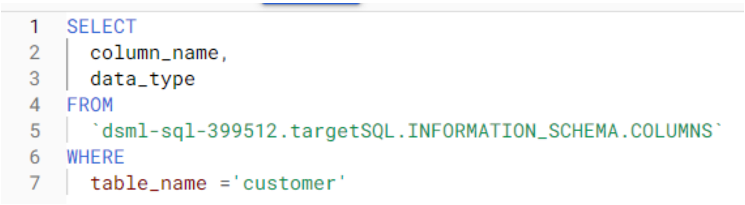
Target SQL

Business case 1

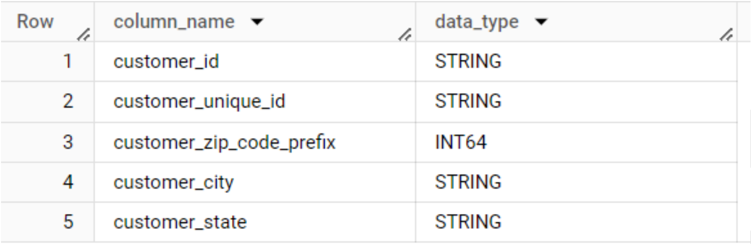
Rishabh Singh
DSML SEPTEMBER 2023 Beginner

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.

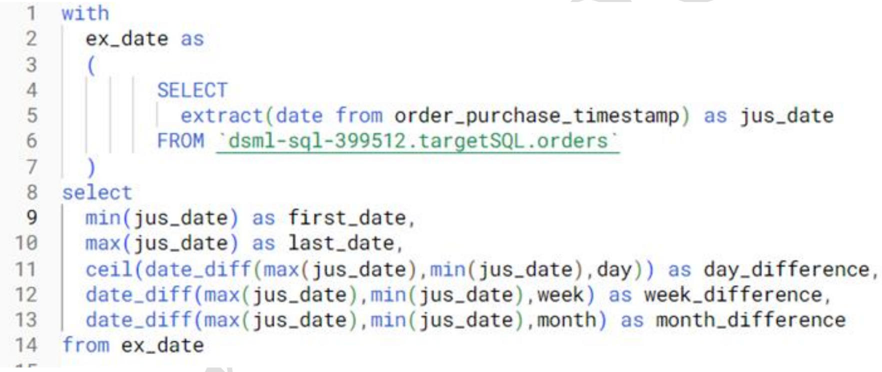


```
SELECT
  column_name,
  data_type
FROM
  `dsml-sql-399512.targetSQL.INFORMATION_SCHEMA.COLUMNS`
WHERE
  table_name ='customer'
```

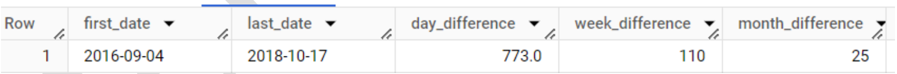


Row	column_name	data_type
1	customer_id	STRING
2	customer_unique_id	STRING
3	customer_zip_code_prefix	INT64
4	customer_city	STRING
5	customer_state	STRING

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

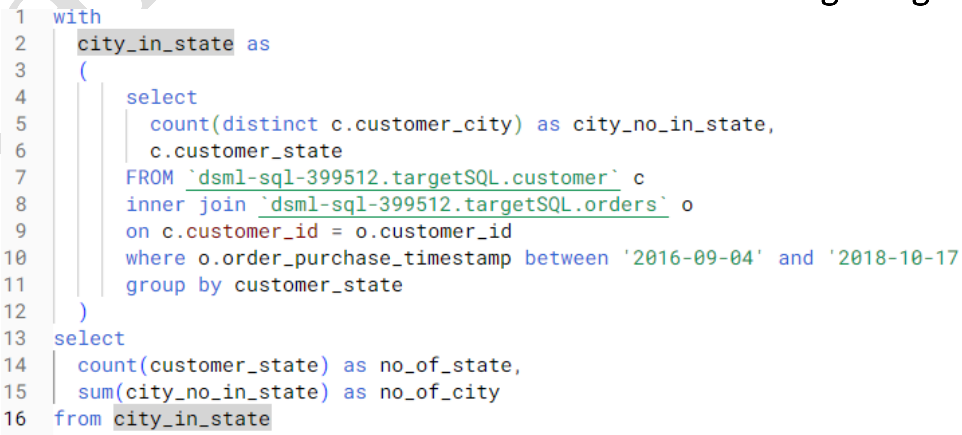


```
with
  ex_date as
  (
      SELECT
        extract(date from order_purchase_timestamp) as jus_date
      FROM `dsml-sql-399512.targetSQL.orders`
  )
select
  min(jus_date) as first_date,
  max(jus_date) as last_date,
  ceil(date_diff(max(jus_date),min(jus_date),day)) as day_difference,
  date_diff(max(jus_date),min(jus_date),week) as week_difference,
  date_diff(max(jus_date),min(jus_date),month) as month_difference
from ex_date
```

