

Business Case: Target SQL

Name: D A Santhosh

Q1 /* Data type of all columns in the "customers" table.*/

SELECT

table_name,

column_name,

data_type

FROM `project.INFORMATION_SCHEMA.COLUMNS`

WHERE table_name = 'customers'

The screenshot shows the Google Cloud BigQuery console interface. The query editor on the right contains the following SQL code:

```
1 /* Data type of all columns in the "customers" table.*/
2
3
4 SELECT
5   table_name,
6   column_name,
7   data_type
8 FROM `project.INFORMATION_SCHEMA.COLUMNS`
9 WHERE table_name='customers'
```

The query has been executed successfully, as indicated by the "Query completed" status. The results are displayed in a table with the following columns: table_name, column_name, and data_type. The results show five rows of data for the 'customers' table.

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

Insights and recommendation: Here we can conclude that there are different types of data in

the table

including alphabets which is string data type and numeric data which is int type

hence the table contains both numeric and alphabetic data types.

```
/* Get the time range between which the orders were placed.*/
```

SELECT

```
MIN(order_purchase_timestamp) AS start_date,
```

```
MAX(order_purchase_timestamp) AS end_date
```

```
FROM `project.orders`
```

The screenshot displays the Google Cloud BigQuery console interface. On the left, the 'Explorer' pane shows the project hierarchy for 'target-389500', including datasets like 'customers', 'geolocation', 'order_items', 'order_reviews', 'orders', 'payments', 'products', and 'sellers'. The main editor area shows a SQL query titled 'Untitled' with the following code:

```
/* Get the time range between which the orders were placed.*/  
  
SELECT  
  MIN(order_purchase_timestamp) AS start_date,  
  MAX(order_purchase_timestamp) AS end_date  
FROM `project.orders`  
  
/* Count the number of Cities and States in our dataset.*/
```

Below the query editor, the 'Query results' section shows a table with one row of data:

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

The bottom of the screen shows a Windows taskbar with the date and time as 00:32 on 19-06-2023.

Insights and recommendation: The time range between which orders were mostly placed was between

2016-09-04 at 21:15:19 to

2018-10-17 at 17:30:18 , hence we can conclude that the customers ordered items the most between these two years .

/ Count the number of Cities and States in our dataset.*/*

SELECT

COUNT(DISTINCT(geolocation_city)) AS number_of_cities,

COUNT(DISTINCT(geolocation_state)) AS number_of_state

FROM `project.geolocation`

The screenshot shows the Google Cloud BigQuery console interface. At the top, there's a navigation bar with tabs for 'Business Case: Target SQL - Proj...', 'Business Case: Target SQL Intro...', and 'Query results - BigQuery - Target...'. Below the navigation bar, there's a toolbar with buttons for 'RUN', 'SAVE', 'SHARE', 'SCHEDULE', and 'MORE'. The main area displays a SQL query in a text editor. The query is as follows:

```
18
19 /* Count the number of Cities and States in our dataset.*/
20
21 SELECT
22   COUNT(DISTINCT(geolocation_city)) AS number_of_cities,
23   COUNT(DISTINCT(geolocation_state)) AS number_of_state
24 FROM `project.geolocation`
25
26
27 /* Is there a growing trend in the no. of orders placed over the past years?*/
```

Below the query editor, there's a section titled 'Query results' with a 'SAVE RESULTS' button. Underneath, there's a table with the following data:

Row	number_of_cities	number_of_state
1	8011	27

The bottom of the screenshot shows a Windows taskbar with various icons and a system tray indicating the date and time as 19-06-2023 00:34.

Insights and recommendation: Hence from the above query we can conclude that there are around 8011 cities and 27 states

Q2 /* Is there a growing trend in the no. of orders placed over the past years?*/

SELECT

EXTRACT(Year from order_purchase_timestamp) AS order_year,

COUNT(*) AS order_count

FROM `project.orders`

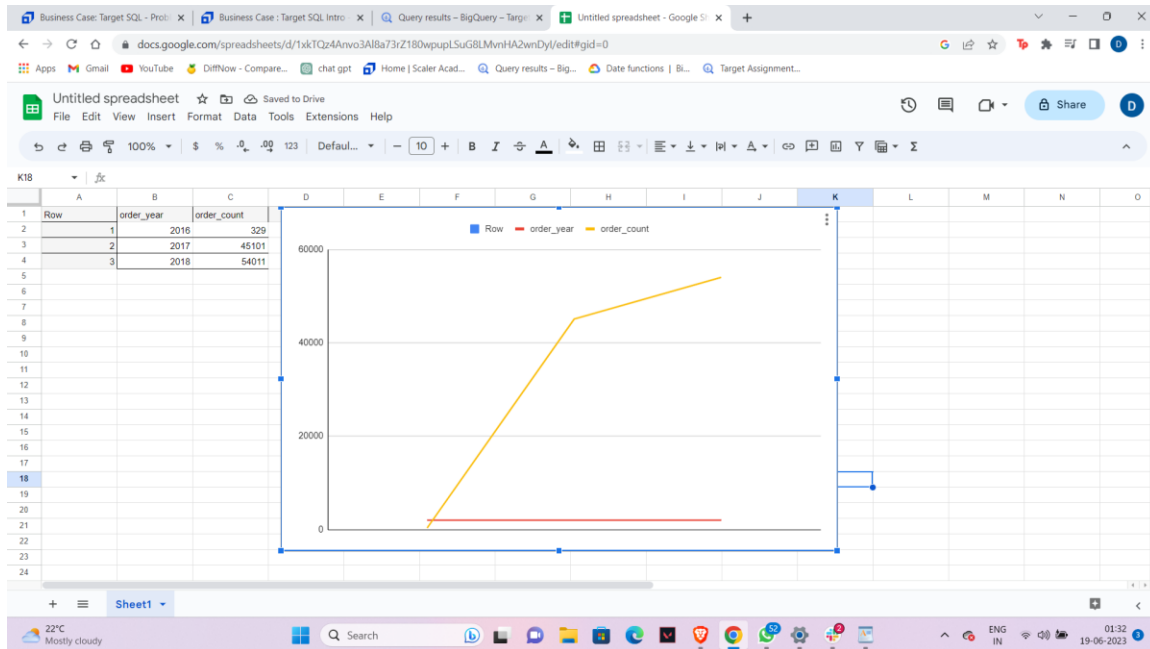
GROUP BY order_year

ORDER BY order_year;

The screenshot shows the Google Cloud BigQuery console interface. At the top, there's a browser tab for 'Business Case: Target SQL - Proj...' and another for 'Business Case: Target SQL Intro...'. The main area displays a SQL query in a text editor, which is the same query as provided in the previous blocks. Below the editor, the 'Query results' section is visible, showing a table with three rows of data. The table has columns for 'order_year' and 'order_count'. The results show a significant increase in orders from 2016 to 2017, and a more gradual increase from 2017 to 2018.

