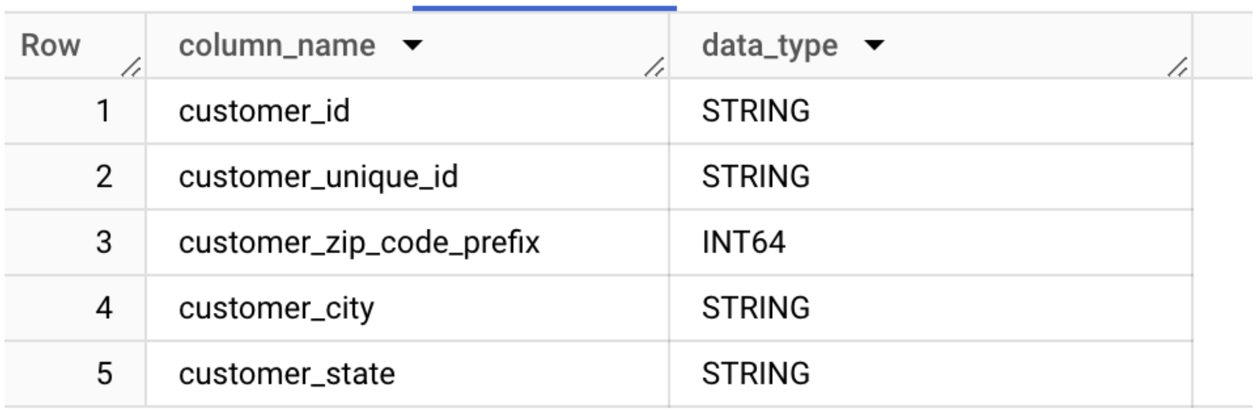
Business Case: Target SQL

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

A. Data type of all columns in the “customers” table.

Sol:

```
select column_name,data_type from
`case-study-419114.target.INFORMATION_SCHEMA.COLUMNS`
where table_name='customers';
```

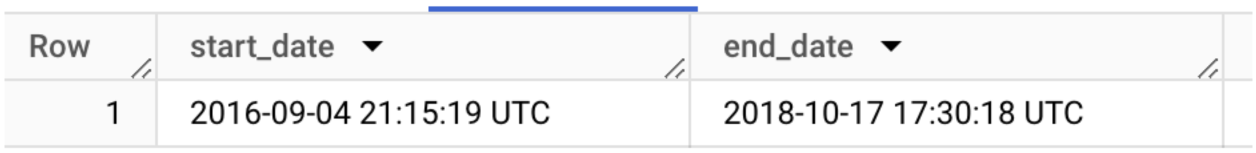


Row	column_name	data_type
1	customer_id	STRING
2	customer_unique_id	STRING
3	customer_zip_code_prefix	INT64
4	customer_city	STRING
5	customer_state	STRING

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

Sol:

```
select
min(order_purchase_timestamp) as start_date,
max(order_purchase_timestamp) as end_date
from `target.orders`;
```



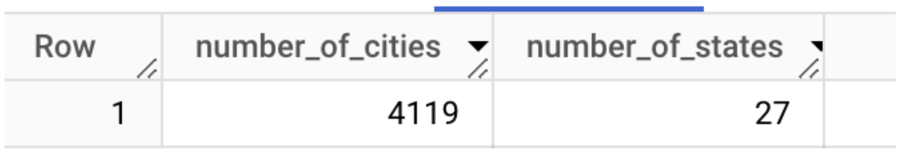
Row	start_date	end_date
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC

Insight: According to the given dataset, the start date from which the orders were placed was 4th Sept 2016 and the end date was 17th Oct 2018.

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

Sol:

```
select
count(distinct customer_city) as number_of_cities,
count(distinct customer_state) as number_of_states
from `target.customers`
;
```



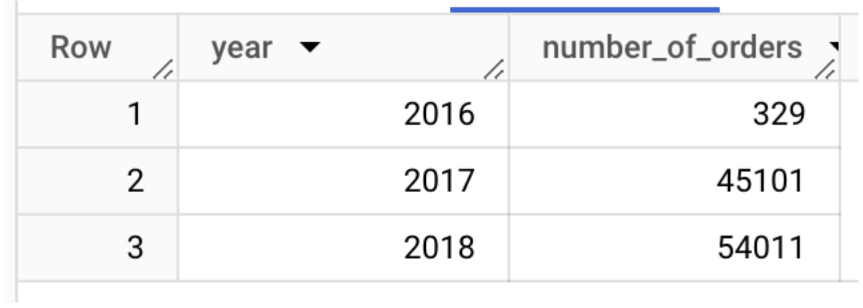
Row	number_of_cities	number_of_states
1	4119	27

II. In-depth Exploration:

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

Sol:

```
select
year,
count(order_id) as number_of_orders
from
(
select
order_id,
extract(month from order_purchase_timestamp) as month,
extract(year from order_purchase_timestamp) as year,
from `target.orders`
) t
group by 1
order by 1
;
```



