1. **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.

   Ans A:

```
select
  column_name,
  data_type
from
  Target_SQL.INFORMATION_SCHEMA.COLUMNS
WHERE
  table_name='customers';
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

ow	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: Most of the columns datatypes are in STRING.

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

Ans B:

```
select
  min(order_purchase_timestamp) as frist_order,
  max(order_purchase_timestamp) as last_order
from Target_SQL.orders;
```

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

Insights: The first order was placed at 2016 and last order is in 2018 there is two years the customer is not purchasing.

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

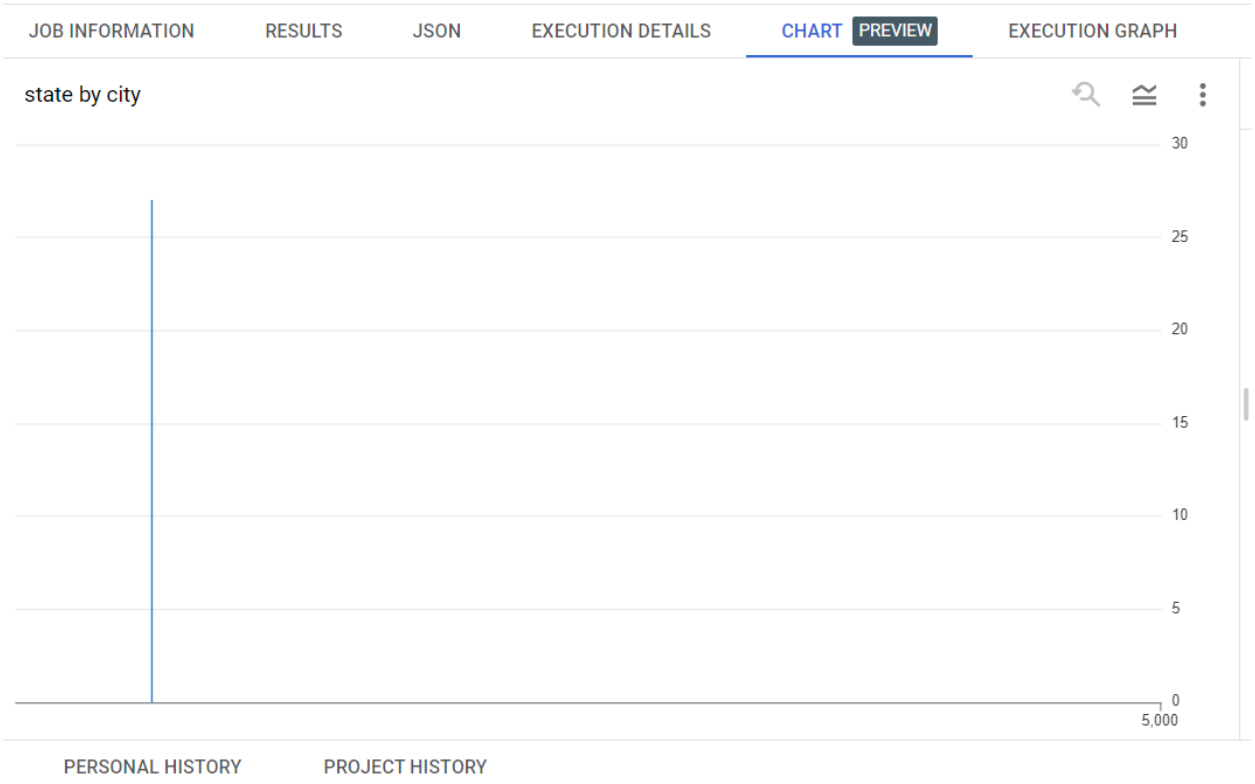
Ans C:

```
select count(distinct customer_city) as city,
    count(distinct customer_state) as state
from `Target_SQL.customers`
```

Row	city	state
1	4119	27

Insights: There were a total cities 4119 & states are 27.

CHART VIEW:



2. In-depth Exploration:

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

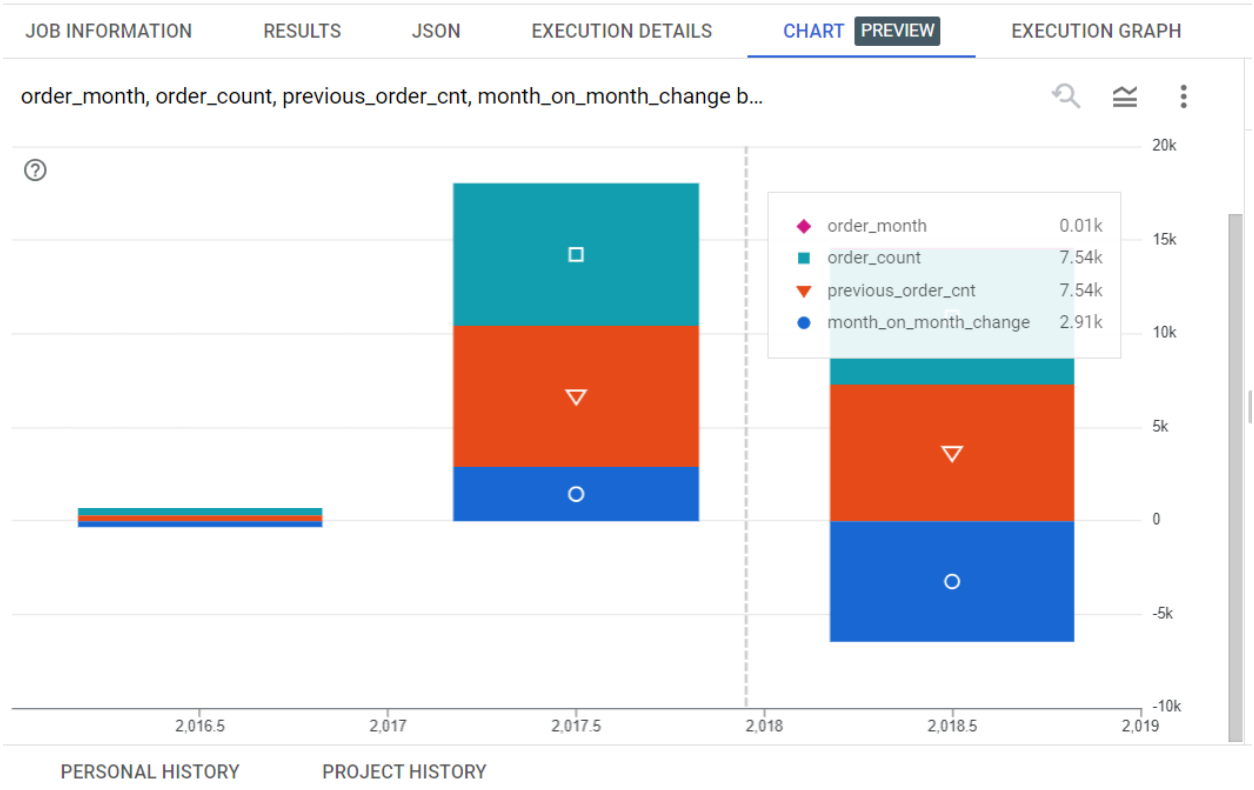
```
with pastyr as(
SELECT
    EXTRACT(YEAR FROM order_purchase_timestamp) AS order_year,
    EXTRACT(MONTH FROM order_purchase_timestamp) AS order_month,
    COUNT(*) AS order_count
  FROM
    `Target_SQL.orders`
  GROUP BY
    EXTRACT(YEAR FROM order_purchase_timestamp),
    EXTRACT(MONTH FROM order_purchase_timestamp)
)
select *,
  lag(order_count) over (order by order_year,order_month) as
previous_order_cnt,
  order_count - lag (order_count) over (order by order_year,order_month) as
month_on_month_change,
from
  pastyr
ORDER BY
order_year, order_month
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | CHART | PREVIEW | EXE

Row	order_year	order_month	order_count	previous_order_cnt	month_on_month_ch
1	2016	9	4	*null*	*null*
2	2016	10	324	4	320
3	2016	12	1	324	-323
4	2017	1	800	1	799
5	2017	2	1780	800	980
6	2017	3	2682	1780	902
7	2017	4	2404	2682	-278
8	2017	5	3700	2404	1296
9	2017	6	3245	3700	-455
10	2017	7	4026	3245	781
11	2017	8	4331	4026	305
12	2017	9	4285	4331	-46
13	2017	10	4631	4285	346
14	2017	11	7544	4631	2913

Load more

CHART VIEW:



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

Ans B:

