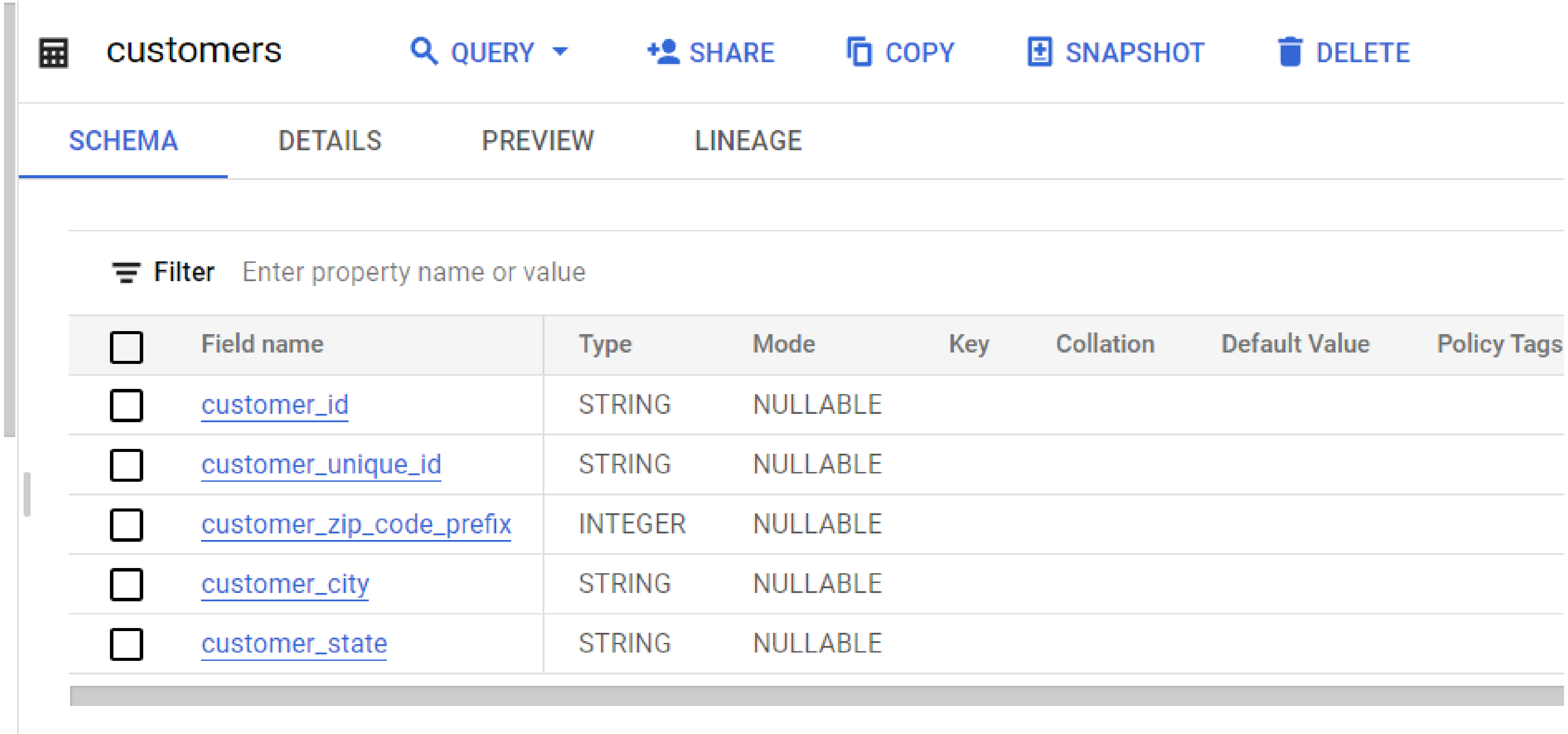
Target : SQL Project

Date: 03/07/2023

By: Pavan Kumaar

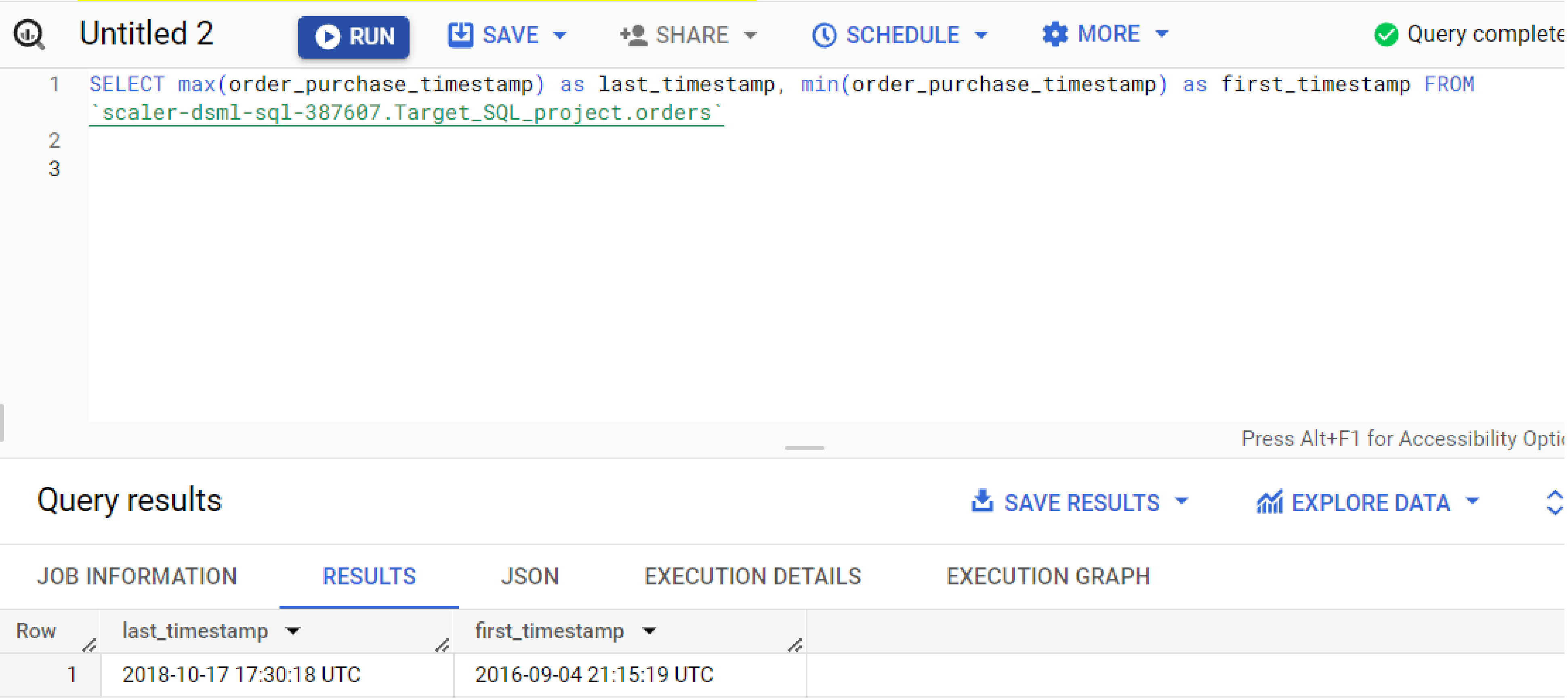
1.1 Data type of all columns in the "customers" table.



customers

□	Field name	Type	Mode	Key	Collation	Default Value	Policy Tags
□	customer_id	STRING	NULLABLE				
□	customer_unique_id	STRING	NULLABLE				
□	customer_zip_code_prefix	INTEGER	NULLABLE				
□	customer_city	STRING	NULLABLE				
□	customer_state	STRING	NULLABLE				

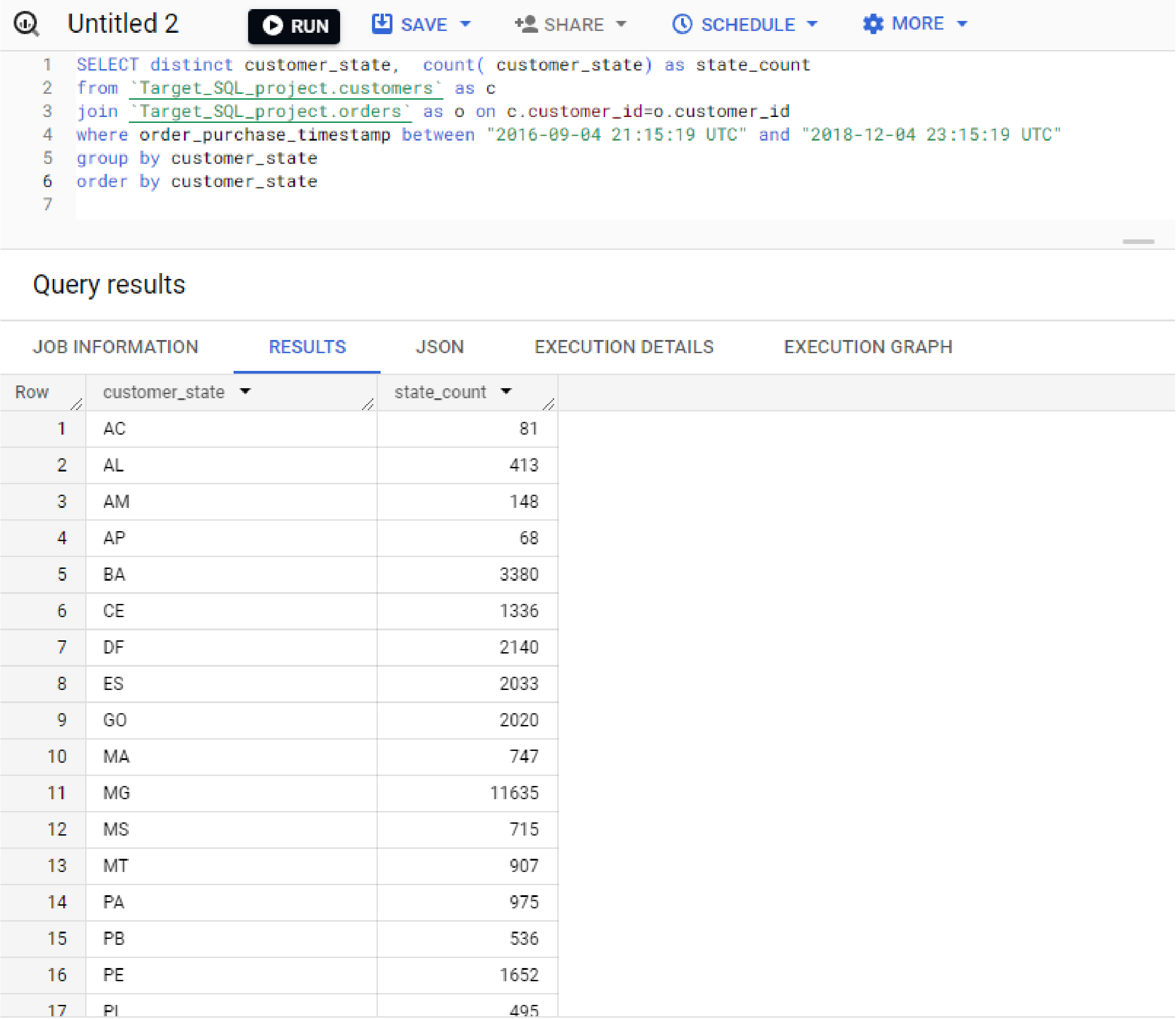
1.2 Get the time range between which the orders were placed.



```sql
SELECT max(order_purchase_timestamp) as last_timestamp, min(order_purchase_timestamp) as first_timestamp FROM
`scaler-dsml-sql-387607.Target_SQL_project.orders`
```

