BUSINESS CASE STUDY- TARGET SQL

Context:

Target is a globally renowned brand and a prominent retailer in the United States. Target makes itself a preferred shopping destination by offering outstanding value, inspiration, innovation and an exceptional guest experience that no other retailer can deliver.

This particular business case focuses on the operations of Target in Brazil and provides insightful information about 100,000 orders placed between 2016 and 2018. The dataset offers a comprehensive view of various dimensions including the order status, price, payment and freight performance, customer location, product attributes, and customer reviews.

By analyzing this extensive dataset, it becomes possible to gain valuable insights into Target's operations in Brazil. The information can shed light on various aspects of the business, such as order processing, pricing strategies, payment and shipping efficiency, customer demographics, product characteristics, and customer satisfaction levels.

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

1(A). Data type of all columns in the "customers" table.

Solution:

Query:

```
SELECT COLUMN_NAME, DATA_TYPE
FROM target_business_case.INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'customers';
```

Output:

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

Insights/Recommendations:

Customer table is mostly categorical. Customer_id can be used as unique customer identifer at order level.This table can be used to count customers and analyse data for different states etc.

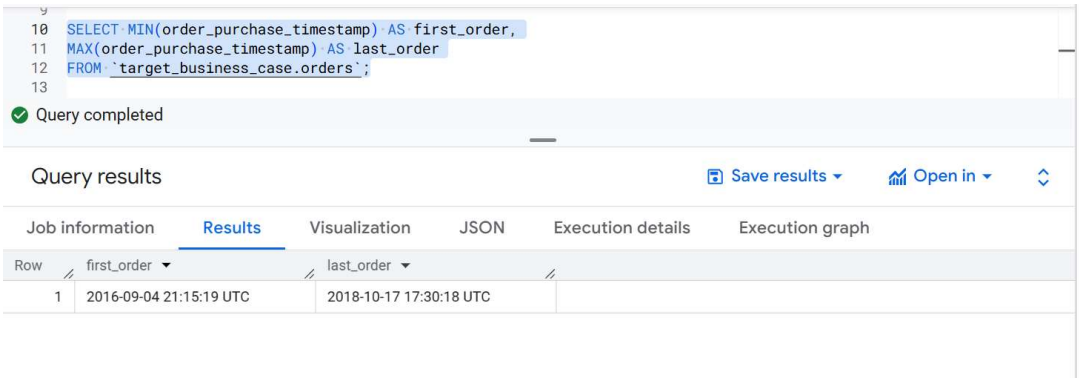
1(B). Get the time range between which the orders were placed.

Solution:

Query:

```
SELECT MIN(order_purchase_timestamp) AS first_order,
MAX(order_purchase_timestamp) AS last_order
FROM `target_business_case.orders`;
```

Output:



Insights/Recommendations:
We can see the data ranges from time span of around 2 years. So we can use this given data to analyze the growth trend from year to year or even month wise. But for better understanding its recommended to update the data to know the latest trends and analyse even better

1(C). Count the Cities & States of customers who ordered during the given period.
Solution:

Query:

```
SELECT COUNT(DISTINCT customer_city) AS order_cities,
COUNT(DISTINCT customer_state) AS order_states
FROM `target_business_case.orders` o JOIN `target_business_case.customers` c
USING(customer_id);
```

Output:

```
13
14 SELECT COUNT(DISTINCT customer_city) AS order_cities,
15 COUNT(DISTINCT customer_state) AS order_states
16 FROM `target_business_case.orders` o JOIN `target_business_case.customers` c USING(customer_id);
17
```

Query completed

Query results Save results Open in

Job information | Results | Visualization | JSON | Execution details | Execution graph

Row	order_cities	order_states
1	4119	27

Insights/Recommendation:
Its clear that the business has wide presnce givrn such large number of cities. So the business can invest in logitics for efficiency. But number of states is not much so it can be considered to spread the business to remaining states.

