Business case: Target SQL

1(a)-Data type of columns in a table

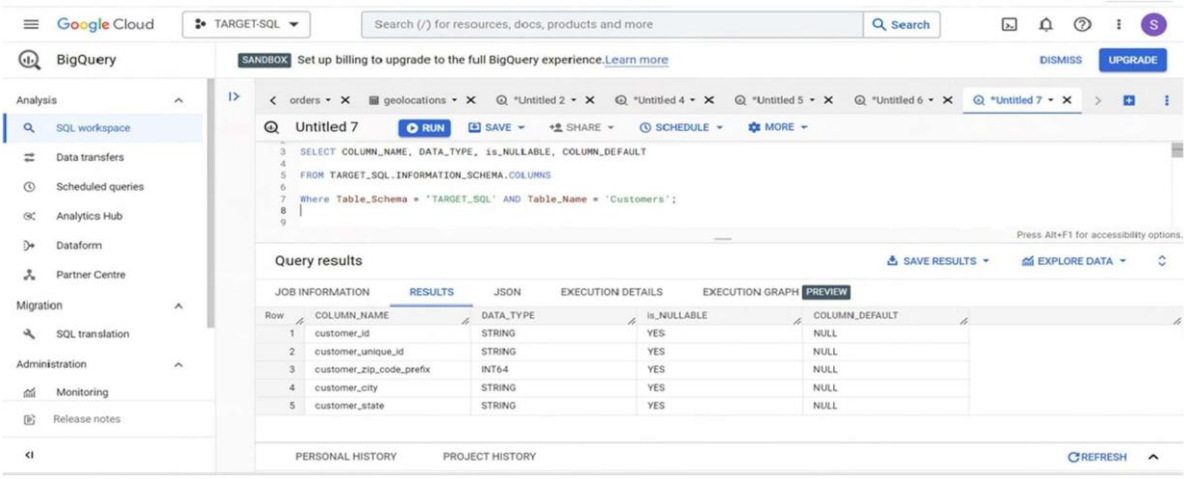
- Datatype ofcolumninacustomertable

  Query-

```
SELECT COLUMN_NAME, DATA_TYPE, is_NULLABLE, COLUMN_DEFAULT

FROM TARGET_SQL.INFORMATION_SCHEMA.COLUMNS

Where Table_Schema = "TARGET_SQL" AND Table_Name = "Customers";
```



Insights - Customer table has the information about the customer and it has 5 column and all the column stores the different type of information about the customers.

1. **Customer_id** - Data type of this column is **STRING** and it stores the id of customer who purchase. It can be null or default value of this column is null.
2. **Customer_unique_id** - Data type of this column is **STRING** and it stores id of the customer created an account on Target so this is the **UNIQUE** value column because every customer unique_id.
3. **Customer_zip_code_prefix** - Data type of this column is **INTEGER(64)** and it stores the pin code of customer.
4. **Customer_city** - Data type of this column is **STRING** and it stores the city from where order placed.
5. **Customer_state** - Data type of this column is **STRING** and it stores the state code where order has been placed.

Recommendations - After analysing the structure of this table my suggestion is that change the column customer_zip_code_prefix data type int(64) to int because customer pin code is not more than 5 digits and int(64) is use for very large number if you use int(64) then it creates performance issue

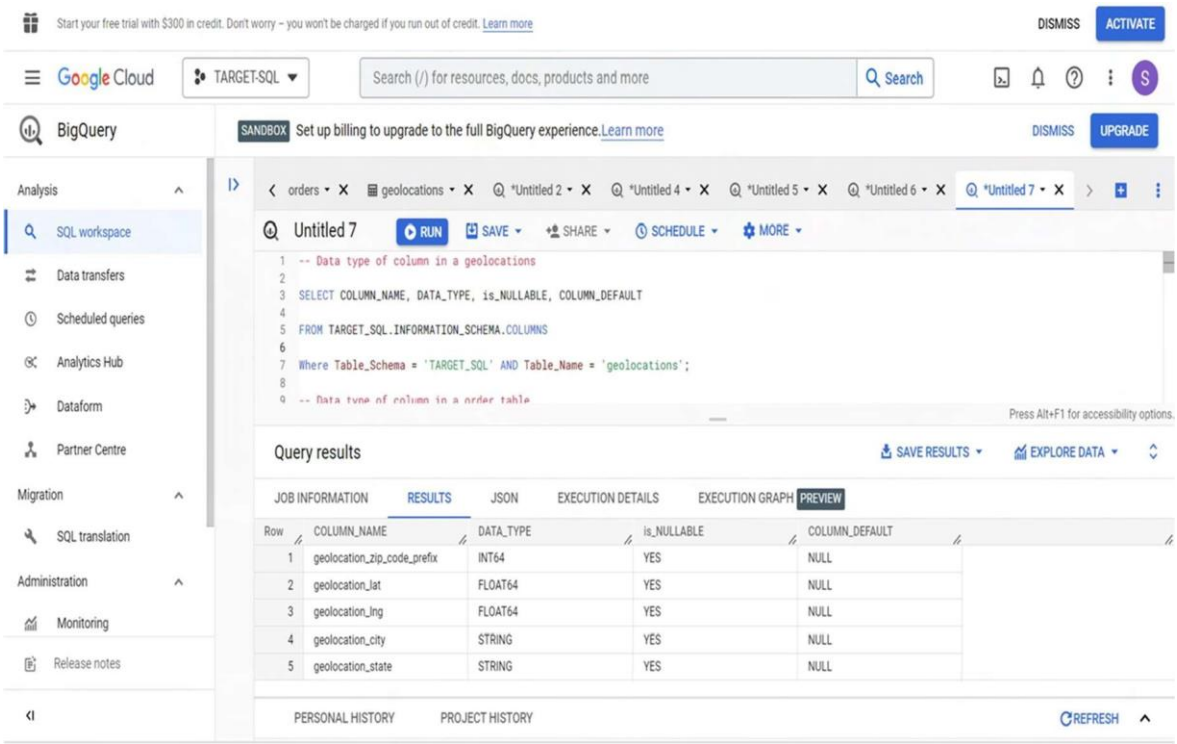
- Data type of column in a Geolocation table

  Query-

```
SELECT COLUMN_NAME, DATA_TYPE, is_NULLABLE, COLUMN_DEFAULT

FROM TARGET_SQL.INFORMATION_SCHEMA.COLUMNS

Where Table_Schema = "TARGET_SQL" AND Table_Name = "geolocations";
```



Insights- Geolocation table stores the information about the customer exact location like longitude, latitude, city and state and it has 5 columns and every column stores different type of information.

1. **geolocation_zip_code_prefix -** Data type of this column is **INTEGER(64)** and it stores the pin code of customer.
2. **geolocation_lat** - Data type of this column is **FLOAT(64)** and it stores the latitude of customer location.
3. **geolocation_lng -** Data type of this column is **FLOAT(64)** and it stores the longitude of the customer location.
4. **geolocation_city** - Data type of this column is **STRING** and it stores the city of customer.
5. **geolocation _state** - Data type of this location is **STRING** and it stores the state code of the customer.

Recommendations - After analysing the structure of this table my suggestion is that change the column customer_zip_code_prefix data type int(64) to int because customer pin code is not more than 5 digits and int(64) is use for very large number if you use int(64) then it creates performance issue in reading and writing both.

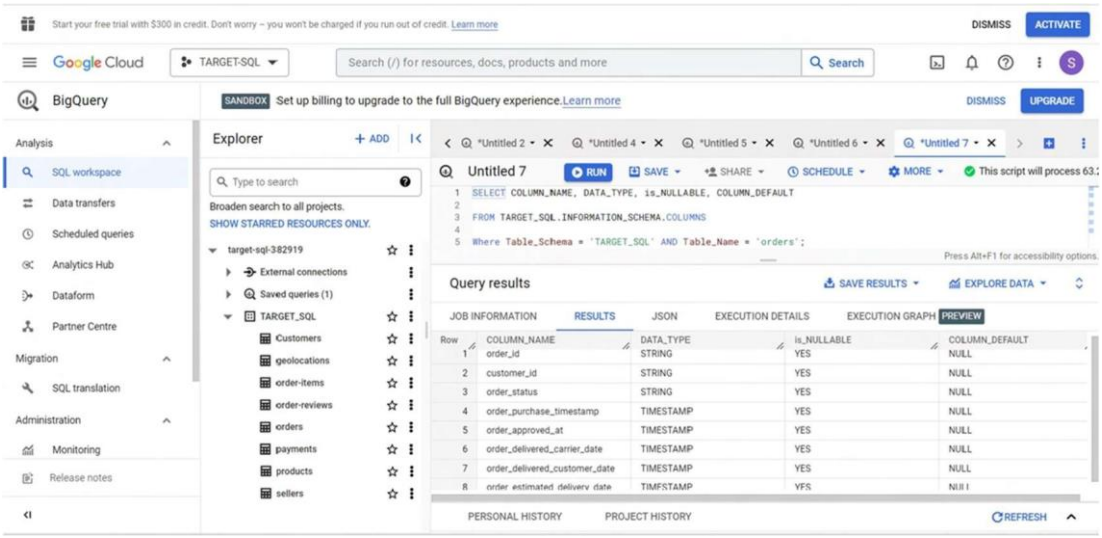
- Data type of column in an orders table

  Query-

```
SELECT COLUMN_NAME, DATA_TYPE, is_NULLABLE, COLUMN_DEFAULT

FROM TARGET_SQL.INFORMATION_SCHEMA.COLUMNS

Where Table_Schema = "TARGET_SQL" AND Table_Name = "orders";
```



