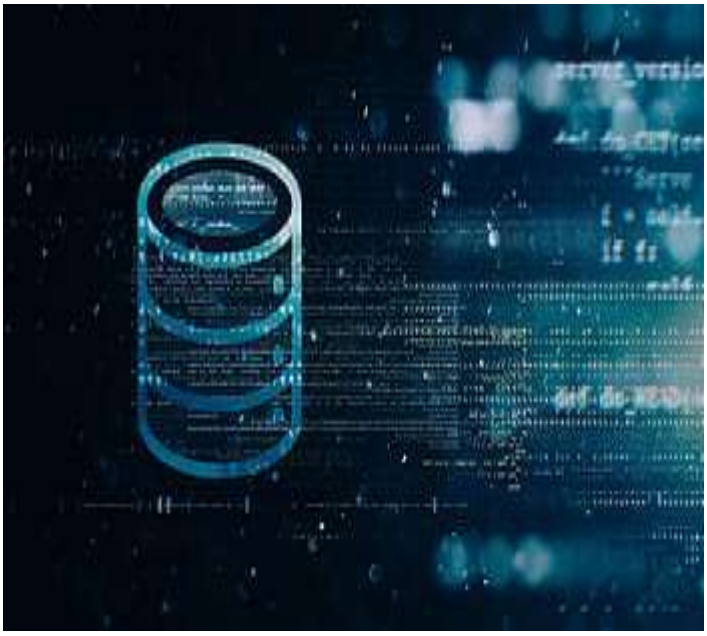


TARGET SQL CASE STUDY

Case study 1

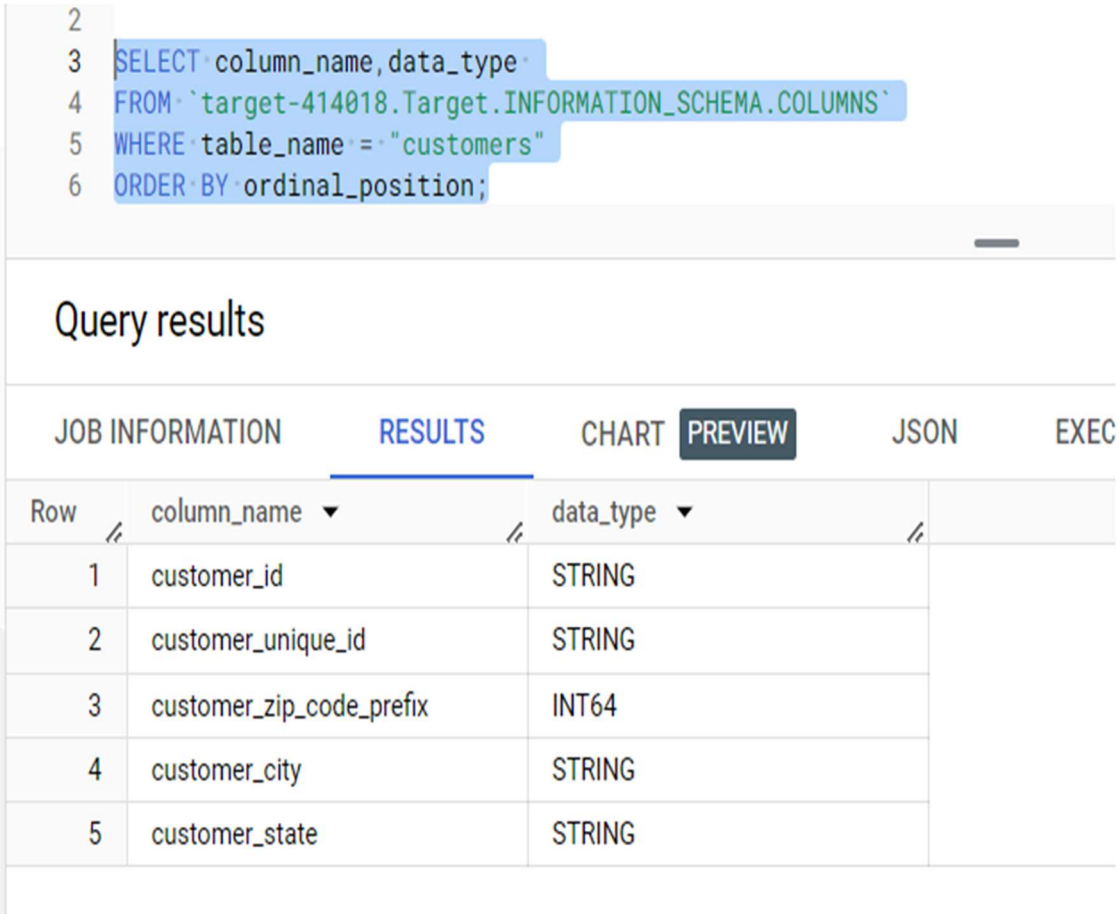
ABSTRACT

Target is a globally renowned brand, a preferred shopping destination, this business case focuses on the operations of Target in Brazil and provides insightful information

Rohit Lanjewar

1. **Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset:**
    1. Data type of all columns in the "customers" table.

```
SELECT column_name,data_type
FROM `target-414018.Target.INFORMATION_SCHEMA.COLUMNS`
WHERE table_name = "customers"
ORDER BY ordinal_position;
```



