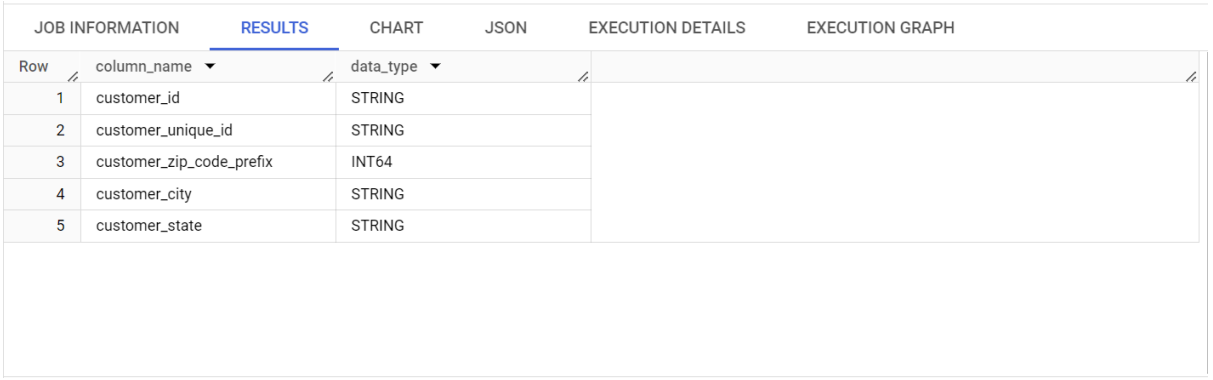
--1. Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset:
--1.1 Data type of all columns in the "customers" table.

Code:

```
select      column_name,
            data_type
from        `amazing-height-433215-c4.target.INFORMATION_SCHEMA.COLUMNS`
where       table_name = 'customers';
```

Result:



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

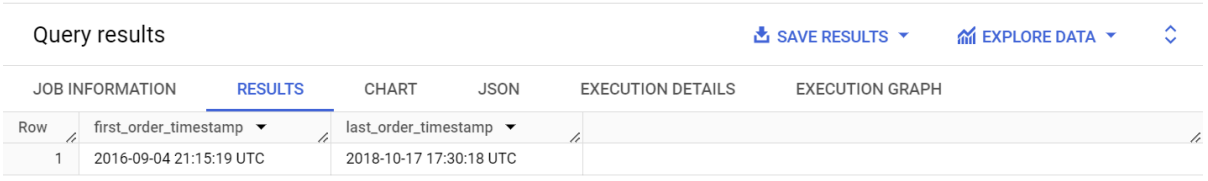
Row	column_name	data_type
1	customer_id	STRING
2	customer_unique_id	STRING
3	customer_zip_code_prefix	INT64
4	customer_city	STRING
5	customer_state	STRING

--1.2 Get the time range between which the orders were placed.

Code:

```
select      min(order_purchase_timestamp) as first_order_timestamp,
            max(order_purchase_timestamp) as last_order_timestamp
from `target.orders`;
```

Result:



Query results — SAVE RESULTS — EXPLORE DATA

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

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

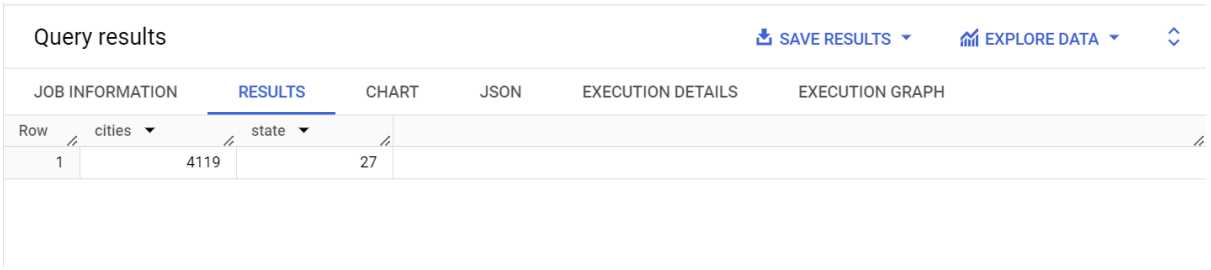
--1.3 Count the Cities & States of customers who ordered during the given period.

