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

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

customer_id=String-VARCHAR

customer_unique_id=String-VARCHAR

customer_zip_code_prefix=Integer

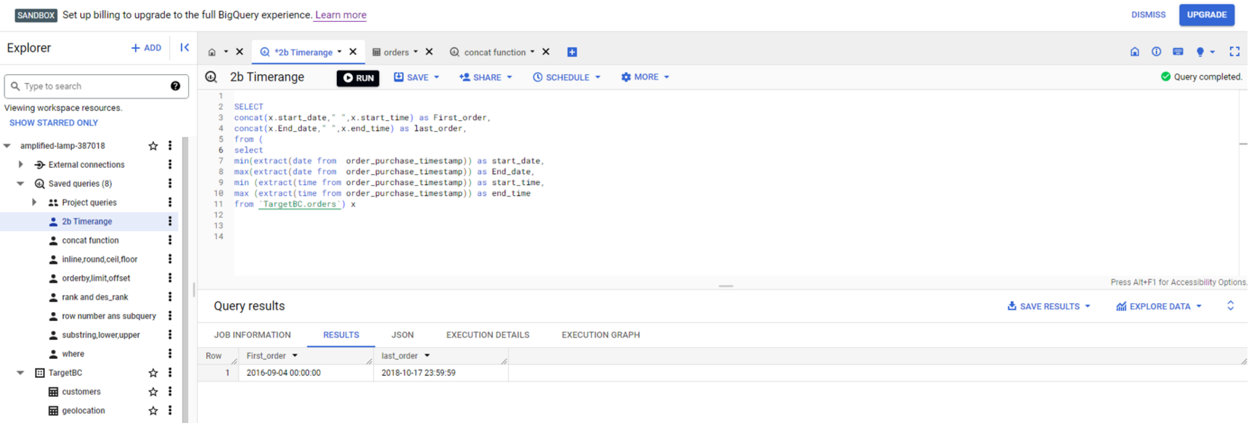
Customer_city= String -CHAR(3)

customer_state= String -CHAR (2)

Insight/Recommendation : Datatype String with VARCHAR,CHAR is used along with Integer

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

```
SELECT
concat(x.start_date," ",x.start_time) as First_order,
concat(x.End_date," ",x.end_time) as last_order,
from (
select
min(extract(date from  order_purchase_timestamp)) as start_date,
max(extract(date from  order_purchase_timestamp)) as End_date,
min (extract(time from order_purchase_timestamp)) as start_time,
max (extract(time from order_purchase_timestamp)) as end_time
from `TargetBC.orders`) x
```

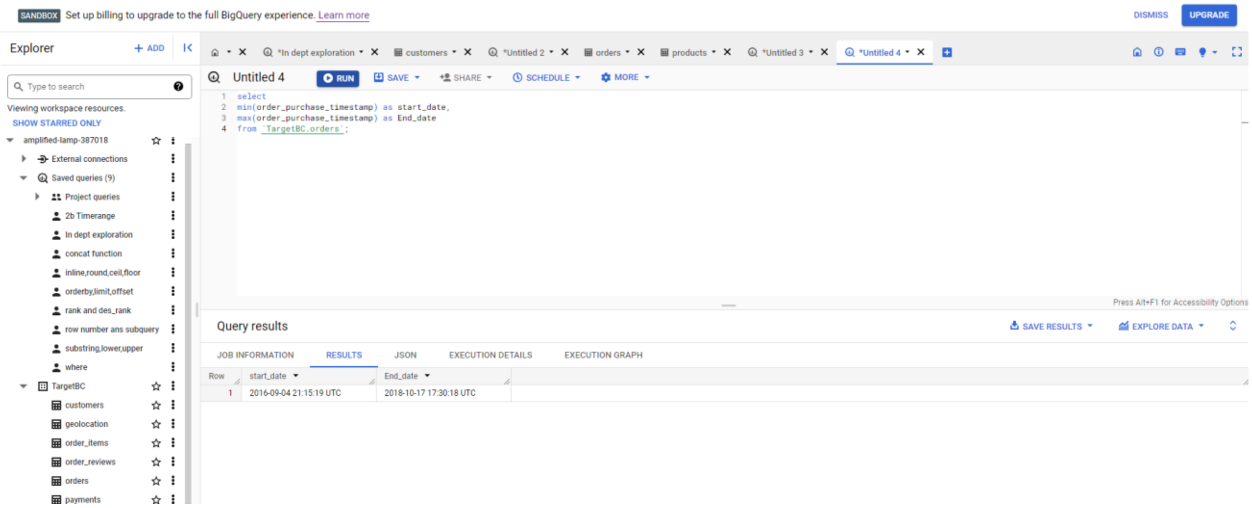


```
SELECT
concat(x.start_date," ",x.start_time) as First_order,
concat(x.End_date," ",x.end_time) as last_order,
from (
select
min(extract(date from  order_purchase_timestamp)) as start_date,
max(extract(date from  order_purchase_timestamp)) as End_date,
min (extract(time from order_purchase_timestamp)) as start_time,
max (extract(time from order_purchase_timestamp)) as end_time
from `TargetBC.orders`) x
```

Row	First_order	last_order
1	2016-09-04 00:00:00	2018-10-17 23:59:59

Or

```
select
min(order_purchase_timestamp) as start_date,
max(order_purchase_timestamp) as End_date
from `TargetBC.orders`;
```



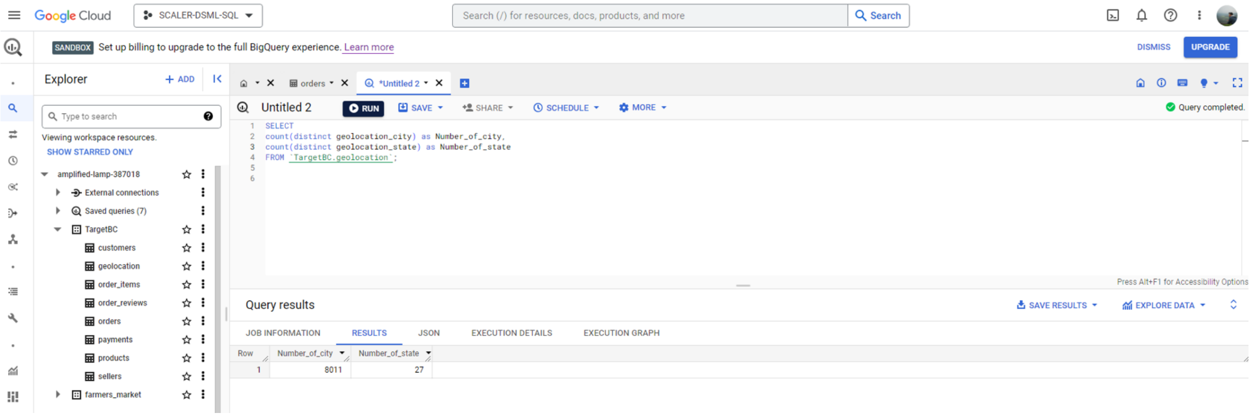
```
select
min(order_purchase_timestamp) as start_date,
max(order_purchase_timestamp) as End_date
from `TargetBC.orders`;
```

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

Insight : Time range for give dataset found to be in between Sept-2016 to Oct-2018

c. Count the number of Cities and States in our dataset.

```
SELECT
count(distinct geolocation_city) as Number_of_city,
count(distinct geolocation_state) as Number_of_state
FROM `TargetBC.geolocation`;
```



```
SELECT
count(distinct geolocation_city) as Number_of_city,
count(distinct geolocation_state) as Number_of_state
FROM `TargetBC.geolocation`;
```

Row	Number_of_city	Number_of_state
1	8011	27

Insight/Recommendation: Target has spectrum of order from across all state 27 states and 8011 cities covering all region within brazil.

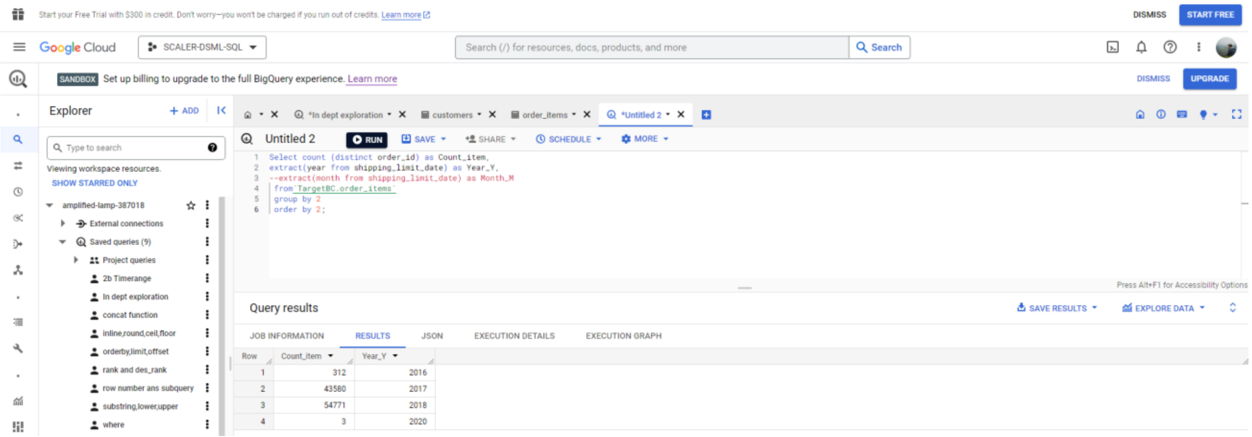
2. In-depth Exploration:

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

For Year :

```
 Select count (distinct order_id) as Count_item,

extract(year from shipping_limit_date) as Year_Y,
from`TargetBC.order_items`
 group by 2
 order by 2;
```



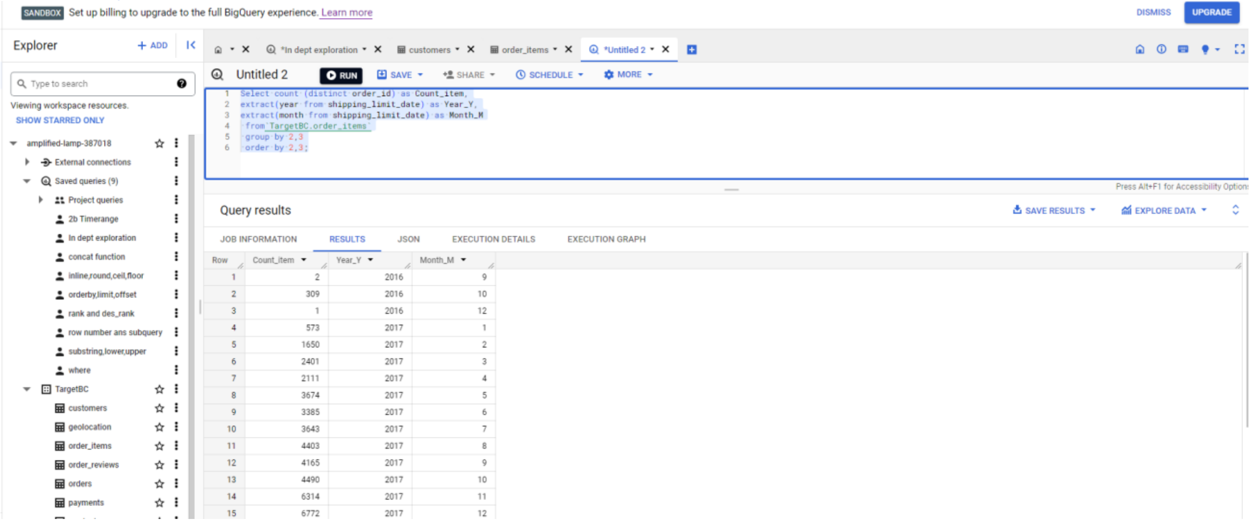
```
Select count (distinct order_id) as Count_item,
extract(year from shipping_limit_date) as Year_Y,
--extract(month from shipping_limit_date) as Month_M
 from`TargetBC.order_items`
 group by 2
 order by 2;
```

Row	Count_item	Year_Y
1	312	2016
2	43580	2017
3	54771	2018
4	3	2020

Insight/Recommendation : In past year 2017,2018 was significant growth but in recent year 2019 onwards it has sharp downfall in number of orders.

