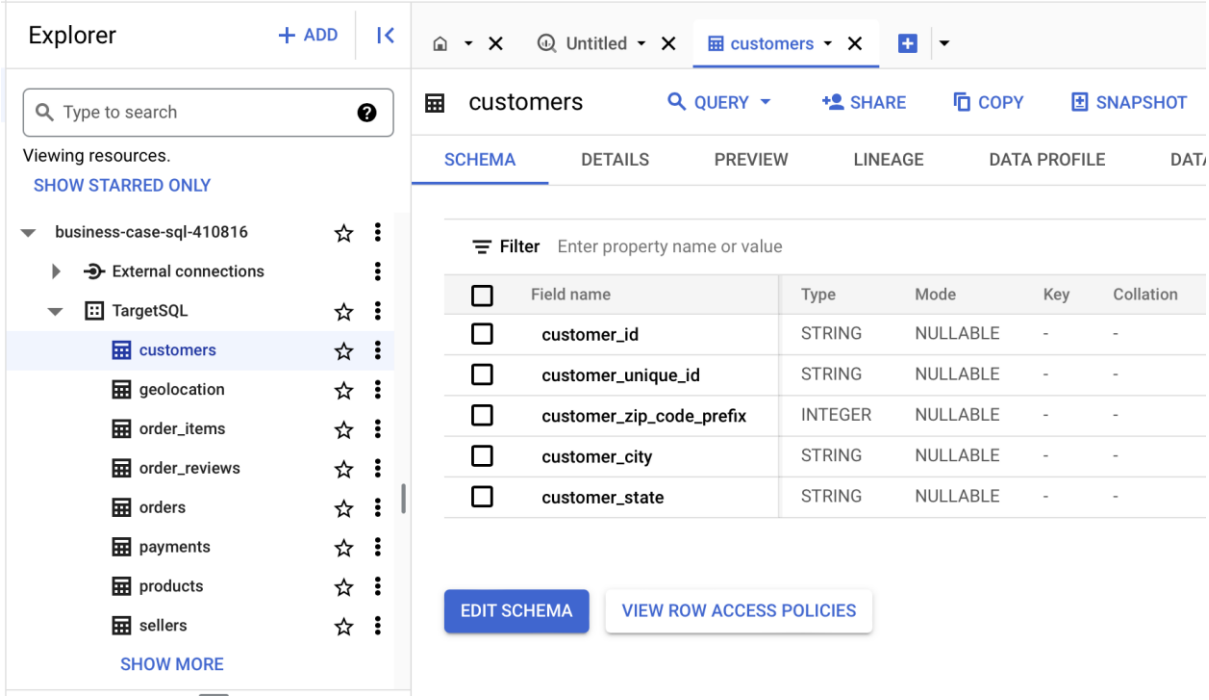
1) **Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset:**

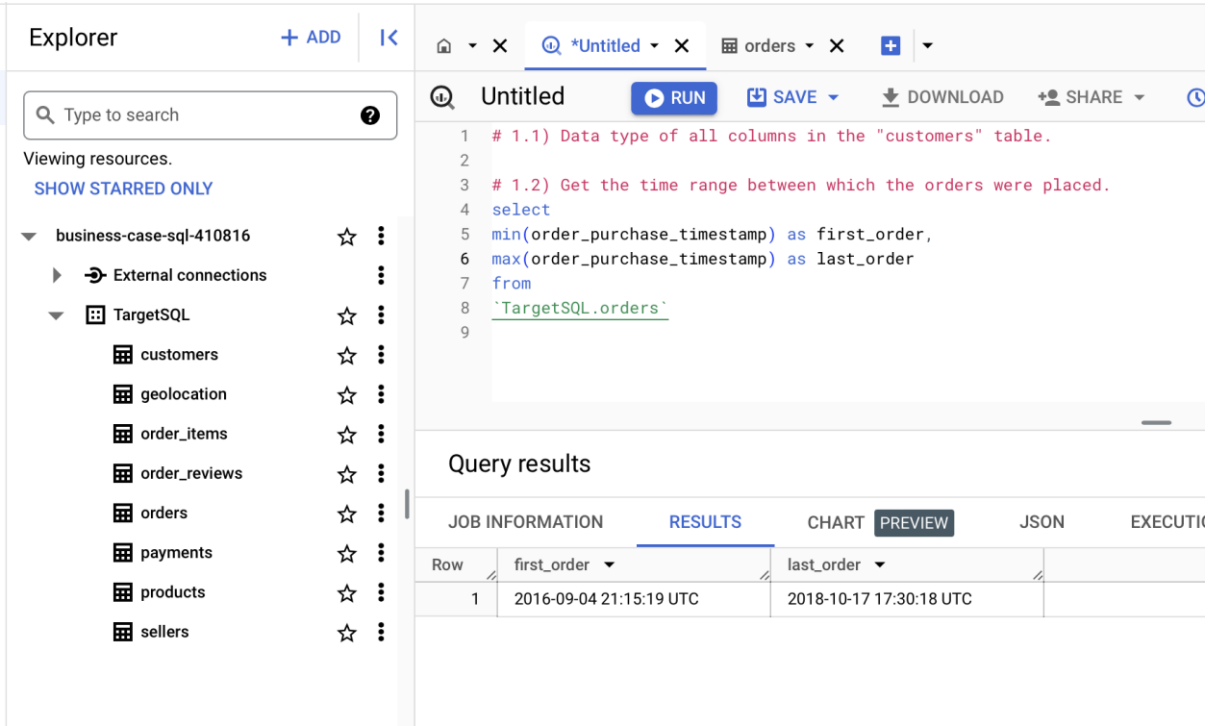
1.1) Data type of all columns in the "customers" table.



Observation:
All columns are of data type “STRING” except ‘customer_zip_code_prefix’ which is of type INTEGER

2.2) Get the time range between which the orders were placed.

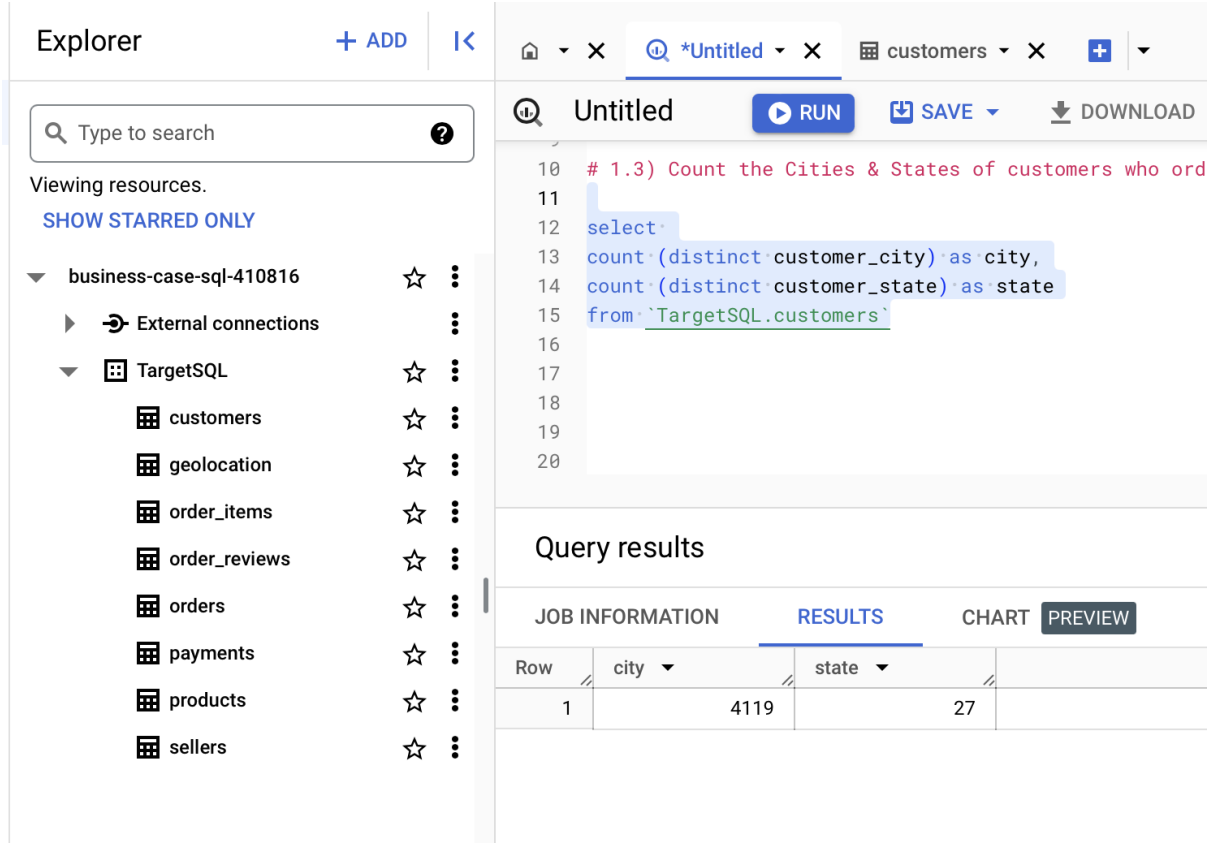
```
select
max(order_purchase_timestamp) as max_date,
min(order_purchase_timestamp) as min_date
from `TargetSQL.orders`
```



