1. Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset
    1. Data type of columns in a table

       Ans: customers table -

Column Name	Data Type
customer_id	String
customer_unique_id	String
customer_zip_code_prefix	Integer
customer_city	String
customer_state	String

geolocation table –

Column Name	Data Type
geolocation_zip_code_prefix	Integer
geolocation_lat	Float
geolocation_lng	Float
geolocation_city	String
geolocation_state	String

Order_items table –

Column Name	Data Type
order_id	String
order_item_id	Integer
product_id	String
seller_id	String
shipping_limit_date	Timestamp
price	Float
freight_value	Float

order_reviews table –

Column Name	Data Type
review_id	String
order_id	String
review_score	Integer
review_comment_title	String
review_creation_date	Timestamp
review_answer_timestamp	Timestamp

orders table –

Column Name	Data Type
order_id	String

customer_id	String
order_status	String
order_purchase_timestamp	Timestamp
order_approved_at	Timestamp
order_delivered_carrier_date	Timestamp
order_delivered_customer_date	Timestamp
order_estimated_delivery_date	Timestamp

payments table –

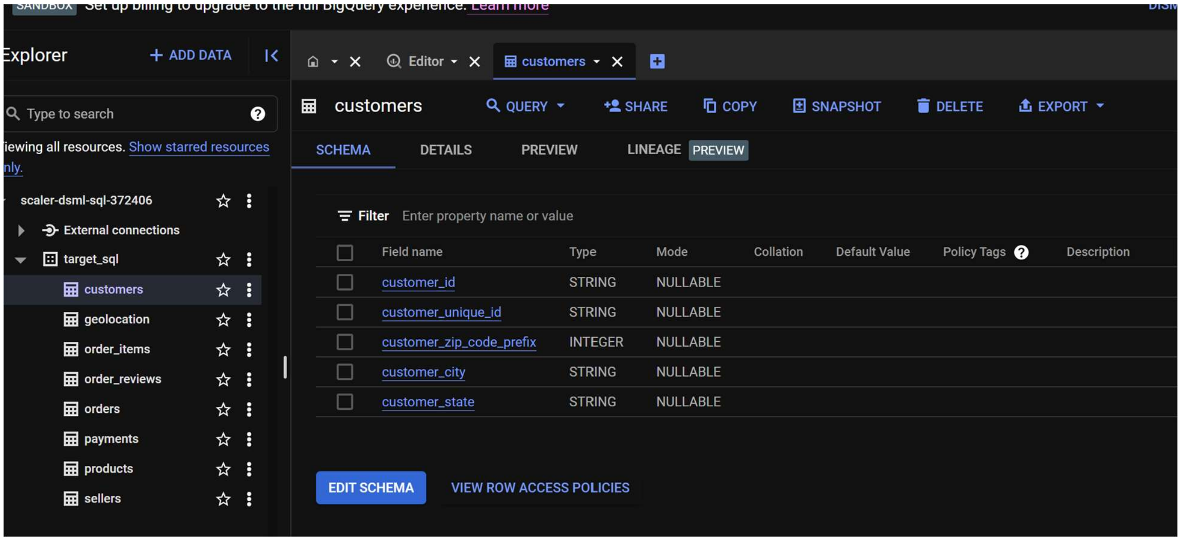
Column Name	Data Type
order_id	String
payment_sequential	Integer
payment_type	String
payment_installments	Integer
payment_value	Float

products table –

Column Name	Data Type
product_id	String
product category	String
product_name_length	Integer
product_description_length	Integer
product_photos_qty	Integer
product_weight_g	Integer
product_length_cm	Integer
product_height_cm	Integer
product_width_cm	Integer

sellers table –

Column Name	Data Type
seller_id	String
seller_zip_code_prefix	Integer
seller_city	String
seller_state	String



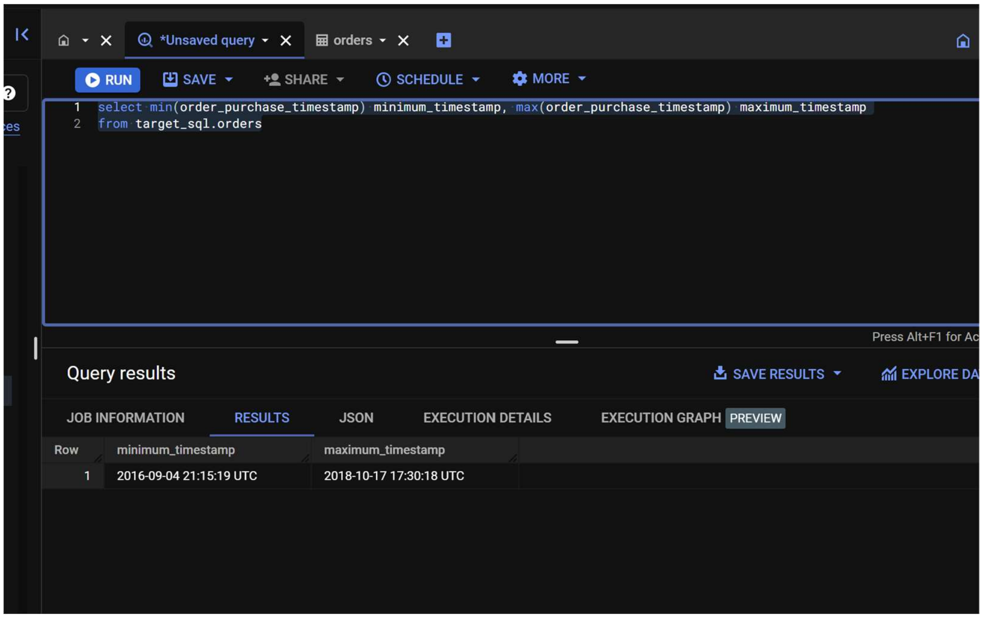
□	Field name	Type	Mode	Collation	Default Value	Policy Tags	Description
□	customer_id	STRING	NULLABLE				
□	customer_unique_id	STRING	NULLABLE				
□	customer_zip_code_prefix	INTEGER	NULLABLE				
□	customer_city	STRING	NULLABLE				
□	customer_state	STRING	NULLABLE				

2. Time period for which the data is given

   Ans: The query –

   select min(order_purchase_timestamp) minimum_timestamp, max(order_purchase_timestamp) maximum_timestamp

   from target_sql.orders



```sql
select min(order_purchase_timestamp) minimum_timestamp, max(order_purchase_timestamp) maximum_timestamp
from target_sql.orders
```

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

3. Cities and States of customers ordered during the given period

   Ans. The query used –

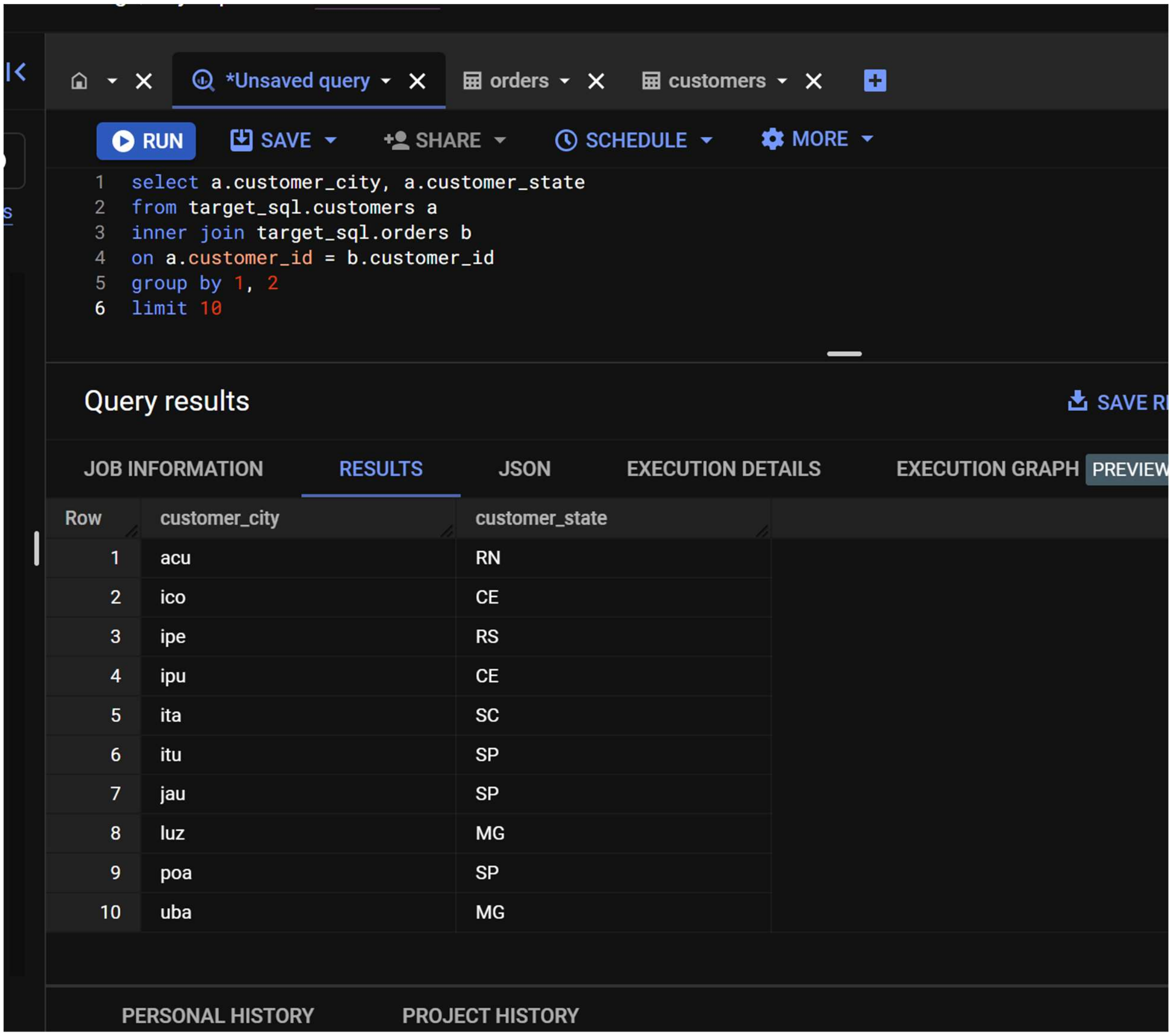
select a.customer_city, a.customer_state

