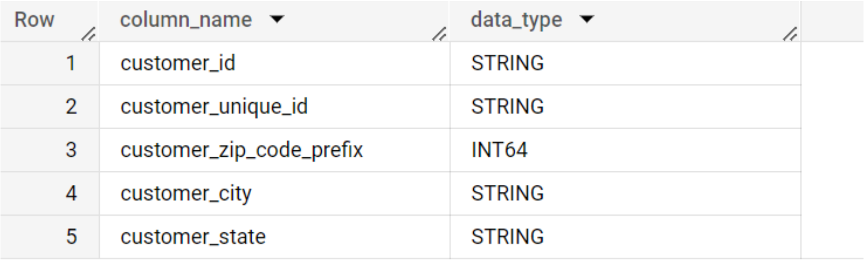
Business Case: Target SQL

Target is a globally renowned brand and a prominent retailer in the United States. This business case focuses on the operations of Target in Brazil and provides insightful information about 100,000 orders placed between 2016 and 2018. The information basically sheds light on various aspects of the business, such as order processing, pricing strategies, payment and shipping efficiency, customer demographics, product characteristics, and customer satisfaction levels.

1. **Description- 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.INFORMATION_SCHEMA.COLUMNS`
where table_name = 'customers'
```



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

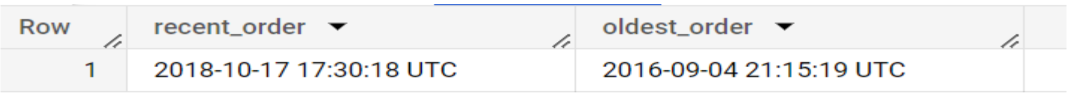
Insights-

a) Data types for the columns- customer_id, customer_unique_id, customer_city, customer_state is STRING while the column- customer_zip_code_prefix is of INT type.

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

Query1-

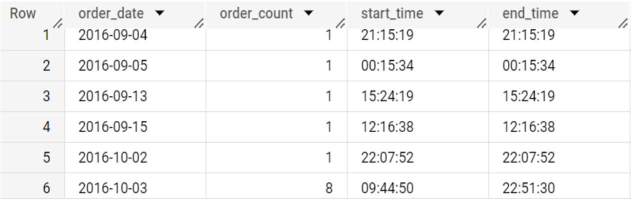
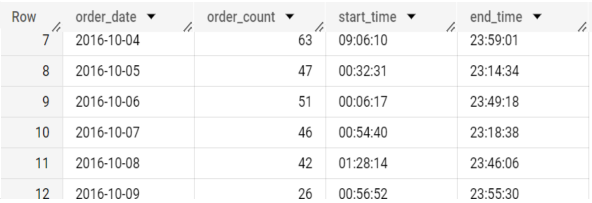
```
Select
max(order_purchase_timestamp) as recent_order,
min(order_purchase_timestamp) as oldest_order
from `target.orders`
```



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

Query2-

```
SELECT
order_date,
count(order_id) as order_count,
Min(order_time) as start_time,
Max(order_time) as end_time
FROM (
Select
order_id,
customer_id,
extract(date from order_purchase_timestamp) as order_date,
extract(TIME FROM order_purchase_timestamp ) as order_time
FROM `target.orders`
)
Group by order_date
order by order_date
```

Row	order_date	order_count	start_time	end_time
1	2016-09-04	1	21:15:19	21:15:19
2	2016-09-05	1	00:15:34	00:15:34
3	2016-09-13	1	15:24:19	15:24:19
4	2016-09-15	1	12:16:38	12:16:38
5	2016-10-02	1	22:07:52	22:07:52
6	2016-10-03	8	09:44:50	22:51:30
7	2016-10-04	63	09:06:10	23:59:01
8	2016-10-05	47	00:32:31	23:14:34
9	2016-10-06	51	00:06:17	23:49:18
10	2016-10-07	46	00:54:40	23:18:38
11	2016-10-08	42	01:28:14	23:46:06
12	2016-10-09	26	00:56:52	23:55:30

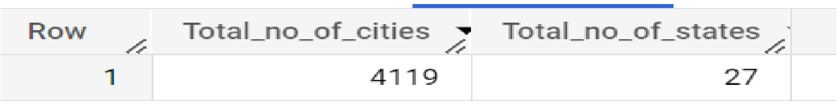
Insights-

a) The time-range for which the dataset is given comes out to be between 2016-09-04 and 2018-10-17.

3. Count the number of Cities and States in our dataset.

Query1-

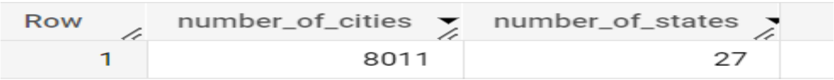
```
Select count(distinct customer_city) as Total_no_of_cities,
count(distinct customer_state) as Total_no_of_states
from
(
SELECT c.customer_id, o.order_id,
c.customer_city, c.customer_state,
from `target.customers` as c
Join `target.orders` as o
on c.customer_id = o.customer_id)
```



Row	Total_no_of_cities	Total_no_of_states
1	4119	27

Query2-

