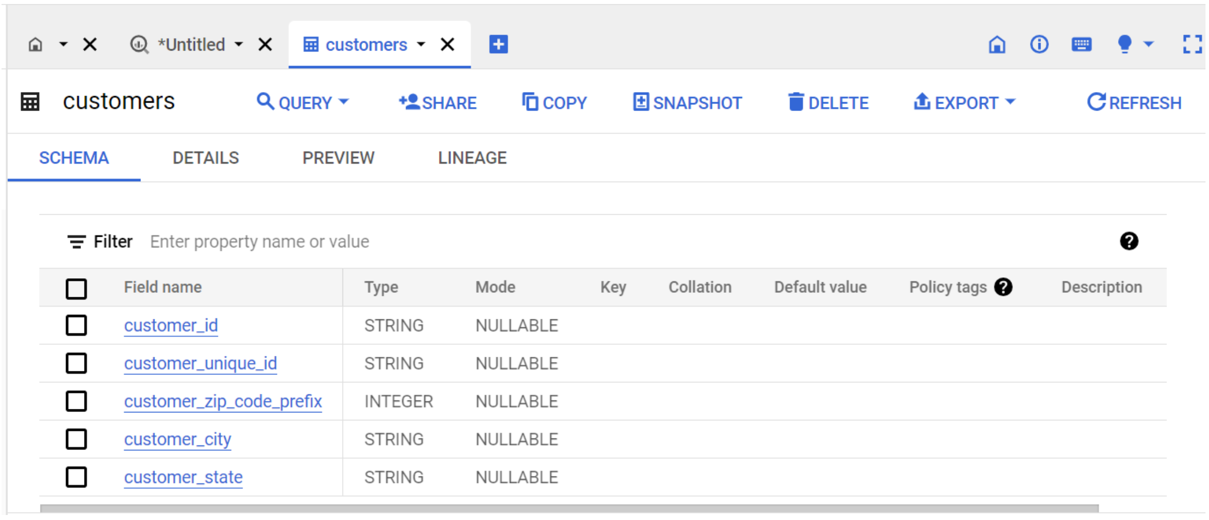
Q1. Business Case: Target SQL

1.1: Data type of all columns in the "customers" table:

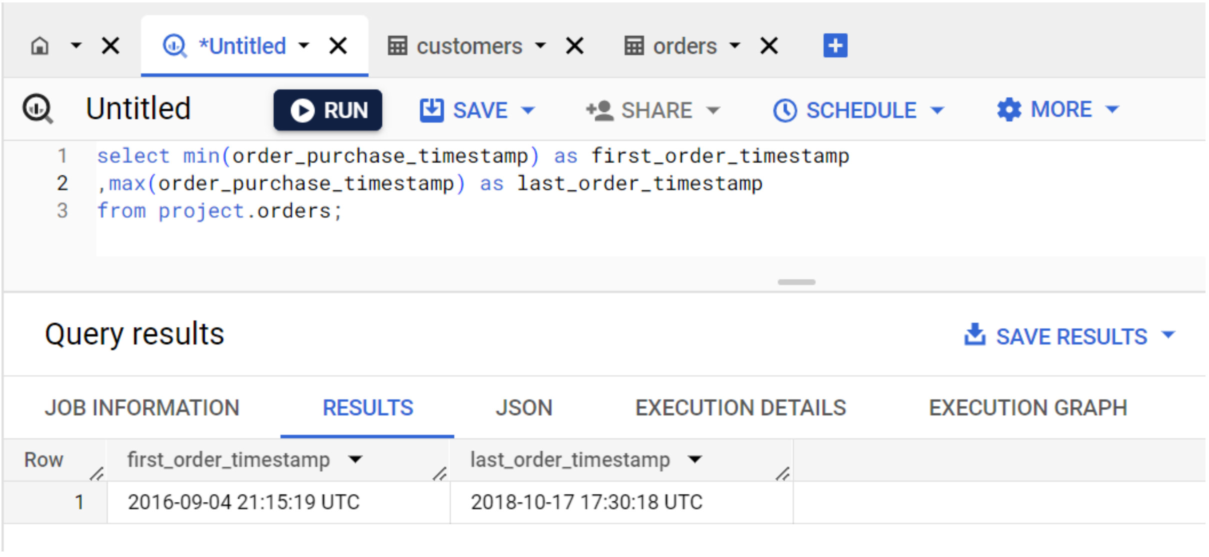
- Query: `` select * from `project.customers`; ``
- We find the data type for any table in properties



Field name	Type	Mode	Key	Collation	Default value	Policy tags	Description
customer_id	STRING	NULLABLE					
customer_unique_id	STRING	NULLABLE					
customer_zip_code_prefix	INTEGER	NULLABLE					
customer_city	STRING	NULLABLE					
customer_state	STRING	NULLABLE					

1.2: Get the time range between which the orders were placed:

- Query: `select min(order_purchase_timestamp) as first_order_timestamp ,max(order_purchase_timestamp) as last_order_date_timestamp from project.orders;`
- As per above query, we could see the time range of orders placed in given in this study is from **September 4th 2016 to October 17th 2018**.

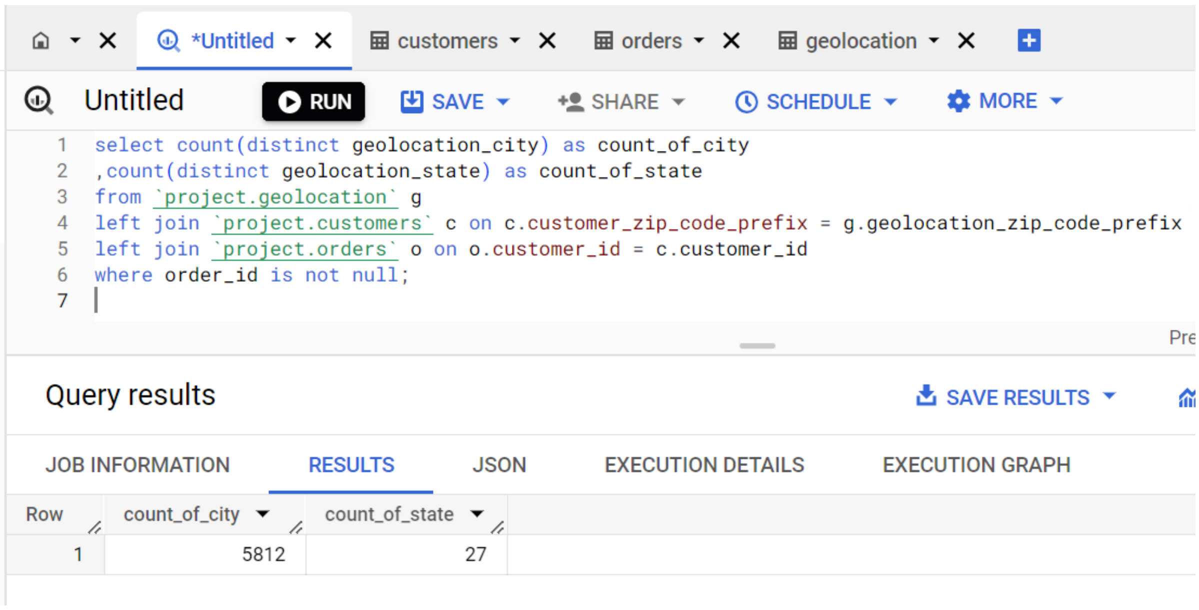


```
select min(order_purchase_timestamp) as first_order_timestamp
,max(order_purchase_timestamp) as last_order_timestamp
from project.orders;
```

```
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 number of Cities and States in our dataset:

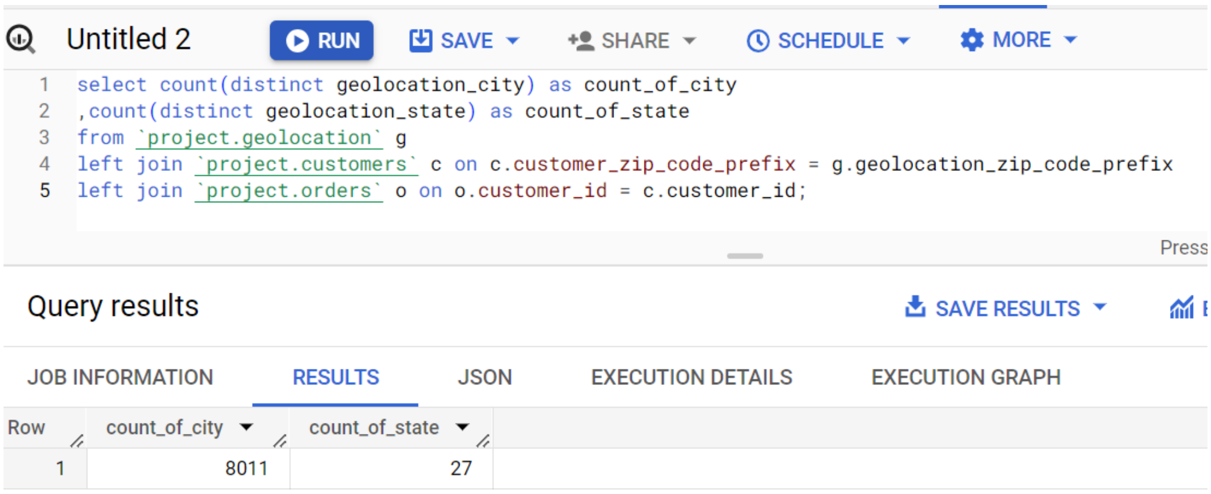
- Query: `select count(distinct geolocation_city) as count_of_city ,count(distinct geolocation_state) as count_of_state from `project.geolocation` g left join `project.customers` c on c.customer_zip_code_prefix = g.geolocation_zip_code_prefix left join `project.orders` o on o.customer_id = c.customer_id where order_id is not null;`
- As per the query, we can see that orders have come from **5812 cities spread across 27 states.**



