

Name – Pawan Rajput

Batch – SEP-2022

SQL Target case study

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

The screenshot displays the Google Cloud BigQuery console interface. On the left, a sidebar shows the project hierarchy: 'case-385417' > 'Case'. Under the 'Case' dataset, a list of tables is shown: customers, geolocation, order_items, order_reviews, orders, payments, products, and sellers. Each table has a star icon and a vertical ellipsis menu. The main panel on the right is titled 'Case' and shows 'Data set info'. The information includes:

- Data set ID: case-385417.Case
- Created: 1 May 2023, 23:03:24 UTC+5:30
- Default table expiry: 60 days
- Last modified: 1 May 2023, 23:03:24 UTC+5:30
- Data location: US
- Description: (empty)
- Default collation: (empty)
- Default rounding mode: ROUNDING_MODE_UNSPECIFIED
- Case insensitive: false
- Labels: (empty)
- Tags: (empty)

1.1 Data type of columns in a table.

Create the Database in Big Query.

Table: **customers**

The screenshot shows the 'customers' table schema in the Google Cloud BigQuery console. The table has the following columns:

Field name	Type	Mode	Collation	Default value	Policy tags	Description
customer_id	STRING	NULLABLE				
customer_unique_id	STRING	NULLABLE				
customer_zip_code_prefix	INTEGER	NULLABLE				
customer_city	STRING	NULLABLE				
customer_state	STRING	NULLABLE				

At the bottom of the schema view, there are two buttons: 'EDIT SCHEMA' and 'VIEW ROW ACCESS POLICIES'.

Table: **geolocation**

geolocation

QUERY

SHARE

COPY

SNAPSHOT

DELETE

EXPORT

SCHEMA

DETAILS

PREVIEW

LINEAGE

Filter

Enter property name or value

<input type="checkbox"/>	Field name	Type	Mode	Collation	Default value	Policy tags	Description
<input type="checkbox"/>	geolocation_zip_code_prefix	INTEGER	NULLABLE				
<input type="checkbox"/>	geolocation_lat	FLOAT	NULLABLE				
<input type="checkbox"/>	geolocation_lng	FLOAT	NULLABLE				
<input type="checkbox"/>	geolocation_city	STRING	NULLABLE				
<input type="checkbox"/>	geolocation_state	STRING	NULLABLE				

EDIT SCHEMA

VIEW ROW ACCESS POLICIES

Table: **order_items**

order_items

QUERY

SHARE

COPY

SNAPSHOT

DELETE

EXPORT

SCHEMA

DETAILS

PREVIEW

LINEAGE

Filter

Enter property name or value

<input type="checkbox"/>	Field name	Type	Mode	Collation	Default value	Policy tags	Description
<input type="checkbox"/>	order_id	STRING	NULLABLE				
<input type="checkbox"/>	order_item_id	INTEGER	NULLABLE				
<input type="checkbox"/>	product_id	STRING	NULLABLE				
<input type="checkbox"/>	seller_id	STRING	NULLABLE				
<input type="checkbox"/>	shipping_limit_date	TIMESTAMP	NULLABLE				
<input type="checkbox"/>	price	FLOAT	NULLABLE				
<input type="checkbox"/>	freight_value	FLOAT	NULLABLE				

EDIT SCHEMA

VIEW ROW ACCESS POLICIES

Table: **order_reviews**

order_reviews

QUERY

SHARE

COPY

SNAPSHOT

DELETE

EXPORT

SCHEMA

DETAILS

PREVIEW

LINEAGE

Filter

Enter property name or value

<input type="checkbox"/>	Field name	Type	Mode	Collation	Default value	Policy tags	Description
<input type="checkbox"/>	review_id	STRING	NULLABLE				
<input type="checkbox"/>	order_id	STRING	NULLABLE				
<input type="checkbox"/>	review_score	INTEGER	NULLABLE				
<input type="checkbox"/>	review_comment_title	STRING	NULLABLE				
<input type="checkbox"/>	review_creation_date	TIMESTAMP	NULLABLE				
<input type="checkbox"/>	review_answer_timestamp	TIMESTAMP	NULLABLE				

EDIT SCHEMA

VIEW ROW ACCESS POLICIES

Table: **orders**

orders

QUERYSHARECOPYSNAPSHOTDELETEDELETEEXPORT

SCHEMADETAILSPREVIEWLINEAGE

Filter

Enter property name or value

	Field name	Type	Mode	Collation	Default value	Policy tags	Description
<input type="checkbox"/>	order_id	STRING	NULLABLE				
<input type="checkbox"/>	customer_id	STRING	NULLABLE				
<input type="checkbox"/>	order_status	STRING	NULLABLE				
<input type="checkbox"/>	order_purchase_timestamp	TIMESTAMP	NULLABLE				
<input type="checkbox"/>	order_approved_at	TIMESTAMP	NULLABLE				
<input type="checkbox"/>	order_delivered_carrier_date	TIMESTAMP	NULLABLE				
<input type="checkbox"/>	order_delivered_customer_date	TIMESTAMP	NULLABLE				
<input type="checkbox"/>	order_estimated_delivery_date	TIMESTAMP	NULLABLE				

EDIT SCHEMAVIEW ROW ACCESS POLICIES

Table: **payments**

payments

QUERYSHARECOPYSNAPSHOTDELETEDELETEEXPORT

SCHEMADETAILSPREVIEWLINEAGE

Filter

Enter property name or value

	Field name	Type	Mode	Collation	Default value	Policy tags	Description
<input type="checkbox"/>	order_id	STRING	NULLABLE				
<input type="checkbox"/>	payment_sequential	INTEGER	NULLABLE				
<input type="checkbox"/>	payment_type	STRING	NULLABLE				
<input type="checkbox"/>	payment_installments	INTEGER	NULLABLE				
<input type="checkbox"/>	payment_value	FLOAT	NULLABLE				