```
SELECT
count(distinct geolocation_city) as number_of_cities,
count(distinct geolocation_state) as number_of_states
from `target.geolocation`
```



Row	number_of_cities	number_of_states
1	8011	27

Insights-

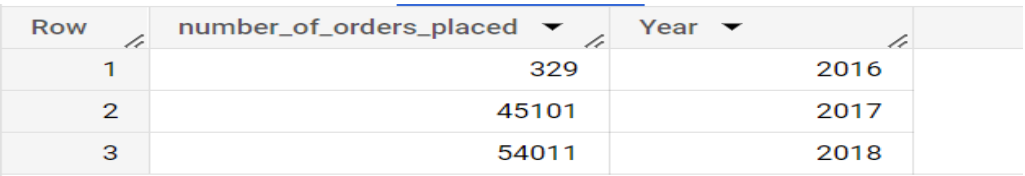
a) Based on orders, the total no. of cities and states for which the dataset is given comes out to be 8011 and 27 respectively.

2. **Description- In-depth Exploration:**

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

Query1-

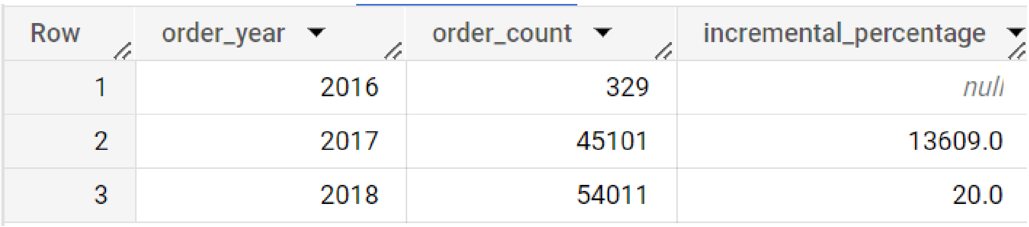
```
SELECT count(order_id) as number_of_orders_placed,
extract(Year from order_purchase_timestamp) as Year,
from `target.orders`
group by Year
order by number_of_orders_placed, year Asc
```



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

Query2-

```
WITH CTE as(
Select
a.order_year,
a.order_count,
lag(a.order_count) over(order by a.order_year) as prev_year_order_count
FROM (
SELECT
order_year,
count(order_id) as order_count,
FROM
(
SELECT
*,
EXTRACT(YEAR FROM order_purchase_timestamp) as order_year
FROM `target.orders`
)
Group by order_year
Order by order_year
) as a
order by order_year
)
SELECT
order_year,
order_count,
round(((order_count- prev_year_order_count)/prev_year_order_count)*100,0) as
incremental_percentage
FROM CTE
```



Row	order_year	order_count	incremental_percentage
1	2016	329	*null*
2	2017	45101	13609.0
3	2018	54011	20.0

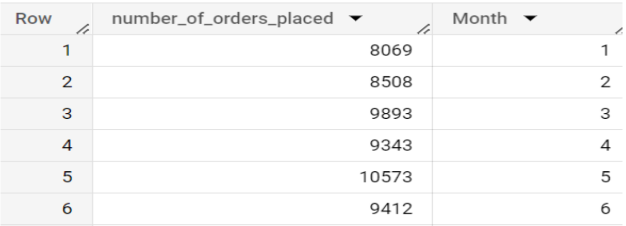
Insights-

a) Over the past years, year-wise growing trend in the no. of orders has been observed.

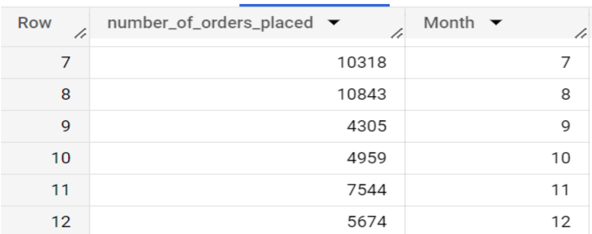
2. Can we see some kind of monthly seasonality in terms of the no. of orders being placed?

Query1-

