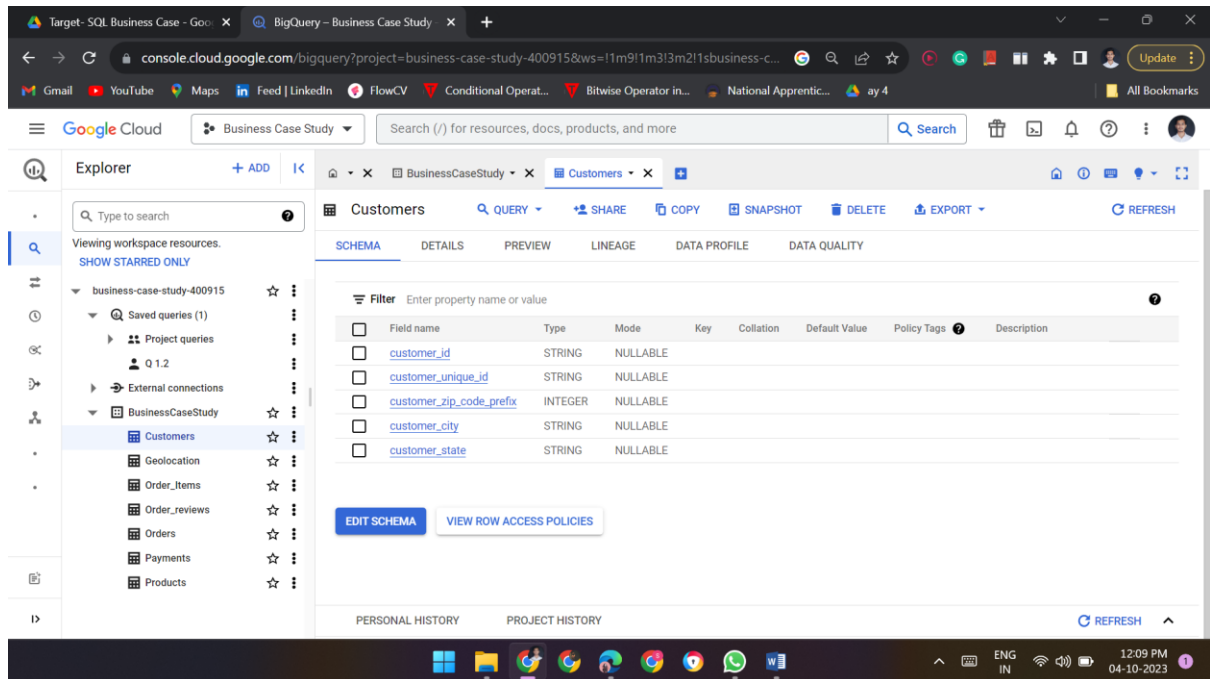


## Business case study by Piyush Aswale

### Case study topic: Target SQL

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

1.1Data type of all columns in the "customers" table. Solution: We can get to know the data type by clicking on the table name whose columns data type we want to know in bigquery.



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

Solution: `select min(order_purchase_timestamp) as startTime,  
max(order_purchase_timestamp) as endTime  
from `BusinessCaseStudy.Orders`;`

The time range between which the orders we placed is from **2016-09-04 21:15:19 UTC** to **2018-10-17 17:30:18 UTC**

The screenshot shows the Google Cloud BigQuery console interface. The Explorer panel on the left displays the project structure, including 'BusinessCaseStudy' and its tables: 'Customers', 'Geolocation', 'Order\_Items', 'Order\_reviews', 'Orders', 'Payments', and 'Products'. The main editor shows a SQL query labeled 'Q 1.2' with the following code:

```
-- Que 1.2
select min(order_purchase_timestamp) as startTime,
max(order_purchase_timestamp) as endTime
from `BusinessCaseStudy.Orders`;
```

The 'Query results' section shows the following table:

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

The bottom status bar indicates the time is 12:11 PM on 04-10-2023.