```
select count(distinct geolocation_city) as count_of_city
,count(distinct geolocation_state) as count_of_state
from `project.geolocation` g
left join `project.customers` c on c.customer_zip_code_prefix = g.geolocation_zip_code_prefix
left join `project.orders` o on o.customer_id = c.customer_id
where order_id is not null;
```

Row	count_of_city	count_of_state
1	5812	27

- We can also conclude that not all registered customers have placed orders. Because if we find out cities and state based on customer information, number of cities are much higher. Query: `select count(distinct geolocation_city) as count_of_city,count(distinct geolocation_state) as count_of_state from `project.geolocation` g left join `project.customers` c on c.customer_zip_code_prefix = g.geolocation_zip_code_prefix left join `project.orders` o on o.customer_id = c.customer_id;`



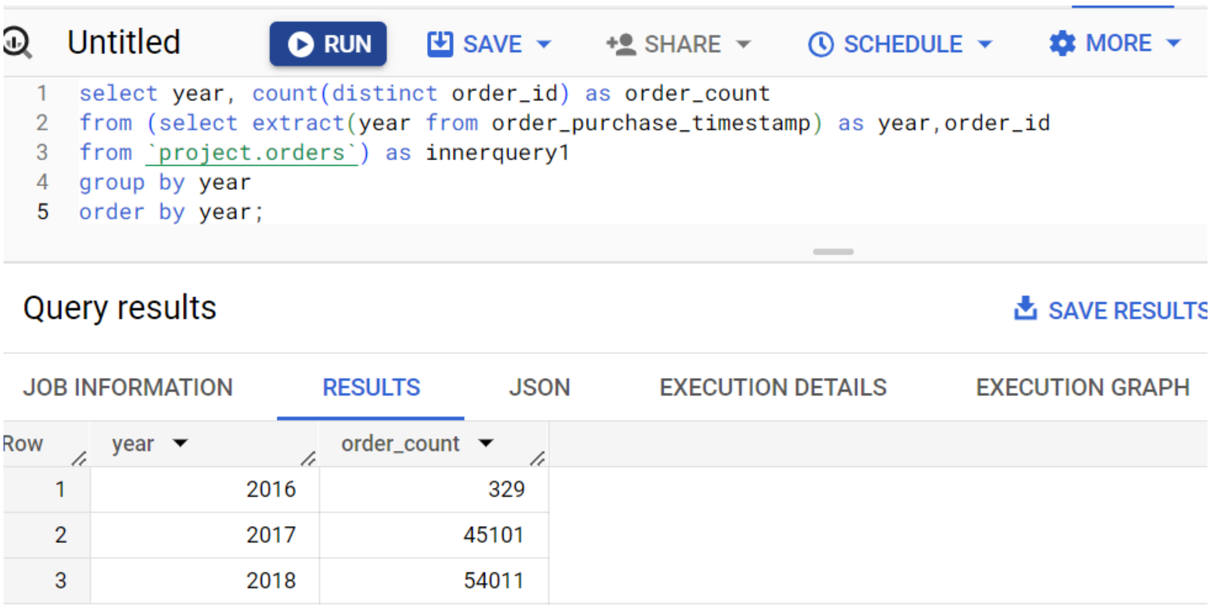
```
select count(distinct geolocation_city) as count_of_city
,count(distinct geolocation_state) as count_of_state
from `project.geolocation` g
left join `project.customers` c on c.customer_zip_code_prefix = g.geolocation_zip_code_prefix
left join `project.orders` o on o.customer_id = c.customer_id;
```

Row	count_of_city	count_of_state
1	8011	27

- As per above query, **customers have registered from 8011 cities, but orders were only placed from 5812 cities**. Although count of states remain same. There are 2199 cities where no order was placed in the given time frame.
- Action items from this analysis:
    - More analysis to be done on why orders weren't placed from remaining 2199 cities. This count contributes **27.4 % of total cities** from which customers have registered.

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

Query: `select year, count(distinct order_id) as order_count from (select extract(year from order_purchase_timestamp) as year,order_id from `project.orders`) as innerquery1 group by year order by year;`

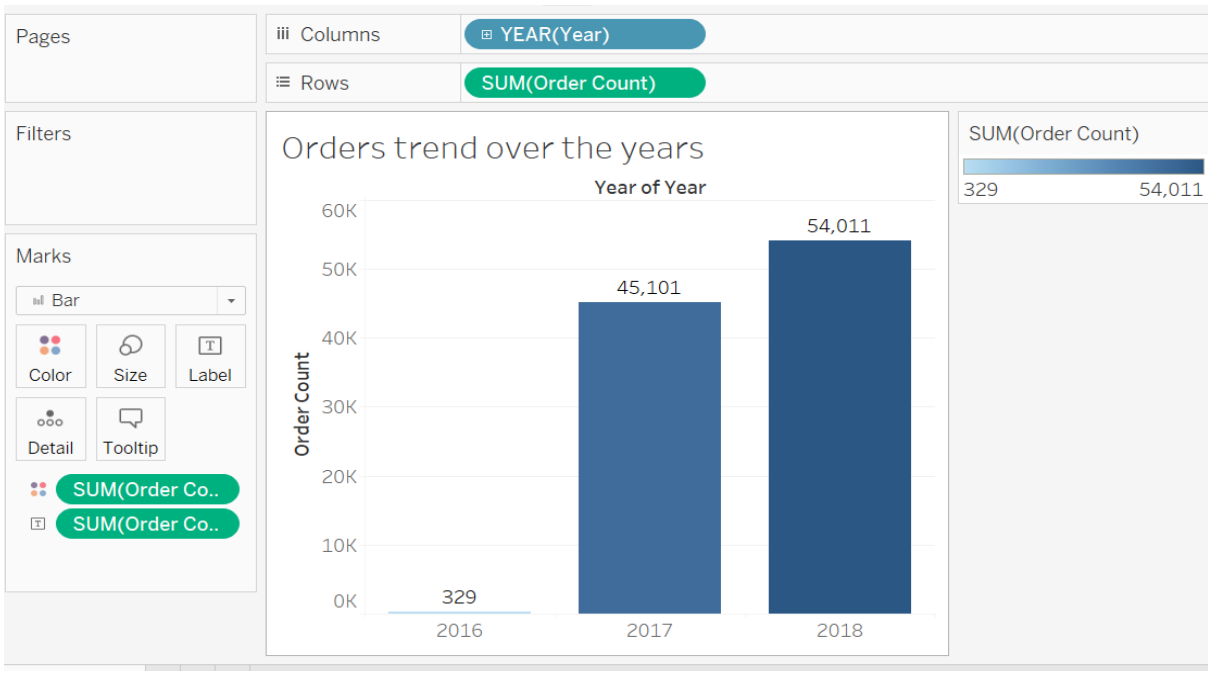


```
select year, count(distinct order_id) as order_count
from (select extract(year from order_purchase_timestamp) as year,order_id
from `project.orders`) as innerquery1
group by year
order by year;
```

Query results

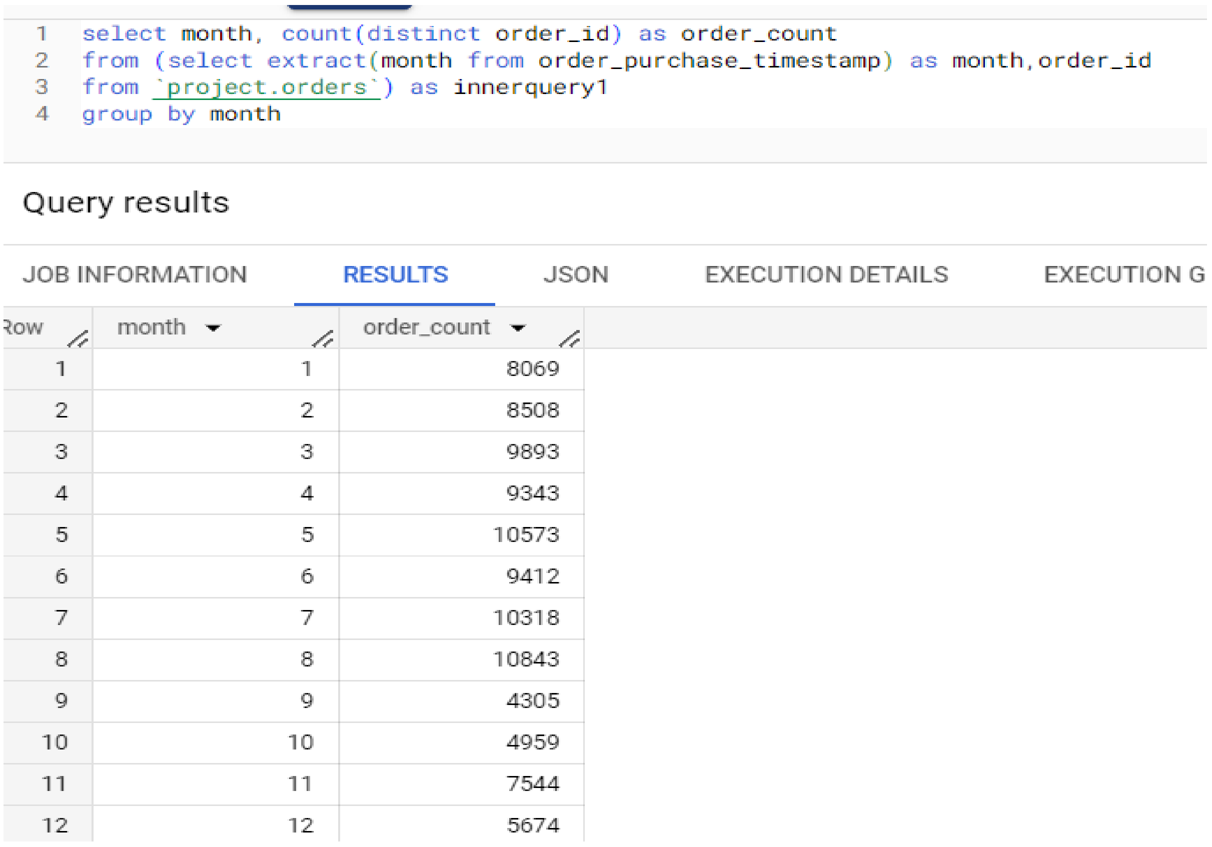
Row	year	order_count
1	2016	329
2	2017	45101
3	2018	54011

- Point to consider here is, **dataset given in this case starts from September 2016** till October 2018. This explains less order count in 2016, but still if we project the data for the whole year, orders in 2016 were the least of all years.
- We had the dataset for entire 2017, there is a significant surge of around 35x in orders placed from 2016.
- Again time range in 2018 is cut-off till October, but still there is a slight uptick in the orders placed from 2017.
- To summarize, after considering above factors, we could say that **order count has increased over the years** given in the dataset.
- The company's brand reach in Brazil has experienced remarkable growth since its inception in 2016, which is evident in the substantial increase in order count over the subsequent two years.



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

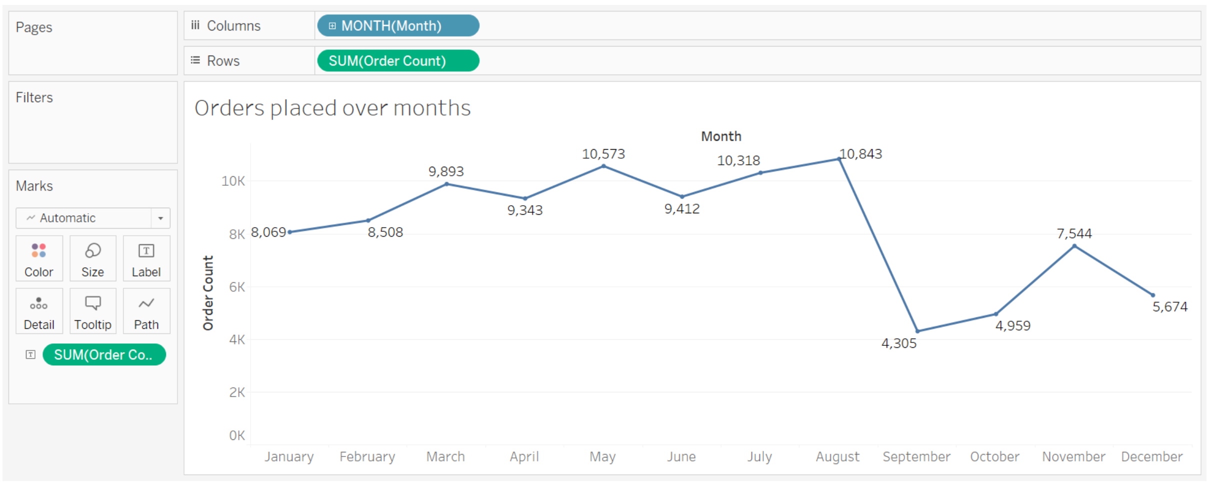
- Query: `select month, count(distinct order_id) as order_count from (select extract(month from order_purchase_timestamp) as month,order_id from `project.orders`) as innerquery1 group by month order by month;`



```
select month, count(distinct order_id) as order_count
from (select extract(month from order_purchase_timestamp) as month,order_id
from `project.orders`) as innerquery1
group by month
```

Query results

Row	month	order_count
1	1	8069
2	2	8508
3	3	9893
4	4	9343
5	5	10573
6	6	9412
7	7	10318
8	8	10843
9	9	4305
10	10	4959
11	11	7544
12	12	5674



- Even though September and October has seen orders from all 3 years, it seems the **total order count for September month is the least**, **followed by October.**
- January to August has seen orders in 2016 and 2017, there is small spike in orders over these months except April and June where we have seen a slight dip from previous months.
- November and December has seen orders in 2016 and 2017, but we have seen most orders in 2018, **maybe if we had 2018 data, November and December months would have be on par with months from January to August.**
- Considering all these factors, **we can conclude that September and October has seen very less orders and August has seen the most orders**. Orders from other months are somewhat near to August. **September and October have distanced themselves with other months.**

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

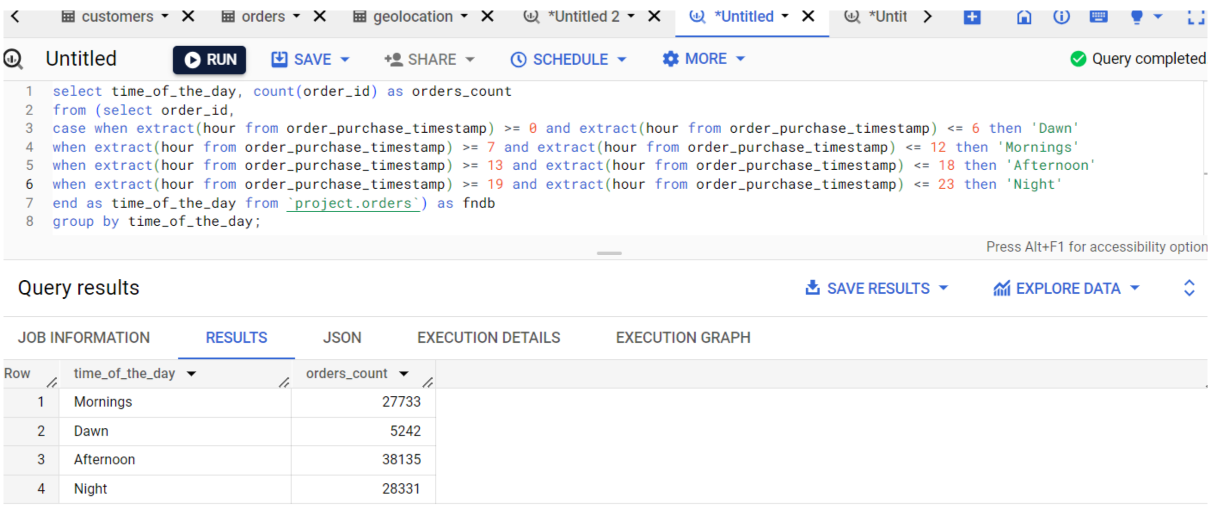
0-6 hrs: Dawn

7-12 hrs: Mornings

13-18 hrs: Afternoon

19-23 hrs: Night

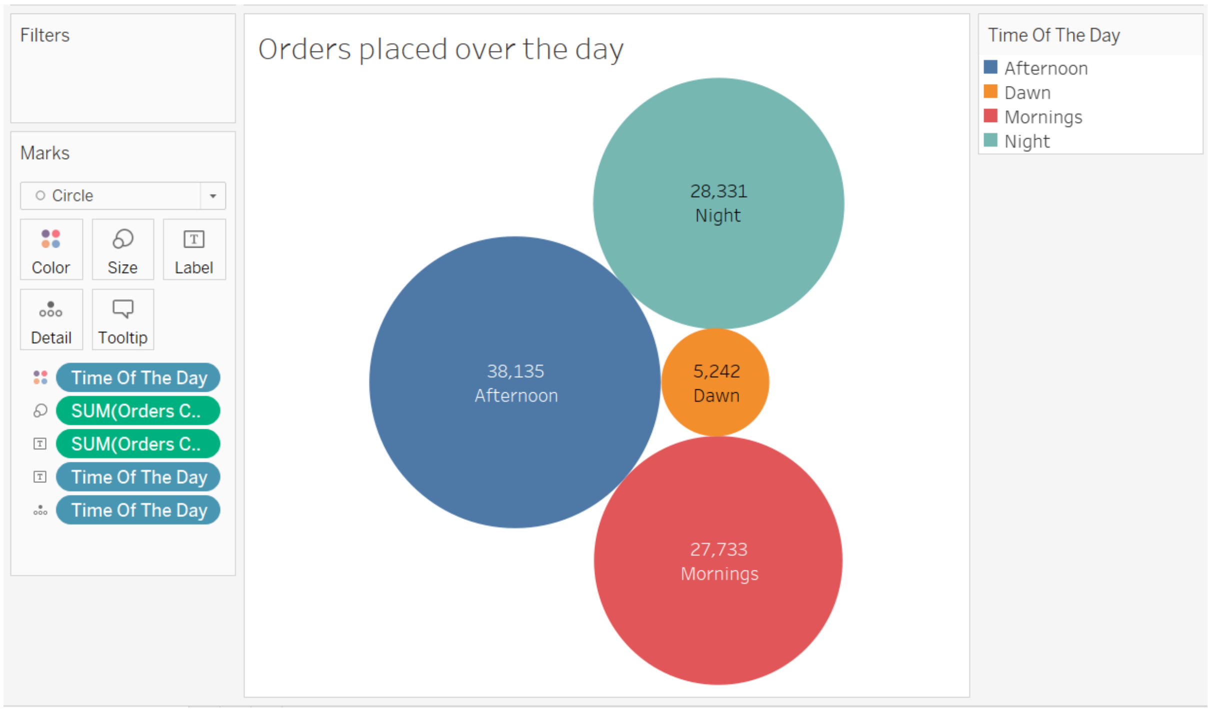
- Query: `select time_of_the_day, count(order_id) as orders_count from (select order_id, case when extract(hour from order_purchase_timestamp) >= 0 and extract(hour from order_purchase_timestamp) <= 6 then 'Dawn' when extract(hour from order_purchase_timestamp) >= 7 and extract(hour from order_purchase_timestamp) <= 12 then 'Mornings' when extract(hour from order_purchase_timestamp) >= 13 and extract(hour from order_purchase_timestamp) <= 18 then 'Afternoon' when extract(hour from order_purchase_timestamp) >= 19 and extract(hour from order_purchase_timestamp) <= 23 then 'Night' end as time_of_the_day from `project.orders`) as fndb group by time_of_the_day;`