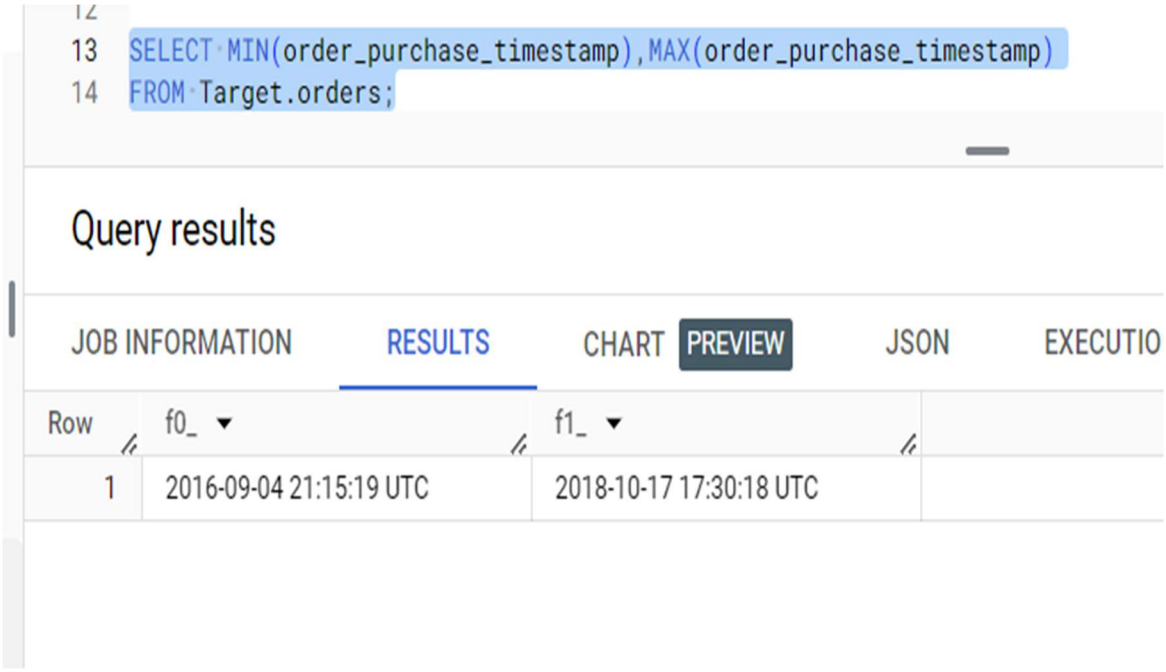
Query results

JOB INFORMATION | RESULTS | CHART | PREVIEW | JSON | EXEC

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

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

```
SELECT
MIN(order_purchase_timestamp),MAX(order_purchase_timest
amp)
FROM Target.orders;
```



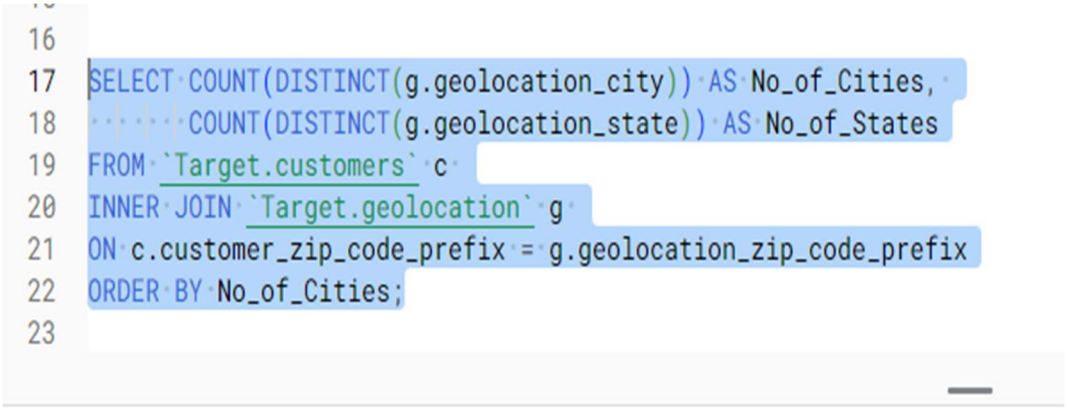
The **First order** as per the data was placed on.
"2016-09-04 21:15:19 UTC".
The **Last Order** was placed on "2018-10-17 17:30:18 UTC".

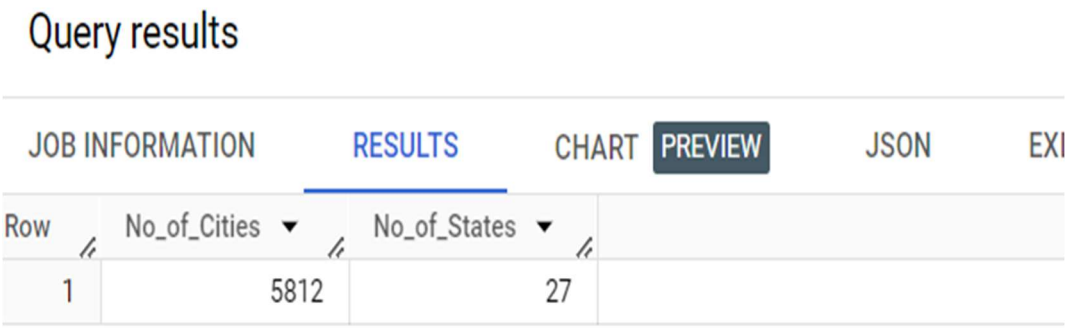
The Time Range is BETWEEN:

"2016-09-04 21:15:19 UTC" and "2018-10-17 17:30:18 UTC".

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

```
SELECT COUNT(DISTINCT(g.geolocation_city)) AS
No_of_Cities,
    COUNT(DISTINCT(g.geolocation_state)) AS No_of_States
FROM `Target.customers` c
INNER JOIN `Target.geolocation` g
ON c.customer_zip_code_prefix =
g.geolocation_zip_code_prefix
ORDER BY No_of_Cities;
```





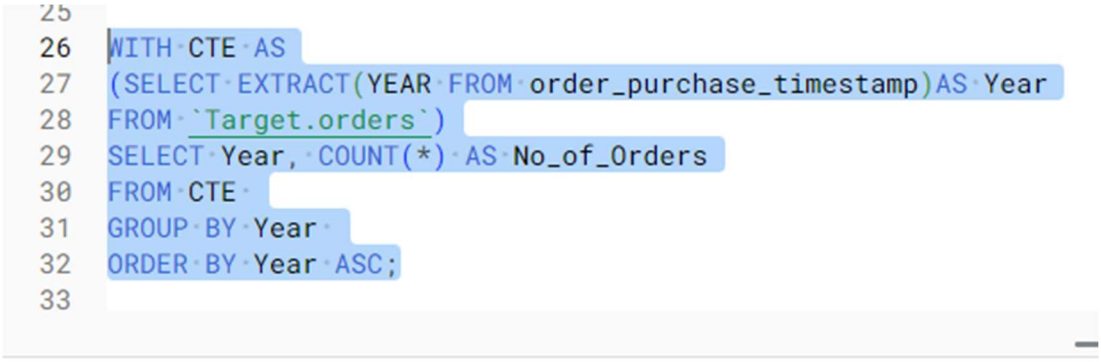
Number of Distinct Cities= 5812

Number of Distinct States to which these cities belonged = 27

2. In-depth Exploration

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

```
WITH CTE AS
(SELECT EXTRACT(YEAR FROM order_purchase_timestamp)AS
Year
FROM `Target.orders` )
SELECT Year, COUNT(*) AS No_of_Orders
FROM CTE
GROUP BY Year
ORDER BY Year ASC;
```



Query results

JOB INFORMATION | RESULTS | CHART | PREVIEW | JSON

Row	Year	No_of_Orders
1	2016	329
2	2017	45101
3	2018	54011

There is a significant difference between the number of orders placed between 2016 and 2017.
There is approximately an increase of 10000 orders between 2017 and 2018.