EDIT SCHEMAVIEW ROW ACCESS POLICIES

Table: **products**

products

QUERYSHARECOPYSNAPSHOTDELETEDELETEEXPORT

SCHEMADETAILSPREVIEWLINEAGE










Filter

Enter property name or value

	Field name	Type	Mode	Collation	Default value	Policy tags	Description
<input type="checkbox"/>	product_id	STRING	NULLABLE				
<input type="checkbox"/>	product_category	STRING	NULLABLE				
<input type="checkbox"/>	product_name_length	INTEGER	NULLABLE				
<input type="checkbox"/>	product_description_length	INTEGER	NULLABLE				
<input type="checkbox"/>	product_photos_qty	INTEGER	NULLABLE				
<input type="checkbox"/>	product_weight_g	INTEGER	NULLABLE				
<input type="checkbox"/>	product_length_cm	INTEGER	NULLABLE				
<input type="checkbox"/>	product_height_cm	INTEGER	NULLABLE				
<input type="checkbox"/>	product_width_cm	INTEGER	NULLABLE				

EDIT SCHEMAVIEW ROW ACCESS POLICIES






Table: **sellers**






	sellers	 QUERY ▾	 SHARE	 COPY	 SNAPSHOT	 DELETE	 EXPORT ▾
SCHEMA							
DETAILS							
PREVIEW							
LINEAGE							
 Filter Enter property name or value							
<input type="checkbox"/>	Field name	Type	Mode	Collation	Default value	Policy tags 	Descript
<input type="checkbox"/>	seller_id	STRING	NULLABLE				
<input type="checkbox"/>	seller_zip_code_prefix	INTEGER	NULLABLE				
<input type="checkbox"/>	seller_city	STRING	NULLABLE				
<input type="checkbox"/>	seller_state	STRING	NULLABLE				
EDIT SCHEMA							
VIEW ROW ACCESS POLICIES							

1.2 Time period for which the data is given

Query and Output

```
SELECT min(order_purchase_timestamp) AS Start_Date,  
max(order_purchase_timestamp) AS End_Date  
FROM `case-385417.Case.orders`
```

 ▾  Untitled ▾  orders ▾  1.2 Time period for which... ven ▾  +

 1.2 Time period for which t...ven  RUN  SAVE ▾  SHARE ▾  SCHEDULE ▾

```
1 SELECT min(order_purchase_timestamp) AS Start_Date,  
2 max(order_purchase_timestamp) AS End_Date  
3 FROM `case-385417.Case.orders`
```

Query results

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS	EXECUTION GRAPH	PREVIEW
Row	Start_Date	End_Date				
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC				

Conclusion:

The start order date is 4th Sep 2016 to 17th Oct 2018.

1.3 Cities and States covered in the dataset.

Query and Below Output (City)

```
SELECT distinct(geolocation_city) FROM `case-385417.Case.geolocation`
```

Untitled 2

RUNSAVESHARESCHEDULEMORE

1 SELECT distinct (geolocation_city) FROM `case-385417.Case.geolocation`
2

Query results

SAVE RESULTS

JOB INFORMATIONRESULTSJSONEXECUTION DETAILSEXECUTION GRAPHPREVIEW

Row	geolocation_city
36	campo do brito
37	ribeiropolis
38	frei paulo
39	macambira
40	pinhao
41	sao domingos
42	carira
43	são domingos
44	moita bonita
45	pinhão
46	ribeirópolis
47	sao miguel do aleixo
48	nossa senhora aparecida
49	areia branca
50	nossa senhora das dores

We have data across 8011 cities across Brazil and distinct cities is 50.

Query and Below Output (States)

Untitled 4

RUN

SAVE

SHARE

SCHEDULE

MORE

1

SELECT distinct geolocation_state FROM `case-385417.Case.geolocation`

Query results

JOB INFORMATION

RESULTS

JSON

EXECUTION DETAILS

EXECUTION GRAPH

PREVIEW

Row	geolocation_state
15	MA
16	MG
17	MS
18	MT
19	PA
20	PB
21	PE
22	PR
23	RJ
24	RN
25	RS
26	SC
27	SP







Results per

2. In-depth Exploration:

2.1 Is there a growing trend on e-commerce in Brazil? How can we describe complete scenario? Can we see some seasonality with peaks at specific months?

Query and Below Output

```
select
ROUND(sum(price)/1000000, 2) as Price_in_milions,
EXTRACT(YEAR from order_purchase_timestamp) as YEAR from `case-385417.Case.orders`ord
left join `case-385417.Case.order_items` ord_item ON ord.order_id =
ord_item.order_id
where order_status = 'delivered'
group by YEAR
order by YEAR;
```

 **Untitled 3**  RUN  SAVE  SHARE  SCHEDULE  MORE

```
1 select
2 ROUND(sum(price)/1000000, 2) as Price_in_milions,
3 EXTRACT(YEAR from order_purchase_timestamp) as YEAR from `case-385417.Case.orders`ord
4 left join `case-385417.Case.order_items` ord_item ON ord.order_id = ord_item.order_id
5 where order_status = 'delivered'
6 group by YEAR
7 order by YEAR;
8
```

Query results

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS	EXECUTION GRAPH	PREVIEW
Row	Price_in_milions	YEAR				
1	0.04	2016				
2	5.96	2017				
3	7.22	2018				

Insights , Extract Query and Output—

```
select
ROUND(sum(price)/1000000, 2) as Price_in_milions,
EXTRACT(YEAR from order_purchase_timestamp) as month from `case-
385417.Case.orders` ord
left join `case-385417.Case.order_items` ord_item ON ord.order_id =
ord_item.order_id
where order_status = 'delivered'
group by month
order by month;
```

2.1 [RUN](#) [SAVE](#) [SHARE](#) [SCHEDULE](#) [MORE](#)

```
8
9 --| Insights, extract the growth over months
10 select
11 ROUND(sum(price)/1000000, 2) as Price_in_milions,
12 FORMAT_DATE('%Y%m', order_purchase_timestamp) as MONTH
13 from `case-385417.Case.orders` ord
14 left join `case-385417.Case.order_items` ord_item
15 ON ord.order_id = ord_item.order_id
16 where order_status = 'delivered'
17 group by MONTH
18 order by MONTH;
```

