Business Case: Target SQL

Deadline for the Business Case:

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 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.

____

Dataset: https://drive.google.com/drive/folders/1TGEc66YKbD443nslRi1bWgVd238gJCnb

The data is available in 8 csv files:

1. customers.csv
2. sellers.csv
3. order_items.csv
4. geolocation.csv
5. payments.csv
6. reviews.csv
7. orders.csv
8. products.csv

____

Problem Statement:

Assuming you are a data analyst/ scientist at Target, you have been assigned the task of analyzing the given data set to extract valuable insights and provide actionable recommendations.

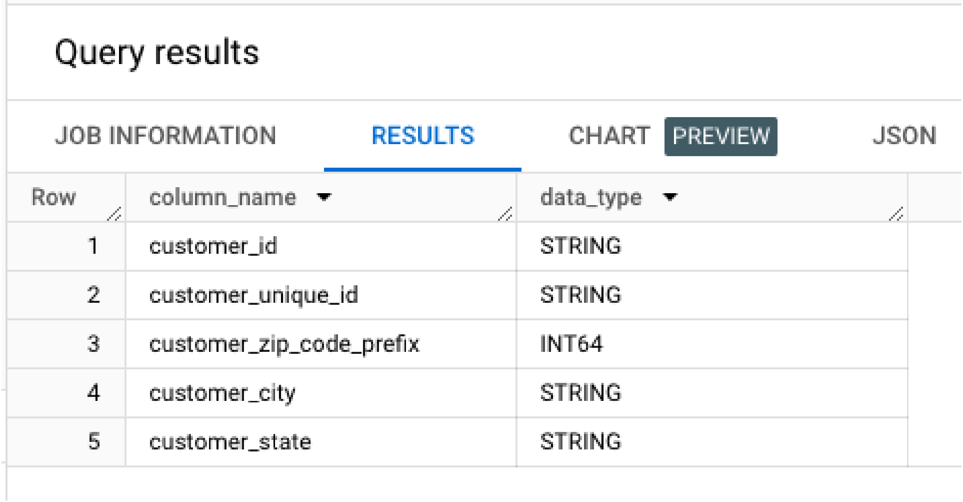
What does 'good' look like? - **GOOGLE BIG QUERY IS USED TO WRITE QUERY**

INCASE THE SCREENSHOTS ARE NOT VISIBLE IN PDF THEN YOU CAN REQUEST THE ACCESS TO WORD FILE HERE
https://docs.google.com/document/d/1uRsbm8T3kHzKpDMuTEtRkZvScgGvg9X5/edit

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

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

```
SELECT column_name, data_type

FROM
`scalaer-dsml-sql-405918.business_case_target.INFORMATION_SCHEMA.COLUMNS`

WHERE table_name = 'customers'
```



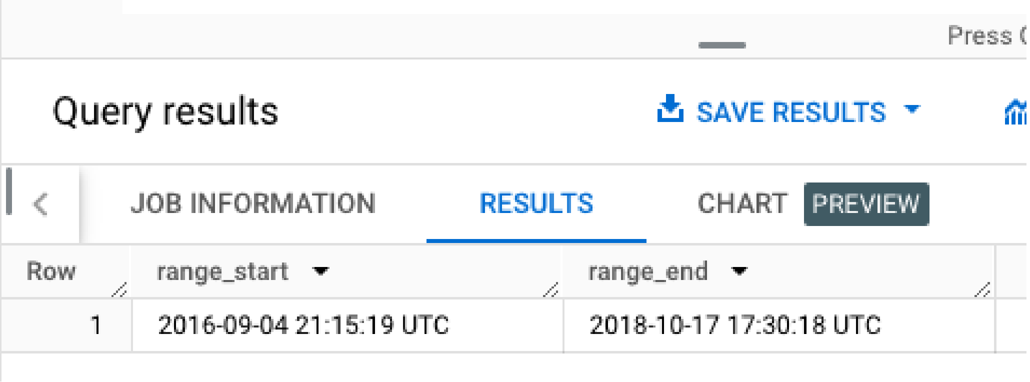
Row	column_name	data_type
1	customer_id	STRING
2	customer_unique_id	STRING
3	customer_zip_code_prefix	INT64
4	customer_city	STRING
5	customer_state	STRING