Row	year	number_of_orders
1	2016	329
2	2017	45101
3	2018	54011

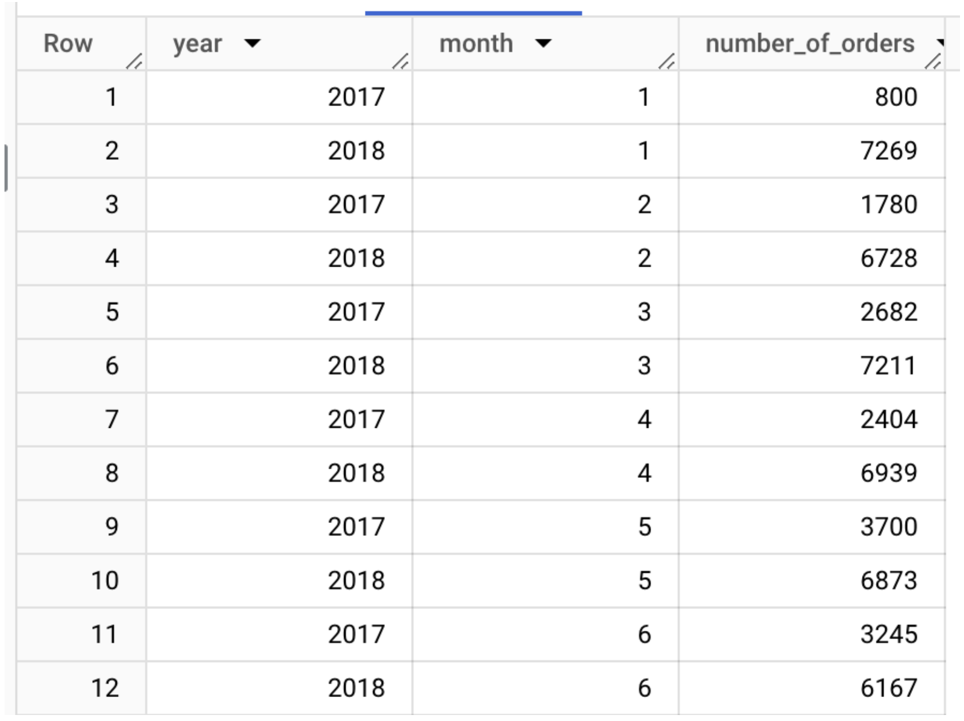
Insights: There was a drastic increase in the number of orders placed from 2016 to 2017 and slight increase from 2017 to 2018.

Recommendations: Advertising and running deals may lead to more increase in sales.

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

Sol:

```
select
year,
month,
count(order_id) as number_of_orders
from
(
select
order_id,
extract(month from order_purchase_timestamp) as month,
extract(year from order_purchase_timestamp) as year,
from `target.orders`
) t
group by 1,2
order by 1,2
;
```



Row	year	month	number_of_orders
1	2017	1	800
2	2018	1	7269
3	2017	2	1780
4	2018	2	6728
5	2017	3	2682
6	2018	3	7211
7	2017	4	2404
8	2018	4	6939
9	2017	5	3700
10	2018	5	6873
11	2017	6	3245
12	2018	6	6167

Insight:

- In 2016, there are barely any orders placed.
- In 2017, from January number of orders placed started to increase and reached peak in November probably because of festive season.
- In 2018, number of orders placed was linear but in September and November there is a huge drop in the sales.

Recommendations:

- Maintain stock inventory.
- Hire more people during the festive season.
- Combine low selling products with high selling products for stock clearance during festive season.
- Give more offers in the non-festive season.

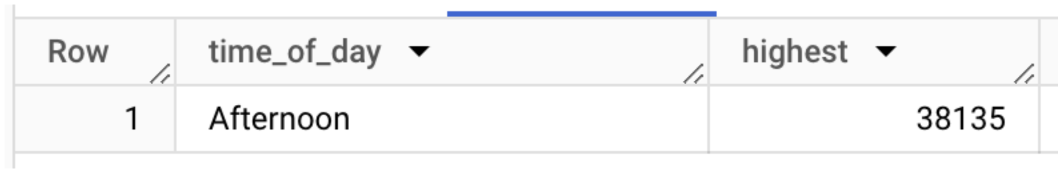
C. During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)

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

Sol:

```
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 '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,
count(*) as highest
from `target.orders`
group by time_of_day
order by highest desc
limit 1
```

;



Row	time_of_day	highest
1	Afternoon	38135

Insights: Brazilian customers mostly place their orders in the afternoon.

Recommendations:

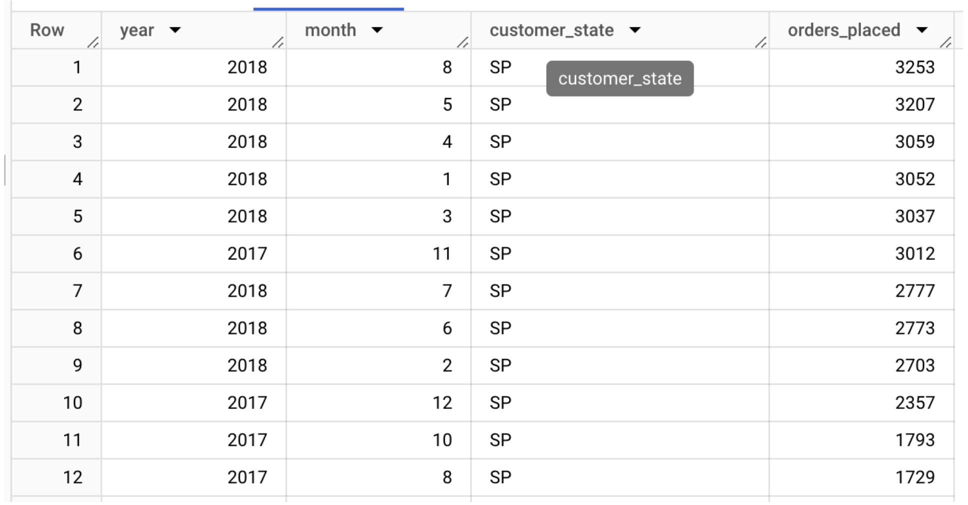
- Maintain website traffic well as there can be a huge load on the server.
- Have the warehouse staff ready to pack the orders from the afternoon for shipping the products.

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

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

Sol:

```
select
extract(year from order_purchase_timestamp) as year,
extract(month from order_purchase_timestamp) as month,
customer_state,
count(order_id) as orders_placed,
from `target.orders` o left join `target.customers` c on o.customer_id=c.customer_id
group by year,month,customer_state
order by 4 desc
;
```



Row	year	month	customer_state	orders_placed
1	2018	8	SP	3253
2	2018	5	SP	3207
3	2018	4	SP	3059
4	2018	1	SP	3052
5	2018	3	SP	3037
6	2017	11	SP	3012
7	2018	7	SP	2777
8	2018	6	SP	2773
9	2018	2	SP	2703
10	2017	12	SP	2357
11	2017	10	SP	1793
12	2017	8	SP	1729

customer_state

Insights :

Highest orders were placed in the state SP

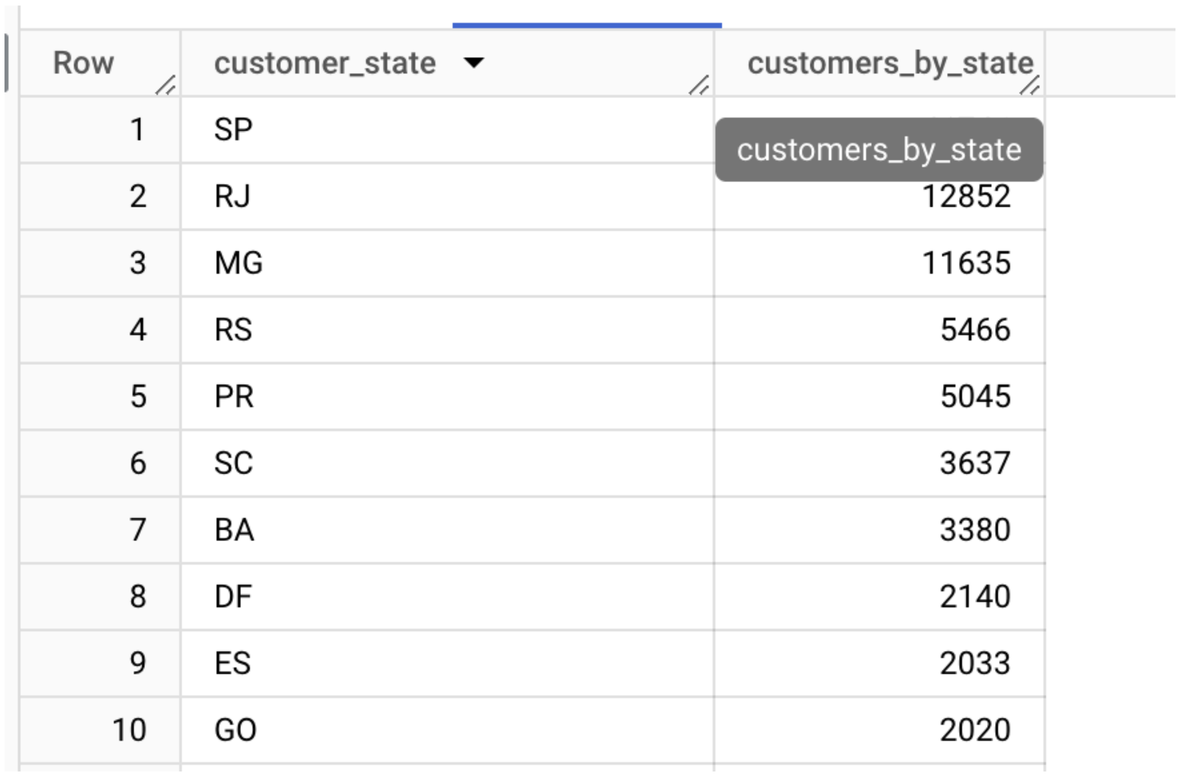
Recommendations:

Maintain large stock inventory and good number of staff near/in the state SP.

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

Sol:

```
select
customer_state,
count(customer_id) as customers_by_state
from `target.customers`
group by customer_state
order by 2 desc
;
```



Row	customer_state	customers_by_state
1	SP	[illegible]
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

Insights:

- State SP has highest number of customers, next are RJ and MG with more number of customers.
- AC, AP, RR being the states with lowest customer count.