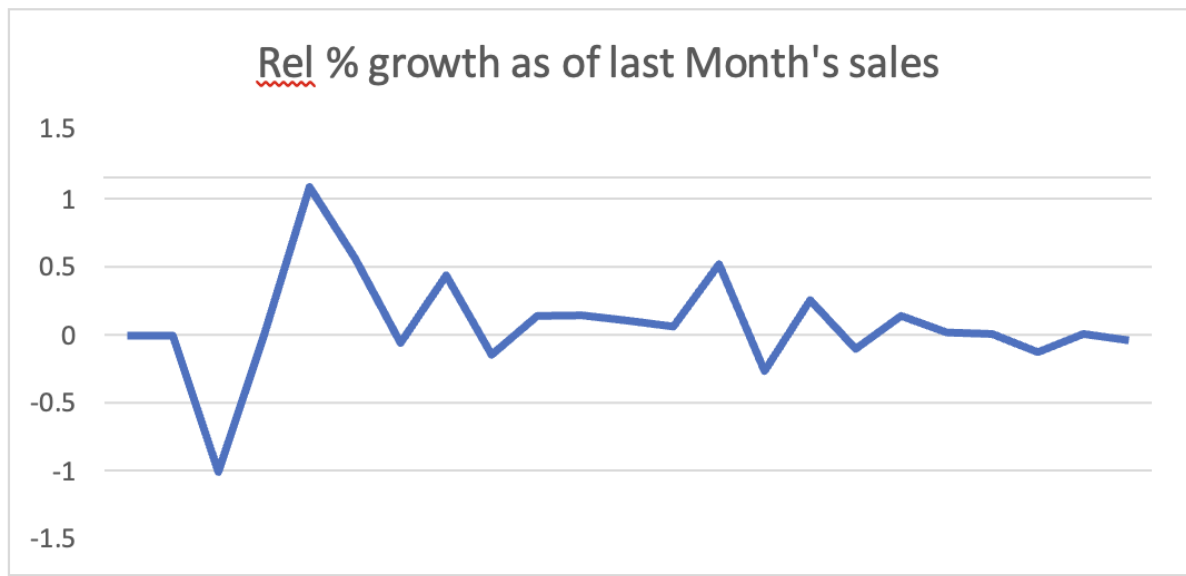
Query results [SAVE RESULT](#)

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS	EXECUTION GRAPH	PREVIEW
Row	Price_in_milions	MONTH				
15	0.73	201712				
16	0.92	201801				
17	0.83	201802				
18	0.95	201803				
19	0.97	201804				
20	0.98	201805				
21	0.86	201806				
22	0.87	201807				
23	0.84	201808				

Analysing the data using Excel, we see a 21.14% of growth over year from 2017 to 2018. Since we only have 8 months of Data in 2018, so expected yearly growth from 2017 to 2018 becomes 31.71%, which clearly shows adoption of online sales by customers in Brazil.



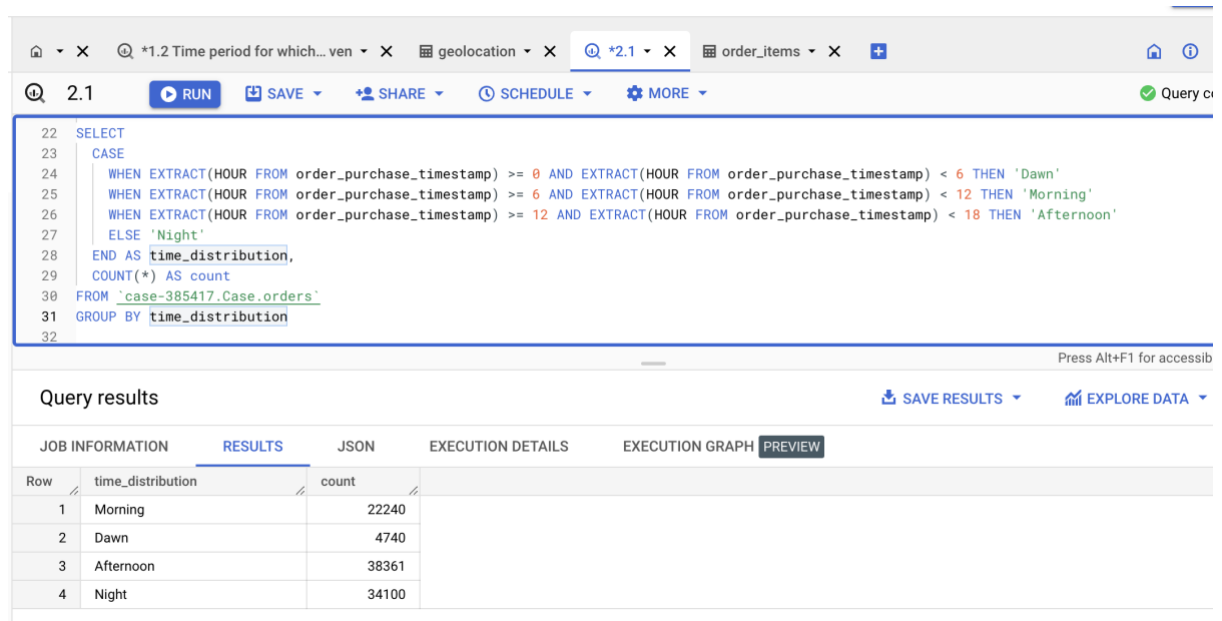
Analysing more on the seasonality, sales dependency over months.



