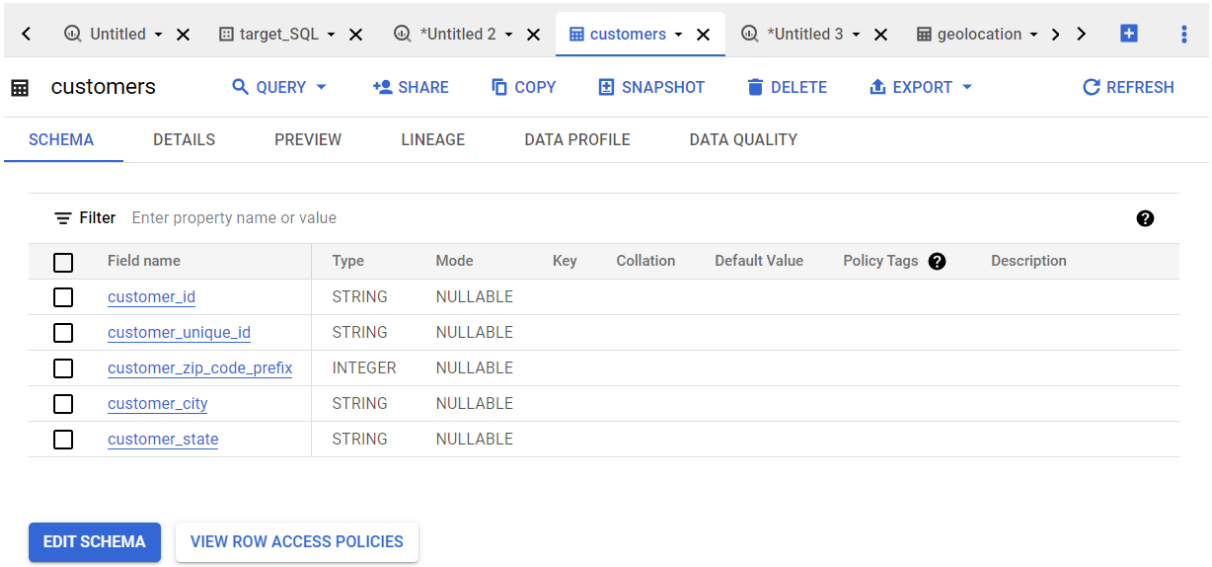
I. Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset.

A. Data type of all columns in the “customers” table.

ANS:



Insights: In the data type more string data types

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

ANS:

```
SELECT min(order_purchase_timestamp) as frist_order,
max(order_purchase_timestamp) as last_order FROM `scalar-dsml-sql-raju.target_SQL.orders`
```

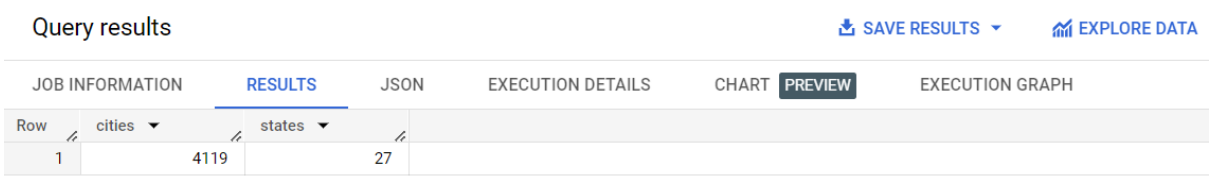
Row	frist_order	last_order
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC

Insights: This market is opened for 2 years from 04/09/2016 and last order placed on 17/09/2018.

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

Ans:

```
select count(distinct c.customer_city) cities ,count(distinct c.customer_state) states from `target_SQL.customers` c join `target_SQL.orders` o on c.customer_id=o.customer_id
```



Insights: the total number of cities 4119 and 27 states.