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

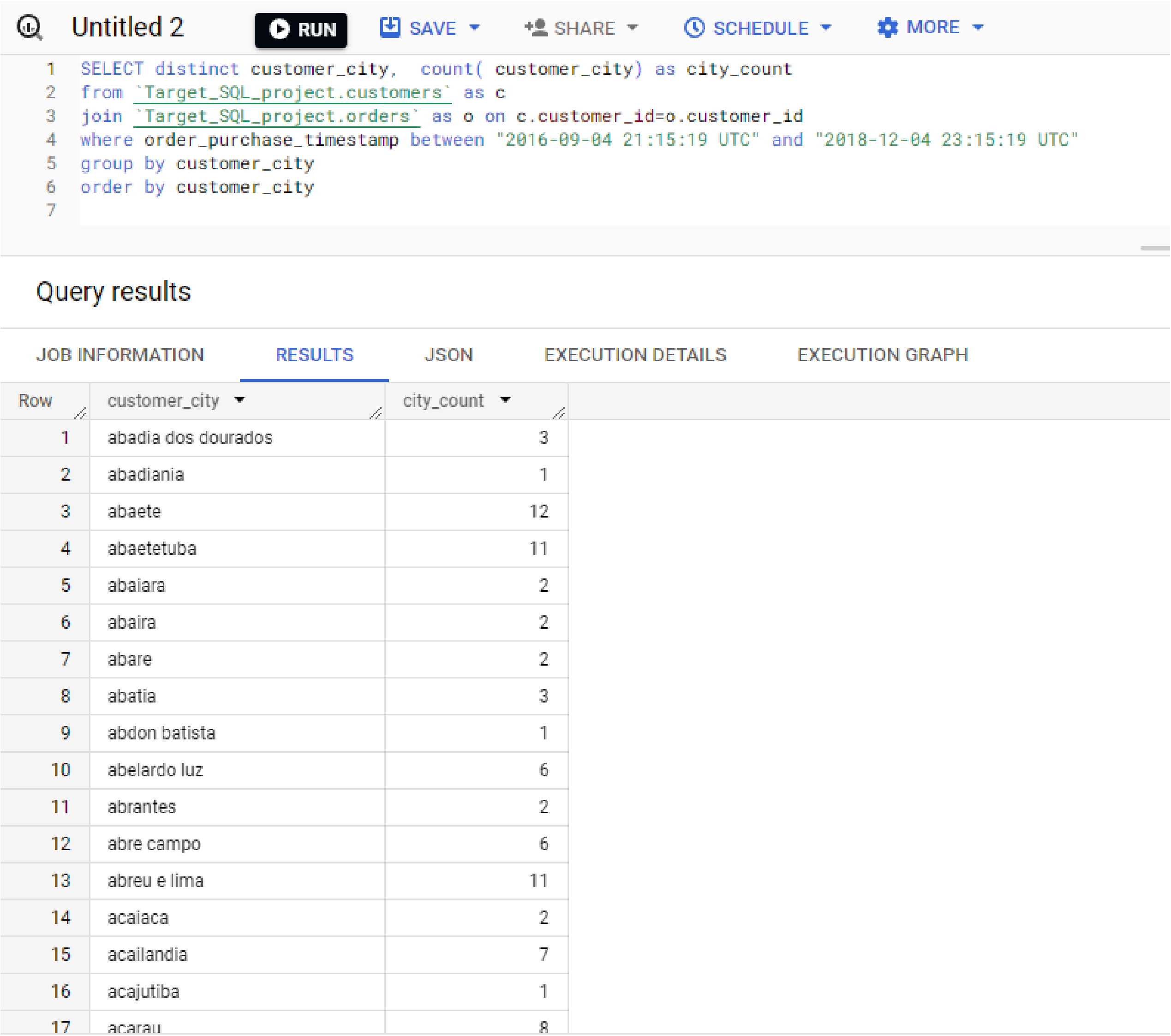
1.3 Count the Cities & States of customers who ordered during the given period.



```
SELECT distinct customer_state,  count( customer_state) as state_count
from `Target_SQL_project.customers` as c
join `Target_SQL_project.orders` as o on c.customer_id=o.customer_id
where order_purchase_timestamp between "2016-09-04 21:15:19 UTC" and "2018-12-04 23:15:19 UTC"
group by customer_state
order by customer_state
```

Query results

Row	customer_state	state_count
1	AC	81
2	AL	413
3	AM	148
4	AP	68
5	BA	3380
6	CE	1336
7	DF	2140
8	ES	2033
9	GO	2020
10	MA	747
11	MG	11635
12	MS	715
13	MT	907
14	PA	975
15	PB	536
16	PE	1652
17	PI	495

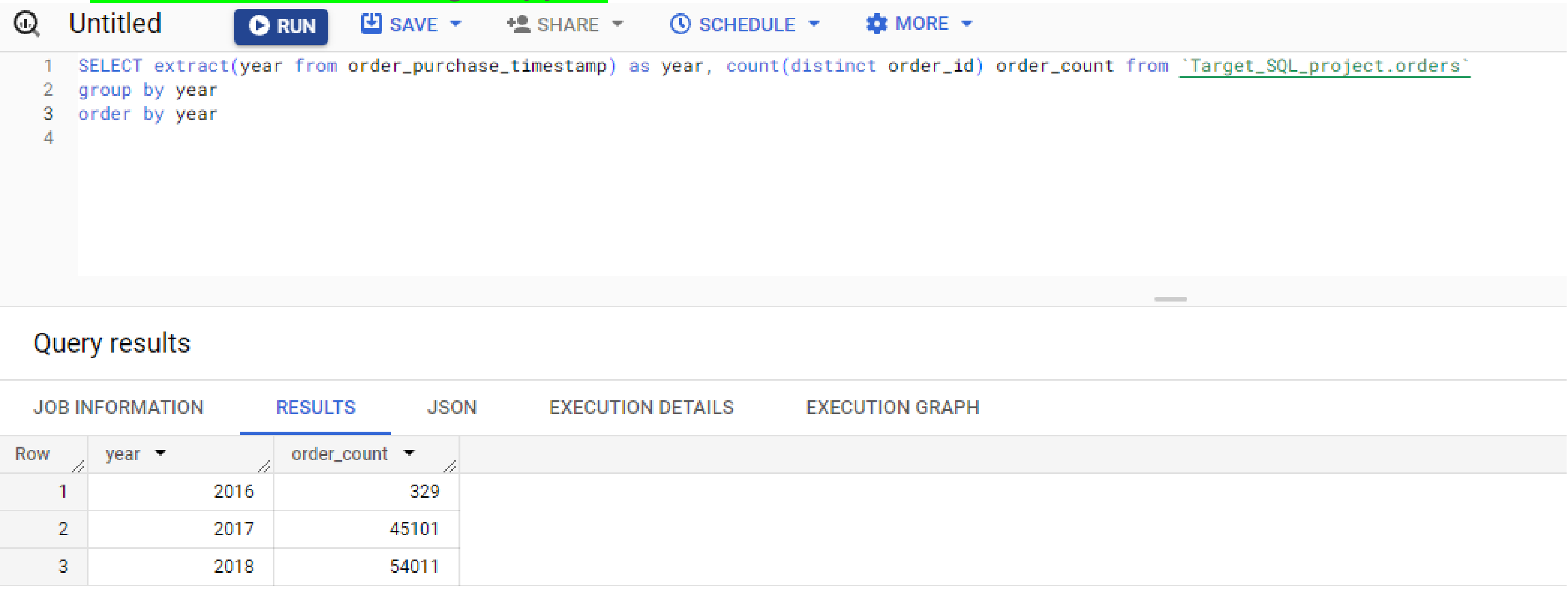


```sql
SELECT distinct customer_city,  count( customer_city) as city_count
from `Target_SQL_project.customers` as c
join `Target_SQL_project.orders` as o on c.customer_id=o.customer_id
where order_purchase_timestamp between "2016-09-04 21:15:19 UTC" and "2018-12-04 23:15:19 UTC"
group by customer_city
order by customer_city
```

Row	customer_city	city_count
1	abadia dos dourados	3
2	abadiania	1
3	abaete	12
4	abaetetuba	11
5	abaiara	2
6	abaira	2
7	abare	2
8	abatia	3
9	abdon batista	1
10	abelardo luz	6
11	abrantes	2
12	abre campo	6
13	abreu e lima	11
14	acaiaca	2
15	acailandia	7
16	acajutiba	1
17	acarau	8

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

Yes, the orders are increasing every year.