from target_sql.customers a

inner join target_sql.orders b

on a.customer_id = b.customer_id

group by 1, 2



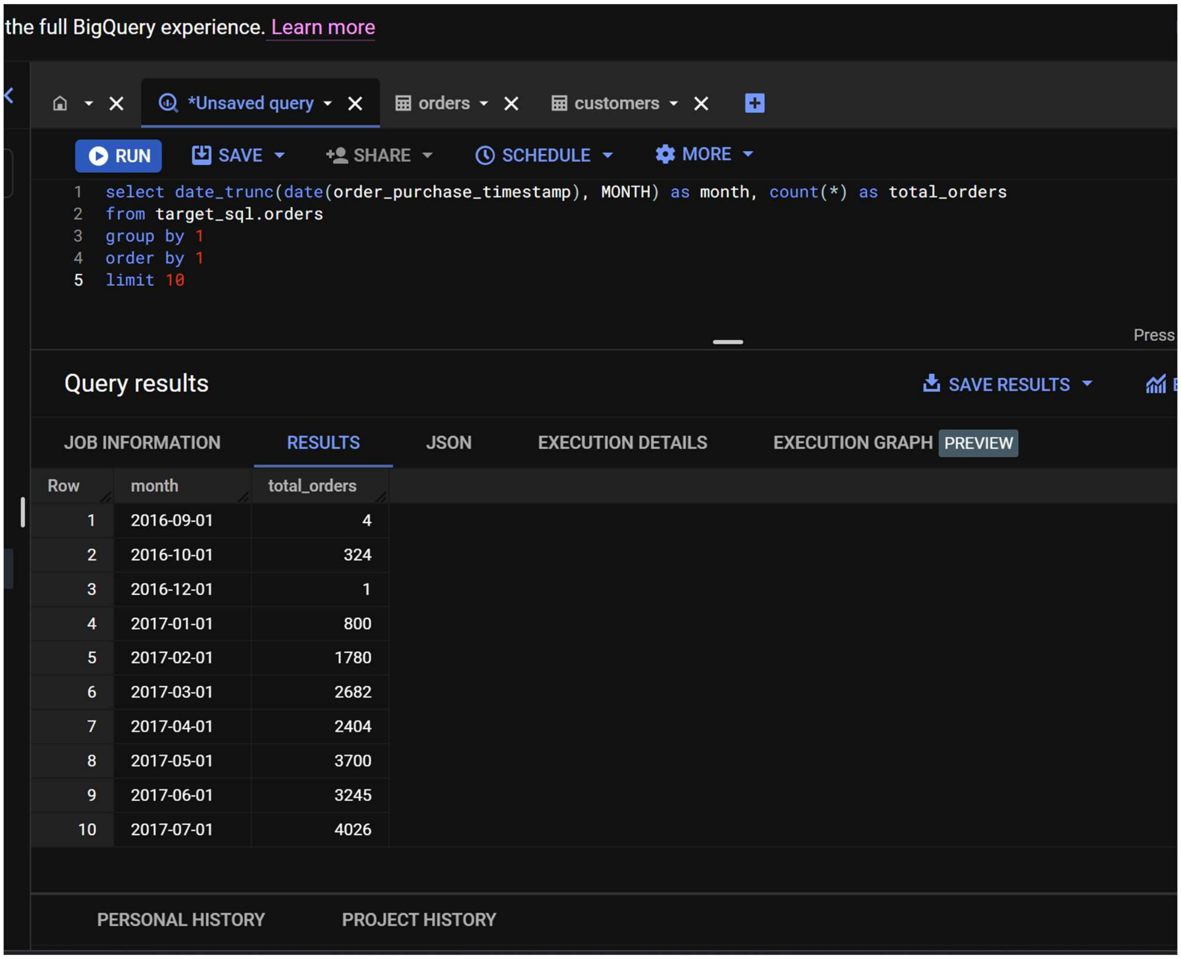
```
select a.customer_city, a.customer_state
from target_sql.customers a
inner join target_sql.orders b
on a.customer_id = b.customer_id
group by 1, 2
limit 10
```

Row	customer_city	customer_state
1	acu	RN
2	ico	CE
3	ipe	RS
4	ipu	CE
5	ita	SC
6	itu	SP
7	jau	SP
8	luz	MG
9	poa	SP
10	uba	MG

2. In-depth Exploration:
    1. Is there a growing trend on e-commerce in Brazil? How can we describe a complete scenario? Can we see some seasonality with peaks at specific months?

       Ans. The query –

```
select date_trunc(date(order_purchase_timestamp), MONTH) as month,
count(*) as total_orders

from target_sql.orders

group by 1

order by 1
```

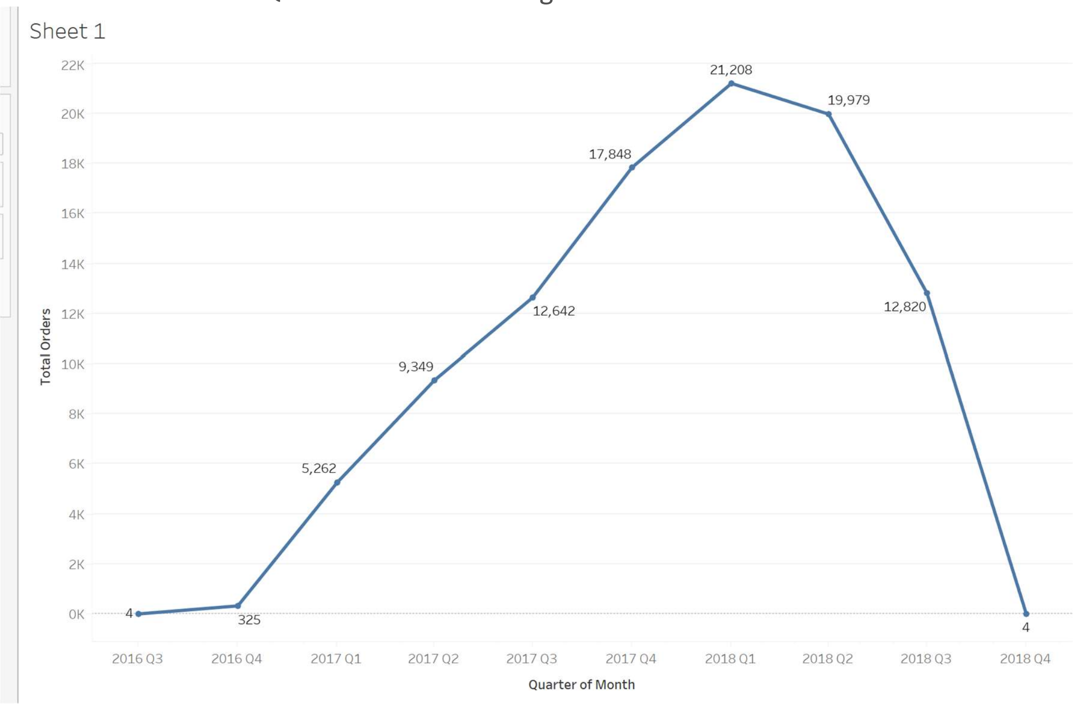


```
select date_trunc(date(order_purchase_timestamp), MONTH) as month, count(*) as total_orders
from target_sql.orders
group by 1
order by 1
limit 10
```

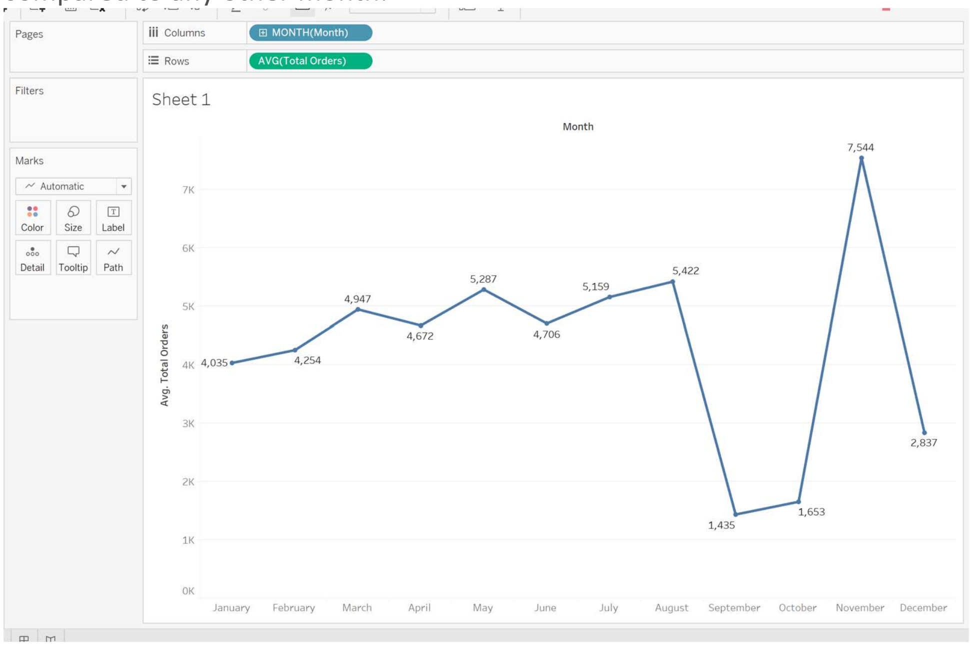
Query results

Row	month	total_orders
1	2016-09-01	4
2	2016-10-01	324
3	2016-12-01	1
4	2017-01-01	800
5	2017-02-01	1780
6	2017-03-01	2682
7	2017-04-01	2404
8	2017-05-01	3700
9	2017-06-01	3245
10	2017-07-01	4026

Yes. There is a trend in the growth of e-commerce in Brazil. If we were to look at the quarter wise sales, we will notice that up till 2018’Q1 there is increase in sales and after 2018'Q1 sales are decreasing.



Regarding seasonality November is seen to have sold most average sales compared to any other month.


Columns MONTH(Month)

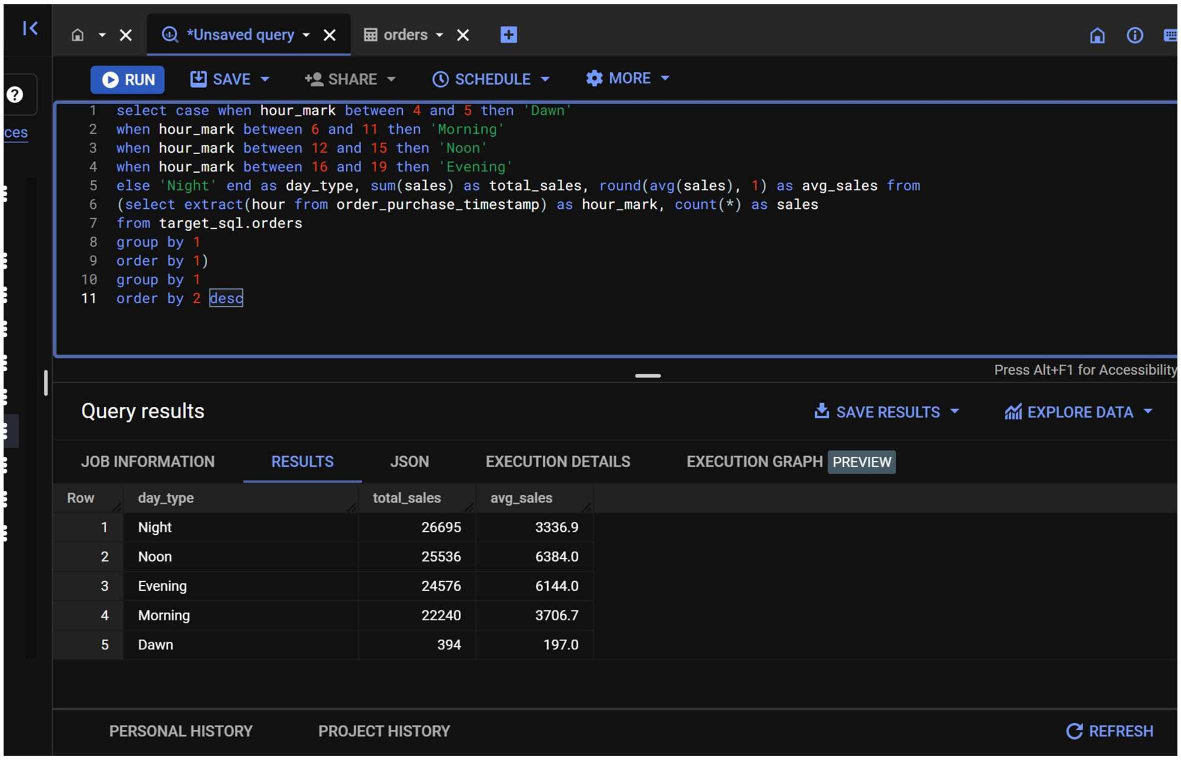
Rows AVG(Total Orders)

2. What time do Brazilian customers tend to buy (Dawn, Morning, Afternoon or Night)?

Ans. Assuming dawn is from 4 am (inclusive) to 6 am (exclusive), morning is from 6 am (inclusive) to 12 pm (exclusive), noon is from 12 pm (inclusive) to 4 pm (exclusive) , evening is from 4 pm (inclusive) to 8 pm (exclusive) and night is from 8 pm (inclusive) onwards till dawn.