```
select time_of_the_day, count(order_id) as orders_count
from (select order_id,
case when extract(hour from order_purchase_timestamp) >= 0 and extract(hour from order_purchase_timestamp) <= 6 then 'Dawn'
when extract(hour from order_purchase_timestamp) >= 7 and extract(hour from order_purchase_timestamp) <= 12 then 'Mornings'
when extract(hour from order_purchase_timestamp) >= 13 and extract(hour from order_purchase_timestamp) <= 18 then 'Afternoon'
when extract(hour from order_purchase_timestamp) >= 19 and extract(hour from order_purchase_timestamp) <= 23 then 'Night'
end as time_of_the_day from `project.orders`) as fndb
group by time_of_the_day;
```

Query results

Row	time_of_the_day	orders_count
1	Mornings	27733
2	Dawn	5242
3	Afternoon	38135
4	Night	28331

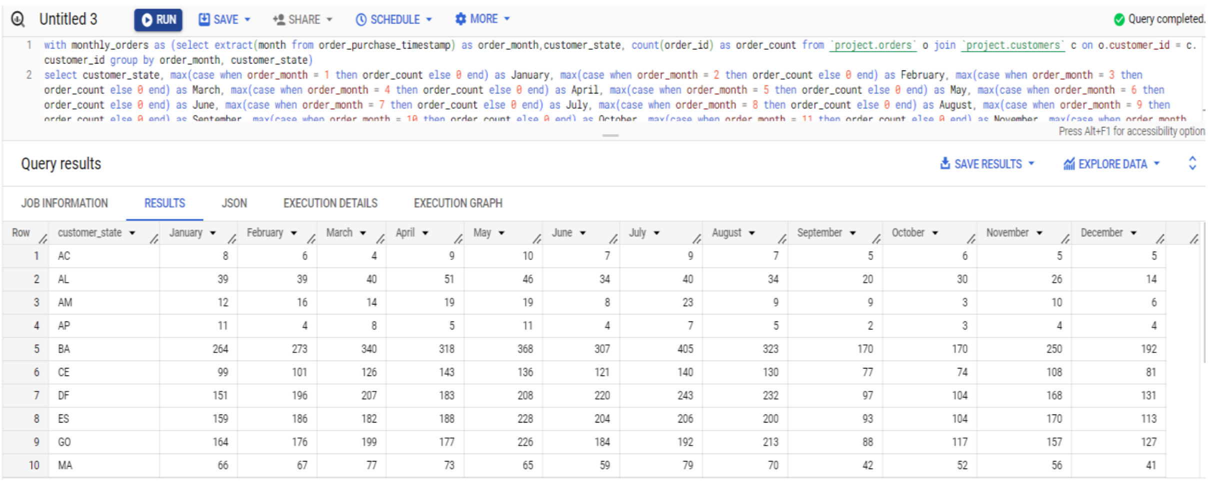


- There is a clear pattern of order placement based on the time of the day.
- The morning period experiences the highest number of orders, suggesting that it might be a popular time for customers to make purchases.
- The afternoon period also shows a significant order count, indicating sustained purchasing activity throughout the day.
- The night period still contributes a substantial number of orders, although it is slightly lower than the morning and afternoon periods.
- The dawn period has the lowest order count, indicating a relatively quieter time for order placements.
- Based on above query and observations we can conclude that **Afternoons have seen most orders** in the given data, Mornings and Night have seen similar number of orders, and **Dawn has seen least number of orders**, distancing itself from others.

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

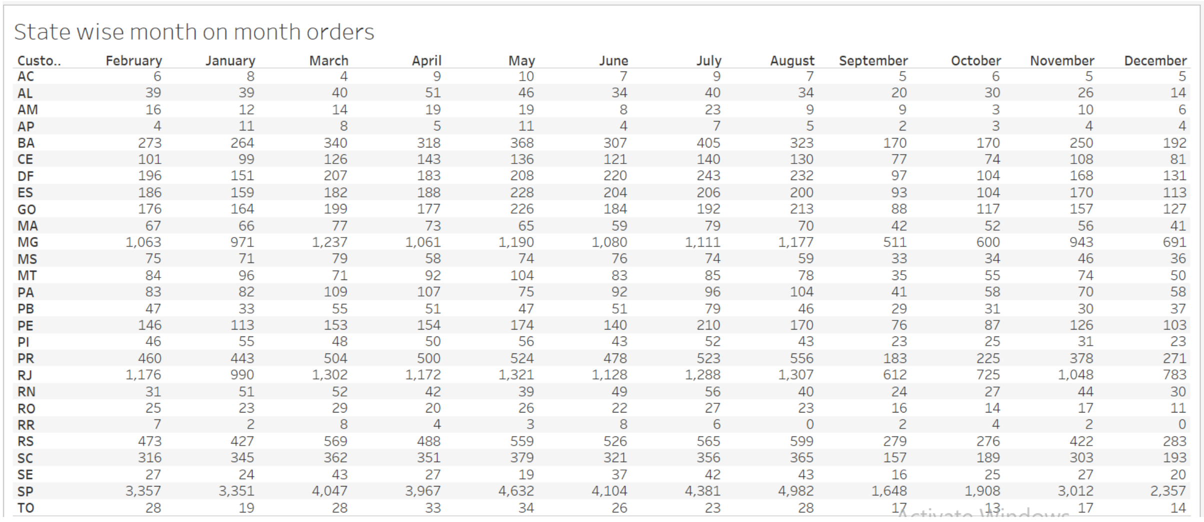
Query:
```
with monthly_orders as (select extract(month from
order_purchase_timestamp) as order_month,customer_state,
count(order_id) as order_count from `project.orders` o join
`project.customers` c on o.customer_id = c.customer_id group by
order_month, customer_state)

select customer_state, max(case when order_month = 1 then order_count
else 0 end) as January, max(case when order_month = 2 then order_count
else 0 end) as February, max(case when order_month = 3 then order_count
else 0 end) as March, max(case when order_month = 4 then order_count
else 0 end) as April, max(case when order_month = 5 then order_count
else 0 end) as May, max(case when order_month = 6 then order_count
else 0 end) as June, max(case when order_month = 7 then order_count
else 0 end) as July, max(case when order_month = 8 then order_count
else 0 end) as August, max(case when order_month = 9 then order_count
else 0 end) as September, max(case when order_month = 10 then
order_count else 0 end) as October, max(case when order_month = 11
then order_count else 0 end) as November, max(case when order_month =
12 then order_count else 0 end) as December from monthly_orders group
by customer_state order by customer_state;
```



