

TARGET – BUSINESS CASE – SQL

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

1.1 Data type of all columns in the "customers" table.

OUTPUT:

The screenshot shows a data catalog interface. On the left, a sidebar lists workspace resources under 'sql-business-case-397604'. The 'customers' table is selected. On the right, the 'SCHEMA' tab is active, displaying a table with columns: Field name, Type, and Mode. The columns listed are customer_id (STRING, NULLABLE), customer_unique_id (STRING, NULLABLE), customer_zip_code_prefix (INTEGER, NULLABLE), customer_city (STRING, NULLABLE), and customer_state (STRING, NULLABLE). Below the table are buttons for 'EDIT SCHEMA' and 'VIEW ROW ACCESS POLICIES'.

Field name	Type	Mode
customer_id	STRING	NULLABLE
customer_unique_id	STRING	NULLABLE
customer_zip_code_prefix	INTEGER	NULLABLE
customer_city	STRING	NULLABLE
customer_state	STRING	NULLABLE

INFERENCE:

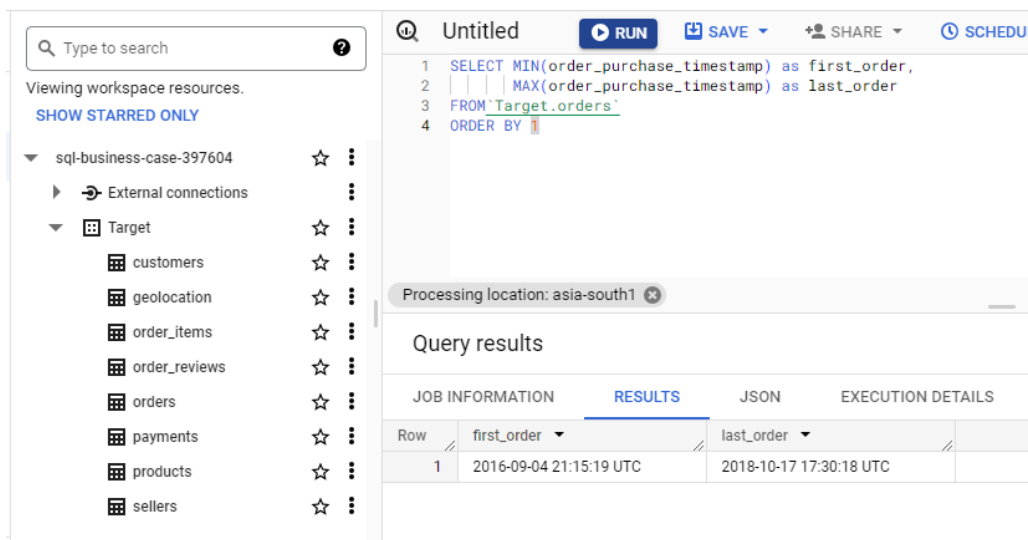
- The above image shows that the **Data type** of **all columns** in the **customers** table.
- In that **customers** table contains 2 types of Data types.
(i.e)
 1. **STRING** - (It stores the **Variables**)
 2. **INTEGER** - (It contains only **Numbers**)

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

QUERY:

```
SELECT MIN(order_purchase_timestamp) as first_order,  
  
MAX(order_purchase_timestamp) as last_order  
  
FROM `Target.orders`  
  
ORDER BY 1;
```

OUTPUT:



The screenshot shows a SQL query editor interface. On the left is a sidebar with a search bar and a list of workspace resources under 'sql-business-case-397604', including 'External connections' and 'Target' (with sub-items like customers, geolocation, order_items, order_reviews, orders, payments, products, and sellers). The main editor area shows a query titled 'Untitled' with the following SQL code:

```
1 SELECT MIN(order_purchase_timestamp) as first_order,  
2     MAX(order_purchase_timestamp) as last_order  
3 FROM `Target.orders`  
4 ORDER BY 1
```

Below the query editor, the 'Query results' section is visible, showing a table with the following data:

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

INFERENCE:

- The Above image shows the result of Time Range between the First order and Last order.
 - **First order** date on **04-09- 2016** (September 4th 2016) and time on **21:15:19 UTC**
 - **Last order** date on **17-10-2018** (October 17th 2018) and time on **17:30:18 UTC**
- The Total Duration between the **First order and Last order** is **2 Years and 1 Month**.

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

QUERY:

```
SELECT COUNT(DISTINCT c.customer_city ) as Cities, COUNT(DISTINCT c.customer_state) as States

FROM `Target.orders` as o
JOIN `Target.customers` as c
ON o.customer_id=c.customer_id
```

OUTPUT:

The screenshot shows a SQL query editor interface. On the left, there's a sidebar with a search bar and a list of workspace resources under 'sql-business-case-397604', including 'External connections', 'Target', and various tables like 'customers', 'geolocation', 'order_items', 'order_reviews', 'orders', 'payments', 'products', and 'sellers'. The main editor area shows the SQL query:
#1.3 Count the Cities & States of customers who ordered during the given period
SELECT COUNT(DISTINCT c.customer_city) as Cities, COUNT(DISTINCT c.customer_state) as States
FROM Target.orders as o
JOIN Target.customers as c
ON o.customer_id=c.customer_id
Below the query, it says 'Processing location: asia-south1'. The 'Query results' section is active, showing a table with columns 'Cities' and 'States'. The first row shows 4119 Cities and 27 States.