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

```
WITH CTE AS
(SELECT EXTRACT(MONTH FROM
order_purchase_timestamp)AS Month_No,
   FORMAT_DATETIME("%B",DATETIME(order_purchase_timest
amp)) as Month
FROM `Target.orders` )
SELECT Month,COUNT(*) AS No_of_Orders
FROM CTE
GROUP BY Month_No,Month
ORDER BY Month_No;
```

Query results

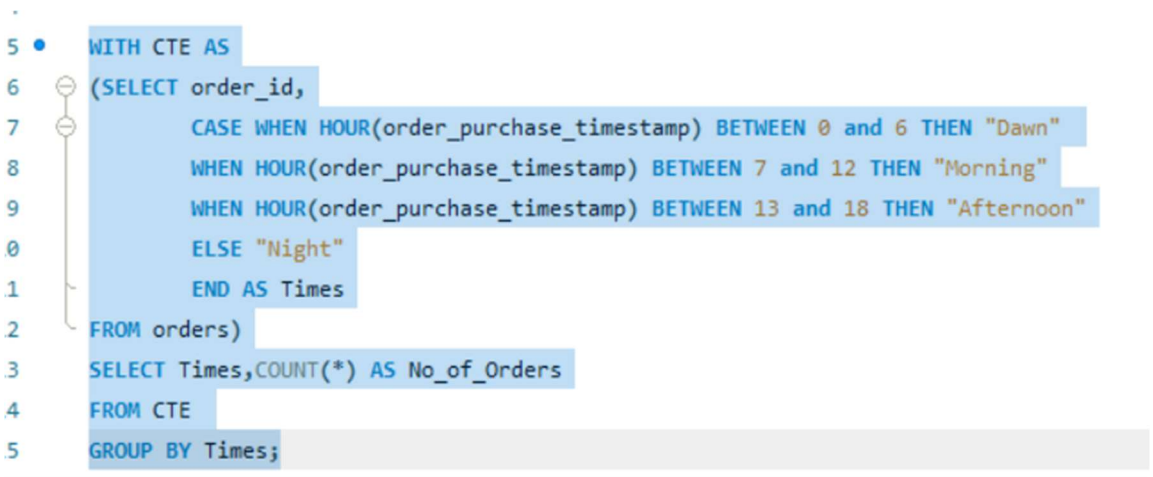
JOB INFORMATION | RESULTS | CHART PREVIEW

Row	Month	No_of_Orders
1	January	8069
2	February	8508
3	March	9893
4	April	9343
5	May	10573
6	June	9412
7	July	10318
8	August	10843
9	September	4305
10	October	4959
11	November	7544
12	December	5674

From the above query result we can see that **March- August** are the **months of maximum orders**, whereas **Maximum orders** were placed in **August**.
From **September-March**, we see **a sudden downfall** on the orders placed and it **gradually increased** by the month of March.

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

```
WITH CTE AS
(SELECT order_id,
        CASE WHEN HOUR(order_purchase_timestamp) BETWEEN 0 and 6 THEN "Dawn"
     WHEN HOUR(order_purchase_timestamp) BETWEEN 7 and 12 THEN "Morning"
     WHEN HOUR(order_purchase_timestamp) BETWEEN 13 and 18 THEN "Afternoon"
     ELSE "Night"
     END AS Times
FROM orders)
SELECT Times,COUNT(*) AS No_of_Orders
FROM CTE
GROUP BY Times;
```



sult Grid | Filter Rows: | Export: | Wrap Cell Content:

Times	No_of_Orders
Morning	27733
Night	28331
Afternoon	38135
Dawn	5242

From the above results we can see that maximum number of orders are placed in the afternoon i.e. 38135.

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

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

```
WITH CTE AS
(SELECT EXTRACT(MONTH FROM order_purchase_timestamp)AS
Month_No,FORMAT_DATETIME("%B",DATETIME(o.order_purchas
e_timestamp)) as Month,o.order_id,
 g.geolocation_state AS State
FROM `Target.customers` c
INNER JOIN `Target.geolocation` g
ON c.customer_zip_code_prefix = g.geolocation_zip_code_prefix
INNER JOIN `Target.orders` o
ON c.customer_id = o.customer_id
ORDER BY Month_No ASC)
SELECT State,Month_No,Month,COUNT(order_id) AS
No_of_Orders
FROM CTE
GROUP BY State,Month_No,Month
ORDER BY Month_No,State;
```

JOB INFORMATION | RESULTS | CHART | PREVIEW | JSON | EXECUTION DETAILS | EXECUTIO

Row	State	Month_No	Month	No_of_Orders
1	AC	1	January	694
2	AL	1	January	3645
3	AM	1	January	392
4	AP	1	January	952
5	BA	1	January	32144
6	CE	1	January	4818
7	DF	1	January	7463
8	ES	1	January	23258
9	GO	1	January	10606
10	MA	1	January	4008
11	MG	1	January	246203
12	MS	1	January	5415
13	MT	1	January	14279

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAILS | EXECUTIO

Row	State	Month_No	Month	No_of_Orders
310	PA	12	December	5303
311	PB	12	December	2181
312	PE	12	December	8860
313	PI	12	December	939
314	PR	12	December	33280
315	RJ	12	December	182098
316	RN	12	December	1289
317	RO	12	December	1094
318	RS	12	December	40636
319	SC	12	December	26059
320	SE	12	December	1879
321	SP	12	December	325198
322	TO	12	December	458

2. How are the customers distributed across all the states?

```
SELECT g.geolocation_state AS State,COUNT(c.customer_id) AS
No_of_Customers
FROM `Target.customers` c
INNER JOIN `Target.geolocation` g
ON c.customer_zip_code_prefix = g.geolocation_zip_code_prefix
GROUP BY g.geolocation_state
ORDER BY g.geolocation_state ASC;
```

