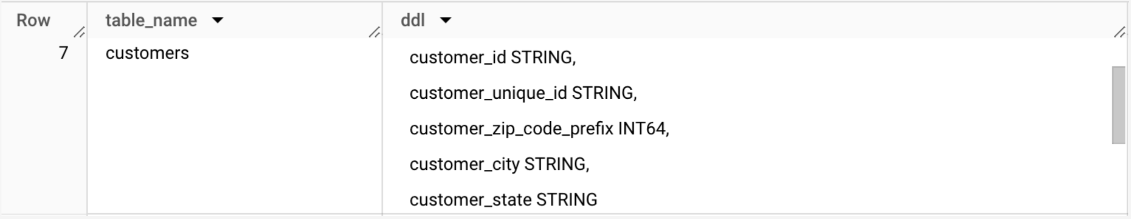
1. Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset:

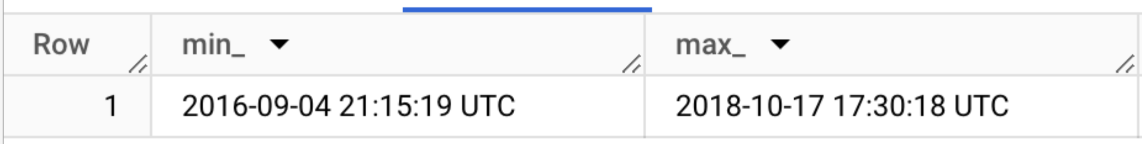
1. Data type of all columns in the "customers" table.

Query	SELECT table_name, ddl FROM scaler-
	dsml-396817.Target.INFORMATION_SCHEMA.TABLES;

Output	 Row | table_name | ddl 7 | customers | customer_id STRING, customer_unique_id STRING, customer_zip_code_prefix INT64, customer_city STRING, customer_state STRING
Insight	Data type for column_id, customer_unique_id, customer_city, and customer_state is STRING. Date type customer_zip_code_prefix is INTEGER.

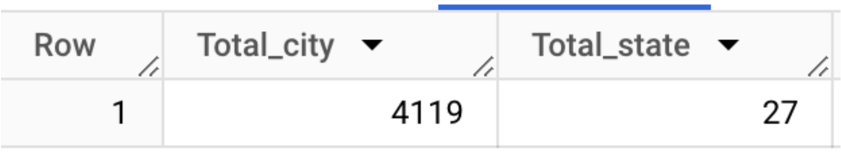
2. Get the time range between which the orders were placed.

Query	SELECT
	MIN(order_purchase_timestamp) as min_,
	MAX(order_purchase_timestamp) as max_
	from Target.orders;

Output	 Row | min_ | max_ 1 | 2016-09-04 21:15:19 UTC | 2018-10-17 17:30:18 UTC
Insight	The time range for orders place is from **4th September, 2016** to **17th October, 2018**.

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

Query	SELECT COUNT (distinct c.customer_city) as Total_city,
	COUNT (distinct c.customer_state) as Total_state
	FROM Target.orders o join Target.customers c
	ON c.customer_id=o.customer_id

Output	 Row | Total_city | Total_state 1 | 4119 | 27
Insight	• In order to get records for the given period we join customer table to orders table. • Count and Distinct function counts the unique number for cities and states of customers. • Orders to the given period came from 4119 cities and 27 state.

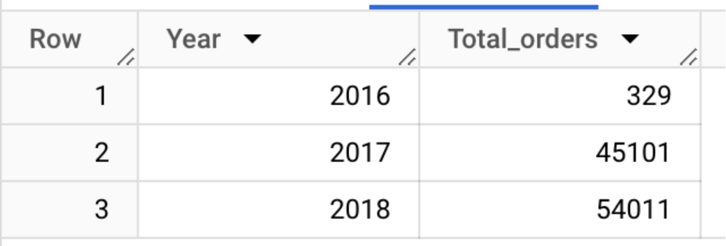
2. In-depth Exploration:

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

Query

```
SELECT Year, Count(*) as Total_orders
FROM (SELECT EXTRACT(YEAR FROM order_purchase_timestamp)
as Year, order_purchase_timestamp
FROM Target.orders) t
GROUP BY Year
ORDER BY Year;
```

Output



Row	Year	Total_orders
1	2016	329
2	2017	45101
3	2018	54011

Insight

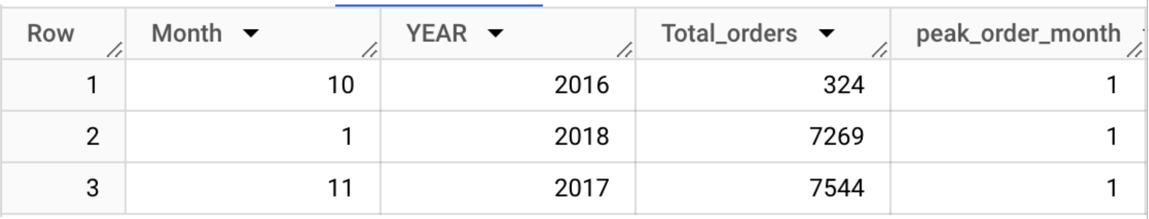
Considering that Target commenced its operation from 4th September, 2016;
total order purchased in 2016 (4 months) was 329.
In 2017 (12 months) it grew to 45,101.
In 2018 (10 months) it further grew to 54,011.
In conclusion, there is growing trend in the no. of orders placed over the past years.