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

Query and Output

```
SELECT
CASE
    WHEN EXTRACT(HOUR FROM order_purchase_timestamp) >= 0 AND EXTRACT(HOUR FROM
order_purchase_timestamp) < 6 THEN 'Dawn'
    WHEN EXTRACT(HOUR FROM order_purchase_timestamp) >= 6 AND EXTRACT(HOUR FROM
order_purchase_timestamp) < 12 THEN 'Morning'
    WHEN EXTRACT(HOUR FROM order_purchase_timestamp) >= 12 AND EXTRACT(HOUR FROM
order_purchase_timestamp) < 18 THEN 'Afternoon'
    ELSE 'Night'
END AS time_distribution,
COUNT(*) AS count
FROM `Case.orders`
GROUP BY time_distribution;
```



The screenshot shows a SQL query editor interface with a query window and a results table. The query window contains the following SQL code:

```
22 SELECT
23 CASE
24     WHEN EXTRACT(HOUR FROM order_purchase_timestamp) >= 0 AND EXTRACT(HOUR FROM order_purchase_timestamp) < 6 THEN 'Dawn'
25     WHEN EXTRACT(HOUR FROM order_purchase_timestamp) >= 6 AND EXTRACT(HOUR FROM order_purchase_timestamp) < 12 THEN 'Morning'
26     WHEN EXTRACT(HOUR FROM order_purchase_timestamp) >= 12 AND EXTRACT(HOUR FROM order_purchase_timestamp) < 18 THEN 'Afternoon'
27     ELSE 'Night'
28 END AS time_distribution,
29 COUNT(*) AS count
30 FROM `case-385417.Case.orders`
31 GROUP BY time_distribution
32
```

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

Row	time_distribution	count
1	Morning	22240
2	Dawn	4740
3	Afternoon	38361
4	Night	34100

Conclusion:

We can clearly see that Brazilian customers tend to buy more during the Afternoon and Night.

2. Evolution of E-commerce orders in the Brazil region:

- 1. Get month on month orders by region, states.

Query and Output

```
SELECT c.customer_state, CONCAT(EXTRACT(YEAR FROM order_purchase_timestamp), '-',  
FORMAT_DATE('%m', order_purchase_timestamp)) YEAR_MONTH, count(*) order_count  
FROM `Case.orders` o INNER JOIN `Case.customers` c ON  
o.customer_id = c.customer_id  
GROUP BY c.customer_state, YEAR_MONTH  
ORDER BY c.customer_state, YEAR_MONTH;
```

2.1 RUN SAVE SHARE SCHEDULE MORE

```
32  
33  
34  
35 SELECT c.customer_state, CONCAT(EXTRACT(YEAR FROM order_purchase_timestamp), '-',  
36 FORMAT_DATE('%m', order_purchase_timestamp)) YEAR_MONTH, count(*) order_count  
37 FROM `case-385417.Case.orders` o INNER JOIN `case-385417.Case.customers` c ON  
38 o.customer_id = c.customer_id  
39 GROUP BY c.customer_state, YEAR_MONTH  
40 ORDER BY c.customer_state, YEAR_MONTH  
41  
42
```

Query results

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS	EXECUTION GRAPH	PREVIEW
Row	customer_state	YEAR_MONTH	order_count			
42	AM	2017-02	8			
43	AM	2017-03	5			
44	AM	2017-04	13			
45	AM	2017-05	10			
46	AM	2017-06	1			
47	AM	2017-07	5			
48	AM	2017-08	5			
49	AM	2017-09	9			
50	AM	2017-10	3			

2. Distribution of customers across the states in Brazil

```
select
  cust.customer_state,
  count(*) as Total_orders
from `Case.orders` ord
left join `Case.customers` cust
ON ord.customer_id = cust.customer_id
where order_status = 'delivered'
group by cust.customer_state
order by 2 desc;
```

Query and Output

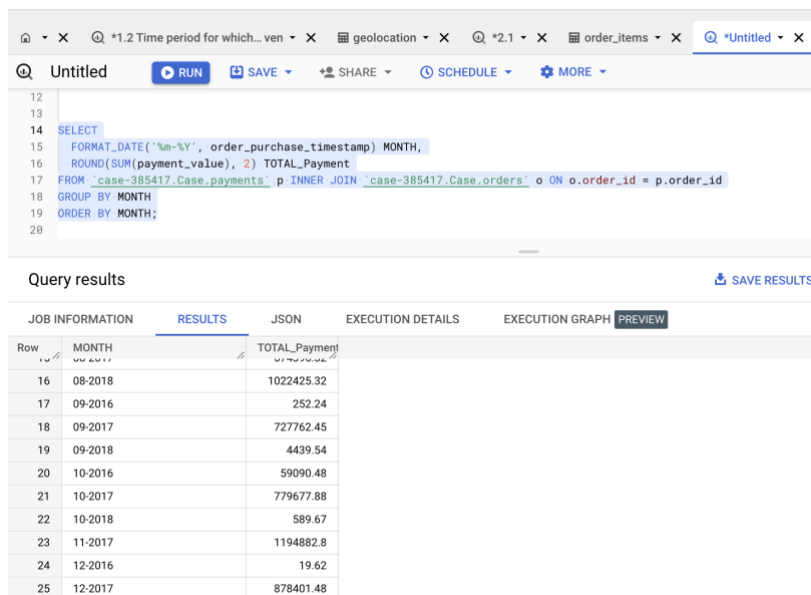
*1.2 Time period for which... ven X geolocation X *2.1 X order_items X				
Untitled RUN SAVE SHARE SCHEDULE MORE				
<pre>1 select 2 cust.customer_state, 3 count(*) as Total_orders 4 from `case-385417.Case.orders` ord 5 left join `case-385417.Case.customers` cust 6 ON ord.customer_id = cust.customer_id 7 where order_status = 'delivered' 8 group by cust.customer_state 9 order by 2 desc;</pre>				
Query results				
JOB INFORMATION RESULTS JSON EXECUTION DETAILS EXECUTION GRAPH PREVIEW				
Row	customer_state	Total_orders		
18	PI	476		
19	RN	474		
20	AL	397		
21	SE	335		
22	TO	274		
23	RO	243		
24	AM	145		
25	AC	80		
26	AP	67		
27	RR	41		

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

1. Get % increase in the cost of orders from 2017 to 2018 (include months between Jan to Aug only) - You can use the "payment value" column in the payments table.