Query results

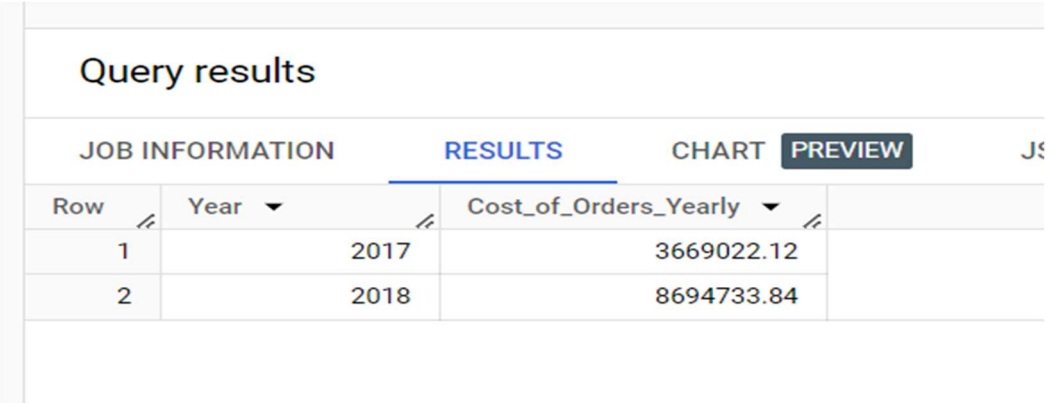
JOB INFORMATION | RESULTS | CHART PREVIEW

Row	State	No_of_Customers
1	AC	7688
2	AL	34861
3	AM	5587
4	AP	4912
5	BA	365875
6	CE	63507
7	DF	93309
8	ES	316654
9	GO	133146
10	MA	53383
11	MG	2878728

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

    1. Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only).

```
WITH CTE AS
(SELECT (SELECT EXTRACT(YEAR FROM
order_purchase_timestamp))AS Year,
    (SELECT EXTRACT(MONTH FROM
order_purchase_timestamp))AS Month,
    SUM(p.payment_value) AS Cost_of_Orders
FROM `Target.orders` o
INNER JOIN `Target.payments` p
ON o.order_id = p.order_id
GROUP BY Year,Month
HAVING Month BETWEEN 1 AND 8
ORDER BY Year,Month ASC)
SELECT Year,ROUND(SUM(Cost_of_Orders),2) AS
Cost_of_Orders_Yearly
FROM CTE
GROUP BY Year
ORDER BY Year;
```



Row	Year	Cost_of_Orders_Yearly
1	2017	3669022.12
2	2018	8694733.84

Percentage increase in cost = ((Cost_of_Orders_Yearly(2018)- Cost_of_Orders_Yearly(2017))/ Cost_of_Orders_Yearly(2017))*100

Percentage increase in cost = (8694733.84- 3669022.12)*100/ 3669022.12
Percentage increase in cost=136.97 %

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

```
SELECT g.geolocation_state AS State,
    ROUND(AVG(oi.price),2) AS Average_Price,
    ROUND(SUM(oi.price),2) AS Total_Price
FROM `Target.customers` c
INNER JOIN `Target.geolocation` g
ON c.customer_zip_code_prefix =
g.geolocation_zip_code_prefix
INNER JOIN `Target.orders` o
ON c.customer_id = o.customer_id
INNER JOIN `Target.order_items` oi
ON o.order_id = oi.order_id
GROUP BY  g.geolocation_state
ORDER BY g.geolocation_state ASC;
```

Query results

JOB INFORMATION | RESULTS | CHART | PREVIEW | JSON | EXEC

Row	State	Average_Price	Total_Price
1	AC	179.31	1494037.73
2	AL	196.64	7191886.1
3	AM	131.67	825147.21
4	AP	177.1	988578.63
5	BA	149.64	62377311.67
6	CE	151.32	10819201.81
7	DF	124.66	13141649.62
8	ES	123.36	43634878.56
9	GO	134.62	20860945.92
10	MA	150.95	9020091.01

Load more

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

```
SELECT g.geolocation_state AS State,
    ROUND(AVG(oi.freight_value),2) AS
Average_Freight_Value,
    ROUND(SUM(oi.freight_value),2) AS Total_Freight_Value
FROM `Target.customers` c
INNER JOIN `Target.geolocation` g
ON c.customer_zip_code_prefix =
g.geolocation_zip_code_prefix
INNER JOIN `Target.orders` o
ON c.customer_id = o.customer_id
INNER JOIN `Target.order_items` oi
ON o.order_id = oi.order_id
GROUP BY  g.geolocation_state
ORDER BY g.geolocation_state ASC;
```