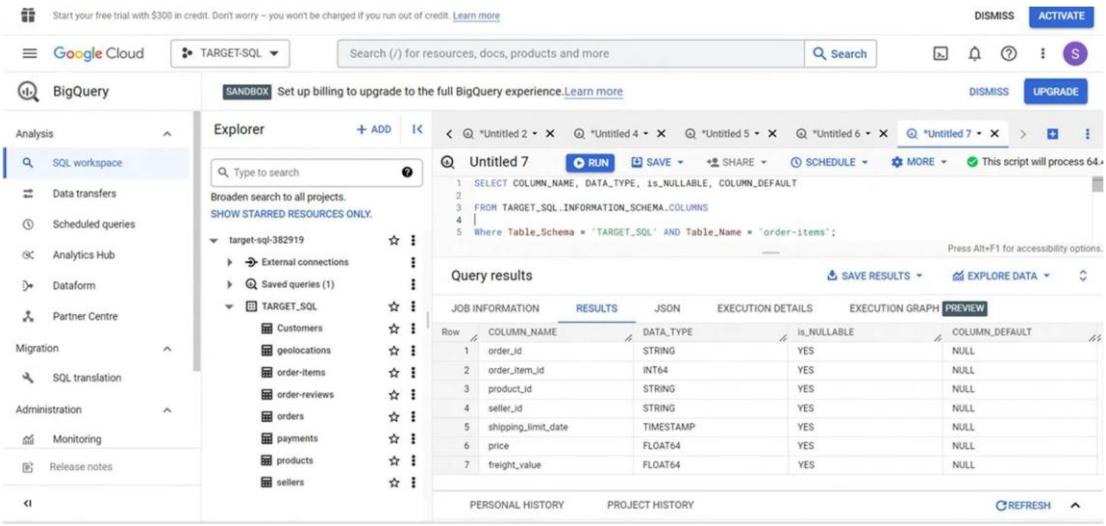
Insights- orders table stores the delivery related information, it has 8 column and all the column stores different type of information about order delivery.

1. **order_id -** Data type of this column is **STRING** and every order has unique id so this is the unique id column.
2. **customer_id** - Data type of this column is **STRING** and it stores the customer_id of customer who made purchase, it can be duplicate.
3. **order_status** - Data type of this column is **STRING** and it stores the status of the order made i.e., delivered, shipped etc.
4. **order_purchase_timestamp** - Data type of this column is **TIMESTAMP** and it stores the timestamp of that when order get placed.
5. **order_approved_at** - Data type of this column is **TIMESTAMP** and it stores the timestamp when order gets approved from sellerend.
6. **order_delivered_carrier_date -** Data type of this column is **TIMESTAMP** and it stores the timestamp when shipped orders get delivered to the customer.
7. **order_delivered_customer_date-** Data type of this column is **TIMESTAMP** and it stores the timestamp of that when order got received by customer.
8. **order_estimated_delievery_date-** Data type of this column is **TIMESTAMP** and it stores the timestamp of when it will delivered on or before.

- **Data type of column in an order items table**

  Query-

```
SELECT COLUMN_NAME, DATA_TYPE, is_NULLABLE, COLUMN_DEFAULT
FROM TARGET_SQL.INFORMATION_SCHEMA.COLUMNS
Where Table_Schema = "TARGET_SQL" AND Table_Name = "order-items";
```



Insights – order items table stores the information like customer_id of customer who placed order, id of order items, id of product, id of seller who supply the product, shipping limit date, price of item, delivery charge it has 7 columns and all the column stores different type of information about order items.

1. **order_id -** Data type of this column is **STRING** and every order has unique id so this is a unique id column, it can't be duplicate.
2. **order_item_id-** Data type of this column is **INT(64)** and every item has unique id so this is the unique_id column, itcan't be duplicate.
3. **product_id** - Data type of this column is **STRING** and every product listed on target has unique id so this is also a unique id column, it can't be duplicate.
4. **seller_id** - Data type of this column is **STRING** and it is a unique id column because every seller listed on target who supplies the product has unique id, it can't be duplicate.
5. **shiping_limit_date-** Data type of this column is **TIMESTAMP** and it stores the timestamp of when product is shipped on or before.
6. **price** - Data type of this column is **FLOAT(64)** and it stores the price of the product.
7. **freight_value** - Data type of this column is **FLOAT(64)** and it stores the delievery charge of the product.

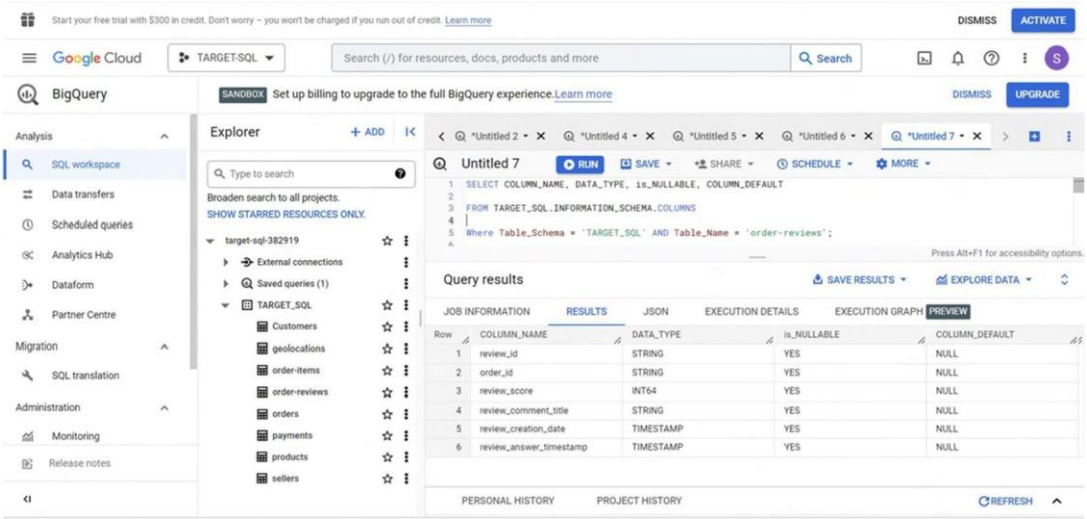
Recommendations - After analysing the structure of this table my suggestion is mention below-

1. change the column order_item_id data type int(64) to int because order item is not more is a very small number and int(64) is use for very large number if you use int(64) then itvcreates performance issue in reading and writing both.
2. change the column price data type float(64) to float because order item has value up to 3 to 4 decimal point and float(64) is use for very huge precision if you use float(64) then it creates performance issue in reading and writing both.
3. change the column freight price data type float(64) to float because order item has value up to 3 to 4 decimal point and float(64) is use for very huge precision if you use float(64) then it creates performance issue in reading and writing both.

- Data type of column in an order reviews table

Query-

```
SELECT COLUMN_NAME, DATA_TYPE, is_NULLABLE, COLUMN_DEFAULT
FROM TARGET_SQL.INFORMATION_SCHEMA.COLUMNS
Where Table_Schema = "TARGET_SQL" AND Table_Name = "order-reviews";
```



Insights- order reviews table stores the information of reviews which has been given by customer, it has 6 column and all the column stores different type of information about order review.

1. **review_id -** Data type of this column is **STRING** and it stores the id of the review given on product by customer.
2. **order_id** - Data type of this column is **STRING** and it stores the unique id of order made by customer.
3. **review_score** - Data type of this column is **INT(64)** and it stores the review score given by the customer for each order on the scale of 1–5.
4. **review_comment_title** - Data type of this column is **STRING** and it stores the comment given on every review.

5. **review_creation_date** - Data type of this column is **TIMESTAMP** and it stores the timestamp of when review has been given.
6. **review_answer_timestamp** - Data type of this column is **TIMESTAMP** and it stores the timestamp of when the review gets answered.

Recommendations - After analysing the structure of this table my suggestion is mention below

1. change the column review score data type int(64) to int because review score range is 1 to 5 which is very small number and int(64) is use for very large number if you use int(64) then it creates performance issue in reading and writing both.

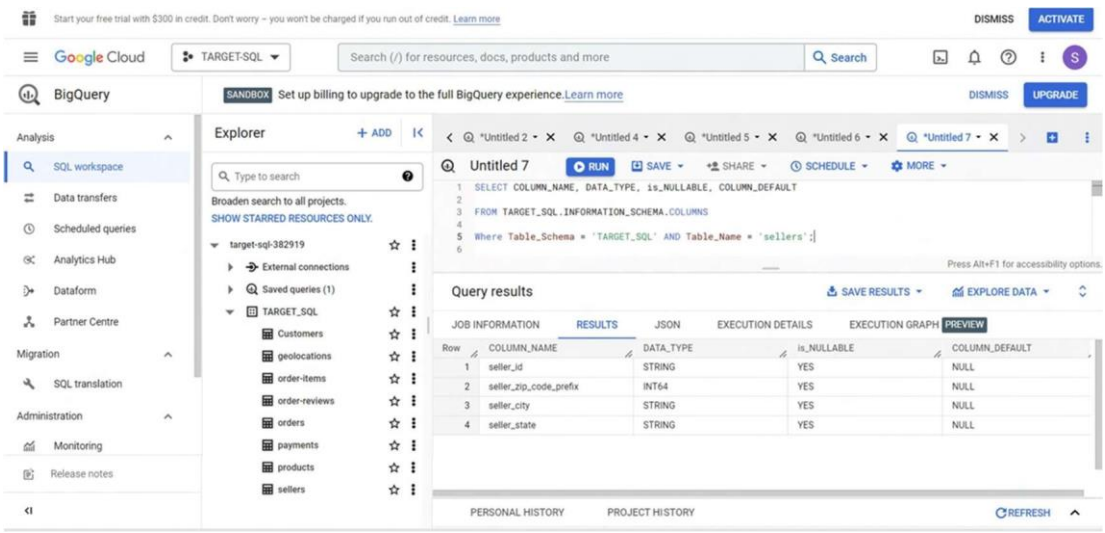
- Data type of column in an order sellers table

  Query-

```
SELECT COLUMN_NAME, DATA_TYPE, is_NULLABLE, COLUMN_DEFAULT

FROM TARGET_SQL.INFORMATION_SCHEMA.COLUMNS

Where Table_Schema = "TARGET_SQL" AND Table_Name = "sellers";
```



Insights - sellers table stores the information about sellers who is listed on Target site, it has 4 column and every store different type of information.