Observations:
The first order was placed on 04th September 2016 and the last order was placed on 17th October 2018

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

```
select
count(distinct customer_city),
count(distinct customer_state)
from `TargetSQL.customers`
```



Observations

There are 4119 cities and 27 states, where customers who ordered during the given period (2016-09-04 and 2018-10-17)

2) In-depth Exploration:

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

```
select
extract (year from order_purchase_timestamp) as year,
count (distinct order_id) as num_orders
from `TargetSQL.orders`
group by 1
order by 1
```

*Untitled orders

Untitled RUN SAVE DOWNLOAD SHARE SCHEDULE MO

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

select
extract (year from order_purchase_timestamp) as year,
count (distinct order_id) as num_orders
from `TargetSQL.orders`
group by 1
order by 1
```

Query results

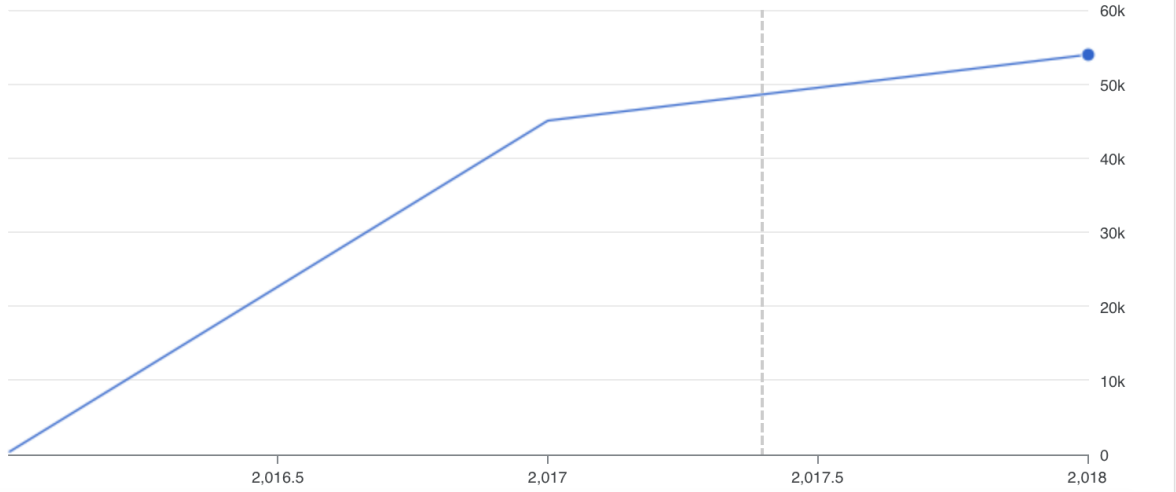
JOB INFORMATION RESULTS CHART PREVIEW JSON EXECUTION DETAILS EX

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

Query results SAVI

JOB INFORMATION RESULTS CHART PREVIEW JSON EXECUTION DETAILS EXECUTION

num_orders by year

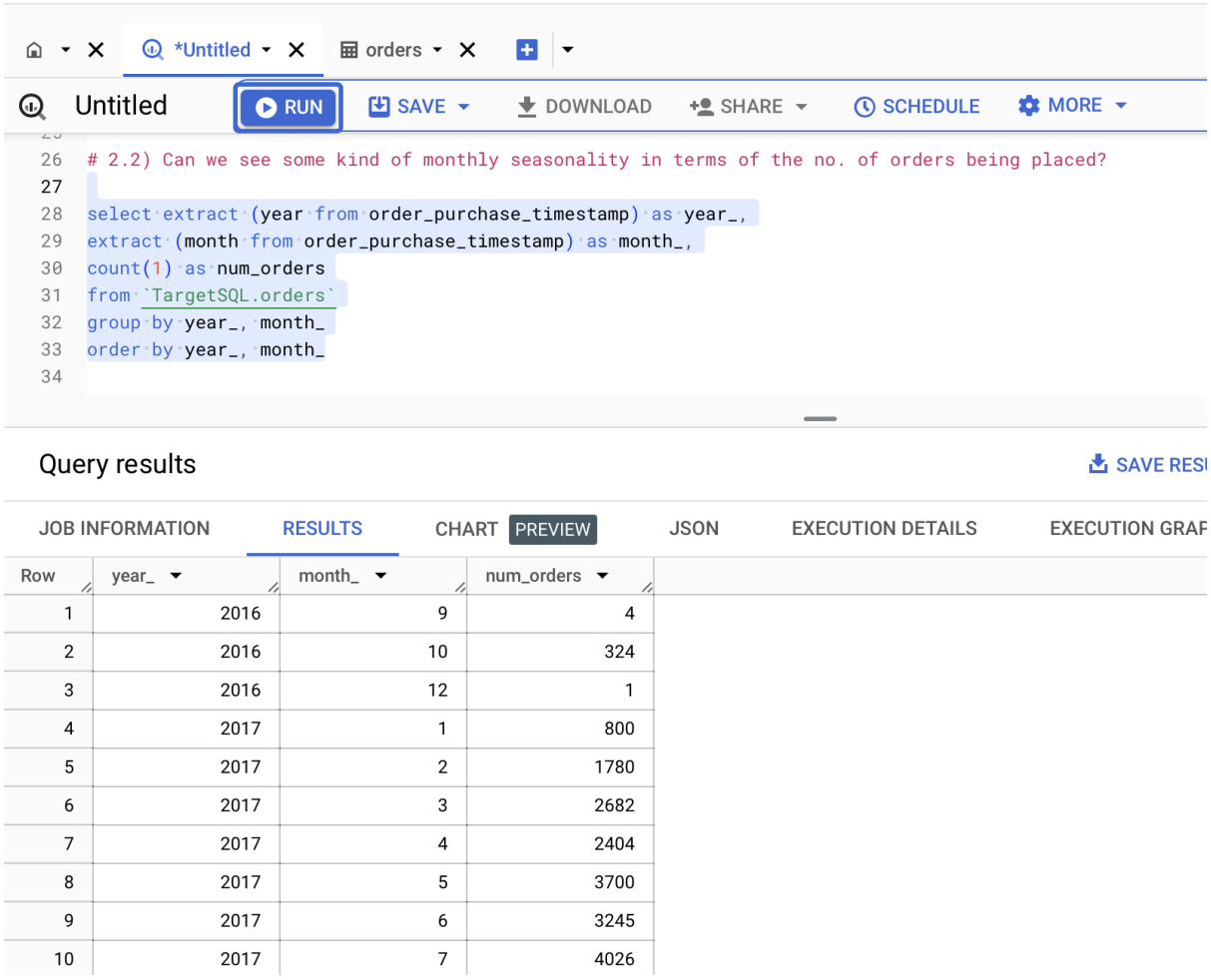


Observations:

In the initial year 2016 the number of orders placed was very minimum but in 2017 the number of orders were show some rapid increase and in 2018, the trend continues

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