Query results

JOB INFORMATION | RESULTS | CHART | PREVIEW | JSON | EXECUTION DET

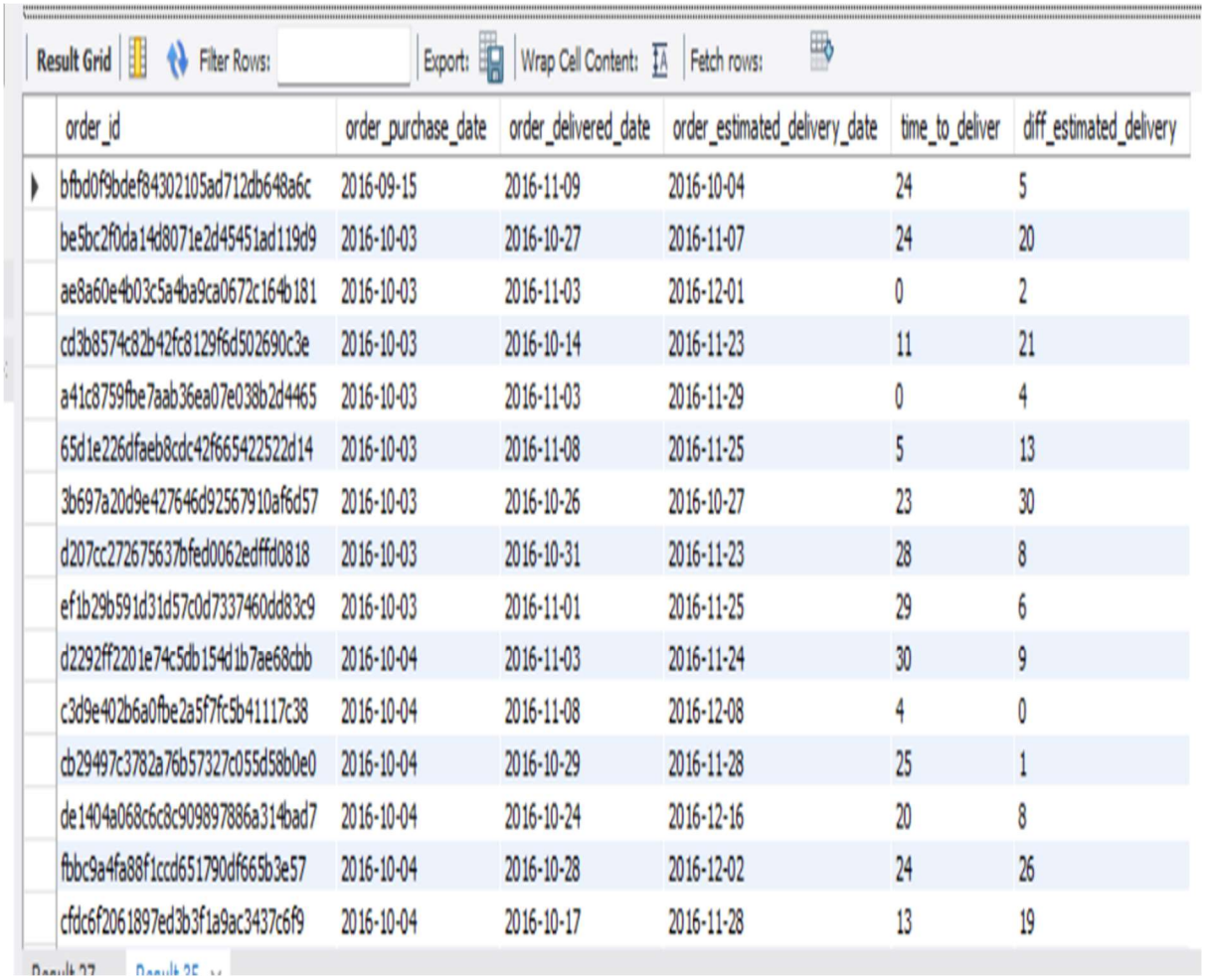
Row	State	Average_Freight_Value	Total_Freight_Value
1	AC	39.1	325767.64
2	AL	33.83	1237356.22
3	AM	34.62	216974.1
4	AP	35.66	199028.01
5	BA	27.22	11345094.0
6	CE	32.26	2306600.06
7	DF	21.01	2214955.55
8	ES	22.05	7799979.09
9	GO	23.17	3590268.56
10	MA	38.08	2275191.86
11	MG	20.46	67058347.09

5. **Analysis based on sales, freight and delivery time.**

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.

```
SELECT order_id,DATE(order_purchase_timestamp)AS
order_purchase_date
                    ,DATE(order_delivered_customer_date) AS
order_delivered_date,
      DATE(order_estimated_delivery_date) As
order_estimated_delivery_date,
          CASE WHEN DAY(order_delivered_customer_date)<
DAY(order_purchase_timestamp) AND
MONTH(order_purchase_timestamp) IN(1,3,5,7,8,10,12)
  THEN (31-
DAY(order_purchase_timestamp)+DAY(order_delivered_customer_d
ate))
                WHEN DAY(order_delivered_customer_date)<
DAY(order_purchase_timestamp) AND
MONTH(order_purchase_timestamp) =2
  THEN (28-
DAY(order_purchase_timestamp)+DAY(order_delivered_customer_d
ate))
                WHEN DAY(order_delivered_customer_date)<
DAY(order_purchase_timestamp) AND
MONTH(order_purchase_timestamp) IN(4,6,9,11)
  THEN (30-
DAY(order_purchase_timestamp)+DAY(order_delivered_customer_d
ate))
                    ELSE DAY(order_delivered_customer_date)-
DAY(order_purchase_timestamp)
      END AS time_to_deliver,
          CASE WHEN DAY(order_delivered_customer_date)<
DAY(order_estimated_delivery_date) AND
MONTH(order_estimated_delivery_date) IN(1,3,5,7,8,10,12)
```

```
      THEN (31-
DAY(order_estimated_delivery_date)+DAY(order_delivered_custome
r_date))
                        WHEN DAY(order_delivered_customer_date)<
DAY(order_estimated_delivery_date) AND
MONTH(order_estimated_delivery_date) =2
                        THEN (28-
DAY(order_estimated_delivery_date)+DAY(order_delivered_custome
r_date))
         WHEN DAY(order_delivered_customer_date)<
DAY(order_estimated_delivery_date) AND
MONTH(order_estimated_delivery_date) IN(4,6,9,11)
      THEN (30-
DAY(order_estimated_delivery_date)+DAY(order_delivered_custome
r_date))
                                ELSE DAY(order_delivered_customer_date)-
DAY(order_estimated_delivery_date)
           END AS diff_estimated_delivery
FROM orders
WHERE order_delivered_customer_date IS NOT NULL AND
order_delivered_customer_date != "" AND
order_estimated_delivery_date IS NOT NULL AND
order_estimated_delivery_date != ""
ORDER BY order_purchase_date ASC;
```



Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

order_id	order_purchase_date	order_delivered_date	order_estimated_delivery_date	time_to_deliver	diff_estimated_delivery
bfbd0f9bdef84302105ad712db648a6c	2016-09-15	2016-11-09	2016-10-04	24	5
be5bc2f0da14d8071e2d45451ad119d9	2016-10-03	2016-10-27	2016-11-07	24	20
ae8a60e4b03c5a4ba9ca0672c164b181	2016-10-03	2016-11-03	2016-12-01	0	2
cd3b8574c82b42fc8129f6d502690c3e	2016-10-03	2016-10-14	2016-11-23	11	21
a41c8759fbe7aab36ea07e038b2d4465	2016-10-03	2016-11-03	2016-11-29	0	4
65d1e226dfaeb8cdc42f665422522d14	2016-10-03	2016-11-08	2016-11-25	5	13
3b697a20d9e427646d92567910af6d57	2016-10-03	2016-10-26	2016-10-27	23	30
d207cc272675637bfed0062edfffd0818	2016-10-03	2016-10-31	2016-11-23	28	8
ef1b29b591d31d57c0d7337460dd83c9	2016-10-03	2016-11-01	2016-11-25	29	6
d2292ff2201e74c5db154d1b7ae68cbb	2016-10-04	2016-11-03	2016-11-24	30	9
c3d9e402b6a0fbe2a5f7fc5b41117c38	2016-10-04	2016-11-08	2016-12-08	4	0
cb29497c3782a76b57327c055d58b0e0	2016-10-04	2016-10-29	2016-11-28	25	1
de1404a068c6c8c909897886a314bad7	2016-10-04	2016-10-24	2016-12-16	20	8
fbbc9a4fa88f1ccd651790df665b3e57	2016-10-04	2016-10-28	2016-12-02	24	26
cfdc6f2061897ed3b3f1a9ac3437c6f9	2016-10-04	2016-10-17	2016-11-28	13	19