### 1.3 Count the Cities & States of customers who ordered during the given period.

Solution:

Query: `select`

```
count(distinct(c.customer_state)) as state,
count(distinct(c.customer_city)) as city
from `BusinessCaseStudy.Orders` o join `BusinessCaseStudy.Customers` c on
c.customer_id=o.customer_id;
```

There were orders from 4119 cities and 27 states

The screenshot shows the Google Cloud BigQuery console interface. The Explorer panel on the left displays the project structure, including 'BusinessCaseStudy' and its tables: 'Customers', 'Geolocation', 'Order\_Items', 'Order\_reviews', 'Orders', 'Payments', and 'Products'. The main editor shows a SQL query labeled 'Q 1.2' with the following code:

```
-- 1.3
select
count(distinct(c.customer_state)) as state,
count(distinct(c.customer_city)) as city
from `BusinessCaseStudy.Orders` o join `BusinessCaseStudy.Customers` c on c.customer_id=o.customer_id;
```

The 'Query results' section shows the following table:

Row	state	city
1	27	4119

The bottom status bar indicates the time is 12:13 PM on 04-10-2023.

## 2 In-depth Exploration: 2.1 Is there a growing trend in the no. of orders placed over the past years? Solution:

Query: 

```
select extract(year from order_purchase_timestamp) as year,
count(order_id)
from `BusinessCaseStudy.Orders`
group by 1
order by 1;
```

Explanation: As per the given data, there is only 3 months data from year 2016, 12 months data from year 2017 and 10 month data from 2018, There is a good amount of growth in the numbers of orders placed from year 2016 to 2017, and a significant growth from 2017 to 2018

The screenshot shows the Google Cloud BigQuery console interface. On the left, the Explorer pane displays the project 'business-case-study-400915' with various datasets like Customers, Geolocation, Order\_Items, Order\_reviews, Orders, Payments, and Products. The main editor shows a SQL query (Q 1.2) that extracts the year from the order\_purchase\_timestamp and counts the number of orders. The query results are displayed in a table with columns 'year' and 'count'. The results show 329 orders for 2016, 45101 for 2017, and 54011 for 2018. The console also shows the query execution details and a 'Query completed' status.

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

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

Query: 

```
select extract(year from order_purchase_timestamp) as year,
extract(month from order_purchase_timestamp) as month,
count(order_id) as Number_of_orders
from `target.orders` group by 1,2 order by 1,2
```

## 2.3 During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)

♣ 0-6 hrs : Dawn

♣ 7-12 hrs : Mornings

♣ 13-18 hrs : Afternoon

♣ 19-23 hrs : Night

Solution:

Query: with cte as(

```
select case
when
  extract(hour from order_purchase_timestamp) between 0 and 6
  then "Dawn"
when
  extract(hour from order_purchase_timestamp) between 7 and 12
  then "Mornings"
when
  extract(hour from order_purchase_timestamp) between 13 and 18
  then "Afternoon"
when
  extract(hour from order_purchase_timestamp) between 19 and 23
  then "Night"
end as time_of_Day
from `BusinessCaseStudy.Orders`)
select time_of_day, count(*) as no_Orders from cte
group by 1 ;
```

The screenshot displays the Google Cloud BigQuery console interface. The top navigation bar shows the 'Business Case Study' project. The left sidebar contains the 'Explorer' panel with a search bar and a list of workspace resources, including 'business-case-study-400915', 'Saved queries (1)', 'Project queries', 'External connections', and 'BusinessCaseStudy' (which includes Customers, Geolocation, Order\_Items, Order\_reviews, Orders, Payments, and Products). The main panel shows a SQL query (Q 1.2) with the following code:

```
with cte as(
select case
when
  extract(hour from order_purchase_timestamp) between 0 and 6
  then "Dawn"
when
  extract(hour from order_purchase_timestamp) between 7 and 12
  then "Mornings"
when
  extract(hour from order_purchase_timestamp) between 13 and 18
  then "Afternoon"
when
  extract(hour from order_purchase_timestamp) between 19 and 23
  then "Night"
end as time_of_Day
from `BusinessCaseStudy.Orders`)
select time_of_day, count(*) as no_Orders from cte
group by 1 ;
```

The query results are displayed in a table with the following data:

Row	time_of_day	no_Orders
1	Mornings	27733
2	Dawn	5242
3	Afternoon	38135
4	Night	28331