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

Query

```
SELECT * FROM (SELECT t.Month, t.YEAR, Count(*) as
Total_orders,
DENSE_RANK() OVER(partition by t.Year Order by Count(*)
desc) as peak_order_month
FROM (SELECT EXTRACT(MONTH FROM order_purchase_timestamp)
as Month,
EXTRACT(YEAR FROM order_purchase_timestamp) as Year,
order_purchase_timestamp
FROM scaler-dsml-396817.Target.orders) t
GROUP BY t.Month,t.YEAR) m
where peak_order_month = 1;
```

Output



Row	Month	YEAR	Total_orders	peak_order_month
1	10	2016	324	1
2	1	2018	7269	1
3	11	2017	7544	1

Insight

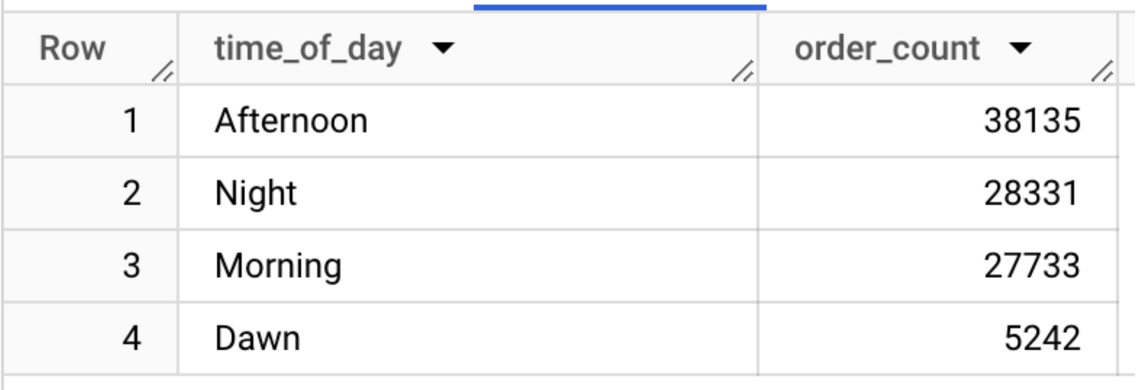
After counting the total order for each month it can be observed that peak order month for each year were as follows;
2016 - October
2017 - November
2018 - January
Due the limitation of data monthly seasonality cannot be determined at this point.

3. During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)
    1. 0-6 hrs : Dawn
    2. 7-12 hrs : Mornings
    3. 13-18 hrs : Afternoon
    4. 19-23 hrs : Night

Query

```
SELECT CASE
WHEN EXTRACT(HOUR FROM order_purchase_timestamp) >= 0 AND
EXTRACT(HOUR FROM order_purchase_timestamp) < 7 THEN
'Dawn'
WHEN EXTRACT(HOUR FROM order_purchase_timestamp) >= 7 AND
EXTRACT(HOUR FROM order_purchase_timestamp) < 13 THEN
'Morning'
WHEN EXTRACT(HOUR FROM order_purchase_timestamp) >= 13 AND
EXTRACT(HOUR FROM order_purchase_timestamp) < 19 THEN
'Afternoon'
ELSE 'Night'
END AS time_of_day, COUNT(*) as order_count
FROM scaler-dsml-396817.Target.orders
GROUP BY time_of_day
ORDER BY order_count DESC;
```

Output



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

Insight

The preferred time of the day for Brazilian customers to place order is in the Afternoon i.e. from 1PM to 6PM.

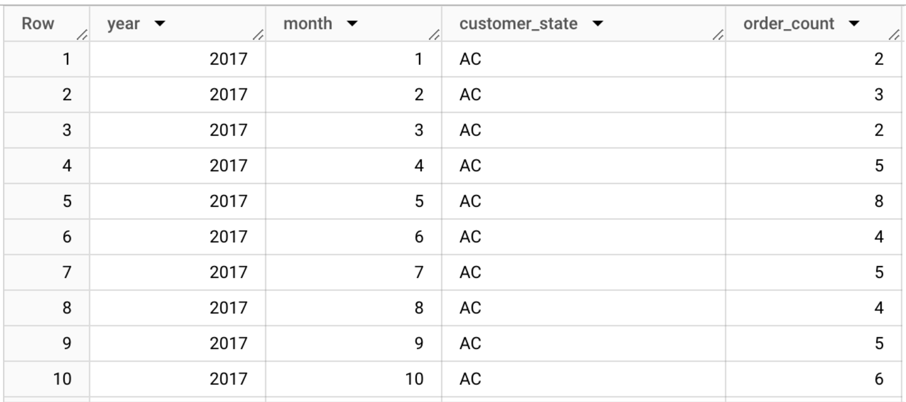
3. Evolution of E-commerce orders in the Brazil region:

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

Query

```
SELECT *, COUNT(*) AS order_count
FROM (SELECT EXTRACT(YEAR FROM order_purchase_timestamp)
AS year,
EXTRACT(MONTH FROM order_purchase_timestamp) AS month,
c.customer_state,
FROM scaler-dsml-396817.Target.orders o join scaler-
dsml-396817.Target.customers c
ON c.customer_id=o.customer_id) t
GROUP BY year,month,customer_state
ORDER BY customer_state,year,month;
```

Output



Row	year	month	customer_state	order_count
1	2017	1	AC	2
2	2017	2	AC	3
3	2017	3	AC	2
4	2017	4	AC	5
5	2017	5	AC	8
6	2017	6	AC	4
7	2017	7	AC	5
8	2017	8	AC	4
9	2017	9	AC	5
10	2017	10	AC	6

Insight

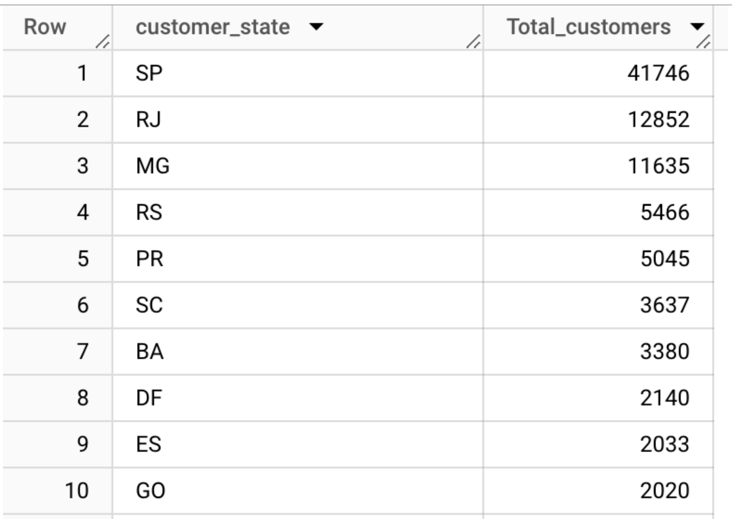
Here is the month on month no. of orders placed in each state.

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

Query

```
select customer_state, COUNT(distinct customer_id) as
Total_customers
from scaler-dsml-396817.Target.customers
group by customer_state
order by Total_customers desc;
```

Output



Row	customer_state	Total_customers
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

Insight

Heres the no. of unique customers distributed across all the states.
SP has highest no. of unique customers.

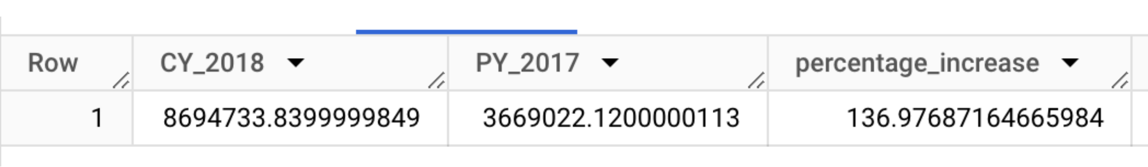
4. Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight and others.

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.

Query

```
WITH order_cost AS (
SELECT
EXTRACT(YEAR FROM o.order_purchase_timestamp) AS
order_year,
EXTRACT(MONTH FROM o.order_purchase_timestamp) AS
order_month,
SUM(p.payment_value) as Total_cost
FROM scaler-dsml-396817.Target.orders o JOIN scaler-
dsml-396817.Target.payments p ON p.order_id=o.order_id
WHERE EXTRACT(YEAR FROM o.order_purchase_timestamp) IN
(2017,2018) AND
EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1
AND 8
GROUP BY EXTRACT(YEAR FROM o.order_purchase_timestamp),
EXTRACT(MONTH FROM o.order_purchase_timestamp)
),
Cost_per_year as (
SELECT
SUM(CASE WHEN order_year=2018 THEN Total_cost END) as
CY_2018,
SUM(CASE WHEN order_year=2017 THEN Total_cost END) as
PY_2017
FROM order_cost
)
SELECT *,((CY_2018 - PY_2017) / PY_2017) * 100 AS
percentage_increase
FROM cost_per_year
```

Output



Row	CY_2018	PY_2017	percentage_increase
1	8694733.8399999849	3669022.1200000113	136.97687164665984