```
SELECT count(order_id) as number_of_orders_placed,
extract(Month from order_purchase_timestamp) as Month
from `target.orders`
group by Month
order by month Asc
```



Row	number_of_orders_placed	Month
1	8069	1
2	8508	2
3	9893	3
4	9343	4
5	10573	5
6	9412	6



Row	number_of_orders_placed	Month
7	10318	7
8	10843	8
9	4305	9
10	4959	10
11	7544	11
12	5674	12

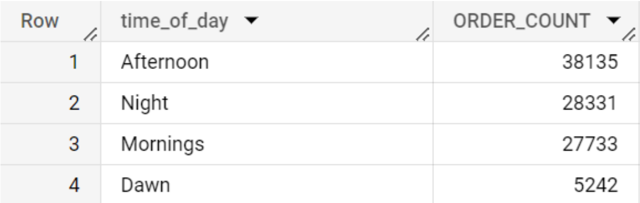
Insights-

a) On a seasonality basis, maximum no. of orders were placed in the month of August.

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

Query-

```
SELECT time_of_day,
COUNT(order_id) AS ORDER_COUNT
FROM (
SELECT order_id,
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 'Mornings'
WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 13 AND
18 THEN 'Afternoon'
WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 19 AND
23 THEN 'Night'
END AS time_of_day
FROM `target.orders`
) X
GROUP BY time_of_day
ORDER BY ORDER_COUNT DESC;
```



Row	time_of_day	ORDER_COUNT
1	Afternoon	38135
2	Night	28331
3	Mornings	27733
4	Dawn	5242

Insights-

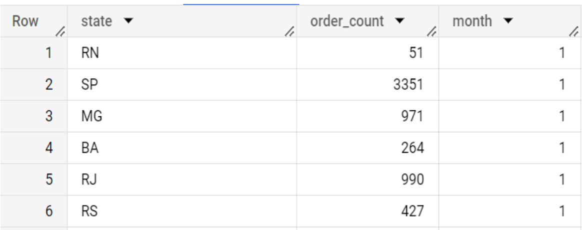
a) At the time of Afternoon, Brazilians have placed the max no. of orders.

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

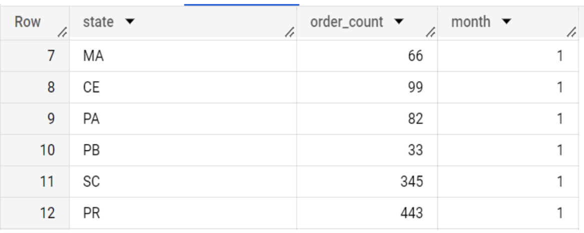
1. Get the month on month no. of orders placed in each state.

Query1-

```
Select distinct(c.customer_state) as state,
count(o.order_id) as order_count,
extract(Month from o.order_purchase_timestamp) as month
from `target.customers` as c
join `target.orders` as o
on c.customer_id=o.customer_id
group by state, month
order by month
```



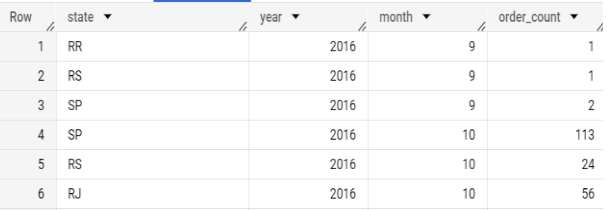
Row	state	order_count	month
1	RN	51	1
2	SP	3351	1
3	MG	971	1
4	BA	264	1
5	RJ	990	1
6	RS	427	1



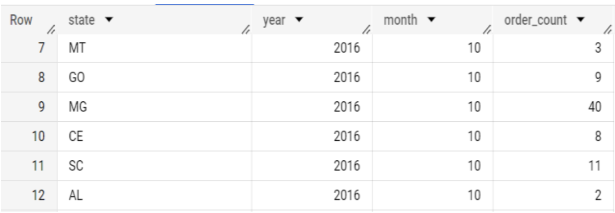
Row	state	order_count	month
7	MA	66	1
8	CE	99	1
9	PA	82	1
10	PB	33	1
11	SC	345	1
12	PR	443	1

Query2-

