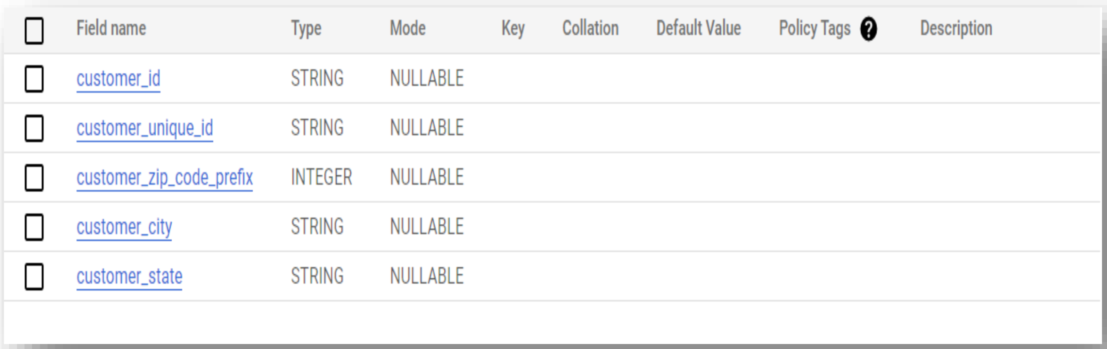
Business case study (Target Company)

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

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

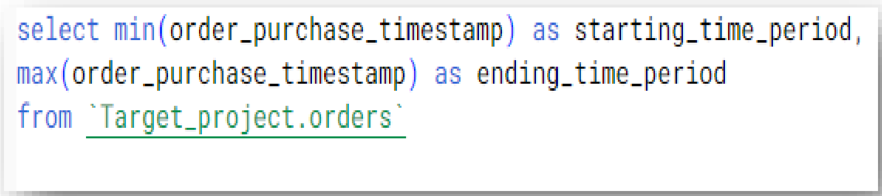
Result: -



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					

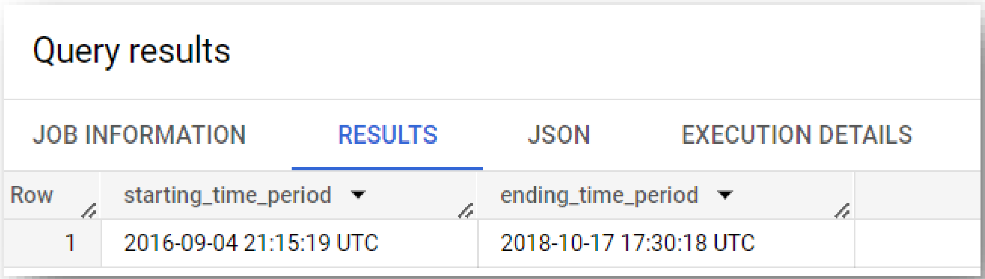
1.2 - Get the time range between which the order were placed.

Query: -



```
select min(order_purchase_timestamp) as starting_time_period,
max(order_purchase_timestamp) as ending_time_period
from `Target_project.orders`
```

Result: -

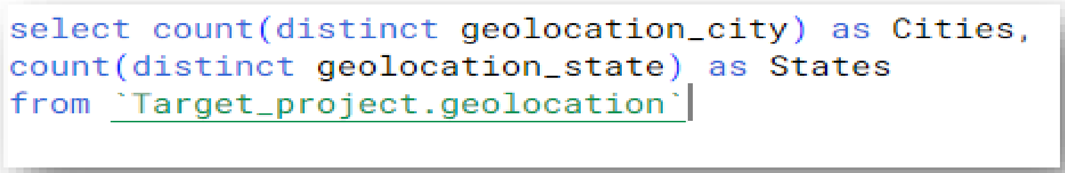


Query results

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

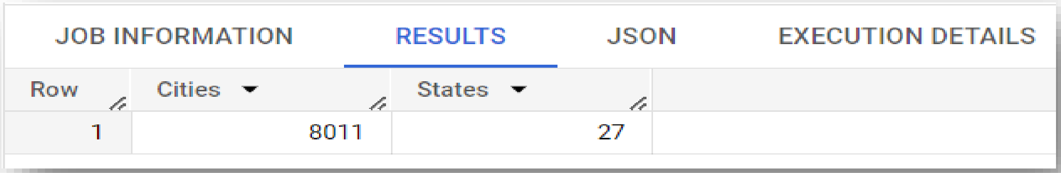
1.3 - Count the Cities & States of customers who ordered during the given period.

Query: -



```
select count(distinct geolocation_city) as Cities,
count(distinct geolocation_state) as States
from `Target_project.geolocation`
```

Result: -



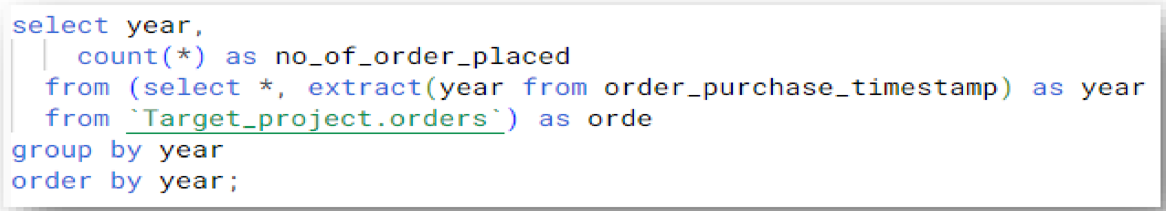
JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	Cities	States
1	8011	27

2. In-depth Exploration:

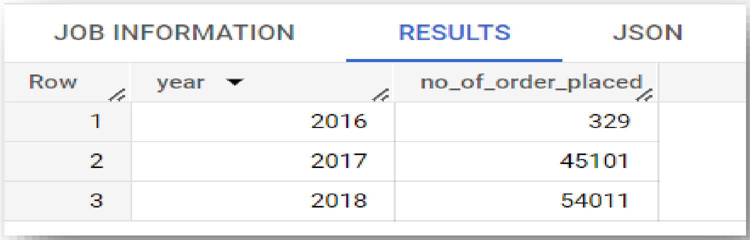
2.1: - Is there a growing trend in the no. of orders placed over the past years?