Row	Cities	States
1	4119	27

INFERENCE:

➤ The Above query finds the **Total Number of Cities and States of Customers purchased in the Given Period.**

- **4119 Cities**
- **27 States**

2. In-depth Exploration

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

QUERY:

```
SELECT EXTRACT( year FROM order_purchase_timestamp ) AS year,  
  
COUNT(order_id) AS orders  
FROM `Target.orders`  
GROUP BY 1  
ORDER BY 1
```

OUTPUT:

The screenshot shows the Google Cloud SQL Business Case interface. On the left, there's a sidebar with a search bar and a list of workspace resources under 'sql-business-case-397604'. The main area displays a query titled 'Untitled' with the following SQL code:

```
#2.1 a. Is there a growing trend in the no. of orders placed over the past years?  
SELECT EXTRACT( year FROM order_purchase_timestamp ) AS year,  
COUNT(order_id) AS orders  
FROM `Target.orders`  
GROUP BY 1  
ORDER BY 1
```

Below the query, the 'Query results' section is visible, showing a table with 3 rows of data:

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

INFERENCE:

- After the analysis of Dataset,
 - During the **First Year 2016**, the total number of orders was **329**.
 - But in the **Second Year 2017** it has an increase by **45101** total number of orders.
 - Then we have **54011** total number of orders in the **Last Year 2018** which is the **highest number of orders among the three years**.
- Finally we can clearly understand that the **number of orders** has a **growing trend** over the given period of time.

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

QUERY:

```
SELECT EXTRACT(month FROM order_purchase_timestamp) AS date,  
  
COUNT(order_id) AS orders  
FROM `Target.orders`  
GROUP BY date  
ORDER BY orders DESC
```

OUTPUT:

SANDBOX Set up billing to upgrade to the full BigQuery experience. [Learn more](#)

Explorer Type to search

Viewing workspace resources.
[SHOW STARRED ONLY](#)

- sql-business-case-39...
- Saved queries (1)
- TARGET**
- SHOW MORE
- External connectio...
- Target
 - customers
 - geolocation
 - order_items
 - order_revi...
 - orders

Query results

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS
Row	date	orders		
1		8	10843	
2		5	10573	
3		7	10318	
4		3	9893	
5		6	9412	
6		4	9343	
7		2	8508	
8		1	8069	
9		11	7544	
10		12	5674	
11		10	4959	
12		9	4305	

INFERENCE:

- The Above Image shows the number of orders placed in monthly seasonality. Here the Top 3 months of orders,
- During **August** the highest number of orders were placed – **10843 orders**.
 - The **Second highest** orders placed is on **May** – **10573 orders**.
 - Third highest number of orders were placed on **July** – **10318 orders**.

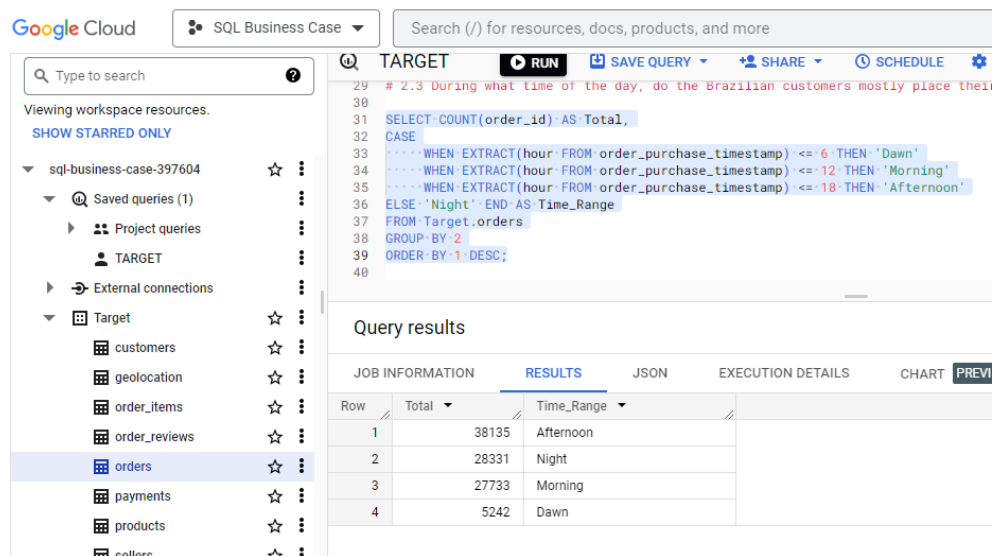
2.3 During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)

- 0-6 hrs : Dawn
- 7-12 hrs : Mornings
- 13-18 hrs : Afternoon
- 19-23 hrs : Night

QUERY:

```
SELECT COUNT(order_id) AS Total,
CASE
  WHEN EXTRACT(hour FROM order_purchase_timestamp) <= 6 THEN 'Dawn'
  WHEN EXTRACT(hour FROM order_purchase_timestamp) <= 12 THEN 'Morning'
  WHEN EXTRACT(hour FROM order_purchase_timestamp) <= 18 THEN 'Afternoon'
  ELSE 'Night' END AS Time_Range
FROM Target.orders
GROUP BY 2
ORDER BY 1 DESC;
```

OUTPUT:



The screenshot shows the Google Cloud SQL Business Case interface. On the left, a sidebar lists workspace resources, including 'sql-business-case-397604' and 'Target'. The 'Target' table is selected. The main area displays a SQL query in a text editor, which is the same query as shown in the 'QUERY' section. Below the query editor, the 'Query results' section is visible, showing a table with 4 rows and 3 columns: 'Row', 'Total', and 'Time_Range'. The results are as follows:

Row	Total	Time_Range
1	38135	Afternoon
2	28331	Night
3	27733	Morning
4	5242	Dawn

INFERENCE:

- After the analysis of dataset, mostly **Brazilian Customers** done the orders in **Peak Time** of **Afternoon** compared to the other timings.
 - **Afternoon – 38135 orders**
 - **Night – 28331 orders**
 - **Morning – 27733 orders**
 - **Dawn – 5242 orders**

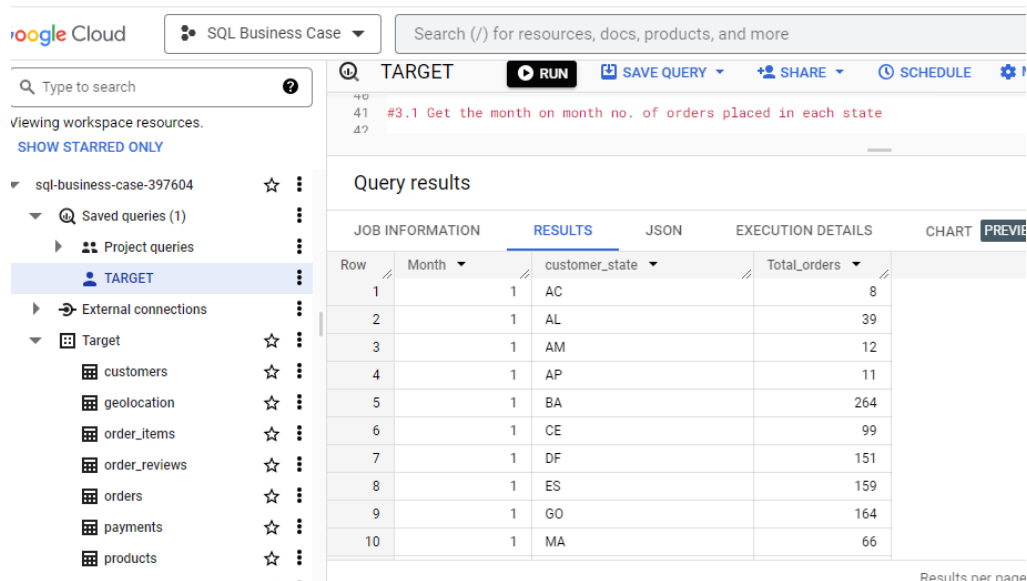
3. Evolution of E-commerce orders in the Brazil region

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

QUERY:

```
SELECT EXTRACT(month FROM order_purchase_timestamp) AS Month, C.customer_state,  
  
       COUNT(order_id) AS Total_orders  
  
FROM `Target.customers` C  
JOIN `Target.orders` O  
ON C.customer_id = O.customer_id  
GROUP BY 1, 2  
ORDER BY 1,2
```

OUTPUT:



The screenshot shows the Google Cloud SQL Business Case interface. On the left, a sidebar lists workspace resources, including a project named 'sql-business-case-397604' and a 'Target' dataset with tables like 'customers', 'orders', and 'products'. The main panel displays a SQL query titled 'TARGET' with the text '#3.1 Get the month on month no. of orders placed in each state'. Below the query, the 'Query results' tab is active, showing a table with 10 rows. The table has columns for 'Month', 'customer_state', and 'Total_orders'. The data shows that for month 1, orders were placed in states AC, AL, AM, AP, BA, CE, DF, ES, GO, and MA, with a total of 66 orders.

Row	Month	customer_state	Total_orders
1	1	AC	8
2	1	AL	39
3	1	AM	12
4	1	AP	11
5	1	BA	264
6	1	CE	99
7	1	DF	151
8	1	ES	159
9	1	GO	164
10	1	MA	66

INFERENCE:

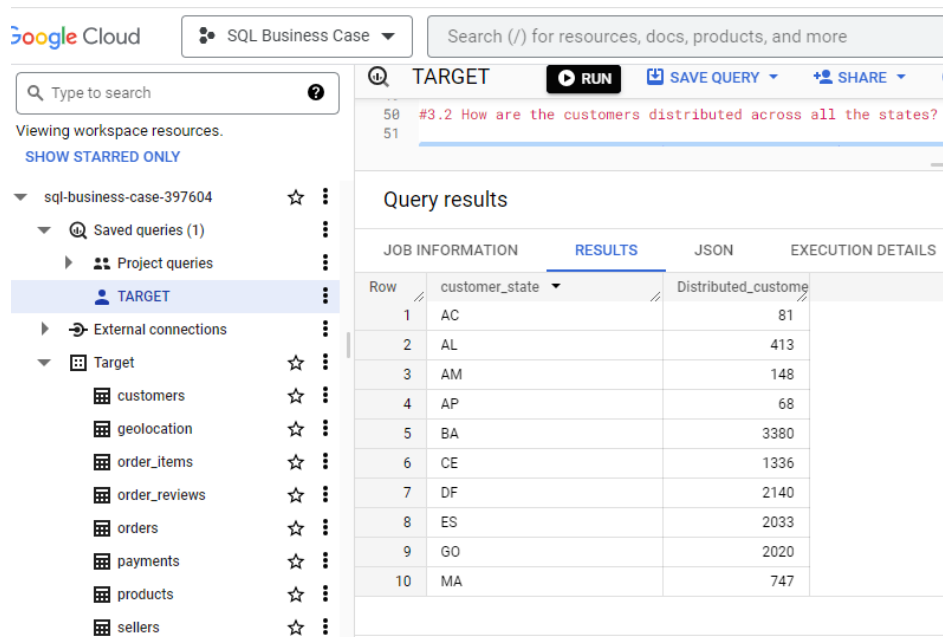
- After analyzing the dataset the **Total Number of orders** placed in **each state on month on month** in the above Image.
 - No. of orders are grouped by **Month on Month** in each state.