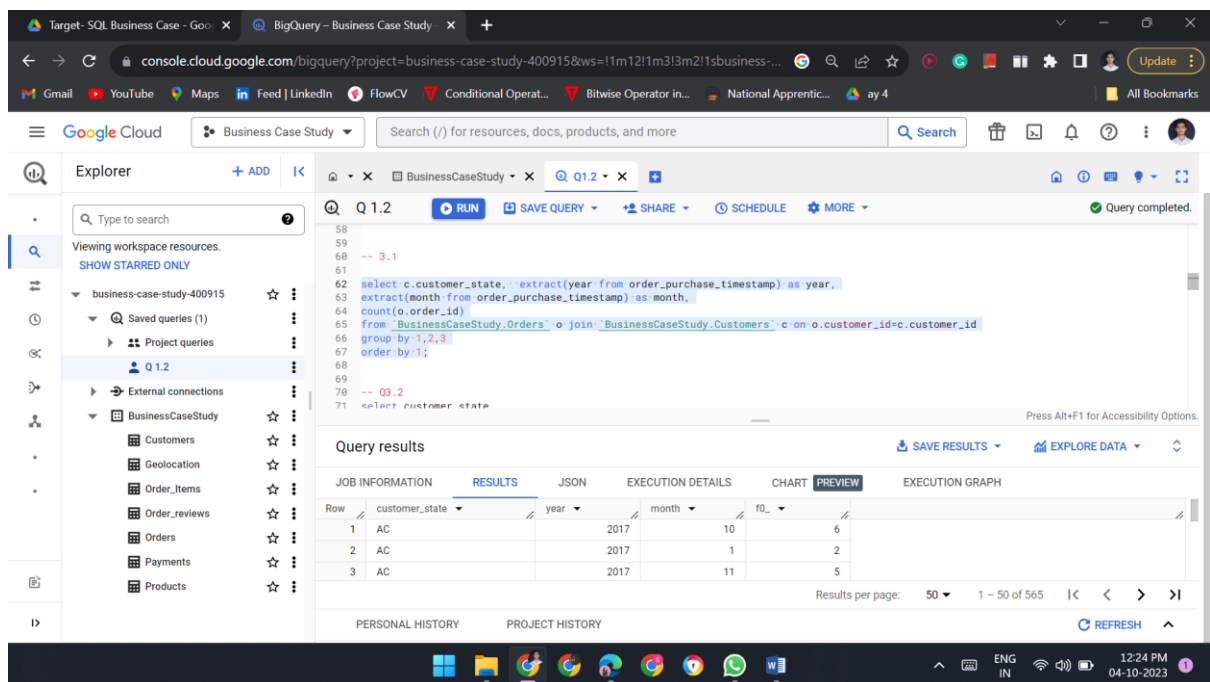
The bottom of the console shows the 'PERSONAL HISTORY' and 'PROJECT HISTORY' sections, along with a 'REFRESH' button. The system tray at the bottom indicates the time as 12:23 PM on 04-10-2023.

### Evolution of E-commerce orders in the Brazil region: 3.1 Get the month on month no. of orders placed in each state.

Solution:

Query: `select c.customer_state, extract(year from order_purchase_timestamp) as year, extract(month from order_purchase_timestamp) as month, count(o.order_id) from `BusinessCaseStudy.Orders` o join `BusinessCaseStudy.Customers` c on o.customer_id=c.customer_id group by 1,2,3 order by 1;`