Recommendation: Can setup warehouse in SP, RJ, MG for faster delivery.

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

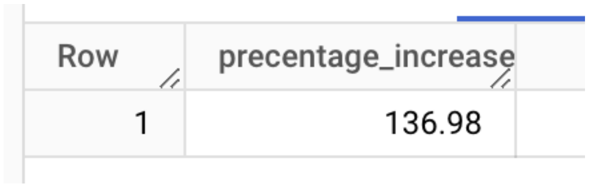
A. Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only).

Sol:

```
with cte as
(
select
sum(payment_value) as cost_of_orders,
extract(year from order_purchase_timestamp) as year_wise
from `target.payments` p left join `target.orders` o using(order_id)
where extract(year from order_purchase_timestamp) between 2017 and 2018 and
extract(month from order_purchase_timestamp) between 01 and 08
group by extract(year from order_purchase_timestamp)
),

other_cte as
(
select
cte.cost_of_orders as _2018,
lag(cte.cost_of_orders,1) over(order by cte.cost_of_orders) as _2017
from cte
order by cte.cost_of_orders desc
limit 1
)

select
round(((_2018-_2017)/ _2017 )*100,2) as precentage_increase
from other_cte
;
```



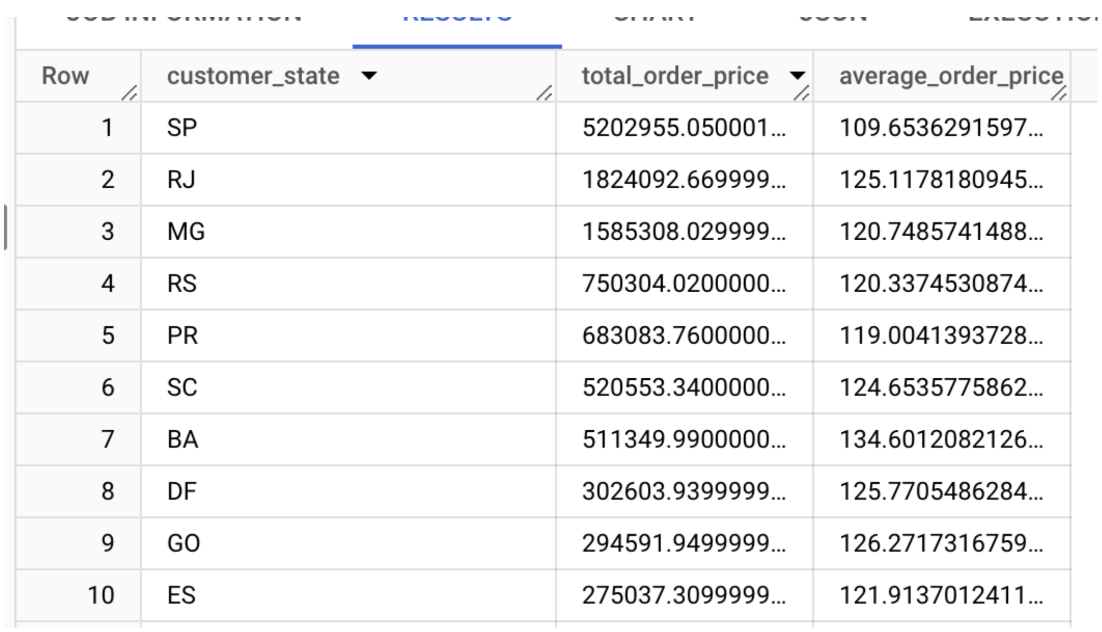
Row	precentage_increase
1	136.98

Insight: Percentage increase is 136.98.

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

Sol:

```
select
c.customer_state,
sum(oi.price) as total_order_price,
avg(oi.price) as average_order_price,
from `target.orders` o left join `target.order_items` oi using(order_id)
left join `target.customers` c using(customer_id)
group by c.customer_state
order by 2 desc
;
```