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

```
Select count (distinct order_id) as Count_item,
extract(year from shipping_limit_date) as Year_Y,
extract(month from shipping_limit_date) as Month_M
 from`TargetBC.order_items`
 group by 2,3
 order by 2,3;
```

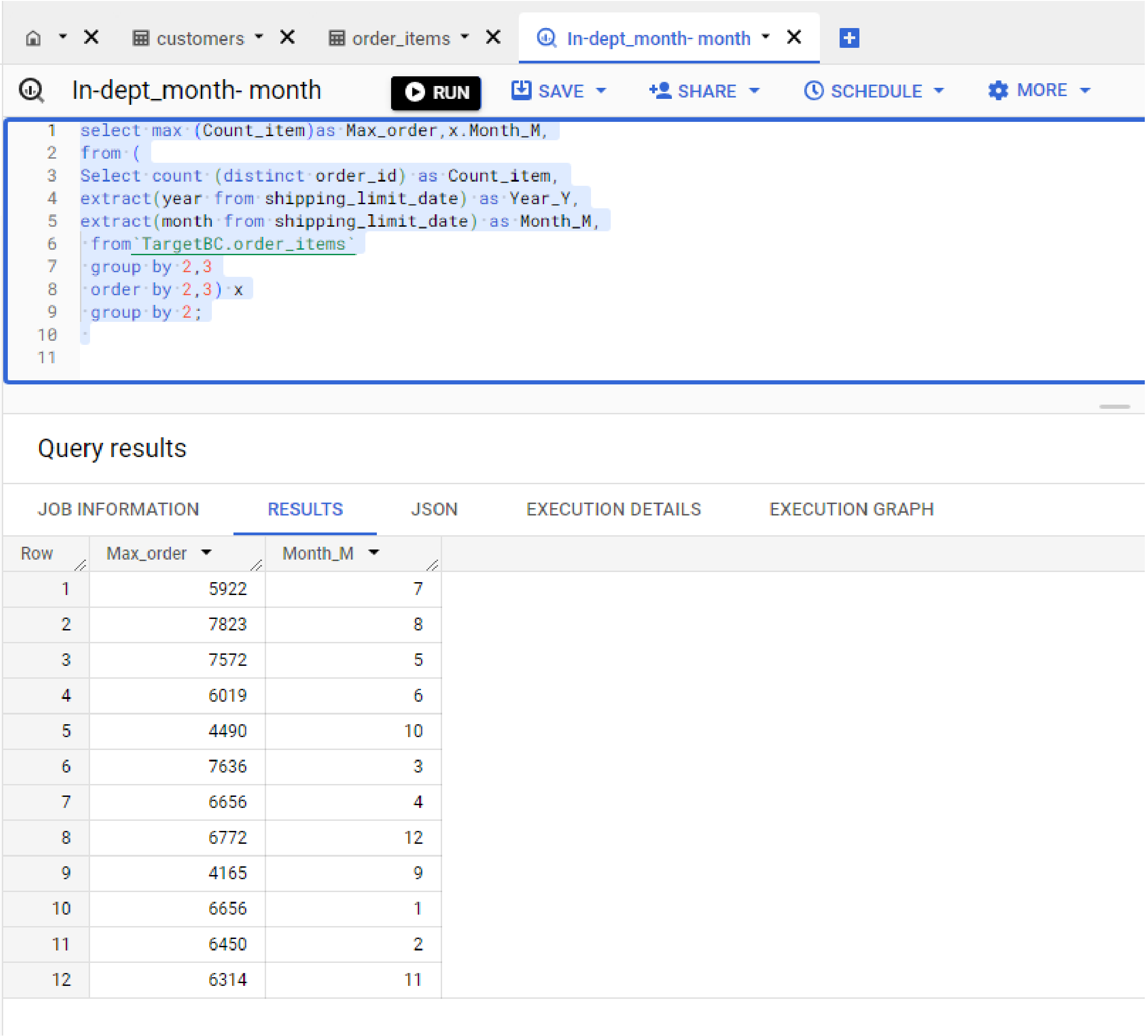


```
Select count (distinct order_id) as Count_item,
extract(year from shipping_limit_date) as Year_Y,
extract(month from shipping_limit_date) as Month_M
 from`TargetBC.order_items`
 group by 2,3
 order by 2,3;
```

Row	Count_item	Year_Y	Month_M
1	2	2016	9
2	309	2016	10
3	1	2016	12
4	573	2017	1
5	1650	2017	2
6	2401	2017	3
7	2111	2017	4
8	3674	2017	5
9	3385	2017	6
10	3643	2017	7
11	4403	2017	8
12	4165	2017	9
13	4490	2017	10
14	6314	2017	11
15	6772	2017	12

And

```
select max (Count_item)as Max_order,x.Month_M,
from (
Select count (distinct order_id) as Count_item,
extract(year from shipping_limit_date) as Year_Y,
extract(month from shipping_limit_date) as Month_M,
 from`TargetBC.order_items`
 group by 2,3
 order by 2,3) x
 group by 2;
```



```
select max (Count_item)as Max_order,x.Month_M,
from (
Select count (distinct order_id) as Count_item,
extract(year from shipping_limit_date) as Year_Y,
extract(month from shipping_limit_date) as Month_M,
 from`TargetBC.order_items`
 group by 2,3
 order by 2,3) x
 group by 2;
```

Row	Max_order	Month_M
1	5922	7
2	7823	8
3	7572	5
4	6019	6
5	4490	10
6	7636	3
7	6656	4
8	6772	12
9	4165	9
10	6656	1
11	6450	2
12	6314	11

Insight/Recommendation : Max order trend is shown in month summer start from Dec to March,and then in winter session June to Aug.

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

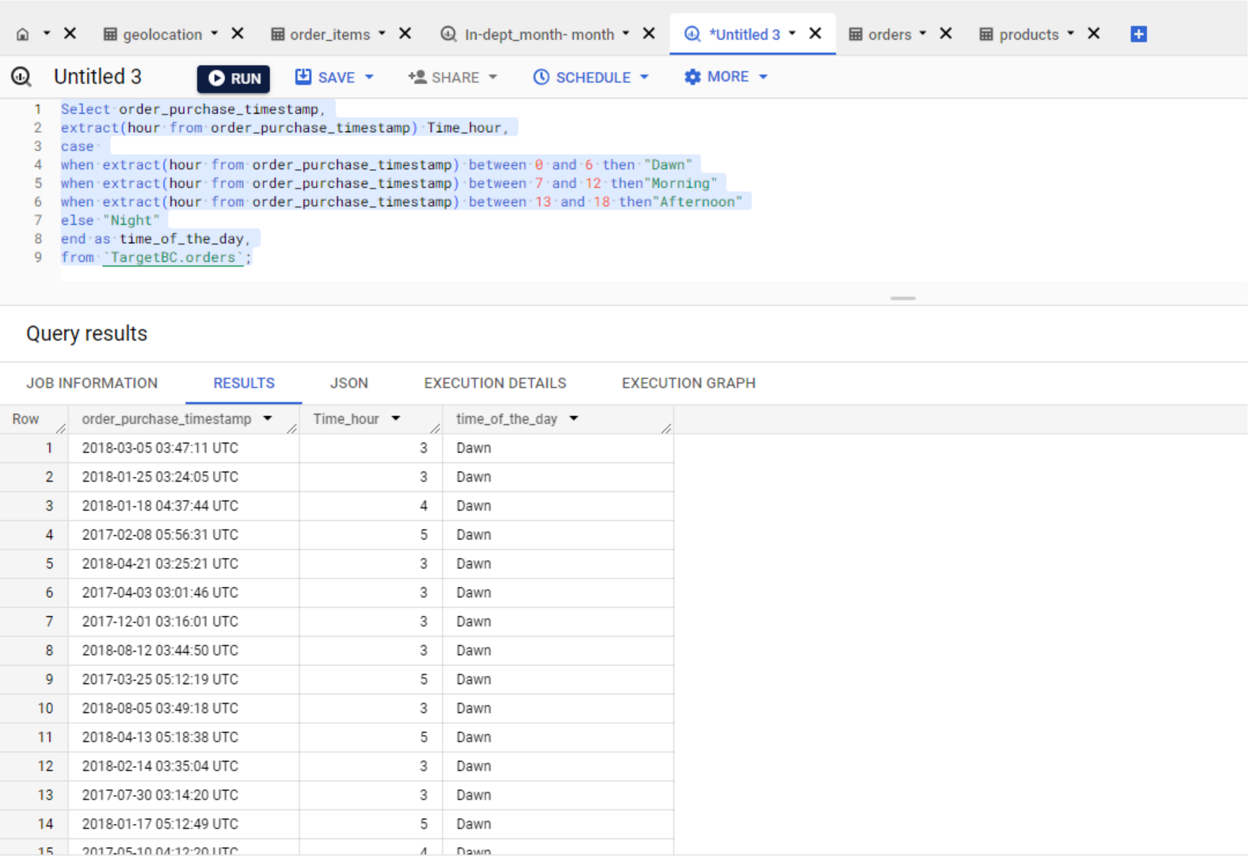
0-6 hrs : Dawn

7-12 hrs : Mornings

13-18 hrs : Afternoon

19-23 hrs : Night

```
Select order_purchase_timestamp,
extract(hour from order_purchase_timestamp) Time_hour,
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"
else "Night"
end as time_of_the_day,
from `TargetBC.orders`;
```



```
Select order_purchase_timestamp,
extract(hour from order_purchase_timestamp) Time_hour,
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"
else "Night"
end as time_of_the_day,
from `TargetBC.orders`;
```