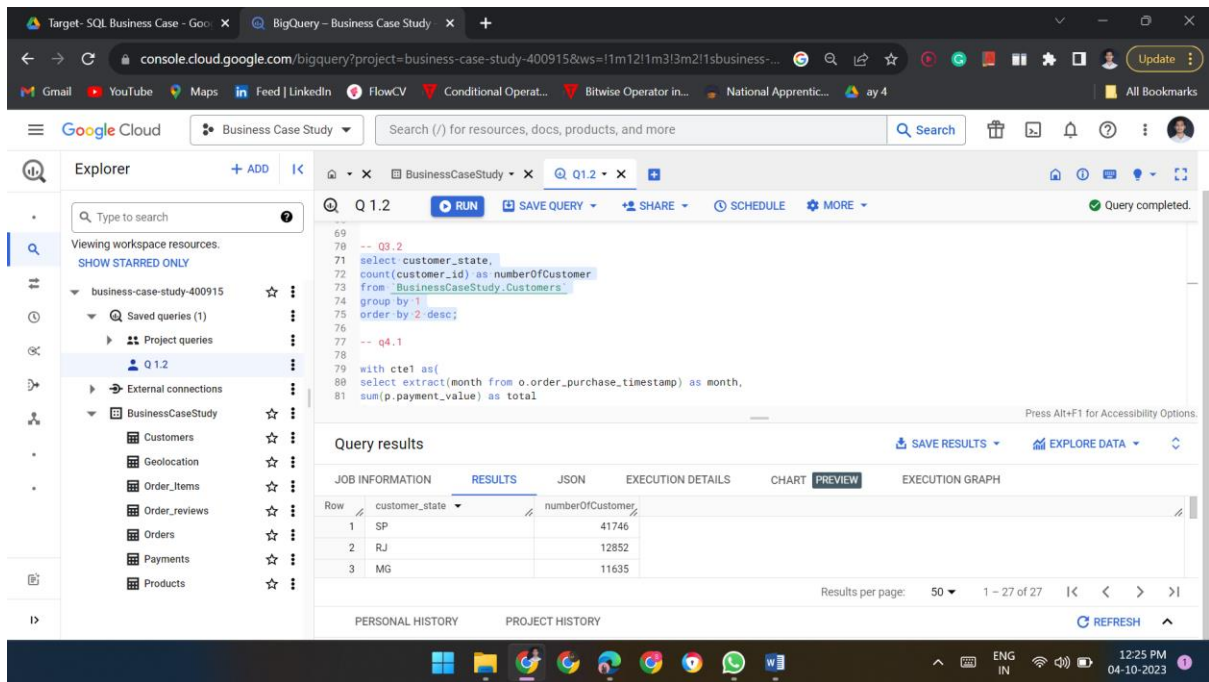
Explanation: Month on month orders placed data is as shown in the screenshot



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

Query: `select customer_state, count(customer_id) as numberOfCustomer from `BusinessCaseStudy.Customers` group by 1 order by 2 desc;`

Explanation: Distribution of the customers can be fetched using the above query with maximum customers from state SP with 41746 customer And least number of customers are from RR with only 46 customer



**4 Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight and others.**

**4.1 Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only). You can use the "payment\_value" column in the payments table to get the cost of orders.**

Solution:

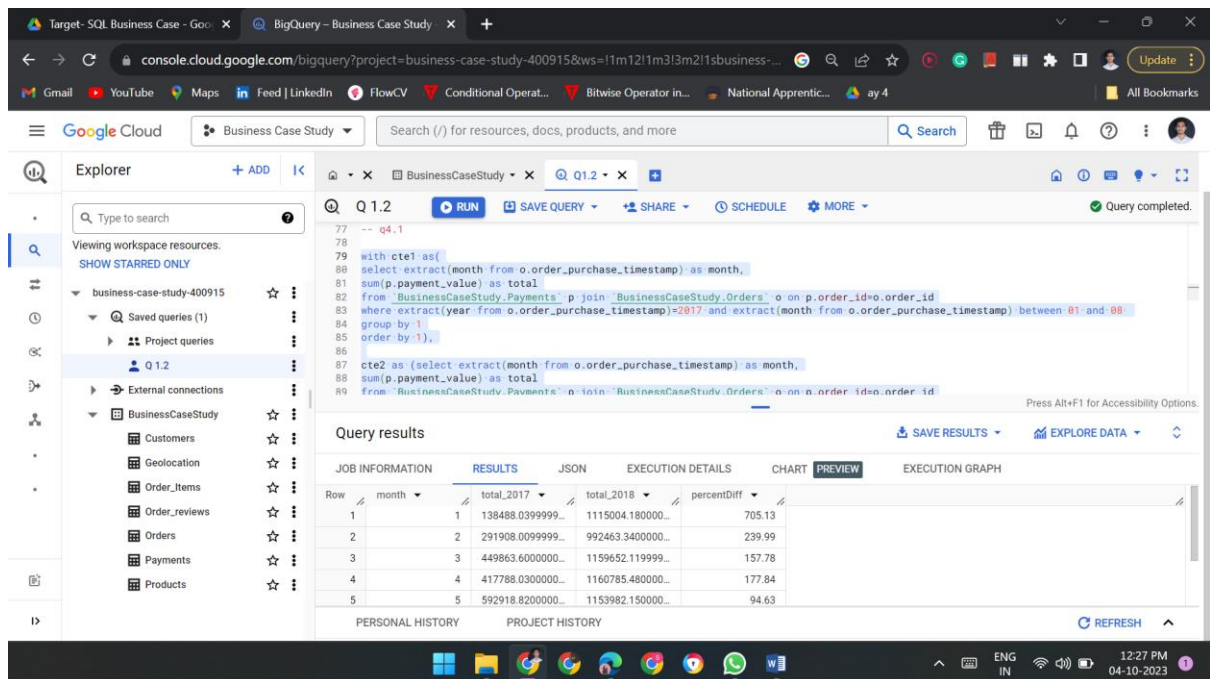
Query: with cte1 as(

```
select extract(month from o.order_purchase_timestamp) as month,
sum(p.payment_value) as total
from `BusinessCaseStudy.Payments` p join `BusinessCaseStudy.Orders` o on
p.order_id=o.order_id
where extract(year from o.order_purchase_timestamp)=2017 and extract(month from
o.order_purchase_timestamp) between 01 and 08
group by 1
order by 1),
```

```
cte2 as (select extract(month from o.order_purchase_timestamp) as month,
sum(p.payment_value) as total
from `BusinessCaseStudy.Payments` p join `BusinessCaseStudy.Orders` o on
p.order_id=o.order_id
where extract(year from o.order_purchase_timestamp)=2018 and extract(month from
o.order_purchase_timestamp) between 01 and 08
group by 1
order by 1)
```

```
select c1.month,
c1.total as total_2017, c2.total as total_2018, round(((c2.total-c1.total)/c1.total)*100,2)
as percentDiff
from cte1 c1 join cte2 c2 on c1.month=c2.month
order by 1;
```