Query: -



```
select year,
    count(*) as no_of_order_placed
  from (select *, extract(year from order_purchase_timestamp) as year
  from `Target_project.orders`) as orde
group by year
order by year;
```

Result: -

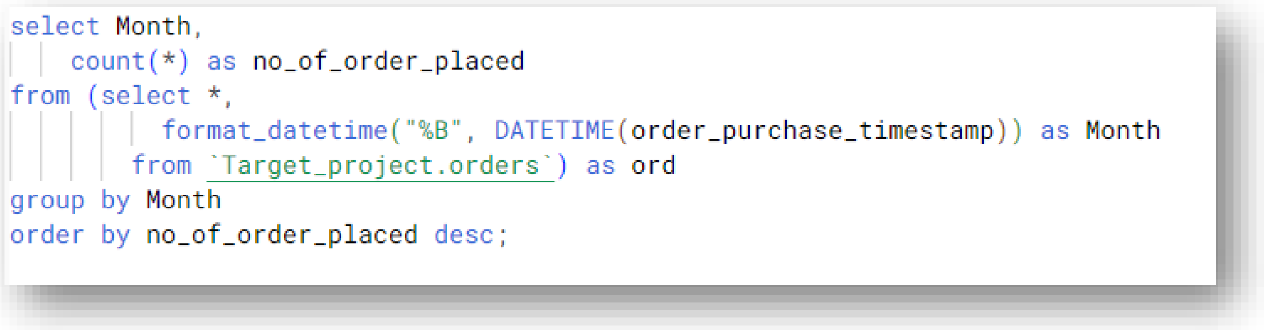


JOB INFORMATION | RESULTS | JSON

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

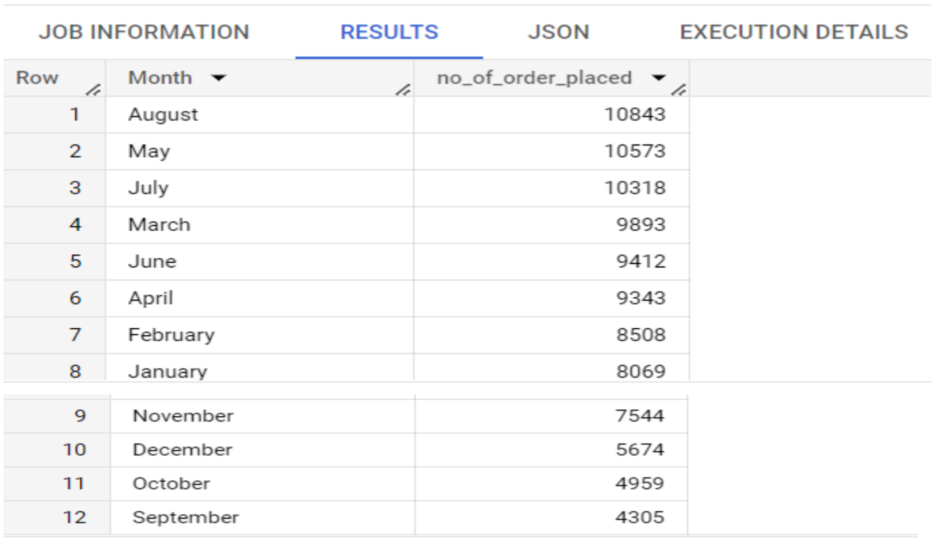
2.2: - Can we see some kind of monthly seasonality in terms of the no. of being placed?

Query: -



```
select Month,
    count(*) as no_of_order_placed
from (select *,
        format_datetime("%B", DATETIME(order_purchase_timestamp)) as Month
      from `Target_project.orders`) as ord
group by Month
order by no_of_order_placed desc;
```

Result: -



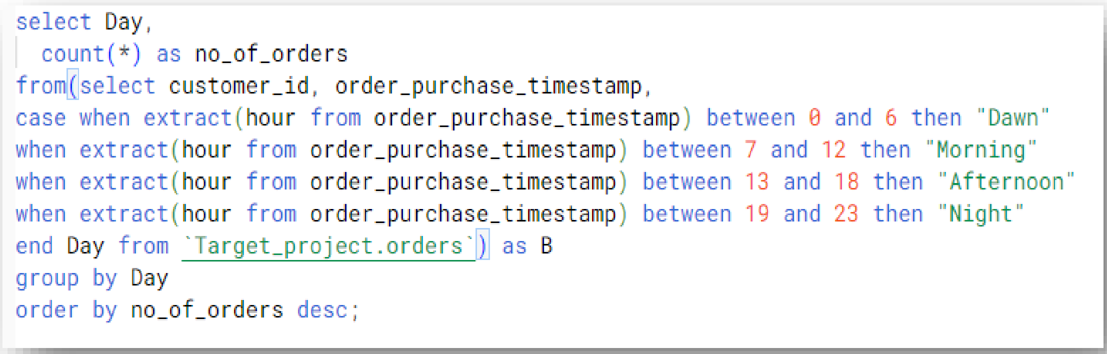
JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	Month	no_of_order_placed
1	August	10843
2	May	10573
3	July	10318
4	March	9893
5	June	9412
6	April	9343
7	February	8508
8	January	8069
9	November	7544
10	December	5674
11	October	4959
12	September	4305

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

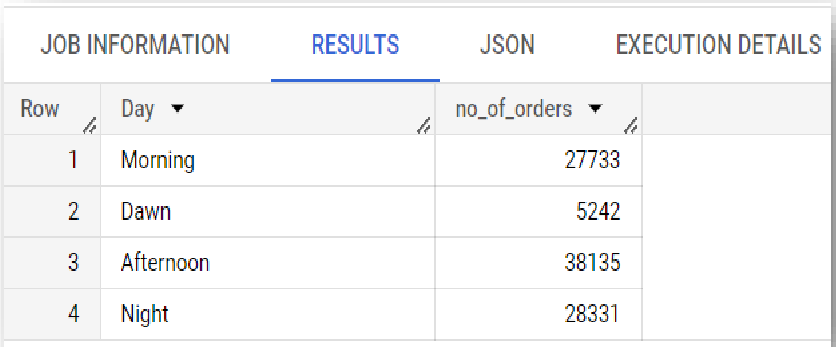
- 0-6 hrs.: Dawn
- 7-12 hrs.: Mornings
- 13-18 hrs.: Afternoon
- 19-23 hrs.: Night

Query: -



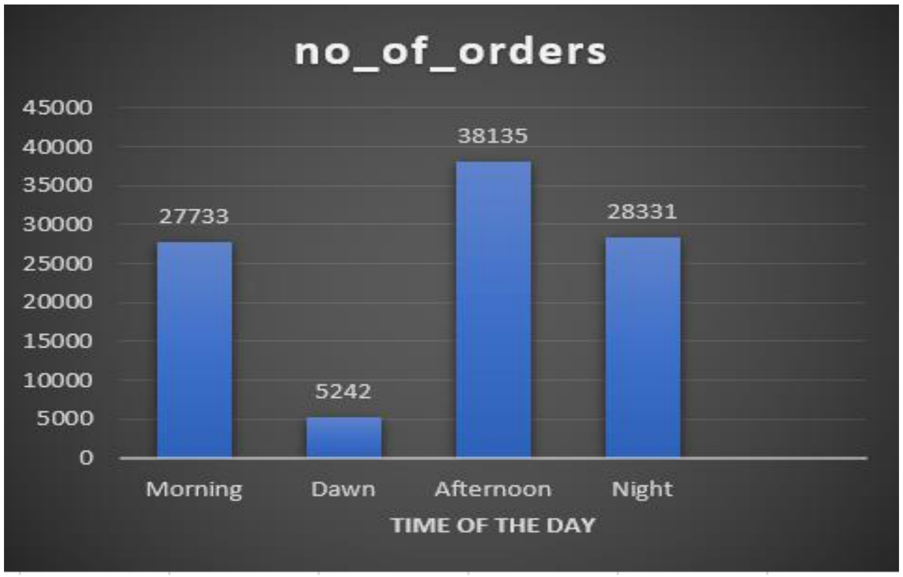
```
select Day,
  count(*) as no_of_orders
from(select customer_id, order_purchase_timestamp,
case when extract(hour from order_purchase_timestamp) between 0 and 6 then "Dawn"
when extract(hour from order_purchase_timestamp) between 7 and 12 then "Morning"
when extract(hour from order_purchase_timestamp) between 13 and 18 then "Afternoon"
when extract(hour from order_purchase_timestamp) between 19 and 23 then "Night"
end Day from `Target_project.orders`) as B
group by Day
order by no_of_orders desc;
```