Row	order_year	order_count
1	2016	329
2	2017	45101
3	2018	54011

Insights and recommendation: There is Obviously a growing Trend as the number of orders has increased rapidly from 2016 to 2017 from 329 orders to a whopping 45101 orders and a sustainable increase between 2017 and 2018 from 45101 to 54011



The graph also shows an increase in the trend

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

SELECT

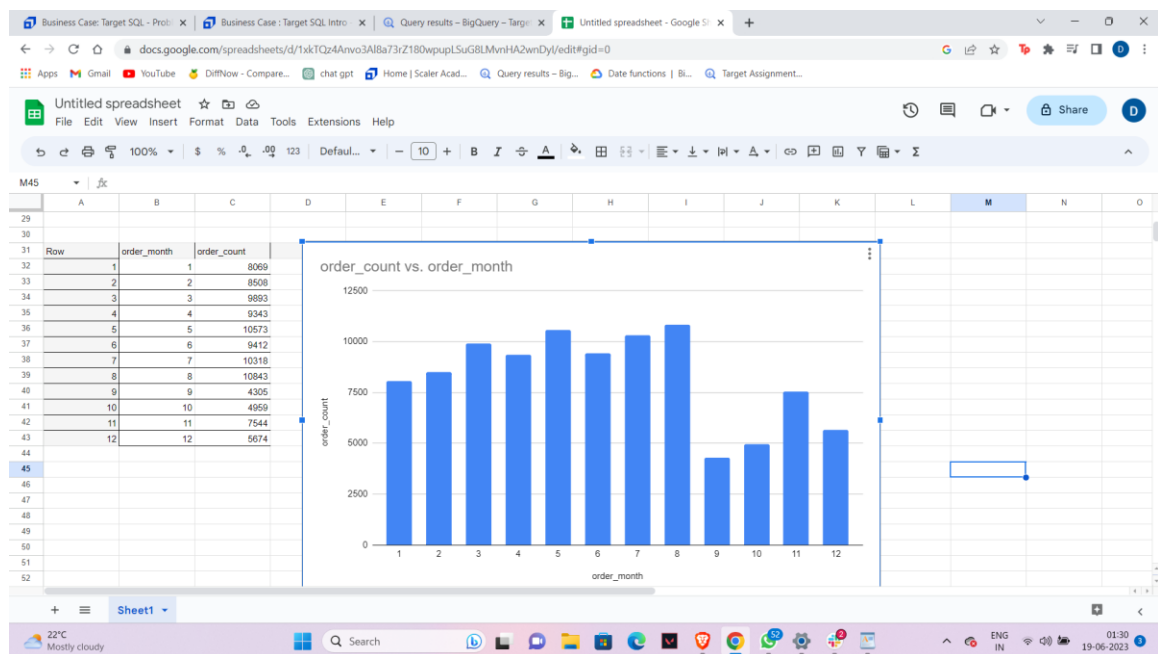
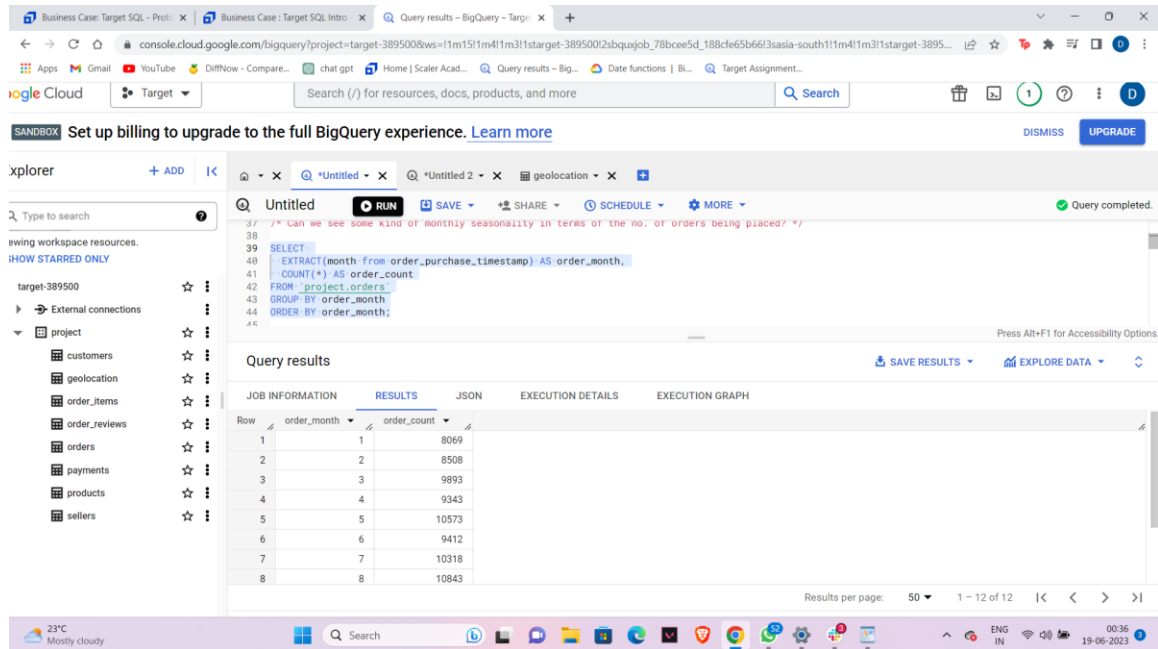
EXTRACT(month from order_purchase_timestamp) AS order_month,

COUNT(*) AS order_count

FROM `project.orders`

GROUP BY order_month

ORDER BY order_month;



Insights and recommendation: we can clearly observe that the sales started off well during the beginning of the year and hit its peak during the 8th month and showed a sudden fall during the ninth month and gradually picked up during the 11th month but dropped again during the last month.

So giving some discounts coupons or having a sale during the months in the year end would stabilize the sales throughout the year.