Query and Output

```
SELECT
  FORMAT_DATE('%m-%Y', order_purchase_timestamp) MONTH,
  ROUND(SUM(payment_value), 2) TOTAL_Payment
FROM `Case.payments` p INNER JOIN `Case.orders` o ON o.order_id = p.order_id
GROUP BY MONTH
ORDER BY MONTH;
```



Query results

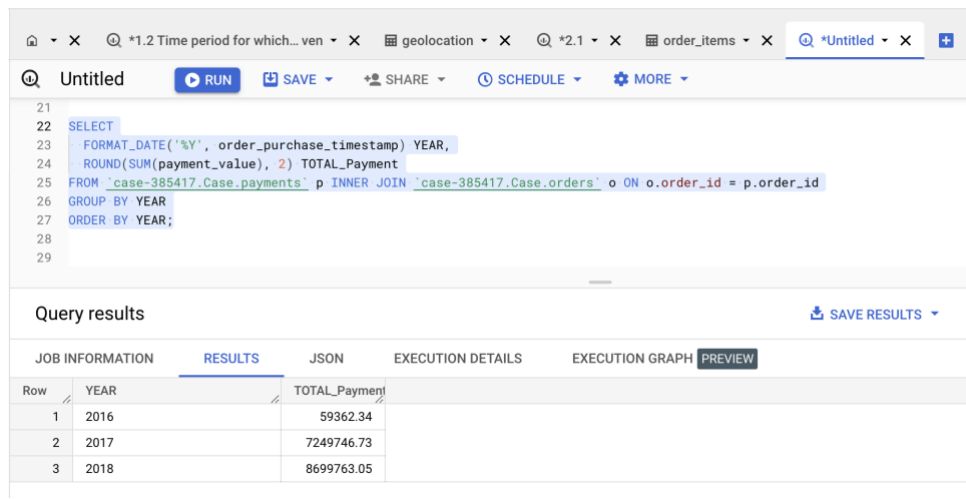
JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS	EXECUTION GRAPH	PREVIEW
Row	MONTH	TOTAL_Payment				
16	08-2018	1022425.32				
17	09-2016	252.24				
18	09-2017	727762.45				
19	09-2018	4439.54				
20	10-2016	59090.48				
21	10-2017	779677.88				
22	10-2018	589.67				
23	11-2017	1194882.8				
24	12-2016	19.62				
25	12-2017	878401.48				

As we can see there is a growing trend in the above table over the month which tells us that the total cost of orders received has increased over the year.

We can analyse the overall order sum by using the below query

Query and Output

```
SELECT
  FORMAT_DATE('%Y', order_purchase_timestamp) YEAR,
  ROUND(SUM(payment_value), 2) TOTAL_Payment
FROM `case.payments` p INNER JOIN `case.orders` o ON o.order_id = p.order_id
GROUP BY YEAR
ORDER BY YEAR;
```



The screenshot shows a SQL query editor interface. At the top, there are several tabs: '*1.2 Time period for which... ven', 'geolocation', '*2.1', 'order_items', and '*Untitled'. The 'Untitled' tab is active. Below the tabs, there is a toolbar with buttons for 'RUN', 'SAVE', 'SHARE', 'SCHEDULE', and 'MORE'. The query is entered in the editor area, and the 'Query results' section is visible below it. The results are displayed in a table with columns 'Row', 'YEAR', and 'TOTAL_Payment'.

Row	YEAR	TOTAL_Payment
1	2016	59362.34
2	2017	7249746.73
3	2018	8699763.05

Conclusion-

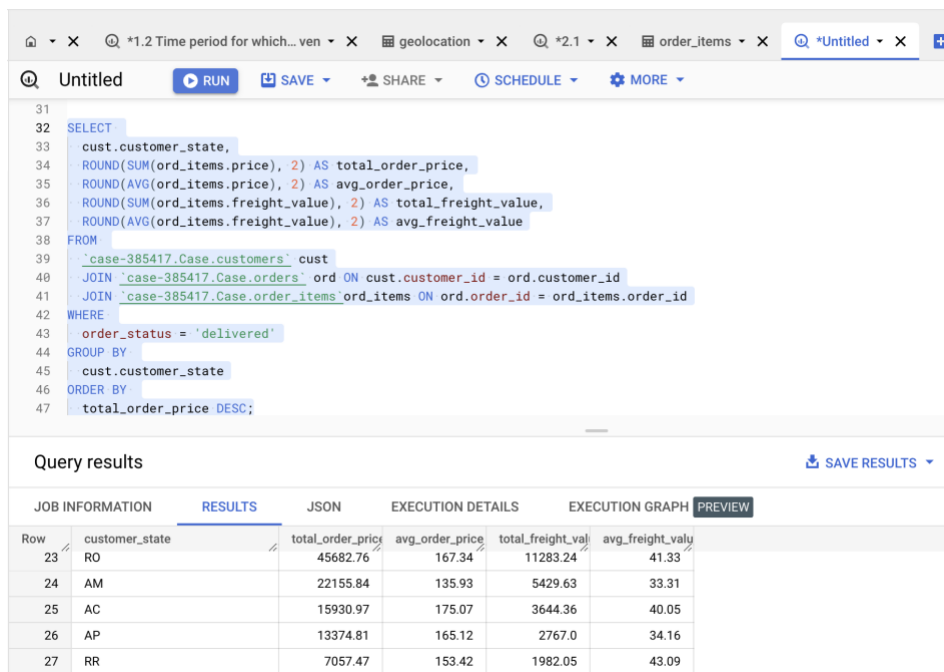
We can see that there has been a definite increase in the cost of the order over years.

We can definitely see this trend increasing over the next years as well.