Question 2: In-depth Exploration:
2(A). Is there a growing trend in the no. of orders placed over the past years?
Solution:

Query:

```
WITH s AS(
SELECT EXTRACT(month from order_purchase_timestamp) AS order_month,
EXTRACT(year from order_purchase_timestamp) AS order_year,
COUNT(order_id) AS month_orders
FROM `target_business_case.orders`
GROUP BY 2,1
ORDER BY 2,1)

SELECT *, LAG(month_orders,1) OVER(ORDER BY order_year,order_month) AS
last_month_orders,
CASE WHEN
month_orders-(LAG(month_orders,1) OVER(ORDER BY order_year,order_month))>0 THEN 'YES'
ELSE 'NO' END AS growth
FROM s
ORDER BY order_year,order_month;
```

Output:

Row	order_month	order_year	month_orders	last_month_orders	growth
1	9	2016	4	null	NO
2	10	2016	324	4	YES
3	12	2016	1	324	NO
4	1	2017	800	1	YES
5	2	2017	1780	800	YES
6	3	2017	2682	1780	YES
7	4	2017	2404	2682	NO
8	5	2017	3700	2404	YES
9	6	2017	3245	3700	NO
10	7	2017	4026	3245	YES
11	8	2017	4331	4026	YES

Insights/Recommendations:
The growth column in the table shows if the months orders increased compared to previous month orders. 'YES' shows a growth while 'NO' signifies dip. This can help us understand if there is seasonality or find out the reason for dip in orders month-wise and it can help increase orders in that period in upcoming year. As mostly rows are 'YES' so we can say there has been growth with few dips in between.

2(B): Can we see some kind of monthly seasonality in terms of the no. of orders being placed?
Solution:

Query:

```
SELECT
*,
ROUND(AVG(month_orders) OVER(PARTITION BY order_year),2) AS avg_year_orders,
ROUND(month_orders/AVG(month_orders) OVER(PARTITION BY order_year),2) AS
seasonality_index
FROM(
SELECT EXTRACT(month from order_purchase_timestamp) AS order_month,
EXTRACT(year from order_purchase_timestamp) AS order_year,
COUNT(order_id) AS month_orders,
FROM `target_business_case.orders`
GROUP BY 2,1
ORDER BY 2,1)
ORDER BY order_year, seasonality_index ASC
```

Output:

Query results

Save results

Job information | Results | Visualization | JSON | Execution details | Execution graph

Row	order_month	order_year	month_orders	avg_year_orders	seasonality_index
1	12	2016	1	109.67	0.01
2	9	2016	4	109.67	0.04
3	10	2016	324	109.67	2.95
4	1	2017	800	3758.42	0.21
5	2	2017	1780	3758.42	0.47
6	4	2017	2404	3758.42	0.64
7	3	2017	2682	3758.42	0.71
8	6	2017	3245	3758.42	0.86
9	5	2017	3700	3758.42	0.98
10	7	2017	4026	3758.42	1.07

Results per page: 50 1 – 25 of 25

Insight/Recommendations:

Here seasonality index is used to check if any month shows higher orders every year compared to others. It compares the months order with the average of months orders that year. The index=1 signifies order are equal to average yearly order, if > 1 it signifies above average orders of that year. On the other hand, if its < 1 it shows below average orders means dip in orders. It helps identify business performance month wise and therefore required action can be taken. In our case given data has only one year whose all months data is given i.e. 2017 which shows orders increase by the end of the year. In 2016 only three months data is given so can't conclude seasonality from there. In 2018 we have ten months of data and we can see almost all months have near average orders hence no seasonality is seen.

2(C). During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)
-- 0-6 hrs : Dawn
-- 7-12 hrs : Mornings
-- 13-18 hrs : Afternoon
-- 19-23 hrs : Night
Solution:

Query:

```
SELECT time_day, COUNT(order_id) AS no_of_orders
FROM(
SELECT *,
CASE WHEN EXTRACT(hour from order_purchase_timestamp) BETWEEN 00 AND 06 THEN 'Dawn'
WHEN EXTRACT(hour from order_purchase_timestamp) BETWEEN 07 AND 12 THEN 'Mornings'
WHEN EXTRACT(hour from order_purchase_timestamp) BETWEEN 13 AND 18 THEN 'Afternoon'
ELSE 'Night'
END AS time_day
FROM `target_business_case.orders`)
GROUP BY 1
ORDER BY 2 DESC;
```

Output:

Query results

Job information | Results | Visualization | JSON | Execution details | Execution graph

Row	time_day	no_of_orders
1	Afternoon	38135
2	Night	28331
3	Mornings	27733
4	Dawn	5242

Insights/Recommendations:
Hence people are ordering most during afternoon time maybe during lunch breaks. Second peak is during night after finishing their work for the day. Least orders are placed during DAWN i.e. late at night and early mornings. This shows us an important insight of when customers are shopping most actually. So accordingly, the business can do digital ads or assure readiness during that time-period. Also based on the time of order we can focus on products that people are most likely to purchase at that time of the day.