The query –

```
select case when hour_mark between 4 and 5 then 'Dawn'

when hour_mark between 6 and 11 then 'Morning'

when hour_mark between 12 and 15 then 'Noon'

when hour_mark between 16 and 19 then 'Evening'

else 'Night' end as day_type, sum(sales) as total_sales, round(avg(sales), 1) as avg_sales from

(select extract(hour from order_purchase_timestamp) as hour_mark, count(*) as sales

from target_sql.orders

group by 1

order by 1)

group by 1

order by 2 desc
```



```
select case when hour_mark between 4 and 5 then 'Dawn'
when hour_mark between 6 and 11 then 'Morning'
when hour_mark between 12 and 15 then 'Noon'
when hour_mark between 16 and 19 then 'Evening'
else 'Night' end as day_type, sum(sales) as total_sales, round(avg(sales), 1) as avg_sales from
(select extract(hour from order_purchase_timestamp) as hour_mark, count(*) as sales
from target_sql.orders
group by 1
order by 1)
group by 1
order by 2 desc
```

Row	day_type	total_sales	avg_sales
1	Night	26695	3336.9
2	Noon	25536	6384.0
3	Evening	24576	6144.0
4	Morning	22240	3706.7
5	Dawn	394	197.0

From the result it is evident that night has the most sales but if we compare average (sales per hour) then noon seems to have the most.

3. Evolution of E-commerce orders in the Brazil region:
    1. Get month on month orders by states

       Ans. The query –

       select date_trunc(date(a.order_purchase_timestamp), MONTH) as month, b.customer_state, count(*) as total_orders

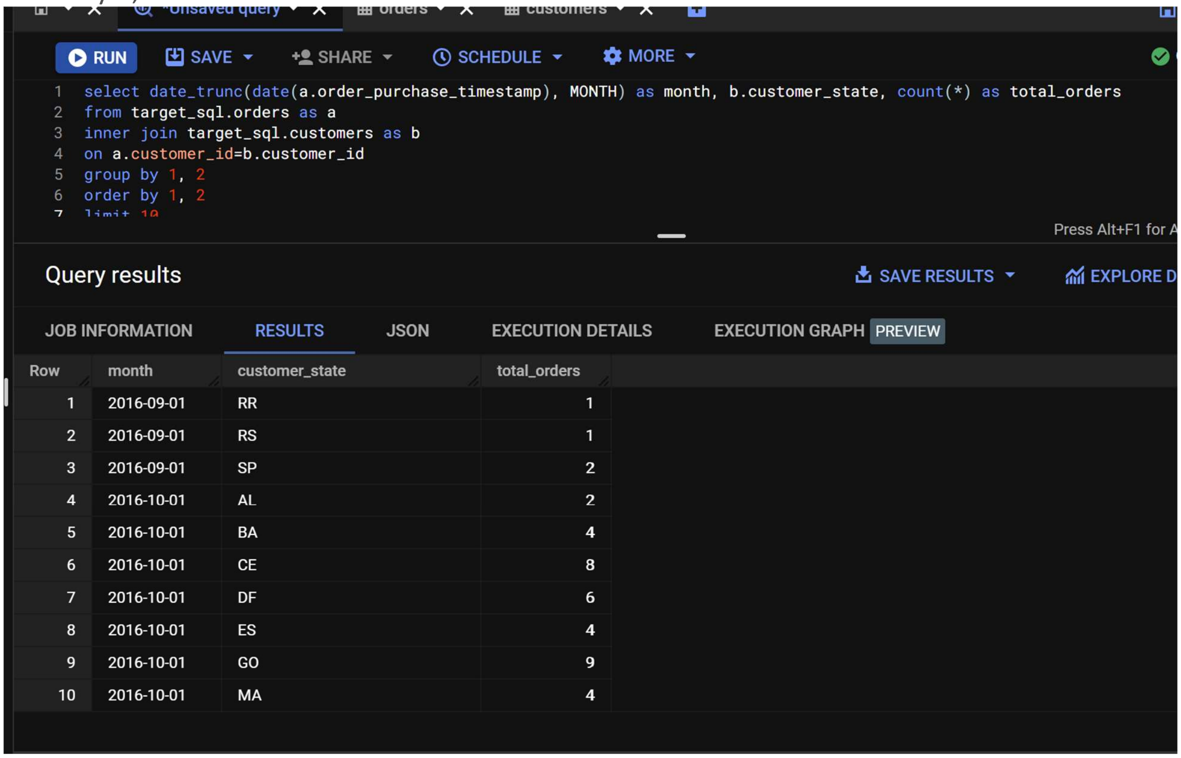
       from target_sql.orders as a

       inner join target_sql.customers as b

       on a.customer_id=b.customer_id

       group by 1, 2

order by 1, 2

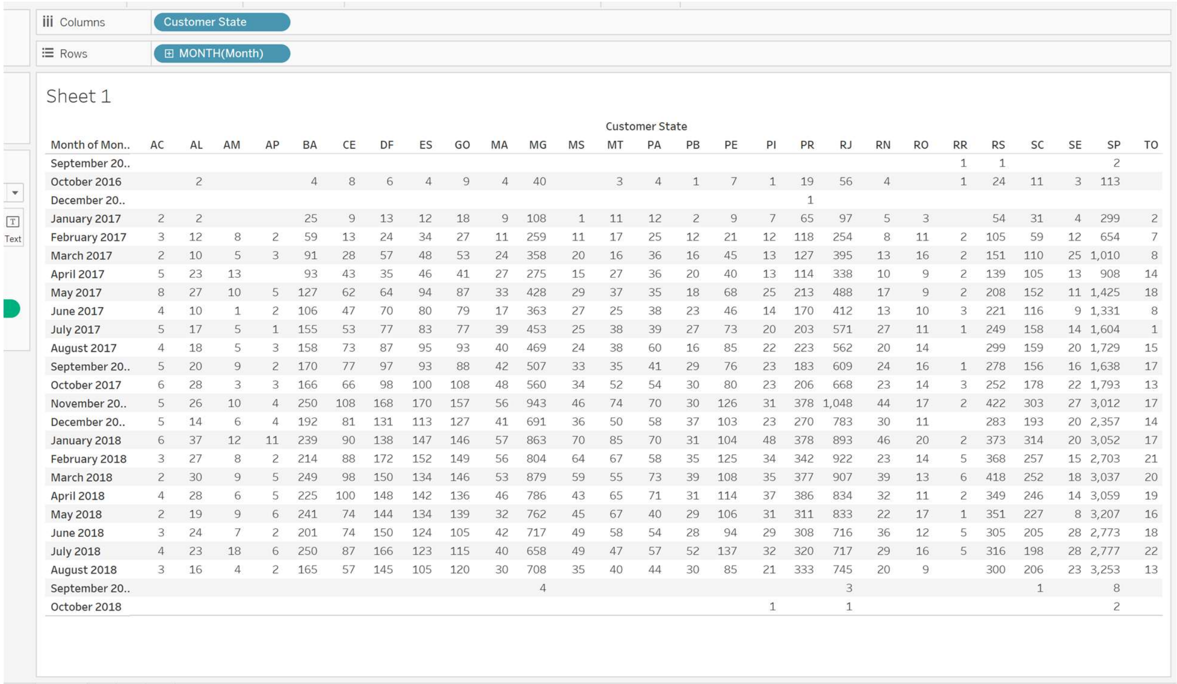


```sql
select date_trunc(date(a.order_purchase_timestamp), MONTH) as month, b.customer_state, count(*) as total_orders
from target_sql.orders as a
inner join target_sql.customers as b
on a.customer_id=b.customer_id
group by 1, 2
order by 1, 2
limit 10
```

Query results

Row	month	customer_state	total_orders
1	2016-09-01	RR	1
2	2016-09-01	RS	1
3	2016-09-01	SP	2
4	2016-10-01	AL	2
5	2016-10-01	BA	4
6	2016-10-01	CE	8
7	2016-10-01	DF	6
8	2016-10-01	ES	4
9	2016-10-01	GO	9
10	2016-10-01	MA	4

The result from SQL in Tableau -



Sheet 1

	Customer State																										
Month of Mon..	AC	AL	AM	AP	BA	CE	DF	ES	GO	MA	MG	MS	MT	PA	PB	PE	PI	PR	RJ	RN	RO	RR	RS	SC	SE	SP	TO
September 20..																						1	1			2	
October 2016		2			4	8	6	4	9	4	40		3	4	1	7	1	19	56	4		1	24	11	3	113	
December 20..																		1									
January 2017	2	2			25	9	13	12	18	9	108	1	11	12	2	9	7	65	97	5	3		54	31	4	299	2
February 2017	3	12	8	2	59	13	24	34	27	11	259	11	17	25	12	21	12	118	254	8	11	2	105	59	12	654	7
March 2017	2	10	5	3	91	28	57	48	53	24	358	20	16	36	16	45	13	127	395	13	16	2	151	110	25	1,010	8
April 2017	5	23	13		93	43	35	46	41	27	275	15	27	36	20	40	13	114	338	10	9	2	139	105	13	908	14
May 2017	8	27	10	5	127	62	64	94	87	33	428	29	37	35	18	68	25	213	488	17	9	2	208	152	11	1,425	18
June 2017	4	10	1	2	106	47	70	80	79	17	363	27	25	38	23	46	14	170	412	13	10	3	221	116	9	1,331	8
July 2017	5	17	5	1	155	53	77	83	77	39	453	25	38	39	27	73	20	203	571	27	11	1	249	158	14	1,604	1
August 2017	4	18	5	3	158	73	87	95	93	40	469	24	38	60	16	85	22	223	562	20	14		299	159	20	1,729	15
September 20..	5	20	9	2	170	77	97	93	88	42	507	33	35	41	29	76	23	183	609	24	16	1	278	156	16	1,638	17
October 2017	6	28	3	3	166	66	98	100	108	48	560	34	52	54	30	80	23	206	668	23	14	3	252	178	22	1,793	13
November 20..	5	26	10	4	250	108	168	170	157	56	943	46	74	70	30	126	31	378	1,048	44	17	2	422	303	27	3,012	17
December 20..	5	14	6	4	192	81	131	113	127	41	691	36	50	58	37	103	23	270	783	30	11		283	193	20	2,357	14
January 2018	6	37	12	11	239	90	138	147	146	57	863	70	85	70	31	104	48	378	893	46	20	2	373	314	20	3,052	17
February 2018	3	27	8	2	214	88	172	152	149	56	804	64	67	58	35	125	34	342	922	23	14	5	368	257	15	2,703	21
March 2018	2	30	9	5	249	98	150	134	146	53	879	59	55	73	39	108	35	377	907	39	13	6	418	252	18	3,037	20
April 2018	4	28	6	5	225	100	148	142	136	46	786	43	65	71	31	114	37	386	834	32	11	2	349	246	14	3,059	19
May 2018	2	19	9	6	241	74	144	134	139	32	762	45	67	40	29	106	31	311	833	22	17	1	351	227	8	3,207	16
June 2018	3	24	7	2	201	74	150	124	105	42	717	49	58	54	28	94	29	308	716	36	12	5	305	205	28	2,773	18
July 2018	4	23	18	6	250	87	166	123	115	40	658	49	47	57	52	137	32	320	717	29	16	5	316	198	28	2,777	22
August 2018	3	16	4	2	165	57	145	105	120	30	708	35	40	44	30	85	21	333	745	20	9		300	206	23	3,253	13
September 20..											4								3					1		8	
October 2018																	1		1							2	