Query results

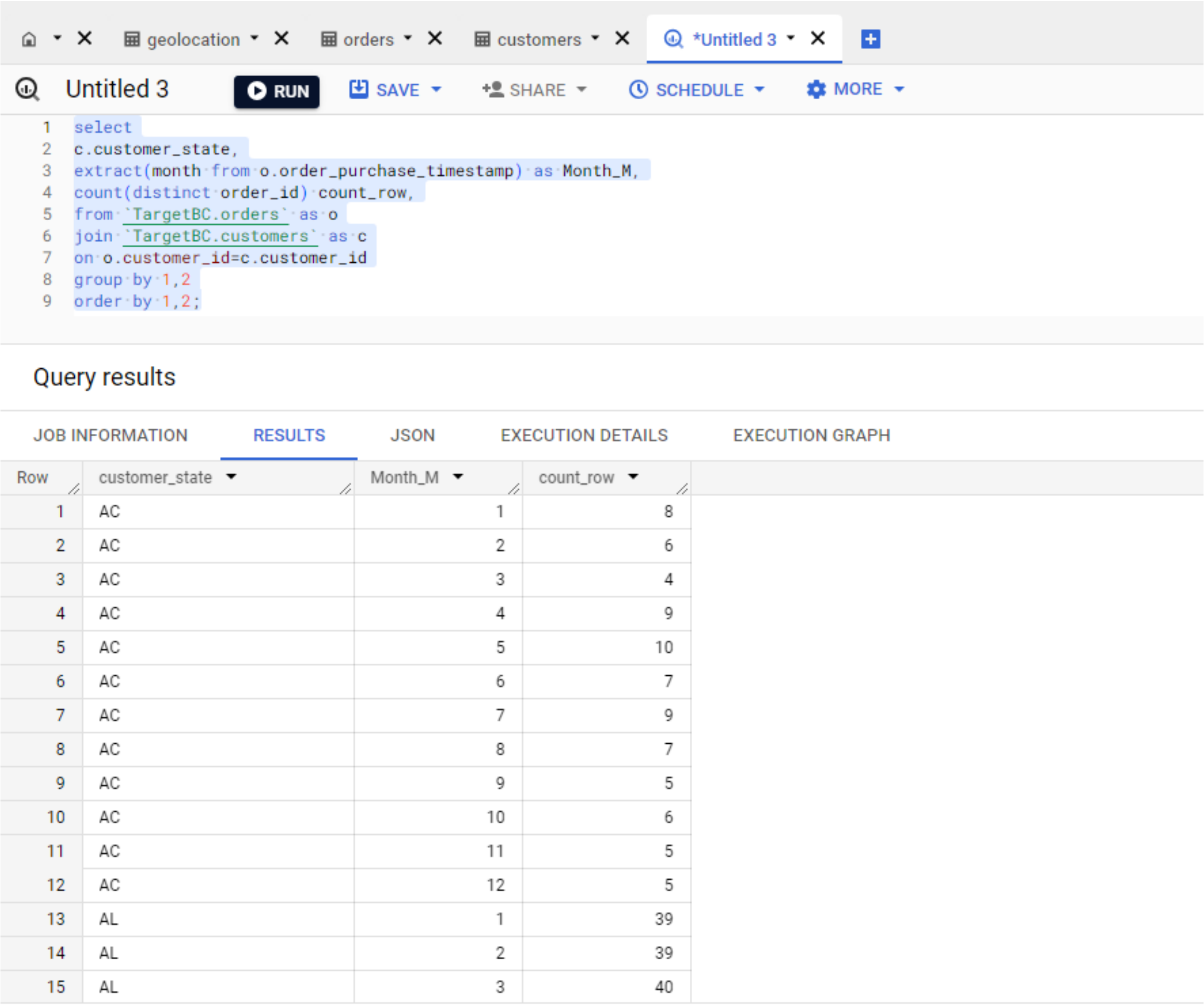
Row	order_purchase_timestamp	Time_hour	time_of_the_day
1	2018-03-05 03:47:11 UTC	3	Dawn
2	2018-01-25 03:24:05 UTC	3	Dawn
3	2018-01-18 04:37:44 UTC	4	Dawn
4	2017-02-08 05:56:31 UTC	5	Dawn
5	2018-04-21 03:25:21 UTC	3	Dawn
6	2017-04-03 03:01:46 UTC	3	Dawn
7	2017-12-01 03:16:01 UTC	3	Dawn
8	2018-08-12 03:44:50 UTC	3	Dawn
9	2017-03-25 05:12:19 UTC	5	Dawn
10	2018-08-05 03:49:18 UTC	3	Dawn
11	2018-04-13 05:18:38 UTC	5	Dawn
12	2018-02-14 03:35:04 UTC	3	Dawn
13	2017-07-30 03:14:20 UTC	3	Dawn
14	2018-01-17 05:12:49 UTC	5	Dawn
15	2017-05-10 04:12:20 UTC	4	Dawn

Insight/Recommendation : Brazilian customers mostly place their orders in Night.

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

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

```
select
c.customer_state,
extract(month from o.order_purchase_timestamp) as Month_M,
count(distinct order_id) count_row,
from `TargetBC.orders` as o
join `TargetBC.customers` as c
on o.customer_id=c.customer_id
group by 1,2
order by 1,2;
```



```
select
c.customer_state,
extract(month from o.order_purchase_timestamp) as Month_M,
count(distinct order_id) count_row,
from `TargetBC.orders` as o
join `TargetBC.customers` as c
on o.customer_id=c.customer_id
group by 1,2
order by 1,2;
```

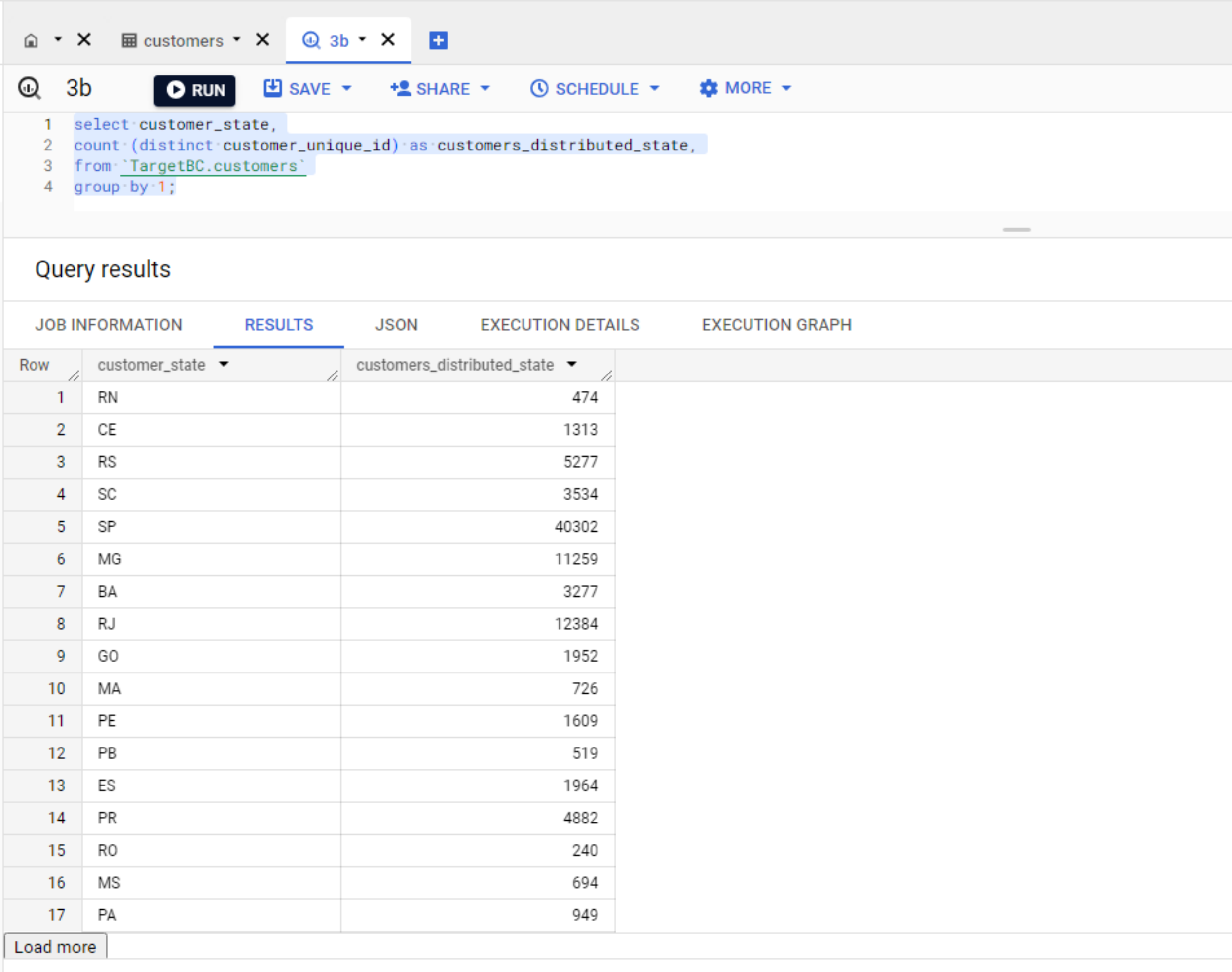
Query results

Row	customer_state	Month_M	count_row
1	AC	1	8
2	AC	2	6
3	AC	3	4
4	AC	4	9
5	AC	5	10
6	AC	6	7
7	AC	7	9
8	AC	8	7
9	AC	9	5
10	AC	10	6
11	AC	11	5
12	AC	12	5
13	AL	1	39
14	AL	2	39
15	AL	3	40

Insight/Recommendation : Brazilline state Bahia (BA) got highest number of Month-on Month order compared to all state.