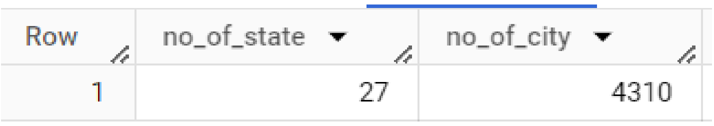


Row	first_date	last_date	day_difference	week_difference	month_difference
1	2016-09-04	2018-10-17	773.0	110	25

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



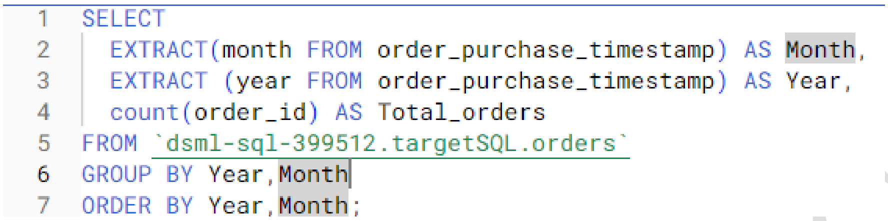
```
with
  city_in_state as
  (
      select
        count(distinct c.customer_city) as city_no_in_state,
        c.customer_state
      FROM `dsml-sql-399512.targetSQL.customer` c
      inner join `dsml-sql-399512.targetSQL.orders` o
      on c.customer_id = o.customer_id
      where o.order_purchase_timestamp between '2016-09-04' and '2018-10-17
      group by customer_state
  )
select
  count(customer_state) as no_of_state,
  sum(city_no_in_state) as no_of_city
from city_in_state
```



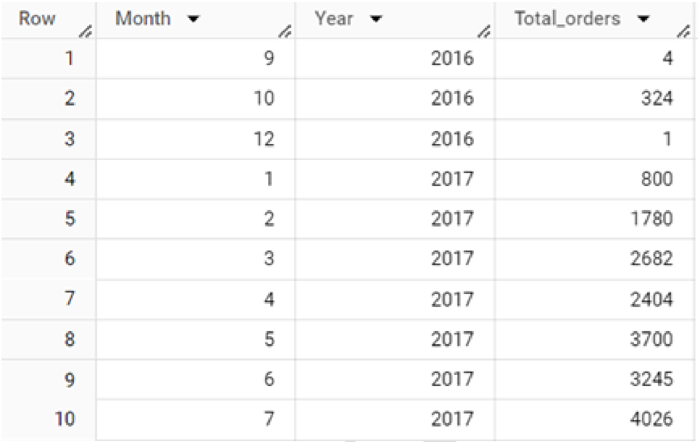
Row	no_of_state	no_of_city
1	27	4310

In-depth Exploration:

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



```sql
SELECT
  EXTRACT(month FROM order_purchase_timestamp) AS Month,
  EXTRACT (year FROM order_purchase_timestamp) AS Year,
  count(order_id) AS Total_orders
FROM `dsml-sql-399512.targetSQL.orders`
GROUP BY Year,Month
ORDER BY Year,Month;
```

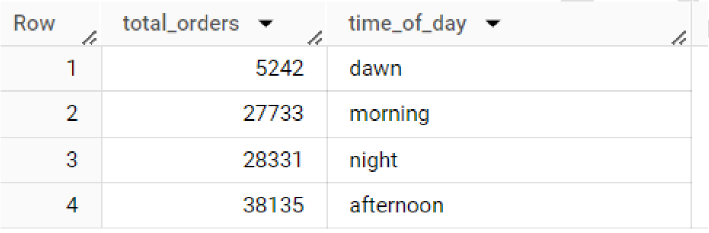


Row	Month	Year	Total_orders
1	9	2016	4
2	10	2016	324
3	12	2016	1
4	1	2017	800
5	2	2017	1780
6	3	2017	2682
7	4	2017	2404
8	5	2017	3700
9	6	2017	3245
10	7	2017	4026