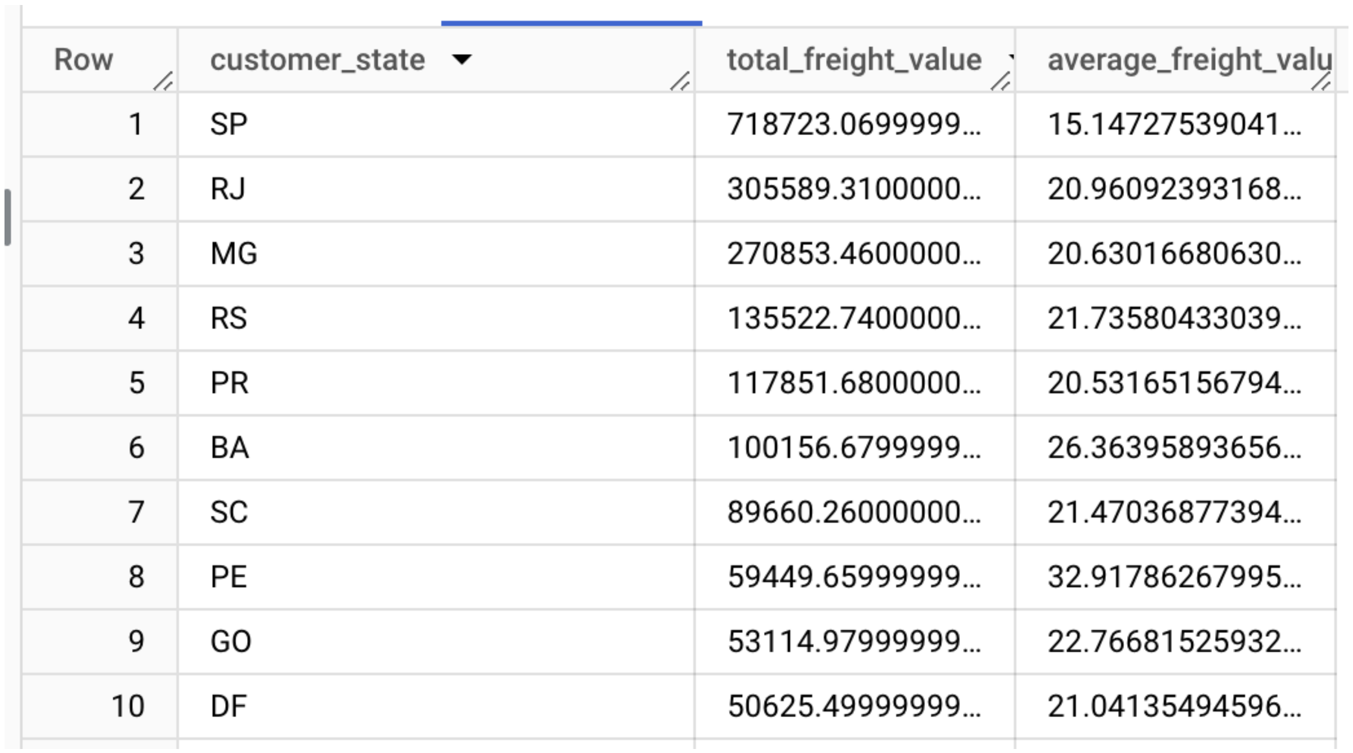


Row	customer_state	total_order_price	average_order_price
1	SP	5202955.050001...	109.6536291597...
2	RJ	1824092.669999...	125.1178180945...
3	MG	1585308.029999...	120.7485741488...
4	RS	750304.0200000...	120.3374530874...
5	PR	683083.7600000...	119.0041393728...
6	SC	520553.3400000...	124.6535775862...
7	BA	511349.9900000...	134.6012082126...
8	DF	302603.9399999...	125.7705486284...
9	GO	294591.9499999...	126.2717316759...
10	ES	275037.3099999...	121.9137012411...

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

Sol:

```
select
c.customer_state,
sum(oi.freight_value) as total_freight_value,
avg(oi.freight_value) as average_freight_value,
from `target.orders` o left join `target.order_items` oi using(order_id)
left join `target.customers` c using(customer_id)
group by c.customer_state
order by 2 desc
;
```



Row	customer_state	total_freight_value	average_freight_valu
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...

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

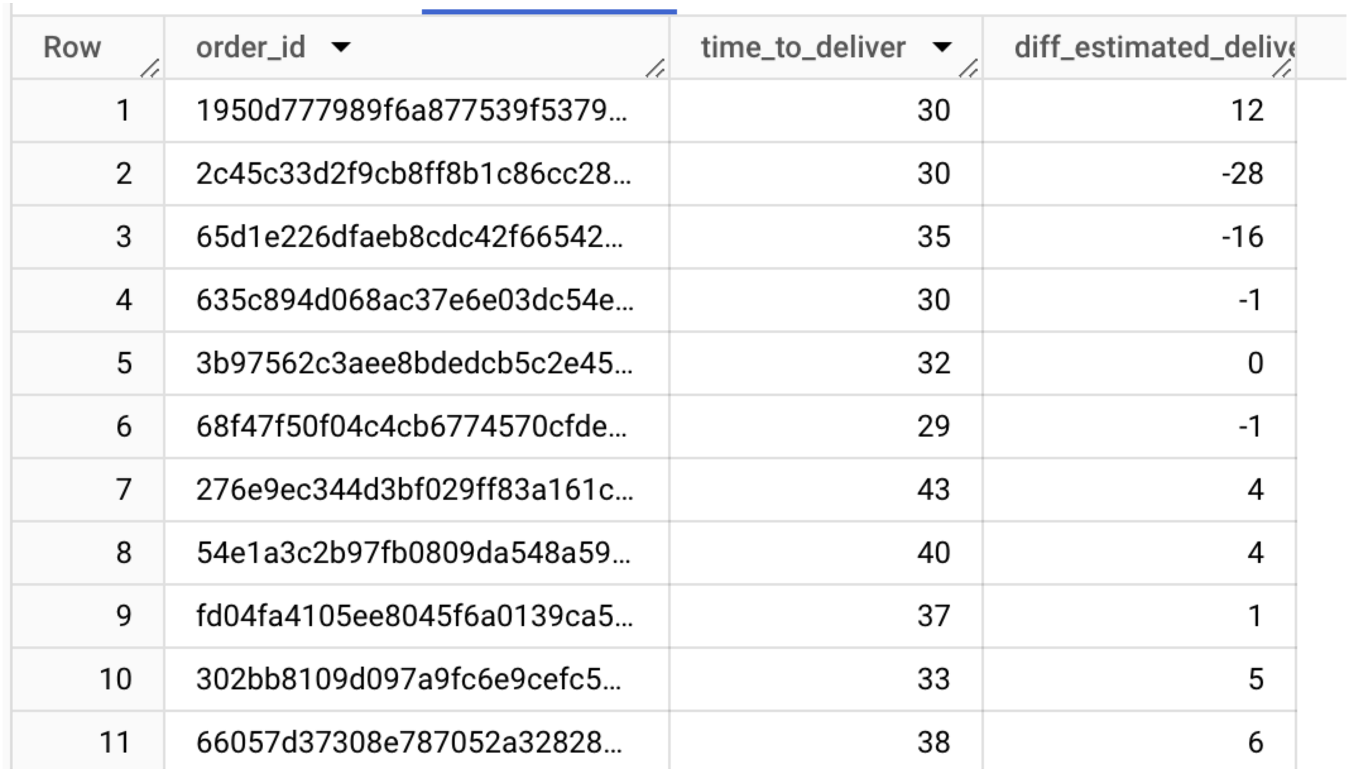
A. Find the no. of days taken to deliver each order from the order's purchase date as delivery time.