2. Distribution of customers across the states in Brazil

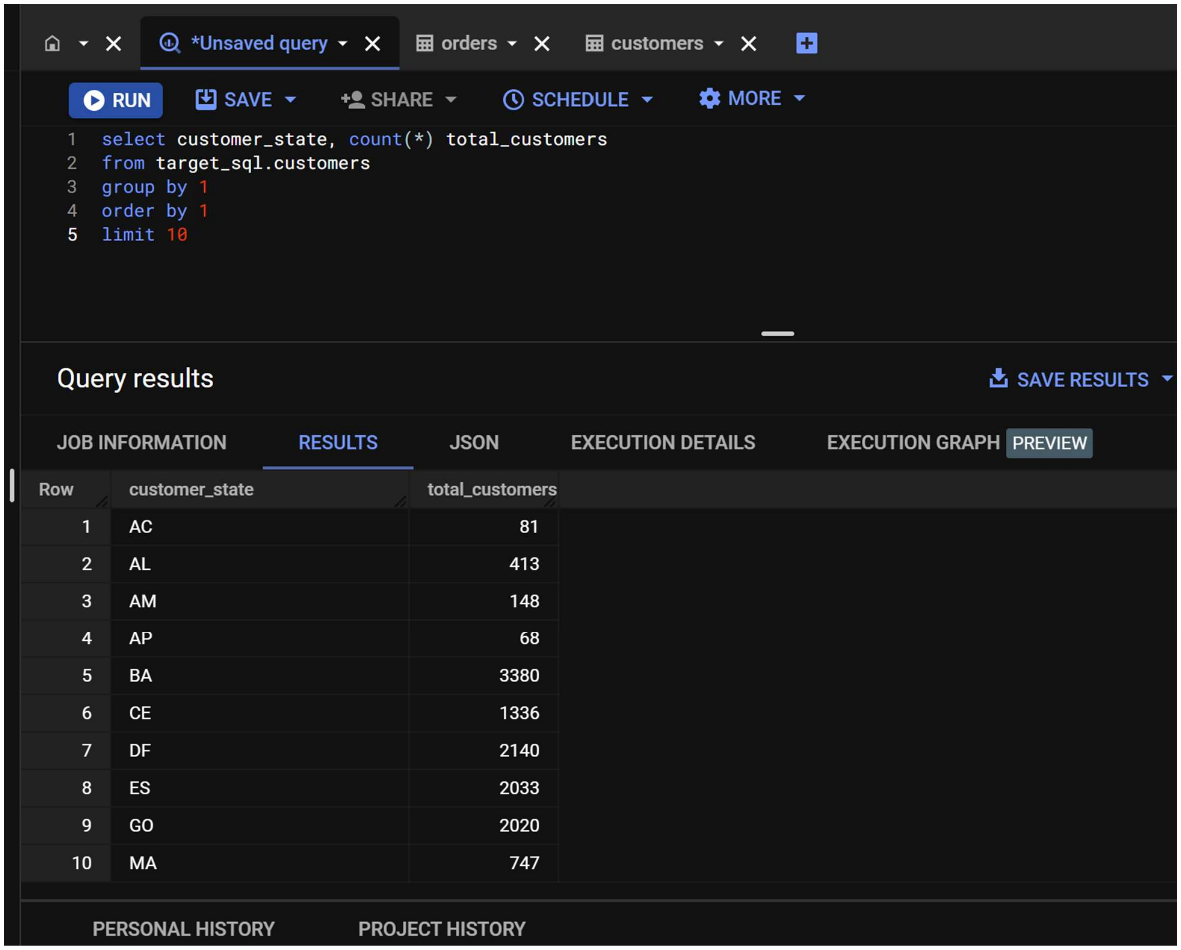
Ans. The query –

select customer_state, count(*) total_customers

from target_sql.customers

group by 1

order by 1



```
select customer_state, count(*) total_customers
from target_sql.customers
group by 1
order by 1
limit 10
```

Query results

Row	customer_state	total_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
10	MA	747

4. Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight and others.
    1. Get % increase in cost of orders from 2017 to 2018 (include months between Jan to Aug only) - You can use “payment_value” column in payments table

       Ans. The query –

       select round((next_payment_value-payment_value)/payment_value*100,2) from (select *, lead(payment_value) over (order by year_value) as next_payment_value from

(select extract(year from order_purchase_timestamp) as year_value, sum(payment_value) as payment_value

from target_sql.orders a

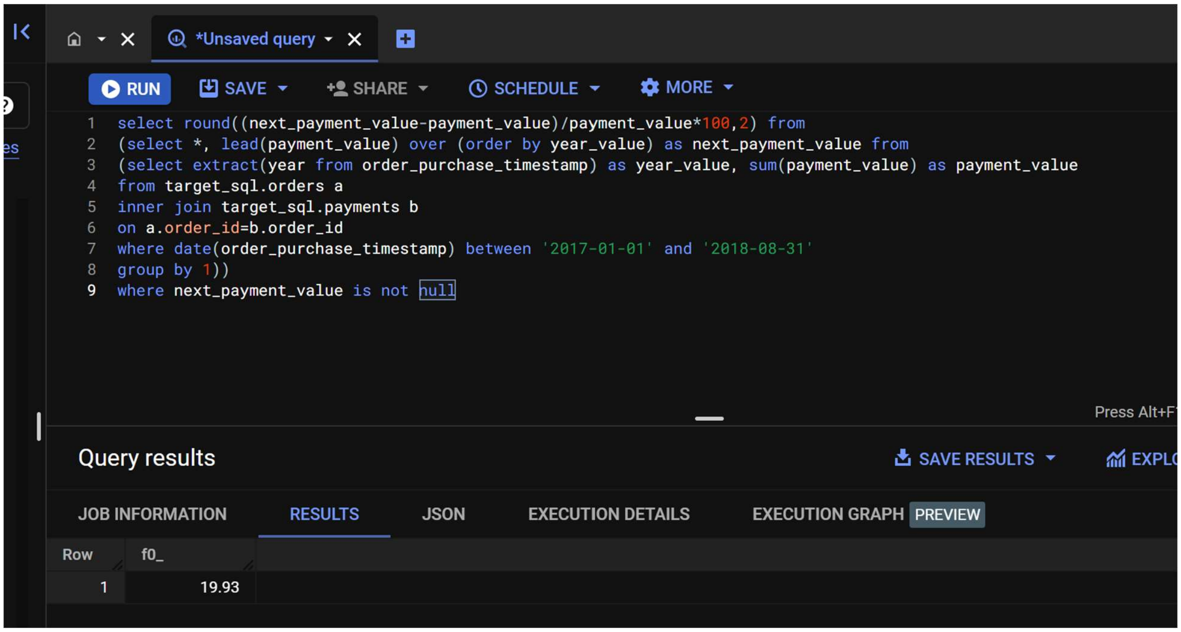
inner join target_sql.payments b

on a.order_id=b.order_id

where date(order_purchase_timestamp) between '2017-01-01' and '2018-08-31'

group by 1))

where next_payment_value is not null



```
select round((next_payment_value-payment_value)/payment_value*100,2) from
(select *, lead(payment_value) over (order by year_value) as next_payment_value from
(select extract(year from order_purchase_timestamp) as year_value, sum(payment_value) as payment_value
from target_sql.orders a
inner join target_sql.payments b
on a.order_id=b.order_id
where date(order_purchase_timestamp) between '2017-01-01' and '2018-08-31'
group by 1))
where next_payment_value is not null
```

Row	f0_
1	19.93

2. Mean & Sum of price and freight value by customer state

   Ans. The query –

   select customer_state, round(sum(price), 2) as total_price, round(avg(price), 2) as avg_price, round(sum(freight_value), 2) as total_freight_value, round(avg(freight_value), 2) as avg_freight_value

   from target_sql.order_items as a

   inner join target_sql.orders as b

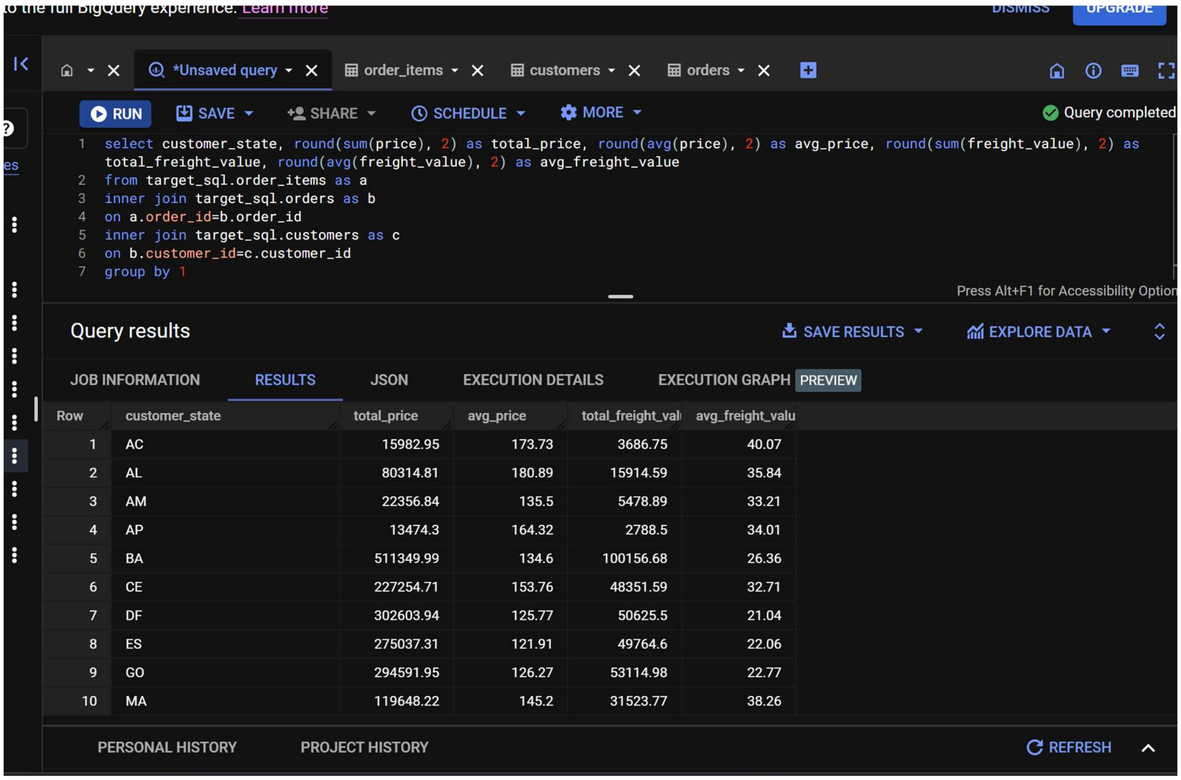
on a.order_id=b.order_id

inner join target_sql.customers as c

on b.customer_id=c.customer_id

group by 1

order by 1



```sql
select customer_state, round(sum(price), 2) as total_price, round(avg(price), 2) as avg_price, round(sum(freight_value), 2) as
total_freight_value, round(avg(freight_value), 2) as avg_freight_value
from target_sql.order_items as a
inner join target_sql.orders as b
on a.order_id=b.order_id
inner join target_sql.customers as c
on b.customer_id=c.customer_id
group by 1
```

Query results

Row	customer_state	total_price	avg_price	total_freight_val	avg_freight_valu
1	AC	15982.95	173.73	3686.75	40.07
2	AL	80314.81	180.89	15914.59	35.84
3	AM	22356.84	135.5	5478.89	33.21
4	AP	13474.3	164.32	2788.5	34.01
5	BA	511349.99	134.6	100156.68	26.36
6	CE	227254.71	153.76	48351.59	32.71
7	DF	302603.94	125.77	50625.5	21.04
8	ES	275037.31	121.91	49764.6	22.06
9	GO	294591.95	126.27	53114.98	22.77
10	MA	119648.22	145.2	31523.77	38.26