/* 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 */

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'

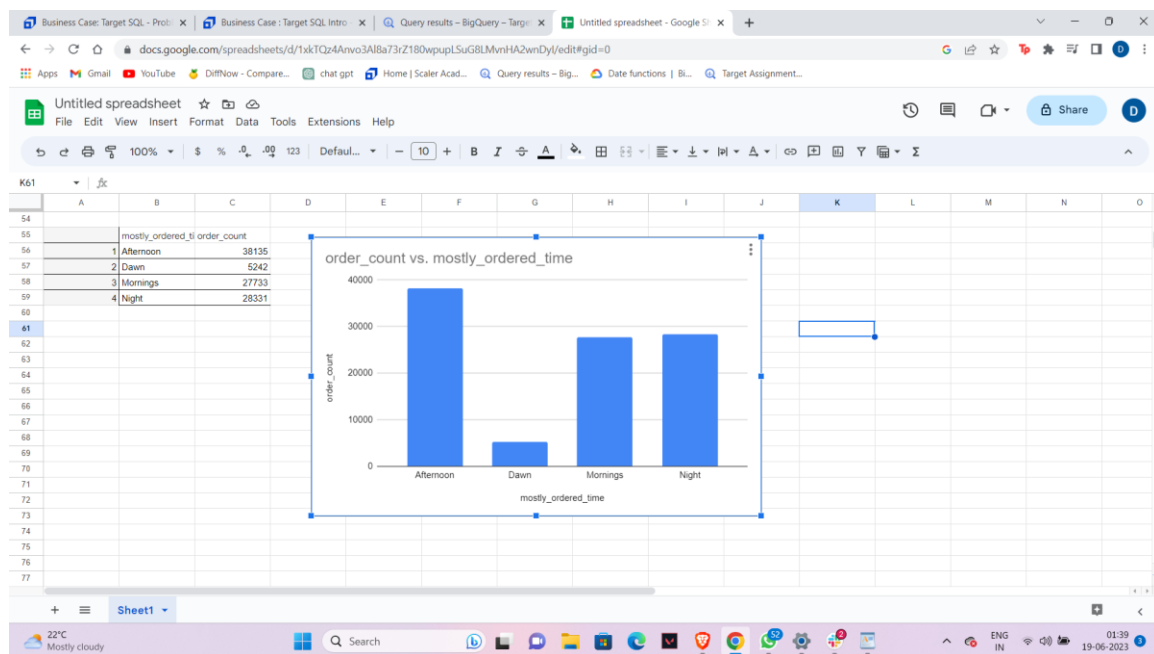
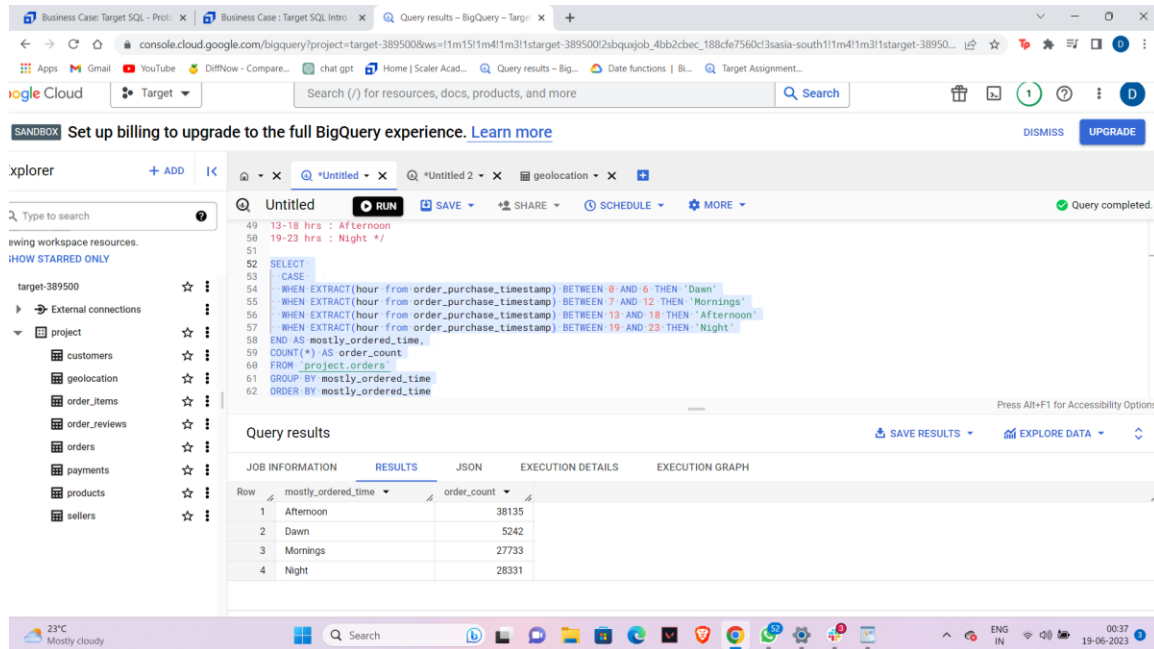
END AS mostly_ordered_time,

COUNT(*) AS order_count

FROM `project.orders`

GROUP BY mostly_ordered_time

ORDER BY mostly_ordered_time



Insights and recommendation: Brazilian customers mostly ordered during the afternoon and good number of orders during the morning and night too, the number of orders is the least during the dawn

so bringing up some promotional events or flash sales during the dawn would increase the number of orders placed during the dawn.

Q3 /* Get the month on month no. of orders placed in each state.*/

SELECT

EXTRACT(month from order_purchase_timestamp) AS Month,

customer_state,

COUNT(*) AS order_count

FROM `project.orders` o

INNER JOIN `project.customers` c ON o.customer_id = c.customer_id

GROUP BY Month, customer_state

ORDER BY Month, customer_state

The screenshot shows the Google Cloud BigQuery console interface. The query editor on the left contains the following SQL code:

```
/* Get the month on month no. of orders placed in each state.*/  
  
SELECT  
  EXTRACT(month from order_purchase_timestamp) AS Month,  
  customer_state,  
  COUNT(*) AS order_count  
FROM `project.orders` o  
INNER JOIN `project.customers` c ON o.customer_id = c.customer_id  
GROUP BY Month, customer_state  
ORDER BY Month, customer_state
```

The query results are displayed in a table with the following columns: Row, Month, customer_state, and order_count. The results show data for the month of January (1) across six states: AC, AL, AM, AP, BA, and CE.