3.2 How are the customers distributed across all the states?

QUERY:

```
SELECT customer_state, COUNT(DISTINCT customer_id) AS Distributed_customers
FROM `Target.customers`
GROUP BY 1
ORDER BY 1
```

OUTPUT:



The screenshot shows the Google Cloud SQL Business Case interface. On the left, a sidebar lists workspace resources, including a project named 'sql-business-case-397604' with a 'TARGET' query selected. The main panel displays the query results for the query titled '#3.2 How are the customers distributed across all the states?'. The results are shown in a table with columns 'customer_state' and 'Distributed_customers'.

Row	customer_state	Distributed_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

INFERENCE:

- After the analysis of dataset, the Distributed Customer present in **all states** is displayed through the above query and the output is displayed in the above image.
 - The **distributed customers** are grouped by the **unique customer in each state**.

4. Impact on Economy: Analyze the money movement by Ecommerce 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).

QUERY:

```
SELECT ROUND(

    ((SUM(CASE WHEN EXTRACT(YEAR FROM order_purchase_timestamp) = 2018 THEN payment_value ELSE 0

END)

- SUM(CASE WHEN EXTRACT(YEAR FROM order_purchase_timestamp) = 2017 THEN payment_value ELSE 0

END))

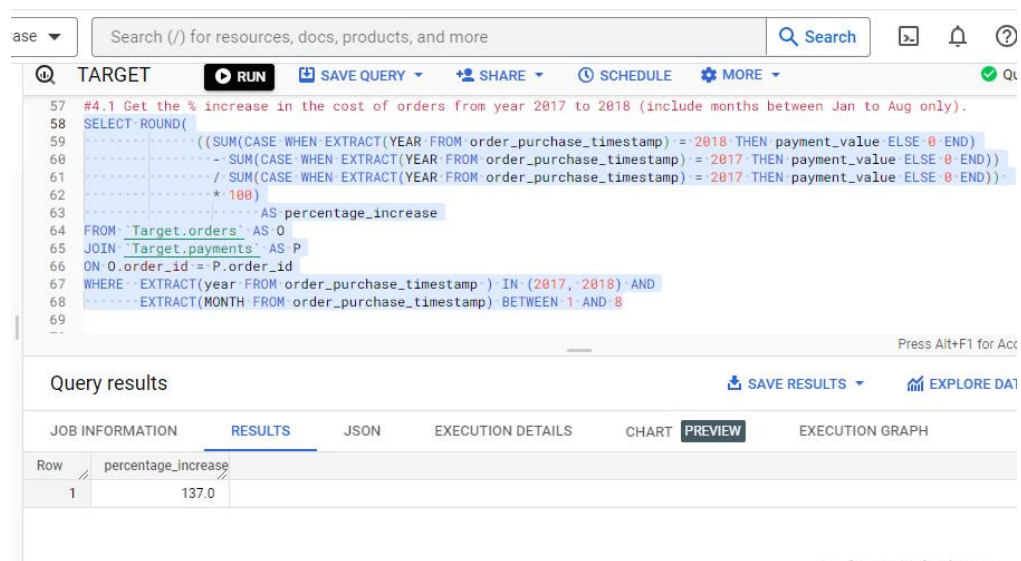
/ SUM(CASE WHEN EXTRACT(YEAR FROM order_purchase_timestamp) = 2017 THEN payment_value ELSE 0

END))

* 100)

    AS percentage_increase
FROM `Target.orders` AS O
JOIN `Target.payments` AS P
ON O.order_id = P.order_id
WHERE EXTRACT(year FROM order_purchase_timestamp) IN (2017, 2018) AND
      EXTRACT(MONTH FROM order_purchase_timestamp) BETWEEN 1 AND 8
```

OUTPUT:



The screenshot shows a SQL query editor interface. At the top, there is a search bar and a toolbar with buttons for RUN, SAVE QUERY, SHARE, SCHEDULE, and MORE. Below the toolbar, the query is displayed with line numbers 57 to 69. The query is the same as the one provided in the previous block. Below the query, there is a section for 'Query results' with tabs for JOB INFORMATION, RESULTS, JSON, EXECUTION DETAILS, CHART, PREVIEW, and EXECUTION GRAPH. The 'RESULTS' tab is selected, showing a table with two columns: 'Row' and 'percentage_increase'. The first row shows the value 137.0.

Row	percentage_increase
1	137.0

INFERENCE:

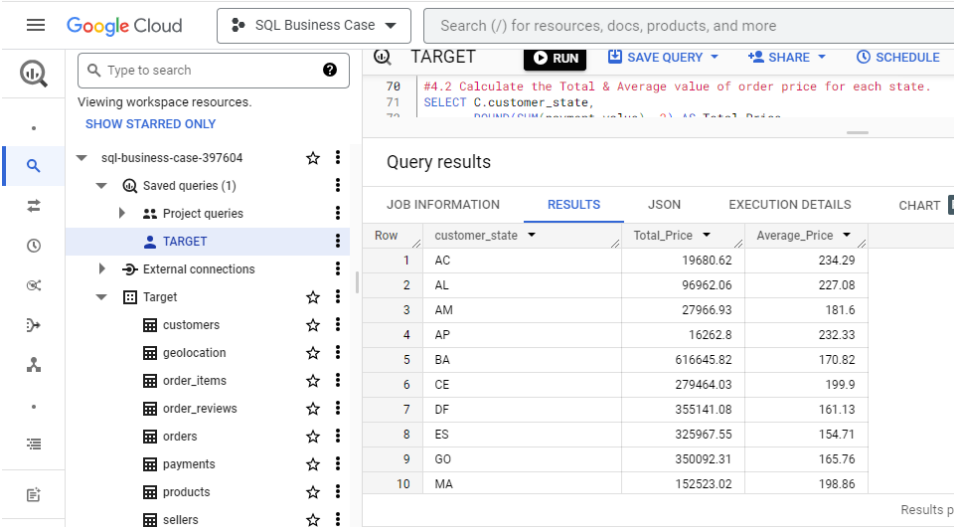
- After the analysis of the dataset, the **Percentage Increase** in the **cost of orders** from the **year 2017 to 2018** (includes the **months** between **Jan to Aug** only) is **137%**.