5. Analysis on sales, freight and delivery time

    1. Calculate days between purchasing, delivering and estimated delivery

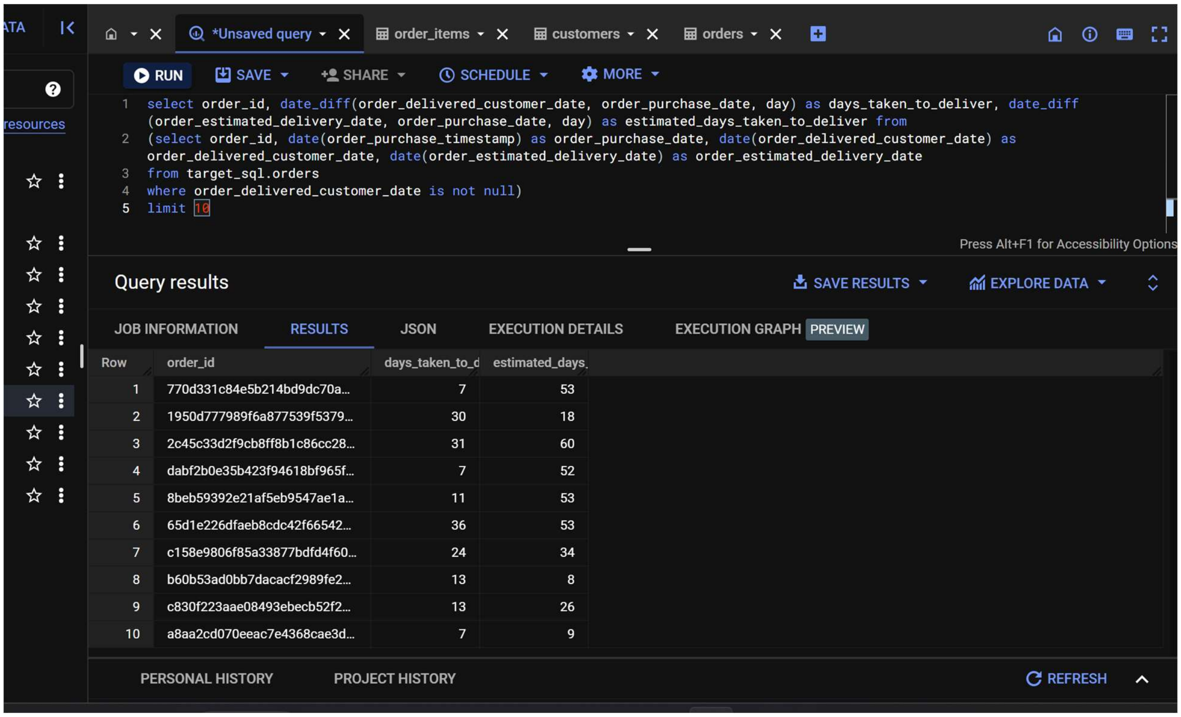
       Ans. Assumption – Only taken orders which were delivered to the customer.

       The query –

       select order_id, date_diff(order_delivered_customer_date, order_purchase_date, day) as days_taken_to_deliver, date_diff(order_estimated_delivery_date, order_purchase_date, day) as estimated_days_taken_to_deliver from

       (select order_id, date(order_purchase_timestamp) as order_purchase_date, date(order_delivered_customer_date) as order_delivered_customer_date, date(order_estimated_delivery_date) as order_estimated_delivery_date from target_sql.orders

where order_delivered_customer_date is not null)



```
select order_id, date_diff(order_delivered_customer_date, order_purchase_date, day) as days_taken_to_deliver, date_diff
(order_estimated_delivery_date, order_purchase_date, day) as estimated_days_taken_to_deliver from
(select order_id, date(order_purchase_timestamp) as order_purchase_date, date(order_delivered_customer_date) as
order_delivered_customer_date, date(order_estimated_delivery_date) as order_estimated_delivery_date
from target_sql.orders
where order_delivered_customer_date is not null)
limit 10
```

Query results

Row	order_id	days_taken_to_d	estimated_days_
1	770d331c84e5b214bd9dc70a...	7	53
2	1950d777989f6a877539f5379...	30	18
3	2c45c33d2f9cb8ff8b1c86cc28...	31	60
4	dabf2b0e35b423f94618bf965f...	7	52
5	8beb59392e21af5eb9547ae1a...	11	53
6	65d1e226dfaeb8cdc42f66542...	36	53
7	c158e9806f85a33877bdfd4f60...	24	34
8	b60b53ad0bb7dacacf2989fe2...	13	8
9	c830f223aae08493ebecb52f2...	13	26
10	a8aa2cd070eeac7e4368cae3d...	7	9

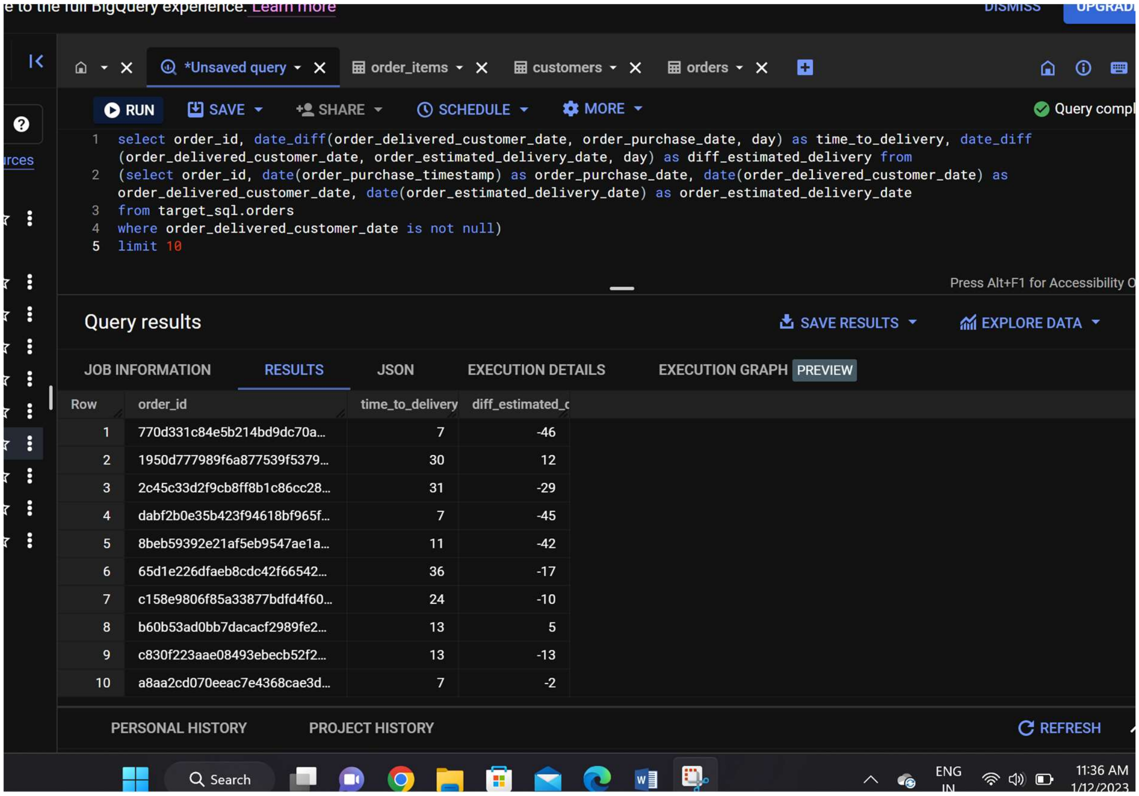
2. Find time_to_delivery & diff_estimated_delivery. Formula for the same given below:
   - time_to_delivery = order_purchase_timestamp-order_delivered_customer_date
   - diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date

   Ans. The query –

   select order_id, date_diff(order_delivered_customer_date, order_purchase_date, day) as time_to_delivery, date_diff(order_delivered_customer_date, order_estimated_delivery_date, day) as diff_estimated_delivery from

   (select order_id, date(order_purchase_timestamp) as order_purchase_date, date(order_delivered_customer_date) as order_delivered_customer_date, date(order_estimated_delivery_date) as order_estimated_delivery_date from target_sql.orders

where order_delivered_customer_date is not null)



```
select order_id, date_diff(order_delivered_customer_date, order_purchase_date, day) as time_to_delivery, date_diff
(order_delivered_customer_date, order_estimated_delivery_date, day) as diff_estimated_delivery from
(select order_id, date(order_purchase_timestamp) as order_purchase_date, date(order_delivered_customer_date) as
order_delivered_customer_date, date(order_estimated_delivery_date) as order_estimated_delivery_date
from target_sql.orders
where order_delivered_customer_date is not null)
limit 10
```

Query results

Row	order_id	time_to_delivery	diff_estimated_c
1	770d331c84e5b214bd9dc70a...	7	-46
2	1950d777989f6a877539f5379...	30	12
3	2c45c33d2f9cb8ff8b1c86cc28...	31	-29
4	dabf2b0e35b423f94618bf965f...	7	-45
5	8beb59392e21af5eb9547ae1a...	11	-42
6	65d1e226dfaeb8cdc42f66542...	36	-17
7	c158e9806f85a33877bdfd4f60...	24	-10
8	b60b53ad0bb7dacacf2989fe2...	13	5
9	c830f223aae08493ebecb52f2...	13	-13
10	a8aa2cd070eeac7e4368cae3d...	7	-2

3. Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery

   Ans. The query –

   select customer_state, round(avg(freight_value), 2) as mean_freight_value, round(avg(time_to_delivery), 2) as mean_time_to_delivery, round(avg(diff_estimated_delivery), 2) as mean_diff_estimated_delivery from

   (select customer_state, freight_value, date_diff(date(order_delivered_customer_date), date(order_purchase_timestamp), day) as time_to_delivery, date_diff(date(order_delivered_customer_date), date(order_estimated_delivery_date), day) as diff_estimated_delivery

   from target_sql.orders as a

   inner join `target_sql.customers` as b

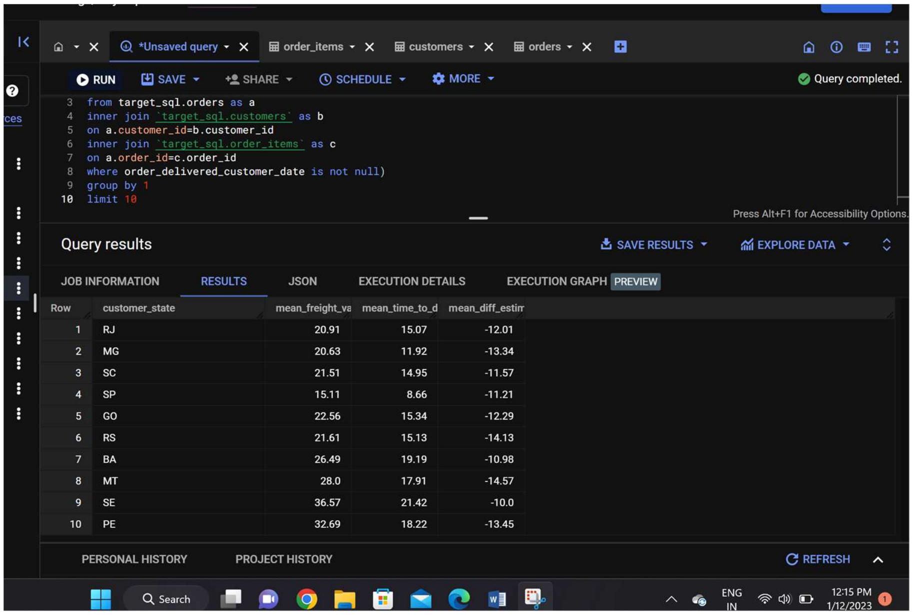
   on a.customer_id=b.customer_id

inner join `target_sql.order_items` as c

on a.order_id=c.order_id

where order_delivered_customer_date is not null)

group by 1



```
from target_sql.orders as a
inner join `target_sql.customers` as b
on a.customer_id=b.customer_id
inner join `target_sql.order_items` as c
on a.order_id=c.order_id
where order_delivered_customer_date is not null)
group by 1
limit 10
```

Row	customer_state	mean_freight_va	mean_time_to_d	mean_diff_estim
1	RJ	20.91	15.07	-12.01
2	MG	20.63	11.92	-13.34
3	SC	21.51	14.95	-11.57
4	SP	15.11	8.66	-11.21
5	GO	22.56	15.34	-12.29
6	RS	21.61	15.13	-14.13
7	BA	26.49	19.19	-10.98
8	MT	28.0	17.91	-14.57
9	SE	36.57	21.42	-10.0
10	PE	32.69	18.22	-13.45

4. Sort the data to get the following:
5. Top 5 states with highest/lowest average freight value - sort in desc/asc limit 5

Ans. The query for lowest 5 –

select customer_state from

(select customer_state, round(avg(freight_value), 2) as mean_freight_value, round(avg(time_to_delivery), 2) as mean_time_to_delivery, round(avg(diff_estimated_delivery), 2) as mean_diff_estimated_delivery from