Also, calculate the difference (in days) between the estimated & actual delivery date of an order.

Do this in a single query.

Sol:

```
select
order_id,
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`
;
```



Row	order_id	time_to_deliver	diff_estimated_delive
1	1950d777989f6a877539f5379...	30	12
2	2c45c33d2f9cb8ff8b1c86cc28...	30	-28
3	65d1e226dfaeb8cdc42f66542...	35	-16
4	635c894d068ac37e6e03dc54e...	30	-1
5	3b97562c3aee8bdedcb5c2e45...	32	0
6	68f47f50f04c4cb6774570cfde...	29	-1
7	276e9ec344d3bf029ff83a161c...	43	4
8	54e1a3c2b97fb0809da548a59...	40	4
9	fd04fa4105ee8045f6a0139ca5...	37	1
10	302bb8109d097a9fc6e9cefc5...	33	5
11	66057d37308e787052a32828...	38	6

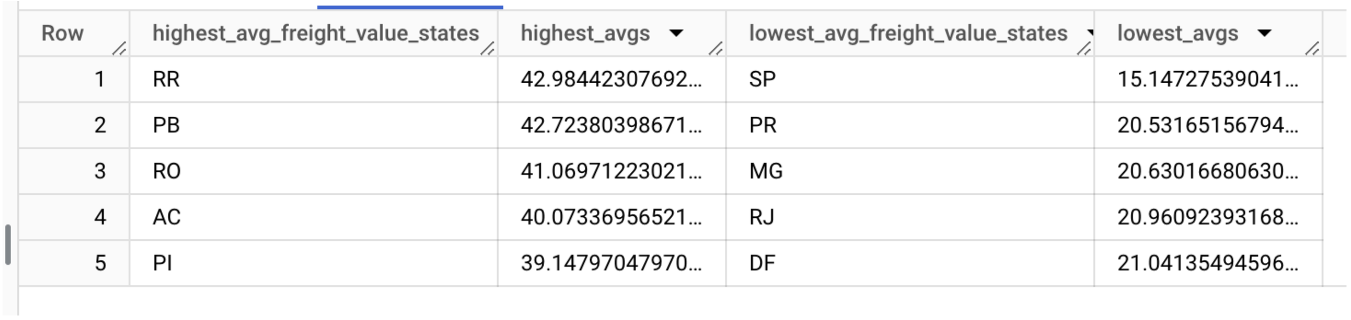
B. Find out the top 5 states with the highest & lowest average freight value.

Sol:

```
with cte as
(
select
c.customer_state,
avg(oi.freight_value) as average_freight_value,
from `target.order_items` oi left join `target.orders` o using(order_id)
left join `target.customers` c using(customer_id)
group by c.customer_state
)


select
```

```
t1.customer_state as highest_avg_freight_value_states,
t1.highest_avgs,
t2.customer_state as lowest_avg_freight_value_states,
t2.lowest_avgs
from
(
select
customer_state,average_freight_value as highest_avgs,
row_number() over(order by average_freight_value desc) as number
from cte
limit 5
) t1
join
(
select
customer_state,average_freight_value as lowest_avgs,
row_number() over(order by average_freight_value asc) as number
from cte
limit 5
)
t2
using(number)
;
```