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

QUERY:

```
SELECT C.customer_state,
       ROUND(SUM(payment_value), 2) AS Total_Price,
       ROUND(AVG(payment_value), 2) AS Average_Price
FROM `Target.customers` C
JOIN `Target.orders` O
ON C.customer_id = O.customer_id
JOIN `Target.payments` P
ON P.order_id = O.order_id
GROUP BY 1
ORDER BY 1
```

OUTPUT:



The screenshot shows the Google Cloud SQL Business Case interface. On the left, a sidebar lists workspace resources including 'sql-business-case-397604', 'Saved queries (1)', 'Project queries', and 'TARGET'. The 'TARGET' query is selected. The main panel displays the query text and its results. The query is: `SELECT C.customer_state, ROUND(SUM(payment_value), 2) AS Total_Price, ROUND(AVG(payment_value), 2) AS Average_Price FROM Target.customers C JOIN Target.orders O ON C.customer_id = O.customer_id JOIN Target.payments P ON P.order_id = O.order_id GROUP BY 1 ORDER BY 1`. The results are shown in a table with columns: Row, customer_state, Total_Price, and Average_Price. The results are grouped by customer_state.

Row	customer_state	Total_Price	Average_Price
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
8	ES	325967.55	154.71
9	GO	350092.31	165.76
10	MA	152523.02	198.86

INFERENCE:

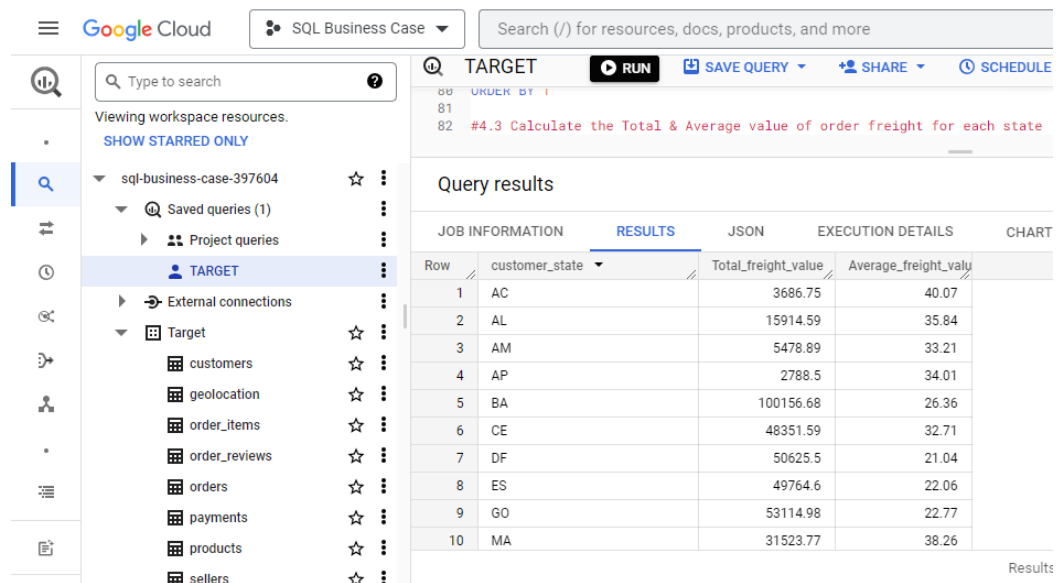
- After the analysis of the dataset, the **Total and Average** value of **order price** for each state as shown in the output image.
 - The **Total Price** and the **Average Price** of the **orders** are grouped by **each State** respectively.

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

QUERY:

```
SELECT C.customer_state,
       ROUND(SUM(freight_value), 2) AS Total_freight_value,
       ROUND(AVG(freight_value), 2) AS Average_freight_value
FROM `Target.customers` C
JOIN `Target.orders` O
ON C.customer_id = O.customer_id
JOIN `Target.order_items` OI
ON OI.order_id = O.order_id
GROUP BY 1
ORDER BY 1
```

OUTPUT:



The screenshot shows the Google Cloud SQL Business Case interface. On the left, a sidebar lists workspace resources, including a project named 'sql-business-case-397604' and a dataset named 'Target' with tables like 'customers', 'geolocation', 'order_items', 'order_reviews', 'orders', 'payments', 'products', and 'sellers'. The main panel displays a SQL query titled 'TARGET' with a 'RUN' button. Below the query, the 'Query results' section shows a table with 10 rows of data. The table has columns for 'customer_state', 'Total_freight_value', and 'Average_freight_value'. The data is grouped by state, showing the total and average freight for each state.

Row	customer_state	Total_freight_value	Average_freight_value
1	AC	3686.75	40.07
2	AL	15914.59	35.84
3	AM	5478.89	33.21
4	AP	2788.5	34.01
5	BA	100156.68	26.36
6	CE	48351.59	32.71
7	DF	50625.5	21.04
8	ES	49764.6	22.06
9	GO	53114.98	22.77
10	MA	31523.77	38.26

INFERENCE:

- After the analysis of the dataset, the **Total and Average** value of **order freight** for each state as shown in the output image.
 - The **Total freight value** and the **Average freight value** of the **orders** are grouped by **each State** respectively.

