Q1 What does 'good' look like?

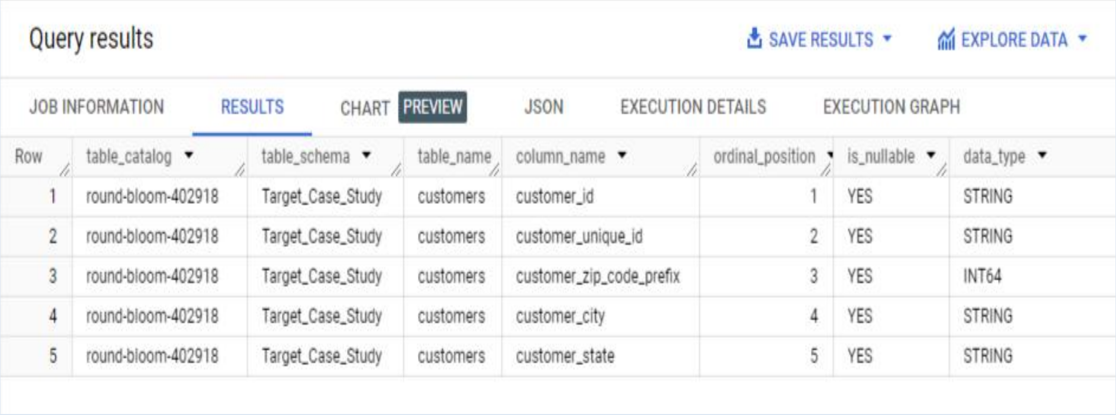
Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset:

1. Data type of all columns in the "customers" table.

Query :

```
select * from round-bloom-402918.Target_Case_Study.INFORMATION_SCHEMA.COLUMNS
where TABLE_NAME='customers'
```

Output :



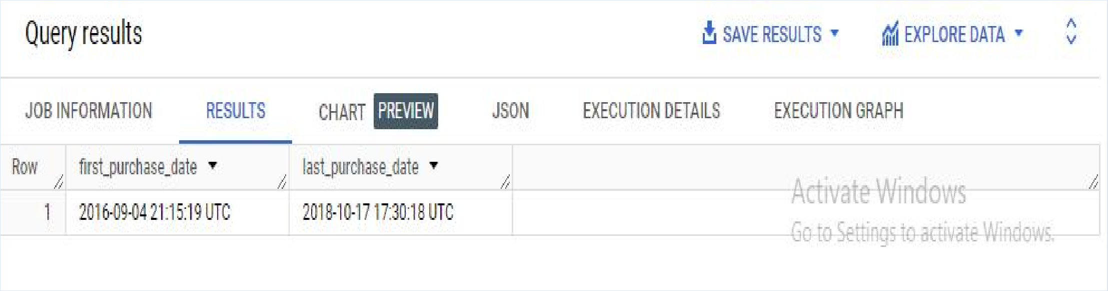
Row	table_catalog	table_schema	table_name	column_name	ordinal_position	is_nullable	data_type
1	round-bloom-402918	Target_Case_Study	customers	customer_id	1	YES	STRING
2	round-bloom-402918	Target_Case_Study	customers	customer_unique_id	2	YES	STRING
3	round-bloom-402918	Target_Case_Study	customers	customer_zip_code_prefix	3	YES	INT64
4	round-bloom-402918	Target_Case_Study	customers	customer_city	4	YES	STRING
5	round-bloom-402918	Target_Case_Study	customers	customer_state	5	YES	STRING

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

Query :

```
select min(order_purchase_timestamp) as first_purchase_date,
max(order_purchase_timestamp) as last_purchase_date from round-bloom-
402918.Target_Case_Study.orders
```

Output :



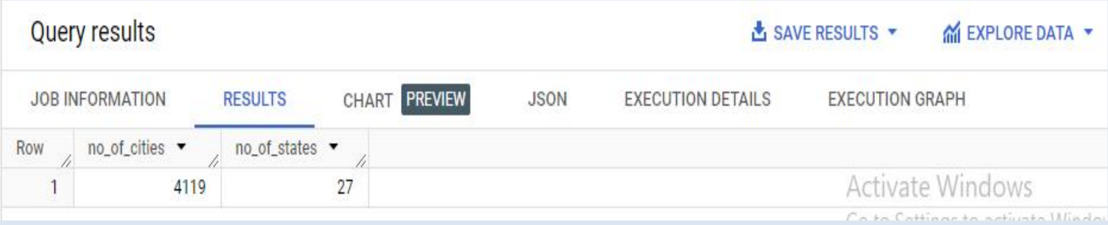
Row	first_purchase_date	last_purchase_date
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC

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

Query :

```
select count(distinct c.customer_city) as no_of_cities, count(distinct
c.customer_state) as no_of_states
from round-bloom-402918.Target_Case_Study.orders o
join round-bloom-402918.Target_Case_Study.customers c
on o.customer_id=c.customer_id
where o.order_purchase_timestamp between '2016-09-04 21:15:19 UTC' and '2018-10-17
17:30:18 UTC'
```

Output :



Row	no_of_cities	no_of_states
1	4119	27

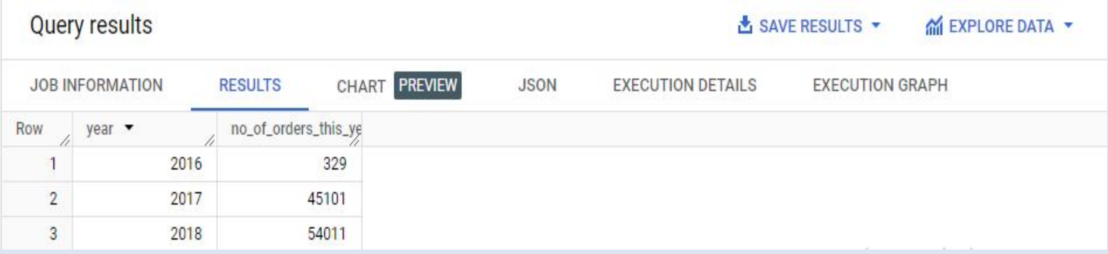
2. In-depth Exploration:

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

Query :

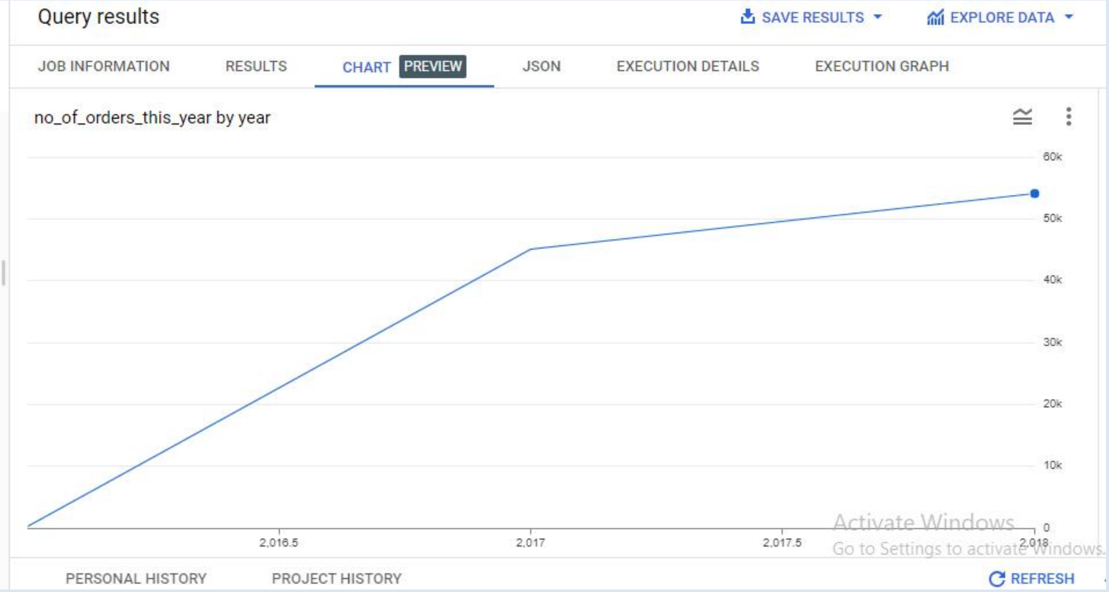
```
select distinct extract(year from order_purchase_timestamp) as year,
count(order_id) over(partition by extract(year from order_purchase_timestamp)) as
no_of_orders_this_year
from round-bloom-402918.Target_Case_Study.orders
order by year
```