Insight

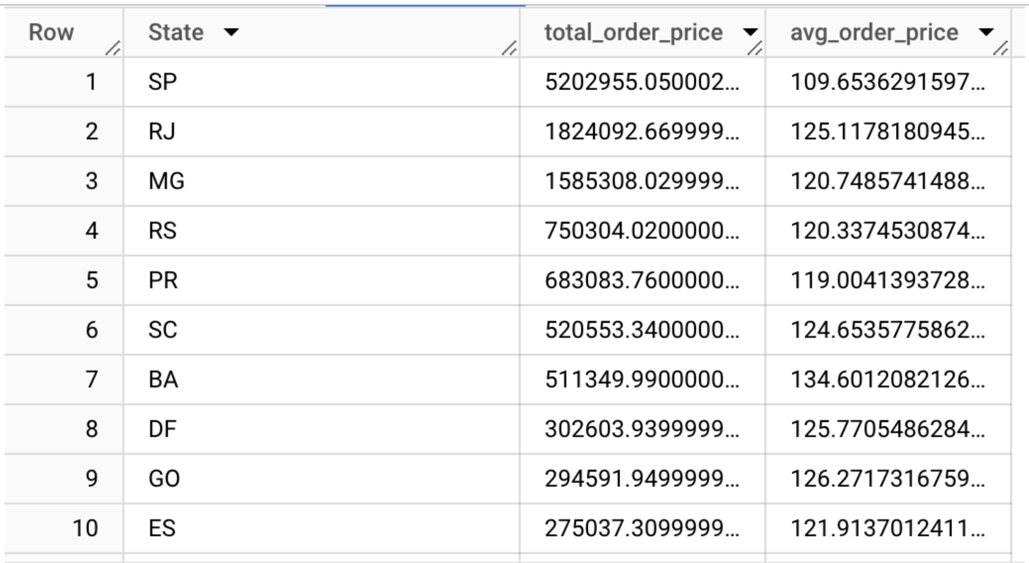
- I have calculated the total cost of orders for each year from January to August, using the payments and orders table.
- To calculate the percentage increase I have used the formula (Current year - Previous Year/ Previous Year)* 100
- We can conclude that there was 136.97% increase in the cost of orders for the given period.

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

Query

```
SELECT c.customer_state AS State,
SUM(i.price) as total_order_price,
AVG(i.price) as avg_order_price
FROM scaler-dsml-396817.Target.order_items i JOIN scaler-
dsml-396817.Target.orders o USING (order_id)
join scaler-dsml-396817.Target.customers c USING
(customer_id)
GROUP BY c.customer_state
ORDER BY total_order_price DESC, avg_order_price DESC;
```

Output



Row	State	total_order_price	avg_order_price
1	SP	5202955.050002...	109.6536291597...
2	RJ	1824092.669999...	125.1178180945...
3	MG	1585308.029999...	120.7485741488...
4	RS	750304.0200000...	120.3374530874...
5	PR	683083.7600000...	119.0041393728...
6	SC	520553.3400000...	124.6535775862...
7	BA	511349.9900000...	134.6012082126...
8	DF	302603.9399999...	125.7705486284...
9	GO	294591.9499999...	126.2717316759...
10	ES	275037.3099999...	121.9137012411...

Insight

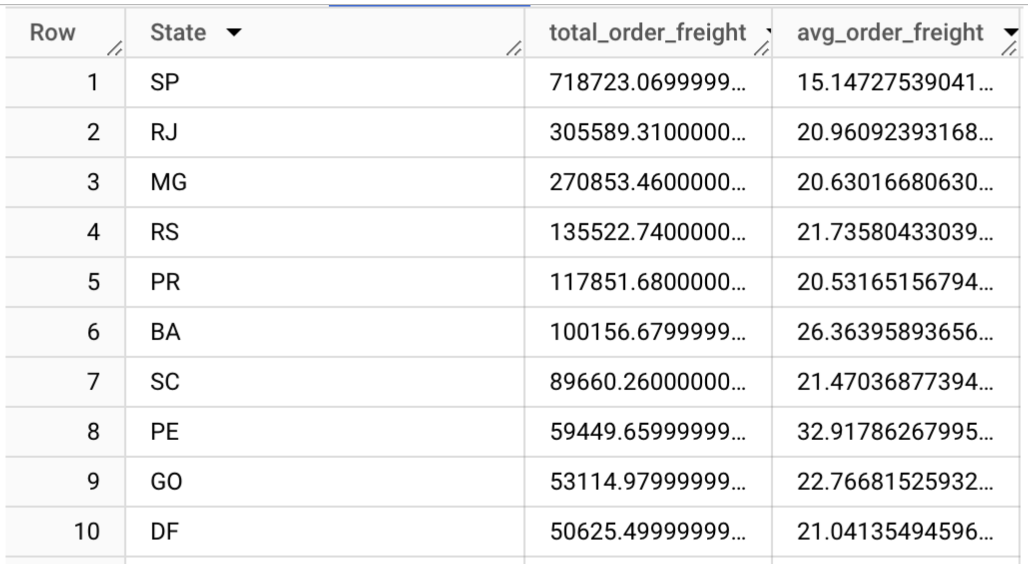
- In order to extract the state and order price information we have joined the order_item table to customers table.
- I have used Group by function to club the records as per states
- SP has the highest total order price i.e. 5202955.05 with average order price of 109.65