1. **seller_id** - Data type of this column is **STRING** and it stores the id of seller who listed on target site and every seller has unique id so it is the unique id column, it can't be duplicate.
2. **seller_zip_code_prefix** - Data type of this column is **INT(64)** and it stores the pin code of seller address.
3. **seller_city** - Data type of this column is **STRING** and it stores the city of the seller address.
4. **seller_state** - Data type of this column is **STRING** and it stores the state code of seller address.

Recommendations - After analysing the structure of this table my suggestion isnmention below-

1. change the column seller_zip_code_prefix data type int(64) to int because customer pin code is not more than 5 digits and int(64) is use for very large number if you use int(64) then it creates performance issue.

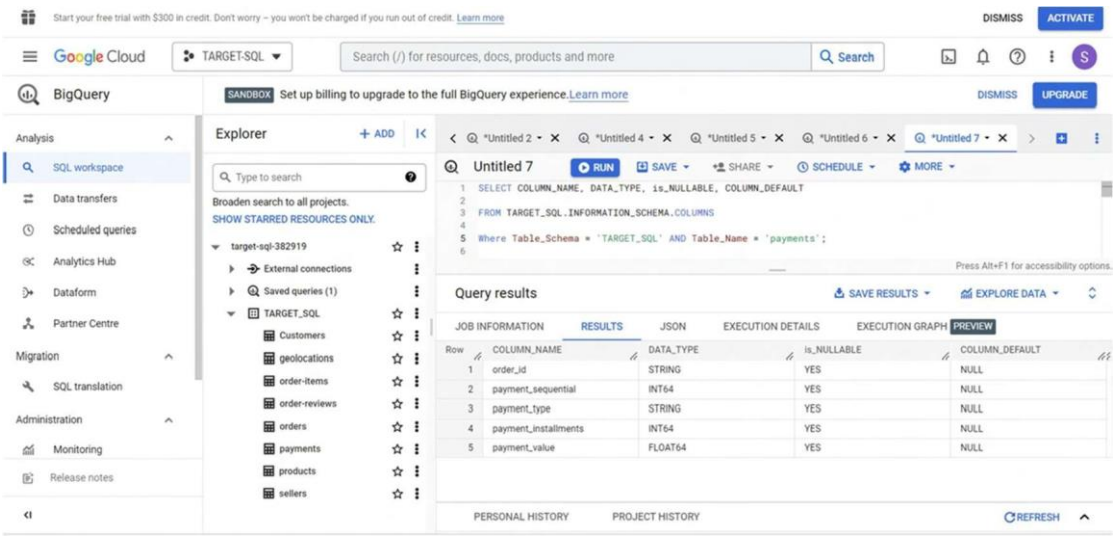
- Data type of column in an order payments table

Query-

```
SELECT COLUMN_NAME, DATA_TYPE, is_NULLABLE, COLUMN_DEFAULT

FROM TARGET_SQL.INFORMATION_SCHEMA.COLUMNS

Where Table_Schema = "TARGET_SQL" AND Table_Name = "payments";
```



Insights - payments table stores the information about payments, it has 4 column and every column store different type of information about payments.

1. **order_id** - Data type of this column is **STRING** and it stores the id of order and every order has unique id so it is a unique id column, it can't be null.
2. **payment_sequential** - Data type of this column is **INT(64)** and it store the payment sequence in case of EMI.
3. **payment_type** - Data type of this column is **STRING** and it store the information about payment mode.
4. **payment_installment** - Data type of this column is **INT(64)** and it stores number of instalments in caseof EMI.
5. **payment_value** - Data type of this column is **FLOAT(64)** and it store the total payment of everyorder.

Recommendations- After analysing the structure of this table my suggestion is mention below.

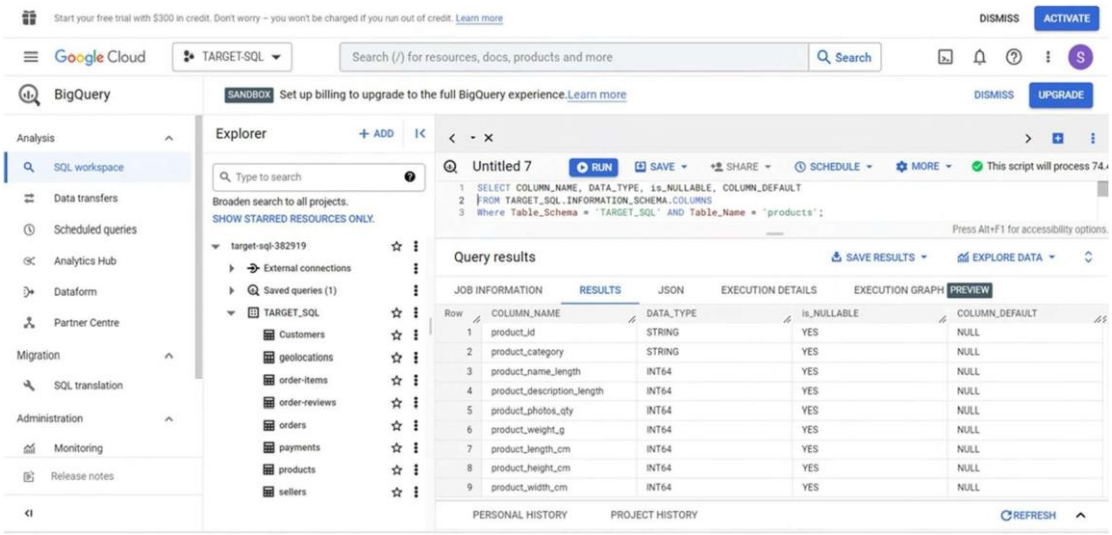
1. change the column payment_sequential data type int(64) to int because order item is not more is a very small number and int(64) is use for very large number if you use int(64) then it creates performance issue in reading and writing both.

2. change the column payment_installment data type int(64) to int because order item is not more is a very small number and int(64) is use for very large number if you use int(64) then it creates performance issue in reading and writing both.
3. change the column payment value data type float(64) to float because order item has value up to 3 to 4 decimal point and float(64) is use for very huge precision if you use float(64) then it creates performance issue in reading and writing both.

- Data type of column in an order products table

  Query-

```
SELECT COLUMN_NAME, DATA_TYPE, is_NULLABLE, COLUMN_DEFAULT
FROM TARGET_SQL.INFORMATION_SCHEMA.COLUMNS
Where Table_Schema = "TARGET_SQL" AND Table_Name = "products";
```



Insights - products table stores the information about product who is listed on Target site, it has 9 column and every column store different type of information about product.

1. **product_id** - Data type of this column is **STRING** and it stores the id of product which is listed on Target site and every product has unique id so this the unique id column.
2. **product_category** - Data type of this column is **STRING** and it stores the category of product which is listed on Target site.
3. **product_name_length** - Data type of this column is **INT(64)** and it stores the length of the product name.
4. **product_description_length** - Data type of this column is **INT(64)** and it store the length of description given about product.
5. **product_photos_qty** - Data type of this column is **INT(64)** and it store the count of photos available for particular product.
6. **product_weight_g** - Data type of this column is **INT(64)** and it store the weight of the product in gram.

7. **product_length_cm** - Data type of this column is **INT(64)** and it store the length of the product in centimetre.
8. **product_height_cm** - Data type of this column is **INT(64)** and it store the height of the product in centimetre.
9. **product_width_cm** - Data type of this column is **INT(64)** and it store the width of the product in centimetre.

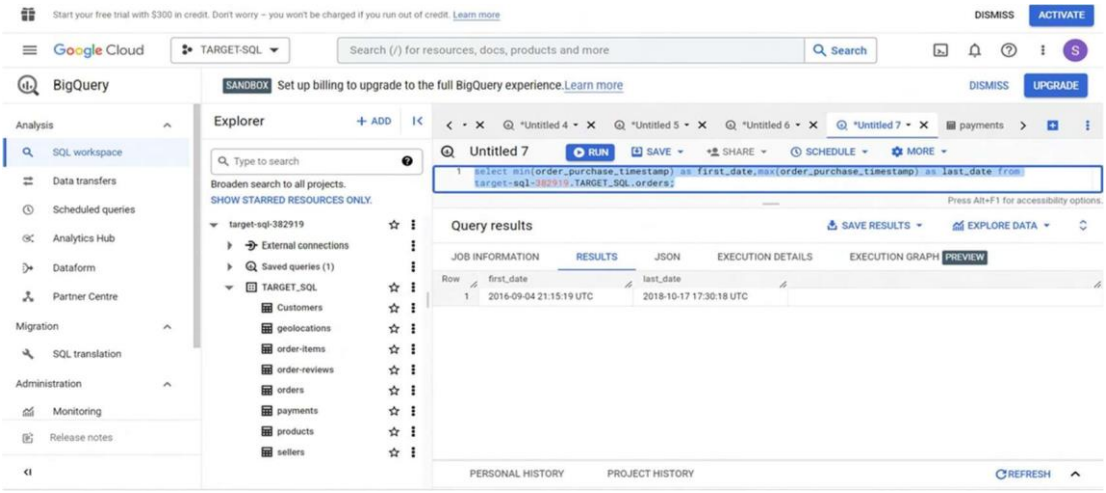
Recommendations- After analysing the structure of this table my suggestion is mention below.