Code:

```
with cte as (select min(order_purchase_timestamp) as min_time,
                    max(order_purchase_timestamp) as max_time
             from `target.orders`)
```

```sql
select    count(distinct b.customer_city) as cities,
          count(distinct b.customer_state) as state
from      `target.orders` a
join      `target.customers` b
on        a.customer_id = b.customer_id
join      cte c
on        a.order_purchase_timestamp between min_time and max_time;
```

Result:

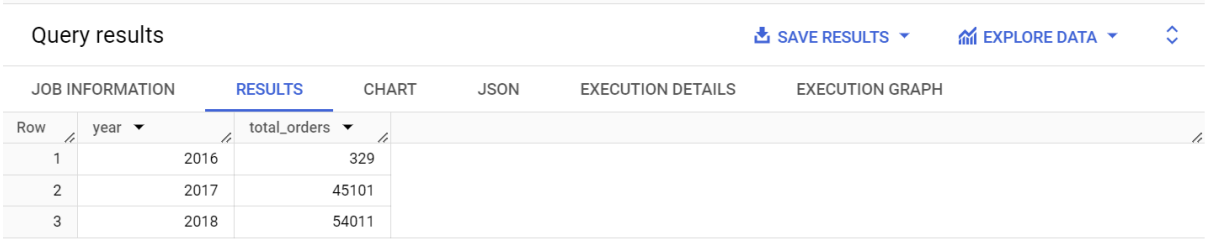



```sql
--2. In-depth Exploration:
--2.1 Is there a growing trend in the no. of orders placed over the past years?
-- YOY change in the total order placed
```

Code:

```sql
select      extract(year from a.order_purchase_timestamp) as year,
            count(a.order_id) as total_orders
from        `target.orders` a
group by 1
order by 1;
```

Result:



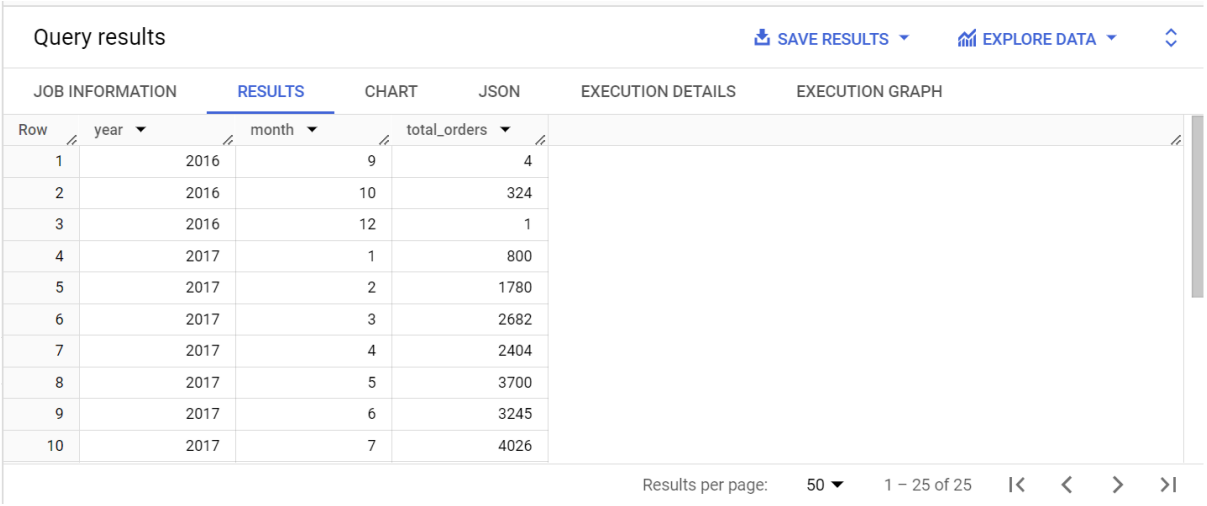

```sql
--2.2 Can we see some kind of monthly seasonality in terms of the no. of orders
being placed?
```