```
select extract (year from order_purchase_timestamp) as year_,
extract (month from order_purchase_timestamp) as month_,
count(1) as num_orders
from `TargetSQL.orders`
group by year_, month_
order by year_, month_
```



Row	year_	month_	num_orders
1	2016	9	4
2	2016	10	324
3	2016	12	1
4	2017	1	800
5	2017	2	1780
6	2017	3	2682
7	2017	4	2404
8	2017	5	3700
9	2017	6	3245
10	2017	7	4026

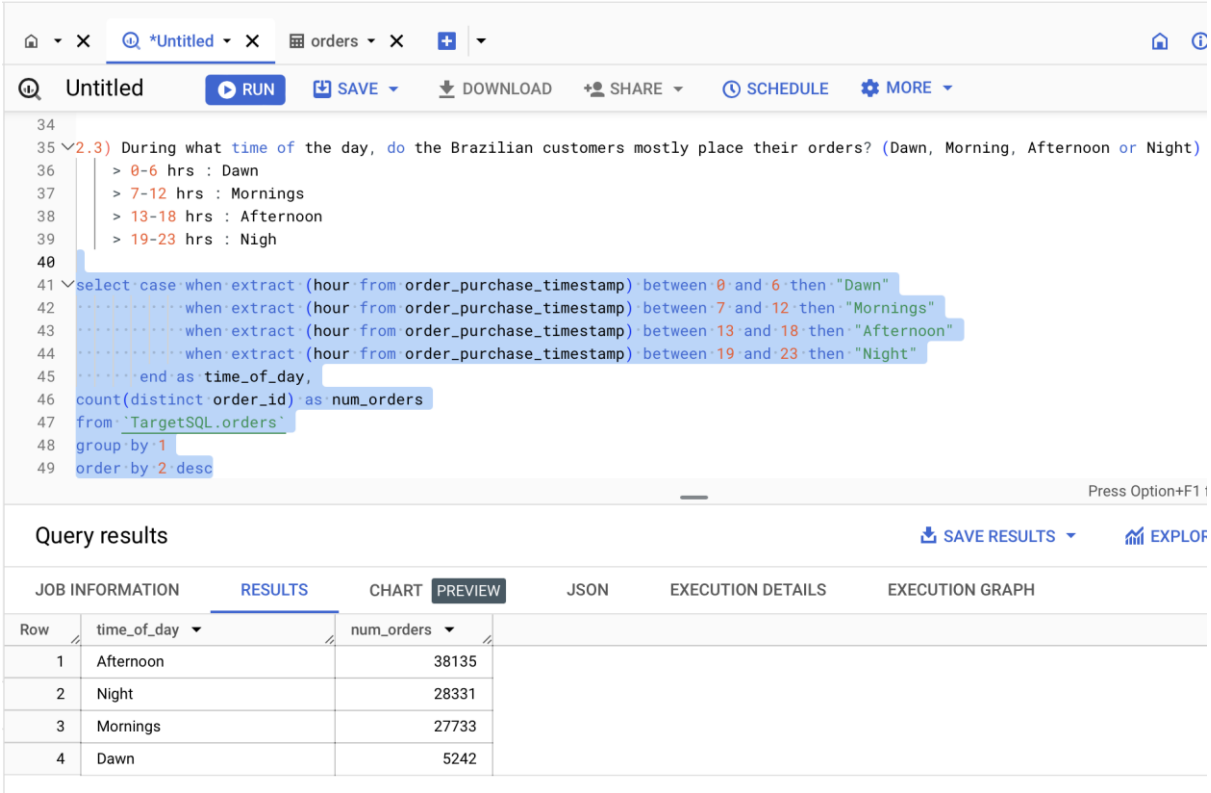
The last months of 2016 the number of orders were placed very less but in 2017 it showed a gradual increase up to October and on November it spikes to a very large value (7544). Then it sustained around that value till 2018 August but in the last two months the value declined dramatically.

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

```
select case when extract (hour from order_purchase_timestamp) between 0 and 6 then
"Dawn"
            when extract (hour from order_purchase_timestamp) between 7 and 12 then
"Mornings"
            when extract (hour from order_purchase_timestamp) between 13 and 18
then "Afternoon"
            when extract (hour from order_purchase_timestamp) between 19 and 23
then "Night"
       end as time_of_day,
count(distinct order_id) as num_orders
from `TargetSQL.orders`
group by 1
order by 2 desc
```



Observations:
During Afternoon the Brazilian customers mostly place their orders, also the least placed order was on Dawn time

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

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

```
select c.customer_state,
```

```sql
extract (year from o.order_purchase_timestamp) as year_,
extract (month from o.order_purchase_timestamp) as month_,
count(distinct o.order_id) as num_order
from `TargetSQL.orders` o
inner join `TargetSQL.customers` c
on o.customer_id = c.customer_id
group by 1, 2, 3
order by 4 desc
```

*Untitled | orders | customers

Untitled

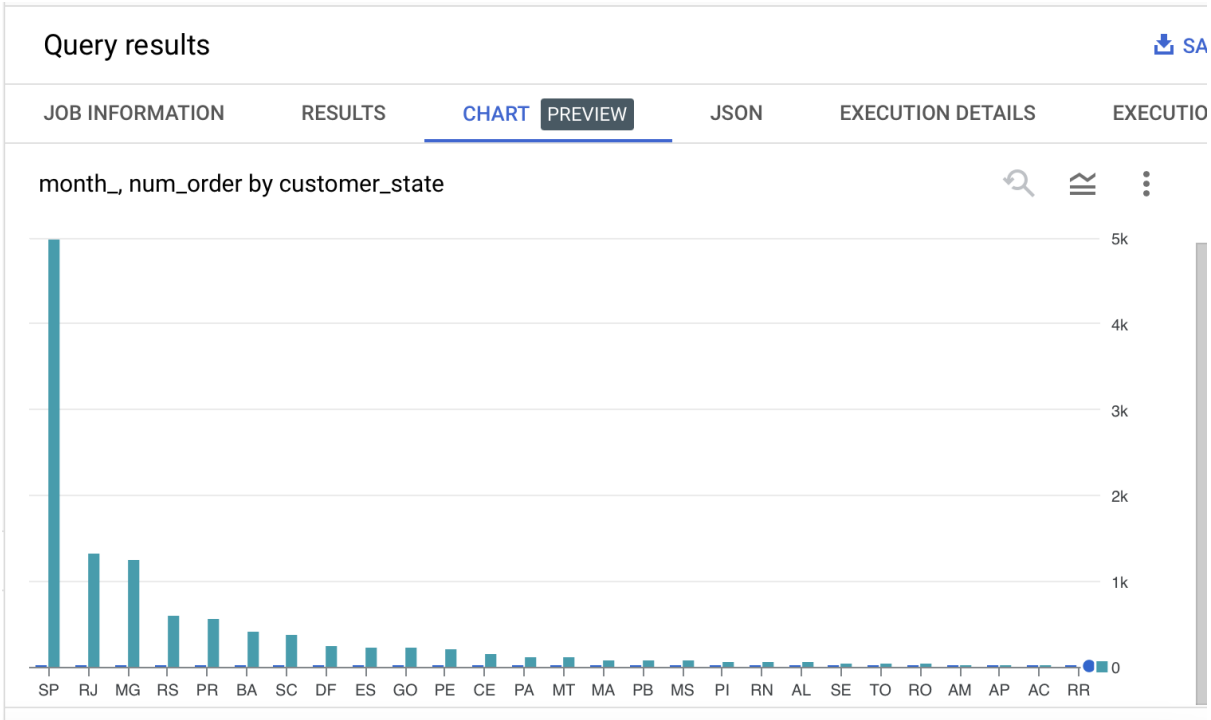
RUN | SAVE | DOWNLOAD | SHARE | SCHEDULE