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

Query

```
SELECT c.customer_state AS State,
SUM(i.freight_value) as total_order_freight,
AVG(i.freight_value) as avg_order_freight
FROM scaler-dsml-396817.Target.order_items i JOIN scaler-
dsml-396817.Target.orders o USING (order_id)
join scaler-dsml-396817.Target.customers c USING
(customer_id)
GROUP BY c.customer_state
ORDER BY total_order_freight DESC, avg_order_freight DESC;
```

Output



Row	State	total_order_freight	avg_order_freight
1	SP	718723.0699999...	15.14727539041...
2	RJ	305589.3100000...	20.96092393168...
3	MG	270853.4600000...	20.63016680630...
4	RS	135522.7400000...	21.73580433039...
5	PR	117851.6800000...	20.53165156794...
6	BA	100156.6799999...	26.36395893656...
7	SC	89660.26000000...	21.47036877394...
8	PE	59449.65999999...	32.91786267995...
9	GO	53114.97999999...	22.76681525932...
10	DF	50625.49999999...	21.04135494596...

Insight

- SP has the highest total order freight i.e. 718723.06 with average order freight of 15.14

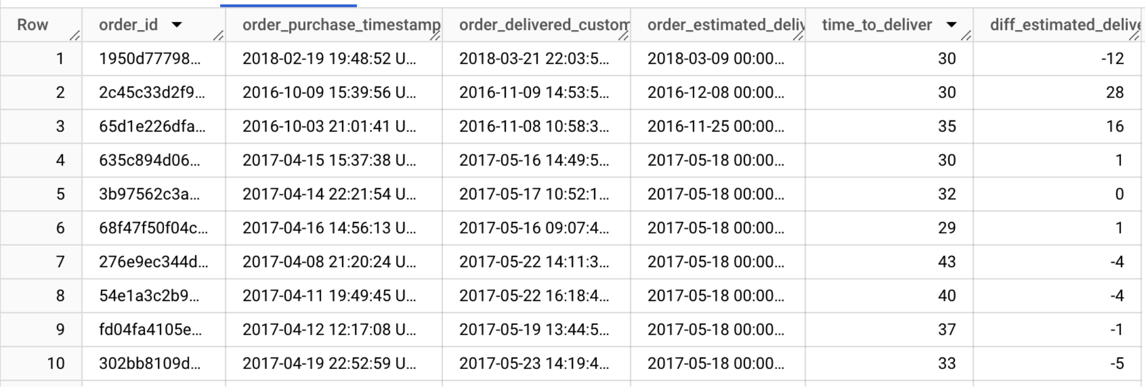
5. **Analysis based on sales, freight and delivery time.**
   1. Find the no. of days taken to deliver each order from the order’s purchase date as delivery time.
      Also, calculate the difference (in days) between the estimated & actual delivery date of an order.
      Do this in a single query.

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

Query

```
SELECT
order_id,
order_purchase_timestamp,
order_delivered_customer_date,
order_estimated_delivery_date,
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 scaler-dsml-396817.Target.orders;
```

Output



Row	order_id	order_purchase_timestamp	order_delivered_custom	order_estimated_deliv	time_to_deliver	diff_estimated_delive
1	1950d77798...	2018-02-19 19:48:52 U...	2018-03-21 22:03:5...	2018-03-09 00:00...	30	-12
2	2c45c33d2f9...	2016-10-09 15:39:56 U...	2016-11-09 14:53:5...	2016-12-08 00:00...	30	28
3	65d1e226dfa...	2016-10-03 21:01:41 U...	2016-11-08 10:58:3...	2016-11-25 00:00...	35	16
4	635c894d06...	2017-04-15 15:37:38 U...	2017-05-16 14:49:5...	2017-05-18 00:00...	30	1
5	3b97562c3a...	2017-04-14 22:21:54 U...	2017-05-17 10:52:1...	2017-05-18 00:00...	32	0
6	68f47f50f04c...	2017-04-16 14:56:13 U...	2017-05-16 09:07:4...	2017-05-18 00:00...	29	1
7	276e9ec344d...	2017-04-08 21:20:24 U...	2017-05-22 14:11:3...	2017-05-18 00:00...	43	-4
8	54e1a3c2b9...	2017-04-11 19:49:45 U...	2017-05-22 16:18:4...	2017-05-18 00:00...	40	-4
9	fd04fa4105e...	2017-04-12 12:17:08 U...	2017-05-19 13:44:5...	2017-05-18 00:00...	37	-1
10	302bb8109d...	2017-04-19 22:52:59 U...	2017-05-23 14:19:4...	2017-05-18 00:00...	33	-5