Output :



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

Chart :



Insight :

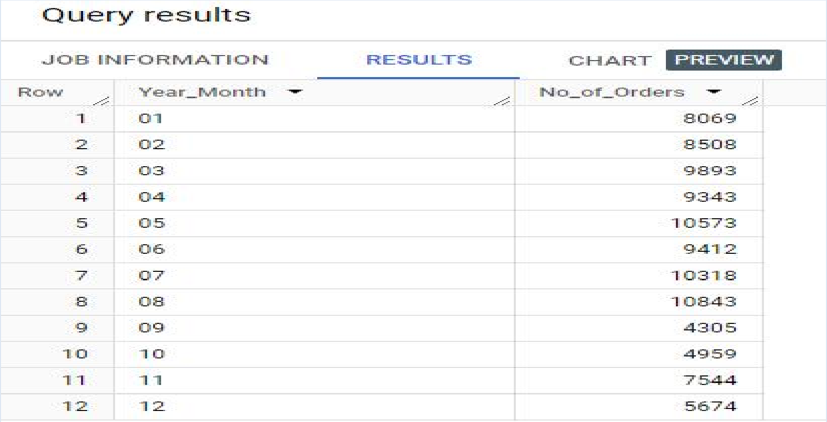
Yes, there is a growing trend as can be seen from numbers and steady upward trajectory of the line chart.

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

Query :

```
select format_date('%m', order_purchase_timestamp) as Month,
count(order_id) as No_of_Orders
from round-bloom-402918.Target_Case_Study.orders
group by Month
order by Month
```

Output :



Query results

JOB INFORMATION | RESULTS | CHART PREVIEW

Row	Year_Month	No_of_Orders
1	01	8069
2	02	8508
3	03	9893
4	04	9343
5	05	10573
6	06	9412
7	07	10318
8	08	10843
9	09	4305
10	10	4959
11	11	7544
12	12	5674

Insights :

As we can see from the bar graph the number of orders keeps picking up in the first half of the year and then a significant drop happens after the month of August where the orders are at peak levels. Orders drop consecutively in next two months of Sept and Oct by a huge margin from peak values. Nov and Dec show some recovery.

Chart :