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

```
SELECT g.geolocation_state AS State,
    ROUND(AVG(oi.freight_value),2) AS Highest_AVG_Freight_Value,
FROM `Target.customers` c
INNER JOIN `Target.geolocation` g
ON c.customer_zip_code_prefix = g.geolocation_zip_code_prefix
INNER JOIN `Target.orders` o
ON c.customer_id = o.customer_id
INNER JOIN `Target.order_items` oi
ON o.order_id = oi.order_id
GROUP BY g.geolocation_state
ORDER BY Highest_AVG_Freight_Value DESC
LIMIT 5;
```

JOB INFORMATION | RESULTS | CHART PREVIEW | JSO

Row	State	Highest_AVG_Freight_Value
1	PB	42.77
2	RR	42.47
3	PI	39.48
4	AC	39.1
5	MA	38.08

```
SELECT g.geolocation_state AS State,
    ROUND(AVG(oi.freight_value),2) AS Lowest_AVG_Freight_Value
FROM `Target.customers` c
INNER JOIN `Target.geolocation` g
ON c.customer_zip_code_prefix = g.geolocation_zip_code_prefix
INNER JOIN `Target.orders` o
ON c.customer_id = o.customer_id
INNER JOIN `Target.order_items` oi
ON o.order_id = oi.order_id
GROUP BY g.geolocation_state
ORDER BY Lowest_AVG_Freight_Value ASC
LIMIT 5;
```

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON

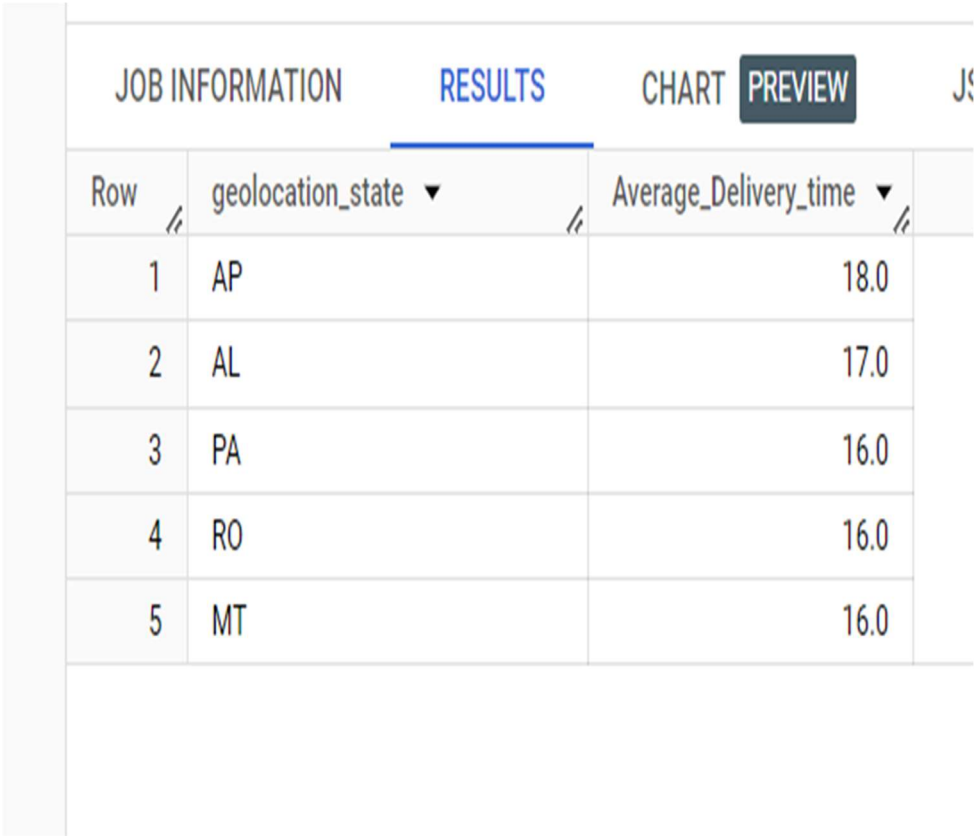
Row	State	Lowest_AVG_Freight_Value
1	SP	15.41
2	PR	20.15
3	MG	20.46
4	RJ	20.9
5	DF	21.01

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

Top 5 states with the highest average delivery time.

```
WITH CTE AS
(SELECT
order_id,geolocation_state,order_purchase_timestamp,(SELECT
EXTRACT(DAY FROM order_purchase_timestamp)) AS
Purchase_Date,
order_delivered_customer_date,
(SELECT EXTRACT(DAY FROM order_delivered_customer_date)) AS
Delivered_Date,
CASE WHEN (SELECT EXTRACT(DAY FROM
order_purchase_timestamp))> (SELECT EXTRACT(DAY FROM
order_delivered_customer_date)) AND (SELECT EXTRACT(MONTH
FROM order_purchase_timestamp)) IN(1,3,5,7,8,10,12)
THEN (31-(SELECT EXTRACT(DAY FROM
order_purchase_timestamp))+(SELECT EXTRACT(DAY FROM
order_delivered_customer_date)))
WHEN (SELECT EXTRACT(DAY FROM order_purchase_timestamp))>
(SELECT EXTRACT(DAY FROM order_delivered_customer_date)) AND
(SELECT EXTRACT(MONTH FROM order_purchase_timestamp)) =2
THEN (28-(SELECT EXTRACT(DAY FROM
order_purchase_timestamp))+(SELECT EXTRACT(DAY FROM
order_delivered_customer_date)))
WHEN (SELECT EXTRACT(DAY FROM order_purchase_timestamp))>
(SELECT EXTRACT(DAY FROM order_delivered_customer_date)) AND
(SELECT EXTRACT(MONTH FROM order_purchase_timestamp))
IN(4,6,9,11)
THEN (30-(SELECT EXTRACT(DAY FROM
order_purchase_timestamp))+(SELECT EXTRACT(DAY FROM
order_delivered_customer_date)))
ELSE (SELECT EXTRACT(DAY FROM
order_delivered_customer_date))-(SELECT EXTRACT(DAY FROM
order_purchase_timestamp))
END AS Delivery_Time
FROM `Target.orders` o
INNER JOIN `Target.customers` c
ON o.customer_id = c.customer_id
INNER JOIN `Target.geolocation` g
ON g.geolocation_zip_code_prefix= c.customer_zip_code_prefix
```

```
WHERE order_delivered_customer_date IS NOT NULL)
SELECT geolocation_state,ROUND(AVG(Delivery_Time),0) AS
Average_Delivery_time
FROM CTE
GROUP BY geolocation_state
ORDER BY Average_Delivery_time DESC
LIMIT 5;
```