Row	Month	customer_state	order_count
1	1	AC	8
2	1	AL	39
3	1	AM	12
4	1	AP	11
5	1	BA	264
6	1	CE	99

Insights and recommendation: Here by bringing the month on month orders placed by customers in different states we can observe state wise sales sorted in a particular month and

push business accordingly as per requirements and push by advertising more during the times the business is low in a particular state.

/ How are the customers distributed across all the states? */*

SELECT

customer_state,

COUNT(DISTINCT customer_id) AS customer_count

FROM `project.customers`

GROUP BY customer_state

ORDER BY customer_count DESC;

The screenshot shows the Google BigQuery console interface. At the top, there's a navigation bar with tabs for 'Business Case: Target SQL - Proj...', 'Business Case: Target SQL Intro...', and 'Query results - BigQuery - Targ...'. Below the navigation bar, there's a banner for 'p billing to upgrade to the full BigQuery experience. Learn more' with 'DISMISS' and 'UPGRADE' buttons. The main area shows a query editor with the following SQL code:

```
/* How are the customers distributed across all the states? */  
SELECT  
  customer_state,  
  COUNT(DISTINCT customer_id) AS customer_count  
FROM `project.customers`  
GROUP BY customer_state  
ORDER BY customer_count DESC;
```

Below the query editor, the 'Query results' section is displayed. It includes a table with the following data:

Row	customer_state	customer_count
1	SP	41746
2	RJ	12852
3	MG	11635
4	RS	5466
5	PR	5045
6	SC	3637
7	BA	3380
8	NC	2140

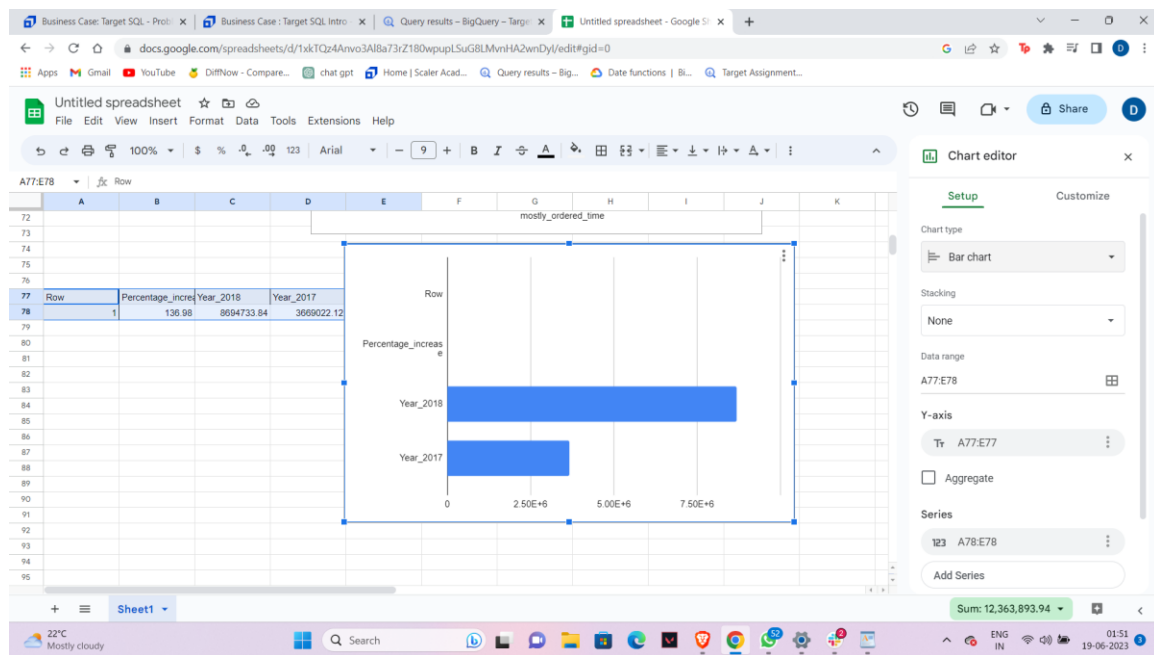
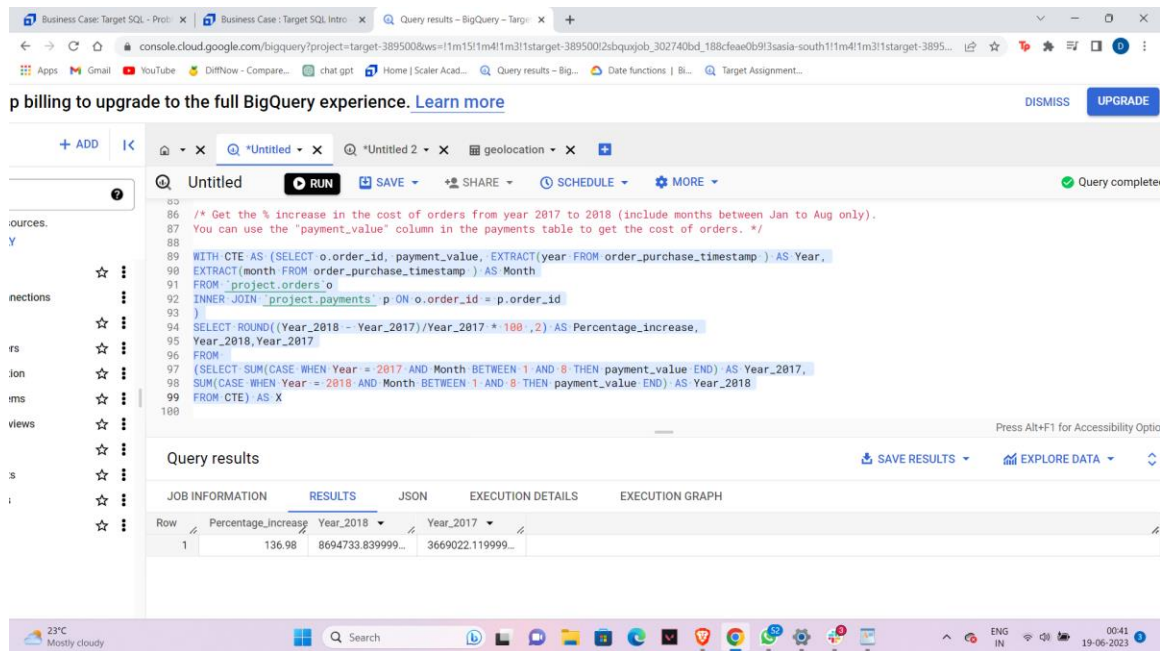
The bottom of the screenshot shows a Windows taskbar with various application icons and a system tray displaying the date and time as 00:40 on 19-06-2023.