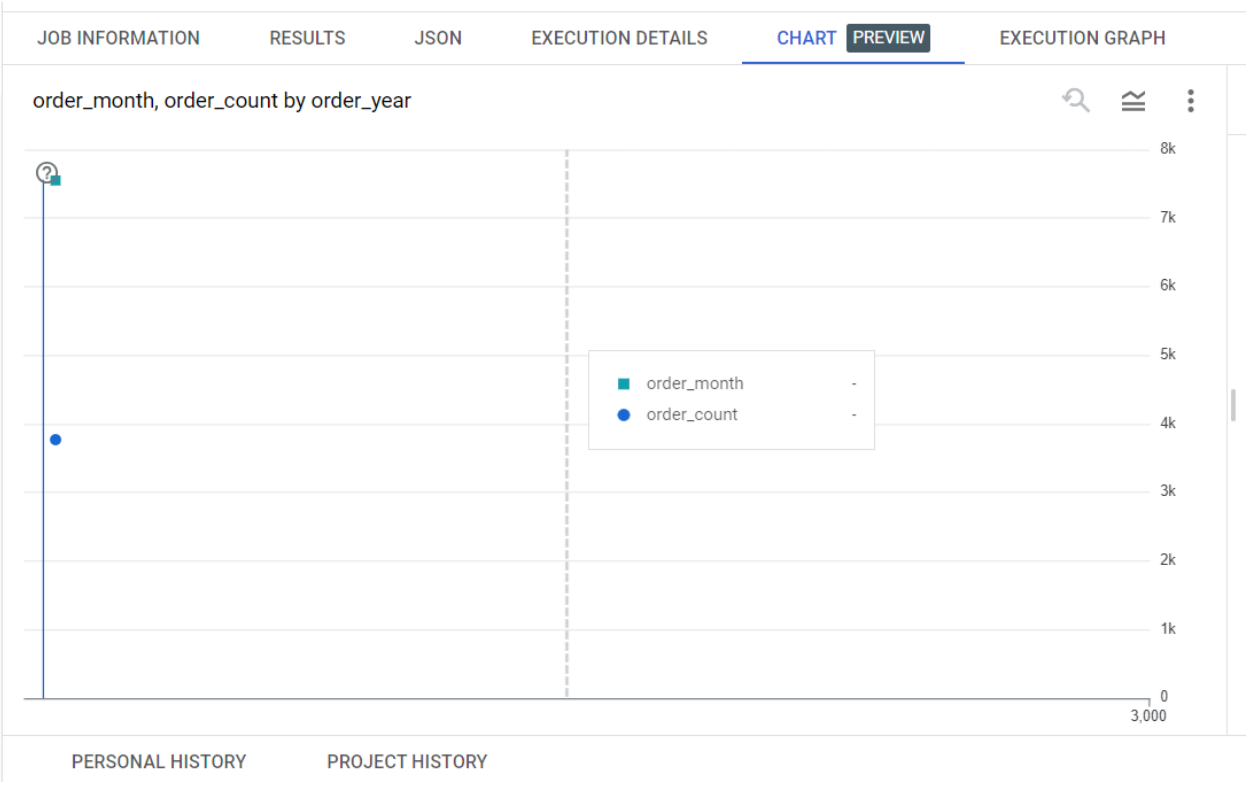
```
with cte as(
 SELECT
     EXTRACT(YEAR FROM order_purchase_timestamp) AS order_year,
     EXTRACT(MONTH FROM order_purchase_timestamp) AS
order_month,
     COUNT(*) AS order_count
  FROM
     `Target_SQL.orders`
  GROUP BY
     EXTRACT(YEAR FROM order_purchase_timestamp),
     EXTRACT(MONTH FROM order_purchase_timestamp)
)
 select * from cte
 where order_count = (SELECT MAX(order_count) FROM cte)
```

order by order_year,order_month

Row	order_year	order_month	order_count
1	2017	11	7544

Insights: The most orders order from 2017 in November month

CHART VIEW:



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

Ans c:

```
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'
    else 'unkown'
 end as time_of_day,
 count(*) as ordered_count
from `Target_SQL.orders`
group by time_of_day
order by time_of_day;
```