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

```
select customer_state,
count (distinct customer_unique_id) as customers_distributed_state,
from `TargetBC.customers`
group by 1;
```



```
select customer_state,
count (distinct customer_unique_id) as customers_distributed_state,
from `TargetBC.customers`
group by 1;
```

Row	customer_state	customers_distributed_state
1	RN	474
2	CE	1313
3	RS	5277
4	SC	3534
5	SP	40302
6	MG	11259
7	BA	3277
8	RJ	12384
9	GO	1952
10	MA	726
11	PE	1609
12	PB	519
13	ES	1964
14	PR	4882
15	RO	240
16	MS	694
17	PA	949

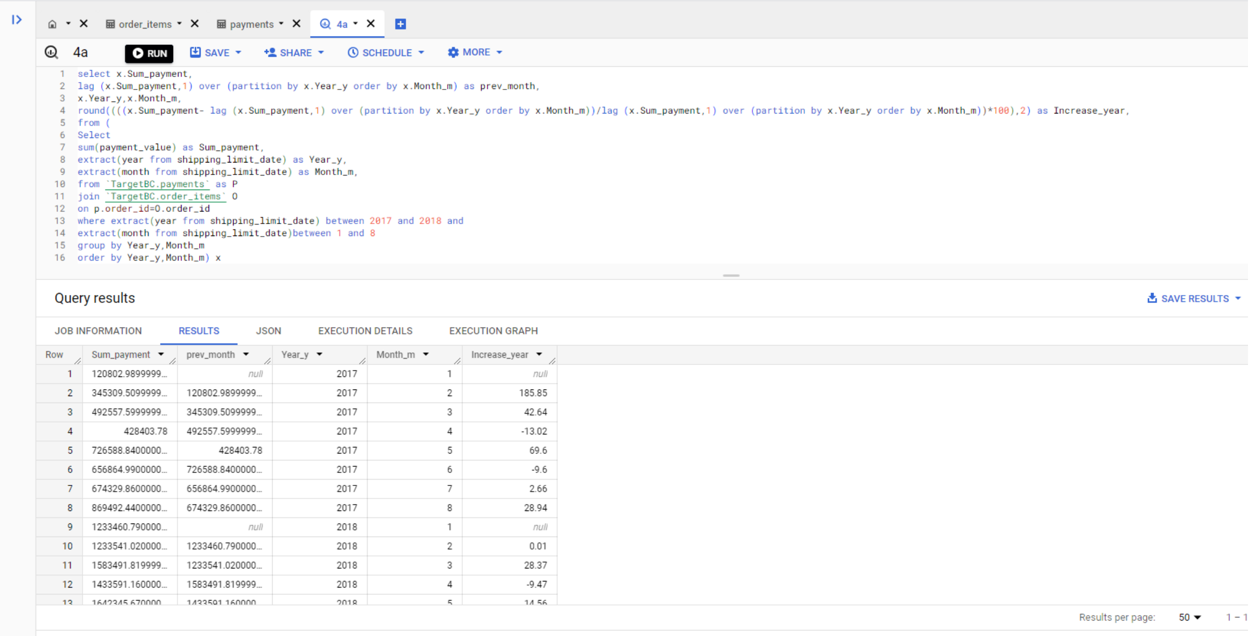
Insight/Recommendation : Customer distribution is more in North-east,south,South-east region state.

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

a. Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only).

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

```
select x.Sum_payment,
lag (x.Sum_payment,1) over (partition by x.Year_y order by x.Month_m) as prev_month,
x.Year_y,x.Month_m,
round((((x.Sum_payment- lag (x.Sum_payment,1) over (partition by x.Year_y order by
x.Month_m))/lag (x.Sum_payment,1) over (partition by x.Year_y order by
x.Month_m))*100),2) as Increase_year,
from (
Select
sum(payment_value) as Sum_payment,
extract(year from shipping_limit_date) as Year_y,
extract(month from shipping_limit_date) as Month_m,
from `TargetBC.payments` as P
join `TargetBC.order_items` O
on p.order_id=O.order_id
where extract(year from shipping_limit_date) between 2017 and 2018 and
extract(month from shipping_limit_date)between 1 and 8
group by Year_y,Month_m
order by Year_y,Month_m) x
```



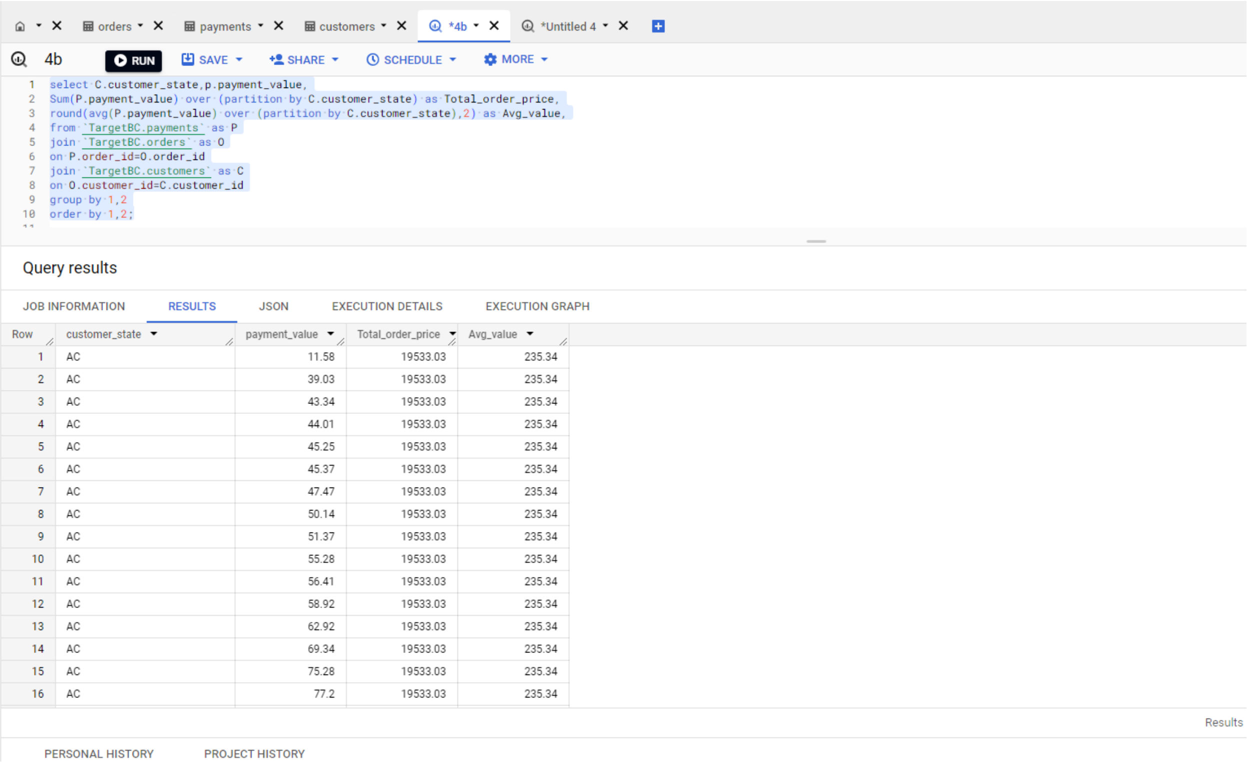
```
select x.Sum_payment,
lag (x.Sum_payment,1) over (partition by x.Year_y order by x.Month_m) as prev_month,
x.Year_y,x.Month_m,
round((((x.Sum_payment- lag (x.Sum_payment,1) over (partition by x.Year_y order by x.Month_m))/lag (x.Sum_payment,1) over (partition by x.Year_y order by x.Month_m))*100),2) as Increase_year,
from (
Select
sum(payment_value) as Sum_payment,
extract(year from shipping_limit_date) as Year_y,
extract(month from shipping_limit_date) as Month_m,
from `TargetBC.payments` as P
join `TargetBC.order_items` O
on p.order_id=O.order_id
where extract(year from shipping_limit_date) between 2017 and 2018 and
extract(month from shipping_limit_date)between 1 and 8
group by Year_y,Month_m
order by Year_y,Month_m) x
```

Row	Sum_payment	prev_month	Year_y	Month_m	Increase_year
1	120802.9899999...	null	2017	1	null
2	345309.5099999...	120802.9899999...	2017	2	185.85
3	492557.5999999...	345309.5099999...	2017	3	42.64
4	428403.78	492557.5999999...	2017	4	-13.02
5	726588.8400000...	428403.78	2017	5	69.6
6	656864.9900000...	726588.8400000...	2017	6	-9.6
7	674329.8600000...	656864.9900000...	2017	7	2.66
8	869492.4400000...	674329.8600000...	2017	8	28.94
9	1233460.790000...	null	2018	1	null
10	1233541.020000...	1233460.790000...	2018	2	0.01
11	1583491.819999...	1233541.020000...	2018	3	28.37
12	1433591.160000...	1583491.819999...	2018	4	-9.47
13	1642345.670000...	1433591.160000...	2018	5	14.56

Insight/Recommendation : Growth in Sales are not consistent (Jan to Aug) for period of 2017 and 2018

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

```
select C.customer_state,p.payment_value,
Sum(P.payment_value) over (partition by C.customer_state) as Total_order_price,
round(avg(P.payment_value) over (partition by C.customer_state),2) as Avg_value,
from `TargetBC.payments` as P
join `TargetBC.orders` as O
on P.order_id=O.order_id
join `TargetBC.customers` as C
on O.customer_id=C.customer_id
group by 1,2
order by 1,2;
```



```sql
select C.customer_state,p.payment_value,
Sum(P.payment_value) over (partition by C.customer_state) as Total_order_price,
round(avg(P.payment_value) over (partition by C.customer_state),2) as Avg_value,
from `TargetBC.payments` as P
join `TargetBC.orders` as O
on P.order_id=O.order_id
join `TargetBC.customers` as C
on O.customer_id=C.customer_id
group by 1,2
order by 1,2;
```