```sql
# 3.1) Get the month on month no. of orders placed in each state.
select c.customer_state,
extract (year from o.order_purchase_timestamp) as year_,
extract (month from o.order_purchase_timestamp) as month_,
count(distinct o.order_id) as num_order
from `TargetSQL.orders` o
inner join `TargetSQL.customers` c
on o.customer_id = c.customer_id
group by 1, 2, 3
order by 4 desc
```

Query results

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAILS

Row	customer_state	year_	month_	num_order
1	SP	2018	8	3253
2	SP	2018	5	3207
3	SP	2018	4	3059
4	SP	2018	1	3052
5	SP	2018	3	3037
6	SP	2017	11	3012
7	SP	2018	7	2777
8	SP	2018	6	2773
9	SP	2018	2	2703
10	SP	2017	12	2357



A huge number of orders placed in SP and other states placed very less number of orders.

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

```
select customer_state,
count(distinct customer_id) customer_num
from `TargetSQL.customers`
group by 1
order by 2 desc
```

*Untitled | orders | customers

Untitled — RUN | SAVE | DOWNLOAD | SHARE | SCHEDULE

```
# 3.2) How are the customers distributed across all the states?
select customer_state,
count(distinct customer_id) customer_num
from `TargetSQL.customers`
group by 1
order by 2 desc
```

Query results

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAILS

Row	customer_state	customer_num
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

Most of the customers are of the state SP and states RJ & MG have customers above 10K while, rest of the states have very less number of customers

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.

```
with base_1 as
(
select
*
from `TargetSQL.payments` p
inner join `TargetSQL.orders` o
```

```
on p.order_id = o.order_id
where extract (year from o.order_purchase_timestamp) between 2017 and 2018
and extract (month from o.order_purchase_timestamp) between 1 and 8
),
base_2 as
(
select
  extract (year from order_purchase_timestamp) as year,
  sum (payment_value) as cost
from base_1
group by 1
order by 1
),
base_3 as
(
  select *,
  lead (cost) over (order by year) as next_year
  from base_2
)
select * ,
round((next_year - cost)/ cost * 100, 2) as percent_increase
from base_3
```

Untitled RUN SAVE DOWNLOAD SHARE SCHEDULE MORE

```
#     You can use the "payment_value" column in the payments table to get the cost of orders.
with base_1 as
(
select
*
from `TargetSQL.payments` p
inner join `TargetSQL.orders` o
on p.order_id = o.order_id
where extract (year from o.order_purchase_timestamp) between 2017 and 2018
and extract (month from o.order_purchase_timestamp) between 1 and 8
),
base_2 as
(
select
  extract (year from order_purchase_timestamp) as year,
  sum (payment_value) as cost
from base_1
group by 1
order by 1
),
base_3 as
(
  select *,
  lead (cost) over (order by year) as next_year
  from base_2
)
select * ,
round((next_year - cost)/ cost * 100, 2) as percent_increase
from base_3
```

Query results

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DET

Row	year	cost	next_year	percent_increase
1	2018	8694733.839999...	null	null
2	2017	3669022.120000...	8694733.839999...	136.98

The percentage increase from 2017 to 2018 (include months between Jan to Aug) is 136.98

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

```
select
c.customer_state,
sum (oi.price) as total_value,
avg (oi.price) as avg_price
from
`TargetSQL.order_items` oi inner join `TargetSQL.orders` o
on oi.order_id = o.order_id
inner join `TargetSQL.customers` c
on c.customer_id = o.customer_id
group by customer_state
order by 2 desc
```

*Untitled | orders | order_items | custom

Untitled

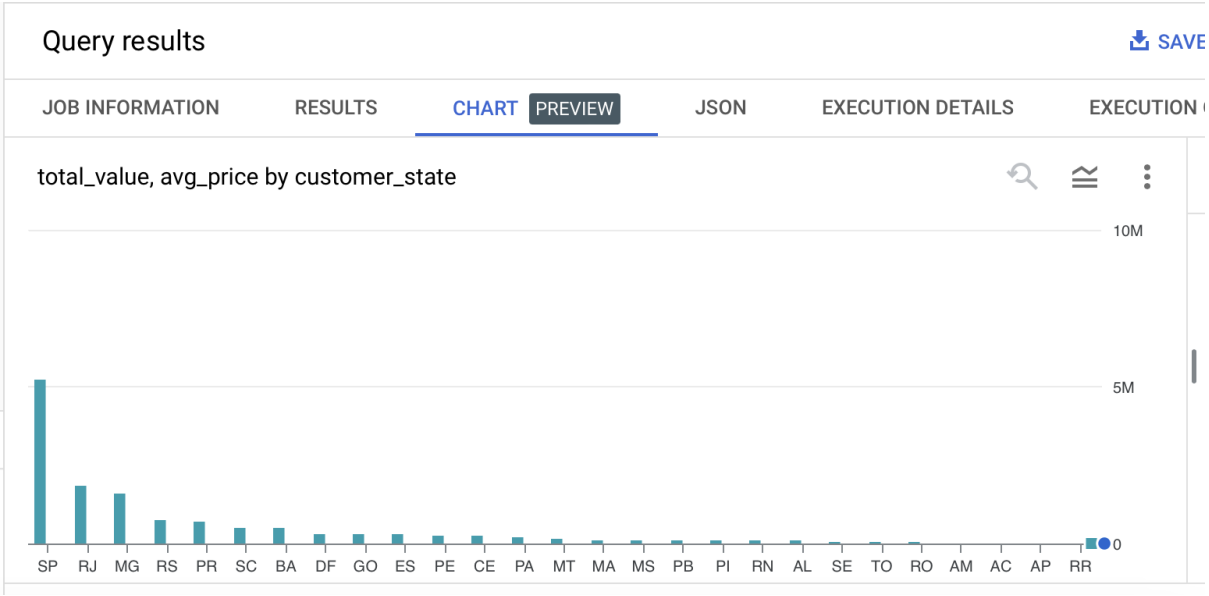
RUN | SAVE | DOWNLOAD | SHARE

```
select
c.customer_state,
sum (oi.price) as total_value,
avg (oi.price) as avg_price
from
`TargetSQL.order_items` oi inner join `TargetSQL.orders` o
on oi.order_id = o.order_id
inner join `TargetSQL.customers` c
on c.customer_id = o.customer_id
group by customer_state
order by 2 desc
```

Query results

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EX

Row	customer_state	total_value	avg_price
1	SP	5202955.050001...	109.6536291597...
2	RJ	1824092.669999...	125.1178180945...
3	MG	1585308.029999...	120.7485741488...
4	RS	750304.0200000...	120.3374530874...
5	PR	683083.7600000...	119.0041393728...
6	SC	520553.3400000...	124.6535775862...
7	BA	511349.9900000...	134.6012082126...
8	DF	302603.9399999...	125.7705486284...
9	GO	294591.9499999...	126.2717316759...
10	ES	275037.3099999...	121.9137012411...



The total value is very high in the state SP while all other state are very less compared to the state SP, but the average price is around 100

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