Explanation: There is a good percentage of growth in the cost of orders from year 2017 to 2018 monthly, with highest in January month 700%



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

Query: `select c.customer_state, round(sum(p.payment_value),2) as total,round((sum(p.payment_value)/count(o.order_id)),2) as average`  
`from `BusinessCaseStudy.Orders` o join `BusinessCaseStudy.Customers` c on`  
`o.customer_id=c.customer_id`  
`join `BusinessCaseStudy.Payments` p on o.order_id=p.order_id`  
`group by 1`  
`order by 1;`

Explanation: The total and average value of order price of each state can be fetched using above query



The screenshot displays the Google Cloud BigQuery interface. On the left, the Explorer panel shows the project 'business-case-study-400915' with various tables like Customers, Orders, and Payments. The main panel shows a query labeled 'Q 1.2' that has been executed successfully. The query text is as follows:

```

99 -- 4.2
100
101 select c.customer_state, round(sum(p.payment_value),2) as total, round((sum(p.payment_value)/count(o.order_id)),2) as average
102 from `BusinessCaseStudy.Orders` o join `BusinessCaseStudy.Customers` c on o.customer_id=c.customer_id
103 join `BusinessCaseStudy.Payments` p on o.order_id=p.order_id
104 group by 1
105 order by 1;

```

The 'Query results' section shows a table with the following data:

Row	customer_state	total	average
1	AC	19680.62	234.29
2	AL	96962.06	227.08
3	AM	27966.93	181.6
4	AP	16262.8	232.33
5	BA	616645.82	170.82
6	CE	279464.03	199.9
7	DF	355141.08	161.13

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

Query: -- 4.3

```

select c.customer_state, round(sum(oi.freight_value),2) as
total, round(avg(oi.freight_value),2) as average
from `BusinessCaseStudy.Orders` o join `BusinessCaseStudy.Customers` c on
o.customer_id=c.customer_id
join `BusinessCaseStudy.Order_Items` oi on o.order_id=oi.order_id
group by 1
order by 2 desc;

```

Explanation: The total and average value of order freight of each state can be fetched using above query.



The screenshot shows the Google Cloud BigQuery console interface. On the left is the Explorer pane showing the project 'business-case-study-400915' and its datasets: Customers, Geolocation, Order\_Items, Order\_reviews, Orders, Payments, and Products. The main editor displays a SQL query (Q 1.2) that calculates the total and average freight value for each customer state. The query results are shown in a table with 7 rows, sorted by total freight value in descending order.

