16
8	000576fe39319847cbb9d288c...	5	15
9	0005a1a1728c9d785b8e2b08...	9	0
10	0005f50442cb953dcd1d21e1f...	2	18

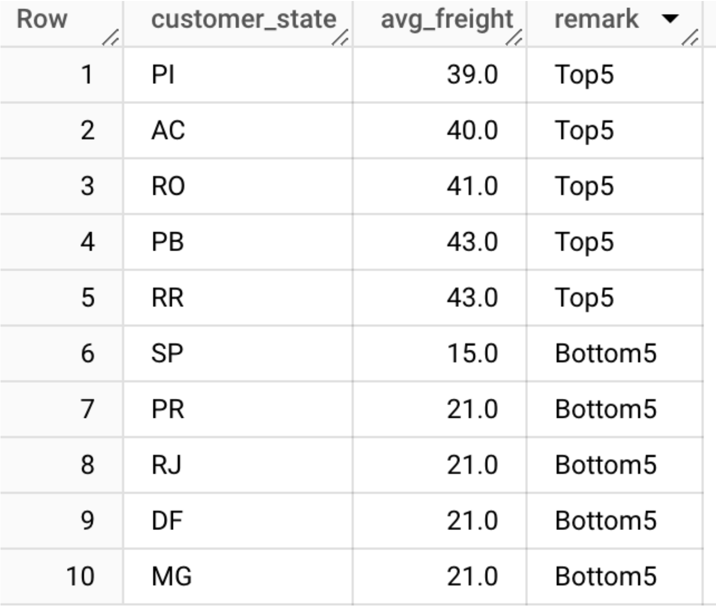
Find out the top 5 states with the highest & lowest average freight value.

Query:



```
WITH freight_info as (
SELECT
 C.customer_state,
 ROUND(AVG(OI.freight_value), 0) AS avg_freight
FROM `target-5.tables.order_items` AS OI INNER JOIN `target-5.tables.orders` AS O ON OI.order_id =
O.order_id
INNER JOIN `target-5.tables.customer` AS C ON O.customer_id = C.customer_id
GROUP BY C.customer_state
),
cte1 AS (
 SELECT customer_state, avg_freight, 'Bottom5' AS remark
  FROM freight_info
  ORDER BY avg_freight
  LIMIT 5),
cte2 AS (
  SELECT customer_state, avg_freight, 'Top5' AS remark
  FROM freight_info
  ORDER BY avg_freight desc
  LIMIT 5)
SELECT customer_state, avg_freight, remark FROM cte1
UNION ALL
SELECT customer_state, avg_freight, remark FROM cte2
ORDER BY remark DESC, avg_freight;
```

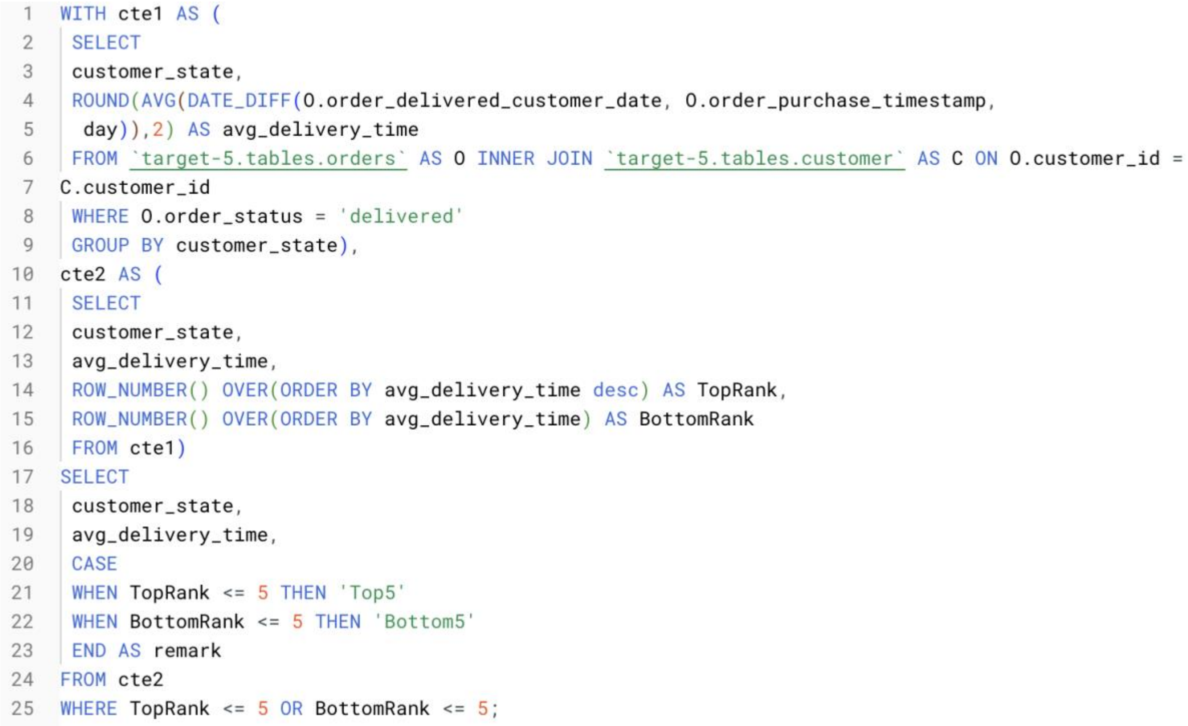
Output:



Row	customer_state	avg_freight	remark
1	PI	39.0	Top5
2	AC	40.0	Top5
3	RO	41.0	Top5
4	PB	43.0	Top5
5	RR	43.0	Top5
6	SP	15.0	Bottom5
7	PR	21.0	Bottom5
8	RJ	21.0	Bottom5
9	DF	21.0	Bottom5
10	MG	21.0	Bottom5

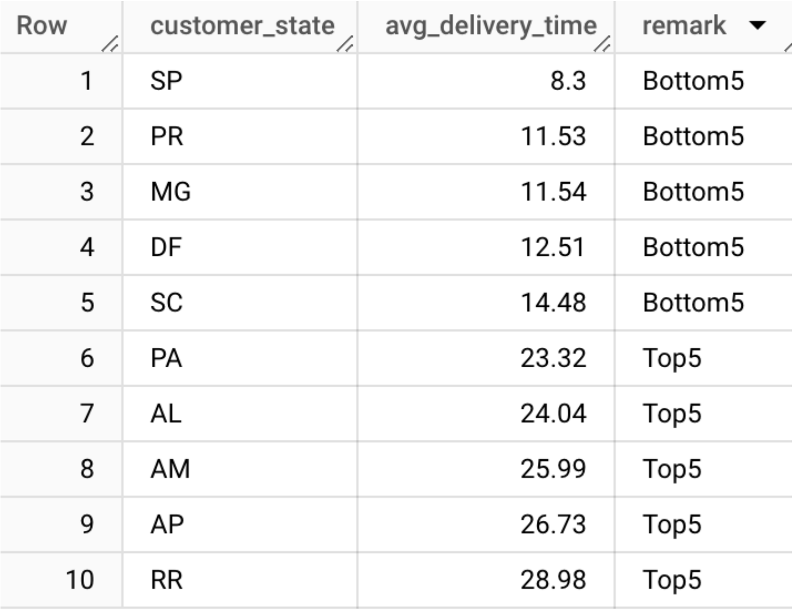
Find out the top 5 states with the highest & lowest average delivery time

Query:



```sql
WITH cte1 AS (
  SELECT
  customer_state,
  ROUND(AVG(DATE_DIFF(O.order_delivered_customer_date, O.order_purchase_timestamp,
   day)),2) AS avg_delivery_time
  FROM `target-5.tables.orders` AS O INNER JOIN `target-5.tables.customer` AS C ON O.customer_id =
C.customer_id
  WHERE O.order_status = 'delivered'
  GROUP BY customer_state),
cte2 AS (
  SELECT
  customer_state,
  avg_delivery_time,
  ROW_NUMBER() OVER(ORDER BY avg_delivery_time desc) AS TopRank,
  ROW_NUMBER() OVER(ORDER BY avg_delivery_time) AS BottomRank
  FROM cte1)
SELECT
  customer_state,
  avg_delivery_time,
  CASE
  WHEN TopRank <= 5 THEN 'Top5'
  WHEN BottomRank <= 5 THEN 'Bottom5'
  END AS remark
FROM cte2
WHERE TopRank <= 5 OR BottomRank <= 5;
```