Question 3. Evolution of E-commerce orders in the Brazil region:
3(A). Get the month on month no. of orders placed in each state.

Solution:

Query:

```
SELECT
c.customer_state AS state,
EXTRACT(month from o.order_purchase_timestamp) AS order_month,
EXTRACT(year from o.order_purchase_timestamp) AS order_year,
COUNT(o.order_id) AS month_orders
FROM `target_business_case.orders` o JOIN `target_business_case.customers` c
USING(customer_id)
GROUP BY 1,3,2
ORDER BY 1,3,2;
```

Output:

Query results

Save results

Job information | Results | Visualization | JSON | Execution details | Execution graph

Row	state	order_month	order_year	month_orders
1	AC	1	2017	2
2	AC	2	2017	3
3	AC	3	2017	2
4	AC	4	2017	5
5	AC	5	2017	8
6	AC	6	2017	4
7	AC	7	2017	5
8	AC	8	2017	4
9	AC	9	2017	5
10	AC	10	2017	6

Results per page: 50 1 – 50 of 565

Insights/Recommendations:
This analysis can help us check for state level growth trends and see which state are economic hubs having highest month on month orders. This can help in decision related to expansion by focusing on weaker states.

3(B). How are the customers distributed across all the states?
Solution:

Query:

```
SELECT customer_state AS state,
COUNT(DISTINCT customer_id) AS no_of_customers,
ROUND(100 * COUNT(DISTINCT customer_id) / SUM(COUNT(DISTINCT customer_id)) OVER(), 2)
AS pct_of_total
FROM `target_business_case.customers`
GROUP BY 1
ORDER BY 2 desc;
```

Output:

Query results

Save results

Job information | Results | Visualization | JSON | Execution details | Execution graph

Row	state	no_of_customers	pct_of_total
1	SP	41746	41.98
2	RJ	12852	12.92
3	MG	11635	11.7
4	RS	5466	5.5
5	PR	5045	5.07
6	SC	3637	3.66
7	BA	3380	3.4
8	DF	2140	2.15
9	ES	2033	2.04
10	GO	2020	2.03

Results per page: 50 1 – 27 of 27

Insights/Recommendations:
As observed SP state has the most concentration of customers. So, business can increase number of stores in that area. On the other hand, there are states like RR, AP, AC who have very few customers. Business should increase ads, campaigning in these areas to grow there.

Question 4. Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight and others.
4(A). Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only).
-You can use the "payment_value" column in the payments table to get the cost of orders.
Solution:

Query:

```
WITH a_sale AS(
SELECT EXTRACT(month from order_purchase_timestamp) AS order_month,
EXTRACT(year from order_purchase_timestamp) AS order_year,
SUM(p.payment_value) AS cost_of_orders
FROM `target_business_case.orders` o JOIN `target_business_case.payments` p
USING(order_id)
GROUP BY 2,1
HAVING (order_year=2017 ) AND (order_month BETWEEN 1 AND 8)
ORDER BY 2,1),

b_sale AS(
SELECT EXTRACT(month from order_purchase_timestamp) AS order_month,
EXTRACT(year from order_purchase_timestamp) AS order_year,
SUM(p.payment_value) AS cost_of_orders
FROM `target_business_case.orders` o JOIN `target_business_case.payments` p
USING(order_id)
GROUP BY 2,1
HAVING (order_year=2018 ) AND (order_month BETWEEN 1 AND 8)
ORDER BY 2,1)

SELECT a.order_month,
ROUND(a.cost_of_orders,2) AS cost_2017,
ROUND(b.cost_of_orders,2) AS cost_2018,
ROUND((b.cost_of_orders-a.cost_of_orders)*100 /a.cost_of_orders,2) AS percent_increase
FROM a_sale a
JOIN b_sale  b ON a.order_month=b.order_month
ORDER BY 1;
```

Output:

Query results

Save results

Job information | Results | Visualization | JSON | Execution details | Execution graph