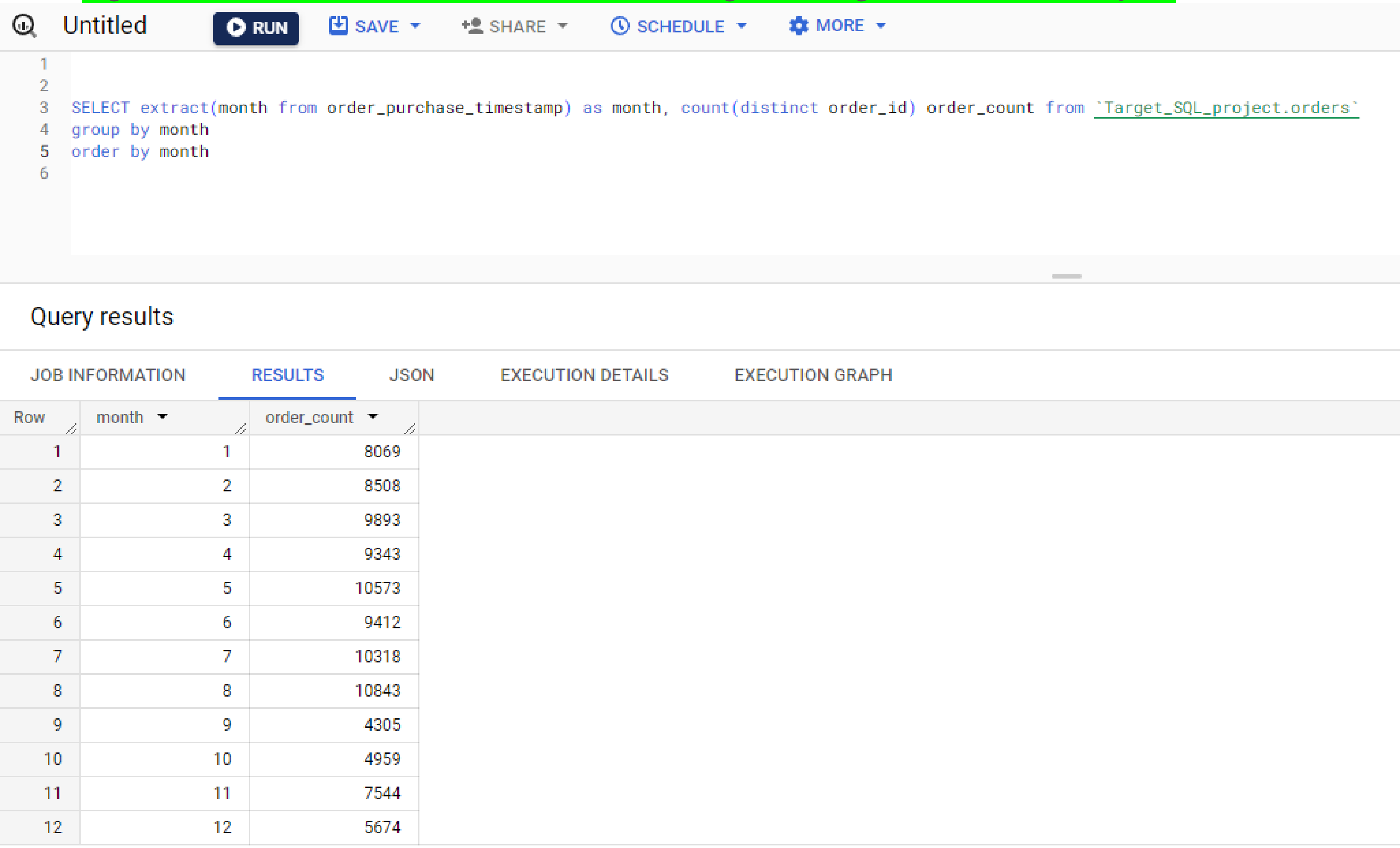


```
SELECT extract(year from order_purchase_timestamp) as year, count(distinct order_id) order_count from `Target_SQL_project.orders`
group by year
order by year
```

Row	year	order_count
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 orders being placed?

Highest sales are recorded between Q2 to Q3. Jan to Aug has the highest sale trend in the year.



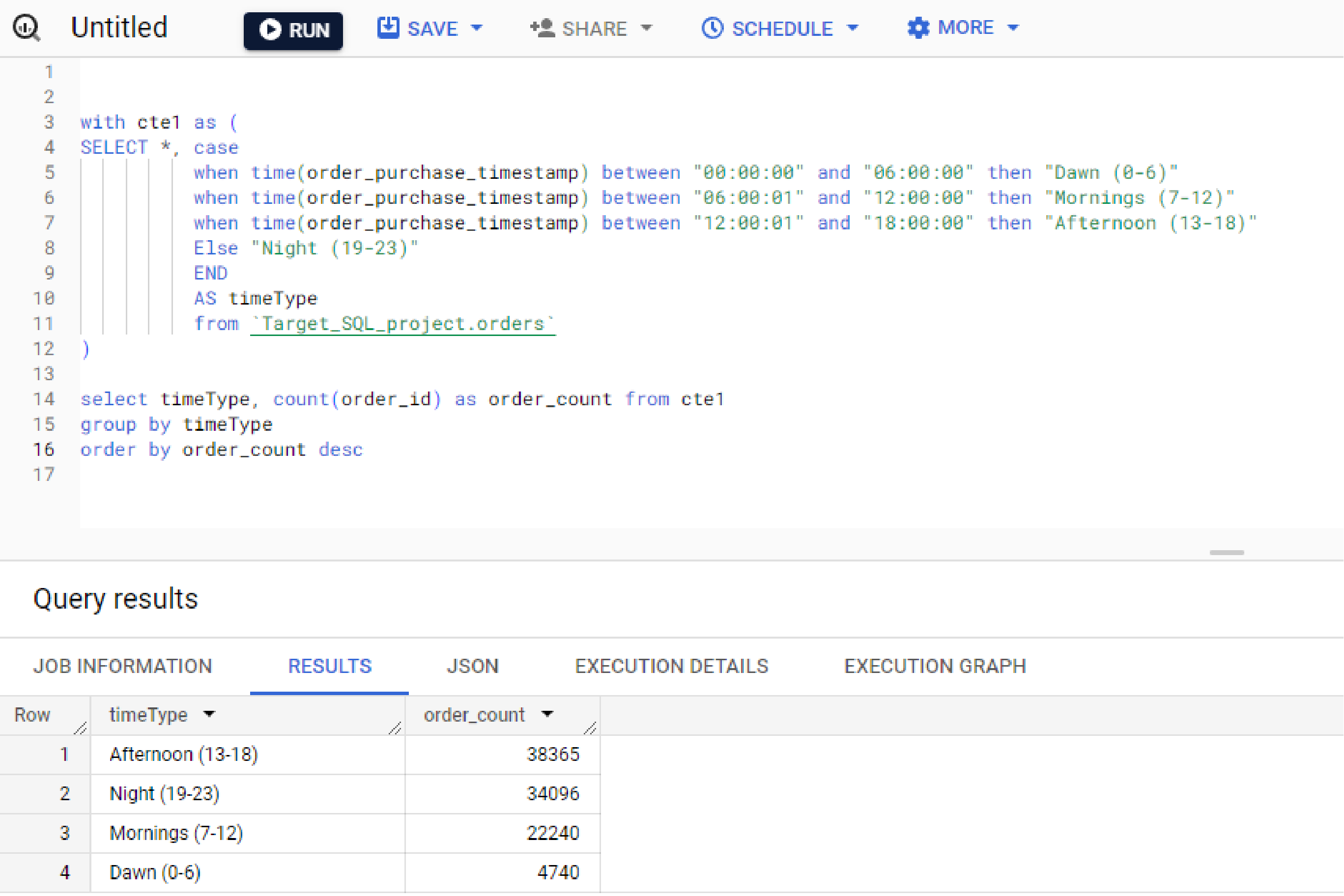
```sql


SELECT extract(month from order_purchase_timestamp) as month, count(distinct order_id) order_count from `Target_SQL_project.orders`
group by month
order by month

```

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

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

Most orders are purchased in Afternoon and Night in comparison with Dawn and Mornings.

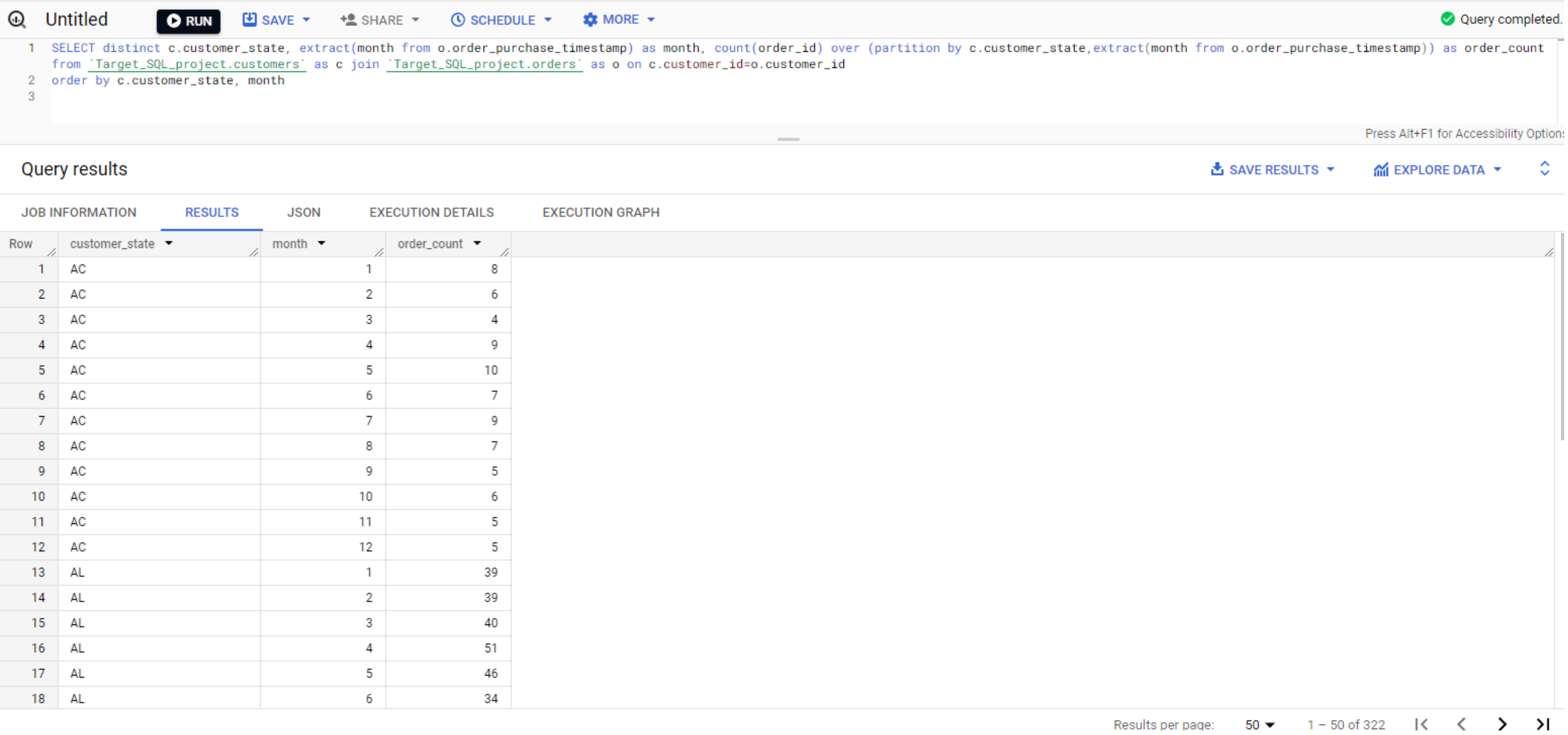


```


with cte1 as (
SELECT *, case
          when time(order_purchase_timestamp) between "00:00:00" and "06:00:00" then "Dawn (0-6)"
          when time(order_purchase_timestamp) between "06:00:01" and "12:00:00" then "Mornings (7-12)"
          when time(order_purchase_timestamp) between "12:00:01" and "18:00:00" then "Afternoon (13-18)"
          Else "Night (19-23)"
          END
          AS timeType
          from `Target_SQL_project.orders`
)

select timeType, count(order_id) as order_count from cte1
group by timeType
order by order_count desc

```

Query results

Row	timeType	order_count
1	Afternoon (13-18)	38365
2	Night (19-23)	34096
3	Mornings (7-12)	22240
4	Dawn (0-6)	4740