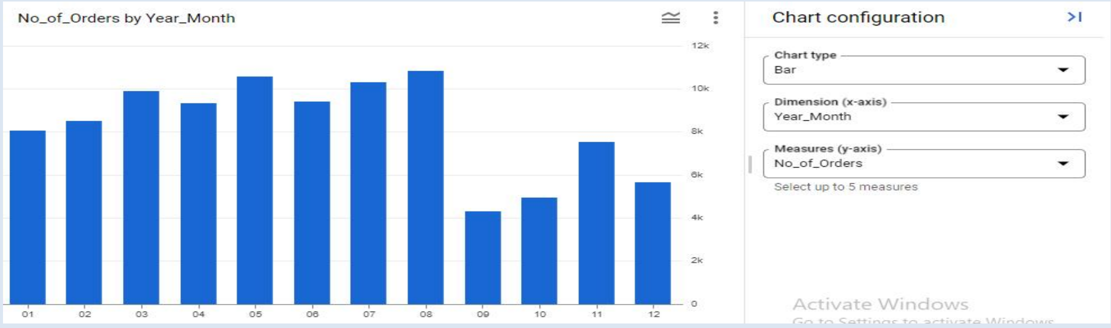


Chart configuration

Chart type: Bar

Dimension (x-axis): Year_Month

Measures (y-axis): No_of_Orders

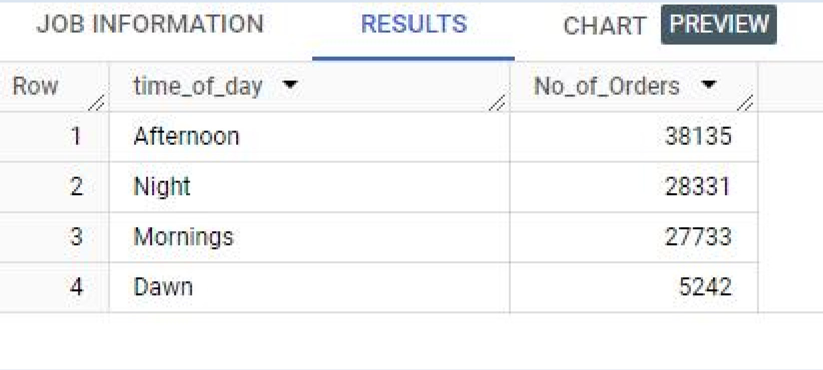
Select up to 5 measures

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
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(distinct order_id) as No_of_Orders
from round-bloom-402918.Target_Case_Study.orders
group by time_of_day
order by No_of_Orders desc
```

Output :



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

Chart :

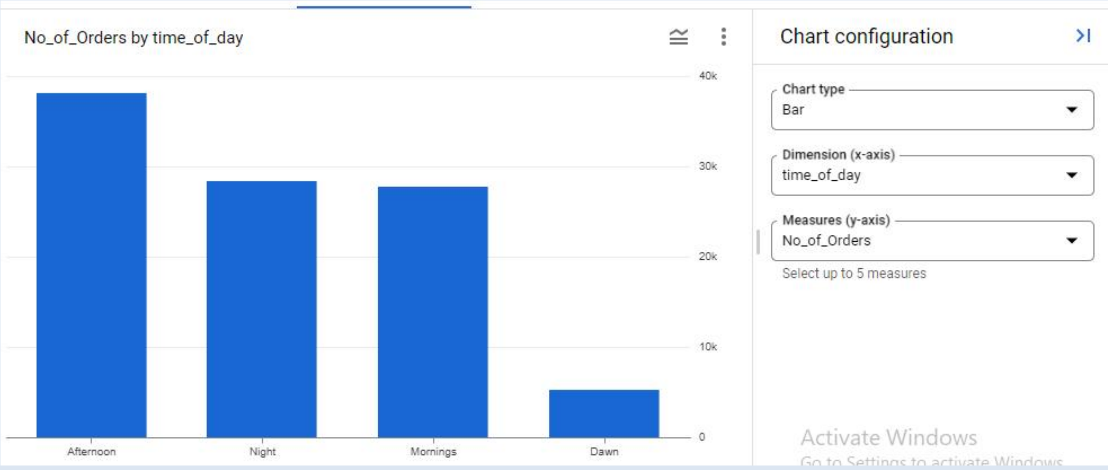


Chart configuration

Chart type: Bar

Dimension (x-axis): time_of_day

Measures (y-axis): No_of_Orders

Select up to 5 measures

Insights:

As we can clearly see from the numbers and the bar graph that most orders were placed by the Brazilian customers during the afternoon period(13-18 hrs) and least during Dawn(0-6 hrs).

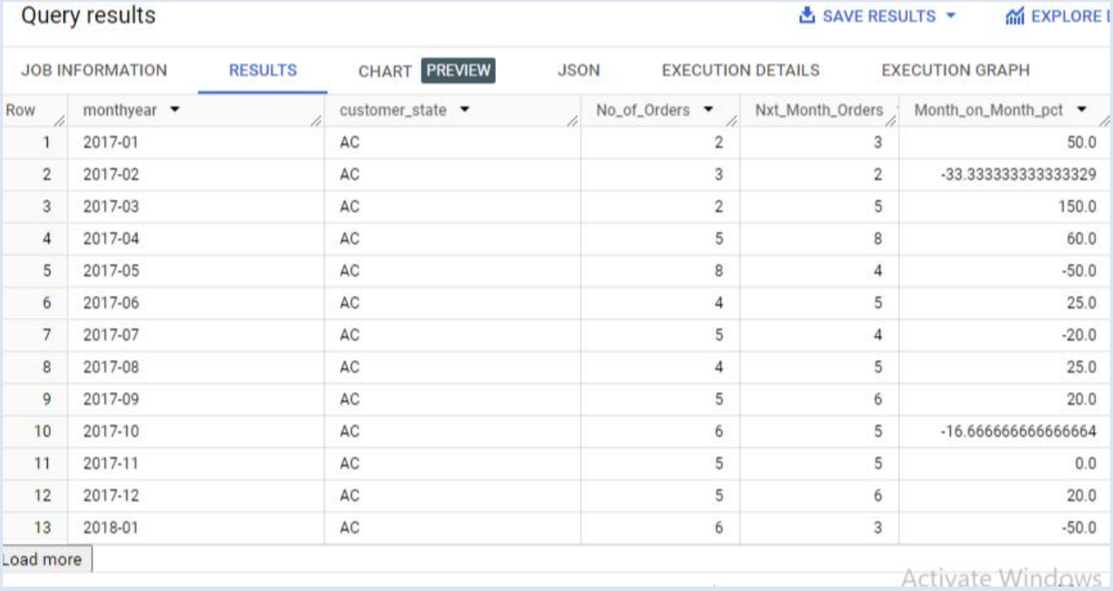
3. Evolution of E-commerce orders in the Brazil region:

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

Query :

```
WITH MonthlyOrders AS (
  SELECT
    FORMAT_TIMESTAMP('%Y-%m', o.order_purchase_timestamp) AS monthyear,
    c.customer_state,
    COUNT(o.order_id) AS No_of_Orders
  FROM
    round-bloom-402918.Target_Case_Study.orders o
    join round-bloom-402918.Target_Case_Study.customers c
    on o.customer_id=c.customer_id
  GROUP BY
    monthyear, c.customer_state
)
SELECT
  monthyear,
  customer_state,
  No_of_Orders,
  LEAD(No_of_Orders) OVER (PARTITION BY customer_state ORDER BY monthyear) AS
Nxt_Month_Orders,
  SAFE_DIVIDE(LEAD(No_of_Orders) OVER (PARTITION BY customer_state ORDER BY
monthyear) - No_of_Orders, No_of_Orders) * 100 AS Month_on_Month_pct
FROM
  MonthlyOrders
ORDER BY
  customer_state, monthyear;
```

Output :



Row	monthyear	customer_state	No_of_Orders	Nxt_Month_Orders	Month_on_Month_pct
1	2017-01	AC	2	3	50.0
2	2017-02	AC	3	2	-33.333333333333329
3	2017-03	AC	2	5	150.0
4	2017-04	AC	5	8	60.0
5	2017-05	AC	8	4	-50.0
6	2017-06	AC	4	5	25.0
7	2017-07	AC	5	4	-20.0
8	2017-08	AC	4	5	25.0
9	2017-09	AC	5	6	20.0
10	2017-10	AC	6	5	-16.666666666666664
11	2017-11	AC	5	5	0.0
12	2017-12	AC	5	6	20.0
13	2018-01	AC	6	3	-50.0

Insights :

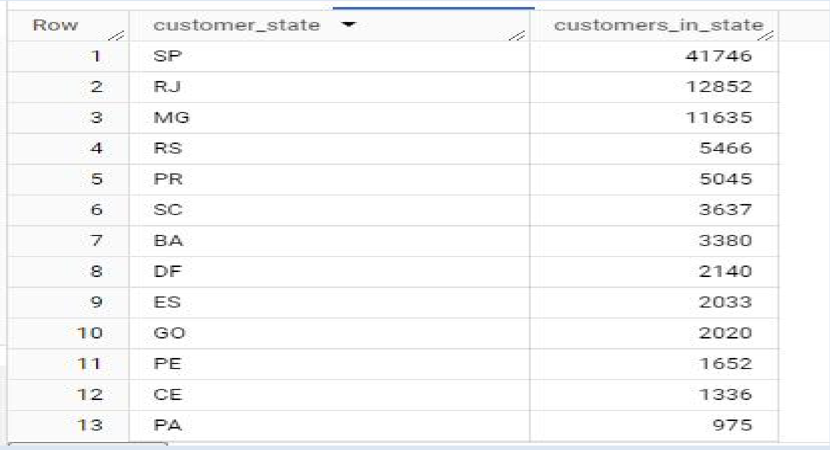
Shows state-wise monthly orders and next-month’s orders to calculate month-on-month growth or decline in the sales. Negative indicates decline in sales and positive month-on-month % indicates growth in sales.

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

Query :

```
select c.customer_state, count(c.customer_id) as customers_in_state
from round-bloom-402918.Target_Case_Study.customers c
join round-bloom-402918.Target_Case_Study.orders o
on c.customer_id=o.customer_id
group by c.customer_state
order by customers_in_state desc
```

Output :



Row	customer_state	customers_in_state
1	SP	41746
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
11	PE	1652
12	CE	1336
13	PA	975

Insights:

SP has highest number of customers with 41746 and RR has the lowest with 46.

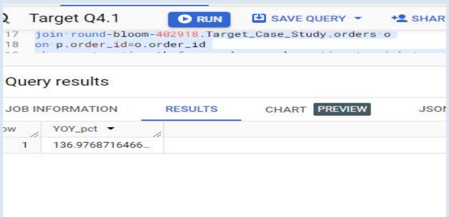
4. Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight and others.

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 :

```
with cte1 as(
select sum(p.payment_value) as cost_of_orders_2017, extract(year from
o.order_purchase_timestamp) as previous_year
from round-bloom-402918.Target_Case_Study.payments p
join round-bloom-402918.Target_Case_Study.orders o
on p.order_id=o.order_id
where extract(month from order_purchase_timestamp) between 01 and 08
and extract(year from order_purchase_timestamp) = 2017
group by extract(year from order_purchase_timestamp)
),
cte2 as(
select sum(p.payment_value) as cost_of_orders_2018, extract(year from
order_purchase_timestamp) as current_year
from round-bloom-402918.Target_Case_Study.payments p
join round-bloom-402918.Target_Case_Study.orders o
on p.order_id=o.order_id
where extract(month from order_purchase_timestamp) between 01 and 08
and extract(year from order_purchase_timestamp) = 2018
group by extract(year from order_purchase_timestamp)
)
select ((cte2.cost_of_orders_2018 -
cte1.cost_of_orders_2017)/cte1.cost_of_orders_2017)*100 as YOY_pct
from cte1
join cte2
on cte1.previous_year+1=cte2.current_year
```

Output :