Insight

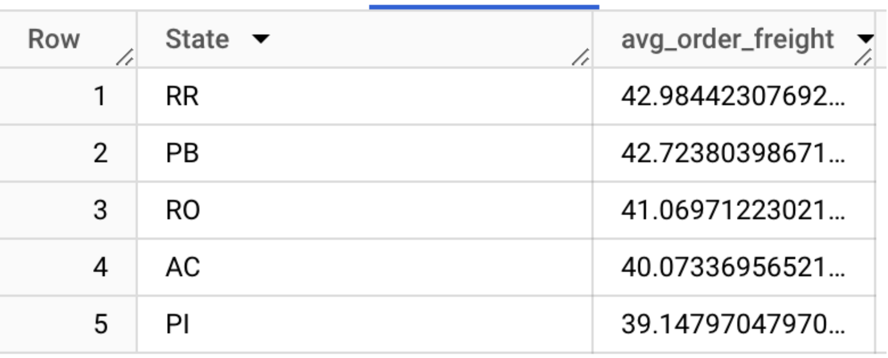
- I have used Date_diff function to calculate no. of days it took to deliver an order.
- time_to_deliver column represents total no. of days it took to fulfil an order
- diff_estimated_deliver represents the difference between the estimated delivery and actual delivery days.
- Positive integer denotes that the order was fulfilled before the time
- Negative integer denotes that delivery was delayed.

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

Query 1

```
SELECT c.customer_state AS State,
AVG(i.freight_value) as avg_order_freight
FROM scaler-dsml-396817.Target.order_items i JOIN scaler-
dsml-396817.Target.orders o USING (order_id)
join scaler-dsml-396817.Target.customers c USING
(customer_id)
GROUP BY c.customer_state
ORDER BY avg_order_freight DESC
LIMIT 5
```

Output 1



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

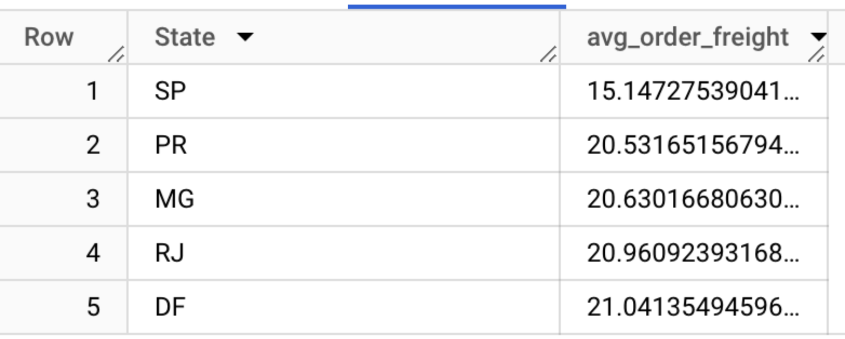
Insight

- These are the top 5 states with highest average freight value
- I have joined the customers table to order_item to get freight value and group them as per states.
- Limit function gets the top 5 states with highest freight values.

Query 2

```
SELECT c.customer_state AS State,
AVG(i.freight_value) as avg_order_freight
FROM scaler-dsml-396817.Target.order_items i JOIN scaler-
dsml-396817.Target.orders o USING (order_id)
join scaler-dsml-396817.Target.customers c USING
(customer_id)
GROUP BY c.customer_state
ORDER BY avg_order_freight
LIMIT 5
```

Output 2



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

Insight

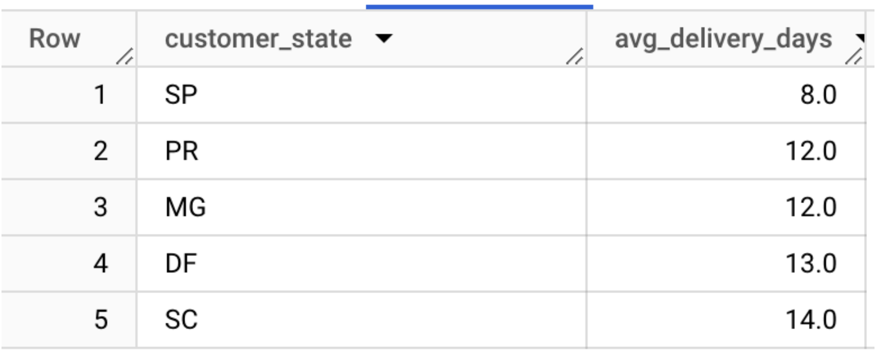
I have ordered the average freight value in ascending order to get top 5 states with lowest freight values.