Insights and recommendation: Here we get to see a data of the number of customers according to the states, which will in turn help us stabilize the positives done to get maximum sales in the states where there is more business and work on where we may be going wrong and cater the needs according to that particular region when the sales is less.

Q4 /* 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. */

```
WITH CTE AS (SELECT o.order_id, payment_value, EXTRACT(year FROM
order_purchase_timestamp ) AS Year,
EXTRACT(month FROM order_purchase_timestamp ) AS Month
FROM `project.orders` o
INNER JOIN `project.payments` p ON o.order_id = p.order_id
)
SELECT ROUND((Year_2018 - Year_2017)/Year_2017 * 100 ,2) AS Percentage_increase,
Year_2018,Year_2017
FROM
(SELECT SUM(CASE WHEN Year = 2017 AND Month BETWEEN 1 AND 8 THEN payment_value
END) AS Year_2017,
SUM(CASE WHEN Year = 2018 AND Month BETWEEN 1 AND 8 THEN payment_value END) AS
Year_2018
FROM CTE) AS X
```



Insights and recommendation: the % increase in the cost of orders from year 2017 to 2018 seems to be about 2 times more , we can compare the same using the graphical visualization of the data of cost between the two years.

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

SELECT

c.customer_state,

ROUND(SUM(t.price),2) AS total_price,

ROUND(AVG(t.price),2) AS average_price

FROM `project.orders` o

INNER JOIN `project.order_items` t ON o.order_id = t.order_id

INNER JOIN `project.customers` c ON o.customer_id = c.customer_id

GROUP BY c.customer_state

The screenshot shows the Google Cloud BigQuery console interface. At the top, there's a navigation bar with tabs for 'Business Case: Target SQL - Proj...', 'Business Case: Target SQL - Intro...', and 'Query results - BigQuery - Targ...'. Below the navigation bar, there's a message: 'p billing to upgrade to the full BigQuery experience. [Learn more](#)'. The main area displays a SQL query in the editor, which is the same query provided in the text blocks. The query is:

```
/* Calculate the Total & Average value of order price for each state.*/  
SELECT  
  c.customer_state,  
  ROUND(SUM(t.price),2) AS total_price,  
  ROUND(AVG(t.price),2) AS average_price  
FROM `project.orders` o  
INNER JOIN `project.order_items` t ON o.order_id = t.order_id  
INNER JOIN `project.customers` c ON o.customer_id = c.customer_id  
GROUP BY c.customer_state
```

 Below the query editor, the 'Query results' section is visible, showing a table with 4 columns: 'customer_state', 'total_price', and 'average_price'. The table contains 6 rows of data, corresponding to the states MT, MA, AL, SP, MG, and PE. The 'Results per page' is set to 50, and the total number of results is 1 of 27. The bottom of the screenshot shows a Windows taskbar with the date and time as 19-06-2023 00:42.

Row	customer_state	total_price	average_price
1	MT	156453.53	148.3
2	MA	119648.22	145.2
3	AL	80314.81	180.89
4	SP	5202955.05	109.65
5	MG	1585308.03	120.75
6	PE	262788.03	145.51

Insights and recommendation: Here we are calculating the total price and average price according to the state by joining multiple tables to sort data accordingly.

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

SELECT

c.customer_state,

ROUND(SUM(t.freight_value),2) AS total_freight_value,

ROUND(AVG(t.freight_value),2) AS average_freight_value

FROM `project.orders` o

INNER JOIN `project.order_items` t ON o.order_id = t.order_id

INNER JOIN `project.customers` c ON o.customer_id = c.customer_id

GROUP BY c.customer_state

The screenshot shows the Google Cloud BigQuery console interface. At the top, there's a navigation bar with tabs for 'Business Case: Target SQL - Proj...', 'Business Case: Target SQL Intro...', and 'Query results - BigQuery - Target...'. Below the navigation bar, there's a banner for 'p billing to upgrade to the full BigQuery experience. Learn more' with 'DISMISS' and 'UPGRADE' buttons. The main area displays a SQL query in the editor, which is the same query provided in the text blocks. The query is:
`SELECT
 c.customer_state,
 ROUND(SUM(t.freight_value),2) AS total_freight_value,
 ROUND(AVG(t.freight_value),2) AS average_freight_value
FROM `project.orders` o
INNER JOIN `project.order_items` t ON o.order_id = t.order_id
INNER JOIN `project.customers` c ON o.customer_id = c.customer_id
GROUP BY c.customer_state`
Below the query editor, the 'Query results' section is visible, showing a table with 3 columns: 'customer_state', 'total_freight_value', and 'average_freight_value'. The table contains 7 rows of data, corresponding to the states MT, MA, AL, SP, MG, PE, and RJ. The 'RESULTS' tab is selected, and the table is displayed in a grid format. The bottom of the screen shows a Windows taskbar with the date and time '19-06-2023 00:43'.

Row	customer_state	total_freight_value	average_freight_value
1	MT	29715.43	28.17
2	MA	31523.77	38.26
3	AL	15914.59	35.84
4	SP	718723.07	15.15
5	MG	270853.46	20.63
6	PE	59449.66	32.92
7	RJ	305589.31	20.96

Insights and recommendation: Here we are calculating the total freight and average freight according to the state by joining multiple tables to sort data accordingly.

Q5 /* 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:

$\text{time_to_deliver} = \text{order_delivered_customer_date} - \text{order_purchase_timestamp}$

$\text{diff_estimated_delivery} = \text{order_estimated_delivery_date} - \text{order_delivered_customer_date} *$

SELECT

order_id,

DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, day) AS
time_to_deliver,

DATE_DIFF(order_estimated_delivery_date, order_delivered_customer_date, day) AS
diff_estimated_delivery

FROM `project.orders`

ORDER BY order_id DESC

Business Case: Target SQL - Prof... x Business Case: Target SQL Intro... x Query results - BigQuery - Targ... x

console.cloud.google.com/bigquery?project=target-389500&ws=1m151m41m311starget-38950002sbqujob_6cef87af_188ced8d5a3sasia-south11m41m311starget-389500...