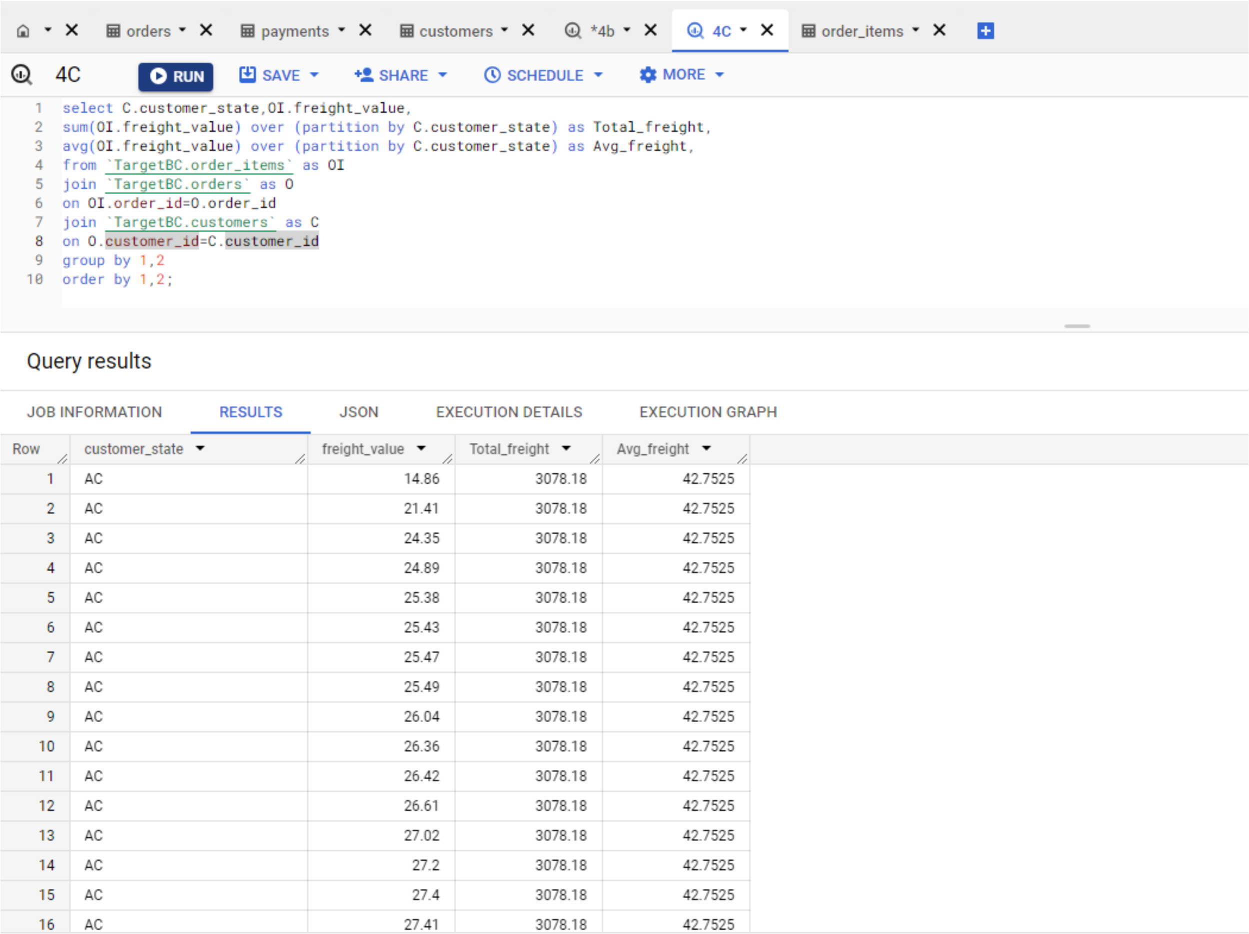
Query results

Row	customer_state	payment_value	Total_order_price	Avg_value
1	AC	11.58	19533.03	235.34
2	AC	39.03	19533.03	235.34
3	AC	43.34	19533.03	235.34
4	AC	44.01	19533.03	235.34
5	AC	45.25	19533.03	235.34
6	AC	45.37	19533.03	235.34
7	AC	47.47	19533.03	235.34
8	AC	50.14	19533.03	235.34
9	AC	51.37	19533.03	235.34
10	AC	55.28	19533.03	235.34
11	AC	56.41	19533.03	235.34
12	AC	58.92	19533.03	235.34
13	AC	62.92	19533.03	235.34
14	AC	69.34	19533.03	235.34
15	AC	75.28	19533.03	235.34
16	AC	77.2	19533.03	235.34

Insight/Recommendation : Average ticket size across all state is more than 200 but need to work to increase Average ticket size.

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

```sql
select C.customer_state,OI.freight_value,
sum(OI.freight_value) over (partition by C.customer_state) as Total_freight,
avg(OI.freight_value) over (partition by C.customer_state) as Avg_freight,
from `TargetBC.order_items` as OI
join `TargetBC.orders` as O
on OI.order_id=O.order_id
join `TargetBC.customers` as C
on O.customer_id=C.customer_id
group by 1,2
order by 1,2;
```



```
select C.customer_state,OI.freight_value,
sum(OI.freight_value) over (partition by C.customer_state) as Total_freight,
avg(OI.freight_value) over (partition by C.customer_state) as Avg_freight,
from `TargetBC.order_items` as OI
join `TargetBC.orders` as O
on OI.order_id=O.order_id
join `TargetBC.customers` as C
on O.customer_id=C.customer_id
group by 1,2
order by 1,2;
```

Query results

Row	customer_state	freight_value	Total_freight	Avg_freight
1	AC	14.86	3078.18	42.7525
2	AC	21.41	3078.18	42.7525
3	AC	24.35	3078.18	42.7525
4	AC	24.89	3078.18	42.7525
5	AC	25.38	3078.18	42.7525
6	AC	25.43	3078.18	42.7525
7	AC	25.47	3078.18	42.7525
8	AC	25.49	3078.18	42.7525
9	AC	26.04	3078.18	42.7525
10	AC	26.36	3078.18	42.7525
11	AC	26.42	3078.18	42.7525
12	AC	26.61	3078.18	42.7525
13	AC	27.02	3078.18	42.7525
14	AC	27.2	3078.18	42.7525
15	AC	27.4	3078.18	42.7525
16	AC	27.41	3078.18	42.7525

Insight/Recommendation : Total and Average freight in North state is quite high~ 40 compared to south region state ~29, Need to setup sub-warehouse to lower down freight costing.

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

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

Also, calculate the difference (in days) between the estimated & actual delivery date of an order.

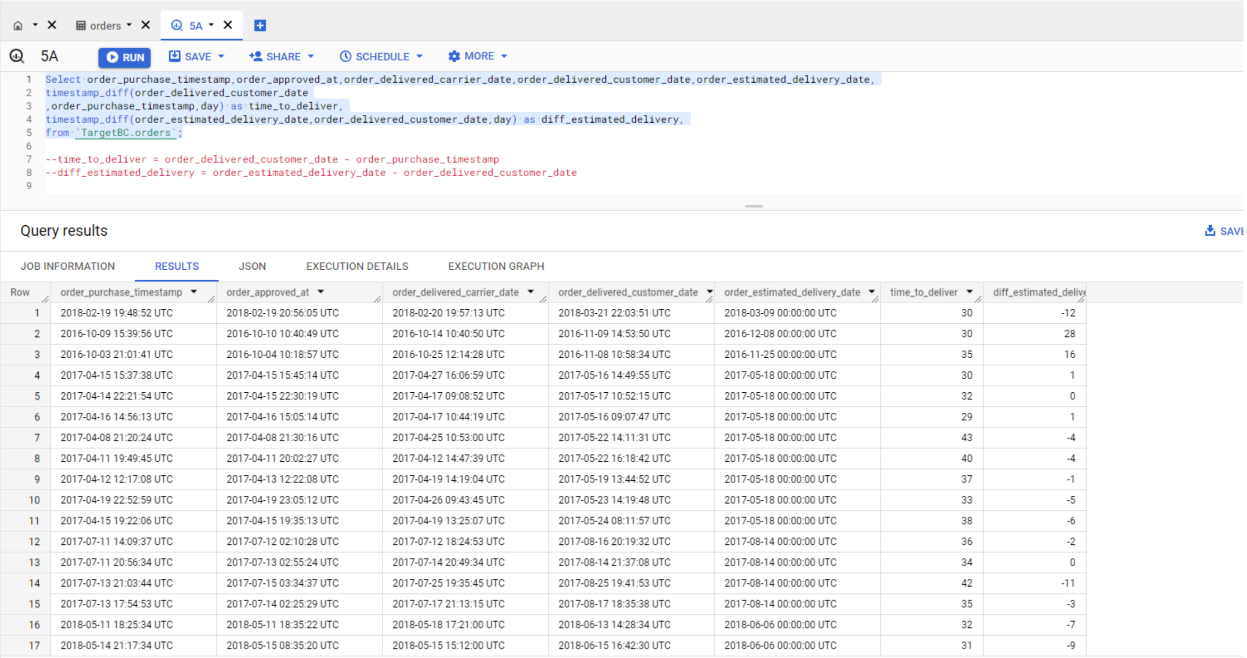
Do this in a single query.

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

```
Select
order_purchase_timestamp,order_approved_at,order_delivered_carrier_date,order_delivere
d_customer_date,order_estimated_delivery_date,
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 `TargetBC.orders`;
```



```
Select order_purchase_timestamp,order_approved_at,order_delivered_carrier_date,order_delivered_customer_date,order_estimated_delivery_date,
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 `TargetBC.orders`;

--time_to_deliver = order_delivered_customer_date - order_purchase_timestamp
--diff_estimated_delivery = order_estimated_delivery_date - order_delivered_customer_date
```