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

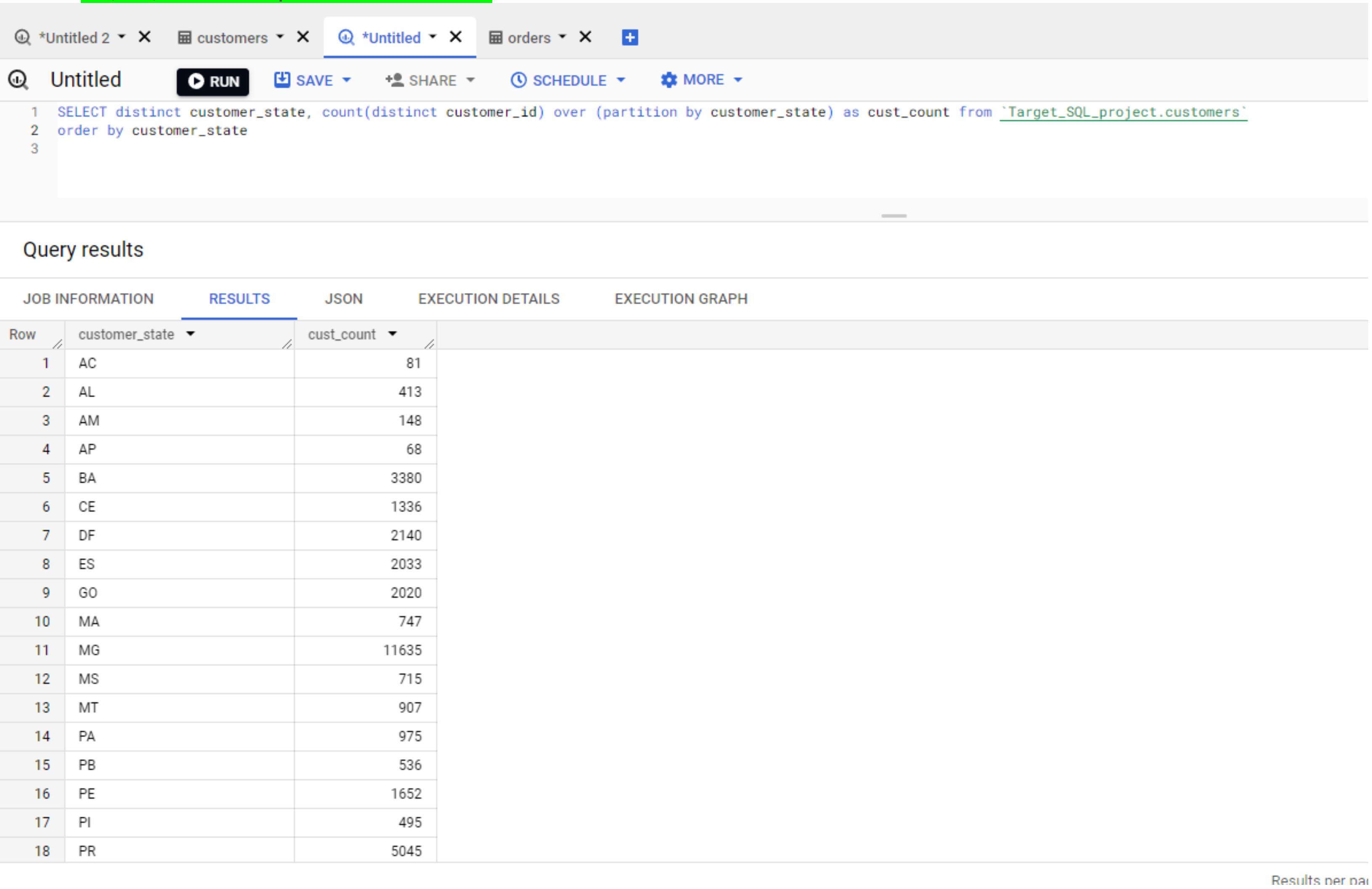


```
SELECT distinct c.customer_state, extract(month from o.order_purchase_timestamp) as month, count(order_id) over (partition by c.customer_state,extract(month from o.order_purchase_timestamp)) as order_count
from `Target_SQL_project.customers` as c join `Target_SQL_project.orders` as o on c.customer_id=o.customer_id
order by c.customer_state, month
```

Row	customer_state	month	order_count
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
16	AL	4	51
17	AL	5	46
18	AL	6	34

3.2 How are the customers distributed across all the states?

SP, RJ, MG has top 3 customer base.

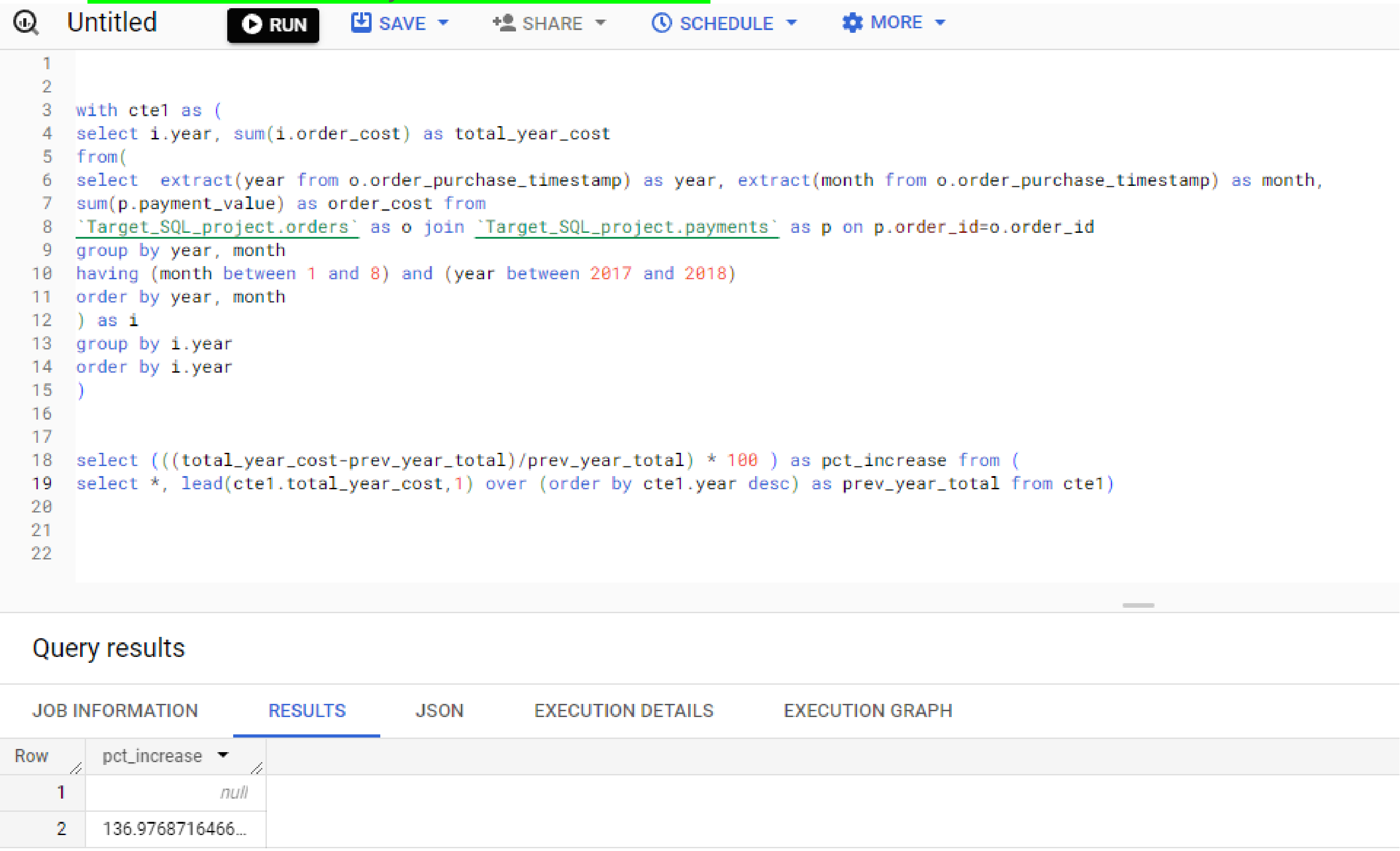


```
SELECT distinct customer_state, count(distinct customer_id) over (partition by customer_state) as cust_count from `Target_SQL_project.customers`
order by customer_state
```

Row	customer_state	cust_count
1	AC	81
2	AL	413
3	AM	148
4	AP	68
5	BA	3380
6	CE	1336
7	DF	2140
8	ES	2033
9	GO	2020
10	MA	747
11	MG	11635
12	MS	715
13	MT	907
14	PA	975
15	PB	536
16	PE	1652
17	PI	495
18	PR	5045

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.

Total orders increased by 136 % from 2017 to 2018

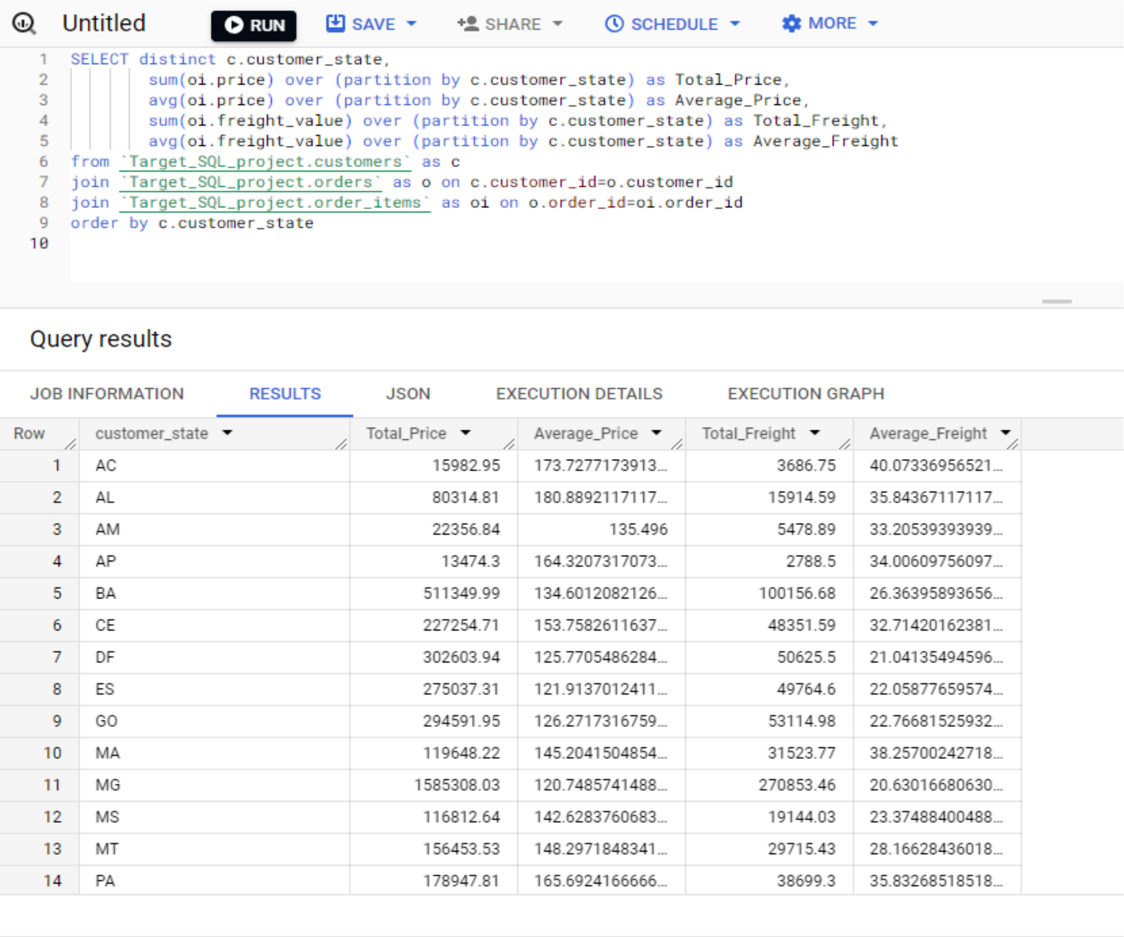


```


with cte1 as (
select i.year, sum(i.order_cost) as total_year_cost
from(
select  extract(year from o.order_purchase_timestamp) as year, extract(month from o.order_purchase_timestamp) as month,
sum(p.payment_value) as order_cost from
`Target_SQL_project.orders` as o join `Target_SQL_project.payments` as p on p.order_id=o.order_id
group by year, month
having (month between 1 and 8) and (year between 2017 and 2018)
order by year, month
) as i
group by i.year
order by i.year
)


select (((total_year_cost-prev_year_total)/prev_year_total) * 100 ) as pct_increase from (
select *, lead(cte1.total_year_cost,1) over (order by cte1.year desc) as prev_year_total from cte1)



```

Row	pct_increase
1	null
2	136.9768716466...