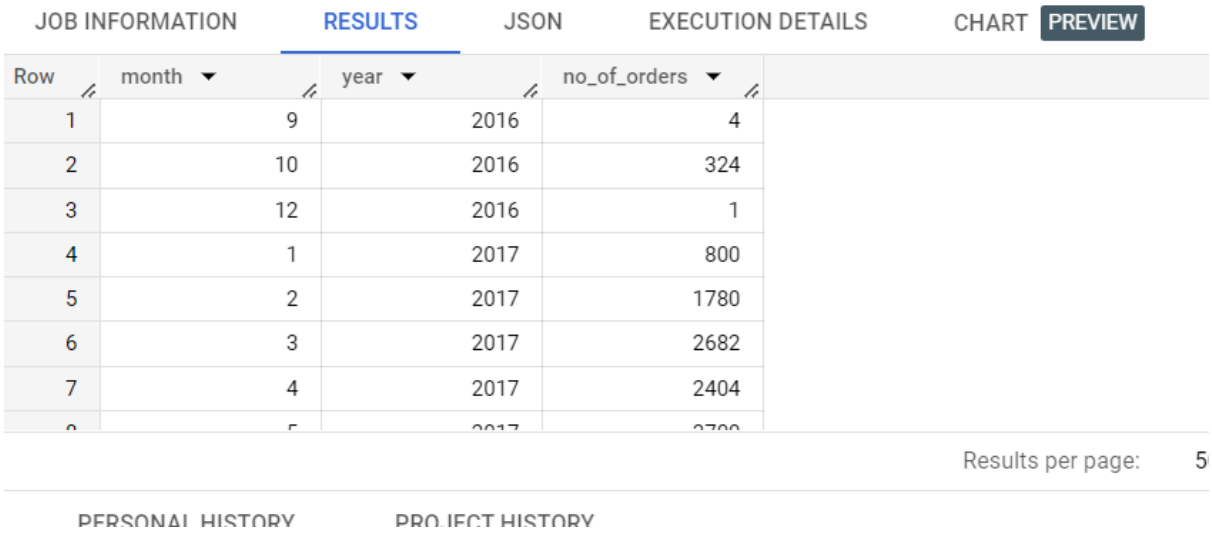
II. In-depth Exploration:

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

ANS:

```
 select extract( month from order_purchase_timestamp) as month, extract(year from
order_purchase_timestamp) as year,

count(*) as no_of_orders from `target_SQL.orders` group by month, year order by
year asc, month asc
```



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

Row	month	year	no_of_orders
1	9	2016	4
2	10	2016	324
3	12	2016	1
4	1	2017	800
5	2	2017	1780
6	3	2017	2682
7	4	2017	2404

Results per page: 5

PERSONAL HISTORY PROJECT HISTORY

Insights: no. of orders in growing trend

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

Ans:
```
select extract(month from order_purchase_timestamp) as month, count(*) as
no_of_orders from `target_SQL.orders`

group by month order by month
```

JOB INFORMATION | RESULTS | JSON | EX

Row	month	no_of_orders
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

PERSONAL HISTORY | PROJECT HISTORY

Insights: 7,8 months are peaked in no.of orders placed

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

ANS:

```
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 '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
as season,
count(*) as no_of_orders from `target_SQL.orders` group by season order by
no_of_orders desc
limit 1
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	season	no_of_orders
1	Afternoon	38135

PERSONAL HISTORY | PROJECT HISTORY

Insight: afternoon is the most placed orders

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

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

Ans:
```
select c.customer_state,extract(month from order_purchase_timestamp) as month,count(*) as no_of_orders from `target_SQL.customers` c join `target_SQL.orders` o on c.customer_id=o.customer_id group by 1,2 order by 1,2
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	month	nooforders
13	AL	1	39
14	AL	2	39
15	AL	3	40
16	AL	4	51
17	AL	5	46
18	AL	6	34
19	AL	7	40
20	AL	8	34
21	AL	9	20

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

Ans:
```
select customer_state ,count(*) as no_of_customers from `target_SQL.customers` group by 1 order by 1,2
```

JOB INFORMATION | RESULTS | JSON | EXECUTION

Row	customer_state	no_of_customers
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

IV. 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)*.

Ans:
```
select ((lag_value-cost_of_orders)/lag_value )* 100 increase_percentage from
(select year,cost_of_orders,lag(cost_of_orders) over (order by year desc)
lag_value, from (select extract ( year from order_purchase_timestamp) as
year,sum(payment_value) as cost_of_orders from `target_SQL.orders` o join
`target_SQL.payments` p on o.order_id=p.order_id where extract ( year from
order_purchase_timestamp) in (2017,2018) and extract ( month from
order_purchase_timestamp) in (1,2,3,4,5,6,7,8) group by 1

) sub)anothersub order by 1 desc limit 1
```

JOB INFORMATION | RESULTS | JSON | EXI

Row	increase_percentage
1	57.80178913446...

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

Ans:
```
select c.customer_state,sum(price) as total ,Avg(price) as Average from
`target_SQL.customers` c join `target_SQL.orders` o on o.customer_id=c.customer_id
join `target_SQL.order_items`as oi on oi.order_id=o.order_id group by 1
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	total	Average
1	MT	156453.5299999...	148.2971848341...
2	MA	119648.2199999...	145.2041504854...
3	AL	80314.81	180.8892117117...
4	SP	5202955.050001...	109.6536291597...
5	MG	1585308.029999...	120.7485741488...
6	PE	262788.0299999...	145.5083222591...
7	RJ	1824092.669999...	125.1178180945...
8	DF	302603.9399999...	125.7705486284...
9	RS	750304.0200000...	120.3374530874...

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

Ans:
```
select c.customer_state,sum(freight_value) as total ,Avg(freight_value) as Average from `target_SQL.customers` c join `target_SQL.orders` o on o.customer_id=c.customer_id join `target_SQL.order_items`as oi on oi.order_id=o.order_id group by 1
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | C

Row	customer_state	total	Average
1	RN	18860.09999999...	35.65236294896...
2	CE	48351.58999999...	32.71420162381...
3	RS	135522.7400000...	21.73580433039...
4	SC	89660.26000000...	21.47036877394...
5	SP	718723.0699999...	15.14727539041...
6	MG	270853.4600000...	20.63016680630...
7	BA	100156.6799999...	26.36395893656...
8	RJ	305589.3100000...	20.96092393168...
9	GO	53114.97999999...	22.76681525932...