Dismiss Upgrade

+ ADD

Untitled

```

133
134 SELECT
135   order_id,
136   DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, day) AS time_to_deliver,
137   DATE_DIFF(order_estimated_delivery_date, order_delivered_customer_date, day) AS diff_estimated_delivery
138 FROM `project.orders`

```

Query results

SAVE RESULTS EXPLORE DATA

Row	order_id	time_to_deliver	diff_estimated_delivery
1	fffe41c64501cc87c801fd61db...	5	13
2	fffe18544ffabc95dfada21779c...	1	8
3	fffc4705a9662cd70adb13d4a...	4	12
4	fffc46ef2263f404302a634eb...	9	8
5	fffc94f6ce00a00581880bf54a...	17	7
6	fffbec3b5462987e66fb49b1c5...	16	17
7	fffb9224b6fc7c43ebb0904318...	21	9
8	fffb2ef8874127f75b52b64388...	17	10
9	fffb0b1a50e65c449020434fa8...	34	-2

Results per page: 50 1 - 50 of 99441

Insights and recommendation: This calculation of delivery time is taken to capture an average of time taken to deliver each product and draw insights to approximate the delivery time better for customer to have an idea of how quick the items would reach them.

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

SELECT

c.customer_state,

MAX(t.freight_value) AS top_5_highest_freight_value,

FROM `project.orders` o

INNER JOIN `project.order_items` t ON o.order_id = t.order_id

INNER JOIN `project.customers` c ON o.customer_id = c.customer_id

GROUP BY c.customer_state

limit 5

The screenshot shows the Google Cloud BigQuery console interface. At the top, there's a navigation bar with tabs for 'Business Case: Target SQL - Proj...', 'Business Case: Target SQL Intro...', and 'Query results - BigQuery - Targ...'. Below the navigation bar, there's a banner for 'p billing to upgrade to the full BigQuery experience. Learn more' with 'DISMISS' and 'UPGRADE' buttons. The main area displays a SQL query in the 'Untitled' tab, which is:

```
/* Find out the top 5 states with the highest & lowest average freight value.*/  
SELECT  
  c.customer_state,  
  MAX(t.freight_value) AS top_5_highest_freight_value,  
  MIN(t.freight_value) AS top_5_lowest_freight_value,  
FROM `project.orders` o  
INNER JOIN `project.order_items` t ON o.order_id = t.order_id  
INNER JOIN `project.customers` c ON o.customer_id = c.customer_id  
GROUP BY c.customer_state  
limit 5
```

 The query results are shown in a table with 5 rows and 2 columns: 'customer_state' and 'top_5_highest_freight_value'. The results are:

customer_state	top_5_highest_freight_value
MT	338.3
MA	245.75
AL	314.4
SP	339.59
MG	322.1

 The bottom of the screenshot shows a Windows taskbar with the date and time '19-06-2023 00:45'.

Insights and recommendation: Here we are checking the top 5 record to see why it is on the top most and implement the similar approach for the other states

SELECT

c.customer_state,

MIN(t.freight_value) AS top_5_lowest_freight_value,

FROM `project.orders` o

INNER JOIN `project.order_items` t ON o.order_id = t.order_id

INNER JOIN `project.customers` c ON o.customer_id = c.customer_id

GROUP BY c.customer_state

limit 5

The screenshot shows the Google Cloud BigQuery console interface. At the top, there's a navigation bar with tabs for 'Business Case: Target SQL - Proj...', 'Business Case: Target SQL Intro', and 'Query results - BigQuery - Targ...'. Below this is a banner for 'p billing to upgrade to the full BigQuery experience. Learn more' with 'DISMISS' and 'UPGRADE' buttons. The main area has a toolbar with 'ADD', 'RUN', 'SAVE', 'SHARE', 'SCHEDULE', and 'MORE' buttons. The SQL editor shows the following query:

```
SELECT
  c.customer_state,
  MIN(t.freight_value) AS top_5_lowest_freight_value,
FROM `project.orders` o
INNER JOIN `project.order_items` t ON o.order_id = t.order_id
INNER JOIN `project.customers` c ON o.customer_id = c.customer_id
GROUP BY c.customer_state
LIMIT 5
```

Below the editor, the 'Query results' section is visible, showing a table with 5 rows and 2 columns: 'customer_state' and 'top_5_lowest_freight'. The results are:

Row	customer_state	top_5_lowest_freight
1	MT	0.0
2	MA	0.0
3	AL	0.0
4	SP	0.0
5	MG	0.0

The bottom of the screen shows a Windows taskbar with the date '19-06-2023' and time '00:49'.

Since all the values of freight in the lowest shows value as 0 , checked for values other than 0's

SELECT

c.customer_state,

MIN(t.freight_value) AS top_5_lowest_freight_value,

FROM `project.orders` o

INNER JOIN `project.order_items` t ON o.order_id = t.order_id

INNER JOIN `project.customers` c ON o.customer_id = c.customer_id

WHERE t.freight_value <> 0

GROUP BY c.customer_state

limit 5

The screenshot shows the Google Cloud BigQuery console. The SQL query in the editor is as follows:

```
SELECT
  c.customer_state,
  MIN(t.freight_value) AS top_5_lowest_freight_value,
FROM `project.orders` o
INNER JOIN `project.order_items` t ON o.order_id = t.order_id
INNER JOIN `project.customers` c ON o.customer_id = c.customer_id
WHERE t.freight_value <= 0
GROUP BY c.customer_state
LIMIT 5
```

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

Row	customer_state	top_5_lowest_freight
1	MT	0.53
2	MA	1.34
3	AL	3.25
4	SP	0.01
5	MG	0.01

Insights and recommendation: Here we are checking the bottom 5 record to see why it is on the bottom most and improve the similar approach for the states

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

SELECT

c.customer_state,

AVG(DATE_DIFF(o.order_delivered_customer_date,o.order_purchase_timestamp, day)) AS
top_5_states_highest_delivery,

FROM `project.orders` o

INNER JOIN `project.order_items` t ON o.order_id = t.order_id

INNER JOIN `project.customers` c ON o.customer_id = c.customer_id

GROUP BY c.customer_state

ORDER BY top_5_states_highest_delivery DESC

limit 5

The screenshot shows the Google Cloud BigQuery console interface. At the top, there's a navigation bar with tabs for 'Business Case: Target SQL - Proj...', 'Business Case: Target SQL Intro', and 'Query results - BigQuery - Targ...'. Below this is a search bar and a 'p billing to upgrade to the full BigQuery experience. Learn more' message. The main area displays a SQL query in a text editor, with a 'RUN' button and 'SAVE', 'SHARE', 'SCHEDULE', and 'MORE' options. The query is as follows:

```
168 /* Find out the top 5 states with the highest & lowest average delivery time.*/  
169  
170 SELECT  
171   c.customer_state,  
172   AVG(DATE_DIFF(o.order_delivered_customer_date,o.order_purchase_timestamp, day)) AS top_5_states_highest_delivery,  
173 FROM `project.orders` o  
174 INNER JOIN `project.order_items` t ON o.order_id = t.order_id  
175 INNER JOIN `project.customers` c ON o.customer_id = c.customer_id  
176 GROUP BY c.customer_state  
177 ORDER BY top_5_states_highest_delivery DESC  
178 limit 5
```