Row	order_month	cost_2017	cost_2018	percent_increase
1	1	138488.04	1115004.18	705.13
2	2	291908.01	992463.34	239.99
3	3	449863.6	1159652.12	157.78
4	4	417788.03	1160785.48	177.84
5	5	592918.82	1153982.15	94.63
6	6	511276.38	1023880.5	100.26
7	7	592382.92	1066540.75	80.04
8	8	674396.32	1022425.32	51.61

Results per page: 50 1 – 8 of 8

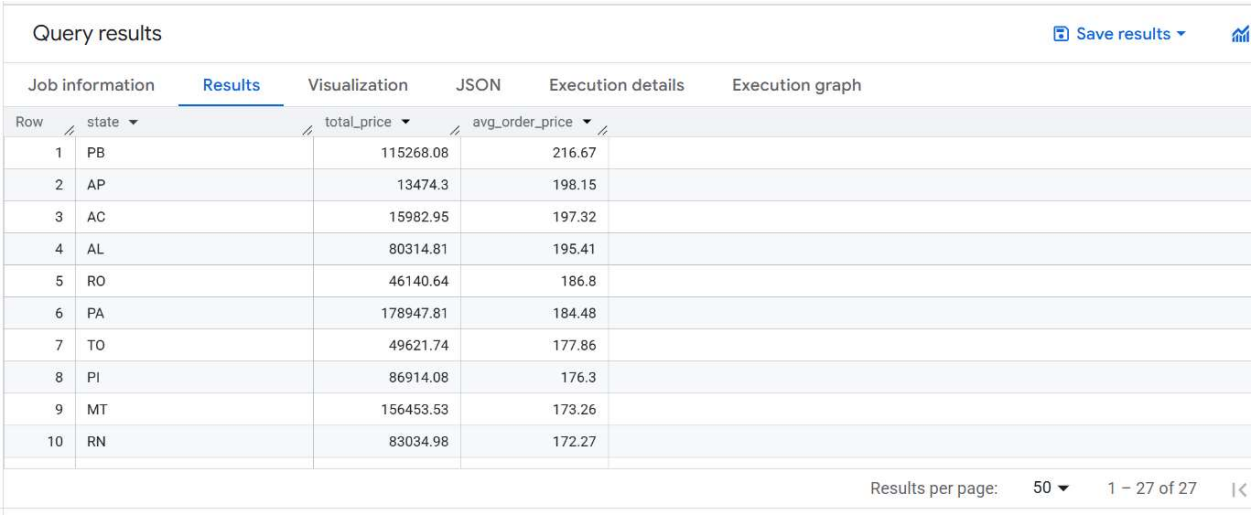
Insights/Recommendations:
Since all the months show positive percentage change from 2017 to 2018 which means consistent growth. We can see the growth is way faster in the beginning of the year which might be due to increase in number of orders or addition of new customers. Overall growth suggests business can contemplate on increasing the capital and other logistics.

4(B). Calculate the Total & Average value of order price for each state.
Solution:

Query:

```
WITH order_totals AS (
  SELECT
    o.order_id,
    c.customer_state,
    SUM(ot.price) AS order_total
  FROM `target_business_case.orders` o
  JOIN `target_business_case.order_items` ot USING(order_id)
  JOIN `target_business_case.customers` c USING(customer_id)
  GROUP BY 1,2
)
SELECT
  customer_state AS state,
  ROUND(SUM(order_total),2) AS total_price,
  ROUND(AVG(order_total),2) AS avg_order_price
FROM order_totals
GROUP BY state
ORDER BY avg_order_price DESC;
```

Output:



Row	state	total_price	avg_order_price
1	PB	115268.08	216.67
2	AP	13474.3	198.15
3	AC	15982.95	197.32
4	AL	80314.81	195.41
5	RO	46140.64	186.8
6	PA	178947.81	184.48
7	TO	49621.74	177.86
8	PI	86914.08	176.3
9	MT	156453.53	173.26
10	RN	83034.98	172.27

Insights/Recommendation:

States like SP, RJ, MG have higher total order price means they contribute most to the business. States like PB, AP, AC have higher spent per order they might be richer states. Accordingly, business need to take action to increase basket size or customer numbers.