Result: -



Row	Day	no_of_orders
1	Morning	27733
2	Dawn	5242
3	Afternoon	38135
4	Night	28331

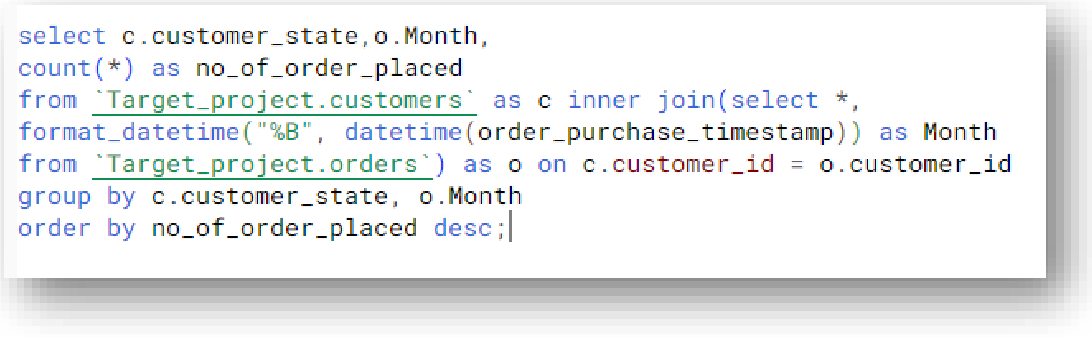
Graph Table (Result): -



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

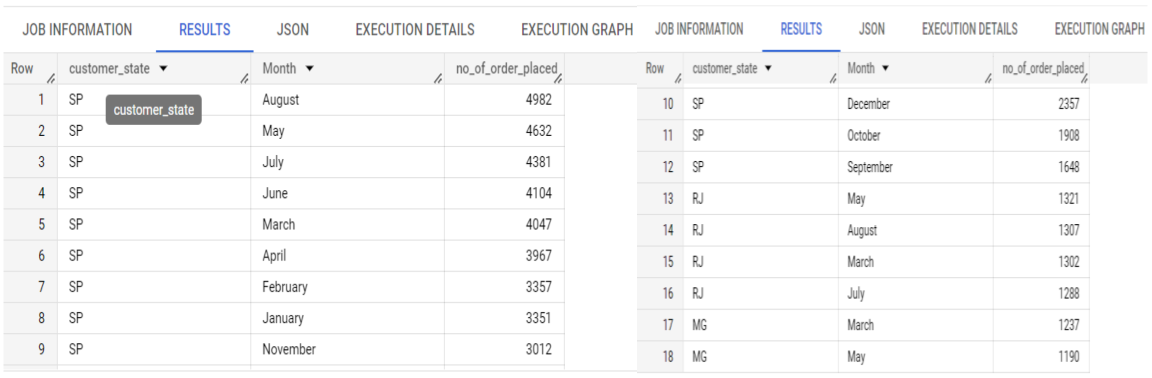
3.1- Get the month-on-month no. of orders placed in each state.

Query: -



```
select c.customer_state,o.Month,
count(*) as no_of_order_placed
from `Target_project.customers` as c inner join(select *,
format_datetime("%B", datetime(order_purchase_timestamp)) as Month
from `Target_project.orders`) as o on c.customer_id = o.customer_id
group by c.customer_state, o.Month
order by no_of_order_placed desc;
```

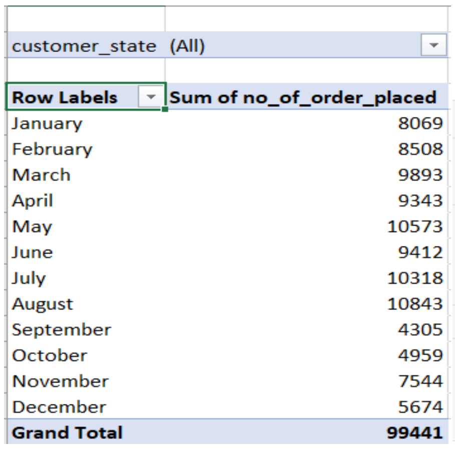
Result: -



Row	customer_state	Month	no_of_order_placed
1	SP	August	4982
2	SP	May	4632
3	SP	July	4381
4	SP	June	4104
5	SP	March	4047
6	SP	April	3967
7	SP	February	3357
8	SP	January	3351
9	SP	November	3012

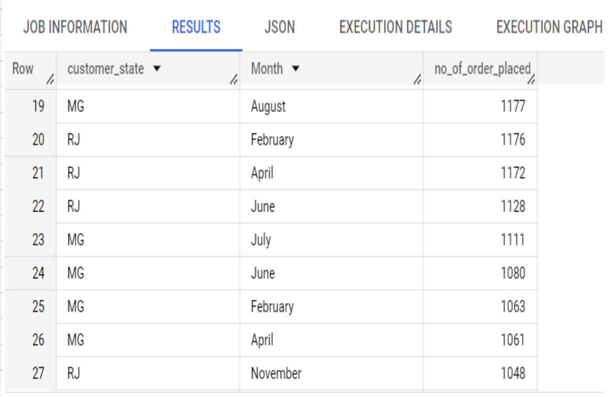
Row	customer_state	Month	no_of_order_placed
10	SP	December	2357
11	SP	October	1908
12	SP	September	1648
13	RJ	May	1321
14	RJ	August	1307
15	RJ	March	1302
16	RJ	July	1288
17	MG	March	1237
18	MG	May	1190

Pivot Table (Month wise total information of output): -



customer_state (All)

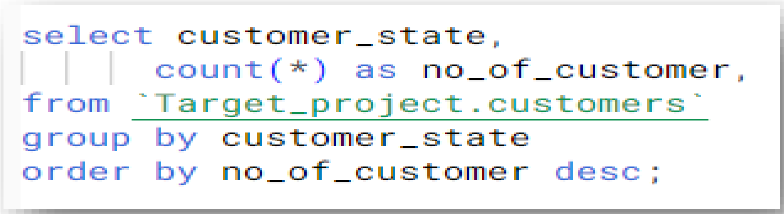
Row Labels	Sum of no_of_order_placed
January	8069
February	8508
March	9893
April	9343
May	10573
June	9412
July	10318
August	10843
September	4305
October	4959
November	7544
December	5674
Grand Total	**99441**



Row	customer_state	Month	no_of_order_placed
19	MG	August	1177
20	RJ	February	1176
21	RJ	April	1172
22	RJ	June	1128
23	MG	July	1111
24	MG	June	1080
25	MG	February	1063
26	MG	April	1061
27	RJ	November	1048

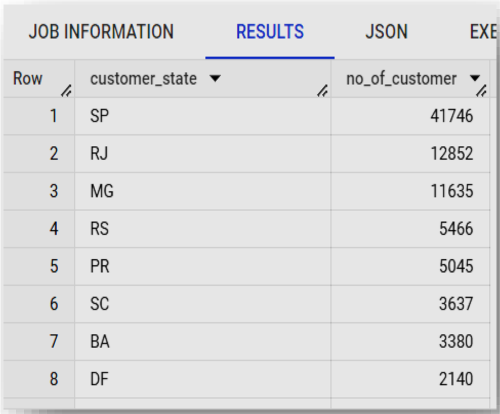
3.2 - How are the customers distributed across all the states?