Insights: Most of the orders are placed in Afternoon

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

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

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

   Ans A:

```
select
  extract(year from o.order_purchase_timestamp) as _years,
  extract(month from o.order_purchase_timestamp) as _month,
  c.customer_state,
  count(*) as number_orders
from `Target_SQL.customers` c join `Target_SQL.orders` o on o.customer_id=c.customer_id
group by _years,_month,c.customer_state
```

```
order by _years,_month,number_orders;
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | CHART PREVIEW

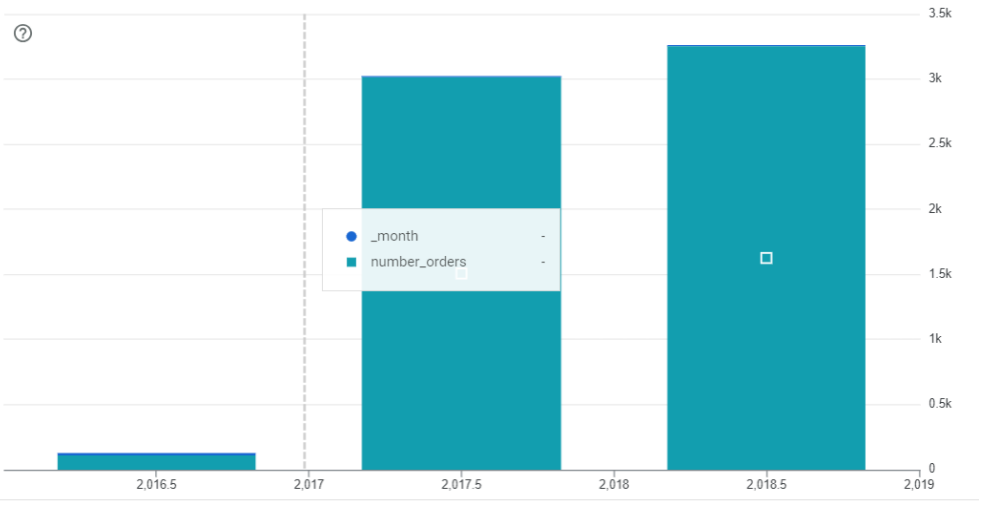
Row	_years	_month	customer_state	number_orders
1	2016	9	RR	1
2	2016	9	RS	1
3	2016	9	SP	2
4	2016	10	PB	1
5	2016	10	PI	1
6	2016	10	RR	1
7	2016	10	AL	2
8	2016	10	MT	3
9	2016	10	SE	3
10	2016	10	BA	4
11	2016	10	RN	4
12	2016	10	PA	4
13	2016	10	MA	4
14	2016	10	ES	4

Load more

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | CHART PREVIEW | EXECUTION GRAPH

_month, number_orders by _years



PERSONAL HISTORY | PROJECT HISTORY

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

Ans B:

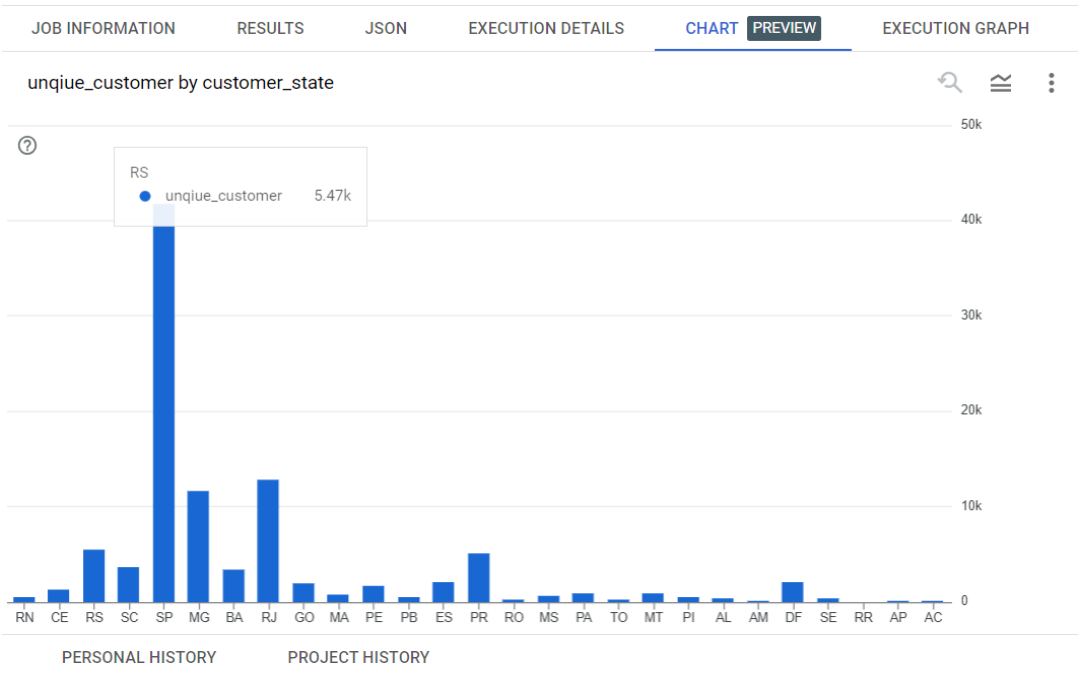
```
select
customer_state,
count(distinct customer_id) as unqiue_customer
from `Target_SQL.customers`
group by customer_state;
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAI

Row	customer_state	unqiue_customer
1	RN	485
2	CE	1336
3	RS	5466
4	SC	3637
5	SP	41746

Insights: MAX of the unique customers in SP state

CHART VIEW:



4. 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).