```
Select distinct(c.customer_state) as state,
extract(Year from o.order_purchase_timestamp) as year,
extract(Month from o.order_purchase_timestamp) as month,
count(o.order_id) as order_count
from `target.customers` as c
join `target.orders` as o
on c.customer_id=o.customer_id
group by state, month, year
order by year, month
```



Row	state	year	month	order_count
1	RR	2016	9	1
2	RS	2016	9	1
3	SP	2016	9	2
4	SP	2016	10	113
5	RS	2016	10	24
6	RJ	2016	10	56

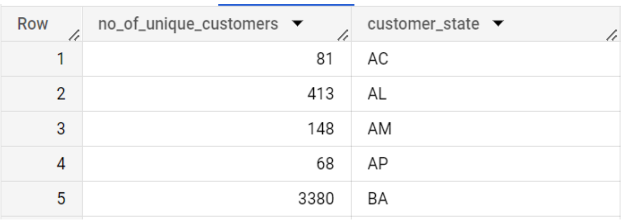


Row	state	year	month	order_count
7	MT	2016	10	3
8	GO	2016	10	9
9	MG	2016	10	40
10	CE	2016	10	8
11	SC	2016	10	11
12	AL	2016	10	2

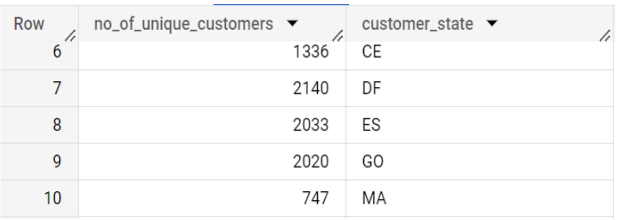
2. How are the customers distributed across all the states?

Query-

```
Select count(customer_unique_id) as no_of_unique_customers,
customer_state,
from  `target.customers`
group by customer_state
order by customer_state asc
```



Row	no_of_unique_customers	customer_state
1	81	AC
2	413	AL
3	148	AM
4	68	AP
5	3380	BA



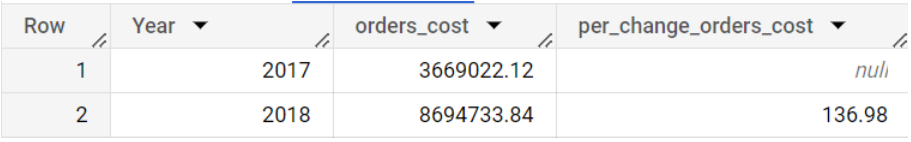
Row	no_of_unique_customers	customer_state
6	1336	CE
7	2140	DF
8	2033	ES
9	2020	GO
10	747	MA

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-