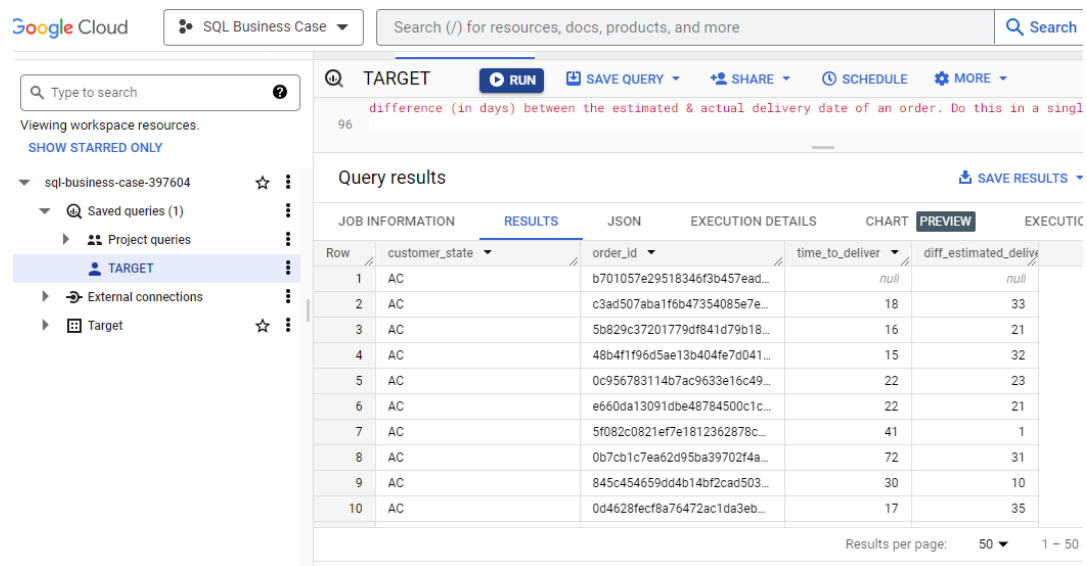
5. Analysis based on sales, freight and delivery time

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.

QUERY:

```
SELECT customer_state,
        order_id,
        DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp,day) AS
time_to_deliver,
        ABS(DATE_DIFF(order_estimated_delivery_date,order_delivered_customer_date,day))
AS diff_estimated_delivery
FROM `Target.orders` AS O
JOIN `Target.customers` AS C
ON O.customer_id = C.customer_id
GROUP BY 1,2,3,4
ORDER BY 1
```

OUTPUT:



Google Cloud SQL Business Case

Search (/) for resources, docs, products, and more

Search

Viewing workspace resources.

SHOW STARRED ONLY

sql-business-case-397604

Saved queries (1)

Project queries

TARGET

External connections

Target

Query results

SAVE RESULTS

Row	customer_state	order_id	time_to_deliver	diff_estimated_delivery
1	AC	b701057e29518346f3b457ead...	null	null
2	AC	c3ad507aba1f6b47354085e7e...	18	33
3	AC	5b829c37201779df841d79b18...	16	21
4	AC	48b4f1f96d5ae13b404fe7d041...	15	32
5	AC	0c956783114b7ac9633e16c49...	22	23
6	AC	e660da13091dbe48784500c1c...	22	21
7	AC	5f082c0821ef7e1812362878c...	41	1
8	AC	0b7cb1c7ea62d95ba39702f4a...	72	31
9	AC	845c454659dd4b14bf2cad503...	30	10
10	AC	0d4628fecf8a76472ac1da3eb...	17	35

Results per page: 50 1 - 50

INFERENCE:

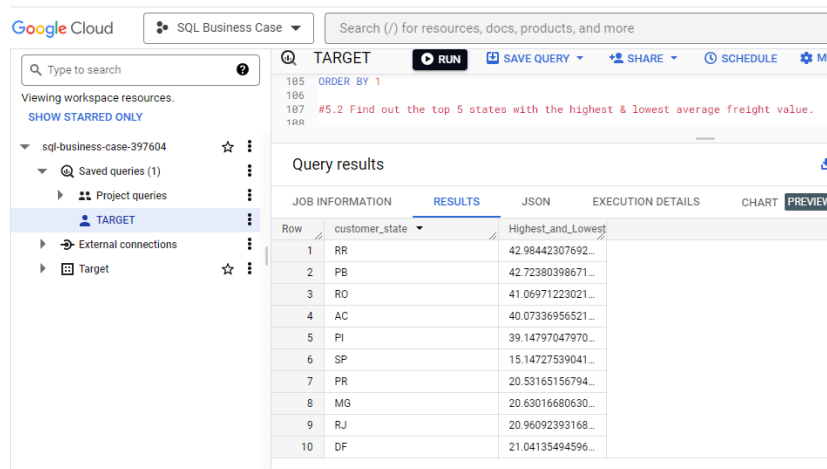
- The no. of days taken to deliver each order from the order's purchase date is shown as **time_to_deliver**.
- And the calculated difference between the estimated & actual delivery date of an order is shown as **diff_estimated_delivery**.