2. Can we see some kind of monthly seasonality in terms of the no. of orders being placed?
    - We observed a steady growth in order purchases starting from the first month of 2017, reaching its peak in the 11th month of that year.
    - However, it's important to note that for certain months, the order data significantly deviates from the average order value, which raises concerns about potential inaccuracies or data corruption.
    - Notably, there is a decline in ordered values between the 5th and 6th months in both 2017 and 2018, suggesting a consistent pattern of reduced order activity during that period.

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

```
SELECT
  count(order_id) as total_orders,
  time_of_day
  from(
    select
      *,
      case
          when time(order_purchase_timestamp) between '00:00:00' and '06:59:59' then
'dawn'
          when time(order_purchase_timestamp) between '07:00:00' and '12:59:59' then
'morning'
          when time(order_purchase_timestamp) between '13:00:00' and '18:59:59' then
'afternoon'
          else 'night'
    end as time_of_day
    FROM `dsml-sql-399512.targetSQL.orders`
  ) as x
group by time_of_day
order by total_orders;
```

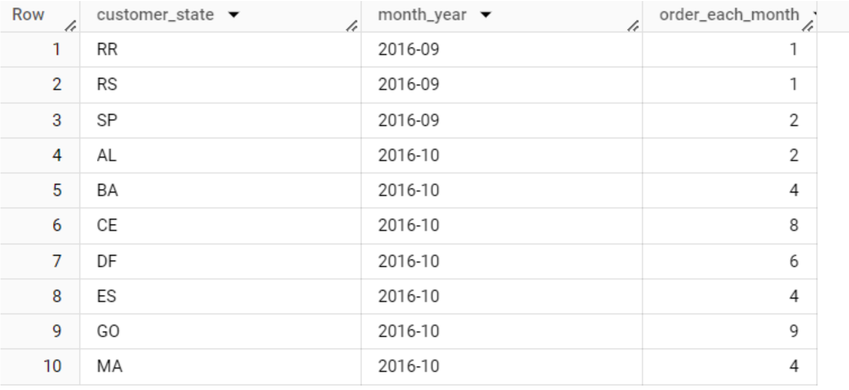


Row	total_orders	time_of_day
1	5242	dawn
2	27733	morning
3	28331	night
4	38135	afternoon

Evolution of E-commerce orders in the Brazil region:

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

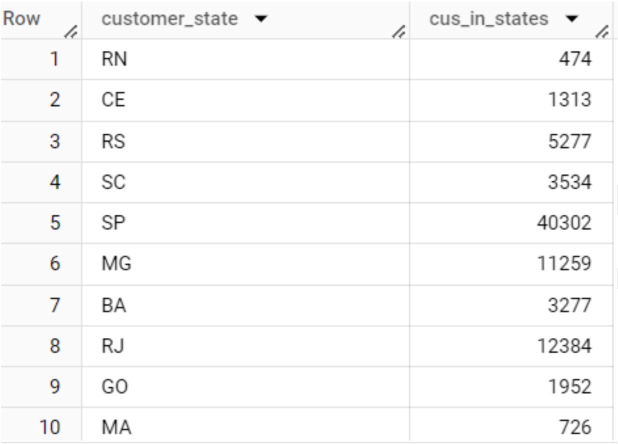
```
with
table1 as
(
   select
      c.customer_id,
      c.customer_state,
      o.order_id,
      format_date('%Y-%m', order_purchase_timestamp) as month_year
    from `targetSQL.customer` c
    inner join `targetSQL.orders` o
    on c.customer_id = o.customer_id
)
select
  customer_state,
  month_year,
  count(*) as order_each_month
from table1
group by month_year,customer_state
order by month_year, customer_state;
```