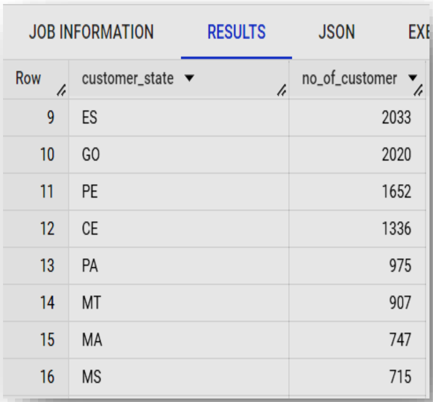
Query: -



```
select customer_state,
       count(*) as no_of_customer,
from `Target_project.customers`
group by customer_state
order by no_of_customer desc;
```

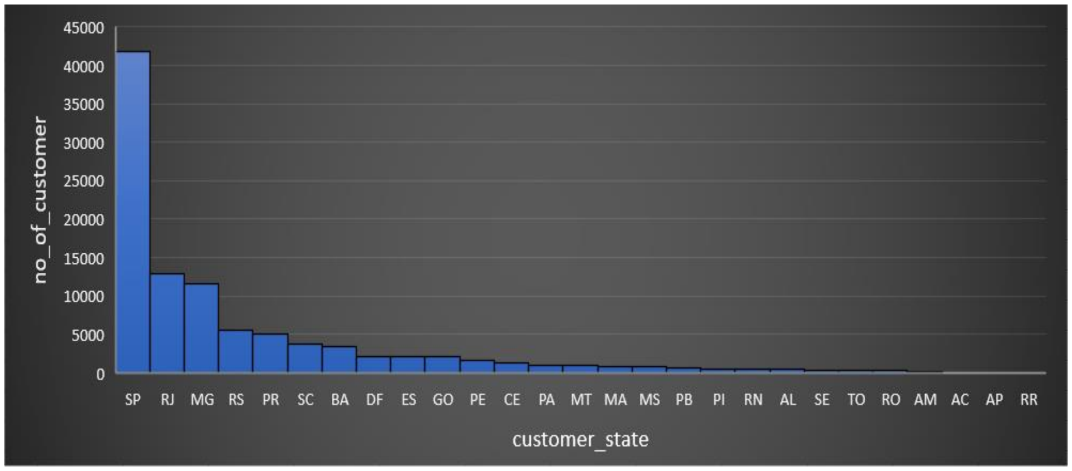
Result: -





Row	customer_state	no_of_customer
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
14	MT	907
15	MA	747
16	MS	715

Graph Table (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).

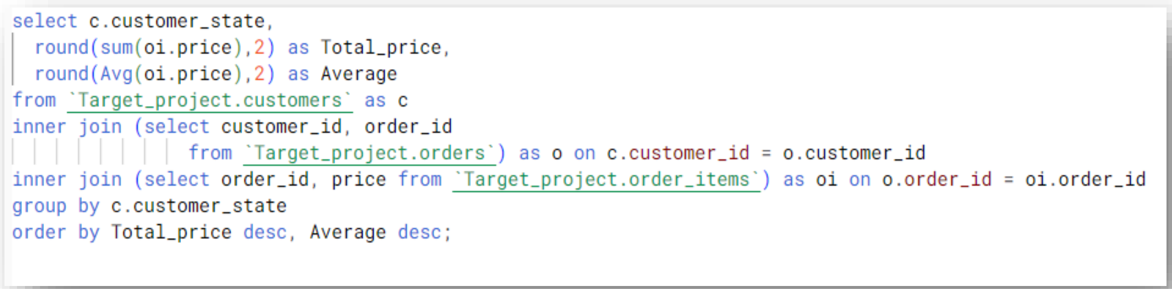
You can use the "payment value" column in the payments table to get the cost of orders.

Query: -

Result: -

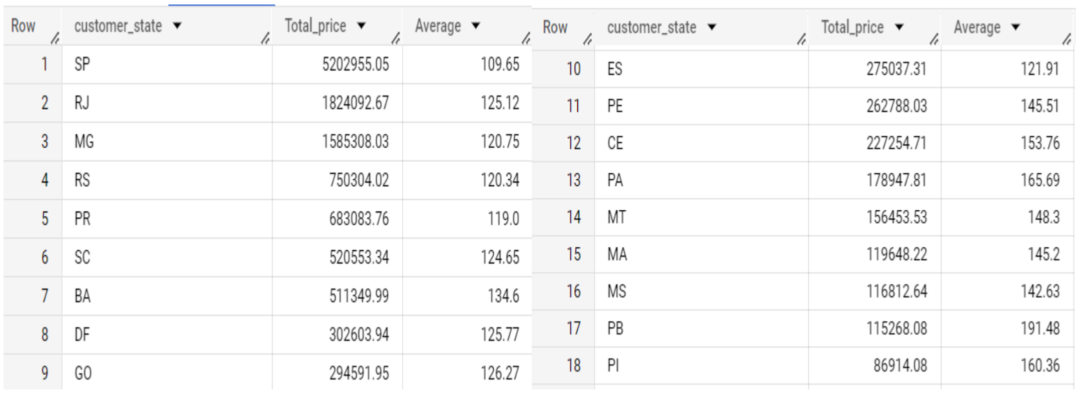
4.2 - Calculate the Total & Average value of order price for each state.

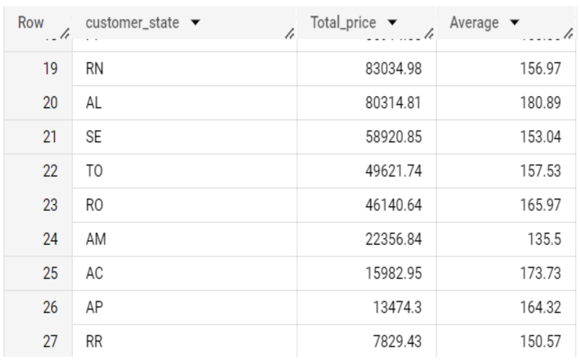
Query: -



```
select c.customer_state,
  round(sum(oi.price),2) as Total_price,
  round(Avg(oi.price),2) as Average
from `Target_project.customers` as c
inner join (select customer_id, order_id
                from `Target_project.orders`) as o on c.customer_id = o.customer_id
inner join (select order_id, price from `Target_project.order_items`) as oi on o.order_id = oi.order_id
group by c.customer_state
order by Total_price desc, Average desc;
```