```
select c.customer_state, sum(oi.freight_value) total_value, avg(oi.freight_value)
avg_price
from `TargetSQL.order_items` oi left join `TargetSQL.orders` o
on oi.order_id = o.order_id
left join `TargetSQL.customers` c
on o.customer_id = c.customer_id
group by 1
```

*Untitled | order_items | orders | customers

Untitled — RUN | SAVE | DOWNLOAD | SHARE | SCHEDULE | MORE

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

select c.customer_state, sum(oi.freight_value) total_value, avg(oi.freight_value) avg_price
from `TargetSQL.order_items` oi left join `TargetSQL.orders` o
on oi.order_id = o.order_id
left join `TargetSQL.customers` c
on o.customer_id = c.customer_id
group by 1
```

Query results — SAVE RES

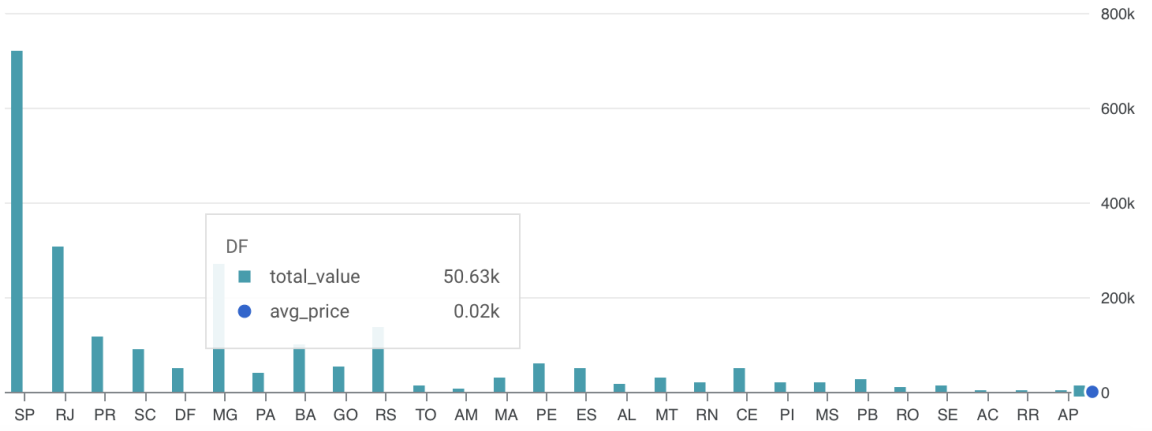
JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAILS | EXECUTION GRAI

Row	customer_state	total_value	avg_price
1	SP	718723.0699999...	15.14727539041...
2	RJ	305589.3100000...	20.96092393168...
3	PR	117851.6800000...	20.53165156794...
4	SC	89660.26000000...	21.47036877394...
5	DF	50625.49999999...	21.04135494596...
6	MG	270853.4600000...	20.63016680630...
7	PA	38699.30000000...	35.83268518518...
8	BA	100156.6799999...	26.36395893656...
9	GO	53114.97999999...	22.76681525932...
10	RS	135522.7400000...	21.73580433039...

Query results — SAV

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAILS | EXECUTION

total_value, avg_price by customer_state



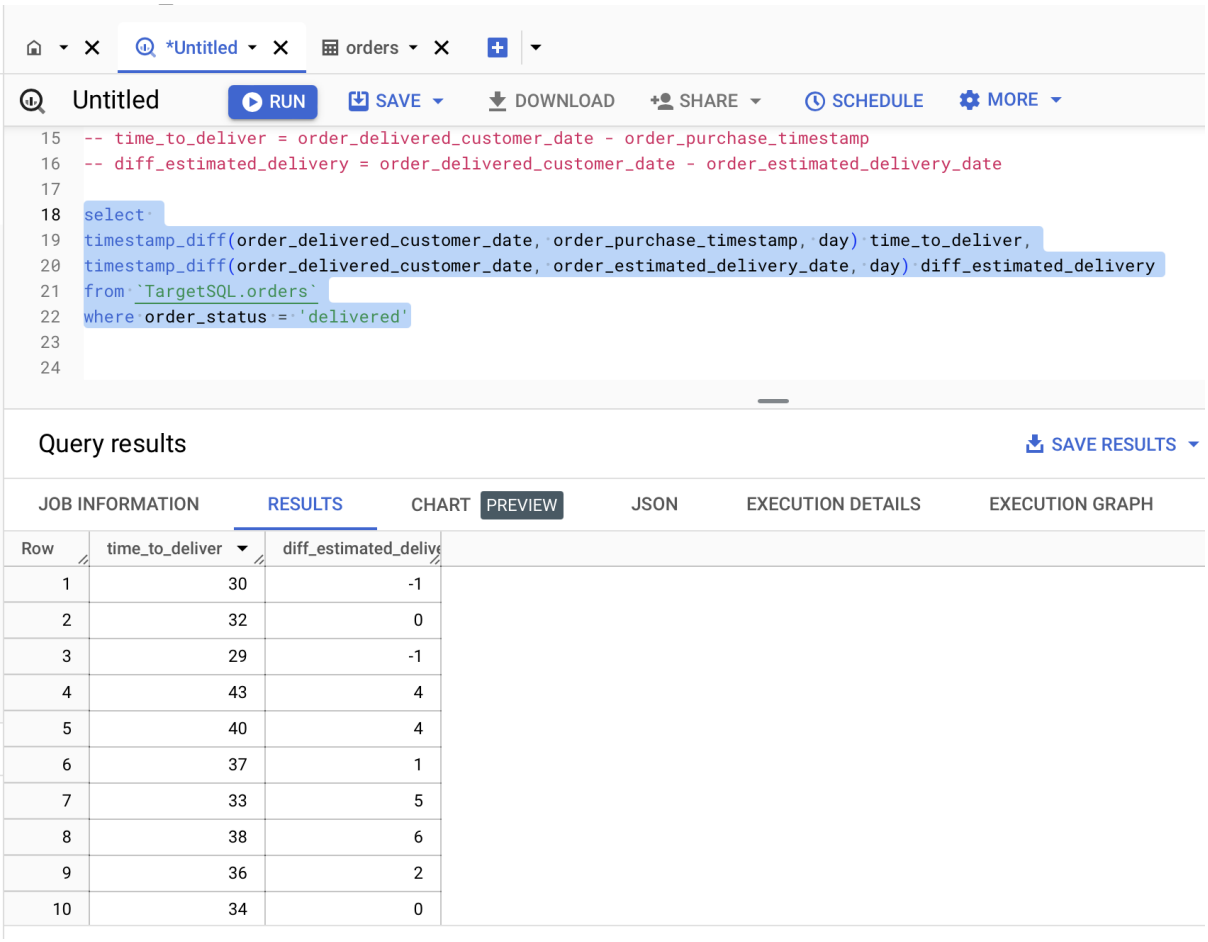
The highest of total value is in SP (~718.72K) while all other states shows significantly very less values. The average price is around 0.02K for most of the states.