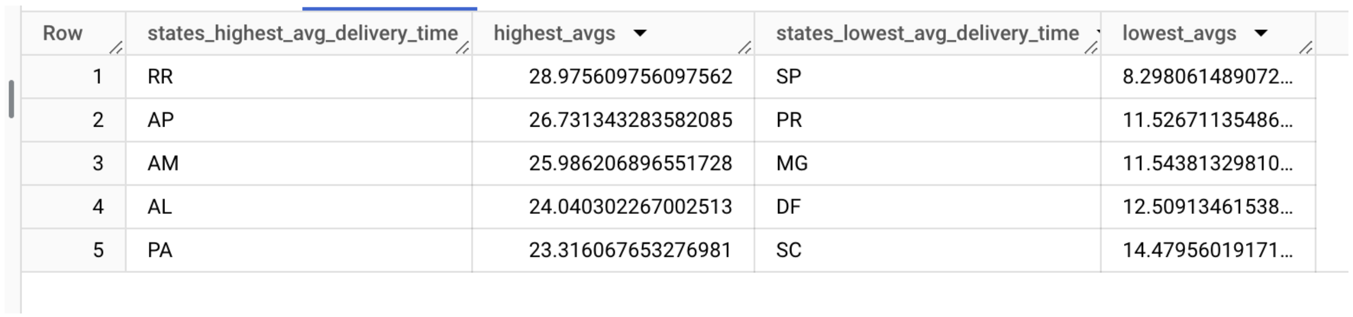
Row	highest_avg_freight_value_states	highest_avgs	lowest_avg_freight_value_states	lowest_avgs
1	RR	42.98442307692...	SP	15.14727539041...
2	PB	42.72380398671...	PR	20.53165156794...
3	RO	41.06971223021...	MG	20.63016680630...
4	AC	40.07336956521...	RJ	20.96092393168...
5	PI	39.14797047970...	DF	21.04135494596...

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

Sol:

```
with cte as
(
select
c.customer_state,
avg(timestamp_diff( order_delivered_customer_date, order_purchase_timestamp, DAY )) as
average_delivery_time,
from `target.orders` o left join `target.customers` c using(customer_id)
group by c.customer_state
)
```

```
select
t1.customer_state as states_highest_avg_delivery_time,
t1.highest_avgs,
t2.customer_state as states_lowest_avg_delivery_time,
t2.lowest_avgs
from
(
select
customer_state,average_delivery_time as highest_avgs,
row_number() over(order by average_delivery_time desc) as number
from cte
limit 5
) t1
join
(
select
customer_state,average_delivery_time as lowest_avgs,
row_number() over(order by average_delivery_time asc) as number
from cte
limit 5
)
t2
using(number)
;
```



Row	states_highest_avg_delivery_time	highest_avgs	states_lowest_avg_delivery_time	lowest_avgs
1	RR	28.975609756097562	SP	8.298061489072...
2	AP	26.731343283582085	PR	11.52671135486...
3	AM	25.986206896551728	MG	11.54381329810...
4	AL	24.040302267002513	DF	12.50913461538...
5	PA	23.316067653276981	SC	14.47956019171...

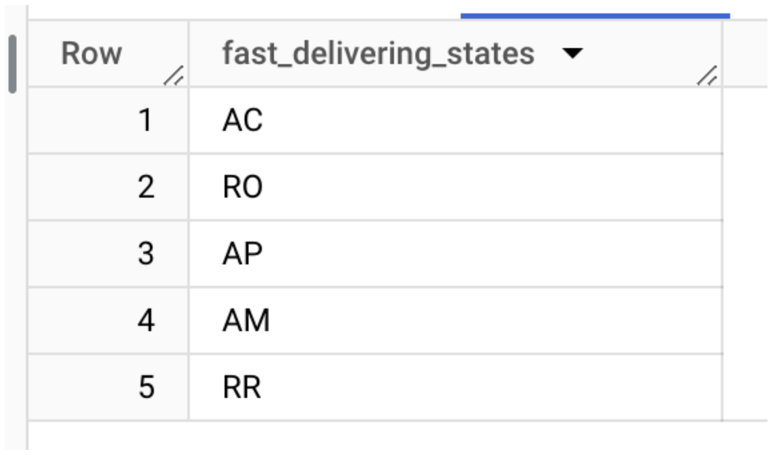
D. Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery.

Sol:

```
with cte as
(
select
c.customer_state,
avg(timestamp_diff( order_delivered_customer_date, order_estimated_delivery_date, DAY )) as
avg_estimated_delivery
from `target.orders` o left join `target.customers` c using(customer_id)
```

```sql
group by c.customer_state
order by 2
)

select
customer_state as fast_delivering_states
from cte
limit 5;
```