Code:

```sql
select  extract(year from a.order_purchase_timestamp) as year,
        extract(month from a.order_purchase_timestamp) as month,
        count(a.order_id) as total_orders
```

```
from    `target.orders` a
group by 1,2
order by 1,2;
```

Result:



Query results

SAVE RESULTS | EXPLORE DATA

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

Row	year	month	total_orders
1	2016	9	4
2	2016	10	324
3	2016	12	1
4	2017	1	800
5	2017	2	1780
6	2017	3	2682
7	2017	4	2404
8	2017	5	3700
9	2017	6	3245
10	2017	7	4026

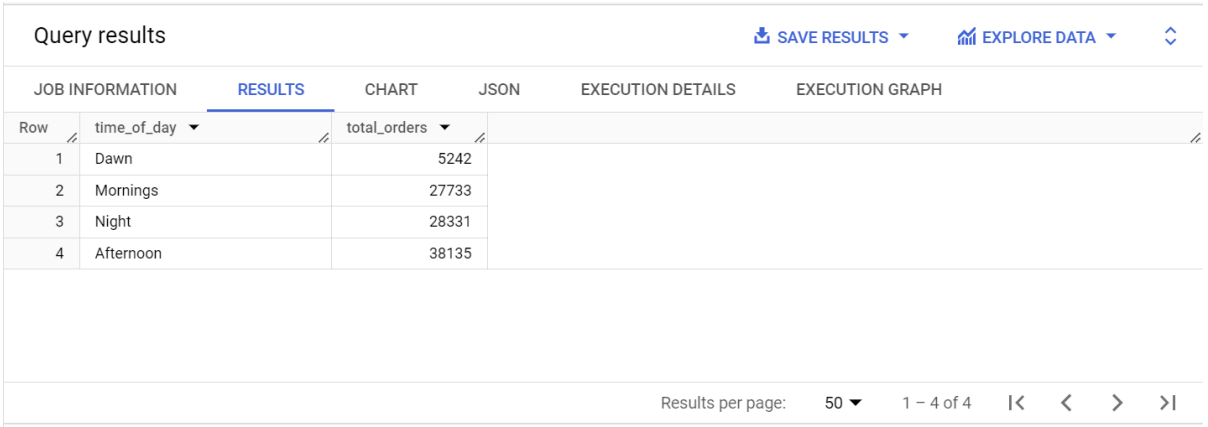
Results per page: 50 | 1 – 25 of 25

```
--2.3 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 */
```

Code:

```
select  case when extract(hour from a.order_purchase_timestamp) between 0 and 6 then 'Dawn'
            when extract(hour from a.order_purchase_timestamp) between 7 and 12 then 'Mornings'
            when extract(hour from a.order_purchase_timestamp) between 13 and 18 then 'Afternoon'
            when extract(hour from a.order_purchase_timestamp) between 19 and 23 then 'Night' end as time_of_day
,       count(order_id) as total_orders
from `target.orders` a
join `target.customers` b
on a.customer_id = b.customer_id
group by 1
order by 2;
```

Result:



Query results

SAVE RESULTS | EXPLORE DATA

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

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

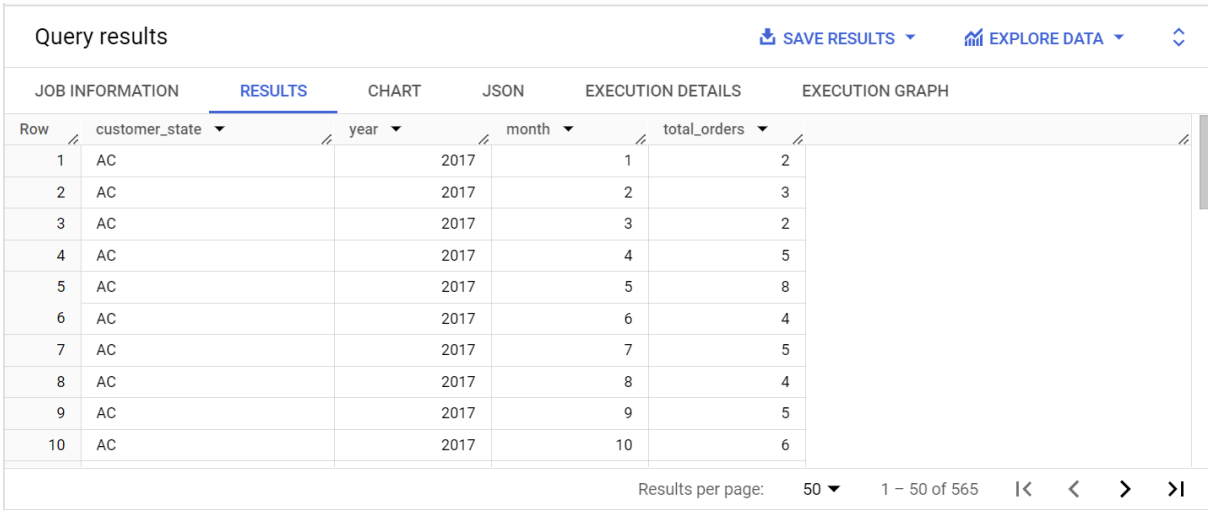
Results per page: 50 | 1 – 4 of 4

```
--3. Evolution of E-commerce orders in the Brazil region:
--3.1 Get the month on month no. of orders placed in each state.
```

Code:

```
select    b.customer_state,
          extract(year from a.order_purchase_timestamp) as year,
          extract(month from a.order_purchase_timestamp) as month,
          count(order_id) as total_orders
from      `target.orders` a
join      `target.customers` b
on        a.customer_id = b.customer_id
group by 1,2,3
order by 1,2,3;
```

Result:



Query results

SAVE RESULTS | EXPLORE DATA

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

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

Results per page: 50 | 1 – 50 of 565

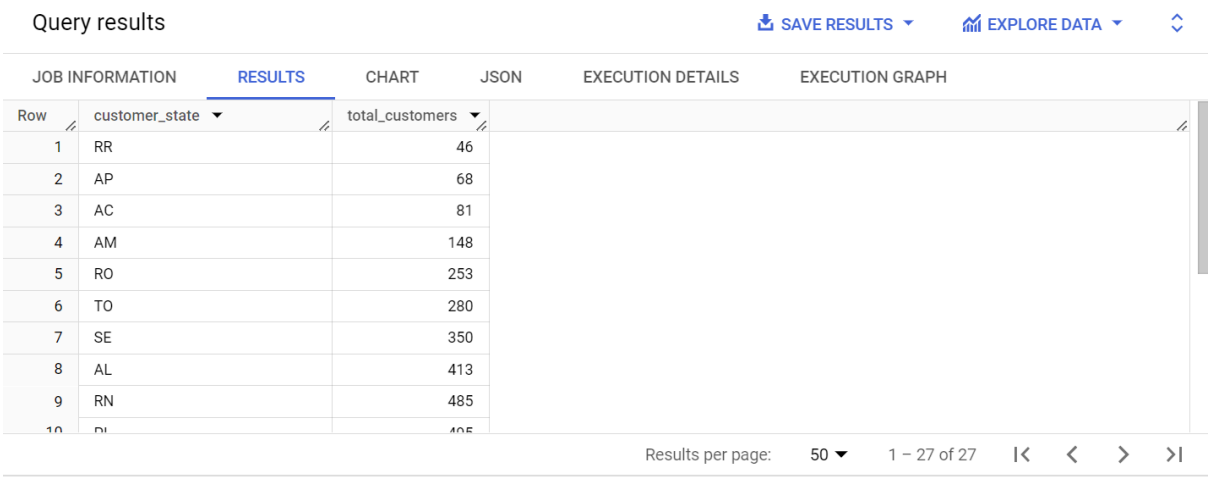
```
--3.2 How are the customers distributed across all the states?
```