1. Data type of column product_name_length, product_description_length, product_photos_qty, product_weight_g, product_length_cm, product_height_cm, product_height_cm and product_width_cm should be **INT** because in this column value are small and **INT(64)** is use for very big value.

1 (b)-Time period for which the data is given.

Query-

```
SELECT MIN(order_purchase_timestamp) as first_date,
MAX(order_purchase_timestamp) as last_date
FROM target-sql-382919.TARGET_SQL.orders;
```

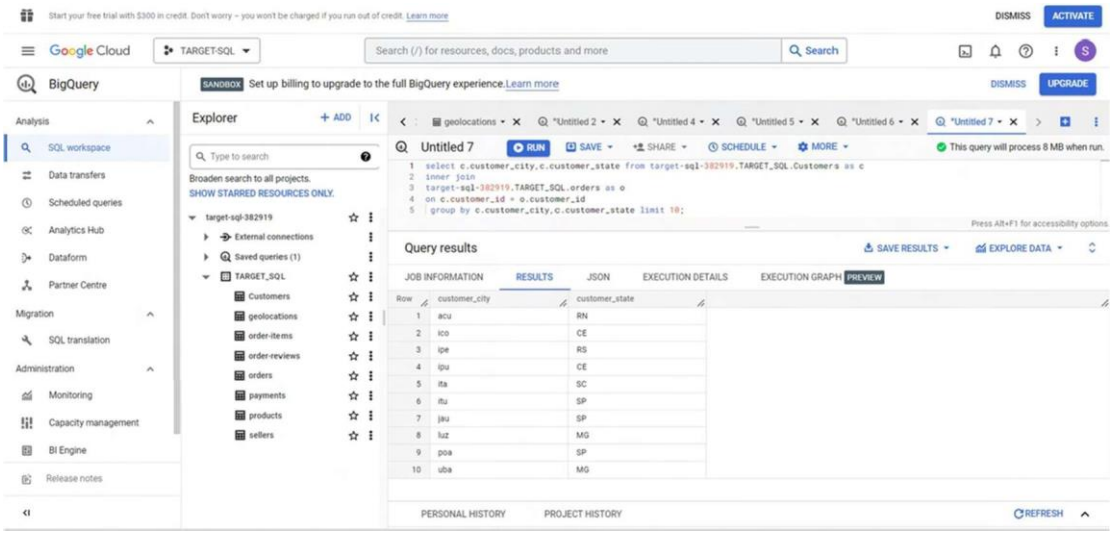


Insight- Time period of given date is 4th September 2016 to 17th October 2018

1 (c)-Cities and state of customer order during given period.

Query-

```
SELECT c.customer_city,c.customer_state
FROM target-sql-382919.TARGET_SQL.Customers as c
INNER JOIN
target-sql-382919.TARGET_SQL.orders as o ON c.customer_id = o.customer_id
GROUP BY c.customer_city,c.customer_state limit 10;
```

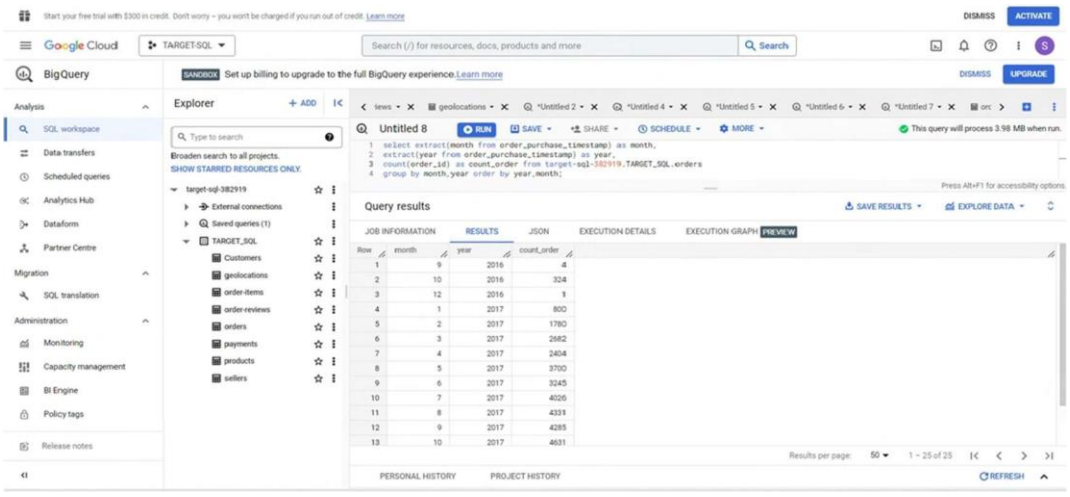


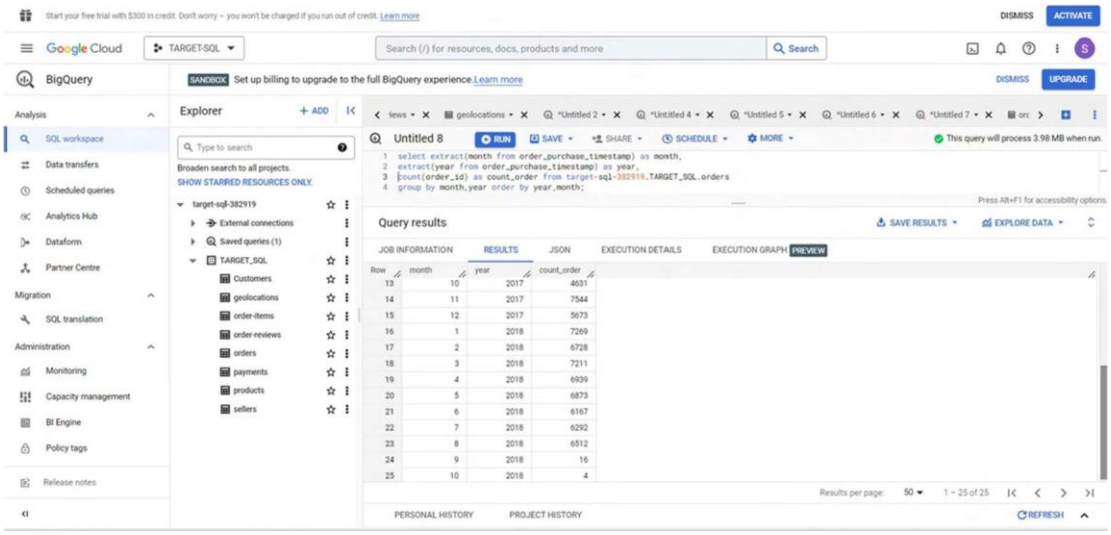
Insights- After analysing the customer state and customer city wise record from the data I find that the total no of city and state from where customer placed order is respectively 4119 and 27. One more thing I observe that the maximum order has given by state **SP** and minimum order has given by state **RR.**

2 (a) 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?

Query-

```
select extract(month from order_purchase_timestamp) as month,
extract(year from order_purchase_timestamp) as year,
count(order_id) as count_order from target-sql-382919.TARGET_SQL.orders
group by month, year order by year, month;
```





Insights - Yes there is a growing trend on E-commerce in Brazil and it is between January 2017 to August 2018.I can describe this scenario after calculating the month-to-month order count. After analysing there is no peeks at specific month but we can say that in summer and spring seasons number of order is slightly greater than other season.

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

Query-

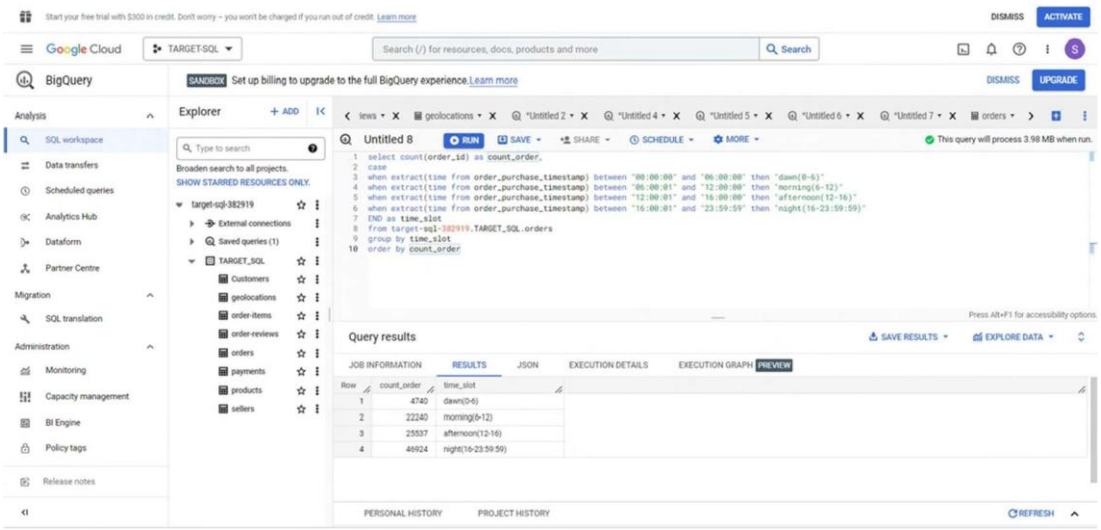
```
select count(order_id) as count_order,
case
when extract(time from order_purchase_timestamp)
between "00:00:00" and "06:00:00" then "dawn (0-6)"

when extract(time from order_purchase_timestamp)
between "06:00:01" and "12:00:00" then "morning (6-12)"

when extract(time from order_purchase_timestamp)
between "12:00:01" and "16:00:00" then "afternoon (12-16)"

when extract(time from order_purchase_timestamp)
between "16:00:01" and "23:59:59" then "night (16-23:59:59)"

END as time_slot
from target-sql-382919.TARGET_SQL.orders
group by time_slot
order by count_order desc;
```