**Query:**

```

187 -- 4.3
188 select c.customer_state, round(sum(o1.freight_value),2) as total, round(avg(o1.freight_value),2) as average
189 from `BusinessCaseStudy.Orders` o join `BusinessCaseStudy.Customers` c on o.customer_id=c.customer_id
190 join `BusinessCaseStudy.Order_Items` oi on o.order_id=oi.order_id
191 group by 1
192 order by 2 desc;
193

```

**Query results:**

Row	customer_state	total	average
1	SP	718723.07	15.15
2	RJ	305589.31	20.96
3	MG	270853.46	20.63
4	RS	135522.74	21.74
5	PR	117851.68	20.53
6	BA	100156.68	26.36
7	SC	89660.26	21.47

Analysis based on sales, freight and delivery time.

5.1 Find the no. of days taken to deliver each order from the order's purchase date as delivery time. Also, calculate the difference (in days) between the estimated & actual delivery date of an order. Do this in a single query. You can calculate the delivery time and the difference between the estimated & actual delivery date using the given formula:

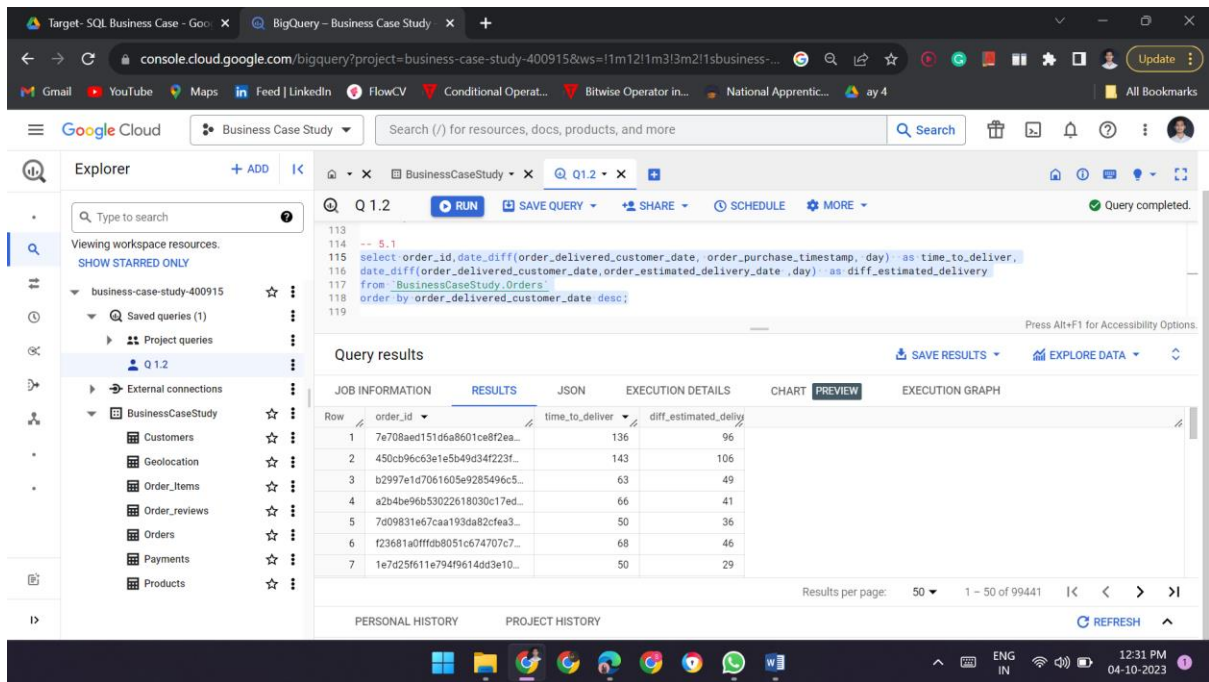
i. **time\_to\_deliver** = order\_delivered\_customer\_date - order\_purchase\_timestamp

ii. **diff\_estimated\_delivery** = order\_estimated\_delivery\_date -  
order\_delivered\_customer\_date

Query: `select order_id, date_diff(order_delivered_customer_date, order_purchase_timestamp, day) as time_to_deliver,`