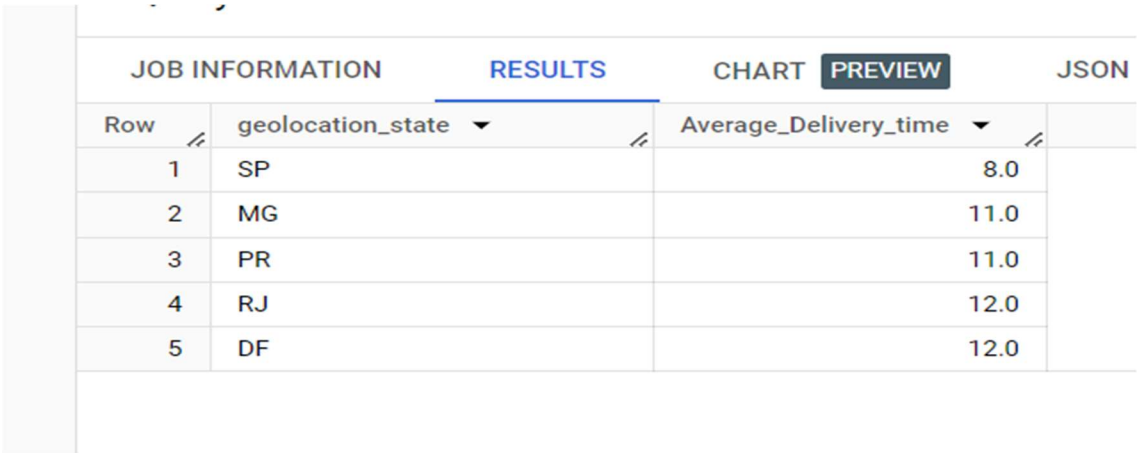
JOB INFORMATION | RESULTS | CHART | PREVIEW | J

Row	geolocation_state	Average_Delivery_time
1	AP	18.0
2	AL	17.0
3	PA	16.0
4	RO	16.0
5	MT	16.0

Top 5 states with the & lowest average delivery time.

```
WITH CTE AS
(SELECT order_id,geolocation_state,order_purchase_timestamp,(SELECT
EXTRACT(DAY FROM order_purchase_timestamp)) AS Purchase_Date,
order_delivered_customer_date,
(SELECT EXTRACT(DAY FROM order_delivered_customer_date)) AS
Delivered_Date,
CASE WHEN (SELECT EXTRACT(DAY FROM order_purchase_timestamp))>
(SELECT EXTRACT(DAY FROM order_delivered_customer_date)) AND
(SELECT EXTRACT(MONTH FROM order_purchase_timestamp))
IN(1,3,5,7,8,10,12)THEN (31-(SELECT EXTRACT(DAY FROM
```

```
order_purchase_timestamp))+(SELECT EXTRACT(DAY FROM
order_delivered_customer_date)))
WHEN (SELECT EXTRACT(DAY FROM order_purchase_timestamp))>
(SELECT EXTRACT(DAY FROM order_delivered_customer_date)) AND
(SELECT EXTRACT(MONTH FROM order_purchase_timestamp)) =2
THEN (28-(SELECT EXTRACT(DAY FROM
order_purchase_timestamp))+(SELECT EXTRACT(DAY FROM
order_delivered_customer_date)))
WHEN (SELECT EXTRACT(DAY FROM order_purchase_timestamp))>
(SELECT EXTRACT(DAY FROM order_delivered_customer_date)) AND
(SELECT EXTRACT(MONTH FROM order_purchase_timestamp))
IN(4,6,9,11)
THEN (30-(SELECT EXTRACT(DAY FROM
order_purchase_timestamp))+(SELECT EXTRACT(DAY FROM
order_delivered_customer_date)))
ELSE (SELECT EXTRACT(DAY FROM order_delivered_customer_date))-
(SELECT EXTRACT(DAY FROM order_purchase_timestamp))
END AS Delivery_Time
FROM `Target.orders` o
INNER JOIN `Target.customers` c
ON o.customer_id = c.customer_id
INNER JOIN `Target.geolocation` g
ON g.geolocation_zip_code_prefix= c.customer_zip_code_prefix
WHERE order_delivered_customer_date IS NOT NULL)
SELECT geolocation_state,ROUND(AVG(Delivery_Time),0) AS
Average_Delivery_time
FROM CTE
GROUP BY geolocation_state
ORDER BY Average_Delivery_time ASC
LIMIT 5;
```

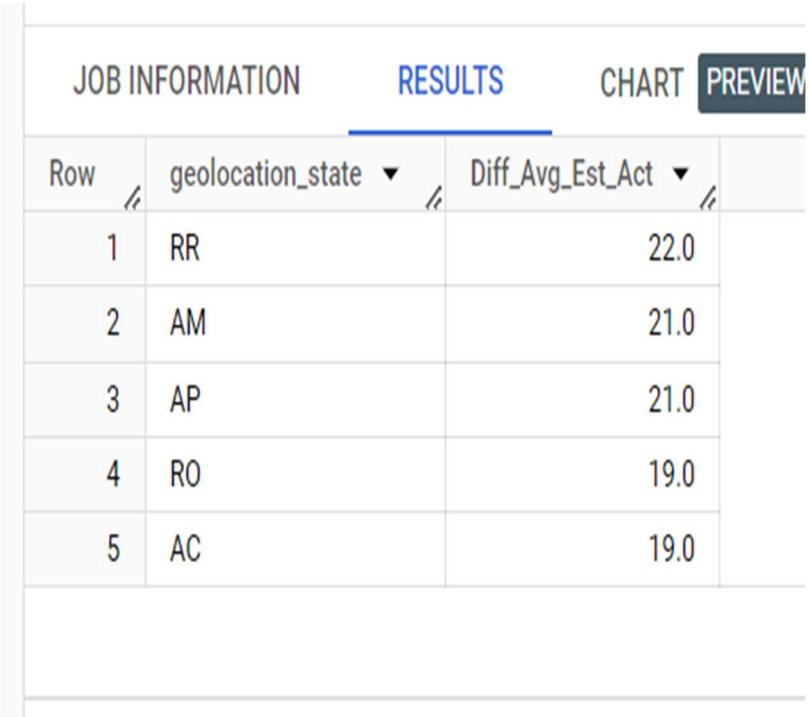


JOB INFORMATION | RESULTS | CHART | PREVIEW | JSON

Row	geolocation_state	Average_Delivery_time
1	SP	8.0
2	MG	11.0
3	PR	11.0
4	RJ	12.0
5	DF	12.0