(select customer_state, freight_value, date_diff(date(order_delivered_customer_date), date(order_purchase_timestamp), day) as time_to_delivery, date_diff(date(order_delivered_customer_date), date(order_estimated_delivery_date), day) as diff_estimated_delivery

from target_sql.orders as a

inner join `target_sql.customers` as b

on a.customer_id=b.customer_id

inner join `target_sql.order_items` as c

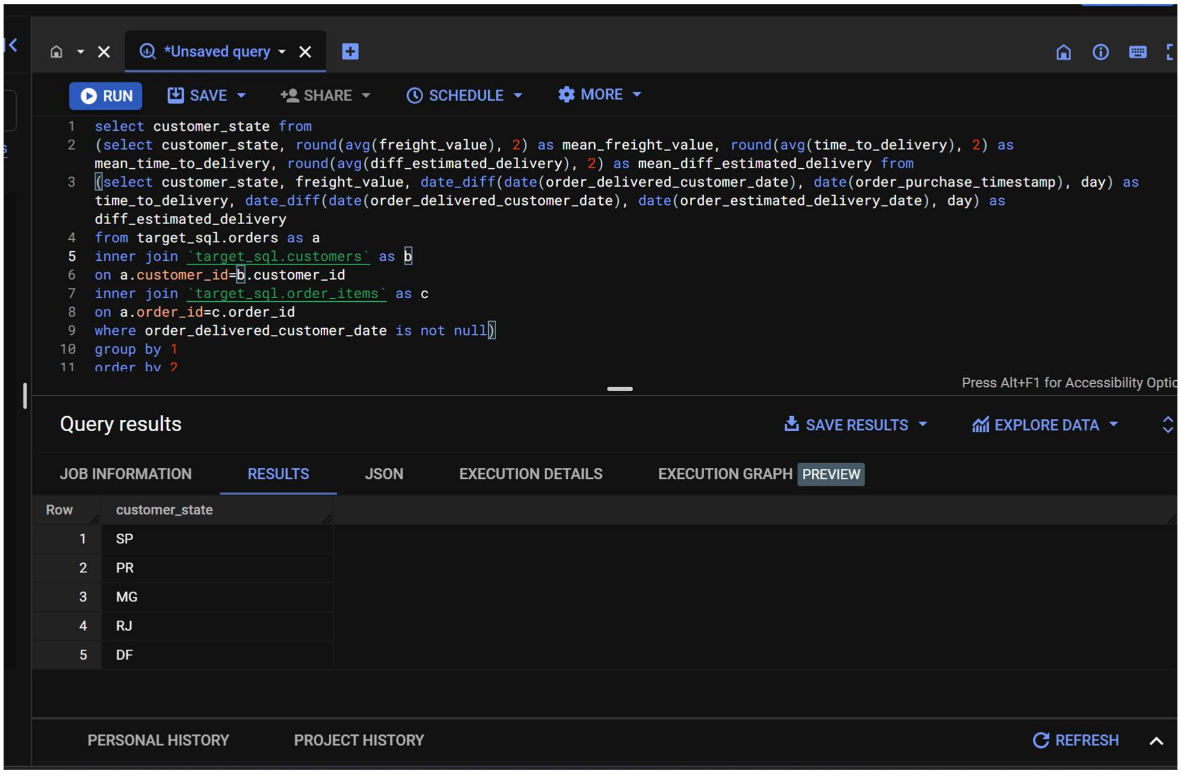
on a.order_id=c.order_id

where order_delivered_customer_date is not null)

group by 1

order by 2

limit 5)



```
select customer_state from
(select customer_state, round(avg(freight_value), 2) as mean_freight_value, round(avg(time_to_delivery), 2) as
mean_time_to_delivery, round(avg(diff_estimated_delivery), 2) as mean_diff_estimated_delivery from
(select customer_state, freight_value, date_diff(date(order_delivered_customer_date), date(order_purchase_timestamp), day) as
time_to_delivery, date_diff(date(order_delivered_customer_date), date(order_estimated_delivery_date), day) as
diff_estimated_delivery
from target_sql.orders as a
inner join `target_sql.customers` as b
on a.customer_id=b.customer_id
inner join `target_sql.order_items` as c
on a.order_id=c.order_id
where order_delivered_customer_date is not null)
group by 1
order by 2
```

Query results

Row	customer_state
1	SP
2	PR
3	MG
4	RJ
5	DF

6. Top 5 states with highest/lowest average time to delivery

Ans. The lowest states with average time to delivery query –

select customer_state from

(select customer_state, round(avg(freight_value), 2) as mean_freight_value, round(avg(time_to_delivery), 2) as mean_time_to_delivery, round(avg(diff_estimated_delivery), 2) as mean_diff_estimated_delivery from

(select customer_state, freight_value, date_diff(date(order_delivered_customer_date),

date(order_purchase_timestamp), day) as time_to_delivery, date_diff(date(order_delivered_customer_date), date(order_estimated_delivery_date), day) as diff_estimated_delivery

from target_sql.orders as a

inner join `target_sql.customers` as b

on a.customer_id=b.customer_id

inner join `target_sql.order_items` as c

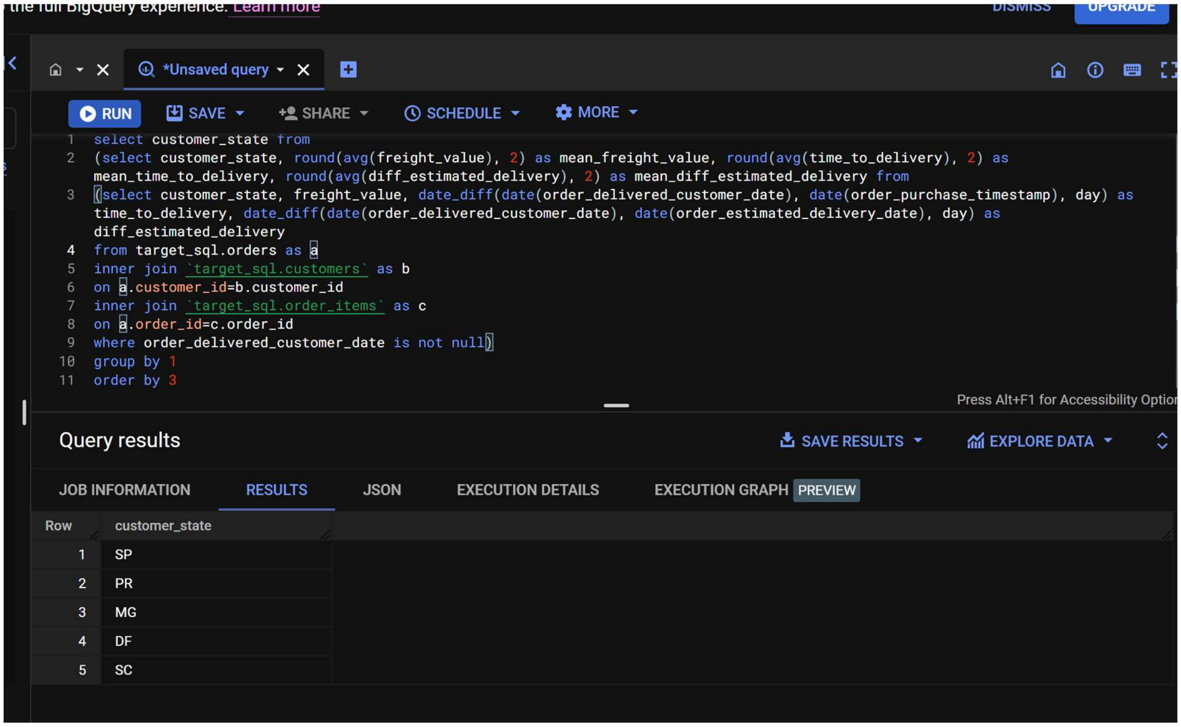
on a.order_id=c.order_id

where order_delivered_customer_date is not null)

group by 1

order by 3

limit 5)



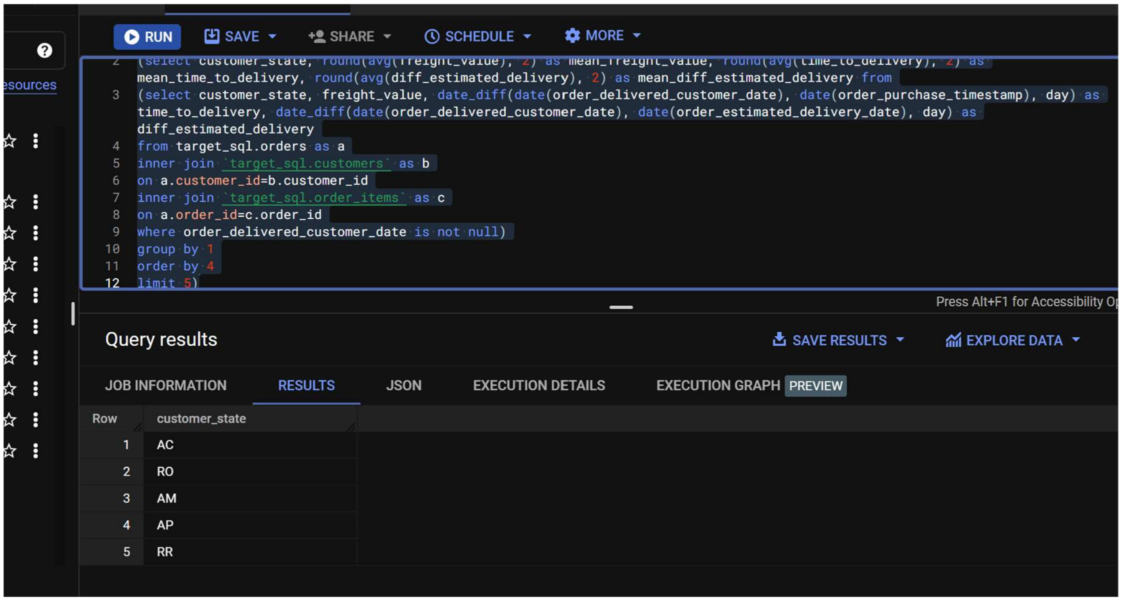
```
select customer_state from
(select customer_state, round(avg(freight_value), 2) as mean_freight_value, round(avg(time_to_delivery), 2) as
mean_time_to_delivery, round(avg(diff_estimated_delivery), 2) as mean_diff_estimated_delivery from
(select customer_state, freight_value, date_diff(date(order_delivered_customer_date), date(order_purchase_timestamp), day) as
time_to_delivery, date_diff(date(order_delivered_customer_date), date(order_estimated_delivery_date), day) as
diff_estimated_delivery
from target_sql.orders as a
inner join `target_sql.customers` as b
on a.customer_id=b.customer_id
inner join `target_sql.order_items` as c
on a.order_id=c.order_id
where order_delivered_customer_date is not null)
group by 1
order by 3
```

Row	customer_state
1	SP
2	PR
3	MG
4	DF
5	SC

7. Top 5 states where delivery is really fast/ not so fast compared to estimated date

Ans. The states where delivery is fast compared to estimated date query –

```
select customer_state from

(select customer_state, round(avg(freight_value), 2) as mean_freight_value,
round(avg(time_to_delivery), 2) as mean_time_to_delivery,
round(avg(diff_estimated_delivery), 2) as mean_diff_estimated_delivery from

(select customer_state, freight_value,
date_diff(date(order_delivered_customer_date),
date(order_purchase_timestamp), day) as time_to_delivery,
date_diff(date(order_delivered_customer_date),
date(order_estimated_delivery_date), day) as diff_estimated_delivery

from target_sql.orders as a

inner join `target_sql.customers` as b

on a.customer_id=b.customer_id

inner join `target_sql.order_items` as c

on a.order_id=c.order_id

where order_delivered_customer_date is not null)

group by 1

order by 4

limit 5)
```



```
(select customer_state, round(avg(freight_value), 2) as mean_freight_value, round(avg(time_to_delivery), 2) as
mean_time_to_delivery, round(avg(diff_estimated_delivery), 2) as mean_diff_estimated_delivery from
(select customer_state, freight_value, date_diff(date(order_delivered_customer_date), date(order_purchase_timestamp), day) as
time_to_delivery, date_diff(date(order_delivered_customer_date), date(order_estimated_delivery_date), day) as
diff_estimated_delivery
from target_sql.orders as a
inner join `target_sql.customers` as b
on a.customer_id=b.customer_id
inner join `target_sql.order_items` as c
on a.order_id=c.order_id
where order_delivered_customer_date is not null)
group by 1
order by 4
limit 5)
```