`date_diff(order_delivered_customer_date, order_estimated_delivery_date, day) as  
diff_estimated_delivery  
from `BusinessCaseStudy.Orders`  
order by order_delivered_customer_date desc;`

Explanation: The required data about no. of days taken to deliver each order from the order's purchase date and the difference (in days) between the estimated & actual delivery date of an order is as shown below



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

Solution: Query for Highest value:

```
with cte as (select avg(oi.freight_value) as averageValue, c.customer_state
from `BusinessCaseStudy.Orders` o join `BusinessCaseStudy.Customers` c on
o.customer_id=c.customer_id join
`BusinessCaseStudy.Order_Items` oi on o.order_id=oi.order_id
group by c.customer_state)
```

```
select customer_state, averageValue from cte
order by averageValue
limit 5 ;
```

Explanation: The top 5 states with highest freight value are RR,PB,RO,AC,PI respectively

Solution: Query for Lowest value:

```
with cte as (select avg(oi.freight_value) as averageValue, c.customer_state
from `BusinessCaseStudy.Orders` o join `BusinessCaseStudy.Customers` c on
o.customer_id=c.customer_id join
`BusinessCaseStudy.Order_Items` oi on o.order_id=oi.order_id
group by c.customer_state)
select customer_state, averageValue from cte
order by averageValue desc
limit 5 ;
```

Explanation: The lowest 5 states with least freight value are SP,PR,MG,RJ,DF respectively

The screenshot shows the Google Cloud BigQuery console interface. The Explorer pane on the left displays the project structure for 'business-case-study-400915'. The main editor shows a SQL query (Q 1.2) that calculates the average delivery time for each state. The query results are displayed in a table with 5 rows, ordered by average value in descending order.

```

118 order by order_delivered_customer_date desc;
119
120 -- 5.2
121 with cte as (select avg(oi.freight_value) as averageValue, c.customer_state
122 from `BusinessCaseStudy.Orders` o join `BusinessCaseStudy.Customers` c on o.customer_id=c.customer_id join
123 `BusinessCaseStudy.Order_Items` oi on o.order_id=oi.order_id
124 group by c.customer_state)
125
126 select customer_state, averageValue from cte
127 order by averageValue
128 limit 5 ;

```

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

The screenshot shows the Google Cloud BigQuery console interface. The Explorer pane on the left displays the project structure for 'business-case-study-400915'. The main editor shows a SQL query (Q 1.2) that calculates the average delivery time for each state. The query results are displayed in a table with 5 rows, ordered by average value in descending order.

```

128 limit 5 ;
129
130 with cte as (select avg(oi.freight_value) as averageValue, c.customer_state
131 from `BusinessCaseStudy.Orders` o join `BusinessCaseStudy.Customers` c on o.customer_id=c.customer_id join
132 `BusinessCaseStudy.Order_Items` oi on o.order_id=oi.order_id
133 group by c.customer_state)
134 select customer_state, averageValue from cte
135 order by averageValue desc
136 limit 5 ;
137

```

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

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

Solution: Query for highest delivery time: `with cte as`

```

(select avg(date_diff(o.order_delivered_customer_date, o.order_purchase_timestamp, day)) as
avgTime, c.customer_state
from `BusinessCaseStudy.Customers` c join `BusinessCaseStudy.Orders` o on
c.customer_id=o.customer_id
group by c.customer_state)
select customer_state, avgTime from cte
order by 2 desc limit 5;

```

Explanation: Top 5 states with highest average delivery time are RR, AP, AM, AL, PA respectively.

Query for least delivery time:

```
with cte as
(select avg(date_diff(o.order_delivered_customer_date, o.order_purchase_timestamp, day)) as
avgTime, c.customer_state
from `BusinessCaseStudy.Customers` c join `BusinessCaseStudy.Orders` o on
c.customer_id=o.customer_id
group by c.customer_state)

select customer_state, avgTime from cte
order by 2 limit 5;
```

Explanation: Top 5 states with least average delivery time are SP,PR,MG,DF,SC respectively

The screenshot displays the Google Cloud BigQuery console interface. The left sidebar shows the Explorer view with the project 'business-case-study-400915' and various datasets like Customers, Orders, and Payments. The main panel shows a SQL query (Q1.2) that calculates the average delivery time for different customer states. The query results are displayed in a table with 5 rows, ordered by average time in descending order.