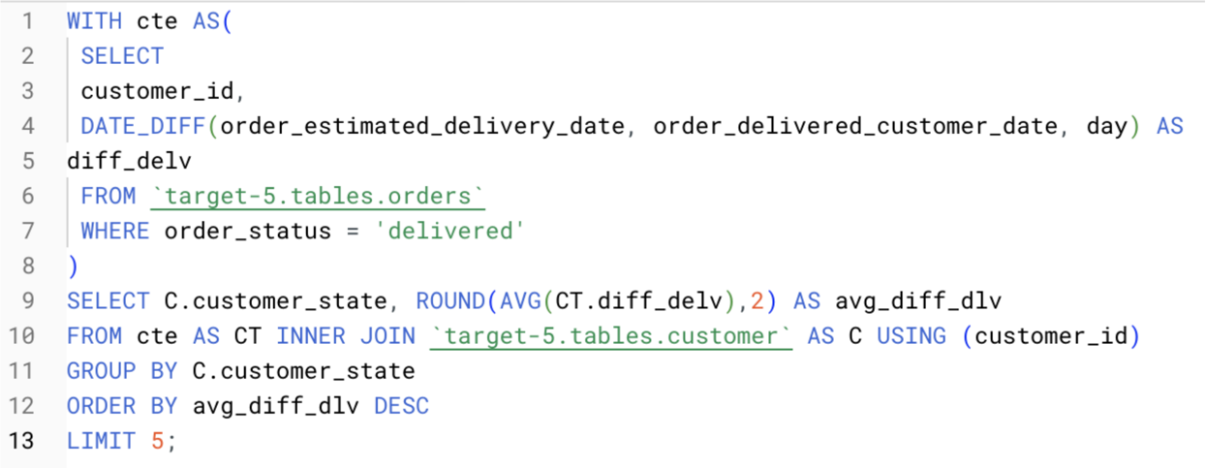
Output:



Row	customer_state	avg_delivery_time	remark
1	SP	8.3	Bottom5
2	PR	11.53	Bottom5
3	MG	11.54	Bottom5
4	DF	12.51	Bottom5
5	SC	14.48	Bottom5
6	PA	23.32	Top5
7	AL	24.04	Top5
8	AM	25.99	Top5
9	AP	26.73	Top5
10	RR	28.98	Top5

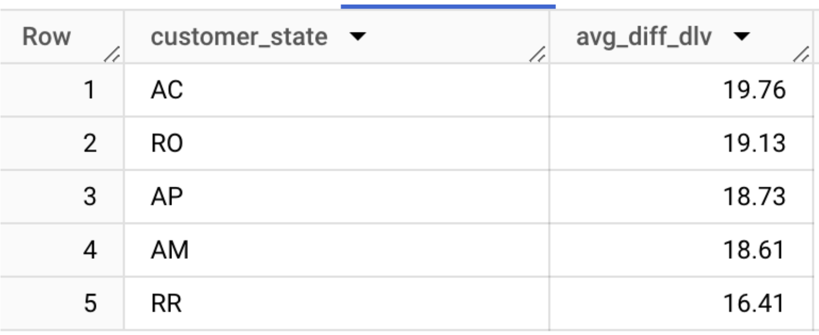
Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery.

Query:



```
WITH cte AS(
  SELECT
  customer_id,
  DATE_DIFF(order_estimated_delivery_date, order_delivered_customer_date, day) AS
diff_delv
  FROM `target-5.tables.orders`
  WHERE order_status = 'delivered'
)
SELECT C.customer_state, ROUND(AVG(CT.diff_delv),2) AS avg_diff_dlv
FROM cte AS CT INNER JOIN `target-5.tables.customer` AS C USING (customer_id)
GROUP BY C.customer_state
ORDER BY avg_diff_dlv DESC
LIMIT 5;
```

Output:

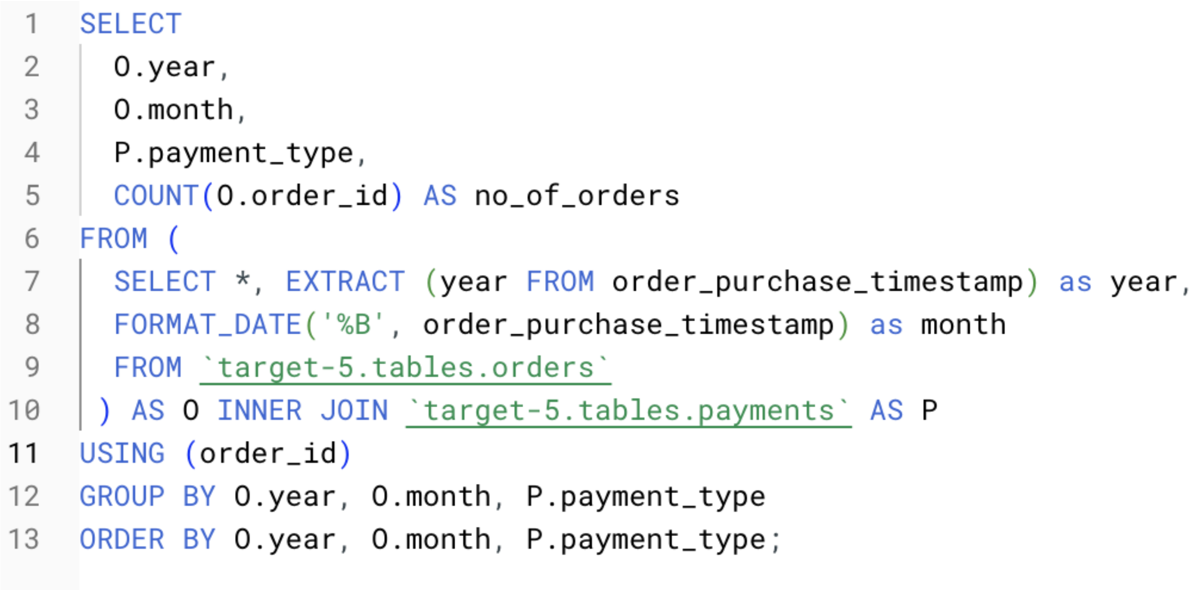


Row	customer_state	avg_diff_dlv
1	AC	19.76
2	RO	19.13
3	AP	18.73
4	AM	18.61
5	RR	16.41

Analysis based on the payments

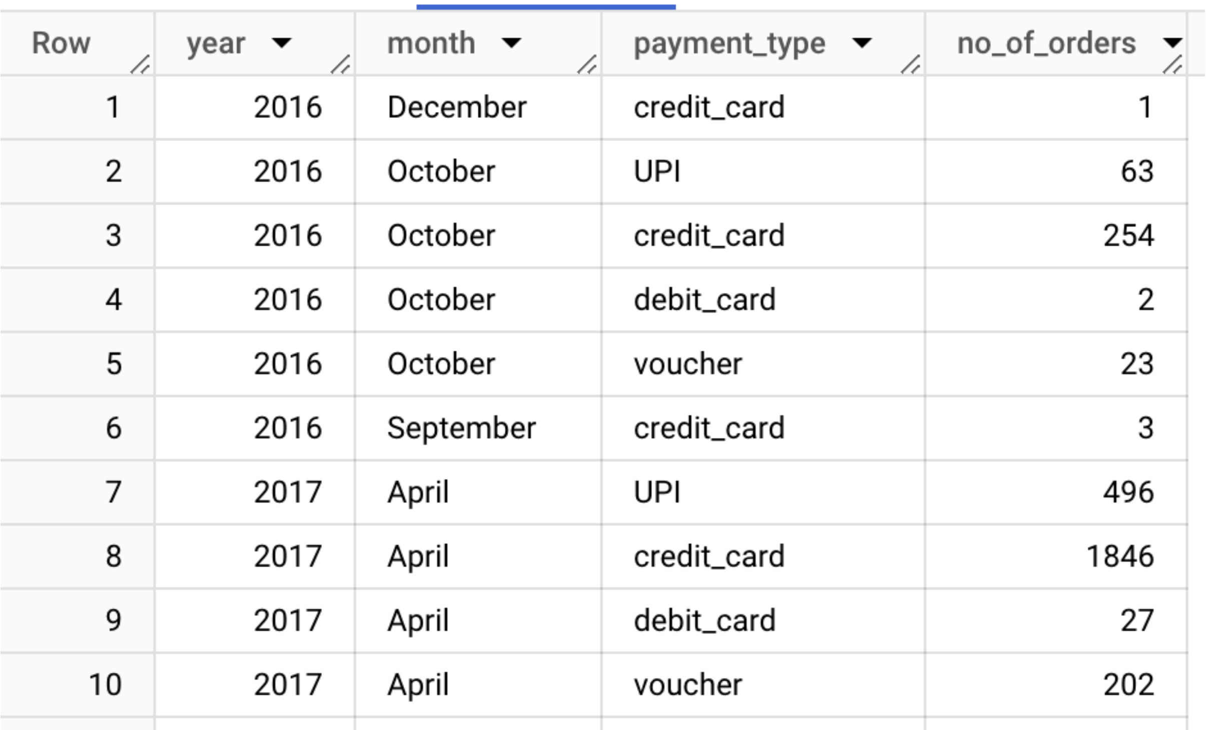
Find the month on month no. of orders placed using different payment types.