V. 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.

Ans:
```
select order_id, ceil(timestamp_diff(order_delivered_customer_date,
order_purchase_timestamp, hour)/24) as time_to_delivered,

ceil(timestamp_diff(order_estimated_delivery_date, order_delivered_customer_date,
hour)/24) as diff_estimated_delivery from
`target_SQL.orders` order by time_to_delivered
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	order_id	time_to_delivered	diff_estimated_delive
1	7a4df5d8cff4090e541401a20a...	null	null
2	35de4050331c6c644cddc86f4...	null	null
3	b5359909123fa03c50bdb0cfe...	null	null
4	dba5062fbda3af4fb6c33b1e04...	null	null
5	90ab3e7d52544ec7bc3363c82...	null	null
6	fa65dad1b0e818e3ccc5cb0e3...	null	null
7	1df2775799eecdf9dd8502425...	null	null
8	6190a94657e1012983a274b8...	null	null
9	58ce513a55c740a3a81e8c8b7...	null	null

Re

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

```
select c.customer_state,avg(freight_value) as highest_average from
`target_SQL.customers` c join `target_SQL.orders` o on o.customer_id=c.customer_id
join `target_SQL.order_items`as oi
on oi.order_id=o.order_id group by 1 order by 2 desc limit 5
```

JOB INFORMATION | RESULTS | JSON | EXECU

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

```
 select c.customer_state,avg(freight_value) as lowest_average from
`target_SQL.customers` c join `target_SQL.orders` o on o.customer_id=c.customer_id
join `target_SQL.order_items`as oi

on oi.order_id=o.order_id group by 1 order by 2 asc  limit 5
```

JOB INFORMATION | RESULTS | JSON | EXECUTION

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

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

ANS:

```
select
c.customer_state,avg(ceil(timestamp_diff(order_estimated_delivery_date,order_delive
red_customer_date ,hour)/24)) as highest_average from `target_SQL.customers` c join
`target_SQL.orders` o on o.customer_id=c.customer_id join
`target_SQL.order_items`as oi on oi.order_id=o.order_id group by 1 order by 2 desc
limit 5
```

JOB INFORMATION | RESULTS | JSON | EXECUT

Row	customer_state	highest_average
1	AC	20.96703296703...
2	RO	19.97802197802...
3	AM	19.85276073619...
4	AP	18.39506172839...
5	RR	18.17391304347...

```
select
c.customer_state,avg(ceil(timestamp_diff(order_estimated_delivery_date,order_delive
red_customer_date ,hour)/24)) as lowest_average from `target_SQL.customers` c join
`target_SQL.orders` o on o.customer_id=c.customer_id join
`target_SQL.order_items`as oi on oi.order_id=o.order_id group by 1 order by 2 asc
limit 5
```

JOB INFORMATION | RESULTS | JSON | EXECUT

Row	customer_state	lowest_average
1	AL	8.725995316159...
2	MA	9.864999999999...
3	SE	9.970666666666...
4	ES	10.59730337078...
5	BA	10.94243822970...

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

ANS:
```
select customer_state, avg(timestamp_diff(order_estimated_delivery_date, order_delivered_customer_date, hour)) as A_V from `target_SQL.orders` o join `target_SQL.customers` c on c.customer_id = o.customer_id group by 1
```

Row	customer_state	A_V
1	RJ	265.0965757305...
2	RS	316.5858907185...
3	SP	248.7211013705...
4	DF	271.7336538461...
5	PR	302.4387568555...
6	MT	328.0282167042...
7	MA	213.0167364016...
8	AL	192.5214105793...
9	MG	300.4570673712...

VI. Analysis based on the payments:

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

ANS:

```
select payment_type, extract(month from order_purchase_timestamp ) as month, count(*) as no_of_orders from `target_SQL.payments` p join `target_SQL.orders` o on o.order_id=p.order_id group by 1,2 order by 2,1
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	payment_type	month	no_of_orders
1	UPI	1	1715
2	credit_card	1	6103
3	debit_card	1	118
4	voucher	1	477
5	UPI	2	1723
6	credit_card	2	6609
7	debit_card	2	82
8	voucher	2	424
9	UPI	3	1942

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

ANS:

```
select count(distinct(order_id)) as installments from `target_SQL.payments` where payment_installments>0
```

JOB INFORMATION | RESULTS | JSON | EXEC

Row	installments
1	99438