Query results table:

Row	customer_state	avgTime
1	RR	28.97560975609...
2	AP	26.73134328358...
3	AM	25.98620689655...
4	AL	24.04030226700...
5	PA	23.31606765327...

The screenshot shows the Google Cloud BigQuery console interface. On the left is the Explorer pane with a search bar and a tree view of workspace resources. The main editor displays a SQL query (Q 1.2) that calculates the average time difference between order delivery and purchase for different customer states. Below the query editor, the 'Query results' section is active, showing a table with 5 rows of data. The table has columns for 'customer\_state' and 'avgTime'. The results are sorted by 'avgTime' in ascending order, showing the top 5 states with the fastest delivery times.

**Query:**

```

147 with cte as
148 (select avg(date_diff(o.order_delivered_customer_date, o.order_purchase_timestamp, day)) as avgTime, c.customer_state
149 from `BusinessCaseStudy.Customers` c join `BusinessCaseStudy.Orders` o on c.customer_id=o.customer_id
150 group by c.customer_state)
151
152 select customer_state, avgTime from cte
153 order by 2 limit 5;
154
155 -- 5.4

```

**Query results:**

Row	customer_state	avgTime
1	SP	8.298061489072...
2	PR	11.52671135486...
3	MG	11.54381329810...
4	DF	12.50913461538...
5	SC	14.47956019171...

**5.4 Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery. You can use the difference between the averages of actual & estimated delivery date to figure out how fast the delivery was for each state.**

Solution: Query:

```

select round(avg(date_diff(o.order_delivered_customer_date,
o.order_purchase_timestamp, day)), 2) as avgTime, c.customer_state
from `BusinessCaseStudy.Customers` c join `BusinessCaseStudy.Orders` o on
c.customer_id=o.customer_id
group by c.customer_state
order by 1;

```

Explanation: The top 5 states where the order delivery is really fast as compared to estimated date of delivery are AL, MA, SE, ES, BA

The screenshot shows the Google Cloud BigQuery console interface. On the left, the Explorer pane displays the project structure for 'business-case-study-400915', including tables like Customers, Geolocation, Order\_Items, Order\_reviews, Orders, Payments, and Products. The main editor shows a SQL query (Q 1.2) that calculates the average time difference between order delivery and purchase for different customer states. The query results are displayed in a table with columns 'avgTime' and 'customer\_state'.

**Query:**

```
select round(avg(date_diff(o.order_delivered_customer_date, o.order_purchase_timestamp, day)), 2) as avgTime, c.customer_state
from `BusinessCaseStudy.Customers` c join `BusinessCaseStudy.Orders` o on c.customer_id=o.customer_id
group by c.customer_state
order by 1;
```

**Query results:**

Row	avgTime	customer_state
1	8.3	SP
2	11.53	PR
3	11.54	IMG
4	12.51	DF
5	14.48	SC
6	14.82	RS

## 6. Analysis based on the payments: 6.1 Find the month on month no. of orders placed using different payment types.

Solution: Query:

```
select extract(month from o.order_purchase_timestamp) as month,
extract(year from o.order_purchase_timestamp) as year, p.payment_type, count(p.order_id) as
orderCount
from `BusinessCaseStudy.Payments` p join `BusinessCaseStudy.Orders` o on
p.order_id=o.order_id
group by 1,2,3
order by 2,1;
```

Explanation: The above query gives us the month on month no. of orders placed using different payments type



Query results

Row	month	year	payment_type	orderCount
1	9	2016	credit_card	3
2	10	2016	credit_card	254
3	10	2016	voucher	23
4	10	2016	debit_card	2
5	10	2016	UPI	63
6	12	2016	credit_card	1

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

Solution: Query: `select p.payment_installments, count(o.order_id) as numberOfOrder`

`from `BusinessCaseStudy.Payments` p join `BusinessCaseStudy.Orders` o on  
p.order_id=o.order_id  
where p.payment_installments != 0  
group by 1`

The above query gives us no. of orders placed on the basis of the payment installments that have been paid.

Query results

Row	payment_installments	numberOfOrder
1	1	52546
2	2	12413
3	3	10461
4	4	7098
5	5	5239
6	6	3920