Row	customer_state	month_year	order_each_month
1	RR	2016-09	1
2	RS	2016-09	1
3	SP	2016-09	2
4	AL	2016-10	2
5	BA	2016-10	4
6	CE	2016-10	8
7	DF	2016-10	6
8	ES	2016-10	4
9	GO	2016-10	9
10	MA	2016-10	4

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

```
select
   customer_state,
   count(distinct customer_unique_id) as cus_in_states
from `targetSQL.customer`
group by customer_state
```

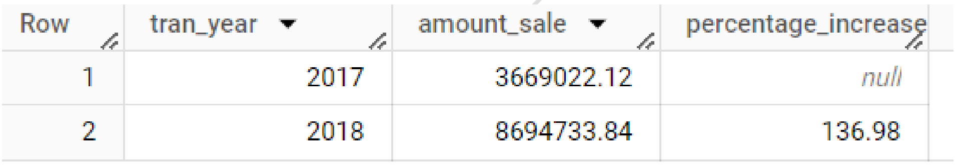


Row	customer_state	cus_in_states
1	RN	474
2	CE	1313
3	RS	5277
4	SC	3534
5	SP	40302
6	MG	11259
7	BA	3277
8	RJ	12384
9	GO	1952
10	MA	726

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.

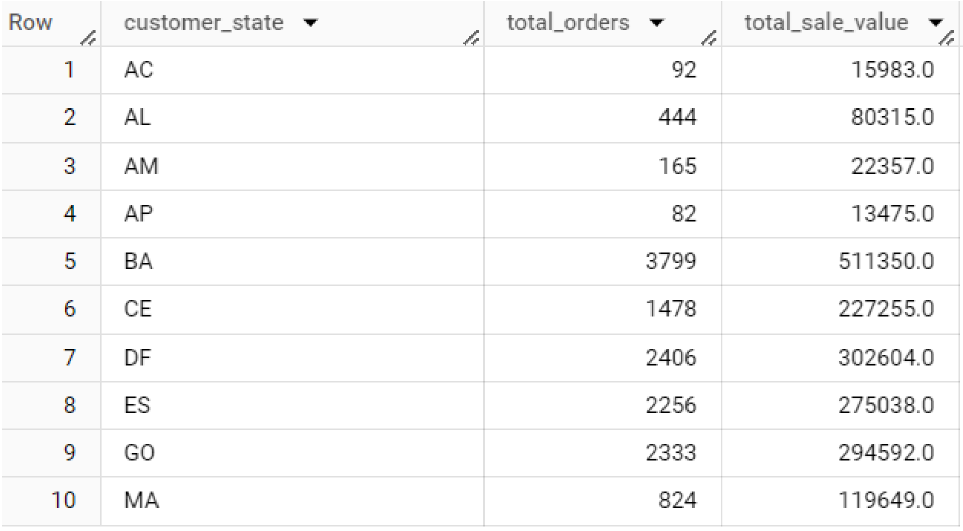
```
with
 table1 as
  (
   select
    extract(month from o.order_purchase_timestamp) as tran_month,
    extract(year from o.order_purchase_timestamp) as tran_year,
    p.payment_value
   from `targetSQL.payments` p
   inner join `targetSQL.orders` o
   on p.order_id = o.order_id
   order by tran_year,tran_month
  ),
 table2 as
  (
   select
    tran_year,
    round(sum(payment_value),2) as amount_sale,
     lag(round(sum(payment_value),2)) over(order by tran_year asc) as lag_amount
   from table1
   where tran_month between 1 and 8
   group by tran_year
   order by tran_year
  )
select
 tran_year,
 amount_sale,
 round((amount_sale - lag_amount)/lag_amount*100,2) as percentage_increase
from table2;
```



Row	tran_year	amount_sale	percentage_increase
1	2017	3669022.12	null
2	2018	8694733.84	136.98