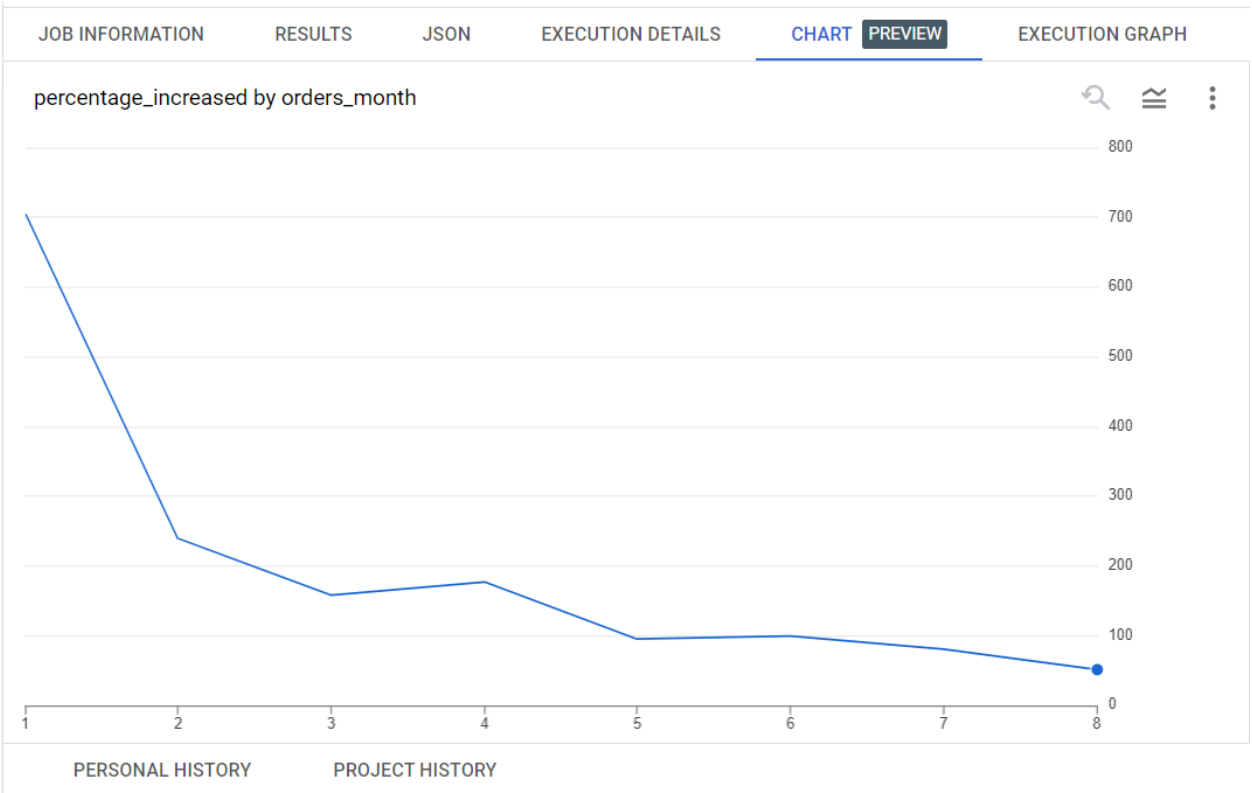
Ans A:

```
select
        extract(month from order_purchase_timestamp) as orders_month,
        round(((sum(case when extract(year from o.order_purchase_timestamp)= 2018 then p.payment_value else 0 end) -
        sum(case when extract(year from o.order_purchase_timestamp)= 2017 then p.payment_value else 0 end)) /
        sum(case when extract(year from o.order_purchase_timestamp)= 2017 then p.payment_value else 0 end)) * 100,2) as percentage_increased
    from `Target_SQL.orders` o join `Target_SQL.payments` p on p.order_id=o.order_id
    where extract(month from order_purchase_timestamp) between 1 and 8
    group by orders_month
    order by orders_month;
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	orders_month	percentage_increase
1	1	705.13
2	2	239.99
3	3	157.78
4	4	177.84
5	5	94.63
6	6	100.26
7	7	80.04
8	8	51.61

CHART VIEW:



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

Ans B:

```
select
    c.customer_state,
     round(sum(oi.price),2) as sum_each_state,
     round(avg(oi.price),2) as avg_each_state
from `Target_SQL.customers` c join `Target_SQL.orders` o on
c.customer_id=o.customer_id
                    join `Target_SQL.order_items` oi on o.order_id=oi.order_id
   group by c.customer_state    order by c.customer_state
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	sum_each_state	avg_each_state
1	AC	15982.95	173.73
2	AL	80314.81	180.89
3	AM	22356.84	135.5
4	AP	13474.3	164.32
5	BA	511349.99	134.6
6	CE	227254.71	153.76
7	DF	302603.94	125.77
8	ES	275037.31	121.91
9	GO	294591.95	126.27
10	MA	119648.22	145.2
11	MG	1585308.03	120.75
12	MS	116812.64	142.63
13	MT	156453.53	148.3
14	PA	178947.81	165.69