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

QUERY:

```
(SELECT customer_state, AVG(OI.freight_value) AS Highest_and_Lowest_freight_value
FROM `Target.orders` O
JOIN `Target.customers` C
ON O.customer_id = C.customer_id
JOIN `Target.order_items` OI
ON O.order_id = OI.order_id
GROUP BY 1
ORDER BY 2 DESC
LIMIT 5)
UNION DISTINCT
(SELECT customer_state, AVG(OI.freight_value) AS Highest_and_Lowest_freight_value
FROM `Target.orders` O
JOIN `Target.customers` C
ON O.customer_id = C.customer_id
JOIN `Target.order_items` OI
ON O.order_id = OI.order_id
GROUP BY 1
ORDER BY 2 ASC
LIMIT 5)
```

OUTPUT:



The screenshot shows the Google Cloud SQL Business Case interface. The query results are displayed in a table with the following data:

Row	customer_state	Highest_and_Lowest
1	RR	42.98442307692...
2	PB	42.72380398671...
3	RO	41.06971223021...
4	AC	40.07336956521...
5	PI	39.14797047970...
6	SP	15.14727539041...
7	PR	20.53165156794...
8	MG	20.63016680630...
9	RJ	20.96092393168...
10	DF	21.04135494596...

INFERENCE:

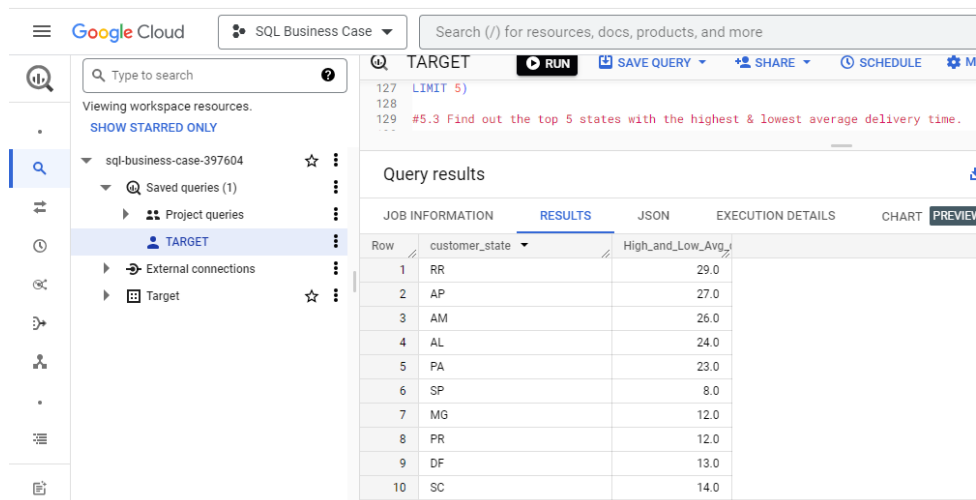
- After the analysis of dataset, The **Top 5 States** with the **Highest and Lowest Average freight value** is displayed in the above output,
 - The **First 5** output values from **1 to 5** shows the **Highest Average freight values**.
 - The **Last 5** output values from **6-10** shows the **Lowest Average freight values**.

5.3 Find out the top 5 states with the Highest & Lowest average delivery time.

QUERY:

```
(SELECT C.customer_state, ROUND(AVG(time_to_deliver)) AS High_and_Low_Avg_delivery_time
FROM (SELECT *, DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp,day) AS
time_to_deliver
FROM `Target.orders`) O
JOIN `Target.customers` C
ON C.customer_id = O.customer_id
GROUP BY 1
ORDER BY 2 DESC
LIMIT 5)
UNION ALL
(SELECT C.customer_state, ROUND(AVG(time_to_deliver)) AS High_and_Low_Avg_delivery_time
FROM (SELECT *, DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp,day) AS
time_to_deliver
FROM `Target.orders`) O
JOIN `Target.customers` C
ON C.customer_id = O.customer_id
GROUP BY 1
ORDER BY 2
LIMIT 5)
```

OUTPUT:



The screenshot shows the Google Cloud SQL Business Case interface. The query results are displayed in a table with the following data:

Row	customer_state	High_and_Low_Avg_delivery_time
1	RR	29.0
2	AP	27.0
3	AM	26.0
4	AL	24.0
5	PA	23.0
6	SP	8.0
7	MG	12.0
8	PR	12.0
9	DF	13.0
10	SC	14.0

INFERENCE:

- After the analysis of dataset, The **Top 5 States** with the **Highest and Lowest Average delivery time** is displayed in the above output,
 - The **First 5** output values from **1 to 5** shows the **Highest Average delivery time (In Days)**.
 - The **Last 5** output values from **6-10** shows the **Lowest Average delivery time (In Days)**.

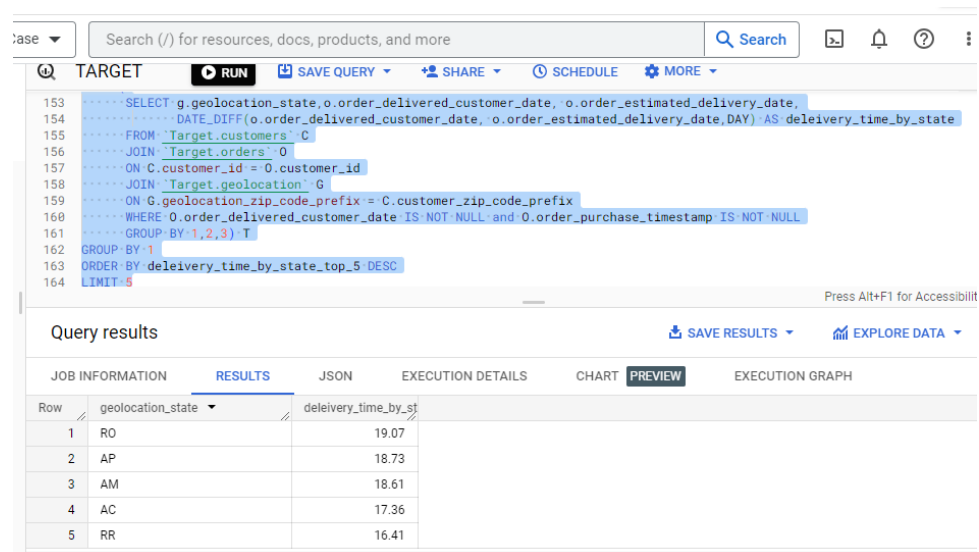
5.4 Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery.