4(C). Calculate the Total & Average value of order freight for each state.
Solution:

Query:

```
WITH freight AS (
  SELECT
    o.order_id,
    c.customer_state,
    SUM(ot.freight_value) AS order_freight
  FROM `target_business_case.orders` o
  JOIN `target_business_case.order_items` ot USING(order_id)
  JOIN `target_business_case.customers` c USING(customer_id)
  GROUP BY o.order_id, c.customer_state
)
SELECT
  customer_state AS state,
  ROUND(SUM(order_freight),2) AS total_freight_value,
  ROUND(AVG(order_freight),2) AS avg_order_freight
FROM freight
GROUP BY state
ORDER BY avg_order_freight DESC;
```

Output:

Query results

Save results

Job information | Results | Visualization | JSON | Execution details | Execution graph

Row	state	total_freight_value	avg_order_freight
1	RR	2235.19	48.59
2	PB	25719.73	48.35
3	RO	11417.38	46.22
4	AC	3686.75	45.52
5	PI	21218.2	43.04
6	MA	31523.77	42.6
7	TO	11732.68	42.05
8	AP	2788.5	41.01
9	SE	14111.47	40.9
10	PA	38699.3	39.9

Results per page: 50 1 – 27 of 27

Insights/Recommendations:
Based on the above observation the business can decide upon the states where average or total freight value is large and can optimize logistics or collaborate with regional service providers.

Question 5. Analysis based on sales, freight and delivery time.
5(A). 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.
Solution:

Query:

```
SELECT order_id,
DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp,day) AS
time_to_deliver,
DATE_DIFF(order_delivered_customer_date, order_estimated_delivery_date,day) AS
diff_estimated_delivery
FROM `target_business_case.orders`
WHERE order_delivered_customer_date IS NOT NULL
ORDER BY 3 DESC;
```

Output:

Query results

Save results

Job information | Results | Visualization | JSON | Execution details | Execution graph

Row	order_id	time_to_deliver	diff_estimated_d...
1	1b3190b2dfa9d789e1f14c05b6...	208	188
2	ca07593549f1816d26a572e06...	209	181
3	47b40429ed8cce3aee9199792...	191	175
4	2fe324febf907e3ea3f2aa96508...	189	167
5	285ab9426d6982034523a855f...	194	166
6	440d0d17af552815d15a9e41a...	195	165
7	c27815f7e3dd0b926b5855262...	187	162
8	0f4519c5f1c541ddec9f21b3bd...	194	161
9	d24e8541128cea179a11a6517...	175	161
10	2d7561026d542c8dbd8f0daea...	188	159
11	2fb597c2f772eca01b1f5c561bf...	194	155

Results per page: 50 1 – 50 of 96476

Insights/Recommendations:
In the above query we calculated time taken by an order delivery and difference between actual delivery date and estimated delivery date. If the difference is positive, it means order was late, if its negative means order was delivered early. If its zero order was delivered exactly on time. To analyze futher we can find average delivery time and proportion of orders delivered early, late or on time as follows.

```
SELECT
  ROUND(AVG(d.time_to_deliver), 2) AS avg_delivery_time,
  ROUND(SUM(CASE WHEN d.diff_estimated_delivery< 0 THEN 1 ELSE 0 END) * 100.0 /
COUNT(*), 2) AS pct_early,
  ROUND(SUM(CASE WHEN d.diff_estimated_delivery = 0 THEN 1 ELSE 0 END) * 100.0 /
COUNT(*), 2) AS pct_on_time,
  ROUND(SUM(CASE WHEN d.diff_estimated_delivery > 0 THEN 1 ELSE 0 END* 100.0 /
COUNT(*), 2) AS pct_late
FROM d
;
```

Query results

Job information | Results | Visualization | JSON | Execution details | Execution graph

Row	avg_delivery_time	pct_early	pct_on_time	pct_late
1	12.09	90.37	2.85	6.77

Therefore, average time taken by orders delivery is 12.09 days and most of the orders are delivered early than expected so its a good thing.