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_delivered_customer_date - order_estimated_delivery_date

```
select
timestamp_diff(order_delivered_customer_date, order_purchase_timestamp, day)
time_to_deliver,
timestamp_diff(order_delivered_customer_date, order_estimated_delivery_date, day)
diff_estimated_delivery
from `TargetSQL.orders`
where order_status = 'delivered'
```



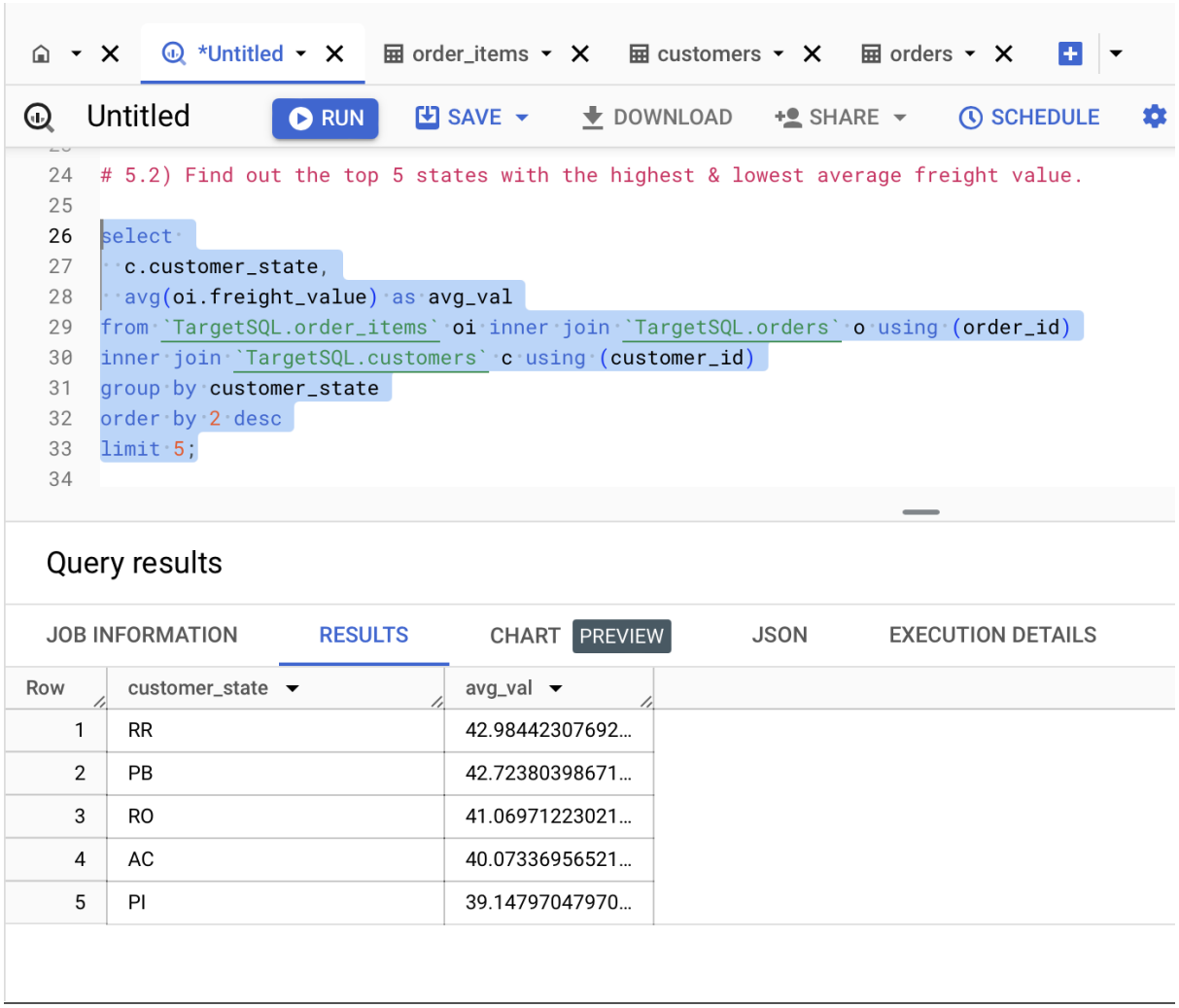
Row	time_to_deliver	diff_estimated_delive
1	30	-1
2	32	0
3	29	-1
4	43	4
5	40	4
6	37	1
7	33	5
8	38	6
9	36	2
10	34	0

The time to deliver ranges around 30 to 50 but difference between the estimated & actual delivery shows a significant range difference.

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

Highest

```
select
  c.customer_state,
  avg(oi.freight_value) as avg_val
from `TargetSQL.order_items` oi inner join `TargetSQL.orders` o using (order_id)
inner join `TargetSQL.customers` c using (customer_id)
group by customer_state
order by 2 desc
limit 5;
```



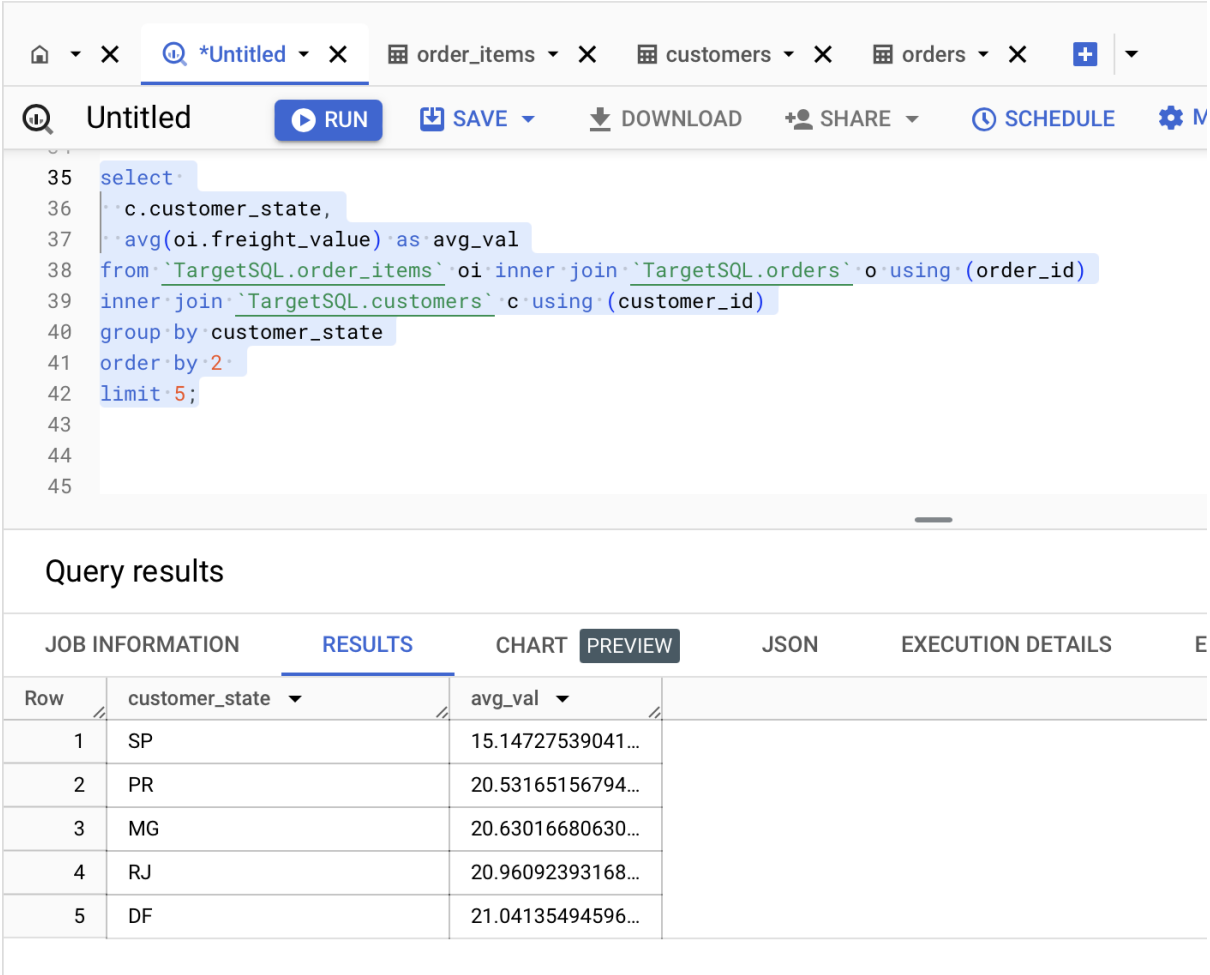
Row	customer_state	avg_val
1	RR	42.98442307692...
2	PB	42.72380398671...
3	RO	41.06971223021...
4	AC	40.07336956521...
5	PI	39.14797047970...