```
select Year, orders_cost,
round((orders_cost -lag(orders_cost) over (order by Year ) ) * 100/lag(orders_cost)
over (order by Year ),2) as per_change_orders_cost
from
(SELECT
Extract(YEAR from order_purchase_timestamp) as Year,
round(sum(payment_value),2) as orders_cost
FROM `target.orders` as o
INNER JOIN
`target.payments` as p
on o.order_id= p.order_id
where
EXTRACT(MONTH from order_purchase_timestamp) between 1 and 8
group by Year
) a
order by Year;
```

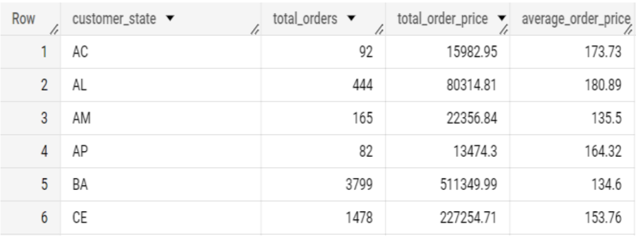


Row	Year	orders_cost	per_change_orders_cost
1	2017	3669022.12	null
2	2018	8694733.84	136.98

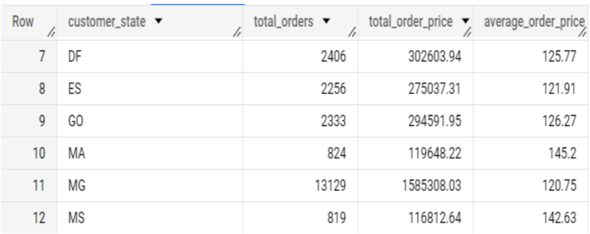
2.Calculate the Total & Average value of order price for each state.

Query-

```
SELECT c.customer_state,
count(o.order_id) as total_orders,
round(sum(oi.price),2) as total_order_price,
round(avg(oi.price),2) as average_order_price
from `target.customers` as c
join `target.orders` as o
on o.customer_id=c.customer_id
 join `target.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_orders	total_order_price	average_order_price
1	AC	92	15982.95	173.73
2	AL	444	80314.81	180.89
3	AM	165	22356.84	135.5
4	AP	82	13474.3	164.32
5	BA	3799	511349.99	134.6
6	CE	1478	227254.71	153.76



Row	customer_state	total_orders	total_order_price	average_order_price
7	DF	2406	302603.94	125.77
8	ES	2256	275037.31	121.91
9	GO	2333	294591.95	126.27
10	MA	824	119648.22	145.2
11	MG	13129	1585308.03	120.75
12	MS	819	116812.64	142.63

Insights-

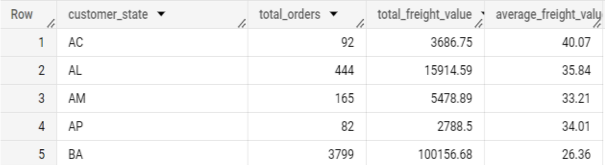
a) Highest average_order_price was 191.48 for the state-PB.

b) Highest total_order_price value was 5202955.05 for the state-SP.

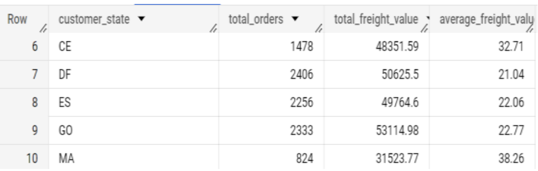
2. Calculate the Total & Average value of order freight for each state.