5(B). Find out the top 5 states with the highest & lowest average freight value.
Solution:

Query:

```
WITH f AS(
SELECT c.customer_state AS state ,
ROUND(AVG(ot.freight_value),2) AS avg_freight_value
FROM `target_business_case.orders` o JOIN `target_business_case.order_items` ot
USING(order_id) JOIN `target_business_case.customers` c USING(customer_id)
GROUP BY 1),

a AS(
SELECT *, RANK() OVER(ORDER BY avg_freight_value DESC) AS position
FROM f
ORDER BY 2 DESC
LIMIT 5),

b AS(
SELECT *, RANK() OVER(ORDER BY avg_freight_value ASC) AS position
FROM f
ORDER BY 2 ASC
LIMIT 5)

(SELECT *
FROM a) UNION ALL (SELECT *
FROM b);
```

Output:

Query results

Save results

Job information | Results | Visualization | JSON | Execution details | Execution graph

Row	state	avg_freight_value	position
1	SP	15.15	1
2	PR	20.53	2
3	MG	20.63	3
4	RJ	20.96	4
5	DF	21.04	5
6	RR	42.98	1
7	PB	42.72	2
8	RO	41.07	3
9	AC	40.07	4
10	PI	39.15	5

Results per page: 50 1 – 10 of 10

Insights/Recommendations:
Using this we can try to reduce freight cost in high freight cost states. And highlight fast and cheap delivery in low freight states.

5(C). Find out the top 5 states with the highest & lowest average delivery time.
Solution:

Query:

```
WITH f AS(
SELECT c.customer_state AS state ,
ROUND(AVG(o.delivery_time),2) AS avg_delivery_time
FROM (SELECT *, DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp,
DAY) AS delivery_time
FROM `target_business_case.orders`
WHERE order_delivered_customer_date IS NOT NULL) o JOIN
`target_business_case.customers` c USING(customer_id)
GROUP BY 1),


a AS(
SELECT *, RANK() OVER(ORDER BY avg_delivery_time DESC) AS position
FROM f
ORDER BY 2 DESC
LIMIT 5),

b AS(
SELECT *, RANK() OVER(ORDER BY avg_delivery_time ASC) AS position
FROM f
ORDER BY 2 ASC
LIMIT 5)

(SELECT *
FROM a) UNION ALL (SELECT *
FROM b);
```

OUTPUT:

Row	state	avg_delivery_time	position
1	SP	8.3	1
2	PR	11.53	2
3	MG	11.54	3
4	DF	12.51	4
5	SC	14.48	5
6	RR	28.98	1
7	AP	26.73	2
8	AM	25.99	3
9	AL	24.04	4
10	PA	23.32	5

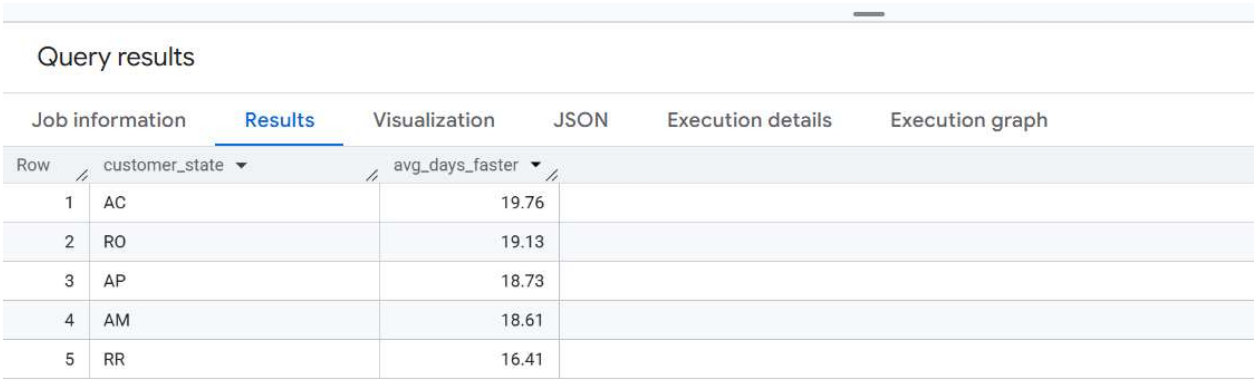
Insights/Recommendations:
High average delivery time states are RR, AP, AM etc. So the business needs to improve logistics in these areas. For fast states like SP, PR, MG etc. There can be improved estimated delivery times accordingly.