```
with monthly_orders as (select extract(month from order_purchase_timestamp) as order_month,customer_state, count(order_id) as order_count from `project.orders` o join `project.customers` c on o.customer_id = c.
customer_id group by order_month, customer_state)
select customer_state, max(case when order_month = 1 then order_count else 0 end) as January, max(case when order_month = 2 then order_count else 0 end) as February, max(case when order_month = 3 then
order_count else 0 end) as March, max(case when order_month = 4 then order_count else 0 end) as April, max(case when order_month = 5 then order_count else 0 end) as May, max(case when order_month = 6 then
order_count else 0 end) as June, max(case when order_month = 7 then order_count else 0 end) as July, max(case when order_month = 8 then order_count else 0 end) as August, max(case when order_month = 9 then
order_count else 0 end) as September, max(case when order_month = 10 then order_count else 0 end) as October, max(case when order_month = 11 then order_count else 0 end) as November, max(case when order_month
```

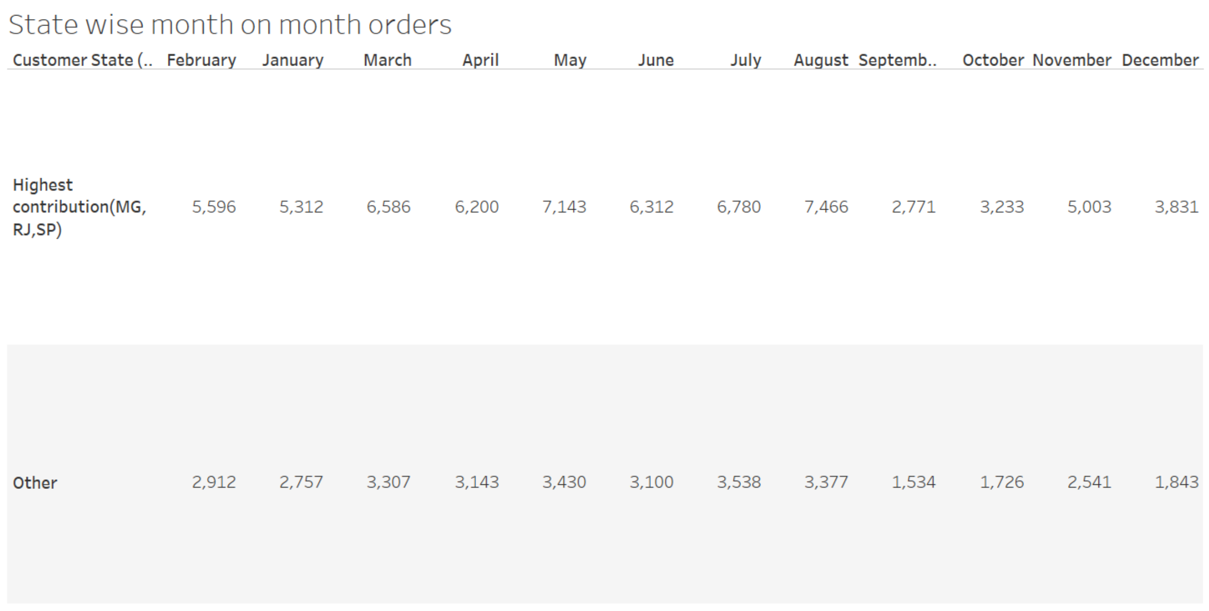
Row	customer_state	January	February	March	April	May	June	July	August	September	October	November	December
1	AC	8	6	4	9	10	7	9	7	5	6	5	5
2	AL	39	39	40	51	46	34	40	34	20	30	26	14
3	AM	12	16	14	19	19	8	23	9	9	3	10	6
4	AP	11	4	8	5	11	4	7	5	2	3	4	4
5	BA	264	273	340	318	368	307	405	323	170	170	250	192
6	CE	99	101	126	143	136	121	140	130	77	74	108	81
7	DF	151	196	207	183	208	220	243	232	97	104	168	131
8	ES	159	186	182	188	228	204	206	200	93	104	170	113
9	GO	164	176	199	177	226	184	192	213	88	117	157	127
10	MA	66	67	77	73	65	59	79	70	42	52	56	41



State wise month on month orders

Custo..	February	January	March	April	May	June	July	August	September	October	November	December
AC	6	8	4	9	10	7	9	7	5	6	5	5
AL	39	39	40	51	46	34	40	34	20	30	26	14
AM	16	12	14	19	19	8	23	9	9	3	10	6
AP	4	11	8	5	11	4	7	5	2	3	4	4
BA	273	264	340	318	368	307	405	323	170	170	250	192
CE	101	99	126	143	136	121	140	130	77	74	108	81
DF	196	151	207	183	208	220	243	232	97	104	168	131
ES	186	159	182	188	228	204	206	200	93	104	170	113
GO	176	164	199	177	226	184	192	213	88	117	157	127
MA	67	66	77	73	65	59	79	70	42	52	56	41
MG	1,063	971	1,237	1,061	1,190	1,080	1,111	1,177	511	600	943	691
MS	75	71	79	58	74	76	74	59	33	34	46	36
MT	84	96	71	92	104	83	85	78	35	55	74	50
PA	83	82	109	107	75	92	96	104	41	58	70	58
PB	47	33	55	51	47	51	79	46	29	31	30	37
PE	146	113	153	154	174	140	210	170	76	87	126	103
PI	46	55	48	50	56	43	52	43	23	25	31	23
PR	460	443	504	500	524	478	523	556	183	225	378	271
RJ	1,176	990	1,302	1,172	1,321	1,128	1,288	1,307	612	725	1,048	783
RN	31	51	52	42	39	49	56	40	24	27	44	30
RO	25	23	29	20	26	22	27	23	16	14	17	11
RR	7	2	8	4	3	8	6	0	2	4	2	0
RS	473	427	569	488	559	526	565	599	279	276	422	283
SC	316	345	362	351	379	321	356	365	157	189	303	193
SE	27	24	43	27	19	37	42	43	16	25	27	20
SP	3,357	3,351	4,047	3,967	4,632	4,104	4,381	4,982	1,648	1,908	3,012	2,357
TO	28	19	28	33	34	26	23	28	17	13	17	14

- **SP has highest orders** in all months across all states, where as **RR has seen least number of orders** across all months.
- August has recorded the highest number of orders across almost all states, while September has witnessed the lowest order count in the majority of states.
- **MG, RJ, SP have contributed around 65% to 68%** in number of orders across all months against all states.

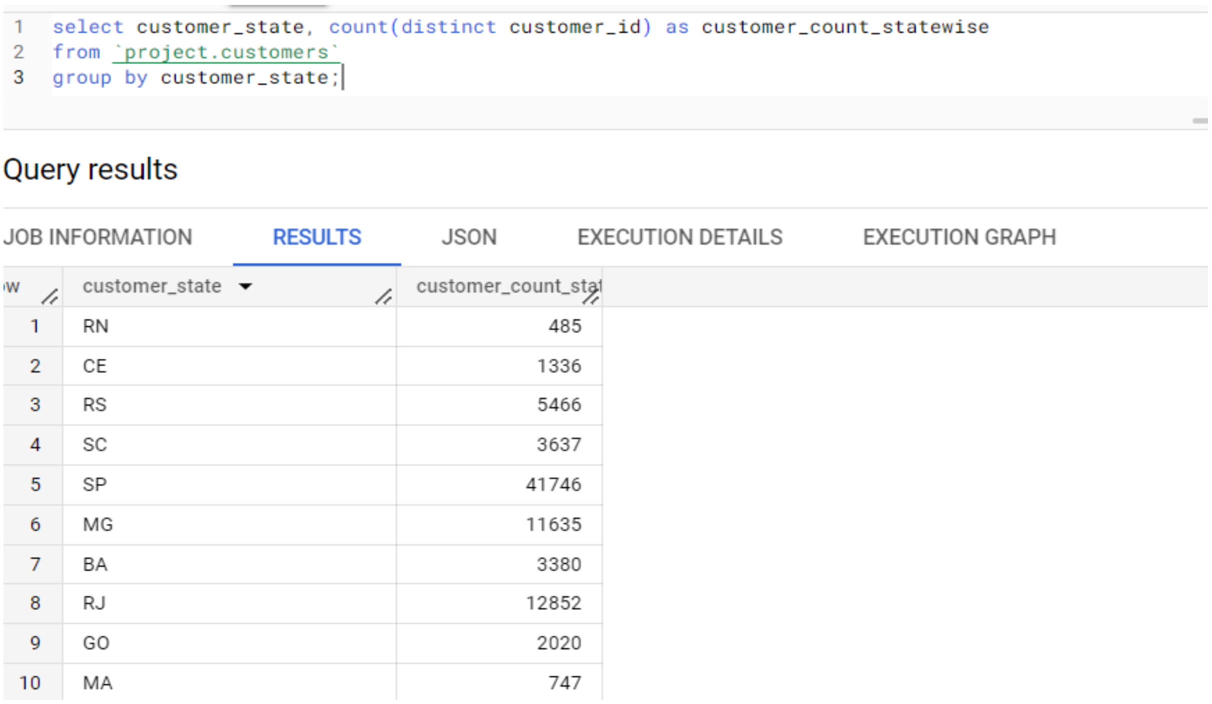


State wise month on month orders

Customer State (..	February	January	March	April	May	June	July	August	Septemb..	October	November	December
Highest contribution(MG, RJ,SP)	5,596	5,312	6,586	6,200	7,143	6,312	6,780	7,466	2,771	3,233	5,003	3,831
Other	2,912	2,757	3,307	3,143	3,430	3,100	3,538	3,377	1,534	1,726	2,541	1,843

3.2: How are the customers distributed across all the states?

- Query: `select customer_state, count(distinct customer_id) as customer_count_statewise from `project.customers` group by customer_state;`



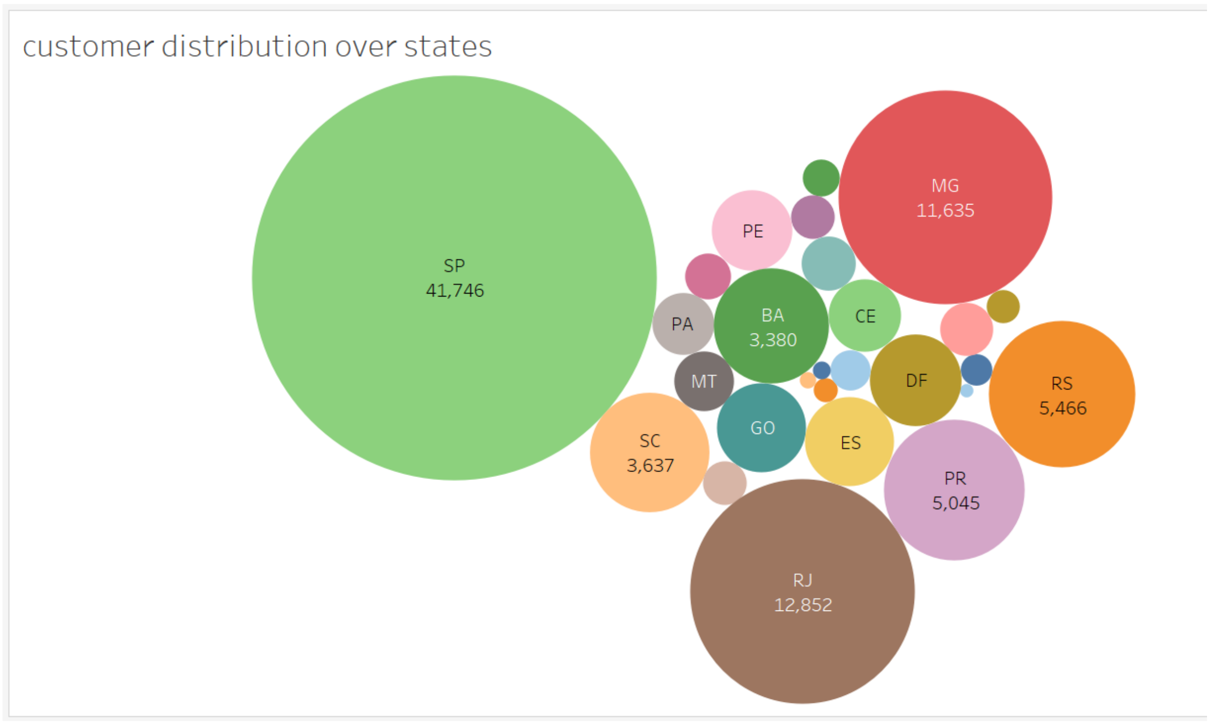
```
select customer_state, count(distinct customer_id) as customer_count_statewise
from `project.customers`
group by customer_state;
```

Query results

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

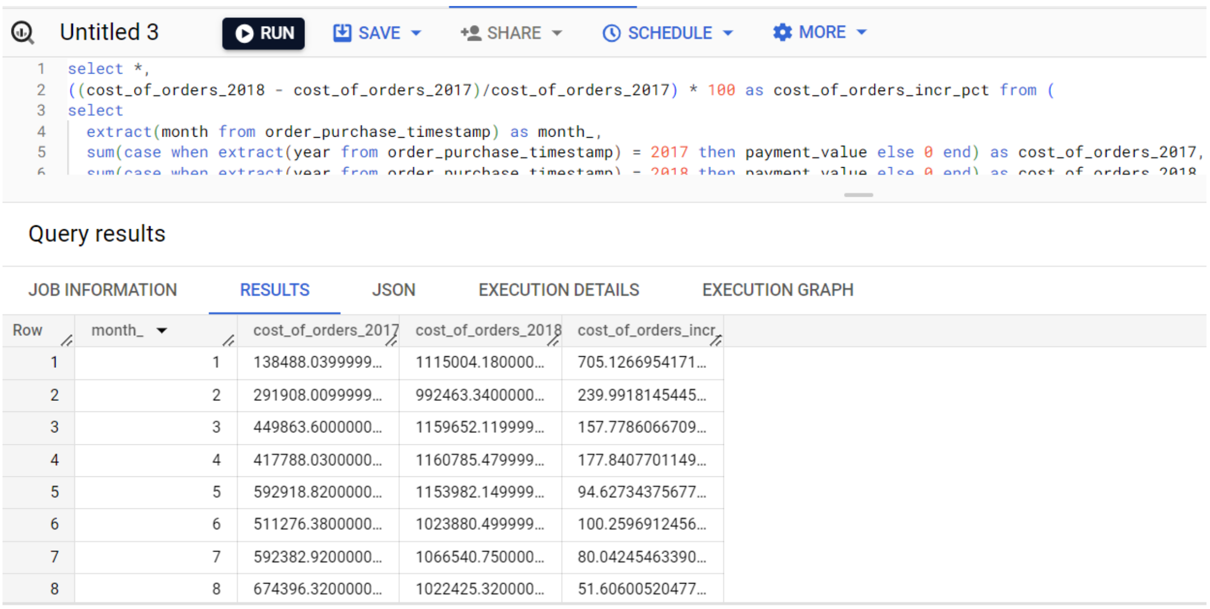
w	customer_state	customer_count_stat
1	RN	485
2	CE	1336
3	RS	5466
4	SC	3637
5	SP	41746
6	MG	11635
7	BA	3380
8	RJ	12852
9	GO	2020
10	MA	747

- SP alone has 41.9% customers in Brazil, much ahead of all states. RJ & MG are next best states contributing 12.8% and 11.7% respectively.
- SP, RJ, MG have 66.6% of customers. And average customers registered in Brazil is 3683.
- RR, AP, AC are among the least in customer count. Need to do more analysis on why we are seeing this less count in these areas along with other states which have less registered customers that average (3683).
- Below representation would give the picture of customer distribution over states.



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.

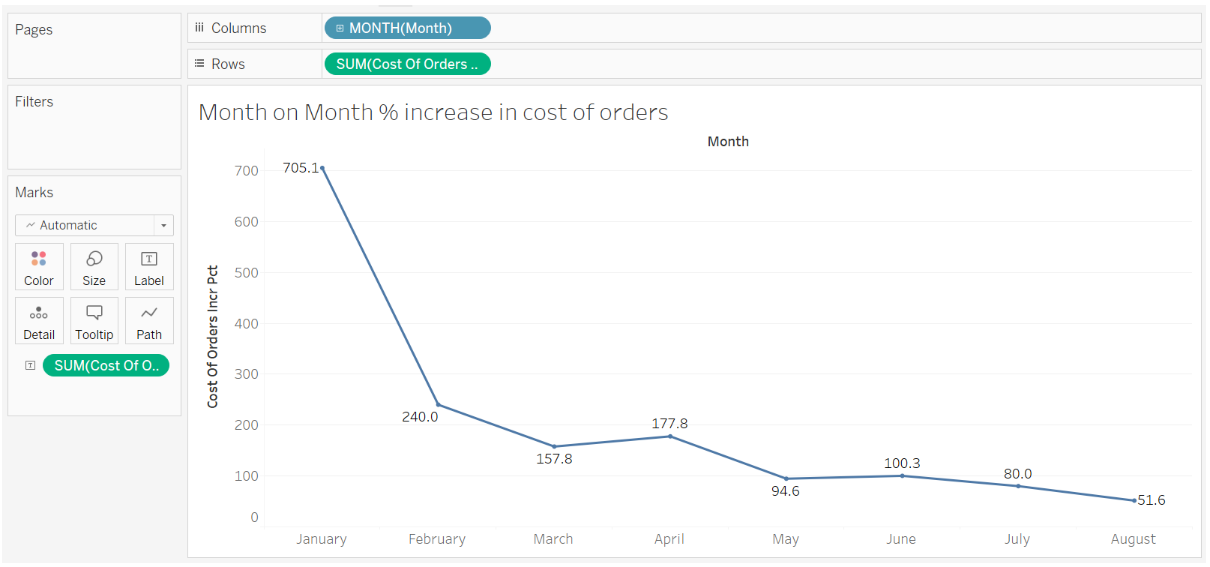
- Query:
```
select *,((cost_of_orders_2018 - cost_of_orders_2017)/cost_of_orders_2017) * 100 as cost_of_orders_incr_pct from (select extract(month from order_purchase_timestamp) as month_, sum(case when extract(year from order_purchase_timestamp) = 2017 then payment_value else 0 end) as cost_of_orders_2017, sum(case when extract(year from order_purchase_timestamp) = 2018 then payment_value else 0 end) as cost_of_orders_2018 from `project.orders` o join `project.payments` p on p.order_id = o.order_id where extract(month from order_purchase_timestamp) <= 8 and extract(year from order_purchase_timestamp) in (2017, 2018) group by month_) order by month_;
```