Query results

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

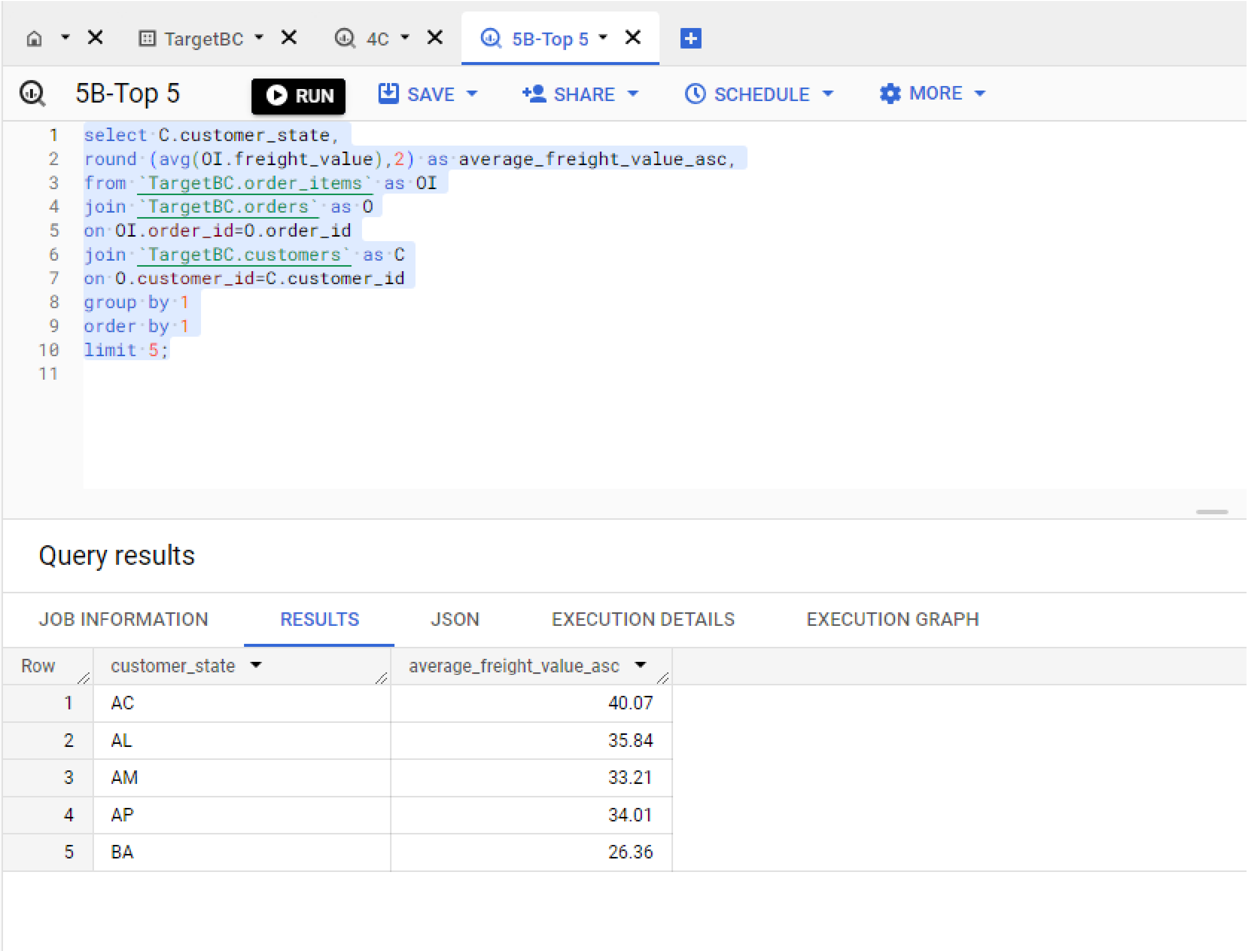
Row	order_purchase_timestamp	order_approved_at	order_delivered_carrier_date	order_delivered_customer_date	order_estimated_delivery_date	time_to_deliver	diff_estimated_delive
1	2018-02-19 19:48:52 UTC	2018-02-19 20:56:05 UTC	2018-02-20 19:57:13 UTC	2018-03-21 22:03:51 UTC	2018-03-09 00:00:00 UTC	30	-12
2	2016-10-09 15:39:56 UTC	2016-10-10 10:40:49 UTC	2016-10-14 10:40:50 UTC	2016-11-09 14:53:50 UTC	2016-12-08 00:00:00 UTC	30	28
3	2016-10-03 21:01:41 UTC	2016-10-04 10:18:57 UTC	2016-10-25 12:14:28 UTC	2016-11-08 10:58:34 UTC	2016-11-25 00:00:00 UTC	35	16
4	2017-04-15 15:37:38 UTC	2017-04-15 15:45:14 UTC	2017-04-27 16:06:59 UTC	2017-05-16 14:49:55 UTC	2017-05-18 00:00:00 UTC	30	1
5	2017-04-14 22:21:54 UTC	2017-04-15 22:30:19 UTC	2017-04-17 09:08:52 UTC	2017-05-17 10:52:15 UTC	2017-05-18 00:00:00 UTC	32	0
6	2017-04-16 14:56:13 UTC	2017-04-16 15:05:14 UTC	2017-04-17 10:44:19 UTC	2017-05-16 09:07:47 UTC	2017-05-18 00:00:00 UTC	29	1
7	2017-04-08 21:20:24 UTC	2017-04-08 21:30:16 UTC	2017-04-25 10:53:00 UTC	2017-05-22 14:11:31 UTC	2017-05-18 00:00:00 UTC	43	-4
8	2017-04-11 19:49:45 UTC	2017-04-11 20:02:27 UTC	2017-04-12 14:47:39 UTC	2017-05-22 16:18:42 UTC	2017-05-18 00:00:00 UTC	40	-4
9	2017-04-12 12:17:08 UTC	2017-04-13 12:22:08 UTC	2017-04-19 14:19:04 UTC	2017-05-19 13:44:52 UTC	2017-05-18 00:00:00 UTC	37	-1
10	2017-04-19 22:52:59 UTC	2017-04-19 23:05:12 UTC	2017-04-26 09:43:45 UTC	2017-05-23 14:19:48 UTC	2017-05-18 00:00:00 UTC	33	-5
11	2017-04-15 19:22:06 UTC	2017-04-15 19:35:13 UTC	2017-04-19 13:25:07 UTC	2017-05-24 08:11:57 UTC	2017-05-18 00:00:00 UTC	38	-6
12	2017-07-11 14:09:37 UTC	2017-07-12 02:10:28 UTC	2017-07-12 18:24:53 UTC	2017-08-16 20:19:32 UTC	2017-08-14 00:00:00 UTC	36	-2
13	2017-07-11 20:56:34 UTC	2017-07-13 02:55:24 UTC	2017-07-14 20:49:34 UTC	2017-08-14 21:37:08 UTC	2017-08-14 00:00:00 UTC	34	0
14	2017-07-13 21:03:44 UTC	2017-07-15 03:34:37 UTC	2017-07-25 19:35:45 UTC	2017-08-25 19:41:53 UTC	2017-08-14 00:00:00 UTC	42	-11
15	2017-07-13 17:54:53 UTC	2017-07-14 02:25:29 UTC	2017-07-17 21:13:15 UTC	2017-08-17 18:35:38 UTC	2017-08-14 00:00:00 UTC	35	-3
16	2018-05-11 18:25:34 UTC	2018-05-11 18:35:22 UTC	2018-05-18 17:21:00 UTC	2018-06-13 14:28:34 UTC	2018-06-06 00:00:00 UTC	32	-7
17	2018-05-14 21:17:34 UTC	2018-05-15 08:35:20 UTC	2018-05-15 15:12:00 UTC	2018-06-15 16:42:30 UTC	2018-06-06 00:00:00 UTC	31	-9

Insight/Recommendation : Time taken to deliver order is almost near to 30 day and every 2nd order is missing estimated delivery date.

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

Top 5 Highest states :

```
select C.customer_state,
round (avg(OI.freight_value),2) as average_freight_value_asc,
from `TargetBC.order_items` as OI
join `TargetBC.orders` as O
on OI.order_id=O.order_id
join `TargetBC.customers` as C
on O.customer_id=C.customer_id
group by 1
order by 1
limit 5;
```



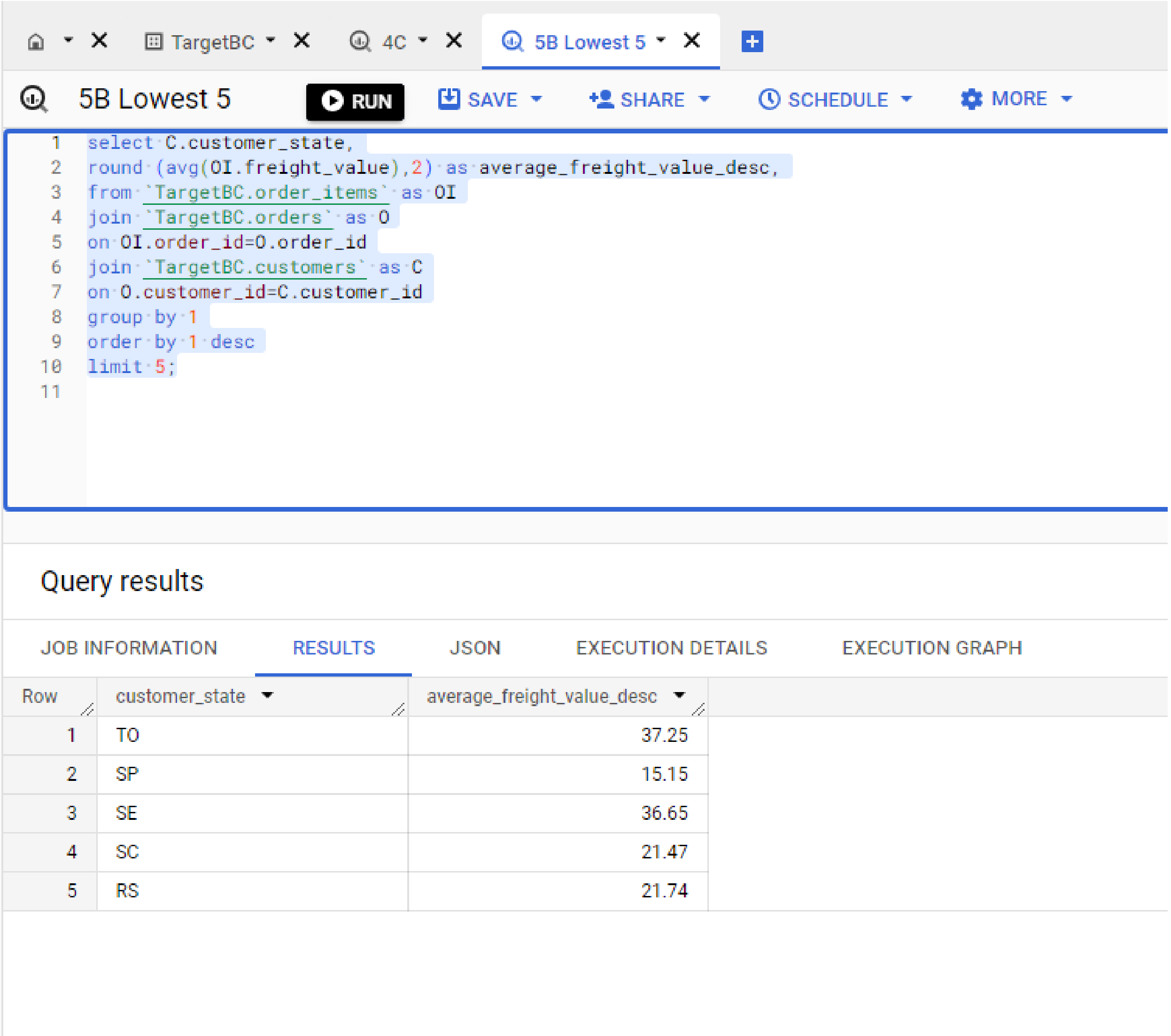
```sql
select C.customer_state,
round (avg(OI.freight_value),2) as average_freight_value_asc,
from `TargetBC.order_items` as OI
join `TargetBC.orders` as O
on OI.order_id=O.order_id
join `TargetBC.customers` as C
on O.customer_id=C.customer_id
group by 1
order by 1
limit 5;
```

Row	customer_state	average_freight_value_asc
1	AC	40.07
2	AL	35.84
3	AM	33.21
4	AP	34.01
5	BA	26.36

Insight/Recommendation : All these 5 state are for North region of brazil where average freight is higher compared to south,south-east region.

Top 5 Lowest states :

```sql
select C.customer_state,
round (avg(OI.freight_value),2) as average_freight_value_desc,
from `TargetBC.order_items` as OI
join `TargetBC.orders` as O
on OI.order_id=O.order_id
join `TargetBC.customers` as C
on O.customer_id=C.customer_id
group by 1
order by 1 desc
limit 5;
```



```
select C.customer_state,
round (avg(OI.freight_value),2) as average_freight_value_desc,
from `TargetBC.order_items` as OI
join `TargetBC.orders` as O
on OI.order_id=O.order_id
join `TargetBC.customers` as C
on O.customer_id=C.customer_id
group by 1
order by 1 desc
limit 5;
```