Below the query editor, the 'Query results' section is visible, showing a table with 5 rows of data. The table has two columns: 'customer_state' and 'top_5_states_highest_delivery'. The results are as follows:

Row	customer_state	top_5_states_highest_delivery
1	RR	27.82608695652...
2	AP	27.75308641975...
3	AM	25.96319018404...
4	AL	23.99297423887...
5	PA	23.30170777988...

Insights and recommendation: Here we are checking the top 5 record to see the most quick deliveries done to the states and check why it is so , may be the fact being there are many warehouses accessible and near the city limits making it faster.

Try improving by implementing similar approach in other states where delivery time is late.

SELECT

c.customer_state,

AVG(DATE_DIFF(o.order_delivered_customer_date,o.order_purchase_timestamp, day)) AS
top_5_states_lowest_delivery,

FROM `project.orders` o

INNER JOIN `project.order_items` t ON o.order_id = t.order_id

INNER JOIN `project.customers` c ON o.customer_id = c.customer_id

GROUP BY c.customer_state

ORDER BY top_5_states_lowest_delivery ASC

limit 5

The screenshot shows the Google Cloud BigQuery console. The query editor displays the following SQL query:

```
SELECT
  c.customer_state,
  AVG(DATE_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp, day)) AS top_5_states_lowest_delivery,
FROM `project.orders` o
INNER JOIN `project.order_items` t ON o.order_id = t.order_id
INNER JOIN `project.customers` c ON o.customer_id = c.customer_id
GROUP BY c.customer_state
ORDER BY top_5_states_lowest_delivery ASC
limit 5
```

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

Row	customer_state	top_5_states_lowest_delivery
1	SP	8.259608552419...
2	PR	11.48079306071...
3	MG	11.51552218007...
4	DF	12.50148619957...
5	SC	14.52098584675...

Insights and recommendation: Here we are checking the lowest 5 record to see the least speed of delivery or a delay in the delivery time and bring improvement on the same.

/* 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. */

SELECT customer_state, ROUND(AVG(diff_estimated_delivery), 2) AS delivery_speed

FROM (

SELECT

o.order_id,

c.customer_state,

DATE_DIFF(o.order_estimated_delivery_date, o.order_delivered_customer_date, DAY) AS

diff_estimated_delivery

FROM `project.orders` o

INNER JOIN `project.customers` c ON o.customer_id = c.customer_id

)

GROUP BY customer_state

ORDER BY 2 ASC;

The screenshot shows the Google BigQuery console interface. At the top, there's a navigation bar with tabs for 'Business Case: Target SQL - Proj...', 'Business Case: Target SQL Intro...', and 'Query results - BigQuery - Targ...'. Below the navigation bar, there's a banner for 'Billing to upgrade to the full BigQuery experience'. The main area displays a SQL query in the 'Untitled' editor. The query is as follows:

```
192 You can use the difference between the averages of actual & estimated delivery date to figure out how fast the delivery was for each state. */
193 SELECT customer_state, ROUND(AVG(diff_estimated_delivery), 2) AS delivery_speed
194 FROM (
195   SELECT
196     o.order_id,
197     c.customer_state,
198     DATE_DIFF(o.order_estimated_delivery_date, o.order_delivered_customer_date, DAY) AS diff_estimated_delivery
199   FROM `project.orders` o
200   INNER JOIN `project.customers` c ON o.customer_id = c.customer_id
201 )
202 GROUP BY customer_state
203 ORDER BY 2 ASC;
```

Below the query editor, the 'Query results' section is visible. It shows a table with two columns: 'customer_state' and 'delivery_speed'. The table contains five rows of data:

Row	customer_state	delivery_speed
1	AL	7.95
2	MA	8.77
3	SE	9.17
4	ES	9.62
5	BA	9.93

The bottom of the screenshot shows the Windows taskbar with the date and time as 19-06-2023, 00:52.

Insights and recommendation: Here we are checking how the delivery is faster than the estimated time and how this can improve customer satisfaction and draw insights from customer reviews about the same.

Q6/* Find the month on month no. of orders placed using different payment types.*/

SELECT

```

EXTRACT(month FROM order_purchase_timestamp) AS Month,

p.payment_type

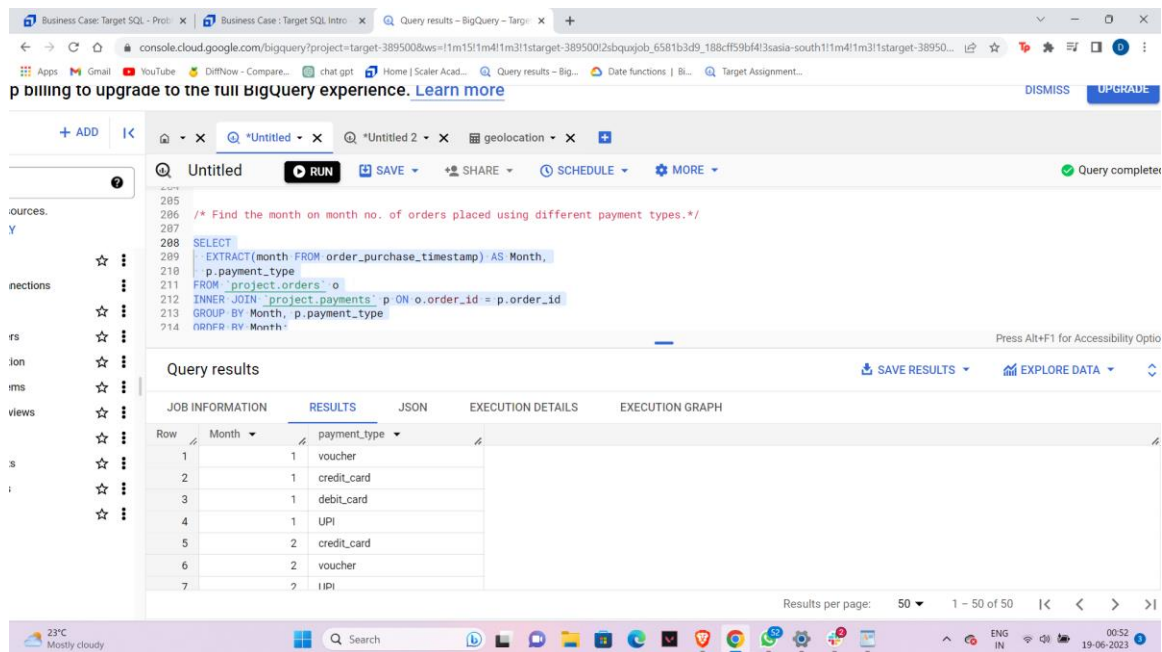
FROM `project.orders` o

INNER JOIN `project.payments` p ON o.order_id = p.order_id

GROUP BY Month, p.payment_type

ORDER BY Month;

