Untitled query

RUN SAVE DOWNLOAD SHARE SCHEDULE MORE This query w

```sql
--Data type of all columns in the "customers" table.

select column_name , data_type
from `ecommerce001-435209.Target_analysis.INFORMATION_SCHEMA.COLUMNS`
where table_name = "customers"
```

Press

Query results

SAVE RESULTS

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

Row	column_name	data_type
1	customer_id	STRING
2	customer_unique_id	STRING
3	customer_zip_code_prefix	INT64
4	customer_city	STRING
5	customer_state	STRING

Untitled query

RUN SAVE DOWNLOAD SHARE SCHEDULE MORE This query wi

```sql
-- Get the time range between which the orders were placed.

Select min(order_purchase_timestamp) as strt_time ,
max(order_purchase_timestamp) as end_time
 from `Target_analysis.orders`
```

Pre

Query results

SAVE RESULTS

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

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

order_items | *Untitled query | *Untitled query | *Untitled query | *Untitled query | *Untitled query

Untitled query

RUN | SAVE | DOWNLOAD | SHARE | SCHEDULE | MORE | This query wil

```sql
-- Count the Cities & States of customers who ordered during the given period.
select count(distinct customer_city) as total_city,
count(distinct customer_state) as total_state
 from `Target_analysis.customers`
```

Press

Query results

SAVE RESULTS

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

Row	total_city	total_state
1	4119	27

order_items | *Untitled query | *Untitled query | *Untitled query | *Untitled query | *Untitled query

Untitled query

RUN | SHARE | SCHEDULE | MORE | SAVE | DOWNLOAD

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

select count(distinct order_id) as total_order_ , extract(month from order_purchase_timestamp) as month_,
extract(year from order_purchase_timestamp) as year_
from `Target_analysis.orders`
group by 3 , 2
order by 3 , 2
```

Press Alt

Query results

SAVE RESULTS | EXP

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

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

Results per page: 50 1 – 25 of 25

Untitled query

RUN SAVE DOWNLOAD SHARE SCHEDULE MORE

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

select count(distinct order_id) as total_order_ , extract(month from order_purchase_timestamp) as month_
 from `Target_analysis.orders`
 group by 2
 order by 1 desc
```

Press Alt+F1 for Accessibility Option

Query results

SAVE RESULTS EXPLORE DATA

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

Row	total_order_	month_
1	10843	8
2	10573	5
3	10318	7
4	9893	3
5	9412	6
6	9343	4
7	8508	2
8	8069	1
9	7544	11

Results per page: 50 1 – 12 of 12

```sql
-- 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
With hour_category as
(select order_id,
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 hours_
 from `Target_analysis.orders`)

 select hours_ , count(order_id) as count_of_order
 from hour_category
 group by hours_
 order by count_of_order desc
```

Press Alt+F1 for Accessibility O

Query results

SAVE RESULTS | EXPLORE DATA

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

Row	hours_	count_of_order
1	Afternoon	38135
2	Night	28331

Results per page: 50 | 1 – 4 of 4

tled query | *Untitled query | *Untitled query | *Untitled query | *Untitled query | *Untitled query

Untitled query

RUN | SAVE | DOWNLOAD | SHARE | SCHEDULE | MORE | Query co

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

select count(distinct order_id) as total_order ,customer_state
 , extract(month from order_purchase_timestamp) as month
  from `Target_analysis.customers` as c
join `Target_analysis.orders` as o
on c.customer_id = o.customer_id
group by 2 ,3
order by 1 desc
```

Press Alt+F1 for Accessibili

Query results

SAVE RESULTS | EXPLORE DATA

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

Row	total_order	customer_state	month
1	4982	SP	8
2	4632	SP	5
3	4381	SP	7
4	4104	SP	6
5	4047	SP	3
6	3967	SP	4
7	3357	SP	2

*Untitled query | *Untitled query | *Untitled query | *Untitled query | *Untitled query | *Untitled query

Untitled query

RUN | SAVE | DOWNLOAD | SHARE | SCHEDULE | MORE | Query comp