Query results

Row	customer_state
1	AC
2	RO
3	AM
4	AP
5	RR

6. Payment type analysis:

    1. Month over Month count of orders for different payment types

       Ans. The query –

       select date_trunc(date(order_purchase_timestamp), month) as month_value, payment_type, count(distinct a.order_id) as total_orders

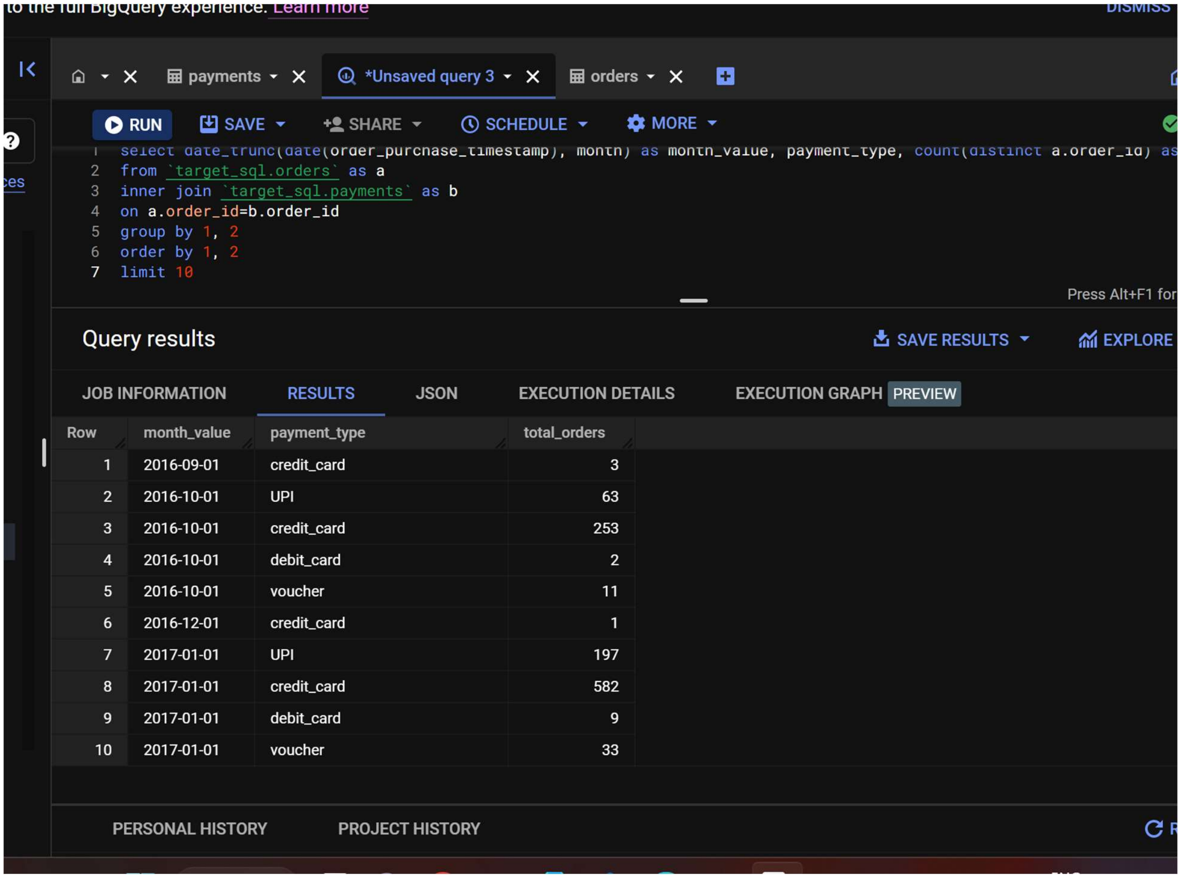
       from `target_sql.orders` as a

       inner join `target_sql.payments` as b

       on a.order_id=b.order_id

       group by 1, 2

       order by 1, 2



```
select date_trunc(date(order_purchase_timestamp), month) as month_value, payment_type, count(distinct a.order_id) as
from `target_sql.orders` as a
inner join `target_sql.payments` as b
on a.order_id=b.order_id
group by 1, 2
order by 1, 2
limit 10
```

Query results

Row	month_value	payment_type	total_orders
1	2016-09-01	credit_card	3
2	2016-10-01	UPI	63
3	2016-10-01	credit_card	253
4	2016-10-01	debit_card	2
5	2016-10-01	voucher	11
6	2016-12-01	credit_card	1
7	2017-01-01	UPI	197
8	2017-01-01	credit_card	582
9	2017-01-01	debit_card	9
10	2017-01-01	voucher	33

    2. Count of orders based on the no. of payment installments

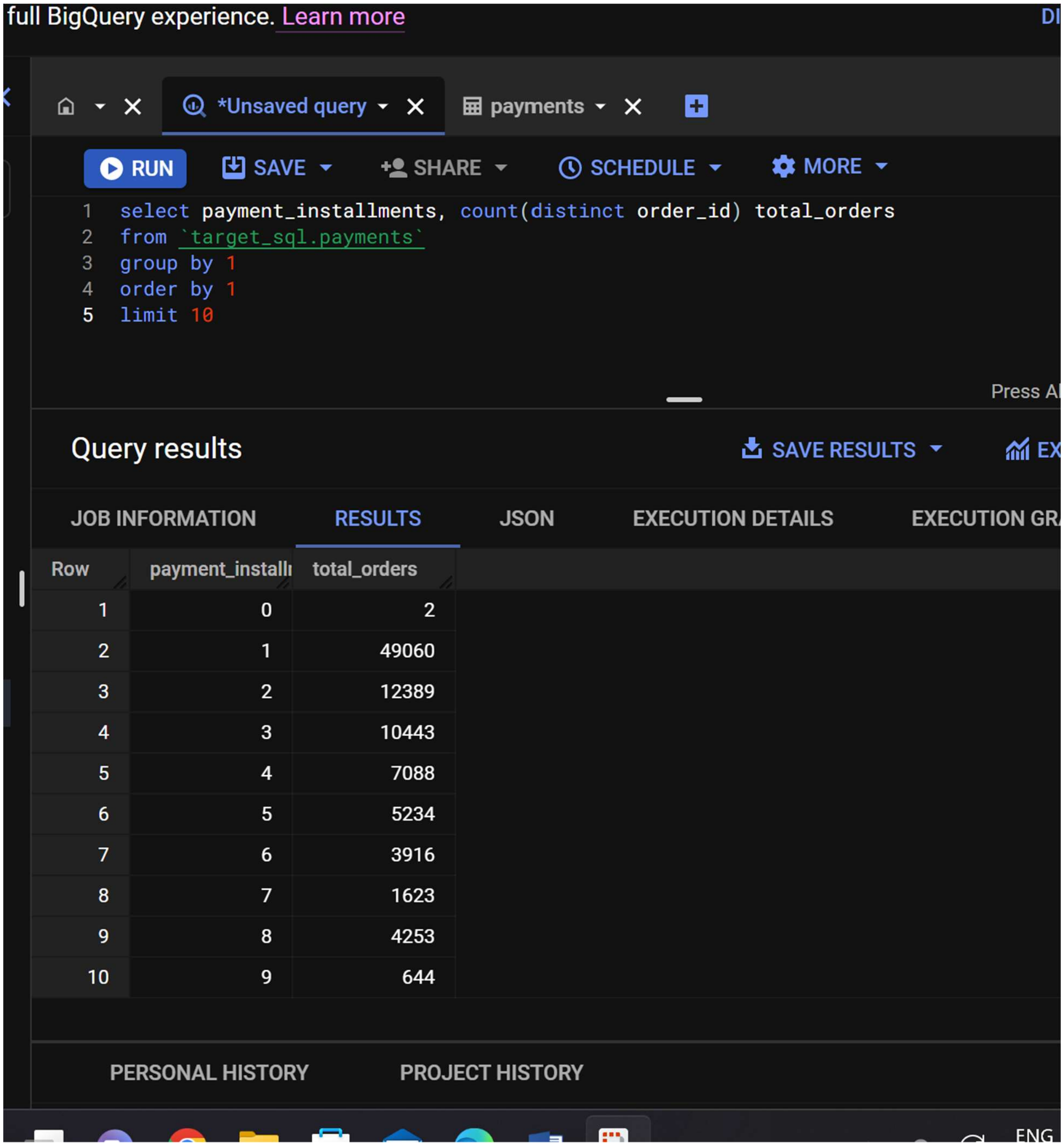
       Ans. The query –

select payment_installments, count(distinct order_id) total_orders

from `target_sql.payments`

group by 1

order by 1



```
select payment_installments, count(distinct order_id) total_orders
from `target_sql.payments`
group by 1
order by 1
limit 10
```

Row	payment_install	total_orders
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

Recommendations and Actionable Insights

1. **Product Category v/s Review**

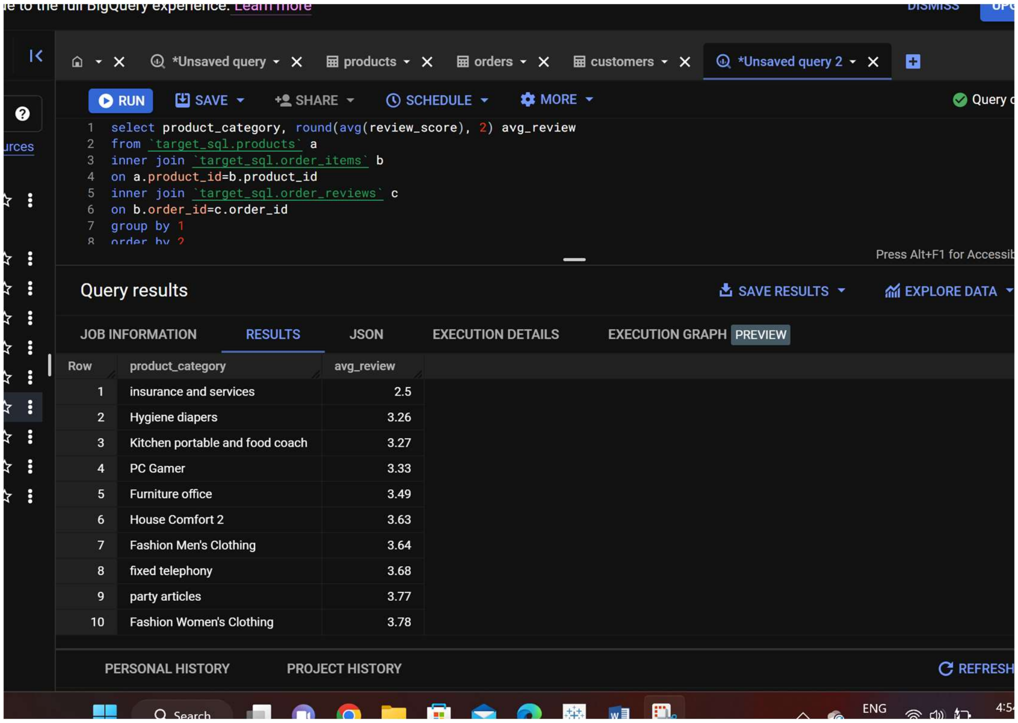
   Assumption :

Review	Comment
1	Very Bad
2	Bad
3	Neutral
4	Good
5	Excellent

4-5 tends to be positive effect.