The highest average freight value ranges around 40 and the top one state is RR

Lowest

```
select
  c.customer_state,
```

```
  avg(oi.freight_value) as avg_val
from `TargetSQL.order_items` oi inner join `TargetSQL.orders` o using (order_id)
inner join `TargetSQL.customers` c using (customer_id)
group by customer_state
order by 2
limit 5;
```



The lowest average freight value is ~15 in SP and the 5th lowest is in DF

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

<u>Highest</u>

```
select
  c.customer_state,
  extract (time from order_delivered_customer_date) time_
from `TargetSQL.orders` oi inner join `TargetSQL.order_items` o using (order_id)
inner join `TargetSQL.customers` c using (customer_id)
group by 1, 2
order by 2 desc
limit 5
```

*Untitled | order_items | customers | orders

Untitled — RUN | SAVE | DOWNLOAD | SHARE | SCHEDULE | MO

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

select
  c.customer_state,
  extract (time from order_delivered_customer_date) time_
from `TargetSQL.orders` oi inner join `TargetSQL.order_items` o using (order_id)
inner join `TargetSQL.customers` c using (customer_id)
group by 1, 2
order by 2 desc
limit 5
```

Query results

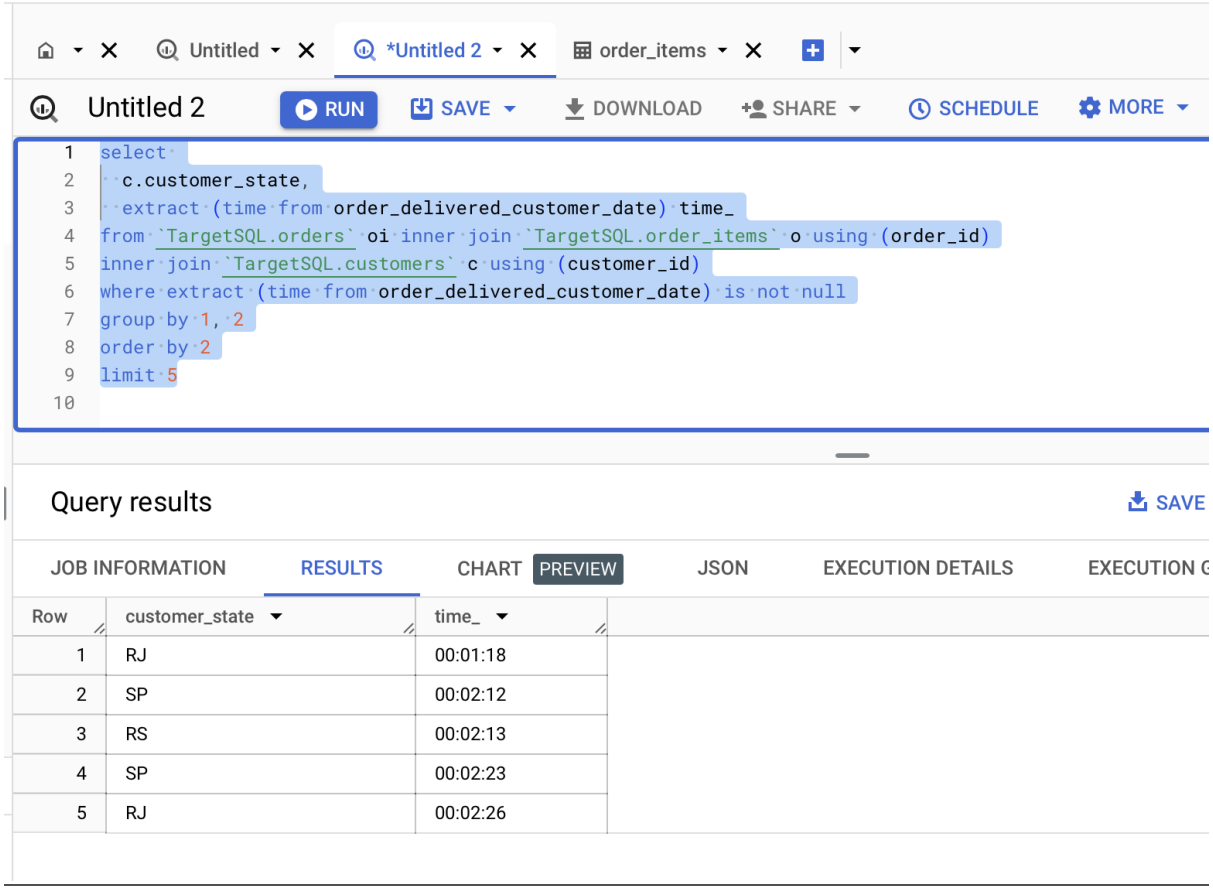
JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAILS | EXE

Row	customer_state	time_
1	MG	23:59:59
2	BA	23:59:55
3	RJ	23:59:52
4	SP	23:59:52
5	SP	23:59:49

The highest average delivery time is in MG.

<u>Lowest</u>

```
select
  c.customer_state,
  extract (time from order_delivered_customer_date) time_
from `TargetSQL.orders` oi inner join `TargetSQL.order_items` o using (order_id)
inner join `TargetSQL.customers` c using (customer_id)
where extract (time from order_delivered_customer_date) is not null
group by 1, 2
order by 2
limit 5
```



The lowest of average delivery time is 00.01.18 in RJ

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.

```
SELECT
customer_state AS state,
ROUND(SUM(TIMESTAMP_DIFF(order_delivered_customer_date, order_purchase_timestamp,
DAY))/COUNT(ORDER_ID), 2) AS average_time_for_del,
ROUND(SUM(TIMESTAMP_DIFF(order_estimated_delivery_date, order_purchase_timestamp,
DAY))/COUNT(ORDER_ID), 2) AS average_est_dil_time,
FROM `TargetSQL.orders` o
INNER JOIN `TargetSQL.customers` c
ON o.customer_id=c.customer_id
WHERE order_status='delivered'
GROUP BY customer_state
ORDER BY (average_time_for_del-average_est_dil_time)
```

*Untitled | order_items | customers | orders

Untitled — RUN | SAVE | DOWNLOAD | SHARE | SCHEDULE | MORE