```
1  -- How are the customers distributed across all the states?
2
3  select customer_state ,count(customer_unique_id) as total_customers
4  from `Target_analysis.customers`
5  group by 1
6  order by 2 desc
```

Press Alt+F1 for Accessibility C

Query results

SAVE RESULTS | EXPLORE DATA

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

Row	customer_state	total_customers
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

Results per page: 50 | 1 – 27 of 27

Job history

REFRESH

*Untitled query | *Untitled query | *Untitled query | *Untitled query | *Untitled query | *Untitled query

Untitled query

RUN | SHARE | SCHEDULE | MORE | SAVE | DOWNLOAD

This query will process 9.35 MB when run

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

with cost_per_month as
(select extract(month from order_purchase_timestamp) as month,
extract(year from order_purchase_timestamp) as year ,
 sum(price * freight_value) as total_price
 from `Target_analysis.orders` as o
join `Target_analysis.order_items` as oi
on o.order_id = oi.order_id
where extract(year from order_purchase_timestamp) in (2017,2018) and
extract(month from order_purchase_timestamp) between 1 and 8
group by 2 ,1
order by 1 , 2 desc)

select month , year ,
round((total_price - (lag(total_price) over(partition by month order by year))) / (lag(total_price) over(partition by month order by year)) * 100 , 2)
as percent_increase
 from cost_per_month
 order by 1,2
```

Press Alt+F1 for Accessibility Optio

Query results

SAVE RESULTS | EXPLORE DATA

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

Row	month	year	percent_increase

*Untitled query | *Untitled query | *Untitled query | *Untitled query | *Untitled query | *Untitled query

Untitled query

RUN | SHARE | SCHEDULE | MORE | SAVE | DOWNLOAD

This query will process 9.35 MB when run.

```
1  -- Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only).
2
```

Press Alt+F1 for Accessibility Option

Query results

SAVE RESULTS | EXPLORE DATA

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

Row	month	year	percent_increase
1	1	2017	null
2	1	2018	852.27
3	2	2017	null
4	2	2018	156.63
5	3	2017	null
6	3	2018	177.5
7	4	2017	null
8	4	2018	180.64
9	5	2017	null
10	5	2018	91.22
11	6	2017	null

Results per page: 50 | 1 – 16 of 16

Job history

REFRESH

Untitled query

RUN | SAVE | DOWNLOAD | SHARE | SCHEDULE | MORE

This query will process 14.56 MB when run

```sql
--Calculate the Total & Average value of order price for each state

select customer_state,
sum(price) as total,
Avg(price) as average
 from `Target_analysis.orders` as o
join `Target_analysis.order_items` as oi
on o.order_id = oi.order_id
join `Target_analysis.customers` as c
on o.customer_id = c.customer_id
group by 1
order by 1
```

Press Alt+F1 for Accessibility Option

Query results

SAVE RESULTS | EXPLORE DATA

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

Row	customer_state	total	average
8	ES	275037.3099999...	121.9137012411...
9	GO	294591.9499999...	126.2717316759...
10	MA	119648.2199999...	145.2041504854...
11	MG	1585308.029999...	120.7485741488...
12	MS	116812.6399999...	142.6283760683...