Query-

```
SELECT c.customer_state,
count(o.order_id) as total_orders,
round(sum(oi.freight_value),2) as total_freight_value,
round(avg(oi.freight_value),2) as average_freight_value
from `target.customers` as c
join `target.orders` as o
on o.customer_id=c.customer_id
 join `target.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_orders	total_freight_value	average_freight_valu
1	AC	92	3686.75	40.07
2	AL	444	15914.59	35.84
3	AM	165	5478.89	33.21
4	AP	82	2788.5	34.01
5	BA	3799	100156.68	26.36



Row	customer_state	total_orders	total_freight_value	average_freight_valu
6	CE	1478	48351.59	32.71
7	DF	2406	50625.5	21.04
8	ES	2256	49764.6	22.06
9	GO	2333	53114.98	22.77
10	MA	824	31523.77	38.26

Insights-

a) Highest average_freight_value was 42.98 for the state-RR.
b) Highest total_freight_value was 718723.07for the state-SP.

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

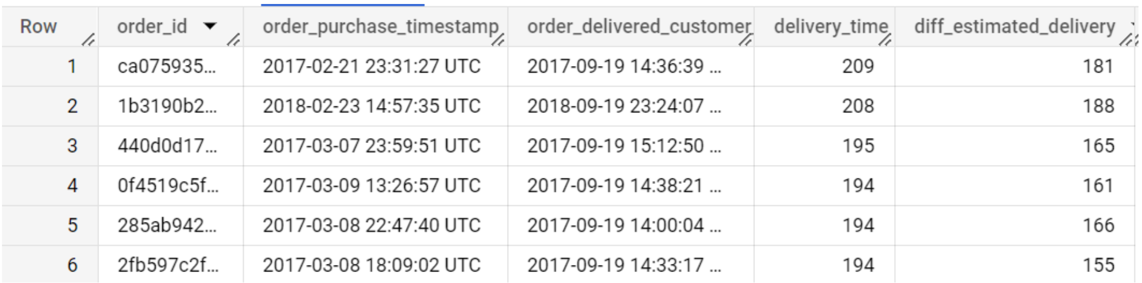
1.Find the no. of days taken to deliver each order from the order's purchase date as delivery time.

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:

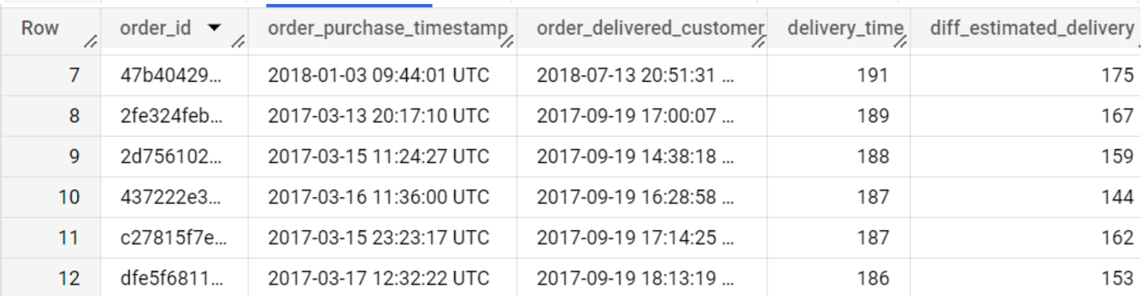
- time_to_deliver = order_delivered_customer_date - order_purchase_timestamp
- diff_estimated_delivery = order_estimated_delivery_date - order_delivered_customer_date

Query-

```
SELECT order_id,
order_purchase_timestamp,
order_delivered_customer_date,
abs(DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, day)) as
delivery_time,
abs(DATE_DIFF(order_estimated_delivery_date, order_delivered_customer_date, day)) as
diff_estimated_delivery
from `target.orders`
where order_status = 'delivered'
order by delivery_time desc
```



Row	order_id	order_purchase_timestamp	order_delivered_customer	delivery_time	diff_estimated_delivery
1	ca075935...	2017-02-21 23:31:27 UTC	2017-09-19 14:36:39 ...	209	181
2	1b3190b2...	2018-02-23 14:57:35 UTC	2018-09-19 23:24:07 ...	208	188
3	440d0d17...	2017-03-07 23:59:51 UTC	2017-09-19 15:12:50 ...	195	165
4	0f4519c5f...	2017-03-09 13:26:57 UTC	2017-09-19 14:38:21 ...	194	161
5	285ab942...	2017-03-08 22:47:40 UTC	2017-09-19 14:00:04 ...	194	166
6	2fb597c2f...	2017-03-08 18:09:02 UTC	2017-09-19 14:33:17 ...	194	155



Row	order_id	order_purchase_timestamp	order_delivered_customer	delivery_time	diff_estimated_delivery
7	47b40429...	2018-01-03 09:44:01 UTC	2018-07-13 20:51:31 ...	191	175
8	2fe324feb...	2017-03-13 20:17:10 UTC	2017-09-19 17:00:07 ...	189	167
9	2d756102...	2017-03-15 11:24:27 UTC	2017-09-19 14:38:18 ...	188	159
10	437222e3...	2017-03-16 11:36:00 UTC	2017-09-19 16:28:58 ...	187	144
11	c27815f7e...	2017-03-15 23:23:17 UTC	2017-09-19 17:14:25 ...	187	162
12	dfe5f6811...	2017-03-17 12:32:22 UTC	2017-09-19 18:13:19 ...	186	153

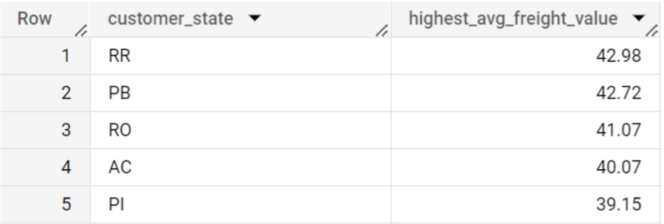
Insights-

a) The maximum delivery time comes out to be 209.

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

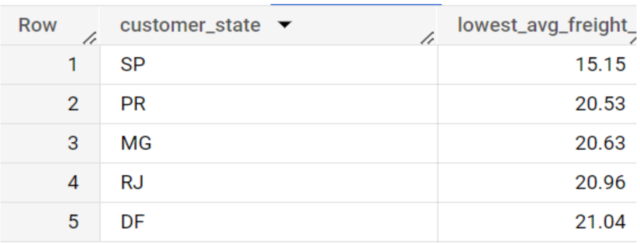
Query-