```
SELECT
customer_state AS state,
ROUND(SUM(TIMESTAMP_DIFF(order_delivered_customer_date, order_purchase_timestamp,
DAY))/COUNT(ORDER_ID), 2) AS average_time_for_del,
ROUND(SUM(TIMESTAMP_DIFF(order_estimated_delivery_date, order_purchase_timestamp,
DAY))/COUNT(ORDER_ID), 2) AS average_est_dil_time,
FROM `TargetSQL.orders` o
INNER JOIN `TargetSQL.customers` c
ON o.customer_id=c.customer_id
WHERE order_status='delivered'
GROUP BY customer_state
ORDER BY (average_time_for_del-average_est_dil_time)
```

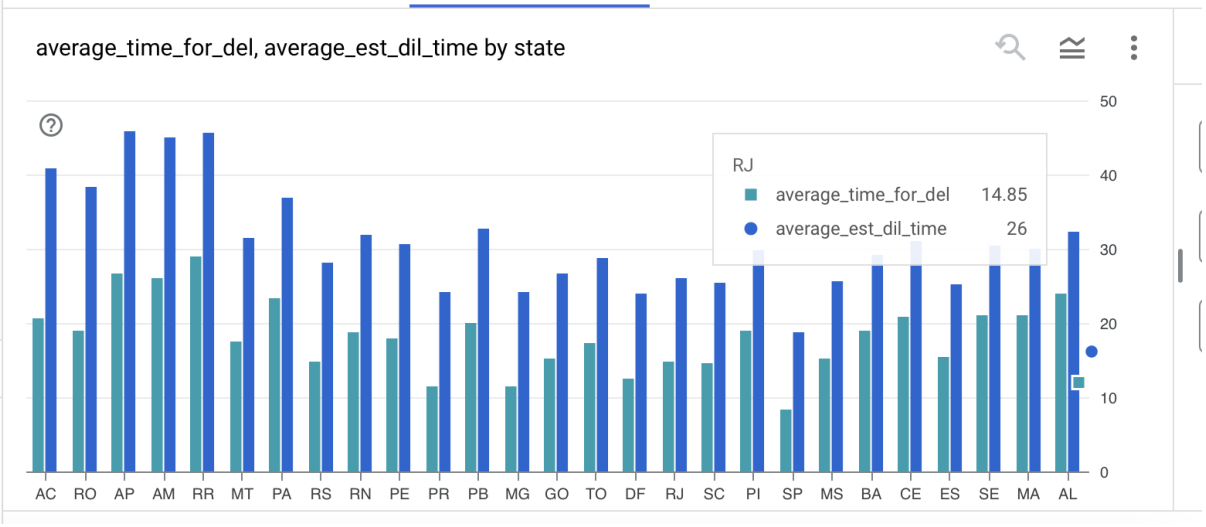
Query results

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAILS | EXECU

Row	state	average_time_for_del	average_est_dil_time
1	AC	20.64	40.73
2	RO	18.91	38.39
3	AP	26.73	45.87
4	AM	25.99	44.92
5	RR	28.98	45.63
6	MT	17.59	31.37
7	PA	23.32	36.79
8	RS	14.82	28.16
9	RN	18.82	31.87

Query results — SAVE

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAILS | EXECUTION G



Average delivery time for the top 5 states is around 40.

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

```
select
p.payment_type,
extract(month from o.order_purchase_timestamp) as month_,
count(distinct p.order_id) as num_order
from `TargetSQL.payments` p inner join `TargetSQL.orders` o using(order_id)
group by 1, month_
order by month_, num_order desc
```

*Untitled | order_items | customers | orders

Untitled — RUN | SAVE | DOWNLOAD | SHARE | SCHEDULE | MORE

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

select
p.payment_type,
extract(month from o.order_purchase_timestamp) as month_,
count(distinct p.order_id) as num_order
from `TargetSQL.payments` p inner join `TargetSQL.orders` o using(order_id)
group by 1, month_
order by month_, num_order desc
```

Query results — SAV

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAILS | EXECUTION

Row	payment_type	month_	num_order
1	credit_card	1	6093
2	UPI	1	1715
3	voucher	1	337
4	debit_card	1	118
5	credit_card	2	6582
6	UPI	2	1723
7	voucher	2	288
8	debit_card	2	82
9	credit_card	3	7682
10	UPI	3	1942

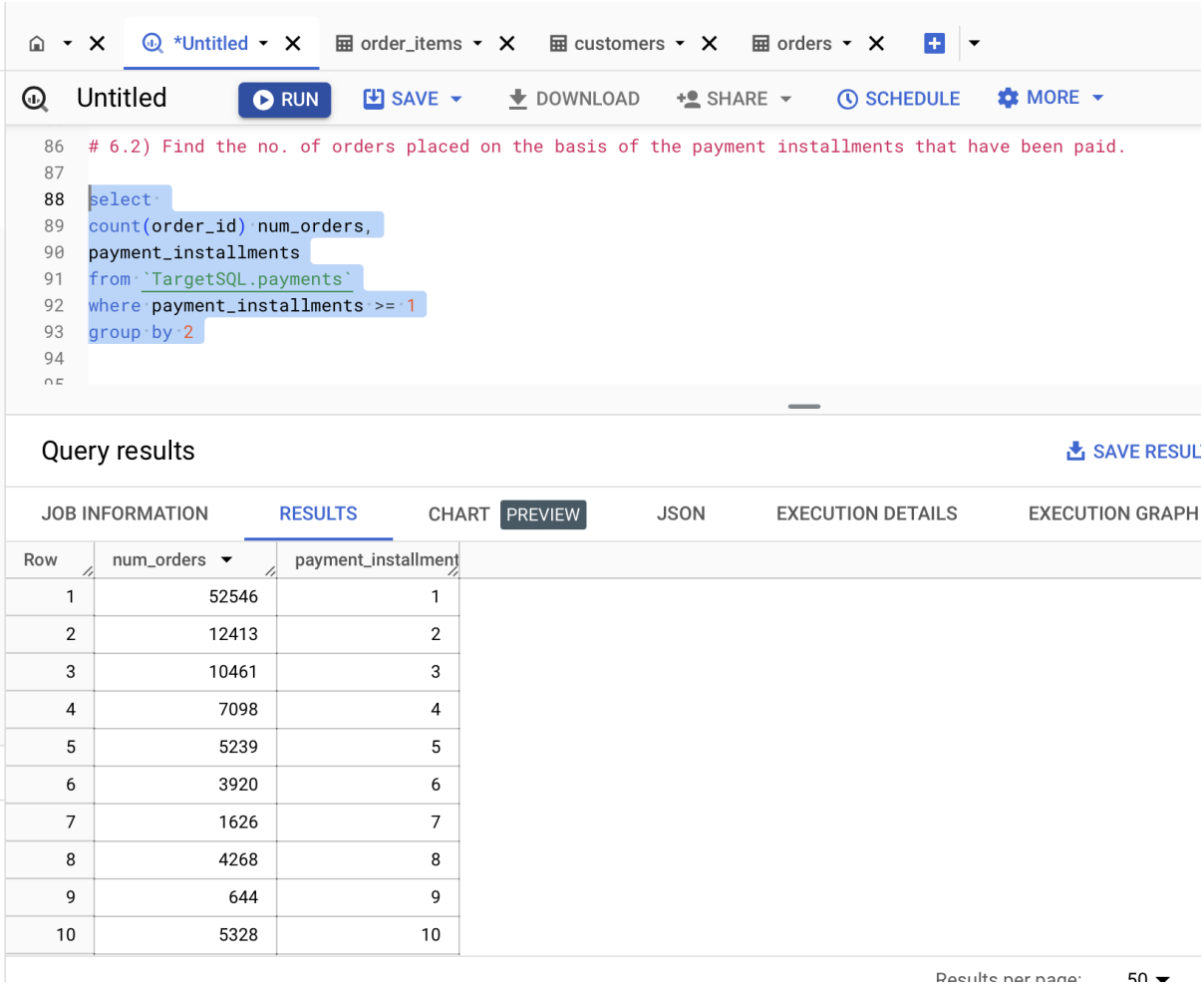
Results per page:

Credit card is the highest used payment type whereas debit card is the least.

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

```
select
count(order_id) num_orders,
payment_installments
from `TargetSQL.payments`
```

```
where payment_installments >= 1
group by 2
```



Majority of people use single payment installment.