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

Query 1

```
WITH deliverytime AS (
SELECT
c.customer_state,
DATE_DIFF(order_delivered_customer_date,
order_purchase_timestamp, DAY) AS time_to_deliver,
AVG(DATE_DIFF(order_delivered_customer_date,
order_purchase_timestamp, DAY)) OVER(PARTITION BY
customer_state) AS avg_delivery_days
FROM scaler-dsml-396817.Target.orders o JOIN scaler-
dsml-396817.Target.customers c
ON o.customer_id=c.customer_id
)
SELECT customer_state,
ROUND(avg_delivery_days,0) AS avg_delivery_days
FROM deliverytime
GROUP BY customer_state,avg_delivery_days
ORDER BY avg_delivery_days;
```

Output 1



Row	customer_state	avg_delivery_days
1	SP	8.0
2	PR	12.0
3	MG	12.0
4	DF	13.0
5	SC	14.0

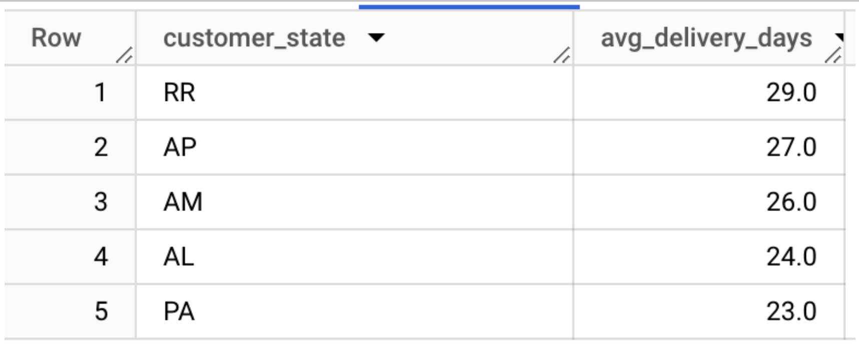
Insight

- These are the top 5 states with lowest delivery time
- Delivery time is calculated in days.
- I have used Round function to get the no. of days as absolute values
- I have joined the customers table to know the orders belong to which state and grouped them accordingly.

Query 2

```
WITH deliverytime AS (
SELECT
c.customer_state,
DATE_DIFF(order_delivered_customer_date,
order_purchase_timestamp, DAY) AS time_to_deliver,
AVG(DATE_DIFF(order_delivered_customer_date,
order_purchase_timestamp, DAY)) OVER(PARTITION BY
customer_state) AS avg_delivery_days
FROM scaler-dsml-396817.Target.orders o JOIN scaler-
dsml-396817.Target.customers c
ON o.customer_id=c.customer_id
)
SELECT customer_state,
ROUND(avg_delivery_days,0) AS avg_delivery_days
FROM deliverytime
GROUP BY customer_state,avg_delivery_days
ORDER BY avg_delivery_days desc
LIMIT 5;
```

Output 2



Row	customer_state	avg_delivery_days
1	RR	29.0
2	AP	27.0
3	AM	26.0
4	AL	24.0
5	PA	23.0

Insight

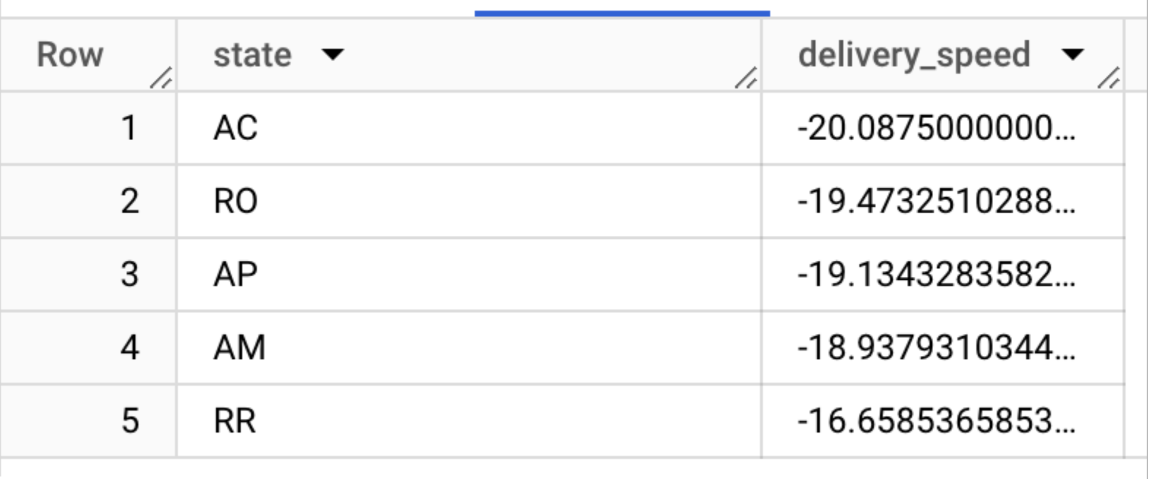
I have ordered the avg_delivery_days in descending order to get top 5 state with highest delivery time.

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.