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

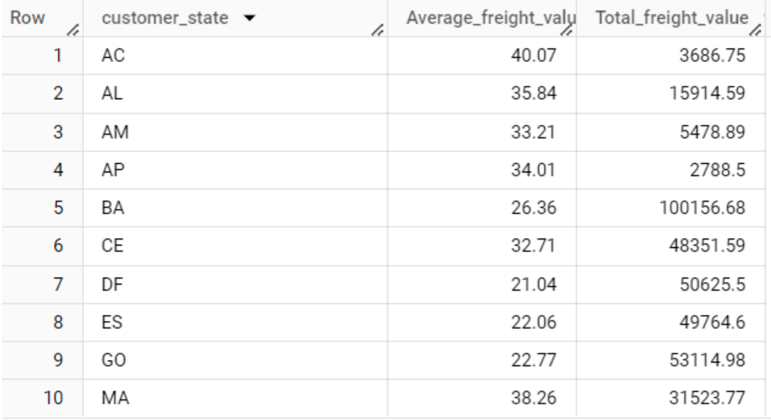
```
select
 c.customer_state,
 count(o.order_id) as total_orders,
 ceil(sum(oi.price)) as total_sale_value

from `targetSQL.orders` o
inner join `targetSQL.customer` c
on o.customer_id = c.customer_id
inner join `targetSQL.order_items` oi
on o.order_id = oi.order_id
group by c.customer_state
order by c.customer_state
```



Row	customer_state	total_orders	total_sale_value
1	AC	92	15983.0
2	AL	444	80315.0
3	AM	165	22357.0
4	AP	82	13475.0
5	BA	3799	511350.0
6	CE	1478	227255.0
7	DF	2406	302604.0
8	ES	2256	275038.0
9	GO	2333	294592.0
10	MA	824	119649.0

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

```
select
  c.customer_state,
  round (avg(oi.freight_value),2) as Average_freight_value,
  round(sum(oi.freight_value),2) as Total_freight_value
from `targetSQL.customer` c
inner join `targetSQL.orders` o ON o.customer_id = c.customer_id
inner join `targetSQL.order_items` oi ON o.order_id = oi.order_id
group by c.customer_state
order by c.customer_state
```

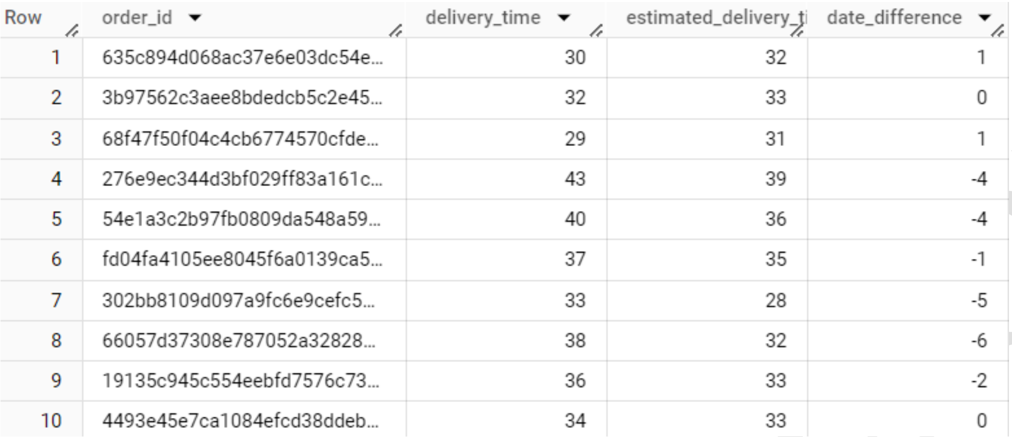


Row	customer_state	Average_freight_valu	Total_freight_value
1	AC	40.07	3686.75
2	AL	35.84	15914.59
3	AM	33.21	5478.89
4	AP	34.01	2788.5
5	BA	26.36	100156.68
6	CE	32.71	48351.59
7	DF	21.04	50625.5
8	ES	22.06	49764.6
9	GO	22.77	53114.98
10	MA	38.26	31523.77

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.