```
join round-bloom-402918.Target_Case_Study.orders o
on p.order_id=o.order_id
```

Query results

ow	YOY_pct
1	136.9768716466...

Insights:

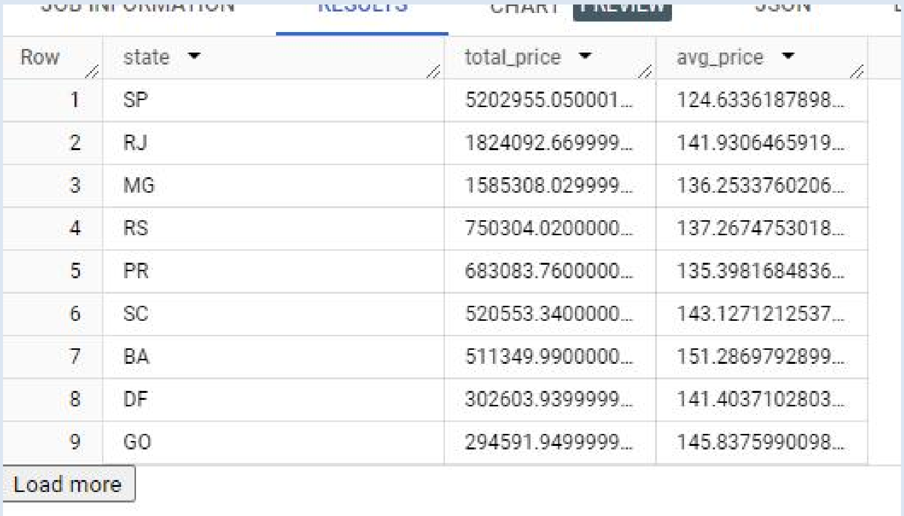
There is a growth in cost of orders from 2017 of approx. 137% in the year 2018.

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

Query :

```
with cte1 as (
select
c.customer_state as state,
sum(price) as total_price,
count(distinct o.order_id) as No_of_Orders
from round-bloom-402918.Target_Case_Study.customers c
left join round-bloom-402918.Target_Case_Study.orders o
on c.customer_id=o.customer_id
left join round-bloom-402918.Target_Case_Study.order_items oi
on o.order_id=oi.order_id
group by state
)
select cte1.state,
cte1.total_price,
(cte1.total_price/cte1.No_of_Orders) as avg_price
from cte1
order by cte1.total_price desc
```

Output :



Row	state	total_price	avg_price
1	SP	5202955.050001…	124.6336187898…
2	RJ	1824092.669999…	141.9306465919…
3	MG	1585308.029999…	136.2533760206…
4	RS	750304.0200000…	137.2674753018…
5	PR	683083.7600000…	135.3981684836…
6	SC	520553.3400000…	143.1271212537…
7	BA	511349.9900000…	151.2869792899…
8	DF	302603.9399999…	141.4037102803…
9	GO	294591.9499999…	145.8375990098…

Chart :

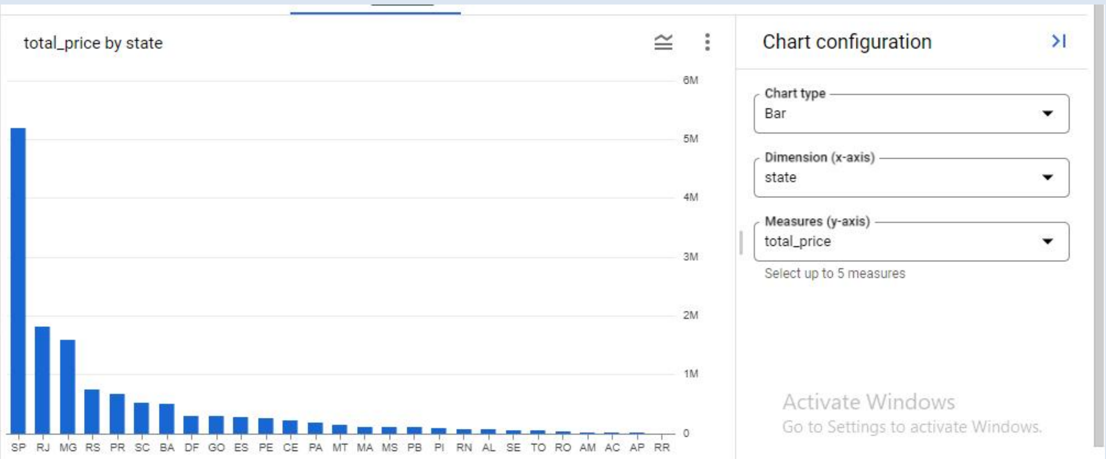

total_price by state
6M
5M
4M
3M
2M
1M
0
SP RJ MG RS PR SC BA DF GO ES PE CE PA MT MA MS PB PI RN AL SE TO RO AM AC AP RR
Chart configuration
Activate Windows
Go to Settings to activate Windows.

Chart type: Bar

Dimension (x-axis): state

Measures (y-axis): total_price

Select up to 5 measures

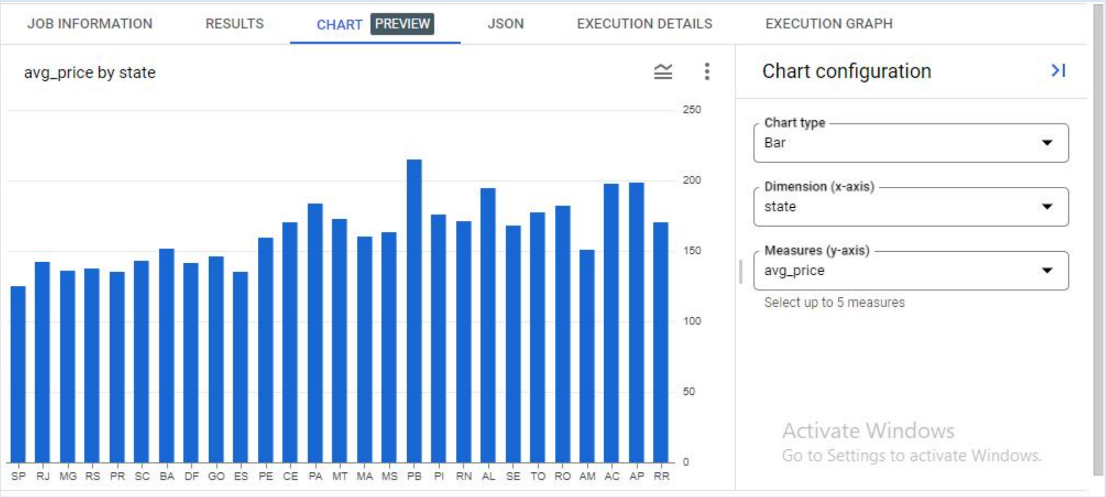


Chart type: Bar

Dimension (x-axis): state

Measures (y-axis): avg_price

Select up to 5 measures

Insights :

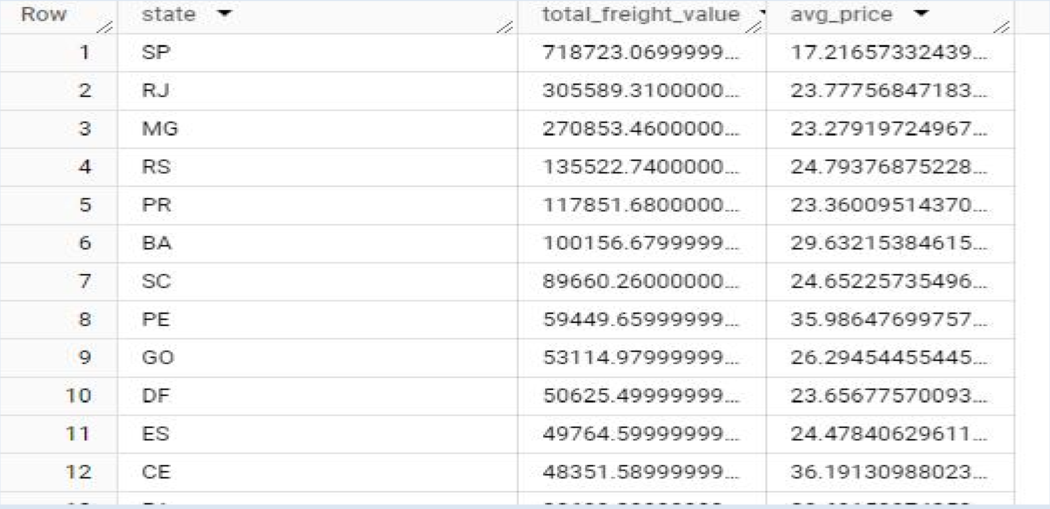
SP has the highest total order price at 5202955.05 and RR has the lowest total at 7829.43.
But SP has the lowest average order price at 124.63 while PB has the highest average at 215.05.

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

Query :