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

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



```sql
SELECT distinct c.customer_state,
          sum(oi.price) over (partition by c.customer_state) as Total_Price,
          avg(oi.price) over (partition by c.customer_state) as Average_Price,
          sum(oi.freight_value) over (partition by c.customer_state) as Total_Freight,
          avg(oi.freight_value) over (partition by c.customer_state) as Average_Freight
from `Target_SQL_project.customers` as c
join `Target_SQL_project.orders` as o on c.customer_id=o.customer_id
join `Target_SQL_project.order_items` as oi on o.order_id=oi.order_id
order by c.customer_state
```

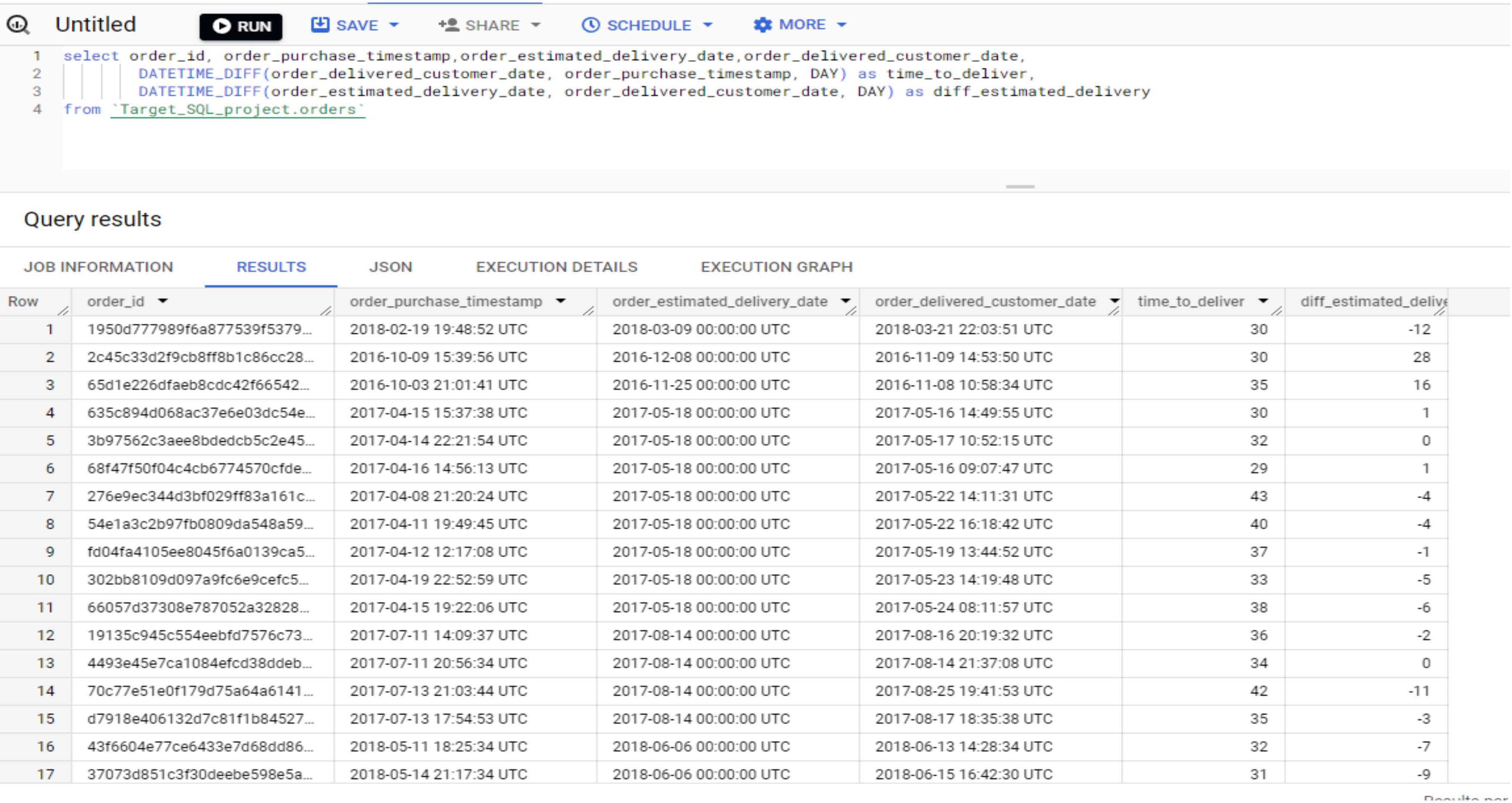
Query results

Row	customer_state	Total_Price	Average_Price	Total_Freight	Average_Freight
1	AC	15982.95	173.7277173913...	3686.75	40.07336956521...
2	AL	80314.81	180.8892117117...	15914.59	35.84367117117...
3	AM	22356.84	135.496	5478.89	33.20539393939...
4	AP	13474.3	164.3207317073...	2788.5	34.00609756097...
5	BA	511349.99	134.6012082126...	100156.68	26.36395893656...
6	CE	227254.71	153.7582611637...	48351.59	32.71420162381...
7	DF	302603.94	125.7705486284...	50625.5	21.04135494596...
8	ES	275037.31	121.9137012411...	49764.6	22.05877659574...
9	GO	294591.95	126.2717316759...	53114.98	22.76681525932...
10	MA	119648.22	145.2041504854...	31523.77	38.25700242718...
11	MG	1585308.03	120.7485741488...	270853.46	20.63016680630...
12	MS	116812.64	142.6283760683...	19144.03	23.37488400488...
13	MT	156453.53	148.2971848341...	29715.43	28.16628436018...
14	PA	178947.81	165.6924166666...	38699.3	35.83268518518...

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

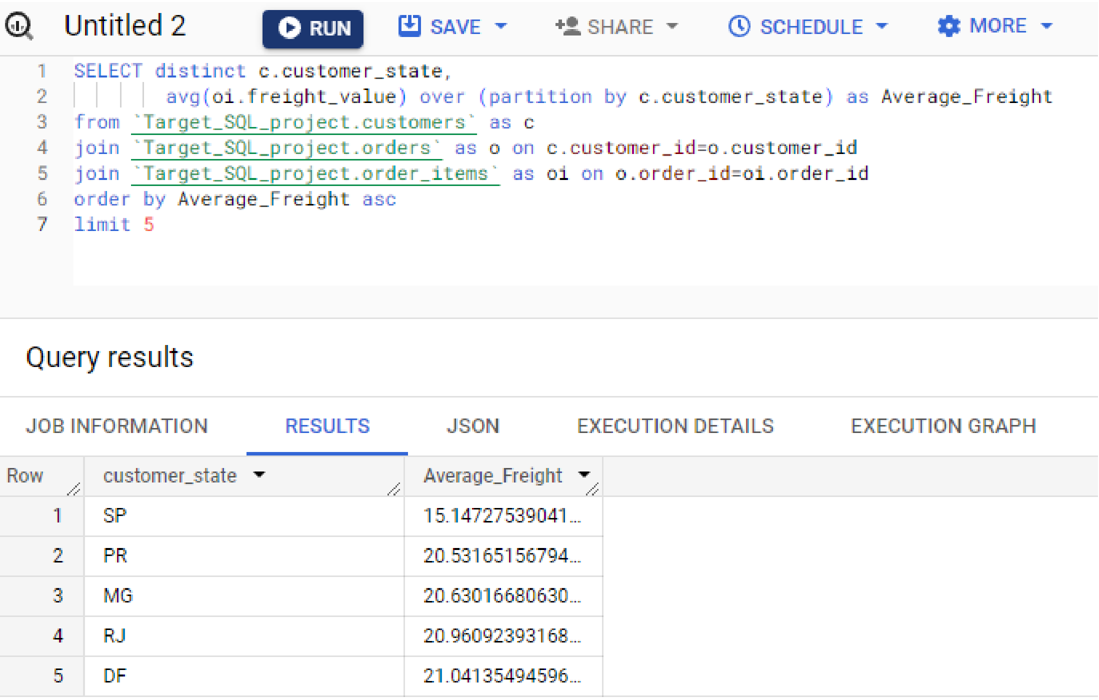


```sql
select order_id, order_purchase_timestamp,order_estimated_delivery_date,order_delivered_customer_date,
        DATETIME_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY) as time_to_deliver,
        DATETIME_DIFF(order_estimated_delivery_date, order_delivered_customer_date, DAY) as diff_estimated_delivery
from `Target_SQL_project.orders`
```