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

```
WITH CTE AS
(SELECT
geolocation_state,order_delivered_customer_date,order_estimated
_delivery_date,
CASE WHEN order_estimated_delivery_date >
order_delivered_customer_date THEN
DATE_DIFF(order_estimated_delivery_date,
order_delivered_customer_date, day)
      ELSE 0
END AS Diff
FROM `Target.orders` o
INNER JOIN `Target.customers` c
ON o.customer_id = c.customer_id
INNER JOIN `Target.geolocation` g
ON g.geolocation_zip_code_prefix= c.customer_zip_code_prefix
WHERE order_delivered_customer_date IS NOT NULL)
SELECT geolocation_state,ROUND(AVG(Diff),0) AS Diff_Avg_Est_Act
FROM CTE
GROUP BY geolocation_state
ORDER BY Diff_Avg_Est_Act DESC
LIMIT 5;
```



JOB INFORMATION | RESULTS | CHART | PREVIEW

Row	geolocation_state	Diff_Avg_Est_Act
1	RR	22.0
2	AM	21.0
3	AP	21.0
4	RO	19.0
5	AC	19.0

6. Analysis based on the payments:

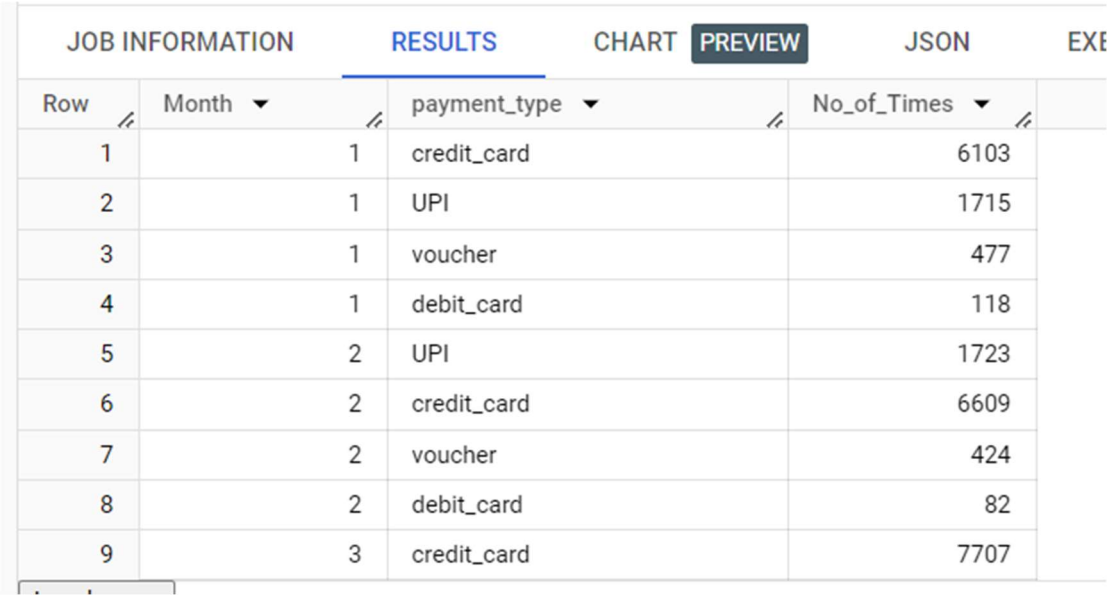
1. Find the month-on-month no. of orders placed using different payment types.

```
SELECT Year,Month,payment_type,COUNT(payment_type)AS
No_of_Times
FROM
(SELECT (SELECT EXTRACT(Year FROM order_purchase_timestamp))
AS Year,
    (SELECT EXTRACT(Month FROM order_purchase_timestamp)) AS
Month,
    payment_type
FROM `Target.orders` o
INNER JOIN `Target.payments` p
ON o.order_id = p.order_id)AS x
GROUP BY Year,Month,payment_type
ORDER BY Year,Month ASC;
```

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAILS

Row	Year	Month	payment_type	No_of_Times
1	2016	9	credit_card	3
2	2016	10	credit_card	254
3	2016	10	UPI	63
4	2016	10	voucher	23
5	2016	10	debit_card	2
6	2016	12	credit_card	1
7	2017	1	credit_card	583
8	2017	1	UPI	197
9	2017	1	voucher	61

```
SELECT Month,payment_type,COUNT(payment_type)AS
No_of_Times
FROM
(SELECT (SELECT EXTRACT(Year FROM order_purchase_timestamp))
AS Year,
    (SELECT EXTRACT(Month FROM order_purchase_timestamp)) AS
Month,
    payment_type
FROM `Target.orders` o
INNER JOIN `Target.payments` p
ON o.order_id = p.order_id)AS x
GROUP BY Month,payment_type
ORDER BY Month ASC;
```



JOB INFORMATION | RESULTS | CHART | PREVIEW | JSON | EXE

Row	Month	payment_type	No_of_Times
1	1	credit_card	6103
2	1	UPI	1715
3	1	voucher	477
4	1	debit_card	118
5	2	UPI	1723
6	2	credit_card	6609
7	2	voucher	424
8	2	debit_card	82
9	3	credit_card	7707

2. Find the no. of orders placed on the basis of the payment instalments that have been paid.

```
SELECT payment_installments,COUNT(order_id) AS
No_of_Orders_Placed
FROM `Target.payments`
GROUP BY payment_installments
ORDER BY payment_installments ASC;
```

JOB INFORMATION | RESULTS | CHART | PR

Row	payment_installments	No_of_Orders_Placed
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