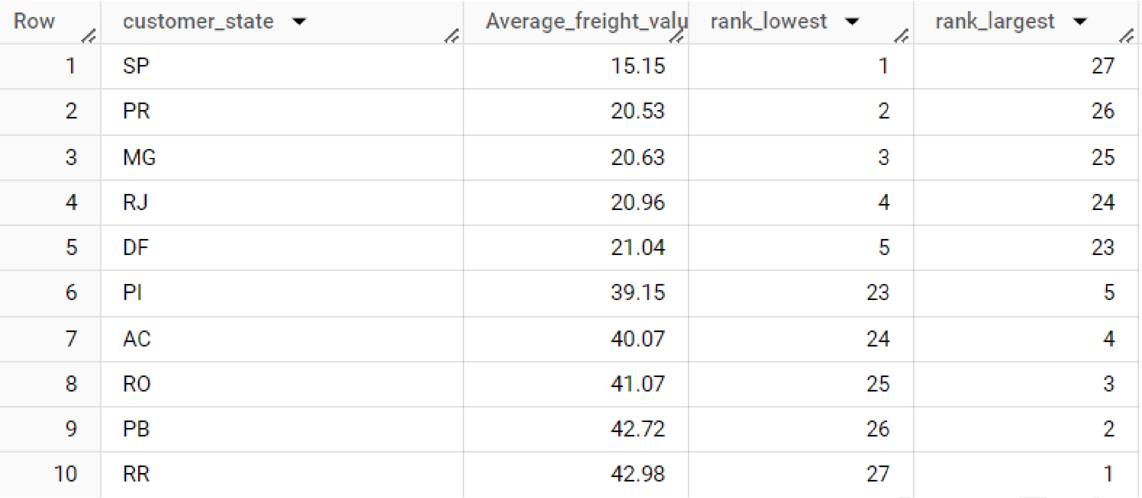
```
select
    order_id,
    date_diff(order_delivered_customer_date, order_purchase_timestamp,
day) as delivery_time,
    date_diff(order_estimated_delivery_date, order_purchase_timestamp, 
day) as estimated_delivery_time,
    date_diff(order_estimated_delivery_date, 
order_delivered_customer_date,day) as date_difference
from `targetSQL.orders`
where lower(order_status) = 'delivered'
```



Row	order_id	delivery_time	estimated_delivery_ti	date_difference
1	635c894d068ac37e6e03dc54e...	30	32	1
2	3b97562c3aee8bdedcb5c2e45...	32	33	0
3	68f47f50f04c4cb6774570cfde...	29	31	1
4	276e9ec344d3bf029ff83a161c...	43	39	-4
5	54e1a3c2b97fb0809da548a59...	40	36	-4
6	fd04fa4105ee8045f6a0139ca5...	37	35	-1
7	302bb8109d097a9fc6e9cefc5...	33	28	-5
8	66057d37308e787052a32828...	38	32	-6
9	19135c945c554eebfd7576c73...	36	33	-2
10	4493e45e7ca1084efcd38ddeb...	34	33	0

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

```
with
table1 as
  (
      select
        c.customer_state,
        round (avg(oi.freight_value),2) as Average_freight_value,
        rank() over(order by avg(oi.freight_value) asc) as rank_lowest,
        rank() over(order by avg(oi.freight_value) desc) as rank_largest
      from `targetSQL.customer` c
      inner join `targetSQL.orders` o ON o.customer_id = c.customer_id
      inner join `targetSQL.order_items` oi ON o.order_id = oi.order_id
      group by c.customer_state

  )
  select
    customer_state,
    Average_freight_value,
    rank_lowest,
    rank_largest
  from table1
  where rank_lowest<=5
  union all
   select
    customer_state,
    Average_freight_value,
    rank_lowest,
    rank_largest
  from table1
  where rank_largest<=5
  order by Average_freight_value;
```

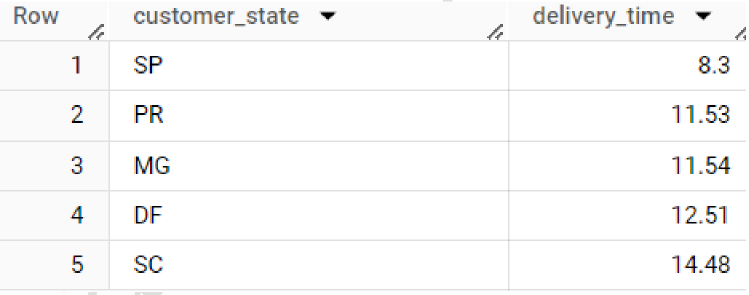


Row	customer_state	Average_freight_valu	rank_lowest	rank_largest
1	SP	15.15	1	27
2	PR	20.53	2	26
3	MG	20.63	3	25
4	RJ	20.96	4	24
5	DF	21.04	5	23
6	PI	39.15	23	5
7	AC	40.07	24	4
8	RO	41.07	25	3
9	PB	42.72	26	2
10	RR	42.98	27	1