Result: -





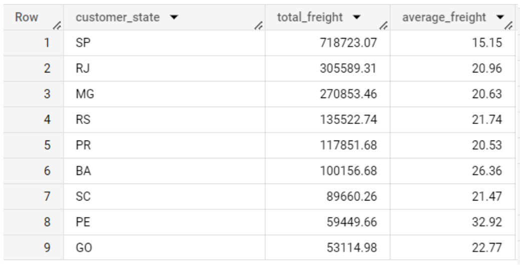
Row	customer_state	Total_price	Average
1	SP	5202955.05	109.65
2	RJ	1824092.67	125.12
3	MG	1585308.03	120.75
4	RS	750304.02	120.34
5	PR	683083.76	119.0
6	SC	520553.34	124.65
7	BA	511349.99	134.6
8	DF	302603.94	125.77
9	GO	294591.95	126.27
10	ES	275037.31	121.91
11	PE	262788.03	145.51
12	CE	227254.71	153.76
13	PA	178947.81	165.69
14	MT	156453.53	148.3
15	MA	119648.22	145.2
16	MS	116812.64	142.63
17	PB	115268.08	191.48
18	PI	86914.08	160.36
19	RN	83034.98	156.97
20	AL	80314.81	180.89
21	SE	58920.85	153.04
22	TO	49621.74	157.53
23	RO	46140.64	165.97
24	AM	22356.84	135.5
25	AC	15982.95	173.73
26	AP	13474.3	164.32
27	RR	7829.43	150.57

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

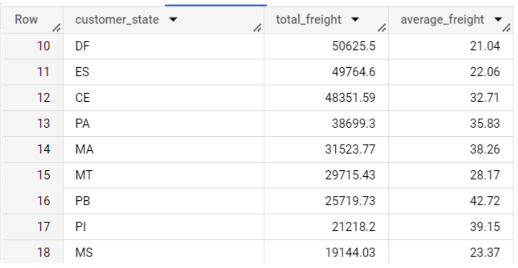
Query: -

```
select c.customer_state,
        round(sum(oi.freight_value),2) as total_freight,
        round(avg(oi.freight_value),2) as average_freight
from `Target_project.customers` as c
inner join (select customer_id, order_id
from `Target_project.orders`) as ord
on c.customer_id = ord.customer_id
inner join(select order_id, freight_value
from `Target_project.order_items`) as oi
on ord.order_id = oi.order_id
group by c.customer_state
order by total_freight desc, average_freight desc;
```

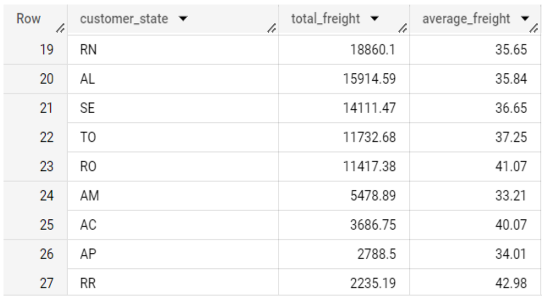
Result: -



Row	customer_state	total_freight	average_freight
1	SP	718723.07	15.15
2	RJ	305589.31	20.96
3	MG	270853.46	20.63
4	RS	135522.74	21.74
5	PR	117851.68	20.53
6	BA	100156.68	26.36
7	SC	89660.26	21.47
8	PE	59449.66	32.92
9	GO	53114.98	22.77



Row	customer_state	total_freight	average_freight
10	DF	50625.5	21.04
11	ES	49764.6	22.06
12	CE	48351.59	32.71
13	PA	38699.3	35.83
14	MA	31523.77	38.26
15	MT	29715.43	28.17
16	PB	25719.73	42.72
17	PI	21218.2	39.15
18	MS	19144.03	23.37



Row	customer_state	total_freight	average_freight
19	RN	18860.1	35.65
20	AL	15914.59	35.84
21	SE	14111.47	36.65
22	TO	11732.68	37.25
23	RO	11417.38	41.07
24	AM	5478.89	33.21
25	AC	3686.75	40.07
26	AP	2788.5	34.01
27	RR	2235.19	42.98

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.

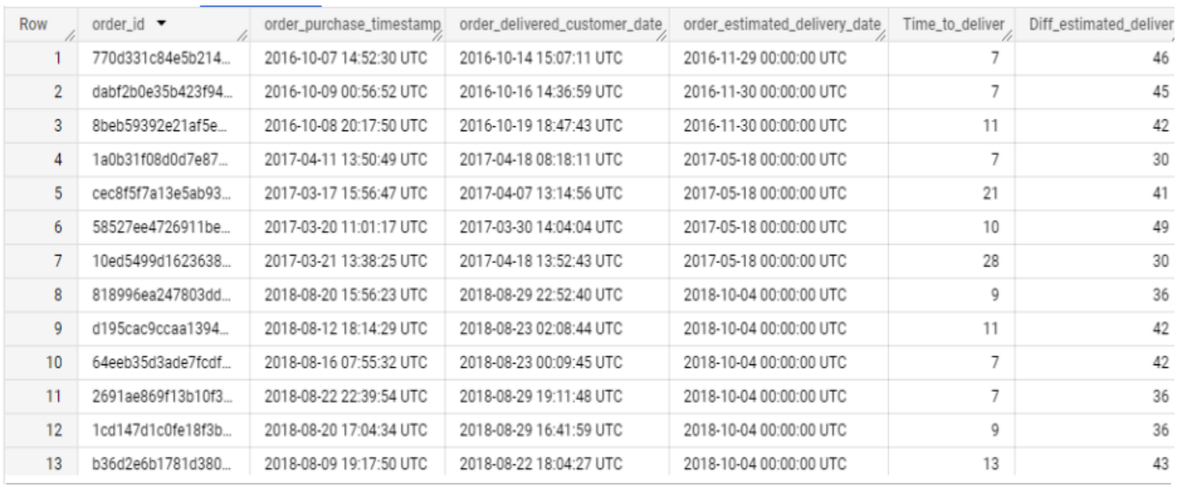
You can calculate the delivery time and the difference between the estimated & actual delivery date using the given formula:

- **time_to_deliver** = order_delivered_customer_date - order_purchase_timestamp
- **diff_estimated_delivery** = order_estimated_delivery_date order_delivered_customer_date.

Query: -

```
SELECT order_id,
       order_purchase_timestamp,
       order_delivered_customer_date,
       order_estimated_delivery_date,
      abs(date_diff(cast(order_purchase_timestamp as
      date),cast(order_delivered_customer_date as date),day)) as
      Time_to_deliver,
      abs(date_diff(cast(order_delivered_customer_date as
      date),cast(order_estimated_delivery_date as date),day)) as
      Diff_estimated_delivery
FROM `sql-scaler-projects.business_cs.orders`;
```