Code:

```
select    customer_state, count(customer_id) as total_customers
from      `target.customers`
group by  1
```

```
order by  2;
```

Result:




```
--4. Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight and others.
--4.1 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.
```

Code:

```
with cte as (
select extract(year from a.order_purchase_timestamp) as year,
       extract(month from a.order_purchase_timestamp) as month,
        round(sum(b.payment_value)) as prev_cost
from `target.orders` a
join `target.payments` b
on a.order_id = b.order_id
where extract(year from a.order_purchase_timestamp) = 2017
and extract(month from a.order_purchase_timestamp) between 1 and 8
group by 1,2),
cte2 as (
select extract(year from a.order_purchase_timestamp) as year,
       extract(month from a.order_purchase_timestamp) as month,
       round(sum(b.payment_value)) as current_cost
from `target.orders` a
join `target.payments` b
on a.order_id = b.order_id
where extract(year from a.order_purchase_timestamp) = 2018
and extract(month from a.order_purchase_timestamp) between 1 and 8
group by 1,2)
select cte.year,cte.month,prev_cost,
```

```
        cte2.year,cte2.month,current_cost,
        round((cte2.current_cost - cte.prev_cost)*100/cte.prev_cost) as
percentage_increase
from cte
join cte2
on cte.year<cte2.year and cte.month = cte2.month
order by 1,2,4,5;
```

Result:

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

Row	year	month	prev_cost	year_1	month_1	current_cost	percentage_increase
1	2017	1	138488.0	2018	1	1115004.0	705.0
2	2017	2	291908.0	2018	2	992463.0	240.0
3	2017	3	449864.0	2018	3	1159652.0	158.0
4	2017	4	417788.0	2018	4	1160785.0	178.0
5	2017	5	592919.0	2018	5	1153982.0	95.0
6	2017	6	511276.0	2018	6	1023880.0	100.0
7	2017	7	592383.0	2018	7	1066541.0	80.0
8	2017	8	674396.0	2018	8	1022425.0	52.0

```
--4.2 Calculate the Total & Average value of order price for each state.
```

Code:

```
select      b.customer_state as state ,
            round(sum(payment_value)) as total_cost,
            round(avg(payment_value)) as avg_cost
from        `target.orders` a
join        `target.customers` b
on          a.customer_id = b.customer_id
join        `target.payments` c
on          a.order_id = c.order_id
group by 1
order by 2,3 desc;
```

Result:

Query results

SAVE RESULTS | EXPLORE DATA

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

Row	state	total_cost	avg_cost
1	RR	10065.0	219.0
2	AP	16263.0	232.0
3	AC	19681.0	234.0
4	AM	27967.0	182.0
5	RO	60866.0	233.0
6	TO	61485.0	204.0
7	SE	75246.0	208.0
8	AL	96962.0	227.0
9	RN	102718.0	197.0
10	PI	108524.0	207.0

Results per page: 50 1 – 27 of 27

--4.3 Calculate the Total & Average value of order freight for each state.

Code:

```
select      b.customer_state,
            round(sum(c.freight_value)) as total_freight,
            round(avg(c.freight_value)) as avg_freight
from        `target.orders` a
join        `target.customers` b
on          a.customer_id = b.customer_id
join        `target.order_items` c
on          a.order_id = c.order_id
group by 1
order by 2,3 desc;
```

Result:

Query results

SAVE RESULTS | EXPLORE DATA

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

Row	customer_state	total_freight	avg_freight
1	RR	2235.0	43.0
2	AP	2789.0	34.0
3	AC	3687.0	40.0
4	AM	5479.0	33.0
5	RO	11417.0	41.0
6	TO	11733.0	37.0
7	SE	14111.0	37.0
8	AL	15915.0	36.0
9	RN	18860.0	36.0
10	MS	19144.0	23.0

Results per page: 50 1 – 27 of 27

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

--5.1 Find the no. of days taken to deliver each order from the order's purchase date as delivery time.