The query –

```
select product_category, round(avg(review_score), 2) avg_review
from `target_sql.products` a
inner join `target_sql.order_items` b
on a.product_id=b.product_id
inner join `target_sql.order_reviews` c
on b.order_id=c.order_id
group by 1
order by 2
```



```
select product_category, round(avg(review_score), 2) avg_review
from `target_sql.products` a
inner join `target_sql.order_items` b
on a.product_id=b.product_id
inner join `target_sql.order_reviews` c
on b.order_id=c.order_id
group by 1
order by 2
```

Row	product_category	avg_review
1	insurance and services	2.5
2	Hygiene diapers	3.26
3	Kitchen portable and food coach	3.27
4	PC Gamer	3.33
5	Furniture office	3.49
6	House Comfort 2	3.63
7	Fashion Men's Clothing	3.64
8	fixed telephony	3.68
9	party articles	3.77
10	Fashion Women's Clothing	3.78

From the results it is seen that insurance and services product category has the least average review. This particular product category is not popular as the review score is very low and total only 2 products have been sold. Survey could be conducted to understand how to improve the product.

2. Customer State vs Review

The query –

select customer_state, round(avg(review_score), 2) avg_review

from `target_sql.products` a

inner join `target_sql.order_items` b

on a.product_id=b.product_id

inner join `target_sql.order_reviews` c

on b.order_id=c.order_id

inner join `target_sql.orders` d

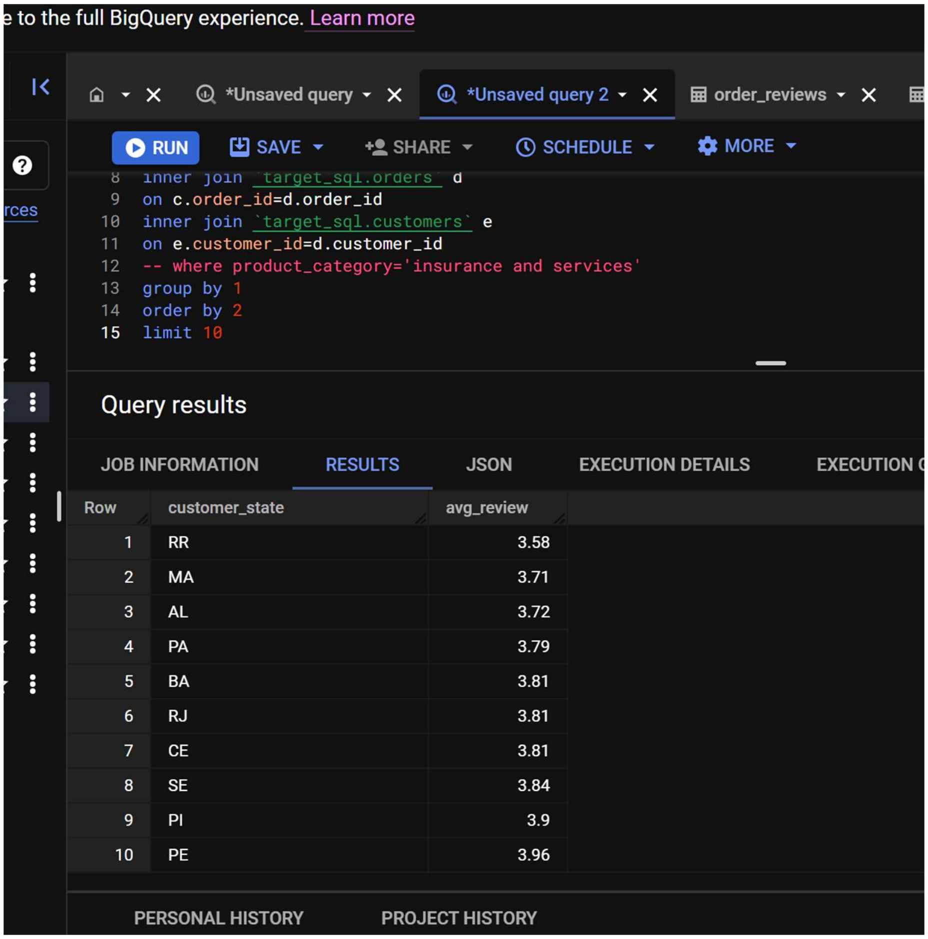
on c.order_id=d.order_id

inner join `target_sql.customers` e

on e.customer_id=d.customer_id

group by 1

order by 2



```
inner join `target_sql.orders` d
on c.order_id=d.order_id
inner join `target_sql.customers` e
on e.customer_id=d.customer_id
-- where product_category='insurance and services'
group by 1
order by 2
limit 10
```

Row	customer_state	avg_review
1	RR	3.58
2	MA	3.71
3	AL	3.72
4	PA	3.79
5	BA	3.81
6	RJ	3.81
7	CE	3.81
8	SE	3.84
9	PI	3.9
10	PE	3.96

All the average reviews of sates is more than 3.5. Sate wise all reviews are good. No immediate action needs to be taken. Main focus needs to be to maintain the review on all states.

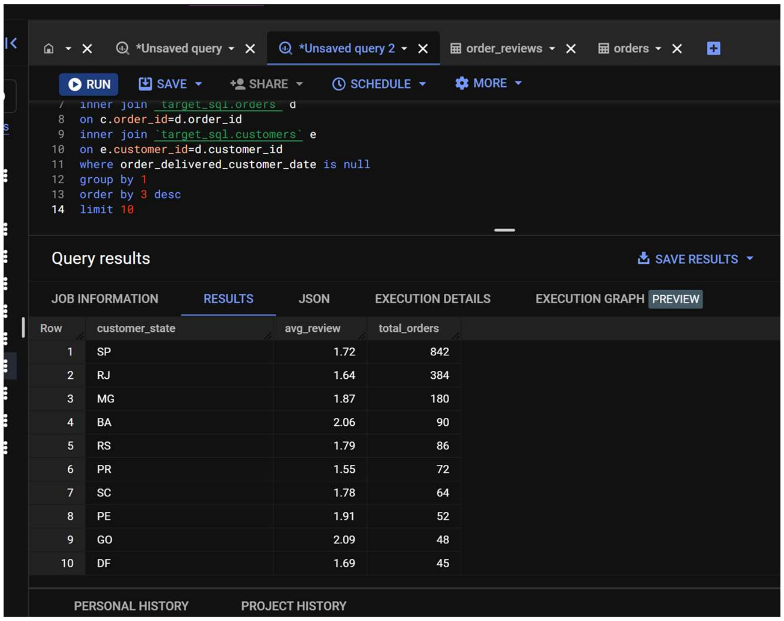
3. Analysis of returned orders

I am taking the average review and total orders of returned orders. The query –

select customer_state, round(avg(review_score), 2) avg_review, count(distinct c.order_id) total_orders

from `target_sql.products` a

```
inner join `target_sql.order_items` b

on a.product_id=b.product_id

inner join `target_sql.order_reviews` c

on b.order_id=c.order_id

inner join `target_sql.orders` d

on c.order_id=d.order_id

inner join `target_sql.customers` e

on e.customer_id=d.customer_id

where order_delivered_customer_date is null

group by 1

order by 3 desc
```



```
inner join `target_sql.orders` d
on c.order_id=d.order_id
inner join `target_sql.customers` e
on e.customer_id=d.customer_id
where order_delivered_customer_date is null
group by 1
order by 3 desc
limit 10
```

Query results

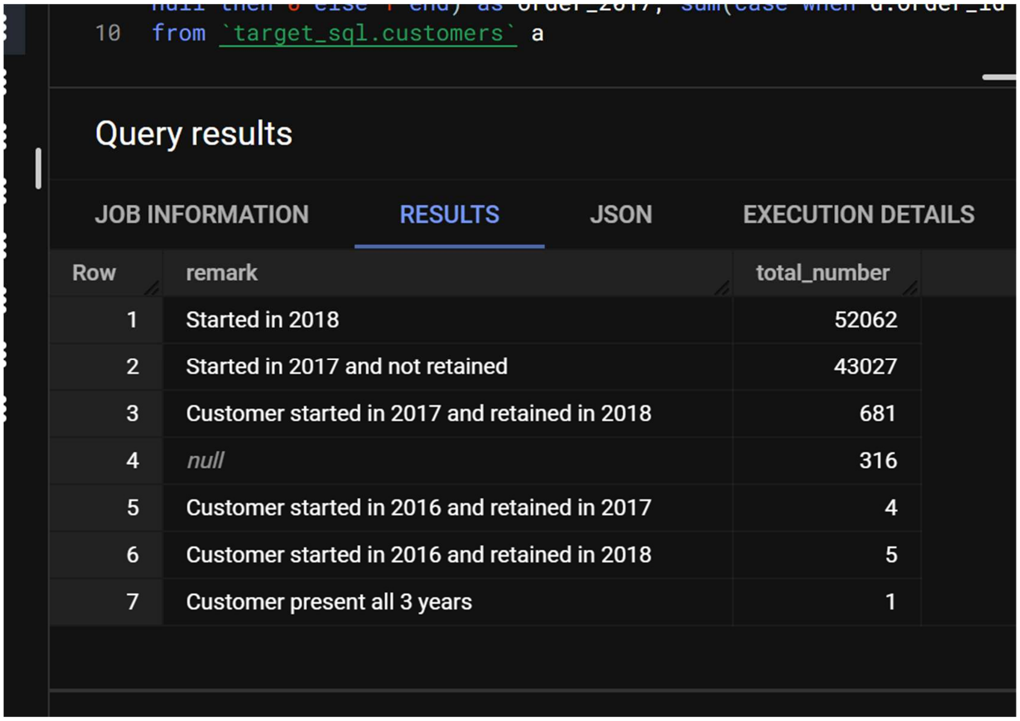
Row	customer_state	avg_review	total_orders
1	SP	1.72	842
2	RJ	1.64	384
3	MG	1.87	180
4	BA	2.06	90
5	RS	1.79	86
6	PR	1.55	72
7	SC	1.78	64
8	PE	1.91	52
9	GO	2.09	48
10	DF	1.69	45

State SP has around 850 orders returned back and average review is 1.72. If we were to look at the comments in the review (review title). Most of them is empty. We need to contact these customers to get proper feedback about the order experience and try to implement their recommendations and see any scope of improvement.

4. Customer Retention Analysis

The query –

```
select case when (order_2018>0 and order_2017>0 and order_2016>0) then 'Customer present all 3 years'

when (order_2018>0 and order_2017>0) then 'Customer started in 2017 and retained in 2018'

when (order_2017>0 and order_2016>0) then 'Customer started in 2016 and retained in 2017'

when (order_2018>0 and order_2016>0) then 'Customer started in 2016 and retained in 2018'

when order_2018>0 then 'Started in 2018'

when order_2017>0 then 'Started in 2017 and not retained'

when order_2017>0 then 'Started in 2016 and not retained'

else null end as remark, count(*) total_number from

(select a.customer_unique_id, sum(case when b.order_id is null then 0 else 1 end) as order_2016, sum(case when c.order_id is null then 0 else 1 end) as order_2017, sum(case when d.order_id is null then 0 else 1 end) as order_2018

from `target_sql.customers` a

left join (select *

from `target_sql.orders`

where extract(year from date(order_purchase_timestamp))=2016) b

on a.customer_id=b.customer_id

left join (select *

from `target_sql.orders`

where extract(year from date(order_purchase_timestamp))=2017) c

on a.customer_id=c.customer_id

left join (select *

from `target_sql.orders`
where extract(year from date(order_purchase_timestamp))=2018) d

on a.customer_id=d.customer_id

group by 1)

group by 1
```



```
null then 0 else 1 end) as order_2017, sum(case when d.order_id
from `target_sql.customers` a
```

Query results

Row	remark	total_number
1	Started in 2018	52062
2	Started in 2017 and not retained	43027
3	Customer started in 2017 and retained in 2018	681
4	*null*	316
5	Customer started in 2016 and retained in 2017	4
6	Customer started in 2016 and retained in 2018	5
7	Customer present all 3 years	1

As seen from the results, 43K customers bought in 2017 but did not buy in 2018. Need to understand the root cause for them not ordering again from us. We can introduce some sort of rewards to buy every year from us (like some sort of royalty program).