Query:



```sql
SELECT
  O.year,
  O.month,
  P.payment_type,
  COUNT(O.order_id) AS no_of_orders
FROM (
  SELECT *, EXTRACT (year FROM order_purchase_timestamp) as year,
  FORMAT_DATE('%B', order_purchase_timestamp) as month
  FROM `target-5.tables.orders`
 ) AS O INNER JOIN `target-5.tables.payments` AS P
USING (order_id)
GROUP BY O.year, O.month, P.payment_type
ORDER BY O.year, O.month, P.payment_type;
```

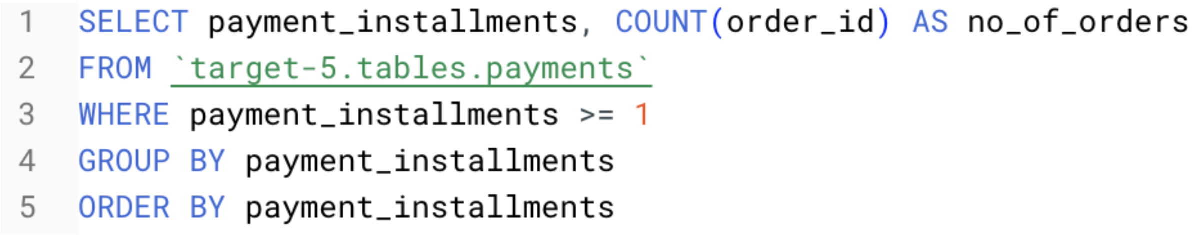
Output:



Row	year	month	payment_type	no_of_orders
1	2016	December	credit_card	1
2	2016	October	UPI	63
3	2016	October	credit_card	254
4	2016	October	debit_card	2
5	2016	October	voucher	23
6	2016	September	credit_card	3
7	2017	April	UPI	496
8	2017	April	credit_card	1846
9	2017	April	debit_card	27
10	2017	April	voucher	202

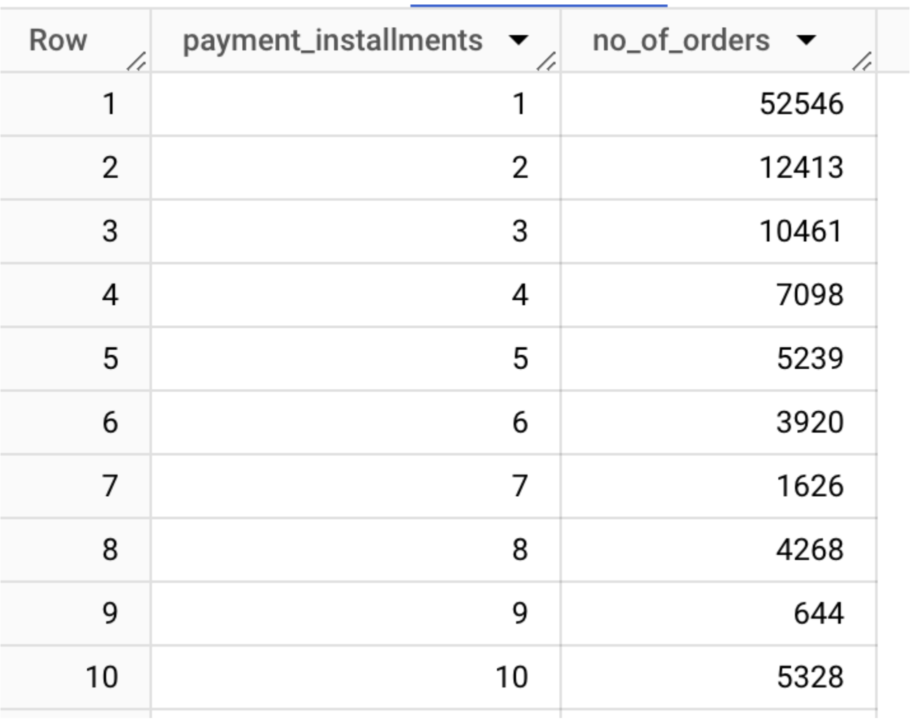
Find the no. of orders placed on the basis of the payment instalments that have been paid.

Query:



```
SELECT payment_installments, COUNT(order_id) AS no_of_orders
FROM `target-5.tables.payments`
WHERE payment_installments >= 1
GROUP BY payment_installments
ORDER BY payment_installments
```

Output:



Row	payment_installments	no_of_orders
1	1	52546
2	2	12413
3	3	10461
4	4	7098
5	5	5239
6	6	3920
7	7	1626
8	8	4268
9	9	644
10	10	5328