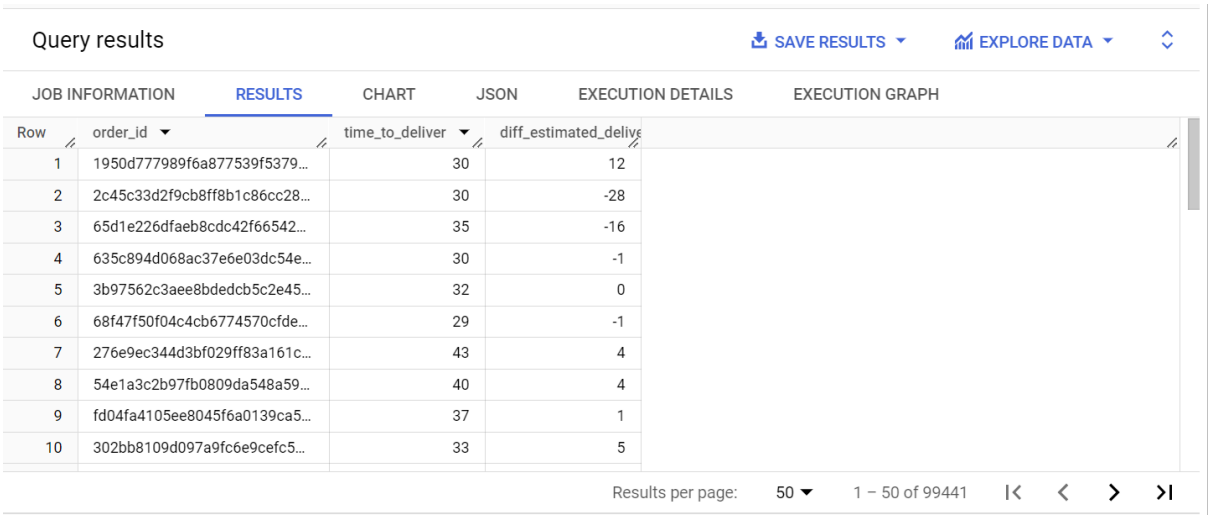
```
/* Also, calculate the difference (in days) between the estimated & actual delivery
date of an order.
   Do this in a single query.

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_delivered_customer_date -
order_estimated_delivery_date */
```

Code:

```
select    distinct order_id,
          date_diff(order_delivered_customer_date,order_purchase_timestamp,day) as
time_to_deliver,
          date_diff(order_delivered_customer_date,
order_estimated_delivery_date,day) as diff_estimated_delivery
from      `target.orders` ;
```

Result:



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


```
--5.2 Find out the top 5 states with the highest & lowest average freight value.
```