Load more

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

Ans C:

```
select
        c.customer_state,
        round(sum(oi.freight_value),2) as sum_freight_val,
        round(avg(oi.freight_value),2) as avg_freight_val
from `Target_SQL.customers` c join `Target_SQL.orders` o on c.customer_id=o.customer_id
        join `Target_SQL.order_items` oi on o.order_id=oi.order_id
group by c.customer_state
order by c.customer_state
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	sum_freight_val	avg_freight_val
1	AC	3686.75	40.07
2	AL	15914.59	35.84
3	AM	5478.89	33.21
4	AP	2788.5	34.01
5	BA	100156.68	26.36
6	CE	48351.59	32.71
7	DF	50625.5	21.04
8	ES	49764.6	22.06
9	GO	53114.98	22.77
10	MA	31523.77	38.26
11	MG	270853.46	20.63
12	MS	19144.03	23.37
13	MT	29715.43	28.17
14	PA	38699.3	35.83

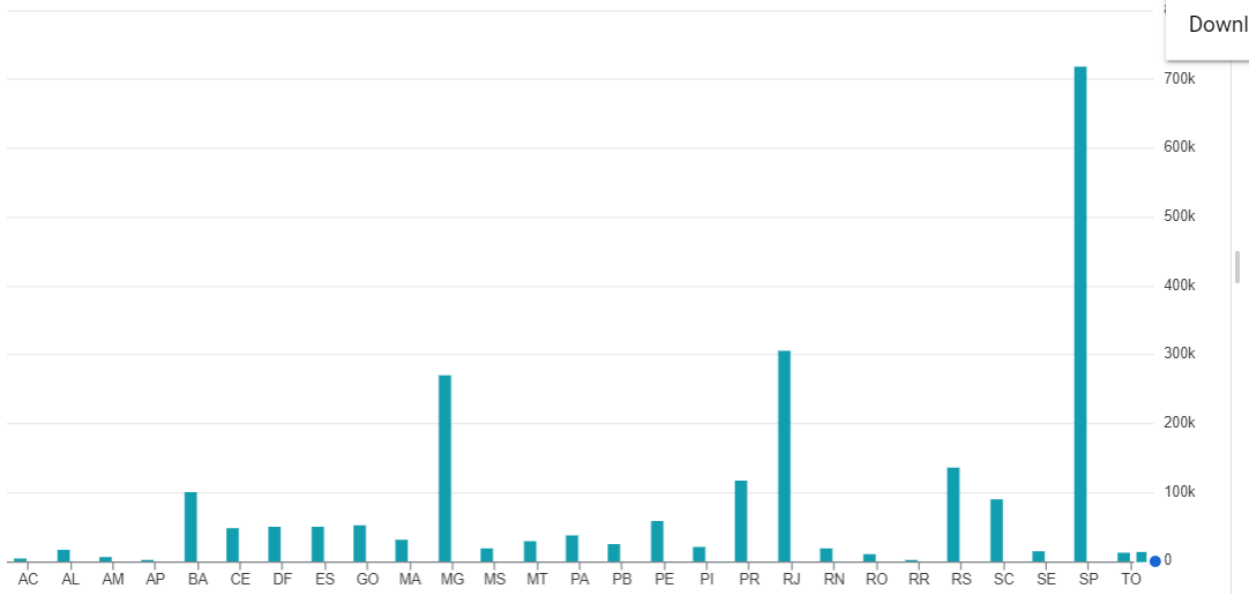
Load more

Query results

SAVE

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | CHART PREVIEW | EXECUTION GRAPH

sum_freight_val, avg_freight_val by customer_state



5. 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**.**

Ans A:

```
select
distinct order_id,
order_delivered_customer_date,
 order_estimated_delivery_date,
datetime_diff(order_estimated_delivery_date,order_delivered_customer_date, DAY ) AS days_diff
from `Target_SQL.orders`
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | CHART PREVIEW | EXECUTION GRAPH

Row	order_id	order_delivered_customer_date	order_estimated_delivery_date	days_diff
1	770d331c84e5b214bd9dc70a1...	2016-10-14 15:07:11 UTC	2016-11-29 00:00:00 UTC	45
2	1950d777989f6a877539f5379...	2018-03-21 22:03:51 UTC	2018-03-09 00:00:00 UTC	-12
3	2c45c33d2f9cb8ff8b1c86cc28...	2016-11-09 14:53:50 UTC	2016-12-08 00:00:00 UTC	28
4	dabf2b0e35b423f94618bf965f...	2016-10-16 14:36:59 UTC	2016-11-30 00:00:00 UTC	44
5	8beb59392e21af5eb9547ae1a...	2016-10-19 18:47:43 UTC	2016-11-30 00:00:00 UTC	41
6	b60b53ad0bb7dacacf2989fe2...	2017-05-23 13:12:27 UTC	2017-05-18 00:00:00 UTC	-5
7	276e9ec344d3bf029ff83a161c...	2017-05-22 14:11:31 UTC	2017-05-18 00:00:00 UTC	-4
8	1a0b31f08d0d7e87935b819ed...	2017-04-18 08:18:11 UTC	2017-05-18 00:00:00 UTC	29
9	cec8f5f7a13e5ab934a486ec9e...	2017-04-07 13:14:56 UTC	2017-05-18 00:00:00 UTC	40
10	2d846c03073b1a424c1be1a77...	2017-05-25 10:49:48 UTC	2017-05-18 00:00:00 UTC	-7
11	54e1a3c2b97fb0809da548a59...	2017-05-22 16:18:42 UTC	2017-05-18 00:00:00 UTC	-4
12	58527ee4726911bee84a0f42c...	2017-03-30 14:04:04 UTC	2017-05-18 00:00:00 UTC	48
13	302bb8109d097a9fc6e9cefc5...	2017-05-23 14:19:48 UTC	2017-05-18 00:00:00 UTC	-5
14	10ed5499d1623638ee810eff1...	2017-04-18 13:52:43 UTC	2017-05-18 00:00:00 UTC	29