```



The screenshot shows a BigQuery console interface. The SQL query is as follows:

```

/* Find the month on month no. of orders placed using different payment types.*/
SELECT
  EXTRACT(month FROM order_purchase_timestamp) AS Month,
  p.payment_type
FROM `project.orders` o
INNER JOIN `project.payments` p ON o.order_id = p.order_id
GROUP BY Month, p.payment_type
ORDER BY Month;

```

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

Row	Month	payment_type
1	1	voucher
2	1	credit_card
3	1	debit_card
4	1	UPI
5	2	credit_card
6	2	voucher
7	2	UPI

Insights and recommendation: Here we can conclude the different payment methods used by customers and understand the ease of payment and offer an analysis to check which of the methods have high success rates of making the payment without causing an error.

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

```

SELECT

payment_installments,

```

```

COUNT(DISTINCT order_id) AS order_count

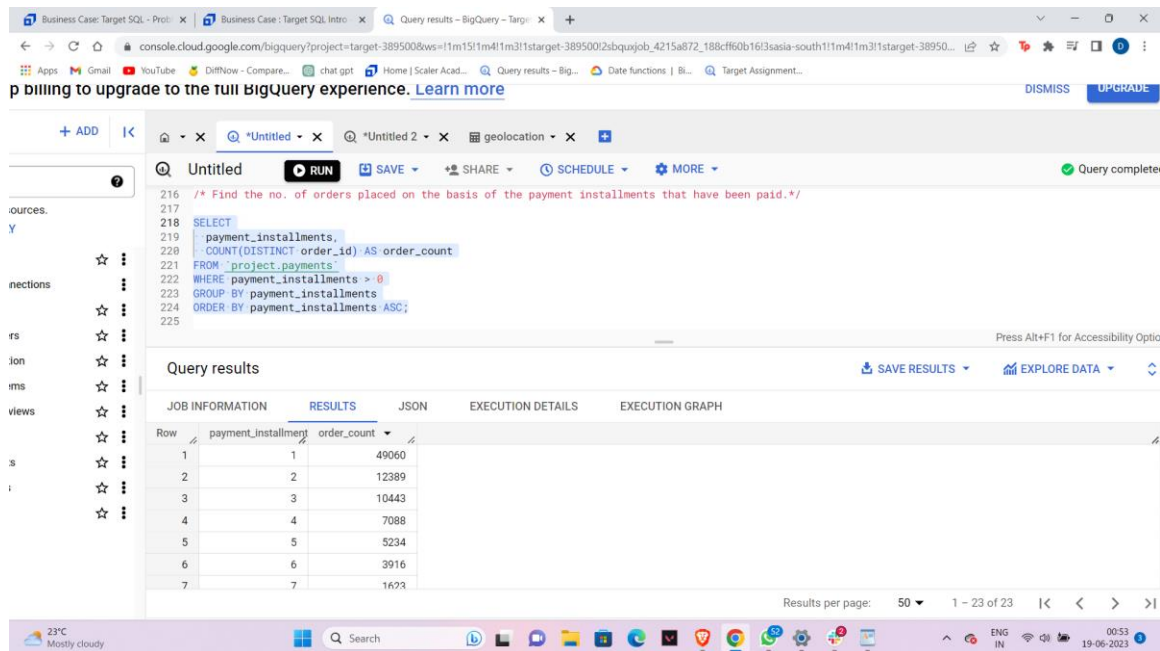
FROM `project.payments`

WHERE payment_installments > 0

GROUP BY payment_installments

ORDER BY payment_installments ASC;

```



The screenshot shows the Google BigQuery console interface. At the top, there's a navigation bar with tabs for 'Business Case: Target SQL - Proj...', 'Business Case: Target SQL Intro', and 'Query results - BigQuery - Targ...'. Below the navigation bar, there's a banner for 'p billing to upgrade to the full BigQuery experience. Learn more'. The main area displays a SQL query in the 'Untitled' editor. The query is as follows:

```

/* Find the no. of orders placed on the basis of the payment installments that have been paid.*/
SELECT
  payment_installments,
  COUNT(DISTINCT order_id) AS order_count
FROM `project.payments`
WHERE payment_installments > 0
GROUP BY payment_installments
ORDER BY payment_installments ASC;

```

Below the query editor, the 'Query results' section is visible. It shows a table with 7 rows of data. The table has two columns: 'payment_installment' and 'order_count'. The results are as follows:

Row	payment_installment	order_count
1	1	49060
2	2	12389
3	3	10443
4	4	7088
5	5	5234
6	6	3916
7	7	1623

At the bottom of the console, there's a status bar showing 'Results per page: 50' and '1 - 23 of 23'. The system tray at the very bottom shows the date and time as '19-06-2023 00:53'.

Insights and recommendation: Here we can see how much percentage of people bought items because of the installment options and we how we could push more sales if given a no cost month installment option.

Conclusion:

- 1) We can improve sales by giving discounts during the year end where the sales is low
- 2) Improve the delivery speed in some regions by setting up warehouses within the city limits
- 3) Launch excited products during the dawn when the sales is really low and advertise more about it to improve the sales.
- 4) We must analyze what customers need according to the state they live in , check the culture , weather conditions and the need of the hour products and launch them exclusively.

5) Reduce the price of products that are being least purchased.

6) Offer discount coupons or vouchers to retain the customer to make another purchase.

And finally take insight from the top products sold in a region and mimic the approach according to the region/ state for better marketing.