Query

```
WITH state_delivery_speed as
(
SELECT
c.customer_state as state,
AVG(DATE_DIFF(o.order_delivered_customer_date,order_purchase_timestamp,day)) OVER(PARTITION BY c.customer_state) as
avg_delivery_time,
AVG(DATE_DIFF(o.order_estimated_delivery_date,order_purchase_timestamp,day)) OVER(PARTITION BY c.customer_state) as
avg_estimated_time
FROM scaler-dsml-396817.Target.orders o JOIN scaler-dsml-396817.Target.customers c
ON o.customer_id=c.customer_id
where order_delivered_customer_date is not null
)
SELECT
state,
AVG(avg_delivery_time-avg_estimated_time) as
delivery_speed
FROM state_delivery_speed
GROUP BY state,(avg_delivery_time- avg_estimated_time)
ORDER BY delivery_speed
LIMIT 5;
```

Output



Row	state	delivery_speed
1	AC	-20.0875000000...
2	RO	-19.4732510288...
3	AP	-19.1343283582...
4	AM	-18.9379310344...
5	RR	-16.6585365853...

Insight

- We calculate the average delivery speed for each state by finding the difference in days between the actual delivery date (order_delivered_customer_date) and the estimated delivery date (order_estimated_delivery_date) using the DATEDIFF function.
- We group the result by customer state
- To get the top 5 state we order the result in ascending order and set Limit to 5

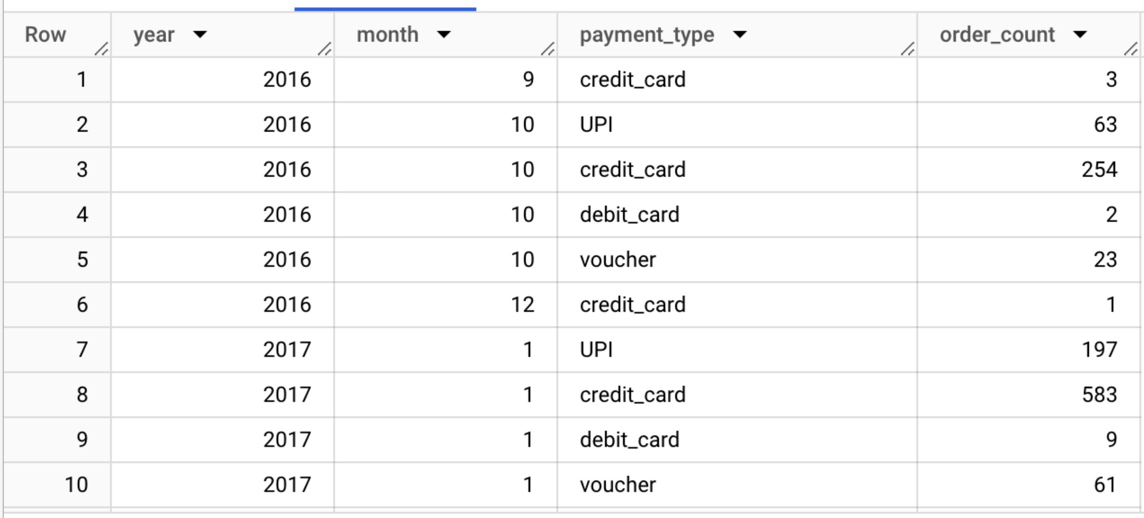
6. Analysis based on the payments:

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

Query

```
SELECT
EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
p.payment_type,
COUNT(*) AS order_count
FROM
scaler-dsml-396817.Target.orders o
JOIN scaler-dsml-396817.Target.payments p ON o.order_id =
p.order_id
GROUP BY year, month, payment_type
ORDER BY year, month, payment_type;
```

Output



Row	year	month	payment_type	order_count
1	2016	9	credit_card	3
2	2016	10	UPI	63
3	2016	10	credit_card	254
4	2016	10	debit_card	2
5	2016	10	voucher	23
6	2016	12	credit_card	1
7	2017	1	UPI	197
8	2017	1	credit_card	583
9	2017	1	debit_card	9
10	2017	1	voucher	61

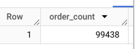
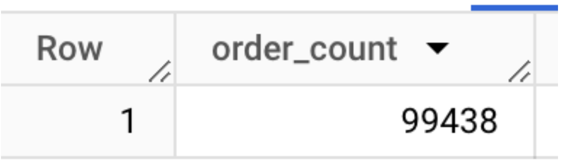
Insight

- We extract the year and month from the order_purchase_timestamp column using the EXTRACT function.
- We include the payment_type column to group the results by payment type.
- We count the number of orders for each combination of year, month, and payment type using COUNT(*).
- We group the results by year, month, and payment_type to get the month-on-month order counts for each payment type.

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

Query

```
SELECT count(distinct order_id) as order_count
FROM scaler-dsml-396817.Target.payments
WHERE payment_installments != 0;
```

Output 	 Row order_count 1 99438
Insight	• We use the WHERE clause to filter the data, considering only orders where payment_installments are greater than 0. • We use COUNT(DISTINCT order_id) to count the distinct order IDs, which eliminates duplicates. • This query will give you the total number of unique orders placed where payment installments are greater than 0.