Assumption- I assume that the time slot is

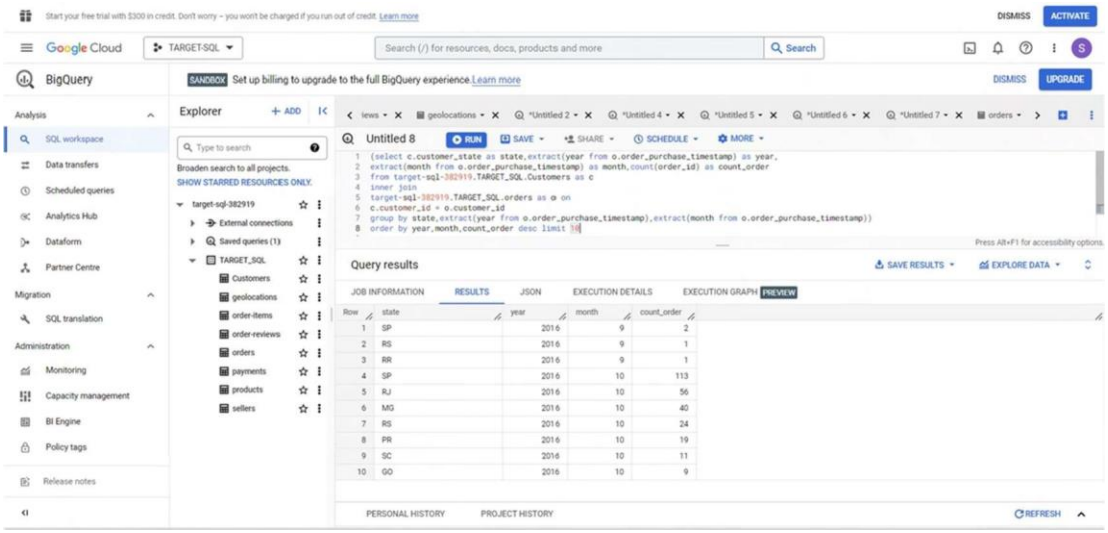
Dawn – (0-6)
Morning – (6-12)
Afternoon – (12-16)
Night or Evening – (16-23:59:59)

Insight - After analysing the data I find that in the Night timeslot(16-23:59:59) customer tends to buy more as compare to another timeslot.

3 (a)- Get month on month orders by states.

Query-

```
(select c.customer_state as state,
extract(year from o.order_purchase_timestamp) as year,
extract(month from o.order_purchase_timestamp) as month,
count(order_id) as count_order
from target-sql-382919.TARGET_SQL.Customers as c
inner join
target-sql-382919.TARGET_SQL.orders as o on
c.customer_id = o.customer_id
group by state, extract(year from o.order_purchase_timestamp),
extract(month from o.order_purchase_timestamp))
order by year, month, count_order desc limit 10;
```

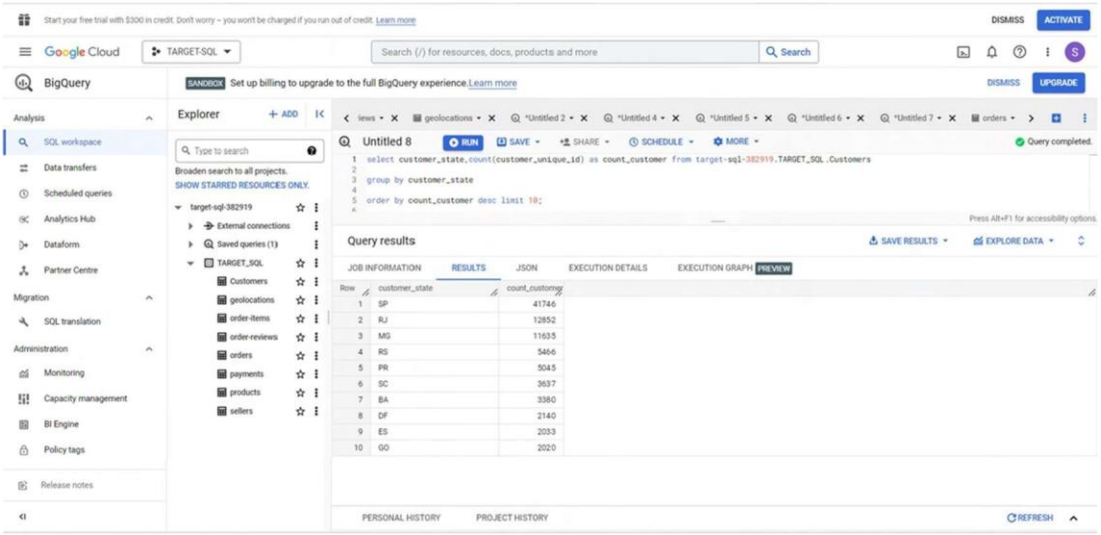


Insight- After analysing the data I find that every month state **SP** gives more order as compare to other state.

3 (b) Distribution of customers across the states in Brazil.

Query-

```
select customer_state, count(customer_unique_id) as count_customer
from target-sql-382919.TARGET_SQL.Customers
group by  customer_state
order by count_customer desc;
```

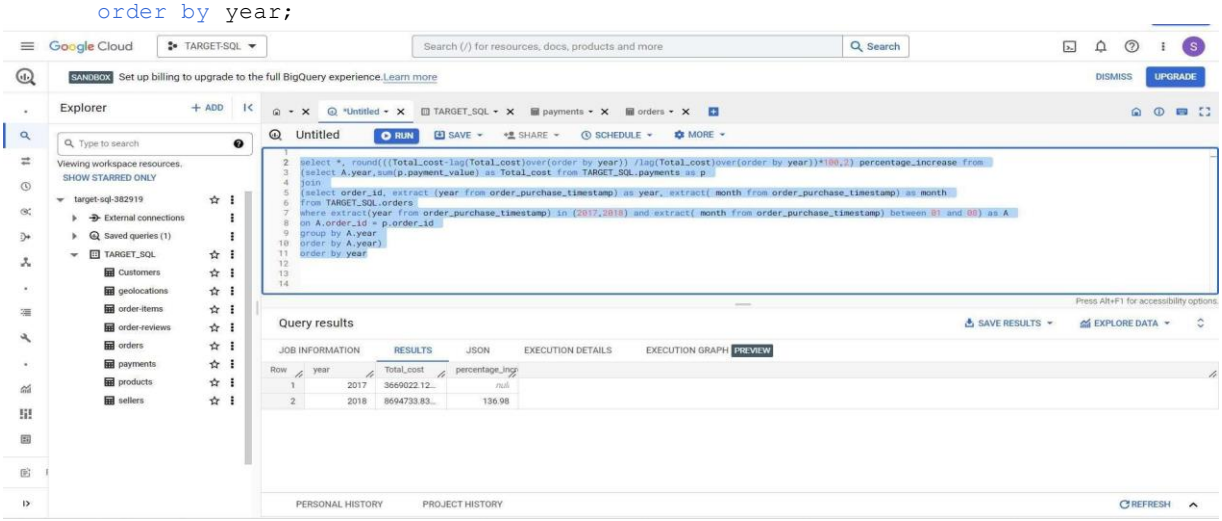


Insight-After analysing the data I find that the maximum and minimum number of customers who sign up in Target is belong from state **SP** and **RR** respectively.

4(a_)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.

Query-

```
select *,round(((Total_cost-
lag(Total_cost)over(order by year)) /lag(Total_cost)over(order by year))*100,2)
percentage_increase from
(select A.year,sum(p.payment_value) as Total_cost from TARGET_SQL.payments as p
join
(select order_id, extract (year from order_purchase_timestamp) as year,
extract (month from order_purchase_timestamp) as month
from TARGET_SQL.orders
where extract(year from order_purchase_timestamp) in (2017,2018) and
extract(month from order_purchase_timestamp) between 01 and 08) as A
on A.order_id = p.order_id
group by A.year
order by A.year)
order by year;
```

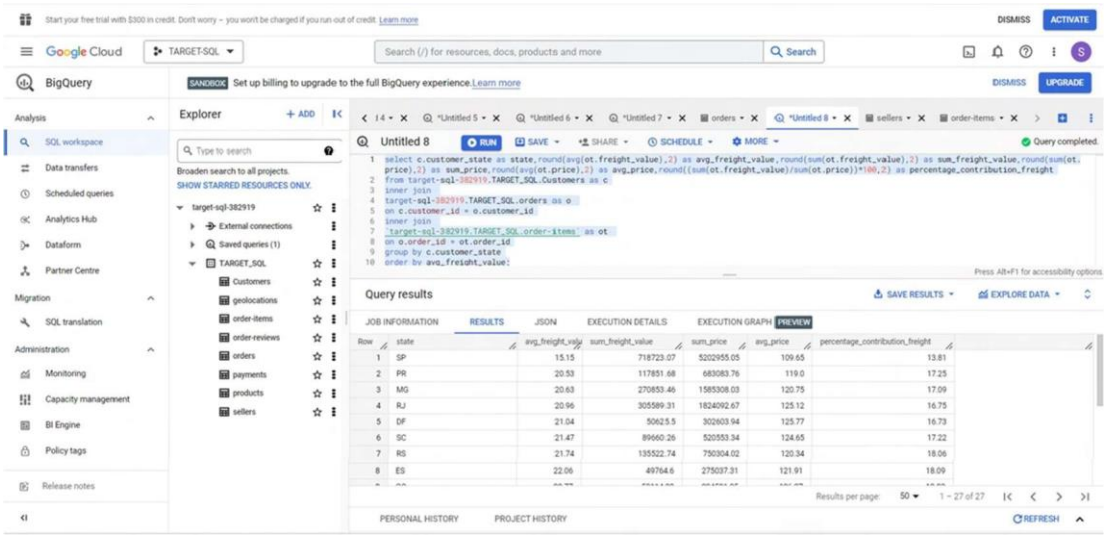


Insight - After analyzing the data I found that % increase in cost of order from 2017 to 2018 (include months between Jan to Aug) is increased by 136.98%

4 (b) Mean & Sum of price and freight value by customer state.