Result: -



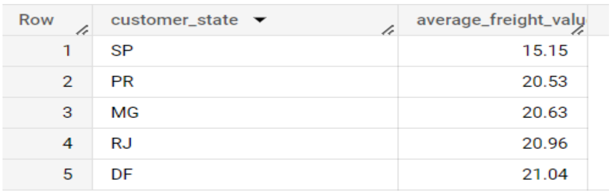
Row	order_id	order_purchase_timestamp	order_delivered_customer_date	order_estimated_delivery_date	Time_to_deliver	Diff_estimated_deliver
1	770d331c84e5b214...	2016-10-07 14:52:30 UTC	2016-10-14 15:07:11 UTC	2016-11-29 00:00:00 UTC	7	46
2	dabf2b0e35b423f94...	2016-10-09 00:56:52 UTC	2016-10-16 14:36:59 UTC	2016-11-30 00:00:00 UTC	7	45
3	8beb59392e21af5e...	2016-10-08 20:17:50 UTC	2016-10-19 18:47:43 UTC	2016-11-30 00:00:00 UTC	11	42
4	1a0b31f08d0d7e87...	2017-04-11 13:50:49 UTC	2017-04-18 08:18:11 UTC	2017-05-18 00:00:00 UTC	7	30
5	cec8f5f7a13e5ab93...	2017-03-17 15:56:47 UTC	2017-04-07 13:14:56 UTC	2017-05-18 00:00:00 UTC	21	41
6	58527ee4726911be...	2017-03-20 11:01:17 UTC	2017-03-30 14:04:04 UTC	2017-05-18 00:00:00 UTC	10	49
7	10ed5499d1623638...	2017-03-21 13:38:25 UTC	2017-04-18 13:52:43 UTC	2017-05-18 00:00:00 UTC	28	30
8	818996ea247803dd...	2018-08-20 15:56:23 UTC	2018-08-29 22:52:40 UTC	2018-10-04 00:00:00 UTC	9	36
9	d195cac9ccaa1394...	2018-08-12 18:14:29 UTC	2018-08-23 02:08:44 UTC	2018-10-04 00:00:00 UTC	11	42
10	64eeb35d3ade7fcdf...	2018-08-16 07:55:32 UTC	2018-08-23 00:09:45 UTC	2018-10-04 00:00:00 UTC	7	42
11	2691ae869f13b10f3...	2018-08-22 22:39:54 UTC	2018-08-29 19:11:48 UTC	2018-10-04 00:00:00 UTC	7	36
12	1cd147d1c0fe18f3b...	2018-08-20 17:04:34 UTC	2018-08-29 16:41:59 UTC	2018-10-04 00:00:00 UTC	9	36
13	b36d2e6b1781d380...	2018-08-09 19:17:50 UTC	2018-08-22 18:04:27 UTC	2018-10-04 00:00:00 UTC	13	43

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

Query: - Top 5 states with the lowest average freight value.

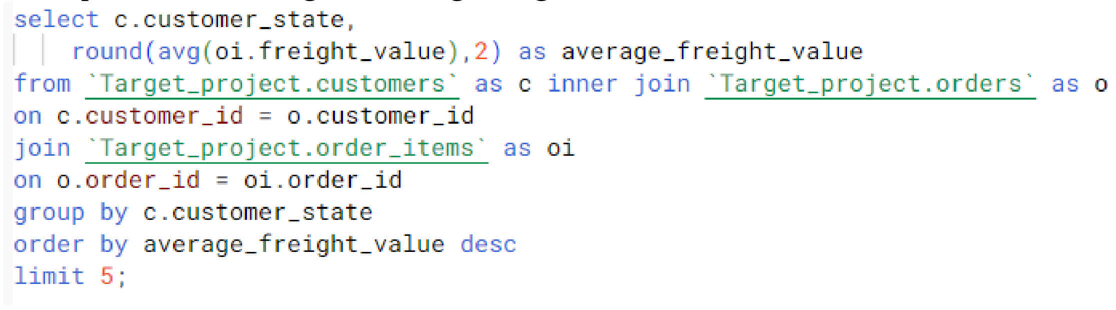
```
select c.customer_state,
    round(avg(oi.freight_value),2) as average_freight_value
from `Target_project.customers` as c inner join `Target_project.orders` as o
on c.customer_id = o.customer_id
join `Target_project.order_items` as oi
on o.order_id = oi.order_id
group by c.customer_state
order by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

Result: -



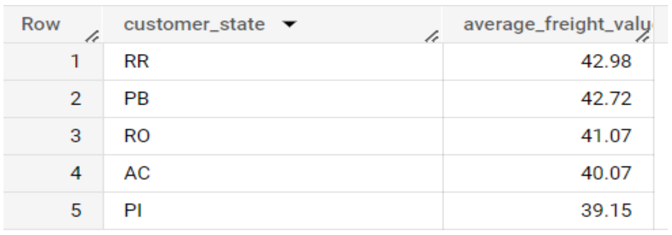
Row	customer_state	average_freight_valu
1	SP	15.15
2	PR	20.53
3	MG	20.63
4	RJ	20.96
5	DF	21.04

Query: - Top 5 states with highest average freight value.



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

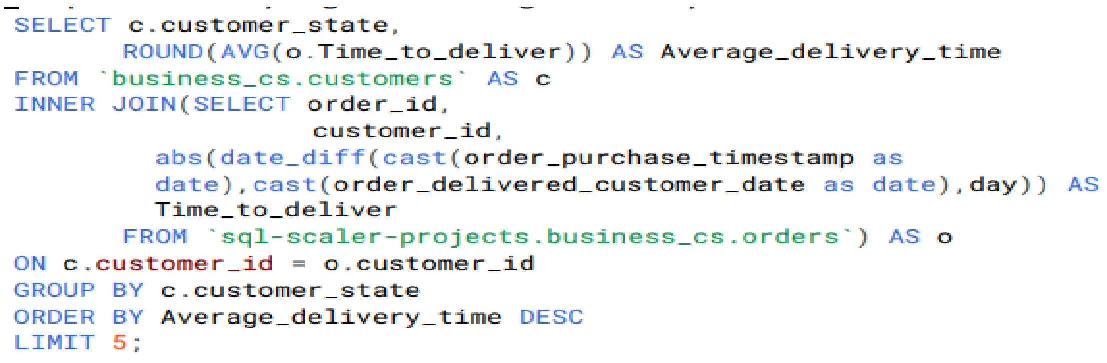
Result: -



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

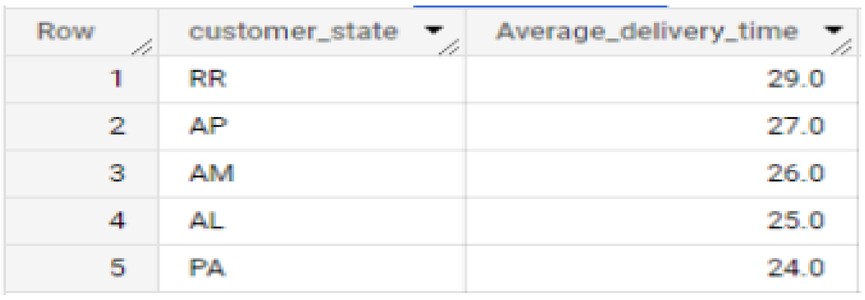
5.3 - Find out the top 5 states with the highest & lowest average delivery time.

Query: - Top 5 states by highest Average delivery time



```
SELECT c.customer_state,
       ROUND(AVG(o.Time_to_deliver)) AS Average_delivery_time
FROM `business_cs.customers` AS c
INNER JOIN(SELECT order_id,
                  customer_id,
           abs(date_diff(cast(order_purchase_timestamp as
           date),cast(order_delivered_customer_date as date),day)) AS
           Time_to_deliver
       FROM `sql-scaler-projects.business_cs.orders`) AS o
ON c.customer_id = o.customer_id
GROUP BY c.customer_state
ORDER BY Average_delivery_time DESC
LIMIT 5;
```

Result: -

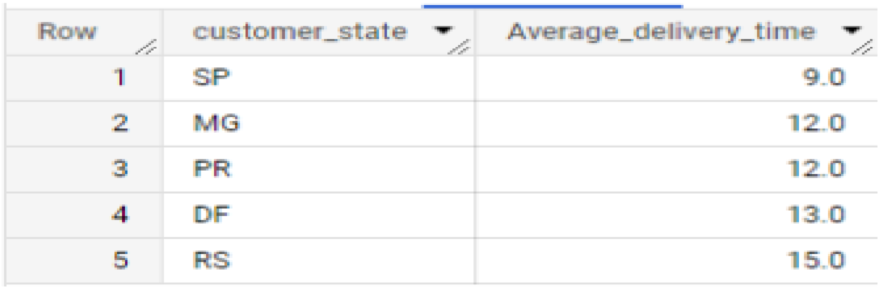


Row	customer_state	Average_delivery_time
1	RR	29.0
2	AP	27.0
3	AM	26.0
4	AL	25.0
5	PA	24.0

Query: - Top 5 states lowest Average delivery time –