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

Output and Query-

```
SELECT
    cust.customer_state,
    ROUND(SUM(ord_items.price), 2) AS total_order_price,
    ROUND(AVG(ord_items.price), 2) AS avg_order_price,
    ROUND(SUM(ord_items.freight_value), 2) AS total_freight_value,
    ROUND(AVG(ord_items.freight_value), 2) AS avg_freight_value
FROM
    `Case.customers` cust
    JOIN `Case.orders` ord ON cust.customer_id = ord.customer_id
    JOIN `Case.order_items` ord_items ON ord.order_id = ord_items.order_id
WHERE
    order_status = 'delivered'
GROUP BY
    cust.customer_state
ORDER BY
    total_order_price DESC;
```



The screenshot displays a SQL query in an editor and its results in a table. The query calculates the total and average order price and freight value for each customer state, filtered by 'delivered' status. The results table shows the following data:

Row	customer_state	total_order_price	avg_order_price	total_freight_val	avg_freight_val
23	RO	45682.76	167.34	11283.24	41.33
24	AM	22155.84	135.93	5429.63	33.31
25	AC	15930.97	175.07	3644.36	40.05
26	AP	13374.81	165.12	2767.0	34.16
27	RR	7057.47	153.42	1982.05	43.09

4. Analysis on sales, freight and delivery time

1. Calculate days between purchasing, delivering and estimated delivery

Output and Query-

```
SELECT
    DATE_DIFF(DATE(order_estimated_delivery_date),
    DATE(order_purchase_timestamp), DAY) AS Expected_Days_to_delivery,
    DATE_DIFF(DATE(order_delivered_customer_date),
    DATE(order_purchase_timestamp), DAY) AS Day_after_delivered,
    DATE_DIFF(DATE(order_estimated_delivery_date),
    DATE(order_delivered_customer_date), DAY) Estimated_VS_Delivered,
    CASE
        WHEN DATE_DIFF(DATE(order_estimated_delivery_date),
    DATE(order_delivered_customer_date), DAY) > 0 THEN "Early Delivery"
        WHEN DATE_DIFF(DATE(order_estimated_delivery_date),
    DATE(order_delivered_customer_date), DAY) = 0 THEN "Delivered on Time"
        ELSE "Late Delivery"
    END Delivery_Status
FROM `Case.orders`
WHERE order_delivered_customer_date IS NOT NULL;
```

Query results

JOB INFORMATION	RESULTS	JSON	EXECUTION DETAILS	EXECUTION GRAPH	PREVIEW
Row 42	Expected_Days_to_delivery: 40	Day_after_delivered: 10	Estimated_VS_Delivered: 30	Delivery_Status: Early Delivery	
43	28	40	-12	Late Delivery	
44	29	35	-6	Late Delivery	
45	23	28	-5	Late Delivery	
46	33	39	-6	Late Delivery	
47	22	28	-6	Late Delivery	
48	16	21	-5	Late Delivery	
49	23	35	-12	Late Delivery	
50	21	29	-8	Late Delivery	

We can also analyse the trend in the delivery status by using the following query

Output and Query-

```
SELECT Delivery_Status, COUNT(*) Delivery_Status_Count
FROM (SELECT
    DATE_DIFF(DATE(order_estimated_delivery_date),
DATE(order_purchase_timestamp), DAY) AS Expected_Days_to_delivery,
    DATE_DIFF(DATE(order_delivered_customer_date),
DATE(order_purchase_timestamp), DAY) AS Day_after_delivered,
    DATE_DIFF(DATE(order_estimated_delivery_date),
DATE(order_delivered_customer_date), DAY) Estimated_VS_Delivered,
    CASE
        WHEN DATE_DIFF(DATE(order_estimated_delivery_date),
DATE(order_delivered_customer_date), DAY) > 0 THEN "Early Delivery"
        WHEN DATE_DIFF(DATE(order_estimated_delivery_date),
DATE(order_delivered_customer_date), DAY) = 0 THEN "Delivered on Time"
        ELSE "Late Delivery"
    END Delivery_Status
FROM `Case.orders`
WHERE order_delivered_customer_date IS NOT NULL)
GROUP BY Delivery_Status
ORDER BY Delivery_Status_Count DESC
```

Untitled

RUN

SAVE

SHARE

SCHEDULE

MORE

```

78
79 SELECT Delivery_Status, COUNT(*) as Delivery_Status_Count
80 FROM (SELECT
81   DATE_DIFF(DATE(order_estimated_delivery_date), DATE(order_purchase_timestamp), DAY) AS Expected_Days_to_delivery,
82   DATE_DIFF(DATE(order_delivered_customer_date), DATE(order_purchase_timestamp), DAY) AS Day_after_delivered,
83   DATE_DIFF(DATE(order_estimated_delivery_date), DATE(order_delivered_customer_date), DAY) Estimated_VS_Delivered,
84   CASE
85     WHEN DATE_DIFF(DATE(order_estimated_delivery_date), DATE(order_delivered_customer_date), DAY) > 0 THEN "Early Delivery"
86     WHEN DATE_DIFF(DATE(order_estimated_delivery_date), DATE(order_delivered_customer_date), DAY) = 0 THEN "Delivered on Time"
87     ELSE "Late Delivery"
88   END Delivery_Status
89 FROM `case-385417.Case.orders`
90 WHERE order_delivered_customer_date IS NOT NULL)
91 GROUP BY Delivery_Status
92 ORDER BY Delivery_Status_Count DESC

```

Query results

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS	EXECUTION GRAPH	PREVIEW
Row	Delivery_Status	Delivery_Status_Count				
1	Early Delivery	88649				
2	Late Delivery	6535				
3	Delivered on Time	1292				

Conclusion- We can see that most of the orders are delivered before the expected delivery date so we can also adjust the expected delivery date shown to the customer.