```
select *,
((cost_of_orders_2018 - cost_of_orders_2017)/cost_of_orders_2017) * 100 as cost_of_orders_incr_pct from (
select
  extract(month from order_purchase_timestamp) as month_,
  sum(case when extract(year from order_purchase_timestamp) = 2017 then payment_value else 0 end) as cost_of_orders_2017,
  sum(case when extract(year from order_purchase_timestamp) = 2018 then payment_value else 0 end) as cost_of_orders_2018
```

Query results

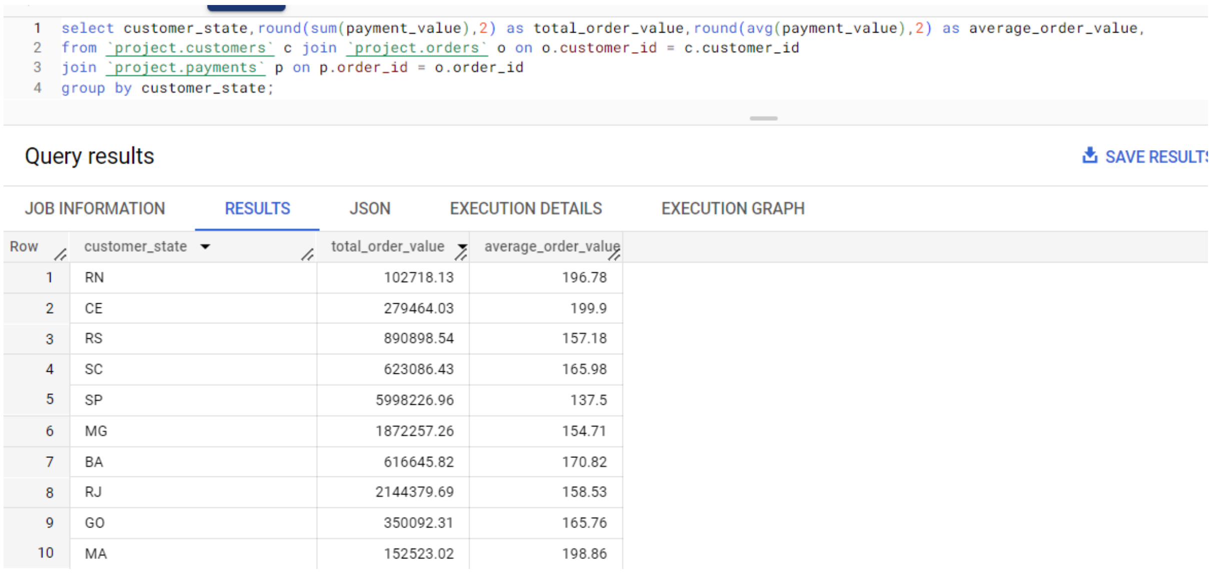
Row	month_	cost_of_orders_2017	cost_of_orders_2018	cost_of_orders_incr
1	1	138488.0399999...	1115004.180000...	705.1266954171...
2	2	291908.0099999...	992463.3400000...	239.9918145445...
3	3	449863.6000000...	1159652.119999...	157.7786066709...
4	4	417788.0300000...	1160785.479999...	177.8407701149...
5	5	592918.8200000...	1153982.149999...	94.62734375677...
6	6	511276.3800000...	1023880.499999...	100.2596912456...
7	7	592382.9200000...	1066540.750000...	80.04245463390...
8	8	674396.3200000...	1022425.320000...	51.60600520477...



- January has the highest increase in the cost of orders, with a significant margin of 705.1% compared to other months in the dataset.
- High month-on-month percentage increases may indicate successful marketing campaigns or promotions during the month of January.
- Barring April, there increment percentage has decreased till August to 51.6%.
- Overall, a gradual decrease in the month-on-month percentage increase in orders indicates a slowdown in growth rates.
- Further analysis and strategic adjustments to be made to sustain business performance.
- It is important to consider various factors and market conditions to understand the underlying reasons behind the decreasing trend and develop appropriate strategies to address it.

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

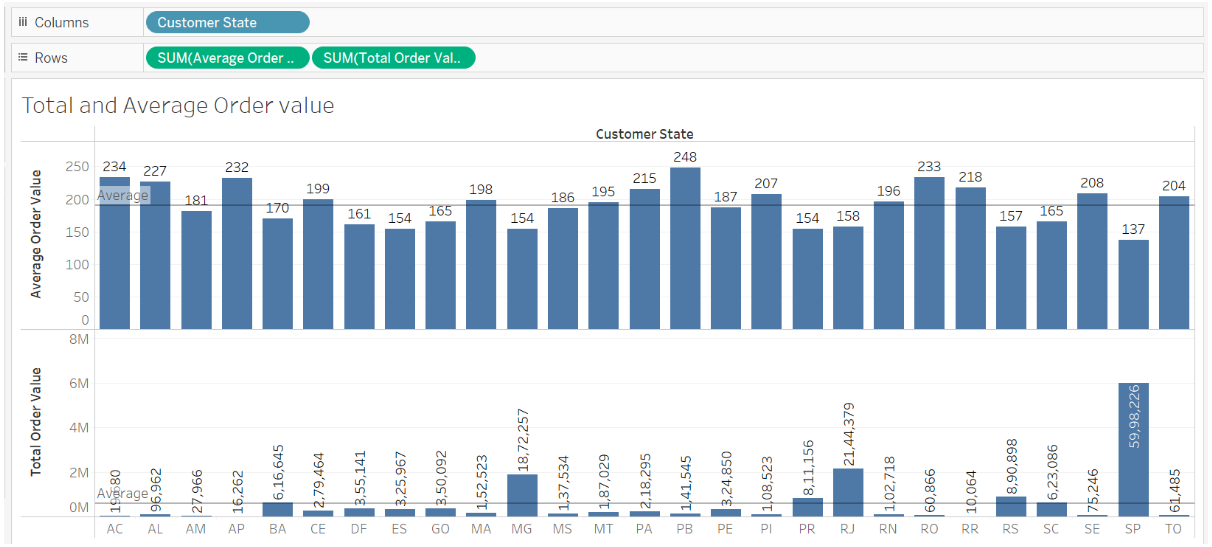
- Query: `select customer_state, sum(payment_value) as total_order_value, avg(payment_value) as average_order_value, from `project.customers` c join `project.orders` o on o.customer_id = c.customer_id join `project.payments` p on p.order_id = o.order_id group by customer_state;`



```
select customer_state,round(sum(payment_value),2) as total_order_value,round(avg(payment_value),2) as average_order_value,
from `project.customers` c join `project.orders` o on o.customer_id = c.customer_id
join `project.payments` p on p.order_id = o.order_id
group by customer_state;
```

Query results

Row	customer_state	total_order_value	average_order_value
1	RN	102718.13	196.78
2	CE	279464.03	199.9
3	RS	890898.54	157.18
4	SC	623086.43	165.98
5	SP	5998226.96	137.5
6	MG	1872257.26	154.71
7	BA	616645.82	170.82
8	RJ	2144379.69	158.53
9	GO	350092.31	165.76
10	MA	152523.02	198.86

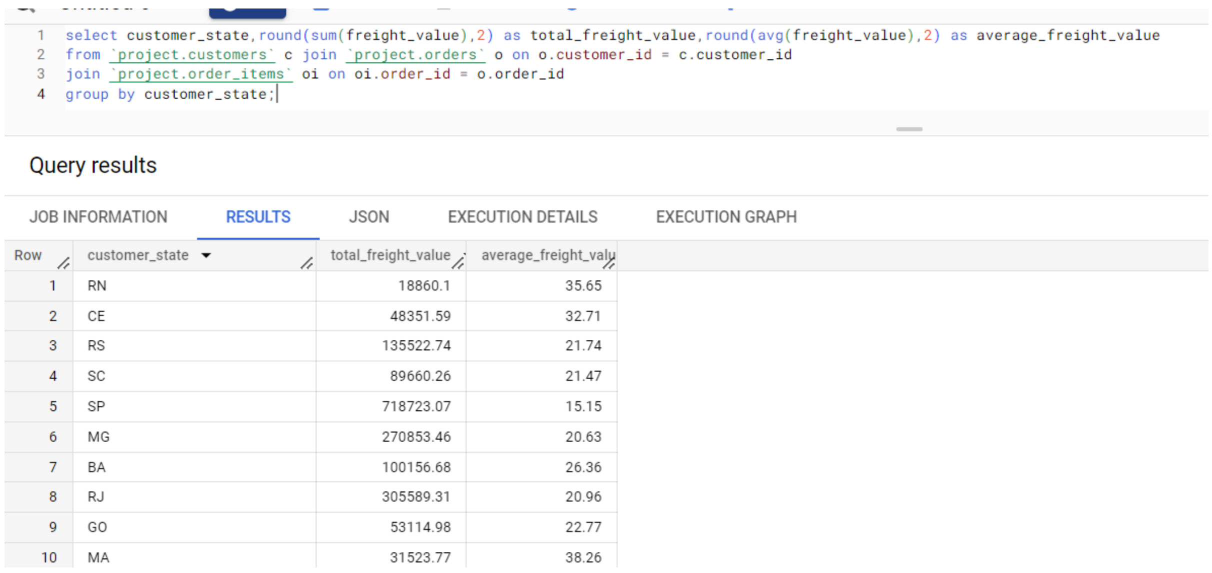


- SP is topping the charts with Total order value. RJ and MG are next best states in terms of total order value.
- These states likely have a higher purchasing power or a larger number of high-value transactions. They can be considered key markets for targeting premium products or services.
- RR, AP and AC are among least states having less Total order value.
- Further analysis is required to **understand the reasons behind the lower order prices and also to make new strategies** to increase sales or improve customer engagement in the regions who have less than average total order value.

- Even though **SP had highest Total order value, it has least Average order value** compared to all states.
- PB has highest average order value of 248, but has less than average total order value.
- Further analysis should be conducted on **the SP state to explore the potential of introducing premium products**. Given the already high footfall and a chance to increase average order values, this presents an opportunity to significantly boost revenues.

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

- Query: `select customer_state,round(sum(freight_value),2) as total_freight_value,round(avg(freight_value),2) as average_freight_value from `project.customers` c join `project.orders` o on o.customer_id = c.customer_id join `project.order_items` oi on oi.order_id = o.order_id group by customer_state;`

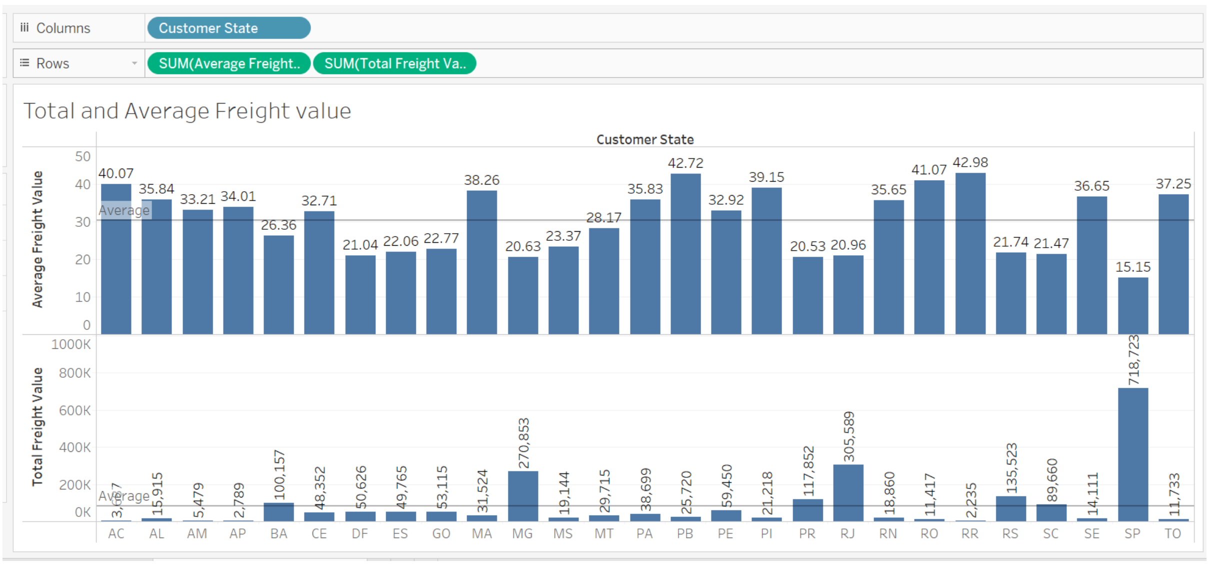