```
with cte1 as (
select
c.customer_state as state,
sum(freight_value) as total_freight_value,
count(distinct o.order_id) as No_of_Orders
from round-bloom-402918.Target_Case_Study.customers c
left join round-bloom-402918.Target_Case_Study.orders o
on c.customer_id=o.customer_id
left join round-bloom-402918.Target_Case_Study.order_items oi
on o.order_id=oi.order_id
group by state
)
select cte1.state,
cte1.total_freight_value,
(cte1.total_freight_value/cte1.No_of_Orders) as avg_price
from cte1
order by cte1.total_freight_value desc
```

Output :



Row	state	total_freight_value	avg_price
1	SP	718723.0699999...	17.21657332439...
2	RJ	305589.3100000...	23.77756847183...
3	MG	270853.4600000...	23.27919724967...
4	RS	135522.7400000...	24.79376875228...
5	PR	117851.6800000...	23.36009514370...
6	BA	100156.6799999...	29.63215384615...
7	SC	89660.26000000...	24.65225735496...
8	PE	59449.65999999...	35.98647699757...
9	GO	53114.97999999...	26.29454455445...
10	DF	50625.49999999...	23.65677570093...
11	ES	49764.59999999...	24.47840629611...
12	CE	48351.58999999...	36.19130988023...

Charts :

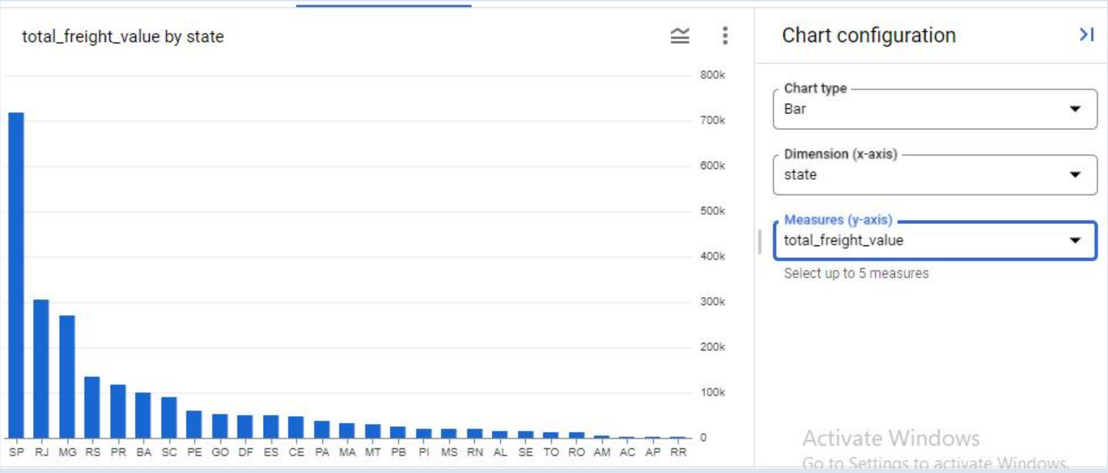

total_freight_value by state
800k
700k
600k
500k
400k
300k
200k
100k
0
SP RJ MG RS PR BA SC PE GO DF ES CE PA MA MT PB PI MS RN AL SE TO RO AM AC AP RR
Activate Windows
Go to Settings to activate Windows.

Chart configuration

Chart type: Bar

Dimension (x-axis): state

Measures (y-axis): total_freight_value

Select up to 5 measures

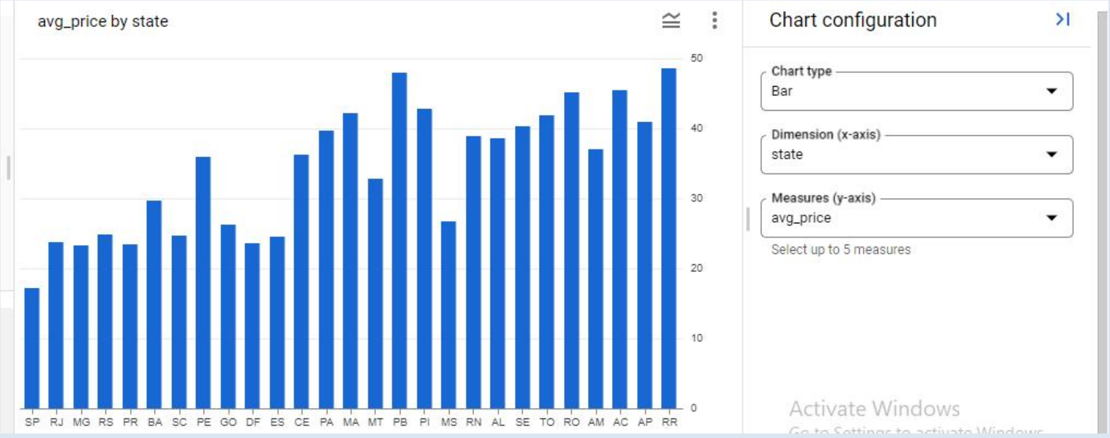

avg_price by state
50
40
30
20
10
0
SP RJ MG RS PR BA SC PE GO DF ES CE PA MA MT PB PI MS RN AL SE TO RO AM AC AP RR
Activate Windows
Go to Settings to activate Windows.

Chart configuration

Chart type: Bar

Dimension (x-axis): state

Measures (y-axis): avg_price

Select up to 5 measures

Insights:

SP has the highest total freight value at 718723.07 and RR has the lowest total at 2235.19.
But RR has the highest average freight value at 48.591 while SP has the lowest average at 17.21.

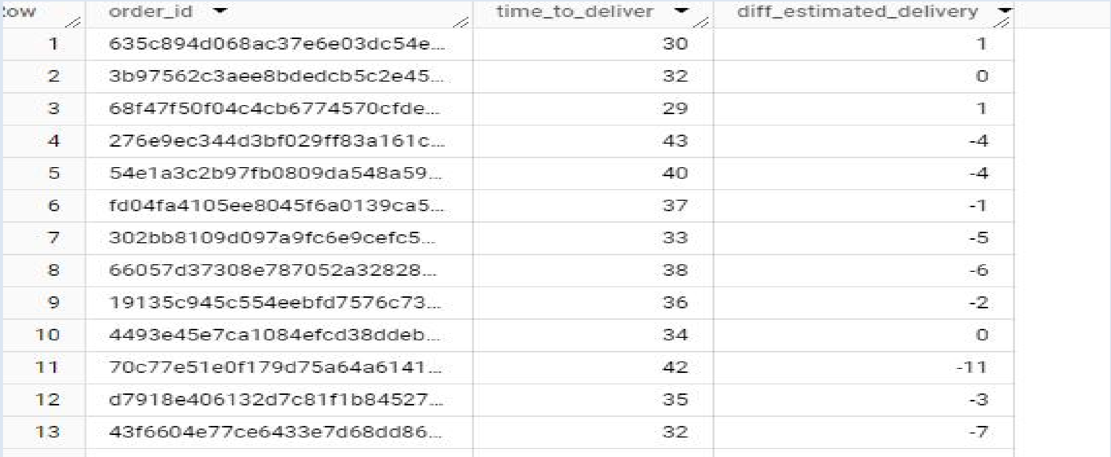
5. Analysis based on sales, freight and delivery time.

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.

Query :