2. Find time_to_delivery & diff_estimated_delivery. The formula for the same is given below:

- $\text{time_to_delivery} = \text{order_purchase_timestamp} - \text{order_delivered_customer_date}$
- $\text{diff_estimated_delivery} = \text{order_estimated_delivery_date} - \text{order_delivered_customer_date}$

Output and Query-

```
SELECT
    DATE_DIFF(DATE(order_delivered_customer_date),
DATE(order_purchase_timestamp), DAY) AS Days_to_delivery,
    DATE_DIFF(DATE(order_estimated_delivery_date),
DATE(order_delivered_customer_date), DAY) AS Delivery_delta
FROM `Case.orders`
WHERE order_delivered_customer_date IS NOT NULL;
```

Q	Untitled	RUN	SAVE	SHARE	SCHEDULE	MORE
97						
98						
99						
100						
101	SELECT					
102	DATE_DIFF(DATE(order_delivered_customer_date), DATE(order_purchase_timestamp), DAY) AS Time_to_delivery ,					
103	DATE_DIFF(DATE(order_estimated_delivery_date), DATE(order_delivered_customer_date), DAY) AS Diff_estimated_delivery					
104	FROM `case-385417.Case.orders`					
105	WHERE order_delivered_customer_date IS NOT NULL;					
106						
107						
108						
109						
110						
111						

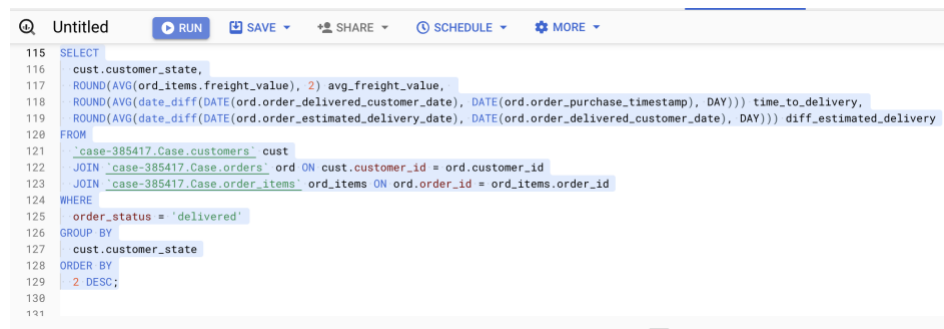
Query results

JOB INFORMATION	RESULTS	JSON	EXECUTION DETAILS	EXECUTION GRAPH	PREVIEW
Row	Time_to_delivery	Diff_estimated_delivery			
9	37	-1			
10	34	-5			
11	39	-6			
12	36	-2			
13	34	0			
14	43	-11			
15	35	-3			
16	33	-7			

- Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery

Output and Query-

```
SELECT
  cust.customer_state,
  ROUND(AVG(ord_items.freight_value), 2) avg_freight_value,
  ROUND(AVG(date_diff(DATE(ord.order_delivered_customer_date),
DATE(ord.order_purchase_timestamp), DAY))) time_to_delivery,
  ROUND(AVG(date_diff(DATE(ord.order_estimated_delivery_date),
DATE(ord.order_delivered_customer_date), DAY))) diff_estimated_delivery
FROM
  `Case.customers` cust
  JOIN `Case.orders` ord ON cust.customer_id = ord.customer_id
  JOIN `Case.order_items` ord_items ON ord.order_id = ord_items.order_id
WHERE
  order_status = 'delivered'
GROUP BY
  cust.customer_state
ORDER BY
  2 DESC;
```



The screenshot shows a SQL query editor with a toolbar at the top containing buttons for RUN, SAVE, SHARE, SCHEDULE, and MORE. The query is displayed in a monospace font with syntax highlighting. Line numbers 115 through 131 are visible on the left margin.

Query results

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS		EXECUTION GRAPH	PREVIEW
Row	customer_state	avg_freight_value	time_to_delivery	diff_estimated_delivery			
1	PB	43.09	21.0	13.0			
2	RR	43.09	28.0	18.0			
3	RO	41.33	20.0	20.0			
4	AC	40.05	21.0	21.0			
5	PI	39.12	19.0	12.0			
6	MA	38.49	22.0	10.0			
7	TO	37.44	17.0	12.0			

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

Output and Query-

```
select
  cust.customer_state,
  ROUND(AVG(ord_items.freight_value), 2) as avg_freight_value
from `Case.customers` cust
join `Case.orders` ord
ON cust.customer_id = ord.customer_id
join `Case.order_items` ord_items
ON ord.order_id = ord_items.order_id
where order_status = 'delivered'
group by cust.customer_state
order by 2 desc
limit 5;
```

```
156 WHERE order_delivered_customer_date IS NOT NULL;
157
158
159 select
160   cust.customer_state,
161   ROUND(AVG(ord_items.freight_value), 2) as avg_freight_value
162 from `Case.customers` cust
163 join `Case.orders` ord
164 ON cust.customer_id = ord.customer_id
165 join `Case.order_items` ord_items
166 ON ord.order_id = ord_items.order_id
167 where order_status = 'delivered'
168 group by cust.customer_state
169 order by 2 desc
170 limit 5;
171
172
173
```

Query results

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS	EXECUTION GRAPH	PREVIEW
Row	customer_state	avg_freight_valu				
1	PB	43.09				
2	RR	43.09				
3	RO	41.33				
4	AC	40.05				
5	PI	39.12				

4.2 Top 5 states with highest/lowest average time to delivery

Output and Output and Query-highest

```
select cust.customer_state,
       ROUND(AVG(date_diff(DATE(order_delivered_customer_date),
                           DATE(order_purchase_timestamp), DAY))) as time_to_delivery
from `Case.customers` cust
join `case-385417.Case.orders` ord
ON cust.customer_id = ord.customer_id
where order_status = 'delivered'
group by cust.customer_state
order by 2 desc
limit 5;
```

Untitled 2

RUNSAVESHARESCHEDULEMORE

```
1
2
3 select
4 cust.customer_state,
5 ROUND(AVG(date_diff(DATE(ord.order_delivered_customer_date), DATE(ord.order_purchase_timestamp), DAY))) as time_to_delivery
6 from `Case.customers` cust
7 join `case-385417.Case.orders` ord
8 ON cust.customer_id = ord.customer_id
9 where order_status = 'delivered'
10 group by cust.customer_state
11 order by 2 desc
12 limit 5;
13
```