Results per page: 50 1 – 27 of 27

Job history

REFRESH

Untitled query | *Untitled query | *Untitled query | *Untitled query | *Untitled query | *Untitled query

Untitled query

RUN | SAVE | DOWNLOAD | SHARE | SCHEDULE | MORE

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

select customer_state,
sum(freight_value) as total,
Avg(freight_value) as average
 from `Target_analysis.orders` as o
join `Target_analysis.order_items` as oi
on o.order_id = oi.order_id
join `Target_analysis.customers` as c
on o.customer_id = c.customer_id
group by 1
order by 1
```

Press Alt+F1 for Accessibility Options.

Query results

SAVE RESULTS | EXPLORE DATA

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

Row	customer_state	total	average
1	AC	3686.749999999...	40.07336956521...
2	AL	15914.58999999...	35.84367117117...
3	AM	5478.889999999...	33.20539393939...
4	AP	2788.500000000...	34.00609756097...
5	BA	100156.6799999...	26.36395893656...

Results per page: 50 | 1 – 27 of 27

Job history

REFRESH

Untitled query

RUN SAVE DOWNLOAD SHARE SCHEDULE MORE

This query will process 5.48 MB when run.

```sql
--  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.
select order_id ,
order_delivered_customer_date - order_purchase_timestamp as time_to_deliver ,
order_estimated_delivery_date - order_delivered_customer_date as deff_estimated_delivery
 from `Target_analysis.orders`
order by 2 desc
```

Press Alt+F1 for Accessibility Options

Query results

SAVE RESULTS EXPLORE DATA

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

Row	order_id	time_to_deliver	deff_estimated_delivery
1	ca07593549f1816d26a572e06...	0-0 0 5031:5:12	0-0 0 -4358:36:39
2	1b3190b2dfa9d789e1f14c05b...	0-0 0 5000:26:32	0-0 0 -4535:24:7
3	440d0d17af552815d15a9e41a...	0-0 0 4695:12:59	0-0 0 -3975:12:50
4	2fb597c2f772eca01b1f5c561b...	0-0 0 4676:24:15	0-0 0 -3734:33:17
5	285ab9426d6982034523a855f...	0-0 0 4671:12:24	0-0 0 -3998:0:4
6	0f4519c5f1c541ddec9f21b3bd...	0-0 0 4657:11:24	0-0 0 -3878:38:21

Results per page: 50 1 – 50 of 99441

Job history

REFRESH

*Untitled query | *Untitled query | *Untitled query | *Untitled query | *Untitled query | *Untitled query

Untitled query

RUN | SAVE | DOWNLOAD | SHARE | SCHEDULE | MORE

This query will process 14.56 MB when run.

```sql
--B. Find out the top 5 states with the highest & lowest average freight value.

with average_of_fv as
(select customer_state, avg(freight_value) as average…
join `Target_analysis.order_items` as oi
on o.order_id = oi.order_id
join `Target_analysis.customers` as c
on o.customer_id = c.customer_id
group by 1)

(select * from average_of_fv
order by average
limit 5)
union all
(select * from average_of_fv
order by average desc
limit 5)
order by average
```

Press Alt+F1 for Accessibility Options

Query results

SAVE RESULTS | EXPLORE DATA

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

Row	customer_state	average

Results per page: 50 | 1 – 10 of 10

Untitled query

RUN SAVE DOWNLOAD SHARE SCHEDULE MORE

This query will process 14.56 MB when run.

```
1  --B. Find out the top 5 states with the highest & lowest average freight value.
2
3  with average_of_fv as
```

Press Alt+F1 for Accessibility Options

Query results

SAVE RESULTS EXPLORE DATA

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

Row	customer_state	average
1	SP	15.14727539041...
2	PR	20.53165156794...
3	MG	20.63016680630...
4	RJ	20.96092393168...
5	DF	21.04135494596...
6	PI	39.14797047970...
7	AC	40.07336956521...
8	RO	41.06971223021...
9	PB	42.72380398671...
10	RR	42.98442307692...

Results per page: 50 1 – 10 of 10

Job history

REFRESH

Untitled query

RUN SAVE DOWNLOAD SHARE SCHEDULE MORE

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

with average_of_fv as
(select customer_state, avg(DATE_DIFF(order_delivered_customer_date , order_purchase_timestamp , day)) as average
 from `Target_analysis.orders` as o
join `Target_analysis.order_items` as oi
on o.order_id = oi.order_id
join `Target_analysis.customers` as c
on o.customer_id = c.customer_id
group by 1)

(select * from average_of_fv
order by average
limit 5)
union all
(select * from average_of_fv
order by average desc
limit 5)
order by average
```