QUERY:

```
SELECT geolocation_state, ROUND(AVG(deleivery_time_by_state),2) AS
deleivery_time_by_state_top_5

FROM (
    SELECT g.geolocation_state,o.order_delivered_customer_date, o.order_estimated_delivery_date,
    DATE_DIFF(o.order_delivered_customer_date, o.order_estimated_delivery_date,DAY) AS
deleivery_time_by_state
    FROM `Target.customers` C
    JOIN `Target.orders` O
    ON C.customer_id = O.customer_id
    JOIN `Target.geolocation` G
    ON G.geolocation_zip_code_prefix = C.customer_zip_code_prefix
    WHERE O.order_delivered_customer_date IS NOT NULL and O.order_purchase_timestamp IS NOT NULL
    GROUP BY 1,2,3) T
GROUP BY 1
ORDER BY deleivery_time_by_state_top_5 DESC
LIMIT 5
```

OUTPUT:



The screenshot shows a SQL query execution interface. The query is displayed in a text area, and the results are shown in a table below. The table has two columns: 'geolocation_state' and 'deleivery_time_by_st'. The results are sorted by 'deleivery_time_by_st' in descending order, showing the top 5 states.

Row	geolocation_state	deleivery_time_by_st
1	RO	19.07
2	AP	18.73
3	AM	18.61
4	AC	17.36
5	RR	16.41

INFERENCE:

- After the analysis o dataset, the **Top 5 States** where the **order delivery** is **really fast** as compared to the estimated date of delivery are shown in the above output image.

6. Analysis based on the payments

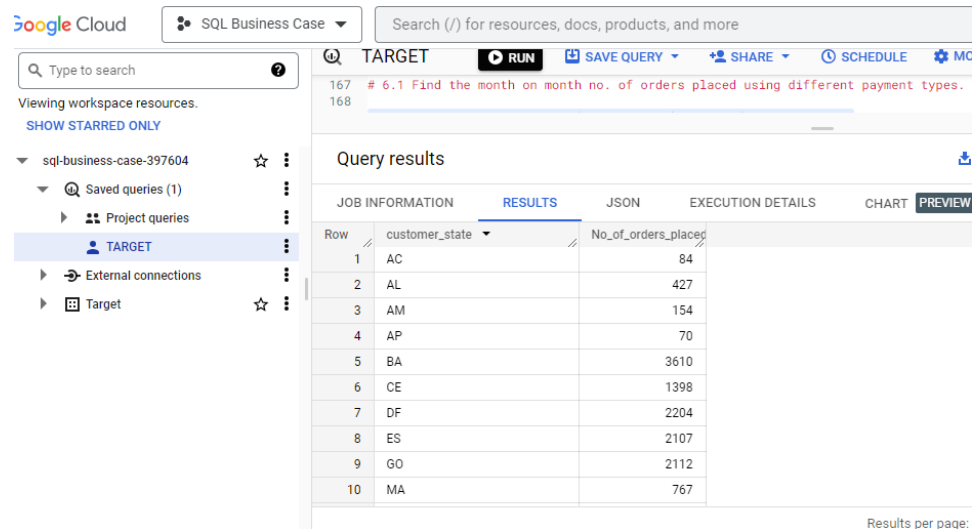
6.1 Find the month on month no. of orders placed using different payment types.

QUERY:

```
SELECT customer_state, COUNT(payment_type) AS No_of_orders_placed

FROM `Target.customers` C
JOIN `Target.orders` O
ON C.customer_id = O.customer_id
JOIN `Target.payments` P
ON P.order_id = O.order_id
GROUP BY 1
ORDER BY 1
```

OUTPUT:



The screenshot shows the Google Cloud SQL Business Case interface. On the left, there's a sidebar with a search bar and a list of workspace resources. The main area displays a query titled 'TARGET' with a 'RUN' button. Below the query, the 'Query results' section shows a table with 10 rows of data. The table has two columns: 'customer_state' and 'No_of_orders_placed'. The data is as follows:

Row	customer_state	No_of_orders_placed
1	AC	84
2	AL	427
3	AM	154
4	AP	70
5	BA	3610
6	CE	1398
7	DF	2204
8	ES	2107
9	GO	2112
10	MA	767

INFERENCE:

- After analyzing the dataset, the **Number of orders** placed using **different payment methods** in each state on month on month in the above Image.
 - No. of orders are grouped by **Month on Month** in each state.

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

QUERY:

```
SELECT COUNT(order_id) AS No_of_orders_placed
FROM `Target.payments`
WHERE payment_installments > 0
```

OUTPUT:

The screenshot displays the Google Cloud SQL Business Case interface. On the left, a sidebar shows the workspace resources, including a project named 'sql-business-case-397604' and a query named 'TARGET'. The main panel shows the query editor with the following SQL code:

```
#6.2 Find the no. of orders placed on the basis of the payment installments that have been paid.
SELECT COUNT(order_id) AS No_of_orders_placed
FROM `Target.payments`
WHERE payment_installments > 0
```

Below the query editor, the 'Query results' section is visible, showing a table with one row and one column:

Row	No_of_orders_placed
1	103884

INFERENCE:

- After the analysis of dataset, the **Total Number of orders** placed on the basis of the **payment installments** that have paid is **103884**.