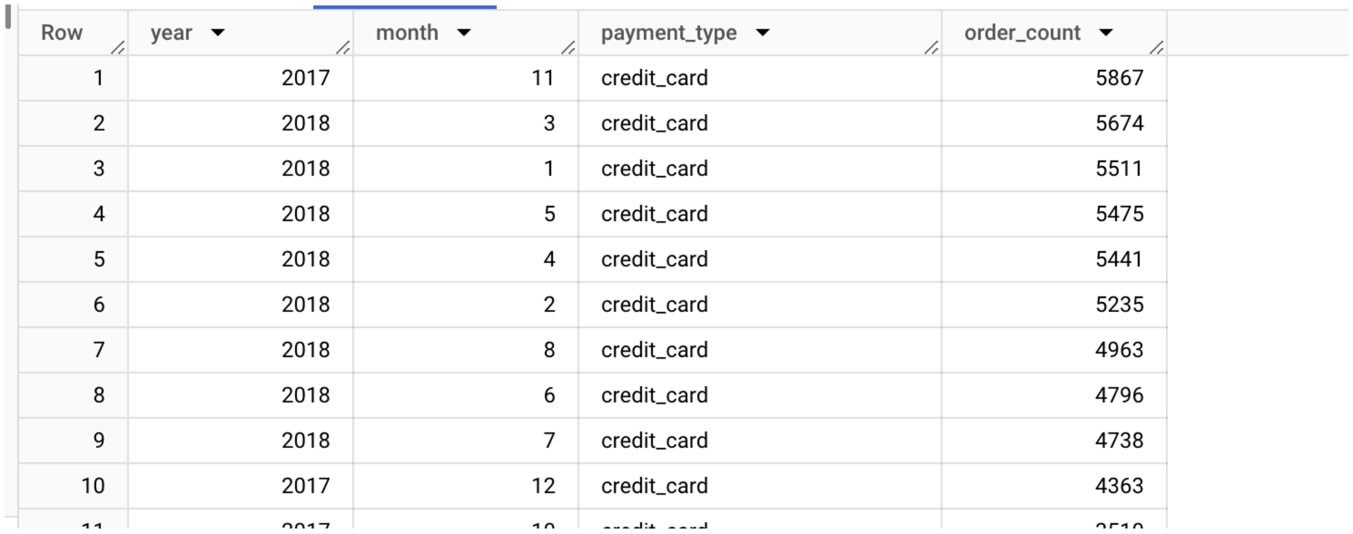
Row	fast_delivering_states
1	AC
2	RO
3	AP
4	AM
5	RR

VI. Analysis based on the payments:

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

Sol:

```sql
select
extract(year from order_purchase_timestamp) as year,
extract(month from order_purchase_timestamp) as month,
payment_type,
count(distinct order_id) as order_count
from `target.orders` o left join `target.payments` p using(order_id)
where payment_type is not null
group by 1,2,3
order by 4 desc
;
```

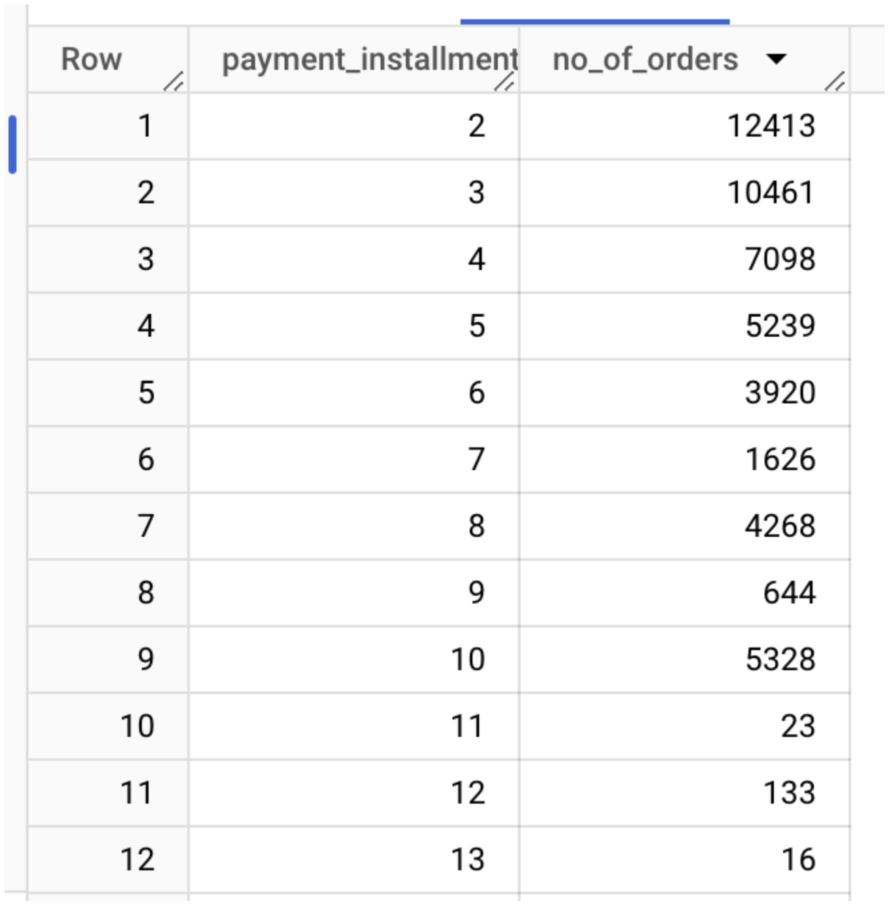


Row	year	month	payment_type	order_count
1	2017	11	credit_card	5867
2	2018	3	credit_card	5674
3	2018	1	credit_card	5511
4	2018	5	credit_card	5475
5	2018	4	credit_card	5441
6	2018	2	credit_card	5235
7	2018	8	credit_card	4963
8	2018	6	credit_card	4796
9	2018	7	credit_card	4738
10	2017	12	credit_card	4363
11	2017	10	credit_card	[illegible]

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

Sol:

```
select
payment_installments,
count(*) as no_of_orders
from `target.payments`
where payment_installments>=1
group by payment_installments ;
```



Row	payment_installment	no_of_orders
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
11	12	133
12	13	16