Press Alt+F1 for Accessibility Option

Query results

SAVE RESULTS EXPLORE DATA

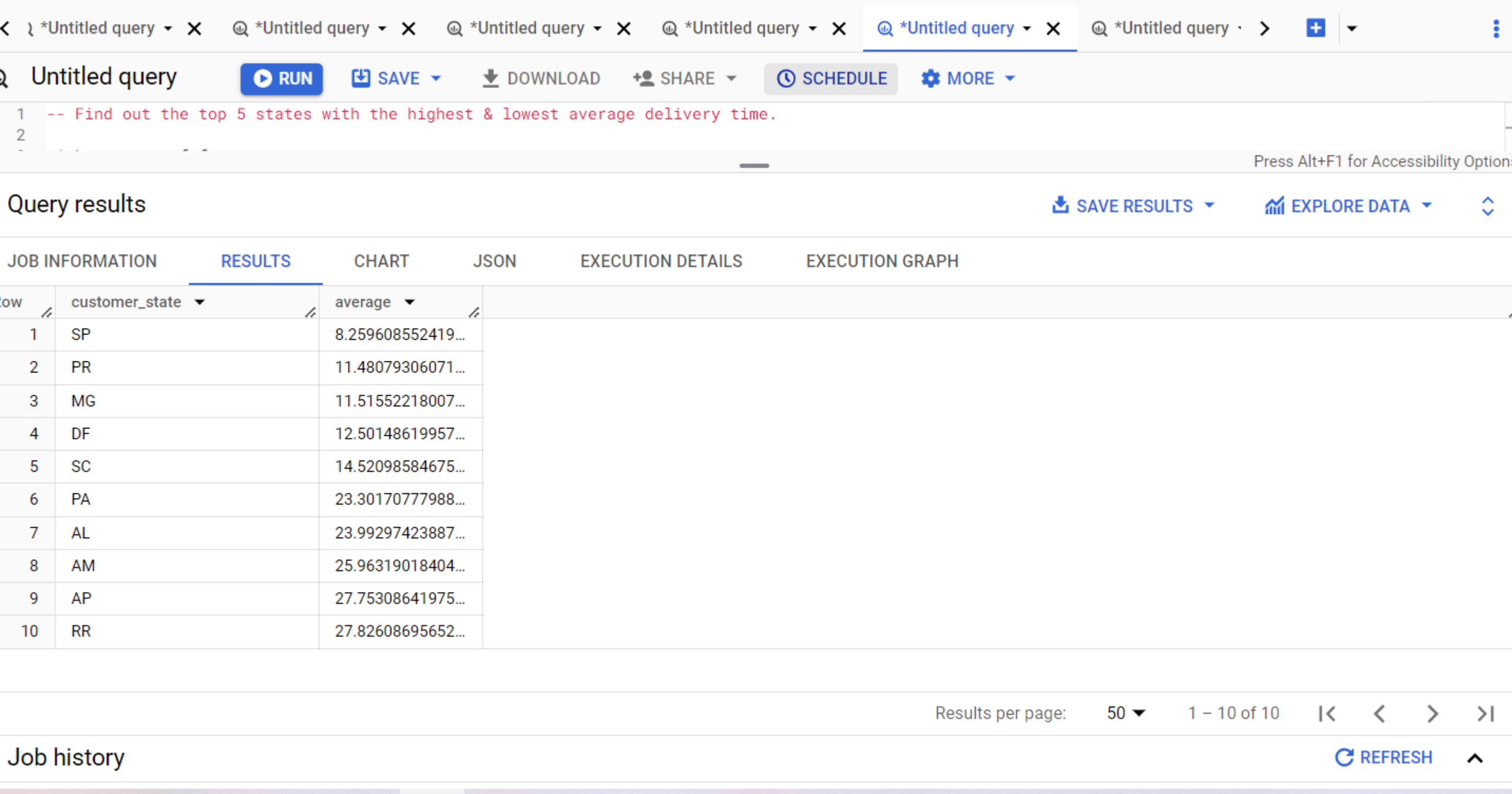
JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	customer_state	average