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

Lowest average time

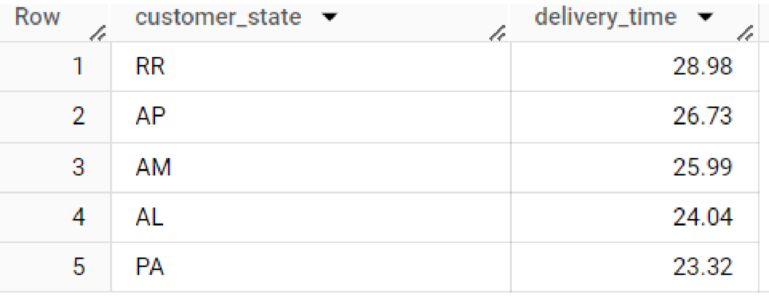
```
select
  c.customer_state,
  round(avg(date_diff(o.order_delivered_customer_date,
o.order_purchase_timestamp, day)),2) as delivery_time
from `targetSQL.customer` c
inner join `targetSQL.orders` o ON o.customer_id = c.customer_id
where lower(order_status) = 'delivered'
group by c.customer_state
order by delivery_time asc
limit 5
```



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

Highest average time.

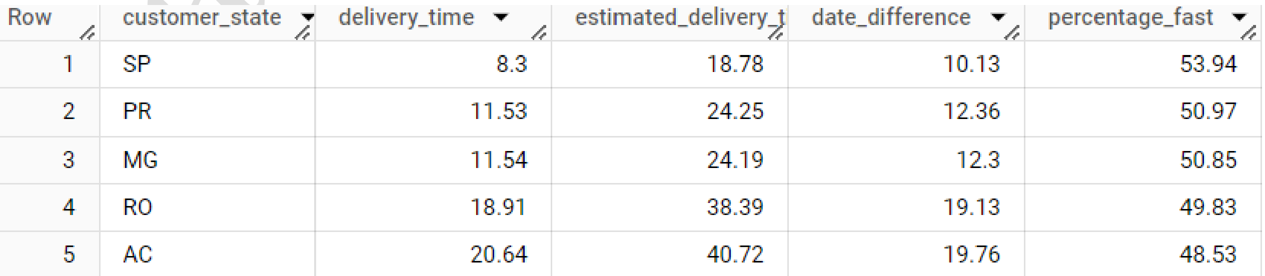
```
select
  c.customer_state,
  round(avg(date_diff(o.order_delivered_customer_date,
o.order_purchase_timestamp, day)),2) as delivery_time
from `targetSQL.customer` c
inner join `targetSQL.orders` o ON o.customer_id = c.customer_id
where lower(order_status) = 'delivered'
group by c.customer_state
order by delivery_time desc
limit 5
```



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

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.

```
with
    table1 as
    (
    select
        c.customer_state,
        round(avg(date_diff(o.order_delivered_customer_date,
o.order_purchase_timestamp, day)),2) as delivery_time,
        round(avg(date_diff(o.order_estimated_delivery_date,
o.order_purchase_timestamp, day)),2) as estimated_delivery_time,
        round(avg(date_diff(o.order_estimated_delivery_date,
o.order_delivered_customer_date,day)),2) as date_difference
    from `targetSQL.orders` o
    inner join `targetSQL.customer` c
    on o.customer_id = c.customer_id
    where lower(o.order_status) = 'delivered'
    group by c.customer_state
    )
select*,
  round((date_difference/estimated_delivery_time)*100,2) as percentage_fast
from table1
order by percentage_fast desc
limit 5
```