```
select order_id, timestamp_diff( order_delivered_customer_date,
order_purchase_timestamp, day) as time_to_deliver,
timestamp_diff(order_estimated_delivery_date, order_delivered_customer_date, day)
as diff_estimated_delivery
from round-bloom-402918.Target_Case_Study.orders
where order_status = 'delivered'
```

Output :



:ow	order_id	time_to_deliver	diff_estimated_delivery
1	635c894d068ac37e6e03dc54e...	30	1
2	3b97562c3aee8bdedcb5c2e45...	32	0
3	68f47f50f04c4cb6774570cfde...	29	1
4	276e9ec344d3bf029ff83a161c...	43	-4
5	54e1a3c2b97fb0809da548a59...	40	-4
6	fd04fa4105ee8045f6a0139ca5...	37	-1
7	302bb8109d097a9fc6e9cefc5...	33	-5
8	66057d37308e787052a32828...	38	-6
9	19135c945c554eebfd7576c73...	36	-2
10	4493e45e7ca1084efcd38ddeb...	34	0
11	70c77e51e0f179d75a64a6141...	42	-11
12	d7918e406132d7c81f1b84527...	35	-3
13	43f6604e77ce6433e7d68dd86...	32	-7

Insights:

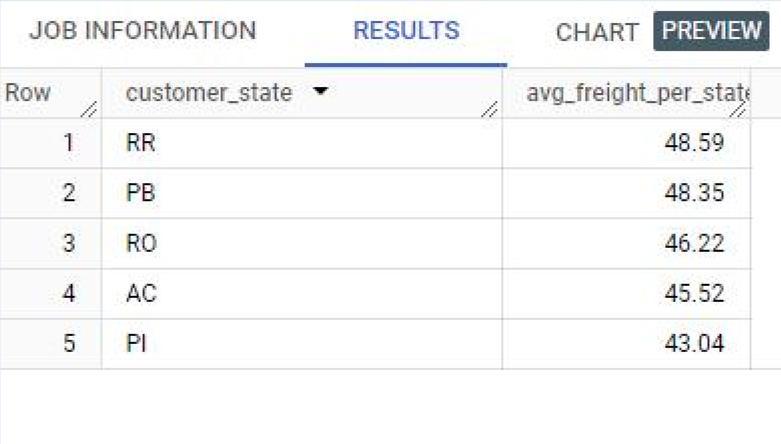
The time_to_deliver mentions time taken for order to be delivered to the customer's place as we can see in this output, mostly around the month mark. Diff_estimated_delivery mentions the number of days by the orders deviated from expected delivery dates where the negative values indicate late delivery.

2. i) Find out the top 5 states with the highest average freight value.

Query:

```
select c.customer_state, round(((sum(oi.freight_value))/(count(distinct
o.order_id))),2) as avg_freight_per_state
from round-bloom-402918.Target_Case_Study.order_items oi
join round-bloom-402918.Target_Case_Study.orders o
on oi.order_id=o.order_id
join round-bloom-402918.Target_Case_Study.customers c
on c.customer_id=o.customer_id
group by c.customer_state
order by avg_freight_per_state desc
limit 5
```

Output:



Row	customer_state	avg_freight_per_stat
1	RR	48.59
2	PB	48.35
3	RO	46.22
4	AC	45.52
5	PI	43.04

Insights:

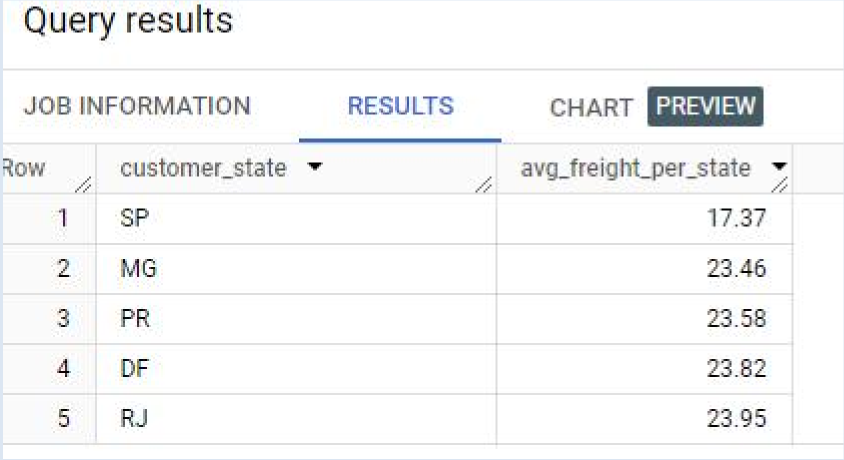
RR, PB, RO, AC and PI are the top 5 states with the highest average freight values upwards of 43.

2. ii) Find out the top 5 states with the lowest average freight value.

Query:

```
select c.customer_state, round(((sum(oi.freight_value))/(count(distinct
o.order_id))),2) as avg_freight_per_state
from round-bloom-402918.Target_Case_Study.order_items oi
join round-bloom-402918.Target_Case_Study.orders o
on oi.order_id=o.order_id
join round-bloom-402918.Target_Case_Study.customers c
on c.customer_id=o.customer_id
group by c.customer_state
order by avg_freight_per_state
limit 5
```

Output:



Row	customer_state	avg_freight_per_state
1	SP	17.37
2	MG	23.46
3	PR	23.58
4	DF	23.82
5	RJ	23.95

Insights:

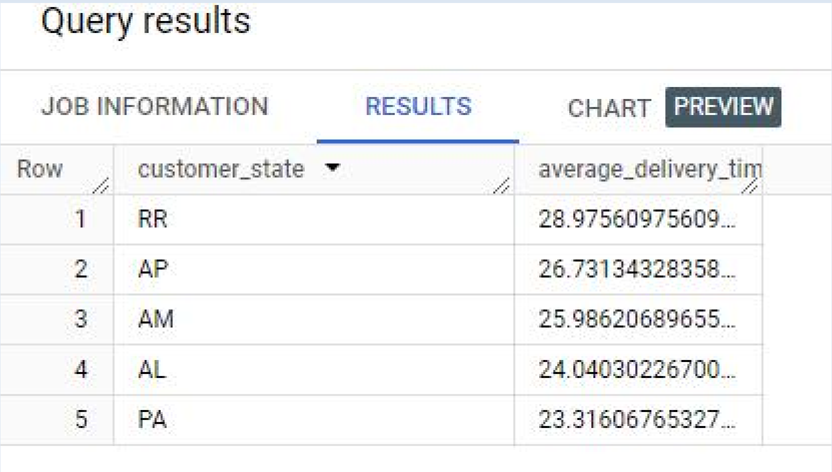
SP, MG, PR, DF and RJ are the states with the lowest average freight values in the range of 17 to 24.

3. i) Find out the top 5 states with the highest average delivery time.

Query:

```
select t.customer_state, sum(t.delivery_time)/count(distinct t.order_id) as
average_delivery_time
from
( select date_diff( o.order_delivered_customer_date, o.order_purchase_timestamp,
day) as delivery_time,
c.customer_state, o.order_id
from round-bloom-402918.Target_Case_Study.orders o
join round-bloom-402918.Target_Case_Study.customers c
on c.customer_id=o.customer_id
where o.order_status='delivered')t
group by t.customer_state
order by average_delivery_time desc
limit 5
```

Output:



Row	customer_state	average_delivery_tim
1	RR	28.97560975609...
2	AP	26.73134328358...
3	AM	25.98620689655...
4	AL	24.04030226700...
5	PA	23.31606765327...

Insights:

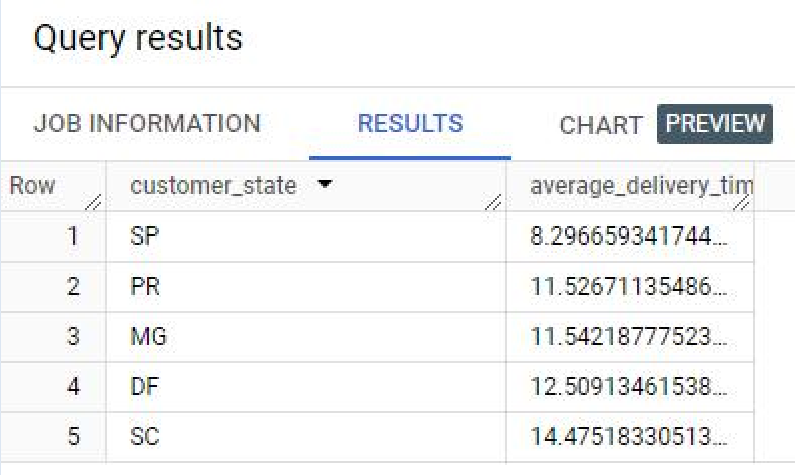
RR, AP, AM, AL and PA are the states with the highest average delivery time which is in the range of 23 to 29 days.

3. ii) Find out the top 5 states with the lowest average delivery time.

Query:

```
select t.customer_state, sum(t.delivery_time)/count(distinct t.order_id) as
average_delivery_time
from
( select date_diff( o.order_delivered_customer_date, o.order_purchase_timestamp,
day) as delivery_time,
c.customer_state, o.order_id
from round-bloom-402918.Target_Case_Study.orders o
join round-bloom-402918.Target_Case_Study.customers c
on c.customer_id=o.customer_id
where o.order_status='delivered')t
group by t.customer_state
order by average_delivery_time
limit 5
```

Output:



Row	customer_state	average_delivery_tim
1	SP	8.296659341744...
2	PR	11.52671135486...
3	MG	11.54218777523...
4	DF	12.50913461538...
5	SC	14.47518330513...

Insights:

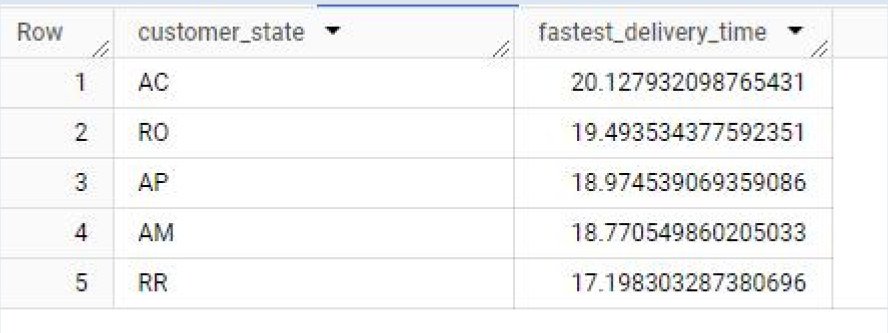
SP, PR, MG, DF and SC are the states with the lowest average delivery time which is in the range of 8 to 15 days.

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:

```
with cte1 as (
select t.customer_state, sum(t.delivery_time)/count(distinct t.order_id) as
average_delivery_time
from
( select date_diff( o.order_delivered_customer_date, o.order_purchase_timestamp,
day) as delivery_time,
c.customer_state, o.order_id
from round-bloom-402918.Target_Case_Study.orders o
join round-bloom-402918.Target_Case_Study.customers c
on c.customer_id=o.customer_id
where o.order_status='delivered')t
group by t.customer_state
order by average_delivery_time),
cte2 as
(select t.customer_state, sum(t.estimated_delivery_time)/count(distinct t.order_id)
as average_estimated_delivery_time
from
( select date_diff( o.order_estimated_delivery_date, o.order_purchase_timestamp,
day) as estimated_delivery_time,
c.customer_state, o.order_id
from round-bloom-402918.Target_Case_Study.orders o
join round-bloom-402918.Target_Case_Study.customers c
on c.customer_id=o.customer_id)t
group by t.customer_state
order by average_estimated_delivery_time)
select cte1.customer_state, (cte2.average_estimated_delivery_time -
cte1.average_delivery_time) as fastest_delivery_time
from
cte1
join
cte2
on cte1.customer_state=cte2.customer_state
order by fastest_delivery_time desc
limit 5
```

Output :



Row	customer_state	fastest_delivery_time
1	AC	20.127932098765431
2	RO	19.493534377592351
3	AP	18.974539069359086
4	AM	18.770549860205033
5	RR	17.198303287380696

Chart:

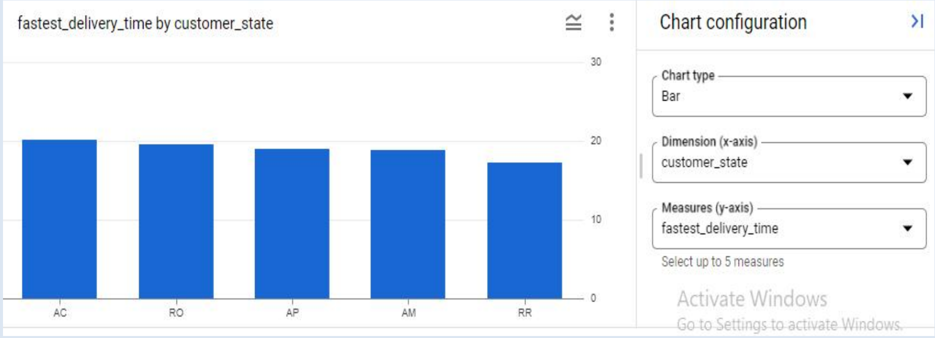


Chart configuration

Chart type: Bar

Dimension (x-axis): customer_state

Measures (y-axis): fastest_delivery_time

Select up to 5 measures

Insights:

AC, RO, AP, AM and RR are the states with the fastest delivery times as compared to the estimated date of delivery.

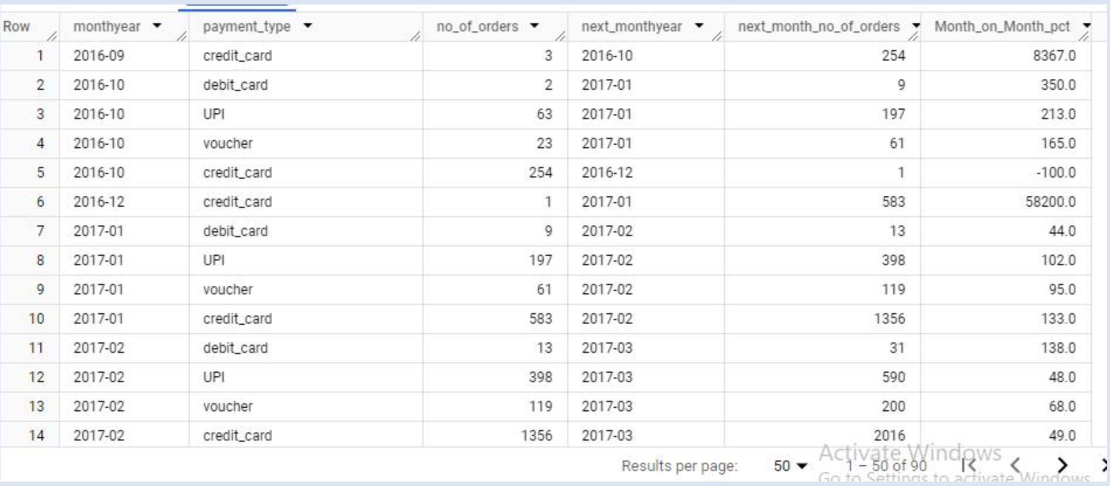
6. Analysis based on the payments:

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

Query:

```
select t.monthyear,
t.payment_type,
count(t.payment_type) as no_of_orders,
lead(t.monthyear) over(partition by t.payment_type order by t.monthyear) as
next_monthyear,
lead(count(t.payment_type),1) over(partition by t.payment_type order by t.monthyear)
as next_month_no_of_orders,
round(safe_divide(lead(count(t.payment_type),1) over(partition by
t.payment_type order by t.monthyear) - count(t.payment_type),
count(t.payment_type)),2) * 100 as Month_on_Month_pct
from
(select
row_number() over (partition by format_timestamp('%Y-%m',order_purchase_timestamp))
as row_number,
format_timestamp('%Y-%m',order_purchase_timestamp) as monthyear,
payment_type
from round-bloom-402918.Target_Case_Study.payments p
join round-bloom-402918.Target_Case_Study.orders o
on p.order_id=o.order_id)t
group by t.monthyear, t.payment_type
order by t.monthyear
```

Output:



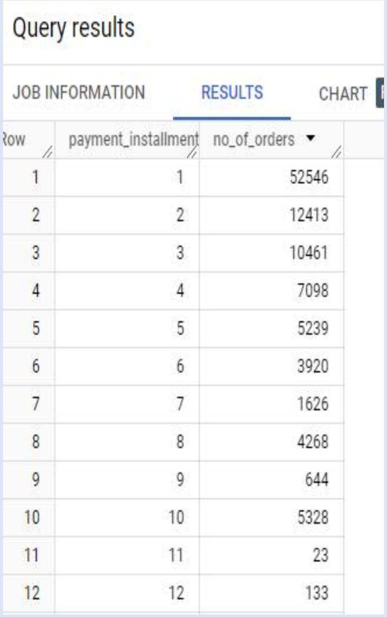
Row	monthyear	payment_type	no_of_orders	next_monthyear	next_month_no_of_orders	Month_on_Month_pct
1	2016-09	credit_card	3	2016-10	254	8367.0
2	2016-10	debit_card	2	2017-01	9	350.0
3	2016-10	UPI	63	2017-01	197	213.0
4	2016-10	voucher	23	2017-01	61	165.0
5	2016-10	credit_card	254	2016-12	1	-100.0
6	2016-12	credit_card	1	2017-01	583	58200.0
7	2017-01	debit_card	9	2017-02	13	44.0
8	2017-01	UPI	197	2017-02	398	102.0
9	2017-01	voucher	61	2017-02	119	95.0
10	2017-01	credit_card	583	2017-02	1356	133.0
11	2017-02	debit_card	13	2017-03	31	138.0
12	2017-02	UPI	398	2017-03	590	48.0
13	2017-02	voucher	119	2017-03	200	68.0
14	2017-02	credit_card	1356	2017-03	2016	49.0

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

Query:

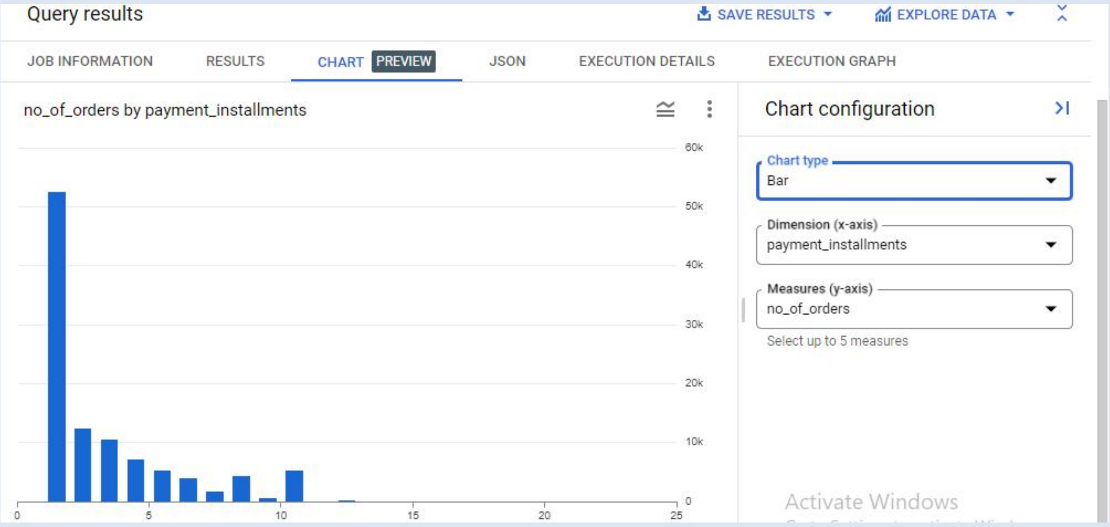
```
select payment_installments, count(p.order_id) as no_of_orders
from round-bloom-402918.Target_Case_Study.payments p
where payment_installments >=1
and exists (
    select 1
    from round-bloom-402918.Target_Case_Study.orders o
    where o.order_id = p.order_id
  )
```

Output:



Row	payment_installment	no_of_orders
1	1	52546
2	2	12413
3	3	10461
4	4	7098
5	5	5239
6	6	3920
7	7	1626
8	8	4268
9	9	644
10	10	5328
11	11	23
12	12	133

Chart:



Query results

no_of_orders by payment_installments

Chart configuration

Chart type: Bar

Dimension (x-axis): payment_installments

Measures (y-axis): no_of_orders

Select up to 5 measures