Query results

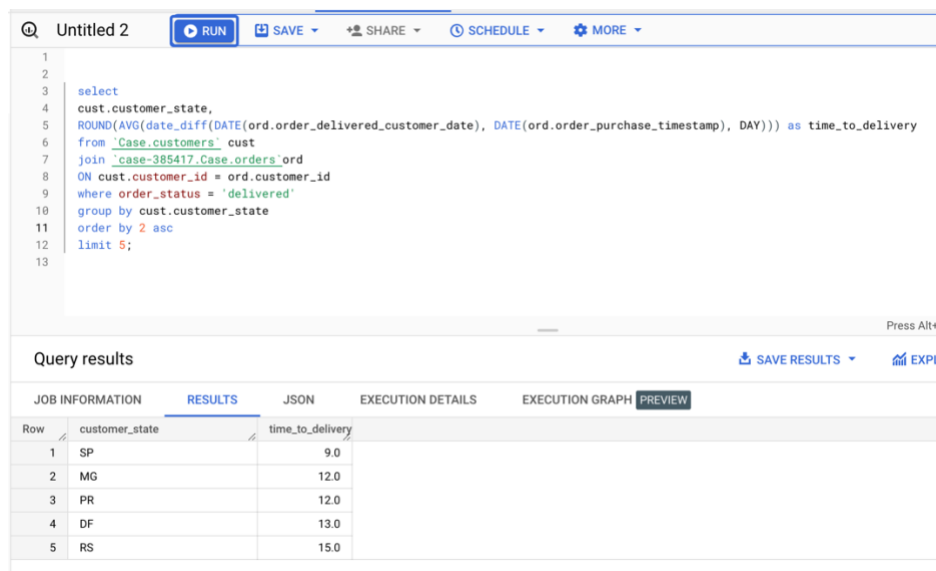
SAVESRESULTSEXP

JOB INFORMATIONRESULTSJSONEXECUTION DETAILSEXECUTION GRAPHPREVIEW

Row	customer_state	time_to_delivery
1	RR	29.0
2	AP	27.0
3	AM	26.0
4	AL	25.0
5	PA	24.0

Output and Query-lowest

```
select
cust.customer_state,
ROUND(AVG(date_diff(DATE(order_delivered_customer_date),
DATE(order_purchase_timestamp), DAY))) as time_to_delivery
from `Case.customers` cust
join `case-385417.Case.orders` ord
ON cust.customer_id = ord.customer_id
where order_status = 'delivered'
group by cust.customer_state
order by 2 asc
limit 5;
```



Query results

Row	customer_state	time_to_delivery
1	SP	9.0
2	MG	12.0
3	PR	12.0
4	DF	13.0
5	RS	15.0

Conclusion : States having faster delivery or slow/delayed delivery of orders :

Here if the orders are delivered before estimated delivery date, we can say that state has faster delivery or vice-versa.

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

Output and Query-

```
select
cust.customer_state,
ROUND(AVG(date_diff(DATE(order_estimated_delivery_date),
DATE(order_delivered_customer_date), DAY))) as diff_estimated_delivery
from `Case.customers` cust
join `case-385417.Case.orders` ord
ON cust.customer_id = ord.customer_id
where order_status = 'delivered'
group by cust.customer_state
order by 2 asc
limit 5;

--diff_estimated_delivery = order_estimated_delivery_date-
order_delivered_customer_date
```

Q Untitled 2 RUN SAVE SHARE SCHEDULE MORE Query c

```
1
2
3 select
4 cust.customer_state,
5 ROUND(AVG(date_diff(DATE(order_estimated_delivery_date), DATE(order_delivered_customer_date), DAY))) as diff_estimated_delivery
6 from `Case.customers` cust
7 join `case-385417.Case.orders` ord
8 ON cust.customer_id = ord.customer_id
9 where order_status = 'delivered'
10 group by cust.customer_state
11 order by 2 asc
12 limit 5;
13
14
15 ---diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date
```

Press Alt+F1 for accessibility

Query results SAVE RESULTS EXPLORE DATA

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS	EXECUTION GRAPH	PREVIEW
Row	customer_state	diff_estimated_c				
1	AL	9.0				
2	ES	10.0				
3	MA	10.0				
4	SE	10.0				
5	SP	11.0				

6. Payment type analysis:

6.1 Month over Month count of orders for different payment types

Output and Query-

```
select
FORMAT_DATE('%Y%m',order_purchase_timestamp) ,
pay.payment_type, count(*) as Total_orders
from `Case.payments` as pay
join `Case.orders` as ord
ON ord.order_id = pay.order_id
group by order_purchase_timestamp, pay.payment_type
order by order_purchase_timestamp, Total_orders desc;
```

Untitled 2

RUNSAVESHARESCHEDULEMORE

```
1 select
2 FORMAT_DATE('%Y%m',order_purchase_timestamp) ,
3 pay.payment_type, count(*) as Total_orders
4 from `Case.payments` as pay
5 join `Case.orders` as ord
6 ON ord.order_id = pay.order_id
7 group by order_purchase_timestamp, pay.payment_type
8 order by order_purchase_timestamp, Total_orders desc
```


Query results


JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS	EXECUTION GRAPH	PREVIEW
Row	f0_	payment_type	Total_orders			
1	201609	credit_card	1			
2	201609	credit_card	1			
3	201609	credit_card	1			
4	201610	credit_card	1			
5	201610	UPI	1			
6	201610	UPI	1			
7	201610	credit_card	1			
8	201610	UPI	1			
9	201610	credit_card	1			
10	201610	credit_card	1			


6.2 Distribution of payment installments and count of orders:


Output and Query-


```
select
pay.payment_installments ,count(*) as orders
from `Case.payments` pay
group by pay.payment_installments;
```

 2023-05-02 22:27:54 -ALL

 RUN

 SAVE ▾

 SHARE ▾

 SCHEDULE ▾

328

329 select

330 pay.payment_installments ,count(*) as orders

331 from `Case.payments` pay

332 group by pay.payment_installments;

333

Query results

JOB INFORMATIONRESULTSJSONEXECUTION DETAILSEXECUTION GRAPHPREVIEW

Row	payment_installments	orders
1	0	2
2	1	52546
3	2	12413
4	3	10461
5	4	7098
6	5	5239
7	6	3920
8	7	1626
9	8	4268
10	9	644
11	10	5328
12	11	23
13	12	133