```
SELECT c.customer_state,
round(avg(oi.freight_value),2) as highest_avg_freight_value,
from `target.customers` as c
join `target.orders` as o
on c.customer_id = o.customer_id
join `target.order_items` as oi
on o.order_id = oi.order_id
group by c.customer_state
order by highest_avg_freight_value desc
limit 5;
```



Row	customer_state	highest_avg_freight_value
1	RR	42.98
2	PB	42.72
3	RO	41.07
4	AC	40.07
5	PI	39.15

```
SELECT c.customer_state,
round(avg(oi.freight_value),2) as lowest_avg_freight_value
from `target.customers` as c
join `target.orders` as o
on c.customer_id = o.customer_id
join `target.order_items` as oi
on o.order_id = oi.order_id
group by c.customer_state
order by lowest_avg_freight_value Asc
limit 5;
```



Row	customer_state	lowest_avg_freight_
1	SP	15.15
2	PR	20.53
3	MG	20.63
4	RJ	20.96
5	DF	21.04

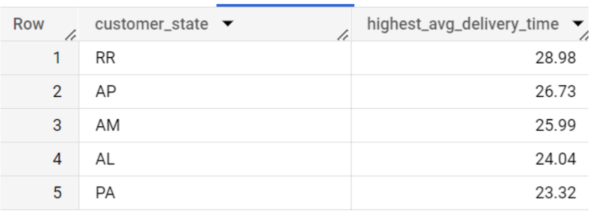
Insights-

a) The highest and lowest avg_freight_value comes out to be 42.98 and 15.15 respectively.

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

Query-

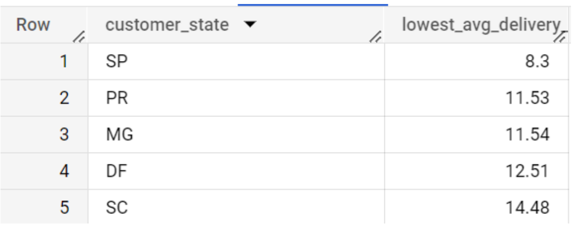
```
SELECT c.customer_state,
round(avg(DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, day)),2)
as highest_avg_delivery_time
from `target.customers` as c
join `target.orders` as o
on c.customer_id = o.customer_id
group by c.customer_state
order by highest_avg_delivery_time desc
Limit 5
```



Row	customer_state	highest_avg_delivery_time
1	RR	28.98
2	AP	26.73
3	AM	25.99
4	AL	24.04
5	PA	23.32

Query-

```
SELECT c.customer_state,
round(avg(DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, day)),2)
as lowest_avg_delivery_time
from `target.customers` as c
join `target.orders` as o
on c.customer_id = o.customer_id
group by c.customer_state
order by lowest_avg_delivery_time Asc
Limit 5
```



Row	customer_state	lowest_avg_delivery_
1	SP	8.3
2	PR	11.53
3	MG	11.54
4	DF	12.51
5	SC	14.48

Insights-

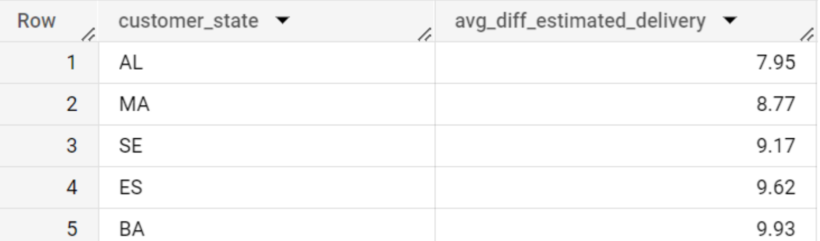
a) The highest and lowest avg_delivery time comes out to be 8.3 and 28.98 respectively.

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-