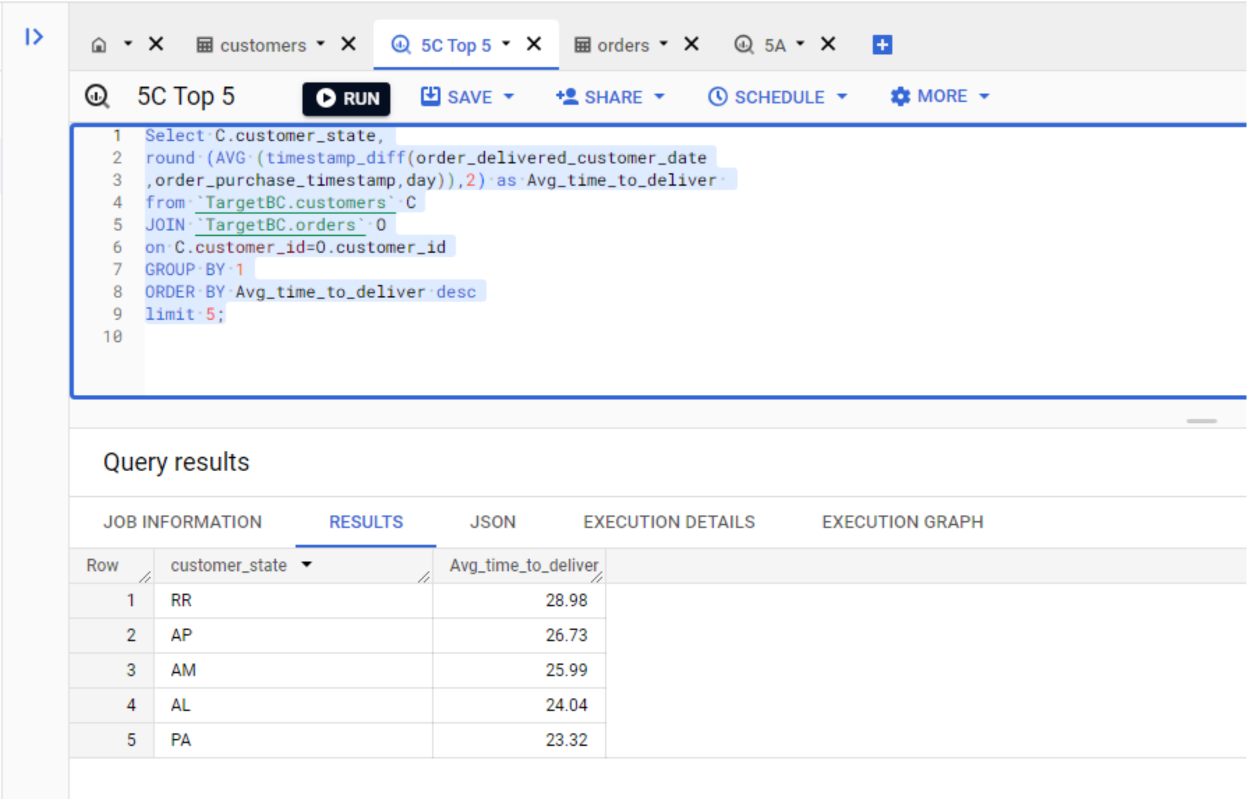
Row	customer_state	average_freight_value_desc
1	TO	37.25
2	SP	15.15
3	SE	36.65
4	SC	21.47
5	RS	21.74

Insight/Recommendation : State from South,South-east region has lowest Avg_freight_value

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

Top 5 state Highest average delivery time :

```
Select C.customer_state,
round (AVG (timestamp_diff(order_delivered_customer_date
,order_purchase_timestamp,day)),2) as Avg_time_to_deliver
from `TargetBC.customers` C
JOIN `TargetBC.orders` O
on C.customer_id=O.customer_id
GROUP BY 1
ORDER BY Avg_time_to_deliver desc
limit 5;
```



```
Select C.customer_state,
round (AVG (timestamp_diff(order_delivered_customer_date
,order_purchase_timestamp,day)),2) as Avg_time_to_deliver
from `TargetBC.customers` C
JOIN `TargetBC.orders` O
on C.customer_id=O.customer_id
GROUP BY 1
ORDER BY Avg_time_to_deliver desc
limit 5;
```

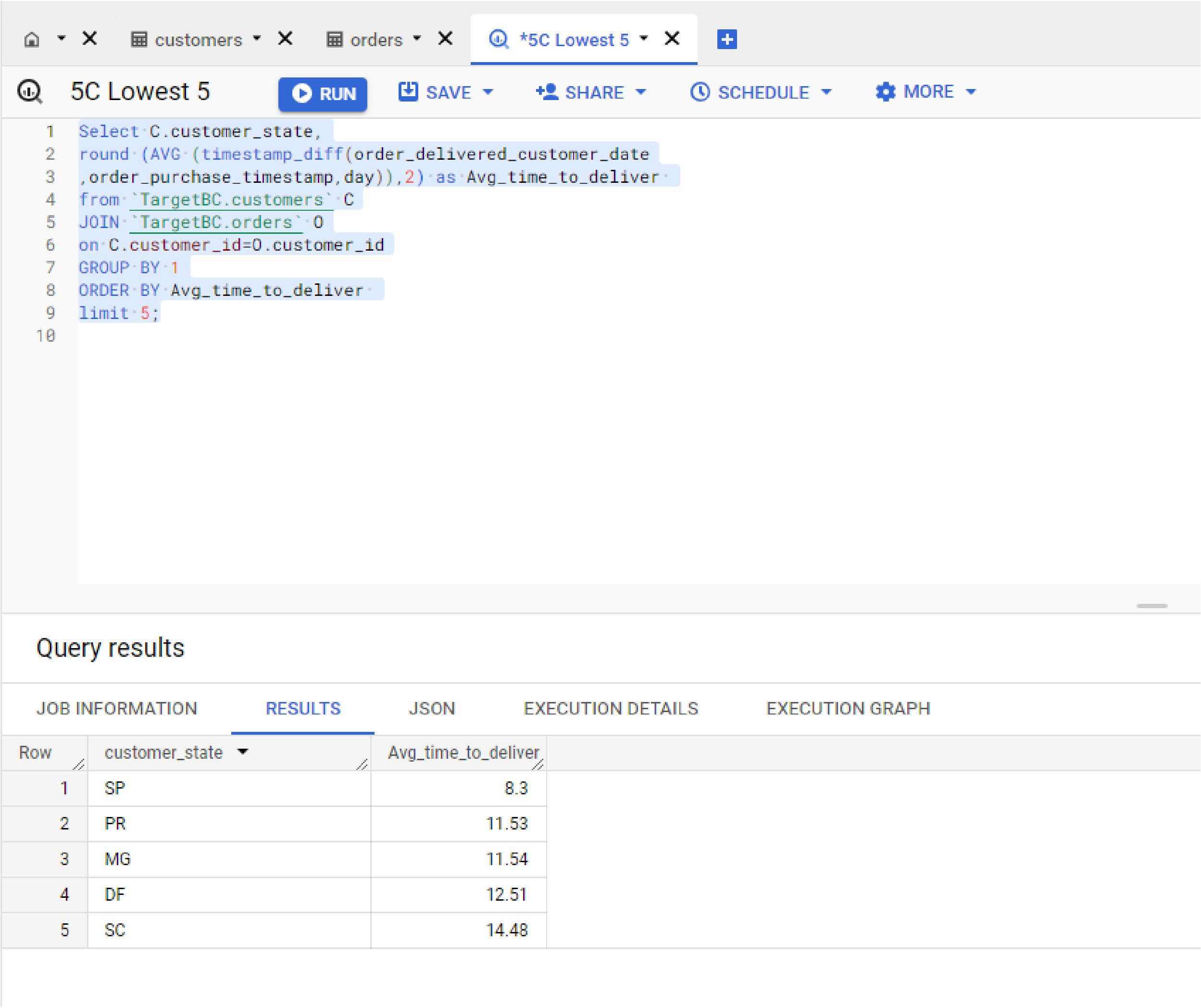
Query results

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

Insight/Recommendation : Delivery time for North state is quite high compared to south,southest state.

Top 5 state lowest average delivery time:

```
Select C.customer_state,
round (AVG (timestamp_diff(order_delivered_customer_date
,order_purchase_timestamp,day)),2) as Avg_time_to_deliver
from `TargetBC.customers` C
JOIN `TargetBC.orders` O
on C.customer_id=O.customer_id
GROUP BY 1
ORDER BY Avg_time_to_deliver
limit 5;
```



```
Select C.customer_state,
round (AVG (timestamp_diff(order_delivered_customer_date
,order_purchase_timestamp,day)),2) as Avg_time_to_deliver
from `TargetBC.customers` C
JOIN `TargetBC.orders` O
on C.customer_id=O.customer_id
GROUP BY 1
ORDER BY Avg_time_to_deliver
limit 5;
```

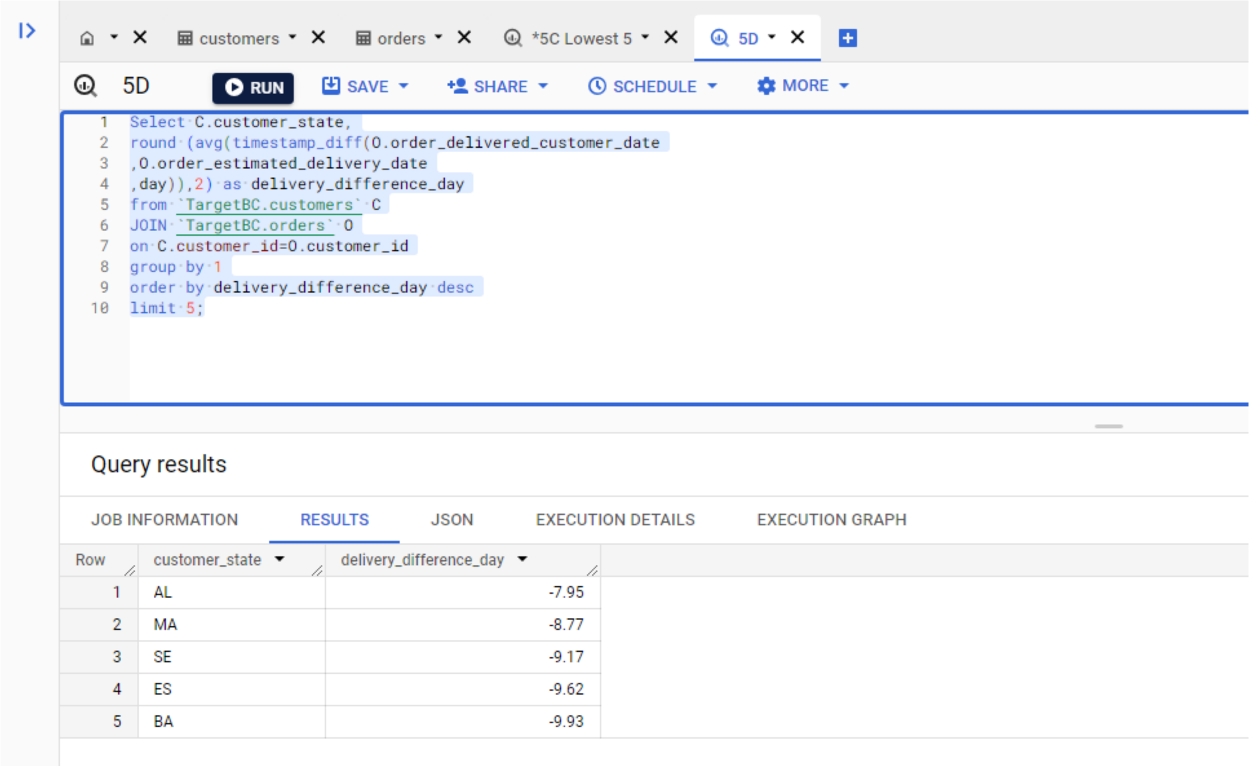
Row	customer_state	Avg_time_to_deliver
1	SP	8.3
2	PR	11.53
3	MG	11.54
4	DF	12.51
5	SC	14.48

Insight/Recommendation : Delivery time for to south,southest state are good compared to north state.