Query results

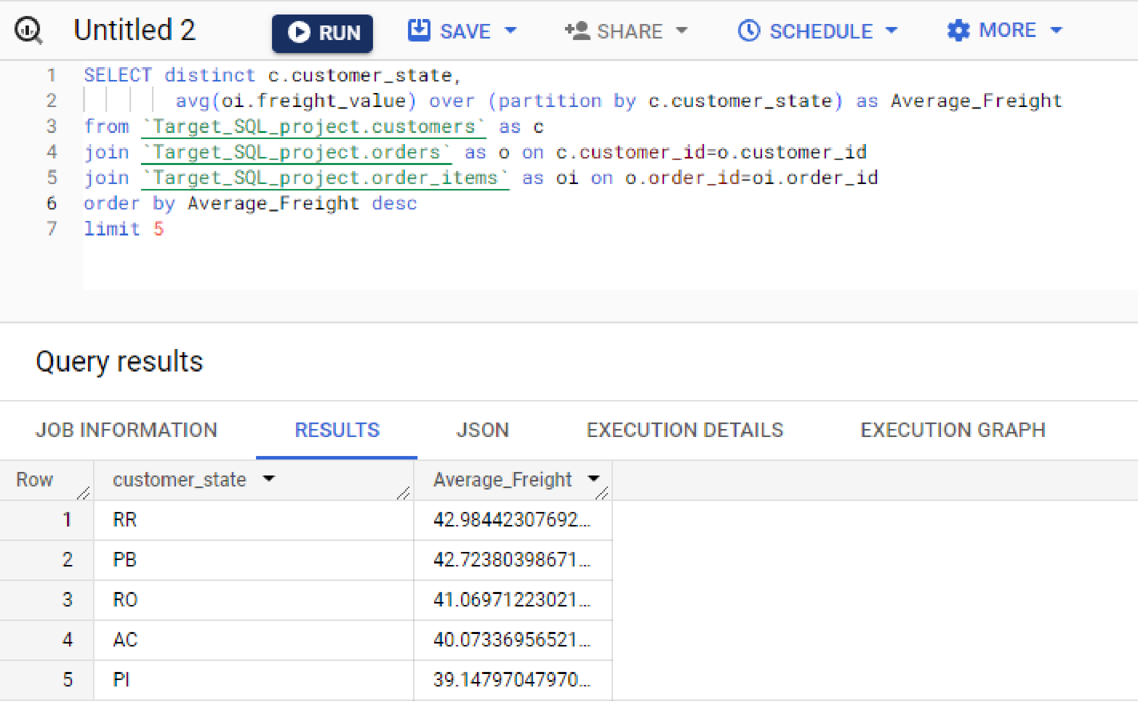
Row	order_id	order_purchase_timestamp	order_estimated_delivery_date	order_delivered_customer_date	time_to_deliver	diff_estimated_delive
1	1950d777989f6a877539f5379...	2018-02-19 19:48:52 UTC	2018-03-09 00:00:00 UTC	2018-03-21 22:03:51 UTC	30	-12
2	2c45c33d2f9cb8ff8b1c86cc28...	2016-10-09 15:39:56 UTC	2016-12-08 00:00:00 UTC	2016-11-09 14:53:50 UTC	30	28
3	65d1e226dfaeb8cdc42f66542...	2016-10-03 21:01:41 UTC	2016-11-25 00:00:00 UTC	2016-11-08 10:58:34 UTC	35	16
4	635c894d068ac37e6e03dc54e...	2017-04-15 15:37:38 UTC	2017-05-18 00:00:00 UTC	2017-05-16 14:49:55 UTC	30	1
5	3b97562c3aee8bdedcb5c2e45...	2017-04-14 22:21:54 UTC	2017-05-18 00:00:00 UTC	2017-05-17 10:52:15 UTC	32	0
6	68f47f50f04c4cb6774570cfde...	2017-04-16 14:56:13 UTC	2017-05-18 00:00:00 UTC	2017-05-16 09:07:47 UTC	29	1
7	276e9ec344d3bf029ff83a161c...	2017-04-08 21:20:24 UTC	2017-05-18 00:00:00 UTC	2017-05-22 14:11:31 UTC	43	-4
8	54e1a3c2b97fb0809da548a59...	2017-04-11 19:49:45 UTC	2017-05-18 00:00:00 UTC	2017-05-22 16:18:42 UTC	40	-4
9	fd04fa4105ee8045f6a0139ca5...	2017-04-12 12:17:08 UTC	2017-05-18 00:00:00 UTC	2017-05-19 13:44:52 UTC	37	-1
10	302bb8109d097a9fc6e9cefc5...	2017-04-19 22:52:59 UTC	2017-05-18 00:00:00 UTC	2017-05-23 14:19:48 UTC	33	-5
11	66057d37308e787052a32828...	2017-04-15 19:22:06 UTC	2017-05-18 00:00:00 UTC	2017-05-24 08:11:57 UTC	38	-6
12	19135c945c554eebfd7576c73...	2017-07-11 14:09:37 UTC	2017-08-14 00:00:00 UTC	2017-08-16 20:19:32 UTC	36	-2
13	4493e45e7ca1084efcd38ddeb...	2017-07-11 20:56:34 UTC	2017-08-14 00:00:00 UTC	2017-08-14 21:37:08 UTC	34	0
14	70c77e51e0f179d75a64a6141...	2017-07-13 21:03:44 UTC	2017-08-14 00:00:00 UTC	2017-08-25 19:41:53 UTC	42	-11
15	d7918e406132d7c81f1b84527...	2017-07-13 17:54:53 UTC	2017-08-14 00:00:00 UTC	2017-08-17 18:35:38 UTC	35	-3
16	43f6604e77ce6433e7d68dd86...	2018-05-11 18:25:34 UTC	2018-06-06 00:00:00 UTC	2018-06-13 14:28:34 UTC	32	-7
17	37073d851c3f30deebe598e5a...	2018-05-14 21:17:34 UTC	2018-06-06 00:00:00 UTC	2018-06-15 16:42:30 UTC	31	-9

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



```
SELECT distinct c.customer_state,
        avg(oi.freight_value) over (partition by c.customer_state) as Average_Freight
from `Target_SQL_project.customers` as c
join `Target_SQL_project.orders` as o on c.customer_id=o.customer_id
join `Target_SQL_project.order_items` as oi on o.order_id=oi.order_id
order by Average_Freight asc
limit 5
```

Row	customer_state	Average_Freight
1	SP	15.14727539041...
2	PR	20.53165156794...
3	MG	20.63016680630...
4	RJ	20.96092393168...
5	DF	21.04135494596...

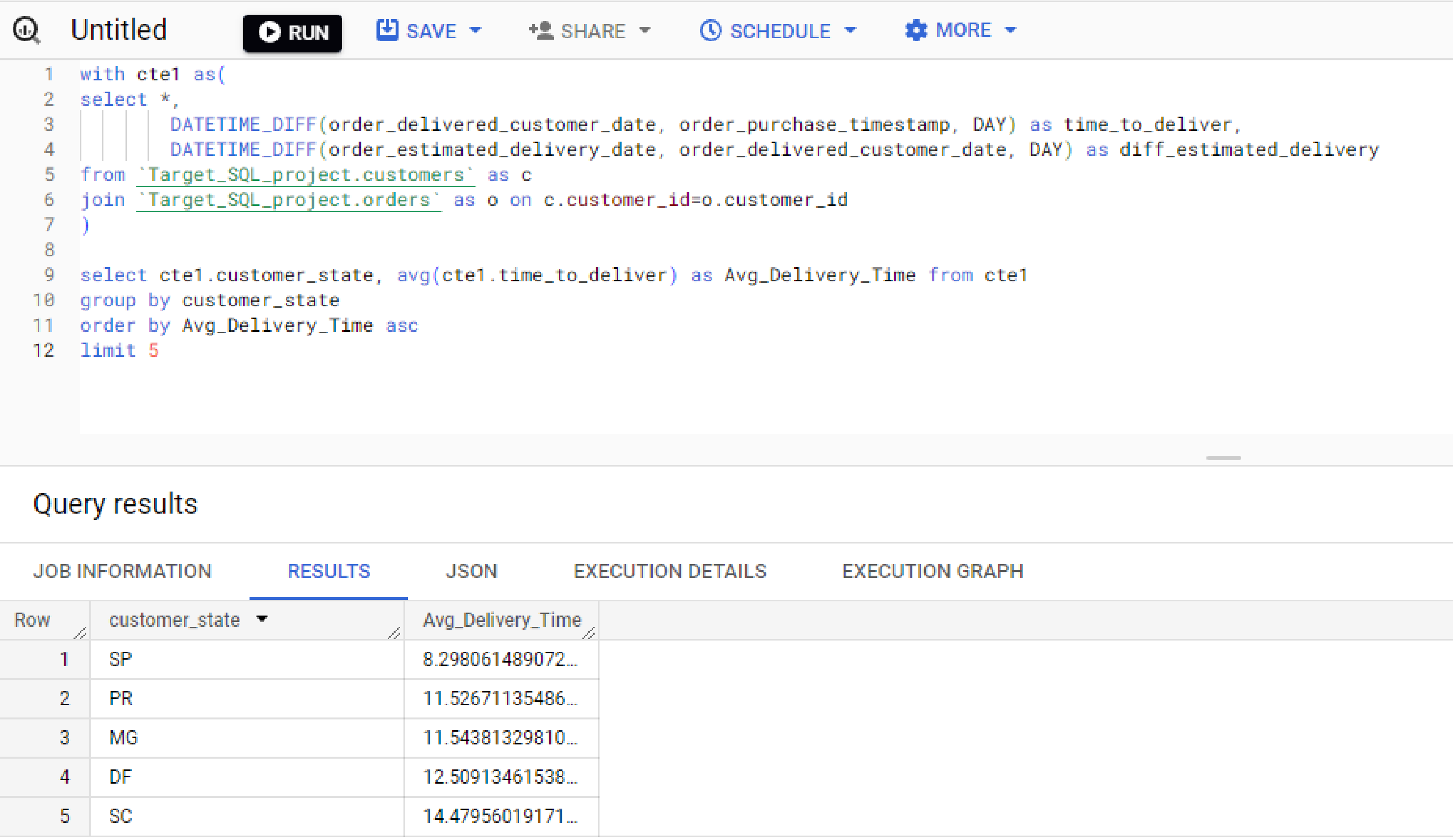


```sql
SELECT distinct c.customer_state,
        avg(oi.freight_value) over (partition by c.customer_state) as Average_Freight
from `Target_SQL_project.customers` as c
join `Target_SQL_project.orders` as o on c.customer_id=o.customer_id
join `Target_SQL_project.order_items` as oi on o.order_id=oi.order_id
order by Average_Freight desc
limit 5
```

Query results

Row	customer_state	Average_Freight
1	RR	42.98442307692...
2	PB	42.72380398671...
3	RO	41.06971223021...
4	AC	40.07336956521...
5	PI	39.14797047970...