Query-

```
select c.customer_state as state,
round(avg(ot.freight_value),2) as avg_freight_value,
round(sum(ot.freight_value),2) as sum_freight_value,
round(sum(ot.price),2) as sum_price,
round(avg(ot.price),2) as avg_price,
round((sum(ot.freight_value)/sum(ot.price))*100,2)
as percentage_contribution_freight
from target-sql-382919.TARGET_SQL.Customers as c
inner join
target-sql-382919.TARGET_SQL.orders as o
on c.customer_id = o.customer_id
inner join
`target-sql-382919.TARGET_SQL.order-items` as ot
on o.order_id = ot.order_id group by c.customer_state
order by avg_freight_value;
```



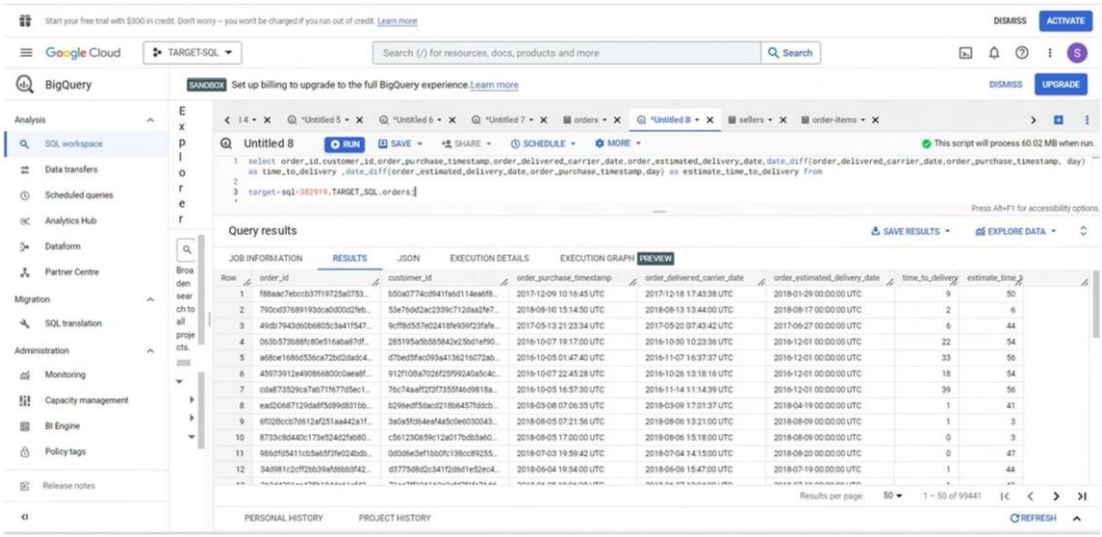
Insight -

1. After analysing the data I find that the which state generating more revenue and which state generating less revenue forcompany.
2. After calculating the freight value is how much percentage of product price then I observe that in the 50% of state freight value is more than 20% of product value.

5 (a)- Calculate days between purchasing, delivering and estimated delivery.

Query-

```
select order_id,customer_id,order_purchase_timestamp,
order_delivered_carrier_date, order_estimated_delivery_date,
date_diff(order_delivered_carrier_date,order_purchase_timestamp,    day)
as time_to_delivery ,
date_diff(order_estimated_delivery_date,order_purchase_timestamp,day)
as estimate_time_to_delivery from
target-sql-382919.TARGET_SQL.orders;
```

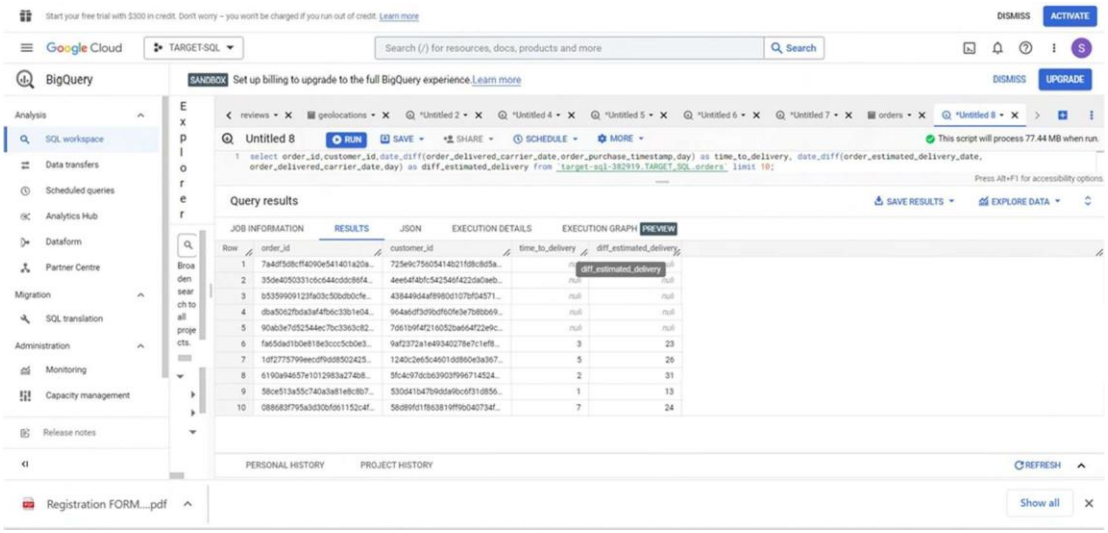


Insight- After analysing the data, I found the information about the order delivery of how much time order is taking to delivered and what is the estimated time to delivered.

5 (b)- Find time_to_delivery & diff_estimated_delivery.

Query-

```
select order_id, customer_id,
date_diff(order_delivered_carrier_date,order_purchase_timestamp,day)
as time_to_delivery,
date_diff(order_estimated_delivery_date,order_delivered_carrier_date,day)
as diff_estimated_delivery from `target-sql-382919.TARGET_SQL.orders`
limit 10;
```

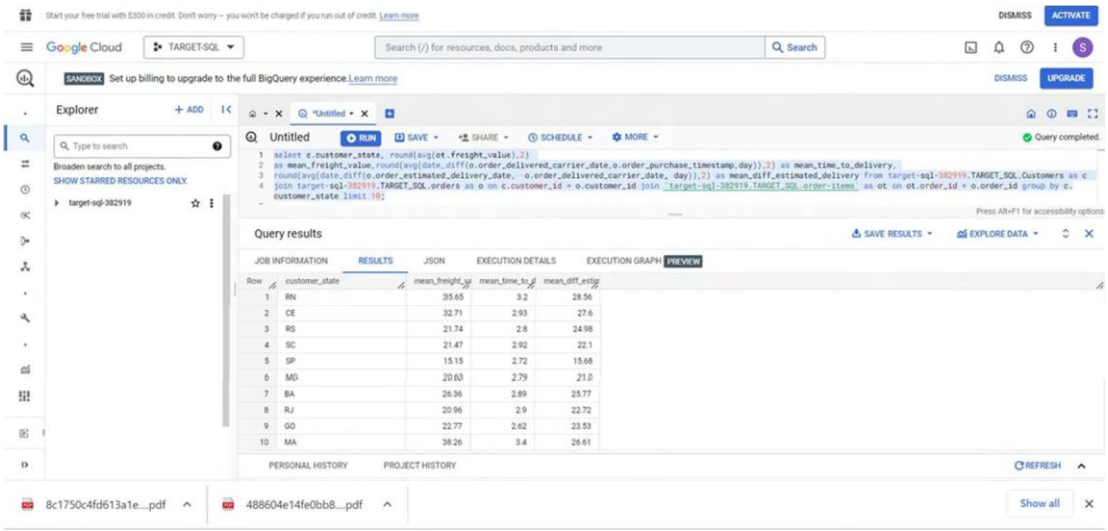


Insight- After analysing the data, I find the information about order delivery of how much time order is taking to delivered and what days difference between order delivered date and estimated delivery date. One more thing I seen in result which is first 5 order time_to_delivery and diff_estimated delivery is null it shows that the after placing the order they get cancelled.

5(c) Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery.