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

Query:

```
SELECT
  c.customer_state,
  ROUND(AVG(DATE_DIFF(order_estimated_delivery_date, order_delivered_customer_date,
DAY)), 2) AS avg_days_faster
FROM `target_business_case.orders` o JOIN `target_business_case.customers` c
USING(customer_id)
WHERE order_delivered_customer_date IS NOT NULL   -- exclude undelivered orders
GROUP BY 1
ORDER BY avg_days_faster DESC
LIMIT 5;
```

Output:



Row	customer_state	avg_days_faster
1	AC	19.76
2	RO	19.13
3	AP	18.73
4	AM	18.61
5	RR	16.41

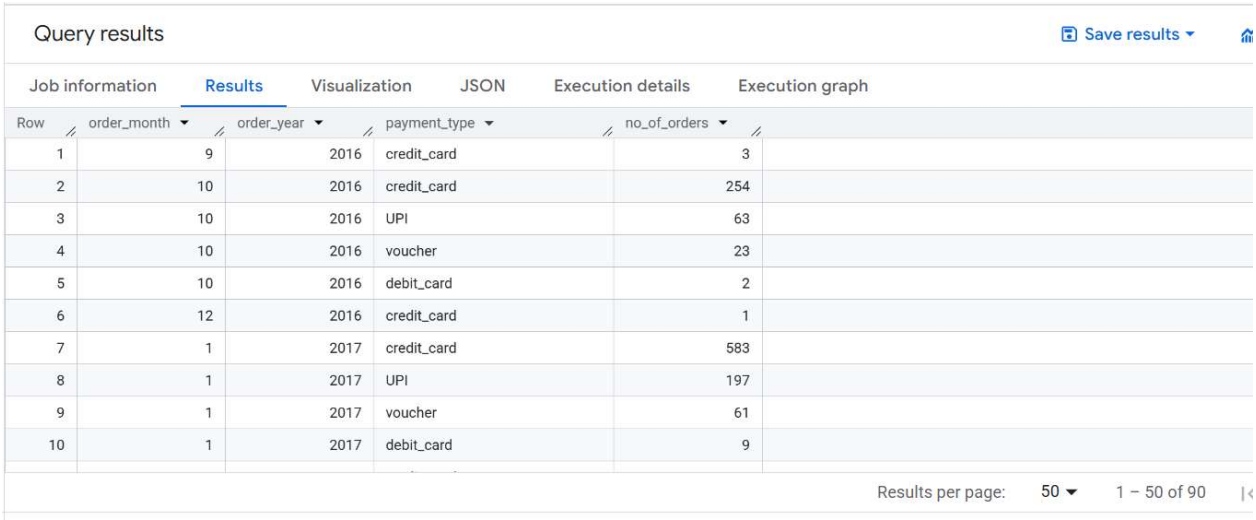
Insights/recommendations:
We have identified states delivering orders really fast than expected. The column avg_days_faster represents the number of days by which the order arrived early. This helps in identifying well performing states and logistics optimization.

Question 6. Analysis based on the payments:
6(A). Find the month on month no. of orders placed using different payment types.
Solution:

Query:

```
SELECT EXTRACT(month from order_purchase_timestamp) AS order_month,
EXTRACT(year from order_purchase_timestamp) AS order_year,
p.payment_type,
COUNT(o.order_id) AS no_of_orders
FROM `target_business_case.orders` o JOIN `target_business_case.payments` p
USING(order_id)
GROUP BY 2,1,3
ORDER BY 2,1,4 DESC;
```

Output:



Insights/Recommendations:
In almost all the months maximum number of orders are paid using credit card. To promote other high margin payment types business can offer discounts/coupons. On 2nd UPI is trending after credit card. So accordingly finance team can plan payment processing.

6(B). Find the no. of orders placed on the basis of the payment installments that have been paid.

Solution:

Query:

```
SELECT
payment_installments,
COUNT(order_id) no_of_orders,
Round(100*COUNT(order_id)/SUM(COUNT(order_id)) OVER(), 2) As pct_orders
FROM `target_business_case.payments` p JOIN `target_business_case.orders`o
USING(order_id)
```

```
WHERE payment_installments>0
GROUP BY 1
ORDER BY 3 DESC;
```

Output:

Query results

Save results

Job information | Results | Visualization | JSON | Execution details | Execution graph

Row	payment_installm...	no_of_orders	pct_orders
1	1	52546	50.58
2	2	12413	11.95
3	3	10461	10.07
4	4	7098	6.83
5	10	5328	5.13
6	5	5239	5.04
7	8	4268	4.11
8	6	3920	3.77
9	7	1626	1.57
10	9	644	0.62

Results per page: 50 1 – 23 of 23

Insights/Recommendations:
As we can see most of the orders almost 50% preferred single installment payment. To ensure cash flow for longer period, business can offer installment promotions for large purchases.