```
select customer_state,round(sum(freight_value),2) as total_freight_value,round(avg(freight_value),2) as average_freight_value
from `project.customers` c join `project.orders` o on o.customer_id = c.customer_id
join `project.order_items` oi on oi.order_id = o.order_id
group by customer_state;
```

Query results

JOB INFORMATION RESULTS JSON EXECUTION DETAILS EXECUTION GRAPH

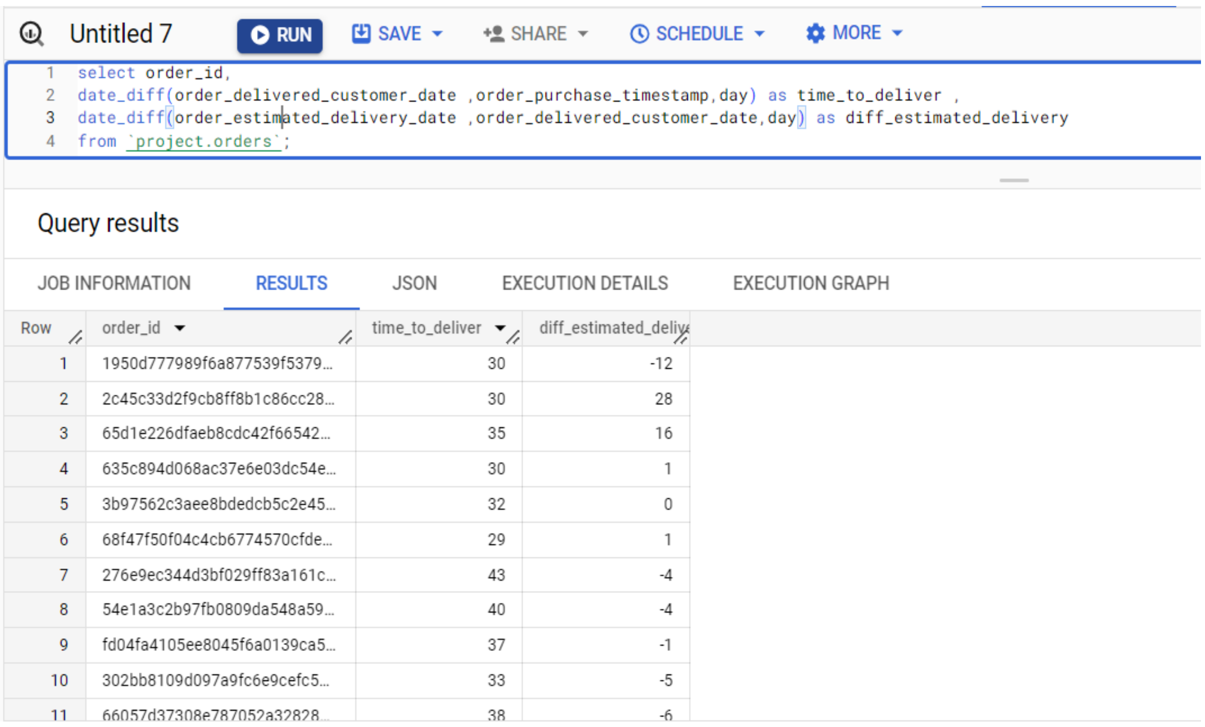
Row	customer_state	total_freight_value	average_freight_valu
1	RN	18860.1	35.65
2	CE	48351.59	32.71
3	RS	135522.74	21.74
4	SC	89660.26	21.47
5	SP	718723.07	15.15
6	MG	270853.46	20.63
7	BA	100156.68	26.36
8	RJ	305589.31	20.96
9	GO	53114.98	22.77
10	MA	31523.77	38.26



- As expected, SP had the highest order value, which is also reflected in the highest freight value. RJ and MG follow closely in this list.
- RR had least total order value and least total freight value, but it has highest average freight value.
- Although RR had the lowest total freight value, it implies that the individual orders that RR did have incurred higher transportation costs on average compared to the other entities. This suggests that RR's shipments may have been smaller or more specialized, resulting in higher freight costs per order.

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.

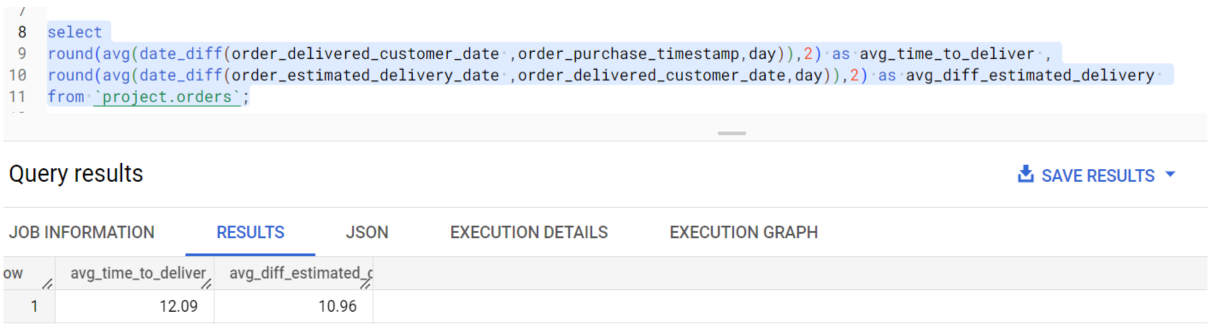
- Query: `select order_id,date_diff(order_delivered_customer_date ,order_purchase_timestamp,day) as time_to_deliver ,date_diff(order_estimated_delivery_date ,order_delivered_customer_date,day) as diff_estimated_delivery from `project.orders`;`



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

Row	order_id	time_to_deliver	diff_estimated_deliv
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

- Query to find out average delivery time and average diff estimated time: `select round(avg(date_diff(order_delivered_customer_date ,order_purchase_timestamp,day)),2) as avg_time_to_deliver , round(avg(date_diff(order_estimated_delivery_date ,order_delivered_customer_date,day)),2) as avg_diff_estimated_delivery from `project.orders`;`



```
select
round(avg(date_diff(order_delivered_customer_date ,order_purchase_timestamp,day)),2) as avg_time_to_deliver ,
round(avg(date_diff(order_estimated_delivery_date ,order_delivered_customer_date,day)),2) as avg_diff_estimated_delivery
from `project.orders`;
```

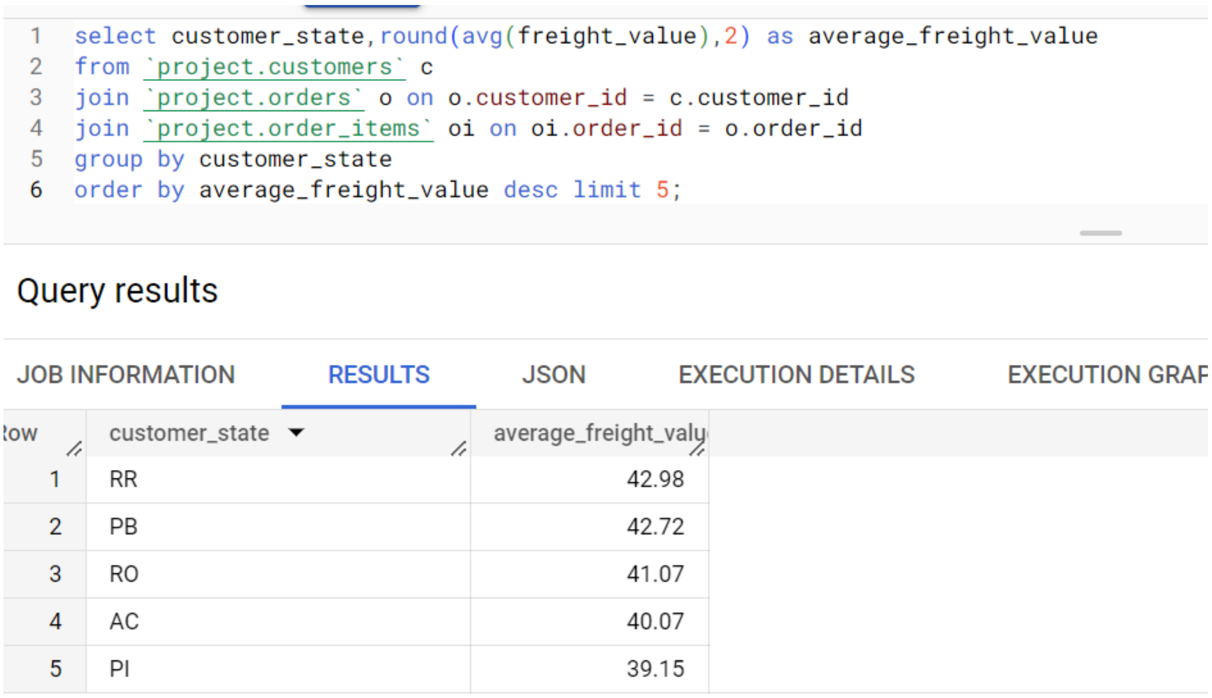
Query results

ow	avg_time_to_deliver	avg_diff_estimated_c
1	12.09	10.96

- On an average, orders take 12.09 days to deliver and are 10.96 days later than the estimated delivery date.
- This means that the company is consistently underestimating the amount of time it takes to fulfil the orders. This could lead to customer dissatisfaction and sales.
- The company may be using outdated or inaccurate data to estimate delivery times.
- Maybe we have to do further more analysis on how we are estimating delivery times, this could help to give more accurate delivery estimation time.

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

- Query for top 5 highest average freight value:
  `select customer_state,round(avg(freight_value),2) as average_freight_value from `project.customers` c join `project.orders` o on o.customer_id = c.customer_id join `project.order_items` oi on oi.order_id = o.order_id group by customer_stateorder by average_freight_value desc limit 5;`

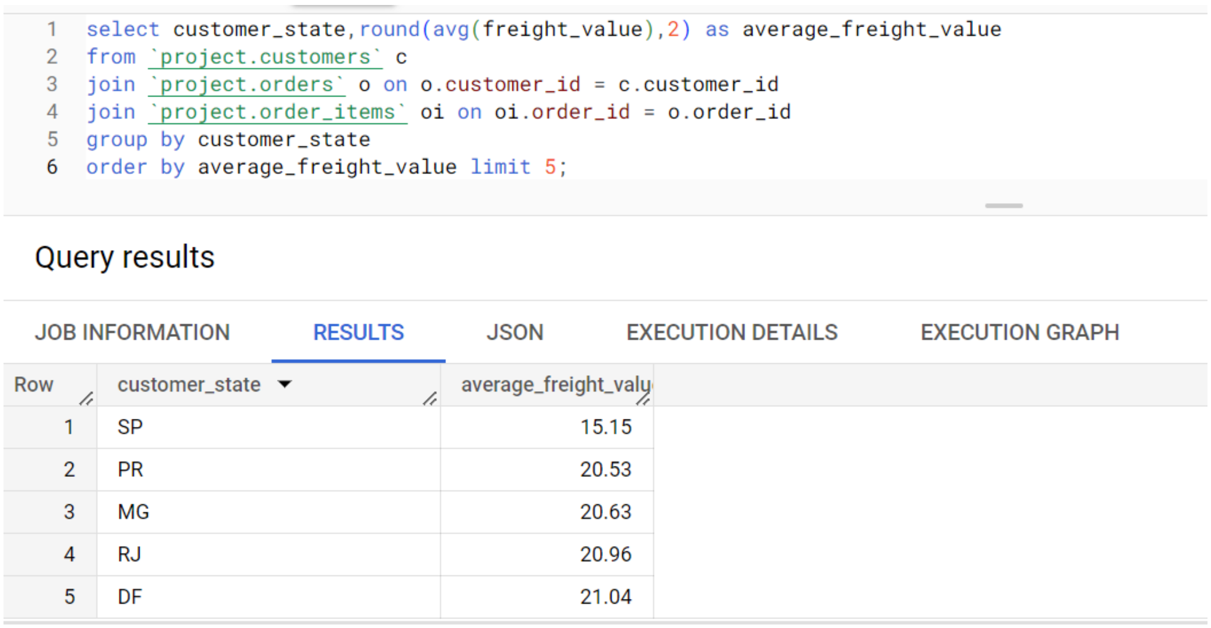