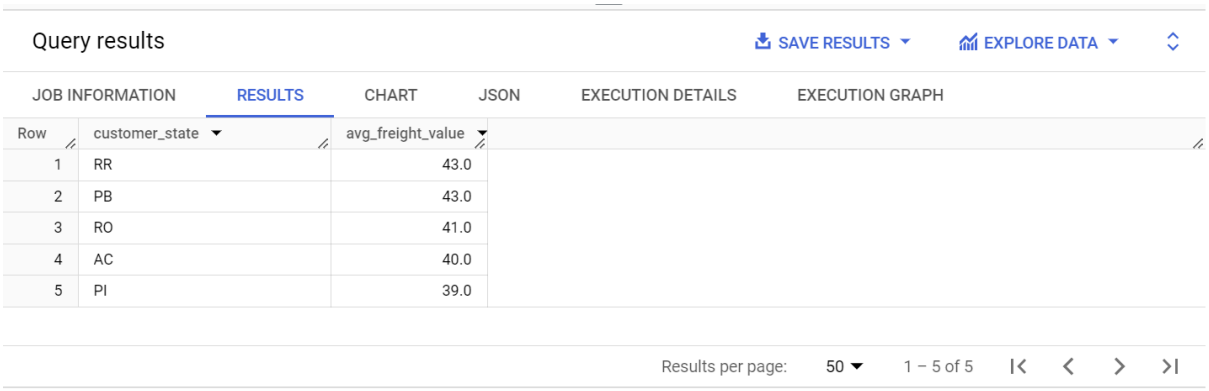
Code:

```
create view target.top_5_freight as (select b.customer_state, avg(c.freight_value)
as avg_freight_value,
                    row_number() over(order by avg(c.freight_value) desc) as rnk1,
```

```
                    row_number() over(order by avg(c.freight_value) asc) as rnk2
from `target.orders` a
join `target.customers` b
on a.customer_id = b.customer_id
join `target.order_items` c
on a.order_id = c.order_id
group by 1)

--Top 5 with highest freight value
select customer_state, round(avg_freight_value) as avg_freight_value
from `target.top_5_freight`
where rnk1 <= 5
order by 2 desc;
```

Result:




```
--Last 5 states with lowest freight value
```

Code:

```
select customer_state, round(avg_freight_value) as avg_freight_value
from `target.top_5_freight`
where rnk2 <= 5
order by 2 ;
```

Result:

Query results

SAVE RESULTS | EXPLORE DATA

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

Row	customer_state	avg_freight_value
1	SP	15.0
2	PR	21.0
3	MG	21.0
4	RJ	21.0
5	DF	21.0

Results per page: 50 | 1 – 5 of 5

```
--5.3 Find out the top 5 states with the highest & lowest average delivery time.
--Highest average
```

Code:

```
with cte as (select    b.customer_state,
             date_diff(order_delivered_customer_date,order_purchase_timestamp,day)
as delivery_time
             from      `target.orders` a
             join      `target.customers` b
             on        a.customer_id = b.customer_id),
cte2 as (select customer_state, round(avg(delivery_time),2) as avg_delivery_time,
               row_number() over(order by avg(cte.delivery_time) desc) as rnk2
         from cte
         group by 1)
select customer_state,avg_delivery_time
from cte2
where rnk2<=5
order by 2;
```

Result:

Query results

SAVE RESULTS | EXPLORE DATA

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

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

Results per page: 50 | 1 – 5 of 5

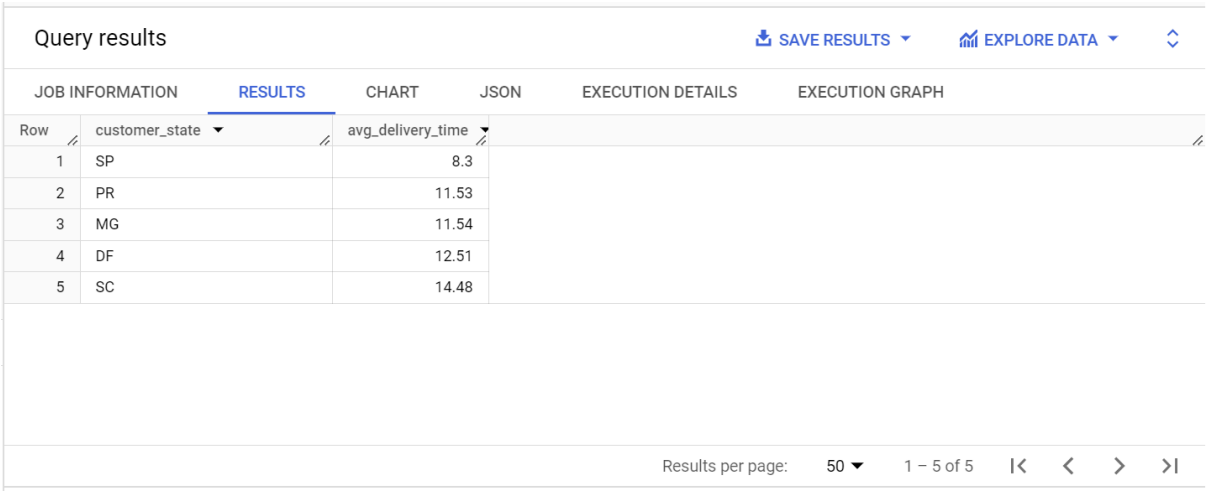
```
--Lowest average
```

Code:

```
with cte as (select    b.customer_state,
```

```
            date_diff(order_delivered_customer_date,order_purchase_timestamp,day)
as delivery_time
            from      `target.orders` a
            join      `target.customers` b
            on        a.customer_id = b.customer_id),
cte2 as (select customer_state, round(avg(delivery_time),2) as avg_delivery_time,
              row_number() over(order by avg(cte.delivery_time)) as rnk1
         from cte
         group by 1)
select customer_state,avg_delivery_time
from cte2
where rnk1<=5
order by 2;
```

Result:




```
--5.4 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.*/
```

Code:

```
with cte as (select     b.customer_state,

date_diff(order_delivered_customer_date,order_purchase_timestamp,day) as
delivery_time,

date_diff(order_estimated_delivery_date,order_purchase_timestamp,day) as
est_delivery_time
              from      `target.orders` a
              join      `target.customers` b
              on        a.customer_id = b.customer_id
              where     a.order_delivered_customer_date is not null),
cte2 as (select customer_state,
                round(avg(delivery_time),2) as avg_delivery_time,
                round(avg(est_delivery_time),2) as avg_est_delivery_time
          from  cte
          group by 1)
select  customer_state,round((avg_est_delivery_time - avg_delivery_time)) as
avg_time_difference
from    cte2
order by 2 desc
limit 5;
```

Result:

Query results

SAVE RESULTS | EXPLORE DATA

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

Row	customer_state	avg_time_difference
1	AC	20.0
2	AP	19.0
3	RO	19.0
4	AM	19.0
5	RR	17.0

Results per page: 50 | 1 – 5 of 5 | Updated 3 days ago

```
--6. Analysis based on the payments:
--6.1 Find the month on month no. of orders placed using different payment types.
```

Code:

```
select    extract(year from order_purchase_timestamp) as year_purchased,
          extract(month from order_purchase_timestamp) as month_purchased,
          b.payment_type,
          count(a.order_id) as total_orders
```

```
from      `target.orders` a
join      `target.payments` b
on        b.order_id = a.order_id
group by  1,2,3
order by  1,2 asc, 4 desc;
```

Result:

Query results

SAVE RESULTS | EXPLORE DATA

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

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

Results per page: 50 | 1 – 50 of 90

```
--6.2 Find the no. of orders placed on the basis of the payment installments that have been paid.
--We want you to count the no. of orders placed based on the no. of payment installments where at least one installment has been successfully paid.
```

Code:

```
select    payment_installments,count(order_id) as total_orders
from      `target.payments`
where     payment_sequential >=1
group by 1;
```

Result:

Query results

SAVE RESULTS | EXPLORE DATA

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

Row	payment_installment	total_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

Results per page: 50 | 1 – 24 of 24

Insights:

1. Highest no.of customers are in the state SP with a total of 41746 customers which is approximately 900 times the customers from the state RR with the least no.of customers.
2. Total basket value is also high in SP state with 590 times the Total basket value of RR state.
3. Maximum orders received are observed to be placed in the afternoon and the least in the dawn.
4. YOY change in the total orders placed sharply increased from 2016 to 2017 and slight growth can be seen from 2017 to 2018.
5. % increase in the cost of orders from 2017 to 2018 is almost 137.
6. States with least average freight value and average delivery time are receiving maximum number of orders, Reducing freight value and delivery time might result in an increase in the number of orders from the states like RR with least number of orders placed.
7. Payments made through credit cards are maximum in all the year.