```
SELECT c.customer_state,
       ROUND(AVG(o.Time_to_deliver)) AS Average_delivery_time
FROM `business_cs.customers` AS c
INNER JOIN(SELECT order_id,
                  customer_id,
        abs(date_diff(cast(order_purchase_timestamp as
        date),cast(order_delivered_customer_date as date),day)) AS
        Time_to_deliver
       FROM `sql-scaler-projects.business_cs.orders`) AS o
ON c.customer_id = o.customer_id
GROUP BY c.customer_state
ORDER BY Average_delivery_time ASC
LIMIT 5;
```

Result: -



Row	customer_state	Average_delivery_time
1	SP	9.0
2	MG	12.0
3	PR	12.0
4	DF	13.0
5	RS	15.0

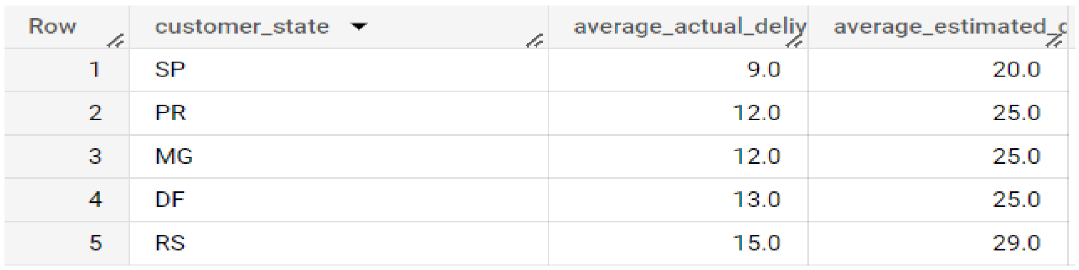
5.4 – Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery.

You can use the difference between the averages of actual & estimated delivery date to figure out how fast the delivery was for each state.

Query: -