```
select customer_state,round(avg(freight_value),2) as average_freight_value
from `project.customers` c
join `project.orders` o on o.customer_id = c.customer_id
join `project.order_items` oi on oi.order_id = o.order_id
group by customer_state
order by average_freight_value desc limit 5;
```

Query results

Row	customer_state	average_freight_valu
1	RR	42.98
2	PB	42.72
3	RO	41.07
4	AC	40.07
5	PI	39.15

- Query for top 5 lowest average freight value:
  `select customer_state,round(avg(freight_value),2) as average_freight_value from `project.customers` c join `project.orders` o on o.customer_id = c.customer_id join `project.order_items` oi on oi.order_id = o.order_id group by customer_state order by average_freight_value limit 5;`



```
select customer_state,round(avg(freight_value),2) as average_freight_value
from `project.customers` c
join `project.orders` o on o.customer_id = c.customer_id
join `project.order_items` oi on oi.order_id = o.order_id
group by customer_state
order by average_freight_value limit 5;
```

Query results

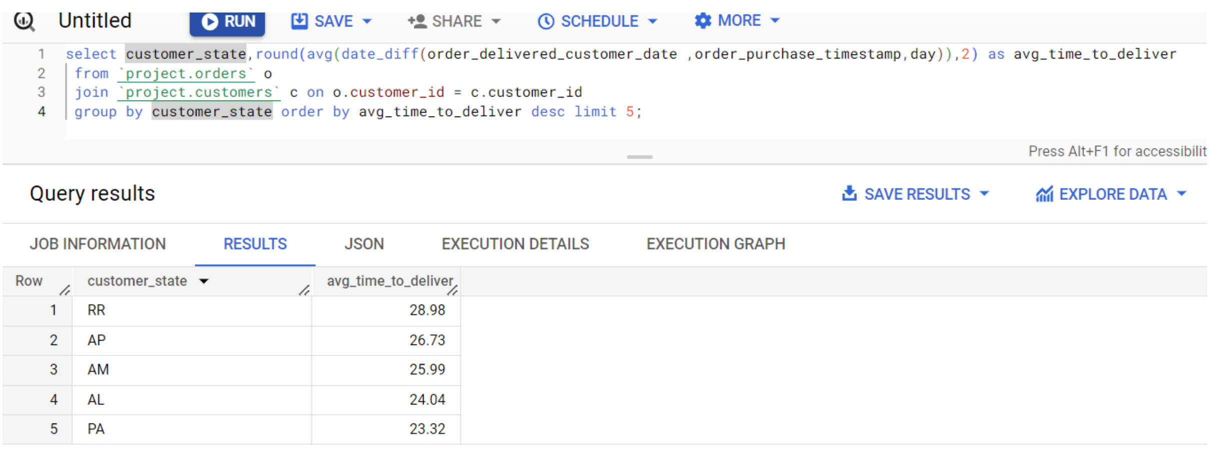
JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	customer_state	average_freight_valu
1	SP	15.15
2	PR	20.53
3	MG	20.63
4	RJ	20.96
5	DF	21.04

- RR had least total order value and least total freight value, but it has highest average freight value
- This means that the freight value was divided by less number of orders, which inflated the average freight value.
- On the other hand, SP had least average freight value, probably because of being the highest in order value and count.

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

- Query for top 5 states with highest average delivery time: `select customer_state,round(avg(date_diff(order_delivered_customer_date ,order_purchase_timestamp,day)),2) as avg_time_to_deliver from `project.orders` o join `project.customers` c on o.customer_id = c.customer_id group by customer_state order by avg_time_to_deliver desc limit 5;`
- It is concerning to observe that the following states consistently exhibit the highest average delivery times.
- We have to analyse customer delivery feedback regarding experiences in these states. Look for common and recurring issues to gain insights into specific points.
- We should develop state-specific strategies based on the unique challenges faced by each state.

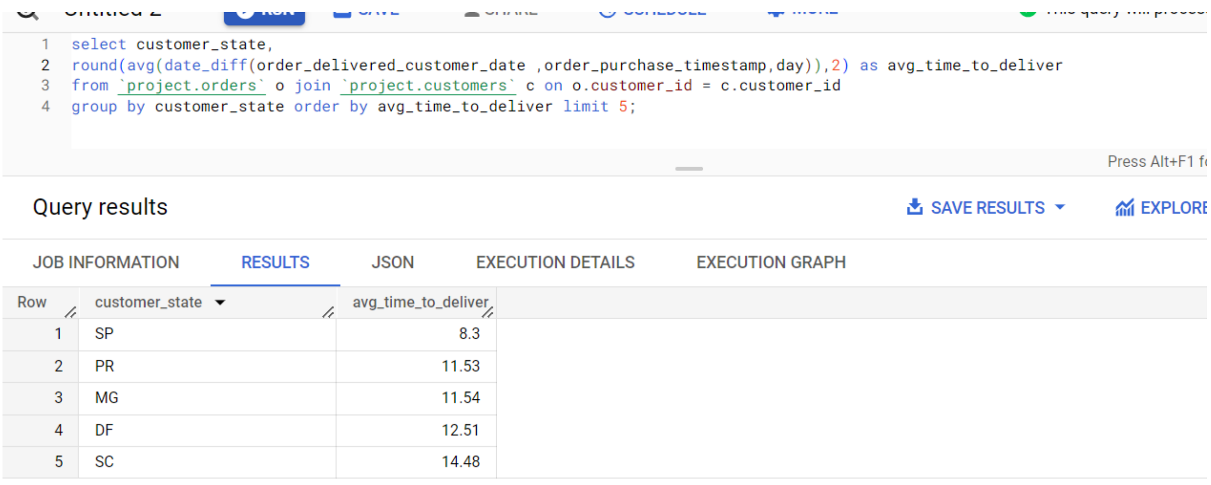


```
select customer_state,round(avg(date_diff(order_delivered_customer_date ,order_purchase_timestamp,day)),2) as avg_time_to_deliver
 from `project.orders` o
 join `project.customers` c on o.customer_id = c.customer_id
 group by customer_state order by avg_time_to_deliver desc limit 5;
```

Query results

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

- Query for top 5 states with lowest average delivery time: `select customer_state,round(avg(date_diff(order_delivered_customer_date ,order_purchase_timestamp,day)),2) as avg_time_to_deliver from `project.orders` o join `project.customers` c on o.customer_id = c.customer_id group by customer_state order by avg_time_to_deliver limit 5;`



```
select customer_state,
round(avg(date_diff(order_delivered_customer_date ,order_purchase_timestamp,day)),2) as avg_time_to_deliver
from `project.orders` o join `project.customers` c on o.customer_id = c.customer_id
group by customer_state order by avg_time_to_deliver limit 5;
```

Query results

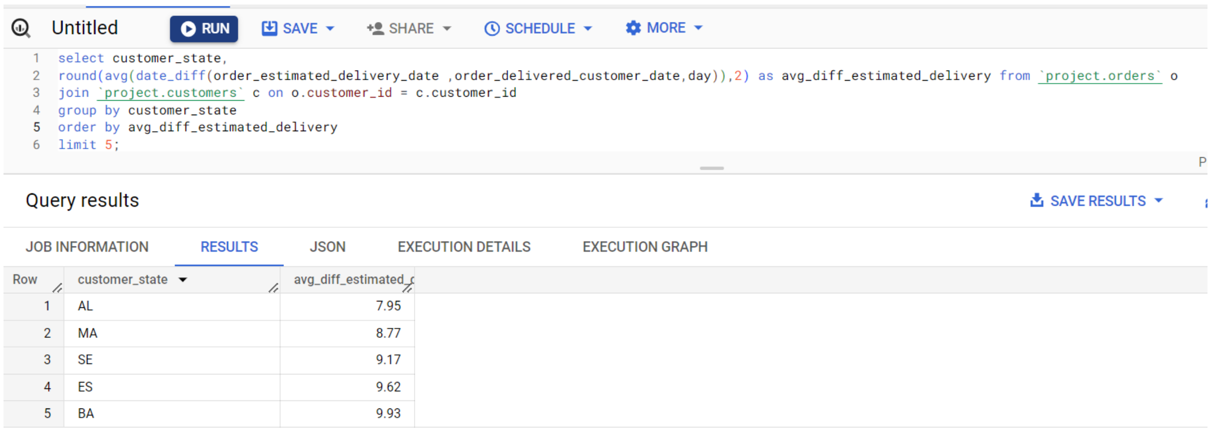
Row	customer_state	avg_time_to_deliver
1	SP	8.3
2	PR	11.53
3	MG	11.54
4	DF	12.51
5	SC	14.48

- SP is being impressive with average delivery time of 8.3 days. This indicates a strong performance in meeting customer expectations.
- However, even though the above states are currently experiencing the best delivery times, it is essential to continuously strive for improvement.
- Again regular feedback from customers have to be evaluated and take more actions from that feedback to make sure orders are delivered as early as possible.

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

- **Query:** `select customer_state, round(avg(date_diff(order_estimated_delivery_date ,order_delivered_customer_date,day)),2) as avg_diff_estimated_delivery from `project.orders` o join `project.customers` c on o.customer_id = c.customer_id group by customer_state order by avg_diff_estimated_delivery limit 5;`

- These states are likely to have efficient shipping and logistics infrastructure. This means that they have a well-developed network which help faster delivery.
- The states may also have a relatively short distance to travel between major cities. This can also help to speed up delivery times.

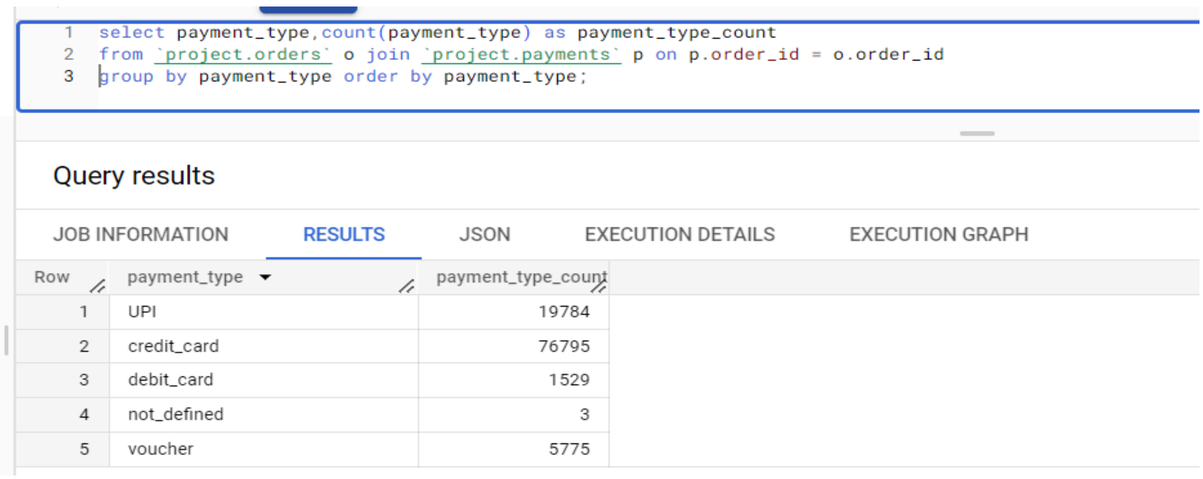