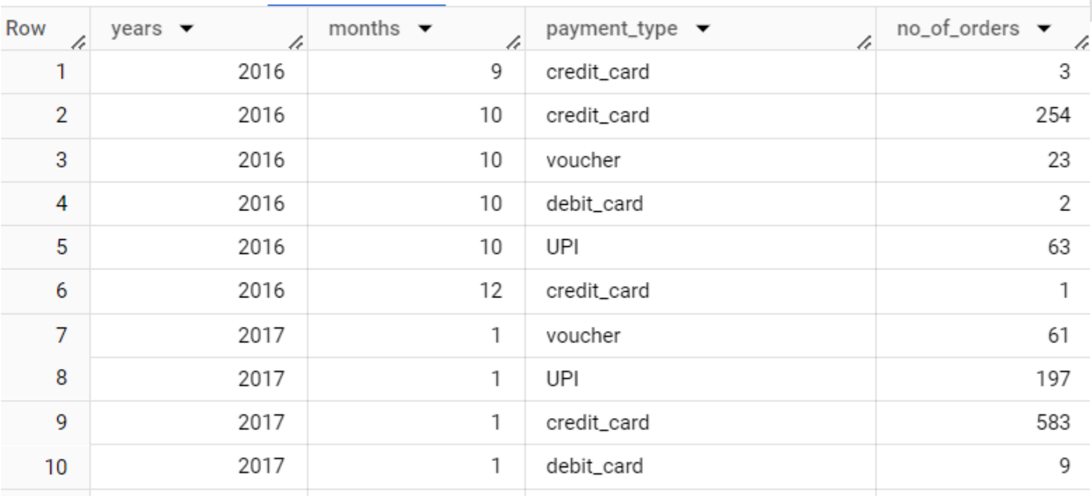


Row	customer_state	delivery_time	estimated_delivery_t	date_difference	percentage_fast
1	SP	8.3	18.78	10.13	53.94
2	PR	11.53	24.25	12.36	50.97
3	MG	11.54	24.19	12.3	50.85
4	RO	18.91	38.39	19.13	49.83
5	AC	20.64	40.72	19.76	48.53

Analysis based on the payments:

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

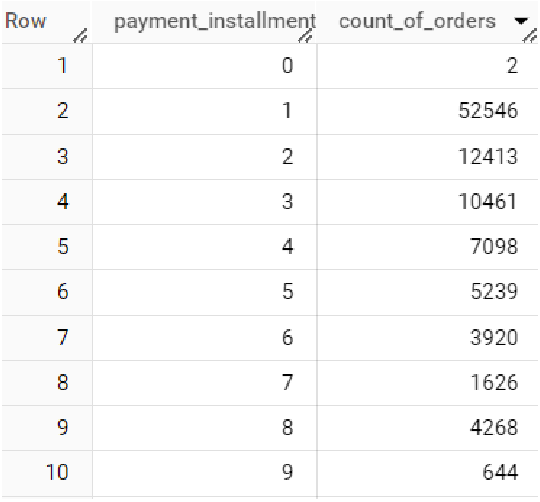
```
select
  extract(year from o.order_purchase_timestamp) as years,
  extract(month from o.order_purchase_timestamp)as months,
  p.payment_type,
  count(*) as no_of_orders
from `targetSQL.orders` o
join `targetSQL.payments` p
on o.order_id = p.order_id
group by years, months,p.payment_type
order by years,months;
```



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

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

```
select
  payment_installments,
  count(distinct order_id) as count_of_orders
from `targetSQL.payments`
group by payment_installments
```



Row	payment_installment	count_of_orders
1	0	2
2	1	52546
3	2	12413
4	3	10461
5	4	7098
6	5	5239
7	6	3920
8	7	1626
9	8	4268
10	9	644

Insights and Recommendations:-

- While states AP and AC have the lowest order volumes, their order approval times are notably fast at 16 and 15 hours, respectively.

- n contrast, states SP, MG, and PR not only lead in the percentage of on-time deliveries but also maintain low order approval times.

- To enhance order processing and delivery efficiency in other states, studying the strategies employed by SP, MG, and PR is crucial.

- Customer distribution in Brazil spans 27 states, with SP having the highest customer count at 40,302, followed by RJ with 12,384 and MG with 11,259 customers. Opening more warehouses and stores in such areas can reduce transportation costs and delivery times.

- The majority of orders are concentrated in the afternoon (12 PM - 6 PM) and night (6 PM - 12 AM), comprising over three-fourths of the total orders. Ensuring website performance and launching enticing offers during these peak hours can attract more customers.

- Prioritizing customer feedback and improving estimated delivery times is essential for enhancing the overall customer experience.

- In regions with a lower customer base such as AP, AC, and RR, Target can implement strategic marketing campaigns, offer discounts, and leverage social media promotions to attract more customers.

- Credit card payments are the most popular choice for orders, followed by UPI, vouchers, and debit cards, with a similar trend in payment values