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

You can use the difference between the averages of actual & estimated delivery date to figure out how

```
Select C.customer_state,
round (avg(timestamp_diff(O.order_delivered_customer_date
,O.order_estimated_delivery_date
,day)),2) as delivery_difference_day
from `TargetBC.customers` C
JOIN `TargetBC.orders` O
on C.customer_id=O.customer_id
group by 1
order by delivery_difference_day desc
limit 5;
```



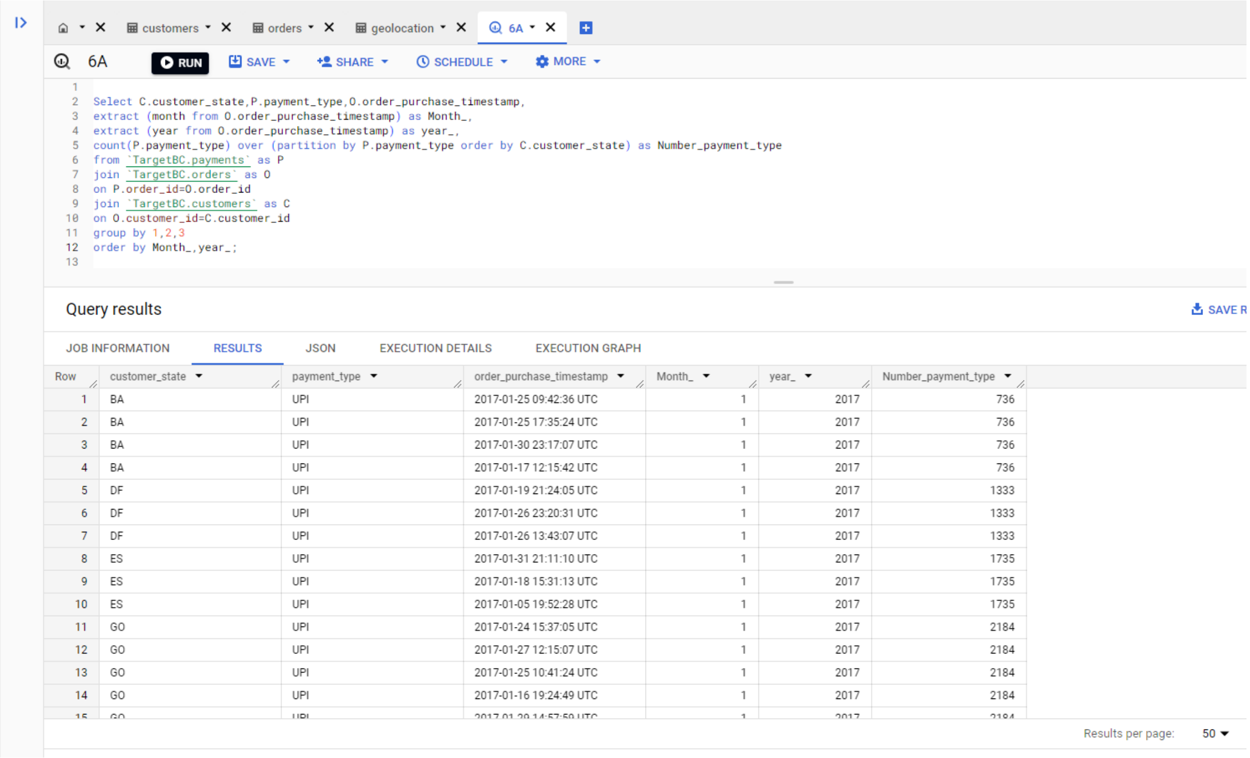
```
Select C.customer_state,
round (avg(timestamp_diff(O.order_delivered_customer_date
,O.order_estimated_delivery_date
,day)),2) as delivery_difference_day
from `TargetBC.customers` C
JOIN `TargetBC.orders` O
on C.customer_id=O.customer_id
group by 1
order by delivery_difference_day desc
limit 5;
```

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

6. **Analysis based on the payments:**

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

```
Select C.customer_state,P.payment_type,O.order_purchase_timestamp,
extract (month from O.order_purchase_timestamp) as Month_,
extract (year from O.order_purchase_timestamp) as year_,
count(P.payment_type) over (partition by P.payment_type order by C.customer_state) as
Number_payment_type
from `TargetBC.payments` as P
join `TargetBC.orders` as O
on P.order_id=O.order_id
join `TargetBC.customers` as C
on O.customer_id=C.customer_id
group by 1,2,3
order by Month_,year_;
```



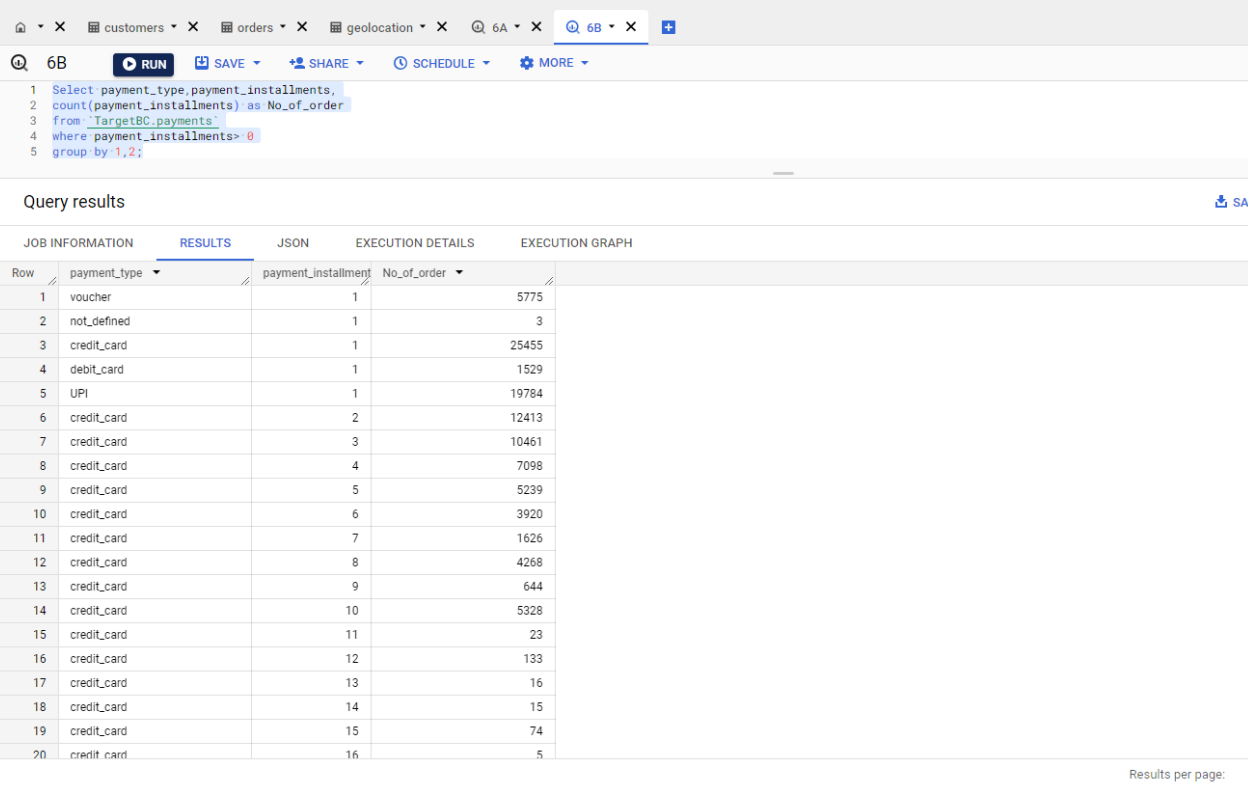
```sql
Select C.customer_state,P.payment_type,O.order_purchase_timestamp,
extract (month from O.order_purchase_timestamp) as Month_,
extract (year from O.order_purchase_timestamp) as year_,
count(P.payment_type) over (partition by P.payment_type order by C.customer_state) as Number_payment_type
from `TargetBC.payments` as P
join `TargetBC.orders` as O
on P.order_id=O.order_id
join `TargetBC.customers` as C
on O.customer_id=C.customer_id
group by 1,2,3
order by Month_,year_;
```

Row	customer_state	payment_type	order_purchase_timestamp	Month_	year_	Number_payment_type
1	BA	UPI	2017-01-25 09:42:36 UTC	1	2017	736
2	BA	UPI	2017-01-25 17:35:24 UTC	1	2017	736
3	BA	UPI	2017-01-30 23:17:07 UTC	1	2017	736
4	BA	UPI	2017-01-17 12:15:42 UTC	1	2017	736
5	DF	UPI	2017-01-19 21:24:05 UTC	1	2017	1333
6	DF	UPI	2017-01-26 23:20:31 UTC	1	2017	1333
7	DF	UPI	2017-01-26 13:43:07 UTC	1	2017	1333
8	ES	UPI	2017-01-31 21:11:10 UTC	1	2017	1735
9	ES	UPI	2017-01-18 15:31:13 UTC	1	2017	1735
10	ES	UPI	2017-01-05 19:52:28 UTC	1	2017	1735
11	GO	UPI	2017-01-24 15:37:05 UTC	1	2017	2184
12	GO	UPI	2017-01-27 12:15:07 UTC	1	2017	2184
13	GO	UPI	2017-01-25 10:41:24 UTC	1	2017	2184
14	GO	UPI	2017-01-16 19:24:49 UTC	1	2017	2184
15	GO	UPI	2017-01-29 14:57:59 UTC	1	2017	2184

Insight/Recommendation : Customer mostly user credit card payment mode compared to debit,UPI this trend is shown across all state.

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

```sql
Select payment_type,payment_installments,
count(payment_installments) as No_of_order
from `TargetBC.payments`
where payment_installments> 0
group by 1,2;
```



```
Select payment_type,payment_installments,
count(payment_installments) as No_of_order
from `TargetBC.payments`
where payment_installments> 0
group by 1,2;
```

Query results

Row	payment_type	payment_installment	No_of_order
1	voucher	1	5775
2	not_defined	1	3
3	credit_card	1	25455
4	debit_card	1	1529
5	UPI	1	19784
6	credit_card	2	12413
7	credit_card	3	10461
8	credit_card	4	7098
9	credit_card	5	5239
10	credit_card	6	3920
11	credit_card	7	1626
12	credit_card	8	4268
13	credit_card	9	644
14	credit_card	10	5328
15	credit_card	11	23
16	credit_card	12	133
17	credit_card	13	16
18	credit_card	14	15
19	credit_card	15	74
20	credit_card	16	5

Insight/Recommendation : Customer mostly take short period installment form credit cards.