```
select customer_state,
round(avg(date_diff(order_estimated_delivery_date ,order_delivered_customer_date,day)),2) as avg_diff_estimated_delivery from `project.orders` o
join `project.customers` c on o.customer_id = c.customer_id
group by customer_state
order by avg_diff_estimated_delivery
limit 5;
```

Row	customer_state	avg_diff_estimated_
1	AL	7.95
2	MA	8.77
3	SE	9.17
4	ES	9.62
5	BA	9.93

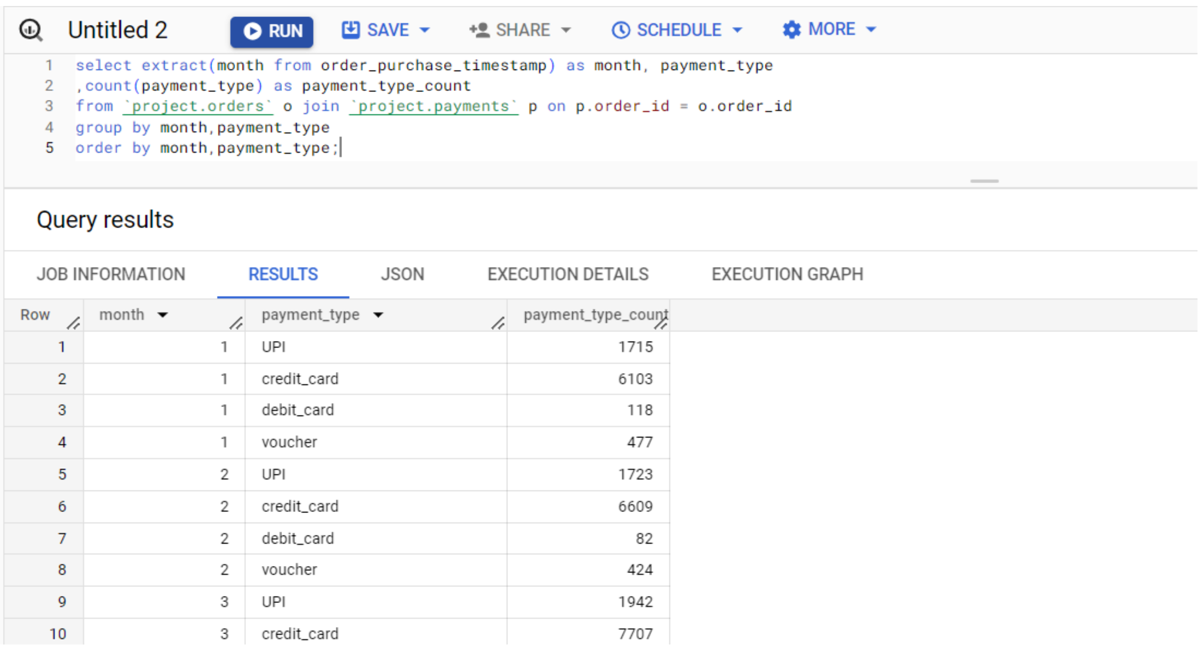
6.1: Find the month on month no. of orders placed using different payment types.

Query: `select extract(month from order_purchase_timestamp) as month, payment_type,count(payment_type) as payment_type_count from `project.orders` o join `project.payments` p on p.order_id = o.order_id group by month,payment_type order by month,payment_type;`



```
select payment_type,count(payment_type) as payment_type_count
from `project.orders` o join `project.payments` p on p.order_id = o.order_id
group by payment_type order by payment_type;
```

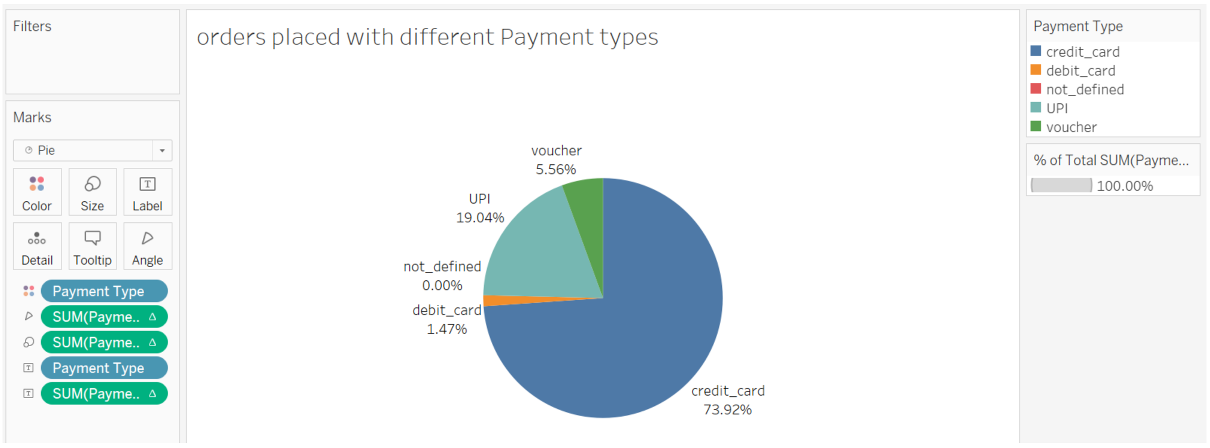
Row	payment_type	payment_type_count
1	UPI	19784
2	credit_card	76795
3	debit_card	1529
4	not_defined	3
5	voucher	5775

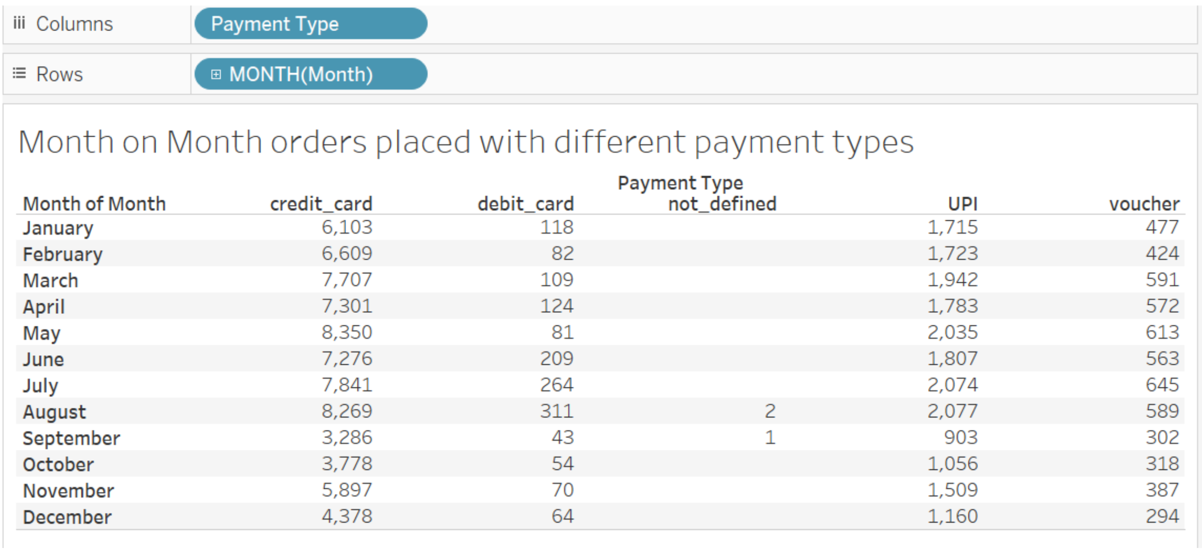


```sql
select extract(month from order_purchase_timestamp) as month, payment_type
,count(payment_type) as payment_type_count
from `project.orders` o join `project.payments` p on p.order_id = o.order_id
group by month,payment_type
order by month,payment_type;
```

Query results

Row	month	payment_type	payment_type_count
1	1	UPI	1715
2	1	credit_card	6103
3	1	debit_card	118
4	1	voucher	477
5	2	UPI	1723
6	2	credit_card	6609
7	2	debit_card	82
8	2	voucher	424
9	3	UPI	1942
10	3	credit_card	7707




Columns
Payment Type
Rows
MONTH(Month)

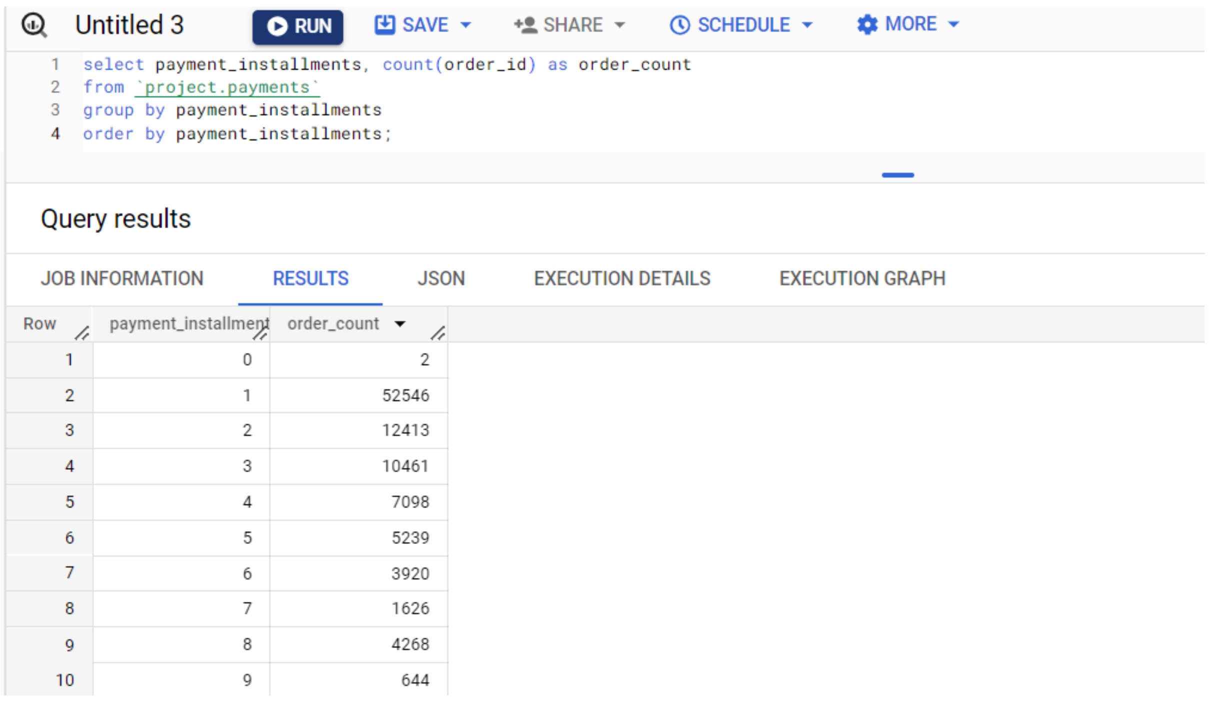
Month on Month orders placed with different payment types

	Payment Type				
Month of Month	credit_card	debit_card	not_defined	UPI	voucher
January	6,103	118		1,715	477
February	6,609	82		1,723	424
March	7,707	109		1,942	591
April	7,301	124		1,783	572
May	8,350	81		2,035	613
June	7,276	209		1,807	563
July	7,841	264		2,074	645
August	8,269	311	2	2,077	589
September	3,286	43	1	903	302
October	3,778	54		1,056	318
November	5,897	70		1,509	387
December	4,378	64		1,160	294

- Based on the analysis, we can say that credit card is the most commonly used payment type across all months and seasons, accounting for approximately 74% of all payments. This indicates a strong preference for credit card payments among customers.
- Also, we can say that credit card usage remains consistent throughout the year, suggesting that customers rely on this payment method regardless of seasonal variations.
- In addition to credit card payments, UPI accounts 19.04% of all payments. This indicates a growing popularity and adoption of UPI as a preferred payment method among customers.
- The increasing usage of UPI reflects the shift towards digital payment solutions in the market.
- Vouchers account for 5.56% of all payments, we have to do more analysis to track and analyse voucher usage patterns.

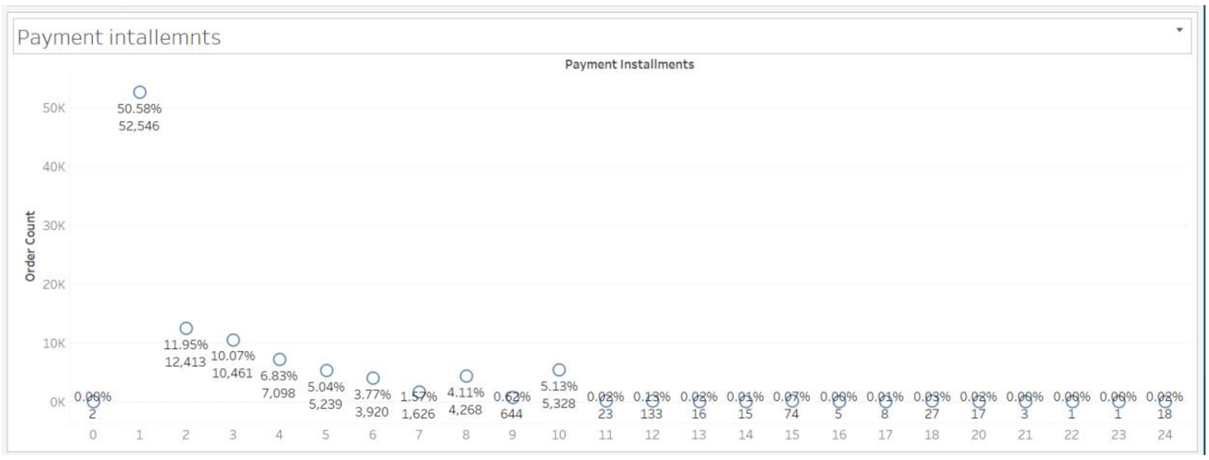
6.2: Find the no. of orders placed on the basis of the payment instalments that have been paid.

Query: `select payment_installments, count(order_id) as order_count from `project.payments` group by payment_installments order by payment_installments;`



```
select payment_installments, count(order_id) as order_count
from `project.payments`
group by payment_installments
order by payment_installments;
```

Row	payment_installments	order_count
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



- 50% of orders were paid in a single installment, indicating that customers prefer to pay upfront for their purchases.
- 80% of orders were paid in 5 installments, indicating that customers prefer to spread out the cost of their purchases over time.
- We can say that customers value the flexibility and convenience of spreading their payments over multiple installments.
- The data indicates a diversity of payment preferences among customers, with a significant proportion opting for both upfront payment and installment-based plans.