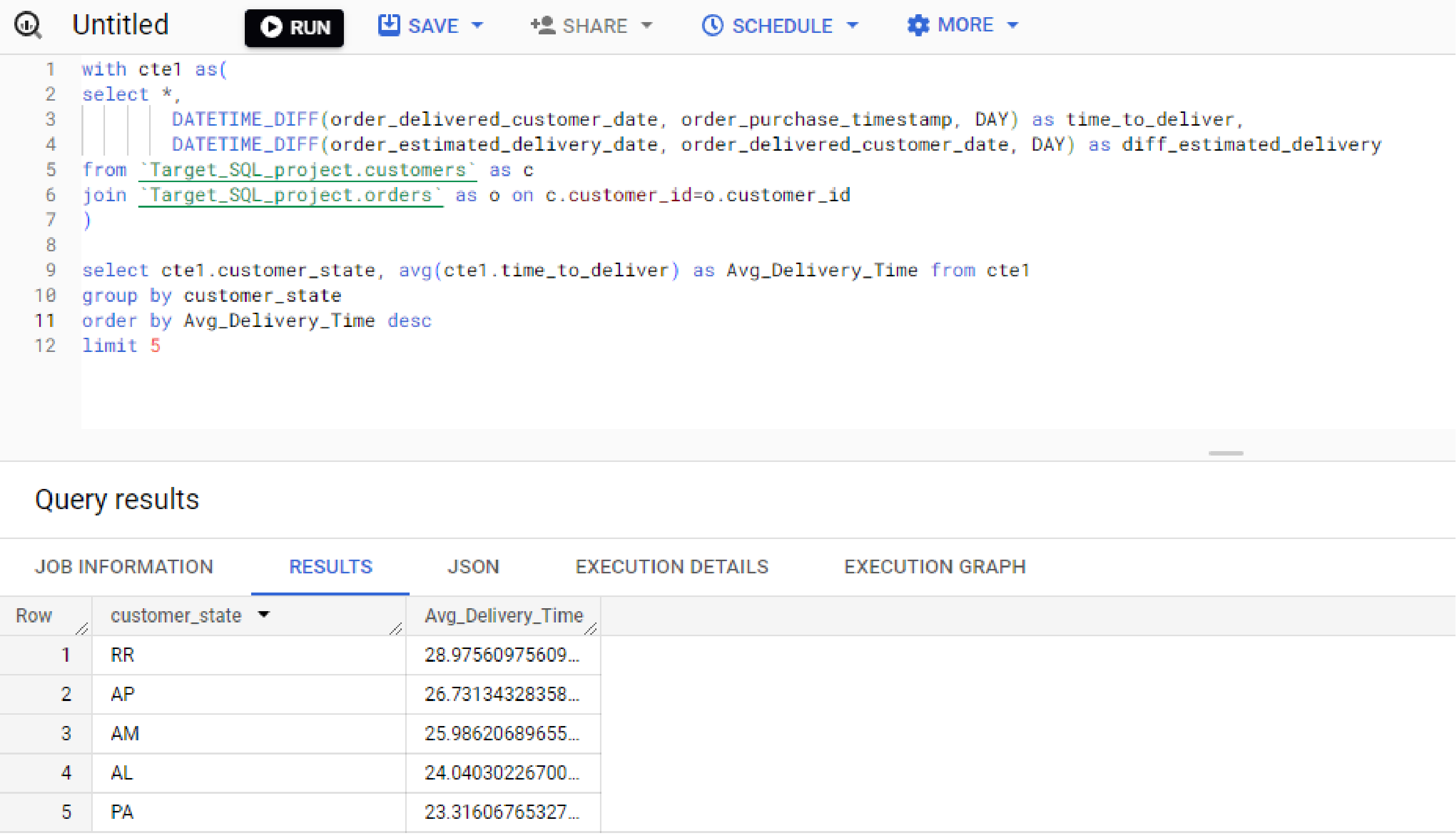
5.3 Find out the top 5 states with the highest & lowest average delivery time.



```
with cte1 as(
select *,
      DATETIME_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY) as time_to_deliver,
      DATETIME_DIFF(order_estimated_delivery_date, order_delivered_customer_date, DAY) as diff_estimated_delivery
from `Target_SQL_project.customers` as c
join `Target_SQL_project.orders` as o on c.customer_id=o.customer_id
)

select cte1.customer_state, avg(cte1.time_to_deliver) as Avg_Delivery_Time from cte1
group by customer_state
order by Avg_Delivery_Time asc
limit 5
```

Query results

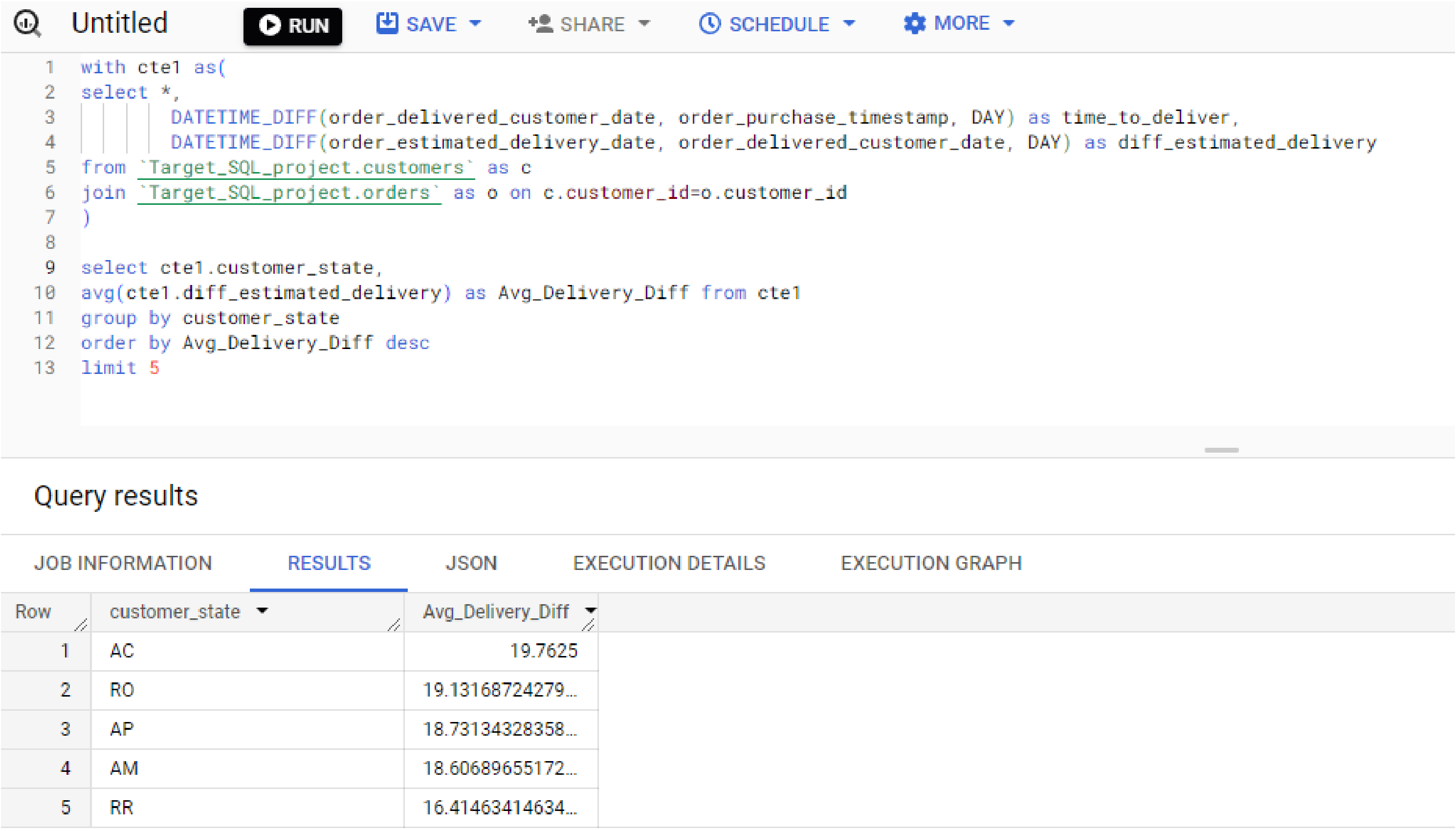
Row	customer_state	Avg_Delivery_Time
1	SP	8.298061489072...
2	PR	11.52671135486...
3	MG	11.54381329810...
4	DF	12.50913461538...
5	SC	14.47956019171...



```sql
with cte1 as(
select *,
        DATETIME_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY) as time_to_deliver,
        DATETIME_DIFF(order_estimated_delivery_date, order_delivered_customer_date, DAY) as diff_estimated_delivery
from `Target_SQL_project.customers` as c
join `Target_SQL_project.orders` as o on c.customer_id=o.customer_id
)

select cte1.customer_state, avg(cte1.time_to_deliver) as Avg_Delivery_Time from cte1
group by customer_state
order by Avg_Delivery_Time desc
limit 5
```

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

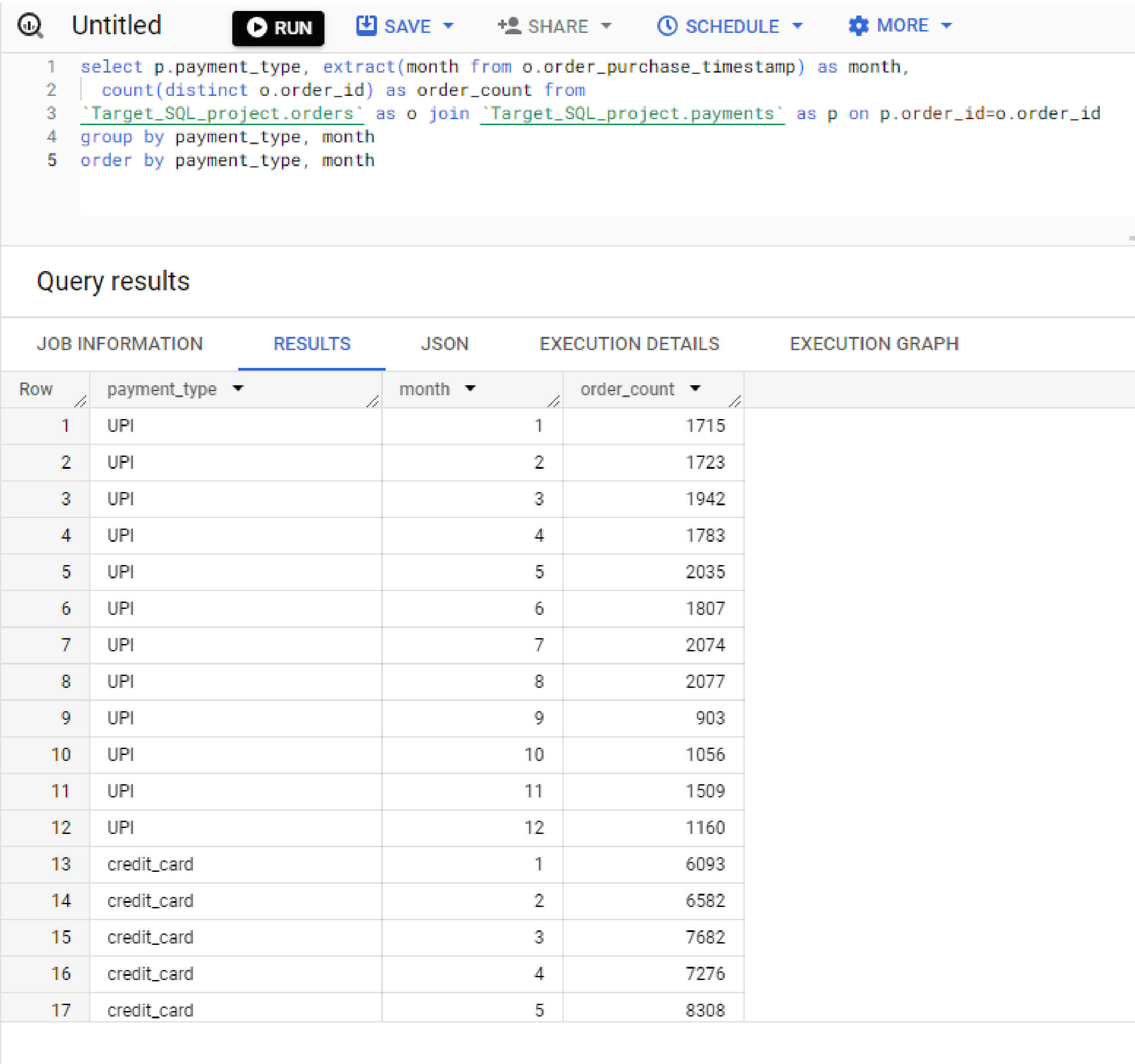
5.4 Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery.
You can use the difference between the averages of actual & estimated delivery date to figure out how fast the delivery was for each state.



```
with cte1 as(
select *,
        DATETIME_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY) as time_to_deliver,
        DATETIME_DIFF(order_estimated_delivery_date, order_delivered_customer_date, DAY) as diff_estimated_delivery
from `Target_SQL_project.customers` as c
join `Target_SQL_project.orders` as o on c.customer_id=o.customer_id
)

select cte1.customer_state,
avg(cte1.diff_estimated_delivery) as Avg_Delivery_Diff from cte1
group by customer_state
order by Avg_Delivery_Diff desc
limit 5
```