Query-

```
select c.customer_state, round(avg(ot.freight_value),2)
as mean_freight_value,
round(avg(date_diff(o.order_delivered_carrier_date,
o.order_purchase_timestamp,day)),2)
as mean_time_to_delivery,
round(avg(date_diff(o.order_estimated_delivery_date,
o.order_delivered_carrier_date, day)),2)
as mean_diff_estimated_delivery
from target-sql-382919.TARGET_SQL.Customers as c
join target-sql-382919.TARGET_SQL.orders as o on
c.customer_id = o.customer_id
join `target-sql-382919.TARGET_SQL.order-items` as ot
on ot.order_id = o.order_id group by c.customer_state;
```



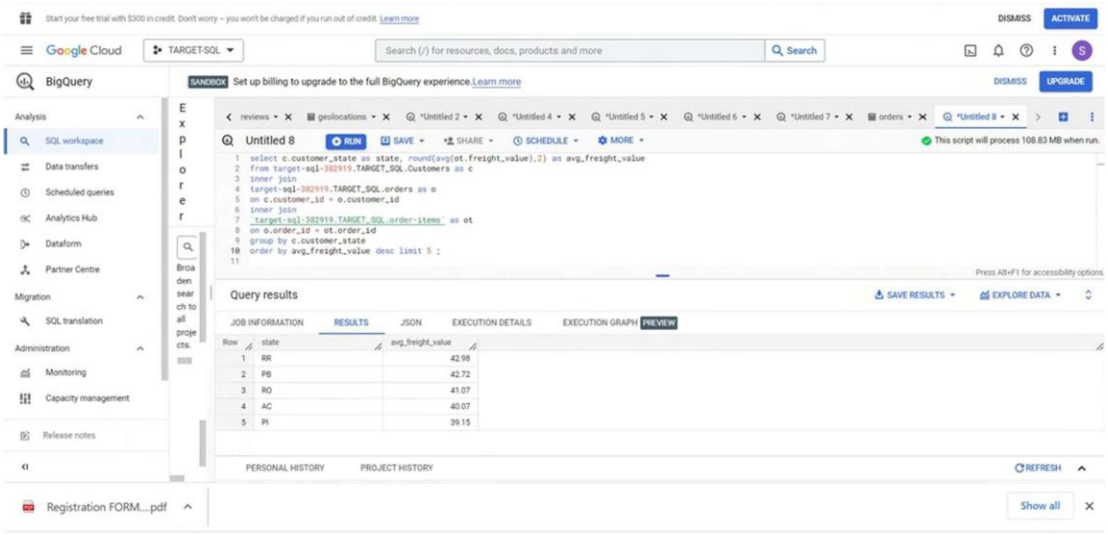
Insight - After analysing the data, I find that the mean of freight value approx. range is between 20 to 40 in every state and time to delivery is approx. 3 days in every state.one more thing I observe that the estimated delivery date is too high as compare to order delivery date.

5(d)- Top 5 states with highest/lowest average freight value - sort in desc/asc limit 5.

Query-

Top 5 states with highest average freight value.

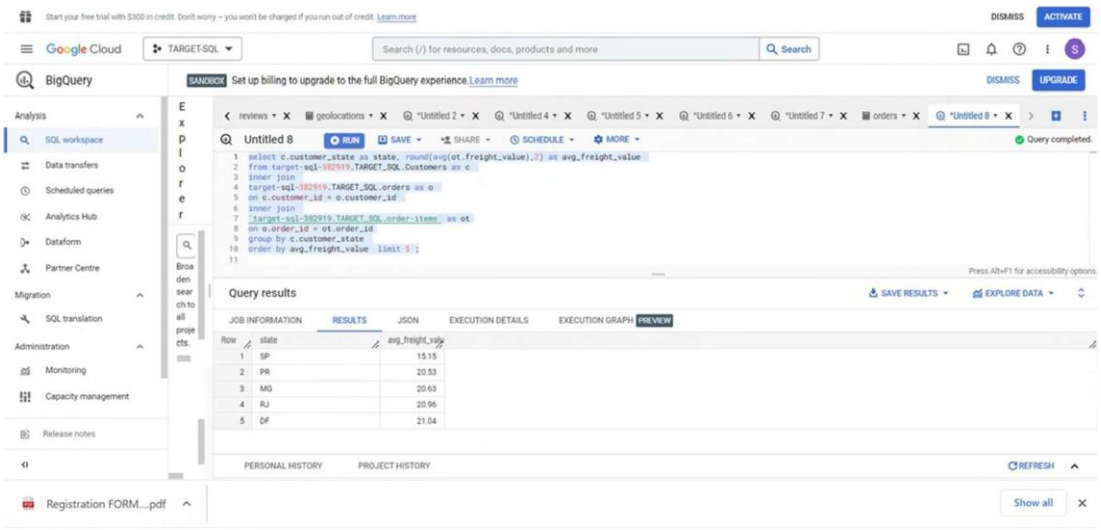
```
select c.customer_state as state, round(avg(ot.freight_value),2)
as avg_freight_value
from target-sql-382919.TARGET_SQL.Customers as c
inner join
target-sql-382919.TARGET_SQL.orders as o
on c.customer_id = o.customer_id
inner join
`target-sql-382919.TARGET_SQL.order-items` as ot
on o.order_id = ot.order_id
group by c.customer_state
order by avg_freight_value desc limit 5;
```



Query-

Top 5 state with lowest average freight value

```
select c.customer_state as state, round(avg(ot.freight_value),2)
as avg_freight_value
from target-sql-382919.TARGET_SQL.Customers as c
inner join
target-sql-382919.TARGET_SQL.orders as o
on c.customer_id = o.customer_id
inner join
`target-sql-382919.TARGET_SQL.order-items` as ot
on o.order_id = ot.order_id
group by c.customer_state
order by avg_freight_value limit 5;
```



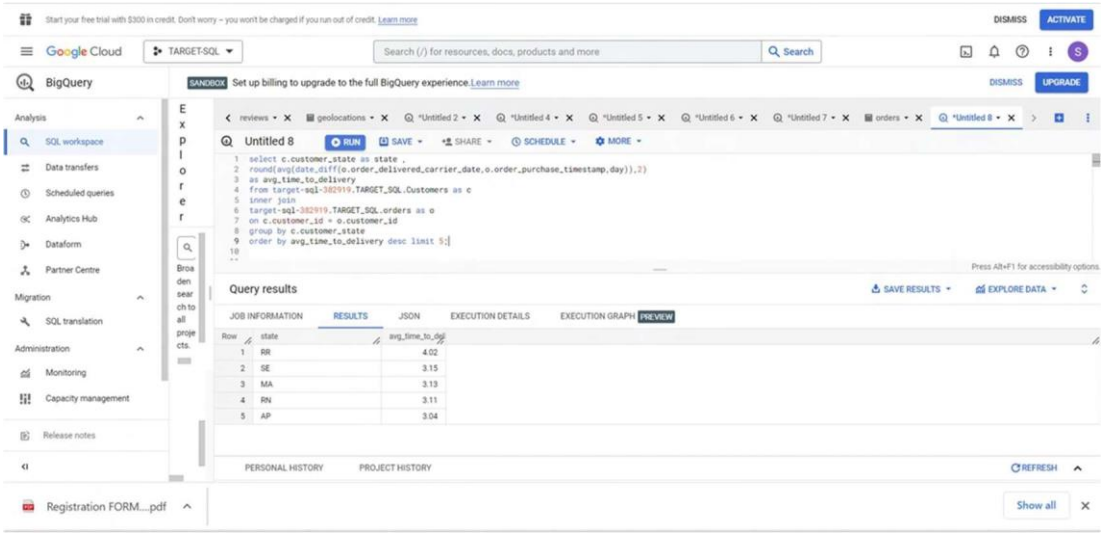
Insight- After analysing the data, I find that the top and bottom average freight value state is RR and SP.

5(e)- Top 5 states with highest/lowest average time to delivery.

Query-

Top 5 states with highest average time to delivery.

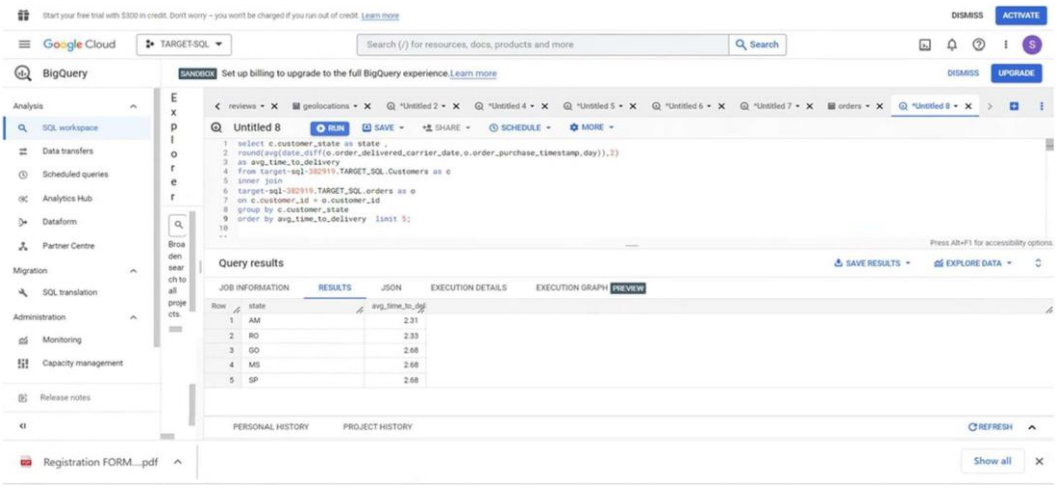
```
select c.customer_state as state,
round(avg(date_diff(o.order_delivered_carrier_date,
o.order_purchase_timestamp,day)),2)
as avg_time_to_delivery
from target-sql-382919.TARGET_SQL.Customers as c
inner join
target-sql-382919.TARGET_SQL.orders as o
on c.customer_id = o.customer_id
group by c.customer_state
order by avg_time_to_delivery desc limit 5;
```