Load more

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

Ans B:

```
        with tuffs as
(
  select
     c.customer_state,
     round(avg(oi.freight_value),2) as top_avg_freight
  from `Target_SQL.customers` c
  join `Target_SQL.orders` o on o.customer_id = c.customer_id
  join `Target_SQL.order_items` oi on o.order_id = oi.order_id
  group by c.customer_state
  order by round(avg(oi.freight_value),2) desc
  limit 5
),
bmc as
(
  select
     c.customer_state,
     round(avg(oi.freight_value),2) as bottomo_avg_freight
  from `Target_SQL.customers` c
  join `Target_SQL.orders` o on o.customer_id = c.customer_id
  join `Target_SQL.order_items` oi on o.order_id = oi.order_id
  group by c.customer_state
  order by round(avg(oi.freight_value),2) asc
  limit 5
)
select * from tuffs
union distinct
select * from bmc;
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	top_avg_freight
1	RR	42.98
2	PB	42.72
3	RO	41.07
4	AC	40.07
5	PI	39.15
6	SP	15.15
7	PR	20.53
8	MG	20.63
9	RJ	20.96
10	DF	21.04

Insights: Here are the Top 5 highest avg and bottom 5 lowest

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

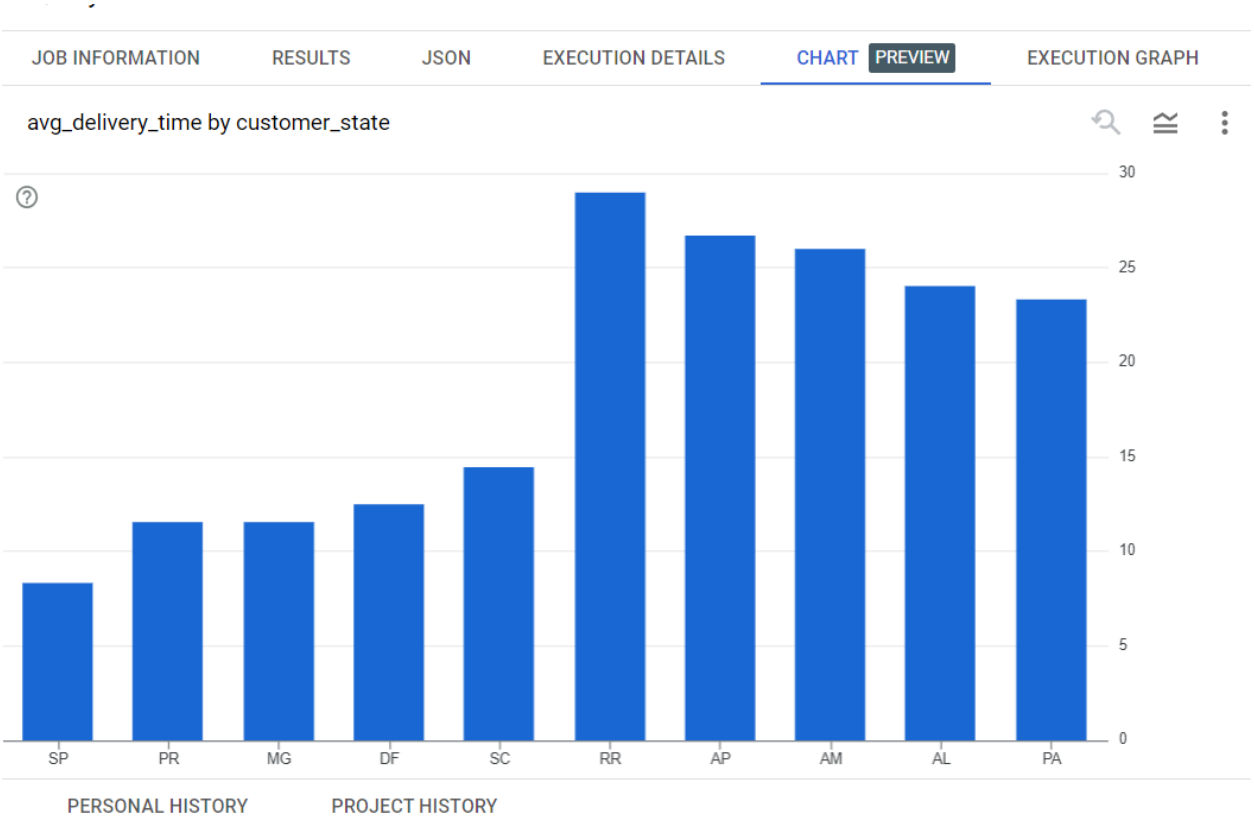
Ans C:

```
with avg_delivery_desc as (
select
    customer_state,

     round(avg(date_diff(order_delivered_customer_date,order_purchase_timestam
     p,DAY)),2) as avg_delivery_time
     from `Target_SQL.customers` c join `Target_SQL.orders` o on
     c.customer_id=o.customer_id
     group by c.customer_state
     order by avg_delivery_time desc
     limit 5
),
avg_delivering_asc as
(select
    customer_state,

     round(avg(date_diff(order_delivered_customer_date,order_purchase_timestam
     p,DAY)),2) as avg_delivery_time
     from `Target_SQL.customers` c join `Target_SQL.orders` o on
     c.customer_id=o.customer_id
     group by c.customer_state
     order by avg_delivery_time asc
l    imit 5
)
select * from avg_delivery_desc
union distinct
select * from avg_delivering_asc
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAIL

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

Insights: Here the state wise delivering avg top and bottom



D. 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.

Ans D:

```
with blc as
(
select
   c.customer_state,

avg(date_diff(o.order_delivered_customer_date,o.order_estimated_delivery_date,D
AY)) as avg_delivering
from
   `Target_SQL.customers` c join `Target_SQL.orders` o on
c.customer_id=o.customer_id
where order_status= 'delivered'
group by c.customer_state
)
select customer_state from blc
order by avg_delivering desc
limit 5;
```

JOB INFORMATION | RESULTS | JSON

Row	customer_state
1	AL
2	MA
3	SE
4	ES
5	BA

Insights: This are the 5 states fast deliveries and considering only the delivered orders

6. Analysis based on the payments:

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

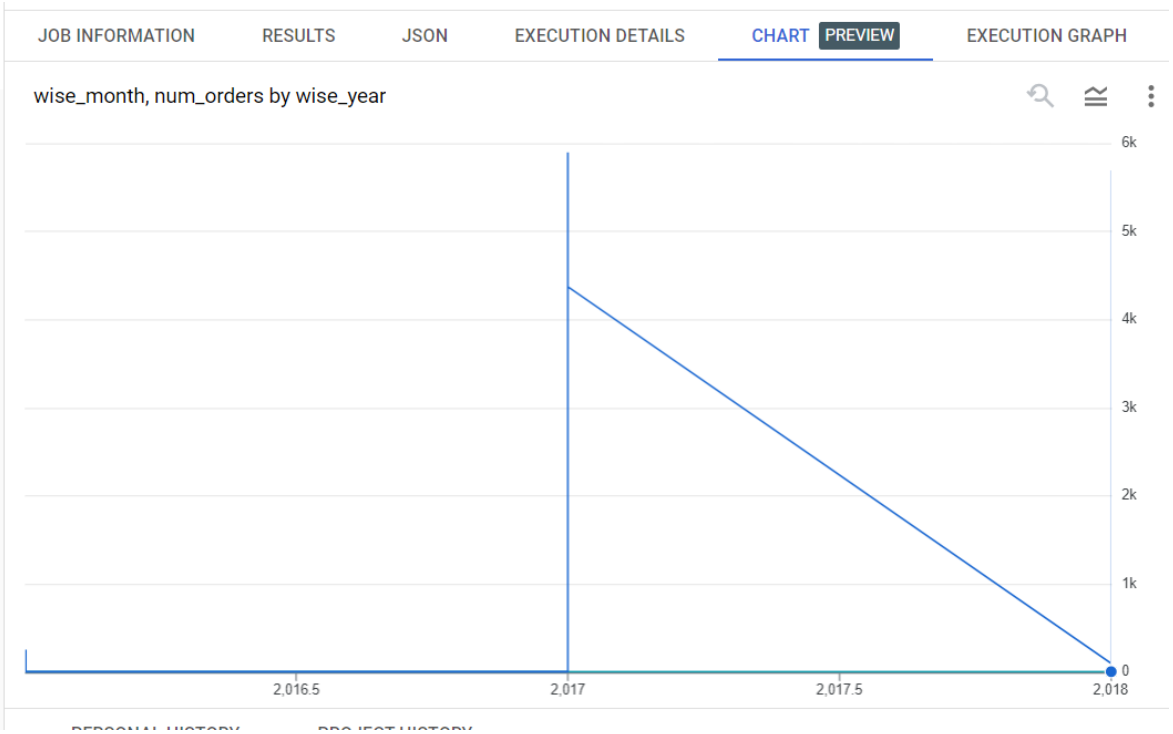
Ans A:

```
select
   extract(year from order_purchase_timestamp) as wise_year,
   extract(MONTH FROM o.order_purchase_timestamp) as wise_month,
   p.payment_type,
   count(*) as num_orders
from `Target_SQL.orders` o join `Target_SQL.payments` p on p.order_id=o.order_id
group by wise_year,wise_month, p.payment_type
order by wise_year,wise_month,num_orders;
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | CHART PREVIEW

Row	wise_year	wise_month	payment_type	num_orders
1	2016	9	credit_card	3
2	2016	10	debit_card	2
3	2016	10	voucher	23
4	2016	10	UPI	63
5	2016	10	credit_card	254
6	2016	12	credit_card	1
7	2017	1	debit_card	9
8	2017	1	voucher	61
9	2017	1	UPI	197
10	2017	1	credit_card	583
11	2017	2	debit_card	13
12	2017	2	voucher	119
13	2017	2	UPI	398
14	2017	2	credit_card	1356

Load more



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

Ans B:

```
select
   count(*) as orders_placed_on_installments
from `Target_SQL.payments`
where payment_installments <= 1
```

JOB INFORMATION | RESULTS | JSON

Row	orders_placed_on_installments
1	52548