Results per page: 50 1 – 10 of 10

Job history

REFRESH


*Untitled query
*Untitled query
*Untitled query
*Untitled query
*Untitled query
*Untitled query
Untitled query
RUN
SAVE
DOWNLOAD
SHARE
SCHEDULE
MORE
1 -- Find out the top 5 states with the highest & lowest average delivery time.
2
Press Alt+F1 for Accessibility Option
Query results
SAVE RESULTS
EXPLORE DATA
JOB INFORMATION
RESULTS
CHART
JSON
EXECUTION DETAILS
EXECUTION GRAPH
Row
customer_state
average
1 SP 8.259608552419...
2 PR 11.48079306071...
3 MG 11.51552218007...
4 DF 12.50148619957...
5 SC 14.52098584675...
6 PA 23.30170777988...
7 AL 23.99297423887...
8 AM 25.96319018404...
9 AP 27.75308641975...
10 RR 27.82608695652...
Results per page:
50
1 – 10 of 10
Job history
REFRESH

Untitled query

RUN SAVE DOWNLOAD SHARE SCHEDULE MORE

This query will process 15.2 MB when run.

```sql
-- 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  delivery_time_comparison as
(select customer_state, avg(DATE_DIFF(order_estimated_delivery_date , order_delivered_customer_date , day)) as average
 from `Target_analysis.orders` as o
join `Target_analysis.order_items` as oi
on o.order_id = oi.order_id
join `Target_analysis.customers` as c
on o.customer_id = c.customer_id
group by 1)
select * from delivery_time_comparison
order by average desc
limit 5
```

Press Alt+F1 for Accessibility Options

Query results

SAVE RESULTS EXPLORE DATA

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

Row	customer_state	average
1	AC	20.01098901098...
2	RO	19.08058608058...
3	AM	18.97546012269...
4	AP	17.44444444444...
5	RR	17.43478260869...

Results per page: 50 1 – 5 of 5

Job history

REFRESH

```sql
-- A. Find the month on month no. of orders placed using different payment types.

select count(distinct o.order_id)as total_orders , payment_type,
extract(year from order_purchase_timestamp) as year,
extract(month from order_purchase_timestamp) as month
 from `Target_analysis.orders` as o
join `Target_analysis.payments` as p
on o.order_id = p.order_id
group by 2,3,4
order by 2,3,4
```

Press Alt+F1 for Accessibility Options

Query results

SAVE RESULTS | EXPLORE DATA

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

Row	total_orders	payment_type	year	month
41	5475	credit_card	2018	5
42	4796	credit_card	2018	6
43	4738	credit_card	2018	7
44	4963	credit_card	2018	8
45	2	debit_card	2016	10
46	9	debit_card	2017	1
47	13	debit_card	2017	2
48	31	debit_card	2017	3

Results per page: 50 1 – 50 of 90

Job history

REFRESH

Untitled query

RUN SAVE DOWNLOAD SHARE SCHEDULE MORE Query completed.

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

select count(order_id) , payment_installments
from  `Target_analysis.payments`
where payment_installments != 0
group by 2
order by 1
```

Press Alt+F1 for Accessibility Option

Query results

SAVE RESULTS EXPLORE DATA

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

Row	f0_	payment_installment
1	1	22
2	1	23
3	3	21
4	5	16
5	8	17
6	15	14
7	16	13
8	17	20
9	18	24
10	23	11

Results per page: 50 1 – 23 of 23

Job history

REFRESH