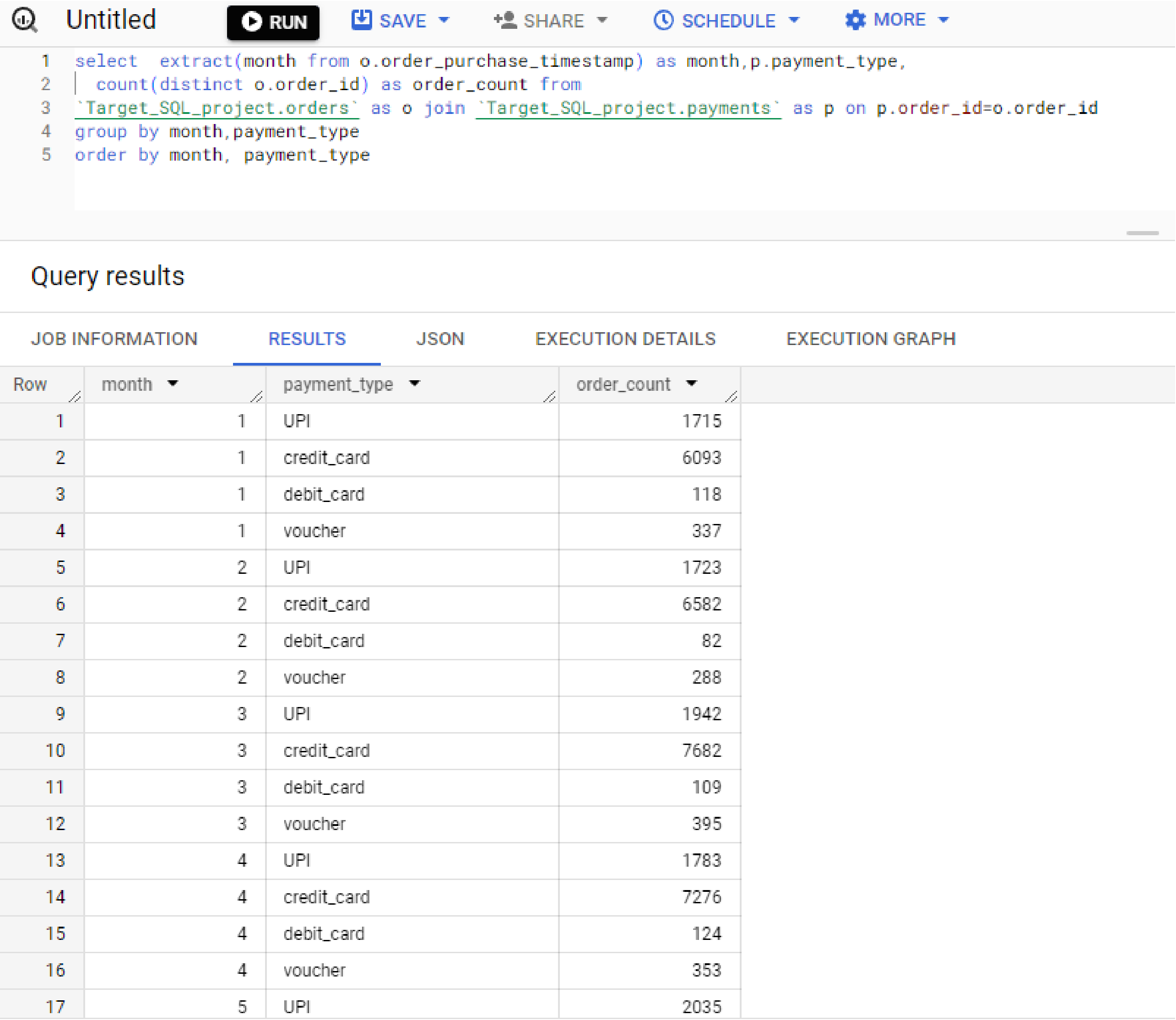
Row	customer_state	Avg_Delivery_Diff
1	AC	19.7625
2	RO	19.13168724279...
3	AP	18.73134328358...
4	AM	18.60689655172...
5	RR	16.41463414634...

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



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

Row	payment_type	month	order_count
1	UPI	1	1715
2	UPI	2	1723
3	UPI	3	1942
4	UPI	4	1783
5	UPI	5	2035
6	UPI	6	1807
7	UPI	7	2074
8	UPI	8	2077
9	UPI	9	903
10	UPI	10	1056
11	UPI	11	1509
12	UPI	12	1160
13	credit_card	1	6093
14	credit_card	2	6582
15	credit_card	3	7682
16	credit_card	4	7276
17	credit_card	5	8308

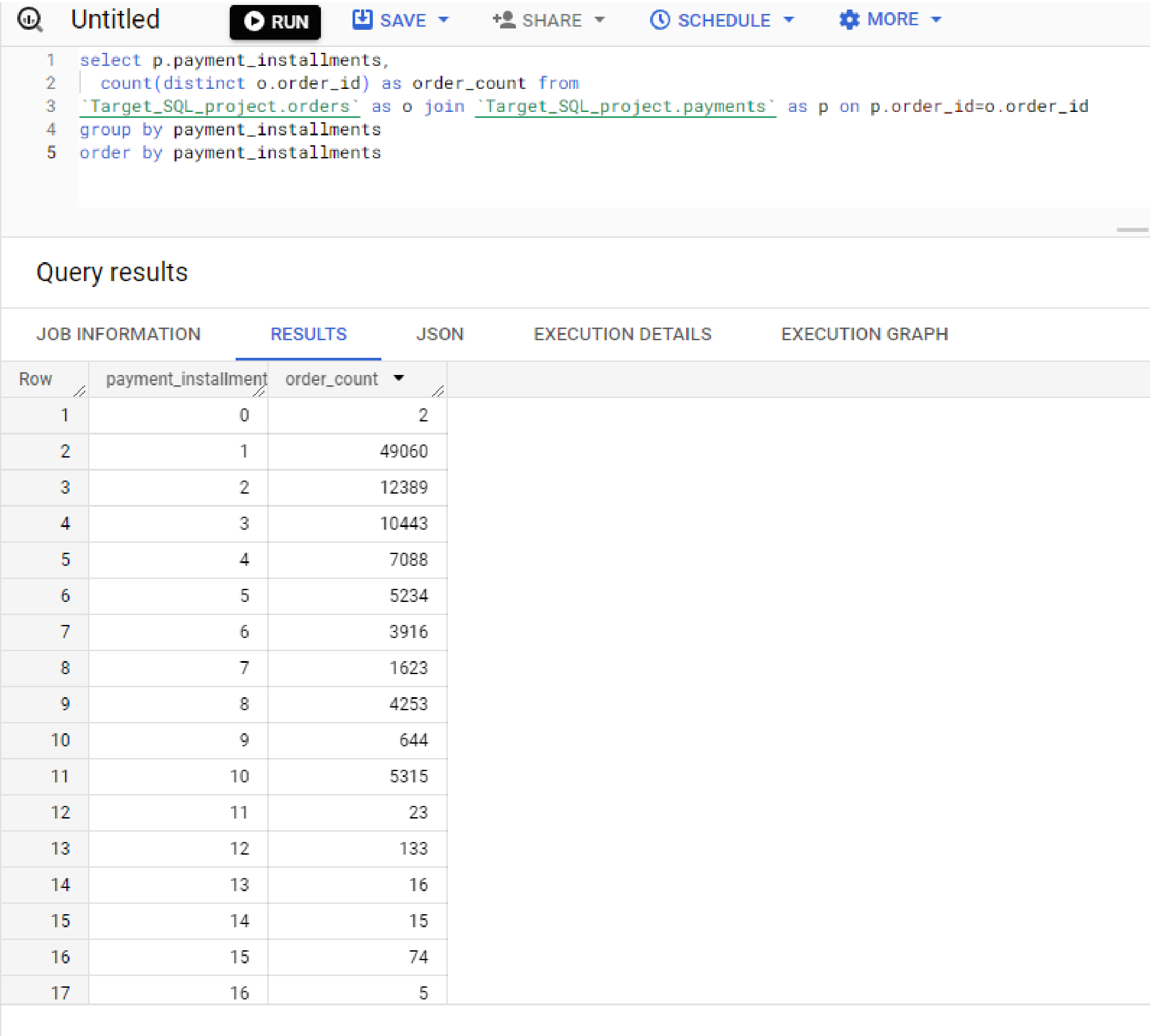


```
select  extract(month from o.order_purchase_timestamp) as month,p.payment_type,
  count(distinct o.order_id) as order_count from
`Target_SQL_project.orders` as o join `Target_SQL_project.payments` as p on p.order_id=o.order_id
group by month,payment_type
order by month, payment_type
```

Query results

Row	month	payment_type	order_count
1	1	UPI	1715
2	1	credit_card	6093
3	1	debit_card	118
4	1	voucher	337
5	2	UPI	1723
6	2	credit_card	6582
7	2	debit_card	82
8	2	voucher	288
9	3	UPI	1942
10	3	credit_card	7682
11	3	debit_card	109
12	3	voucher	395
13	4	UPI	1783
14	4	credit_card	7276
15	4	debit_card	124
16	4	voucher	353
17	5	UPI	2035

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



```sql
select p.payment_installments,
  count(distinct o.order_id) as order_count from
`Target_SQL_project.orders` as o join `Target_SQL_project.payments` as p on p.order_id=o.order_id
group by payment_installments
order by payment_installments
```

Row	payment_installment	order_count
1	0	2
2	1	49060
3	2	12389
4	3	10443
5	4	7088
6	5	5234
7	6	3916
8	7	1623
9	8	4253
10	9	644
11	10	5315
12	11	23
13	12	133
14	13	16
15	14	15
16	15	74
17	16	5