```
select c.customer_state,
    round(avg(o.actual_delivery_time)) as average_actual_delivery_time,
    round(avg(o.estimated_delivery_time)) as average_estimated_delivery_time
from `Target_project.customers` as c
inner join(select customer_id, abs(date_diff(cast(order_delivered_customer_date as date),
 cast(order_purchase_timestamp as date), day)) as actual_delivery_time,
abs(date_diff(cast(order_purchase_timestamp as date), cast(order_estimated_delivery_date as date), day))
 as estimated_delivery_time from `Target_project.orders`) as o
on c.customer_id = o.customer_id
group by c.customer_state
having avg (o.actual_delivery_time) < avg(o.estimated_delivery_time)
order by average_actual_delivery_time
limit 5;
```

Result: -

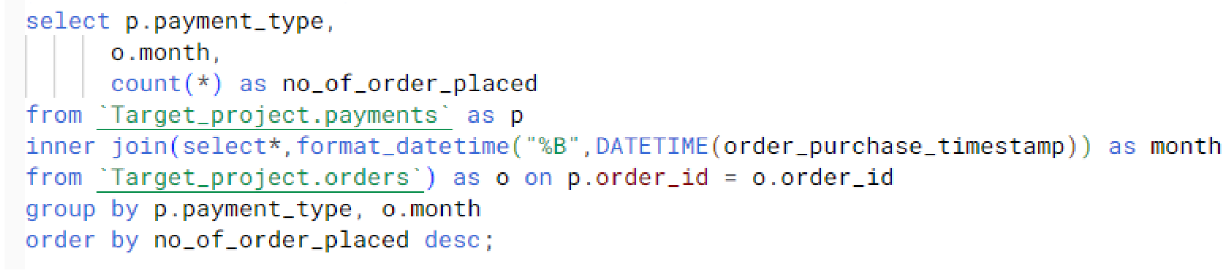


Row	customer_state	average_actual_deliv	average_estimated_c
1	SP	9.0	20.0
2	PR	12.0	25.0
3	MG	12.0	25.0
4	DF	13.0	25.0
5	RS	15.0	29.0

6 - Analysis based on the payments:

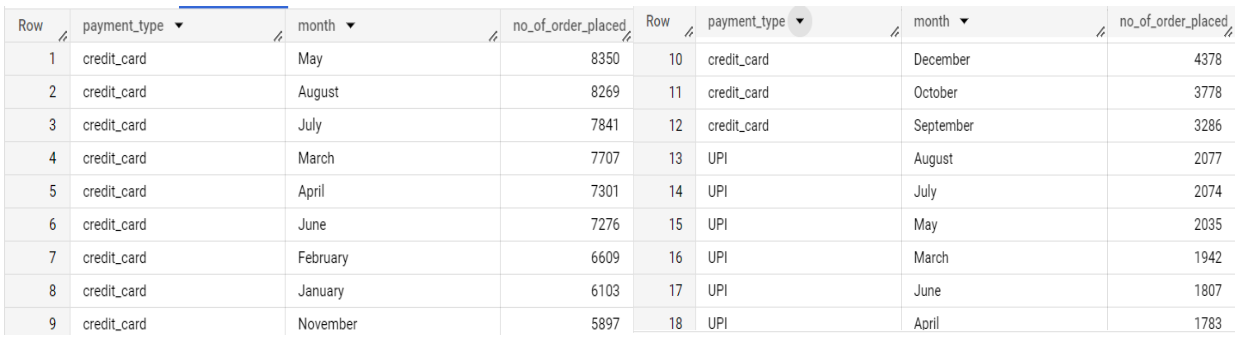
6.1 **-** Find the month-on-month no. of orders placed using different payment types.

Query: -



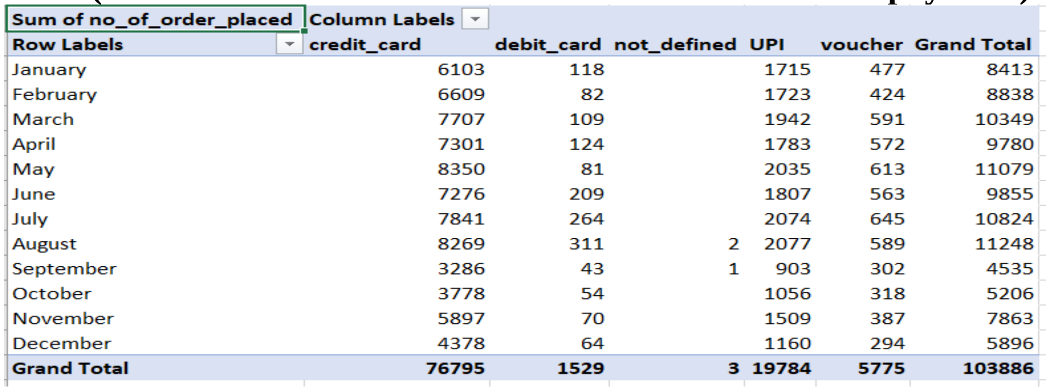
```
select p.payment_type,
       o.month,
       count(*) as no_of_order_placed
from `Target_project.payments` as p
inner join(select*,format_datetime("%B",DATETIME(order_purchase_timestamp)) as month
from `Target_project.orders`) as o on p.order_id = o.order_id
group by p.payment_type, o.month
order by no_of_order_placed desc;
```

Result: -



Row	payment_type	month	no_of_order_placed
1	credit_card	May	8350
2	credit_card	August	8269
3	credit_card	July	7841
4	credit_card	March	7707
5	credit_card	April	7301
6	credit_card	June	7276
7	credit_card	February	6609
8	credit_card	January	6103
9	credit_card	November	5897
10	credit_card	December	4378
11	credit_card	October	3778
12	credit_card	September	3286
13	UPI	August	2077
14	UPI	July	2074
15	UPI	May	2035
16	UPI	March	1942
17	UPI	June	1807
18	UPI	April	1783

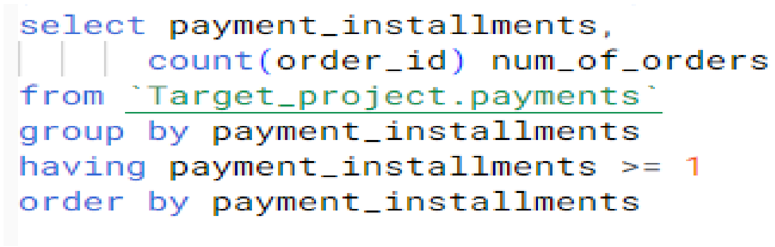
Pivot Table (Month wise total transaction with different mode of payment): -



Sum of no_of_order_placed	Column Labels					
Row Labels	credit_card	debit_card	not_defined	UPI	voucher	Grand Total
January	6103	118		1715	477	8413
February	6609	82		1723	424	8838
March	7707	109		1942	591	10349
April	7301	124		1783	572	9780
May	8350	81		2035	613	11079
June	7276	209		1807	563	9855
July	7841	264		2074	645	10824
August	8269	311	2	2077	589	11248
September	3286	43	1	903	302	4535
October	3778	54		1056	318	5206
November	5897	70		1509	387	7863
December	4378	64		1160	294	5896
Grand Total	**76795**	**1529**	**3**	**19784**	**5775**	**103886**

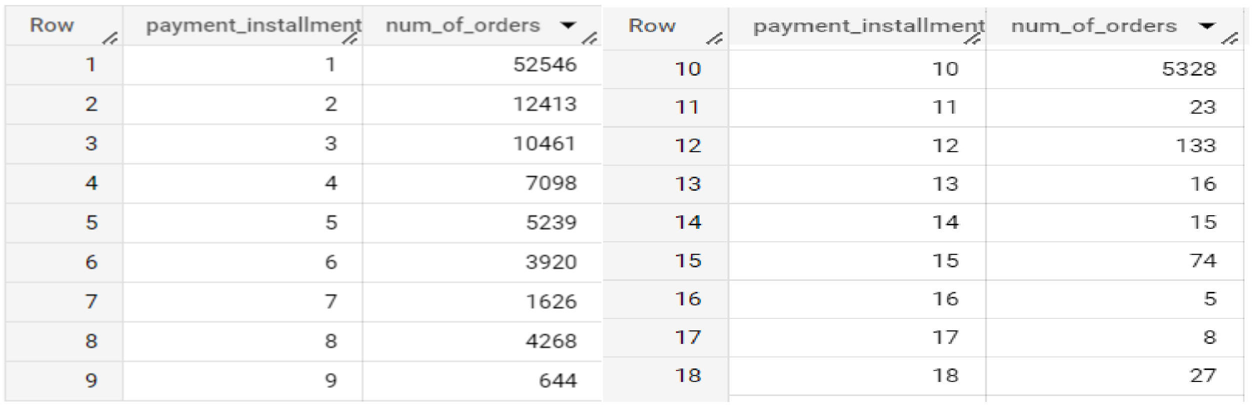
6.2 - Find the no. of orders placed on the basis of the payment installments that have been paid.

Query: -



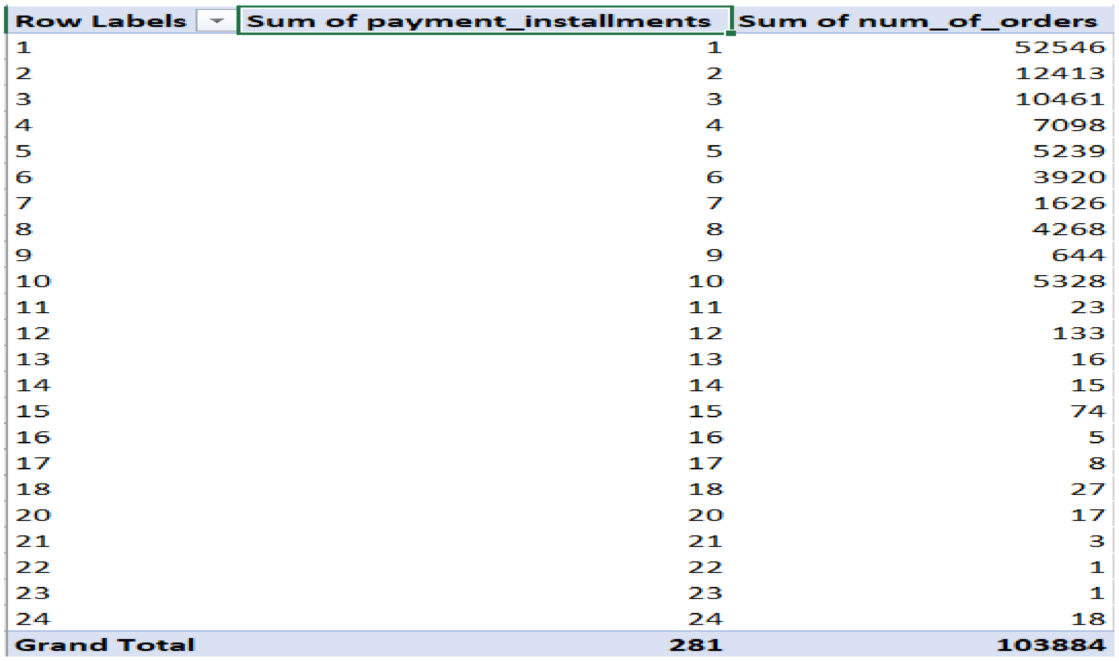
```
select payment_installments,
       count(order_id) num_of_orders
from `Target_project.payments`
group by payment_installments
having payment_installments >= 1
order by payment_installments
```

Result: -



Row	payment_installment	num_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
13	13	16
14	14	15
15	15	74
16	16	5
17	17	8
18	18	27

Pivot Table (shows total no payment_installments and num_of_orders): -



Row Labels	Sum of payment_installments	Sum of num_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
13	13	16
14	14	15
15	15	74
16	16	5
17	17	8
18	18	27
20	20	17
21	21	3
22	22	1
23	23	1
24	24	18
Grand Total	**281**	**103884**