Query-

Top 5 states with lowest average time to delivery.

```
select c.customer_state as state,
round(avg(date_diff(o.order_delivered_carrier_date,
o.order_purchase_timestamp,day)),2)
as avg_time_to_delivery
from target-sql-382919.TARGET_SQL.Customers as c
inner join
target-sql-382919.TARGET_SQL.orders as o
on c.customer_id = o.customer_id
group by c.customer_state
order by avg_time_to_delivery limit 5;
```



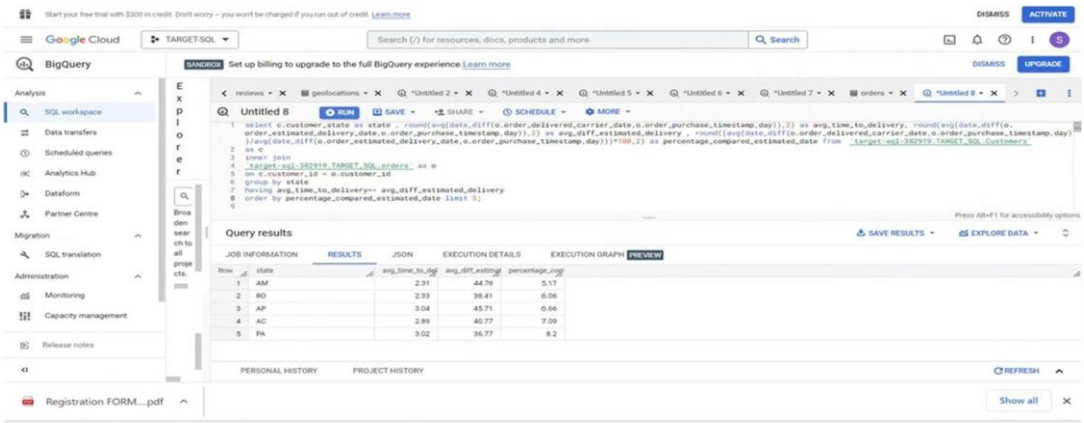
Insight - After analysing the problem statement, I found that the avg time to delivery is approx lies between 2 to 4 days.

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

Query-

Top 5 states where delivery is so fast as compared to estimated date.

```
select c.customer_state as state,
round(avg(date_diff(o.order_delivered_carrier_date,
o.order_purchase_timestamp, day)),2) as avg_time_to_delivery,
round(avg(date_diff(o.order_estimated_delivery_date,
o.order_purchase_timestamp, day)),2) as avg_diff_estimated_delivery,
round((avg(date_diff(o.order_delivered_carrier_date,
o.order_purchase_timestamp,day))/
avg(date_diff(o.order_estimated_delivery_date,
o.order_purchase_timestamp, day))) *100,2)
as percentage_compared_estimated_date
from `target-sql-382919.TARGET_SQL.Customers` as c
inner join
`target-sql-382919.TARGET_SQL.orders` as o
on c.customer_id = o.customer_id
group by state
having avg_time_to_delivery<= avg_diff_estimated_delivery
order by percentage_compared_estimated_date limit 5;
```



Query-

Top 5 states where delivery is not so fast as compared to estimated date.

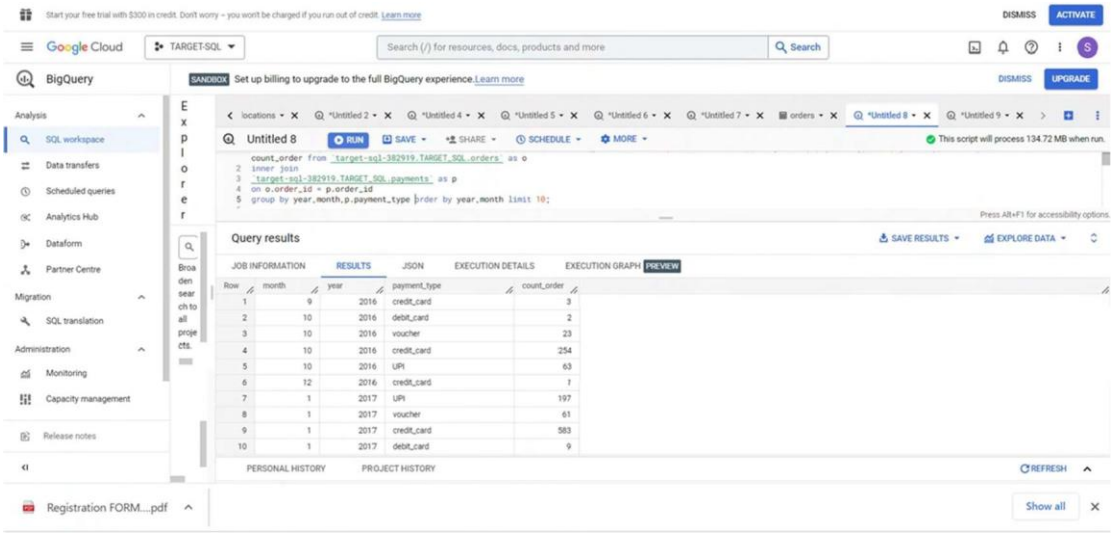
```
select c.customer_state as state,
round(avg(date_diff(o.order_delivered_carrier_date,
o.order_purchase_timestamp, day)),2) as avg_time_to_delivery,
round(avg(date_diff(o.order_estimated_delivery_date,
o.order_purchase_timestamp, day)),2) as avg_diff_estimated_delivery,
round((avg(date_diff(o.order_delivered_carrier_date,
o.order_purchase_timestamp, day))/
avg(date_diff(o.order_estimated_delivery_date,
o.order_purchase_timestamp, day))) *100,2)
as percentage_compared_estimated_date
from `target-sql-382919.TARGET_SQL.Customers` as c
inner join
`target-sql-382919.TARGET_SQL.orders` as o
on c.customer_id = o.customer_id
group by state
having avg_time_to_delivery<= avg_diff_estimated_delivery
order by percentage_compared_estimated_date limit 5;
```

Insight- After analysing the data, I found that the estimated days for delivery is too long as compared to delivery date it is 5 to 15 percentage of estimated day for every state.

6 (a) Month over Month count of orders for different payment types.

Query-

```
select extract(month from o.order_purchase_timestamp) as month,
extract(year from order_purchase_timestamp) as year, p.payment_type
as payment_type,count(o.order_id) as count_order
from `target-sql-382919.TARGET_SQL.orders` as o
inner join
`target-sql-382919.TARGET_SQL.payments` as p
on o.order_id = p.order_id
group by year, month, p.payment_type
order by year, month;
```

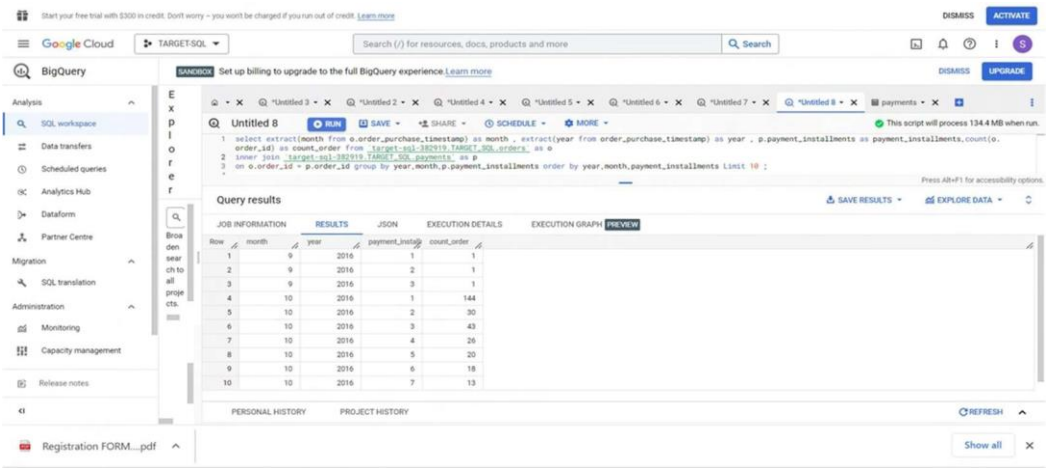


Insight- After calculating the month-to-month count of order for different payment types, I found that every month maximum payment done by credit card.

6 (b) Count of orders based on the no. of payment instalments.

Query-

```
select extract(month from o.order_purchase_timestamp) as month,
extract(year from order_purchase_timestamp) as year,
p.payment_installments as payment_installments,
count(o.order_id) as count_order
from `target-sql-382919.TARGET_SQL.orders` as o
inner join
`target-sql-382919.TARGET_SQL.payments` as p
on o.order_id = p.order_id
group by year,month,p.payment_installments order by year,month;
```



Insight - After calculating the month-to-month count of order based on payment instalments then I found that every month maximum order done by onetime payment.

Recommendations- After analysing all the problem statements I want to give some suggestion to the company so that company will grow more are following below.

1. Give more exciting offer like reward point, coupon code to the customer who gives more order so that customer will attract more and placed moreorders.
2. Customer who gives lesser order, give them offers like Easy EMI payment, no cost EMI, pay later so that they will place more orders.
3. I observe in the season of winter and Autumn order is less as compared to other season so give special offers like buy 2 get 1 free, offers on payment on product for this season so that sales will grow more on this season.
4. Try to minimise the difference between estimated delivery date and delivered date because estimated delivery time was very high as compared to delivered date.
5. Try to minimise the freight value as much as possible.