```
SELECT c.customer_state,
round(avg(DATE_DIFF(o.order_estimated_delivery_date, o.order_delivered_customer_date,
day)),2) as avg_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 avg_diff_estimated_delivery
limit 5
```



Row	customer_state	avg_diff_estimated_delivery
1	AL	7.95
2	MA	8.77
3	SE	9.17
4	ES	9.62
5	BA	9.93

Insights-

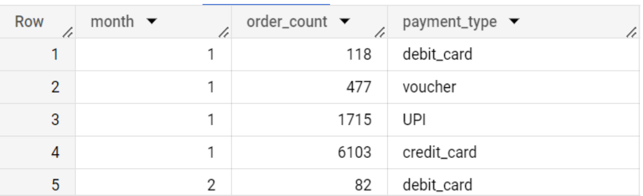
a) The states where order delivery time is fast are AL, MA, SE, ES, BA respectively.

6. Analysis based on the payments:

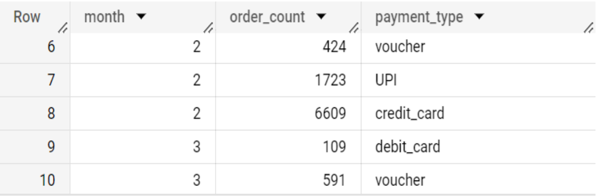
1. Find the month on month no. of orders placed using different payment types.

Query-

```
SELECT Extract(Month from o.order_purchase_timestamp) as month,
count(o.order_id) as order_count,
p.payment_type
from `target.orders` as o
Join `target.payments` as p
on o.order_id = p.order_id
group by month, p.payment_type
order by month, order_count asc
```



Row	month	order_count	payment_type
1	1	118	debit_card
2	1	477	voucher
3	1	1715	UPI
4	1	6103	credit_card
5	2	82	debit_card



Row	month	order_count	payment_type
6	2	424	voucher
7	2	1723	UPI
8	2	6609	credit_card
9	3	109	debit_card
10	3	591	voucher

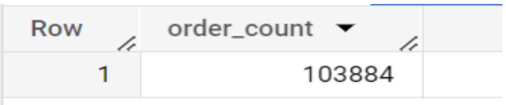
Insights-

a) On monthly basis, the customers used payment type mode in following order: Credit card>UPI>voucher>debit card.

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

Query1-

```
SELECT count(order_id) as order_count,
from `target.payments`
where payment_installments >= 1
```



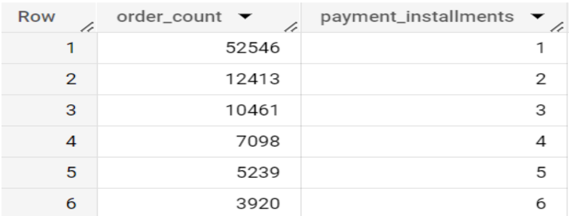
Row	order_count
1	103884

Insights-

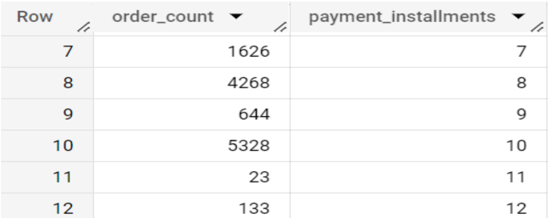
a) Total number of counts comes out to be 103884 when at least one installment has been paid.

Query2-

```
SELECT count(order_id) as order_count,
payment_installments
from `target.payments`
group by payment_installments
having payment_installments >= 1
```



Row	order_count	payment_installments
1	52546	1
2	12413	2
3	10461	3
4	7098	4
5	5239	5
6	3920	6



Row	order_count	payment_installments
7	1626	7
8	4268	8
9	644	9
10	5328	10
11	23	11
12	133	12

Insights-

a) Total number of orders placed were highest when one installment per order has been paid.