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

```
SELECT

MIN(order_purchase_timestamp ) as range_start,

MAX(order_purchase_timestamp) AS range_end

FROM `scalaer-dsml-sql-405918.business_case_target.orders`
```

Insights : All the orders are placed between 04th Sep 2016 and 17th Oct 2018

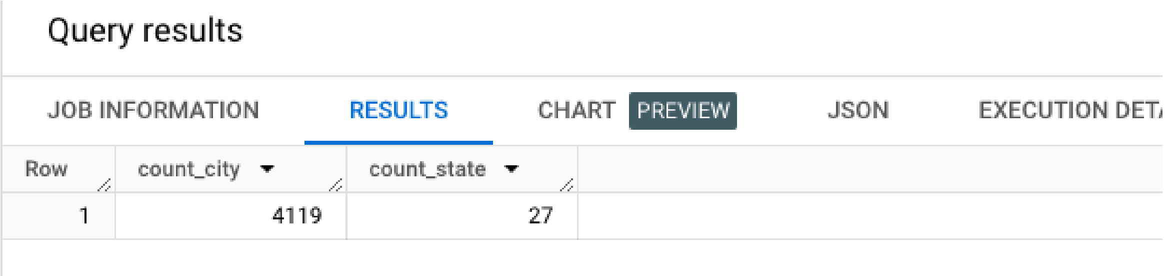
Action :NA



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

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

```
SELECT

COUNT (DISTINCT c.customer_city) as count_city,

COUNT (DISTINCT c.customer_state) AS count_state

FROM `scalaer-dsml-sql-405918.business_case_target.orders` AS o

JOIN `scalaer-dsml-sql-405918.business_case_target.customers` AS c

ON o.customer_id = c.customer_id
```

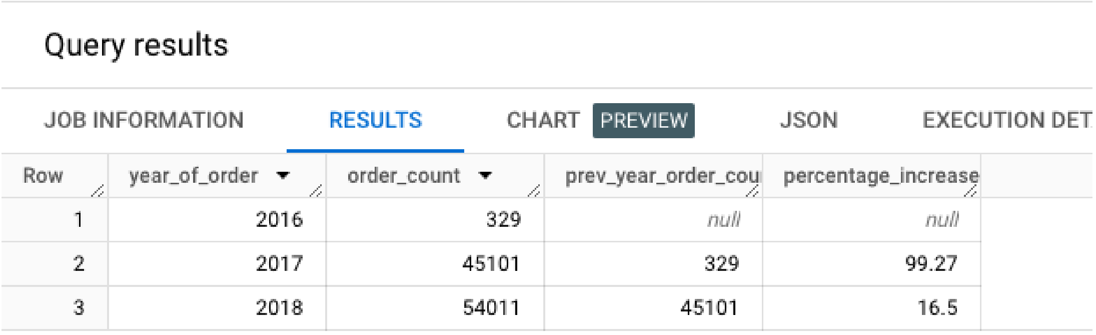


Row	count_city	count_state
1	4119	27

2. In-depth Exploration:

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

```
SELECT year_of_order,

          order_count,

          LAG(order_count) OVER(ORDER BY year_of_order ASC) AS
          prev_year_order_count,

          ROUND((order_count - LAG(order_count) OVER(ORDER BY year_of_order
          ASC))*100/ (order_count),2) AS percentage_increase

          FROM

          (SELECT DISTINCT (EXTRACT (YEAR FROM order_purchase_timestamp))AS
          year_of_order, COUNT(order_id) as order_count

          FROM `scalaer-dsml-sql-405918.business_case_target.orders`

          GROUP BY (EXTRACT (YEAR FROM order_purchase_timestamp))

          )

          ORDER BY year_of_order ASC
```



Query results

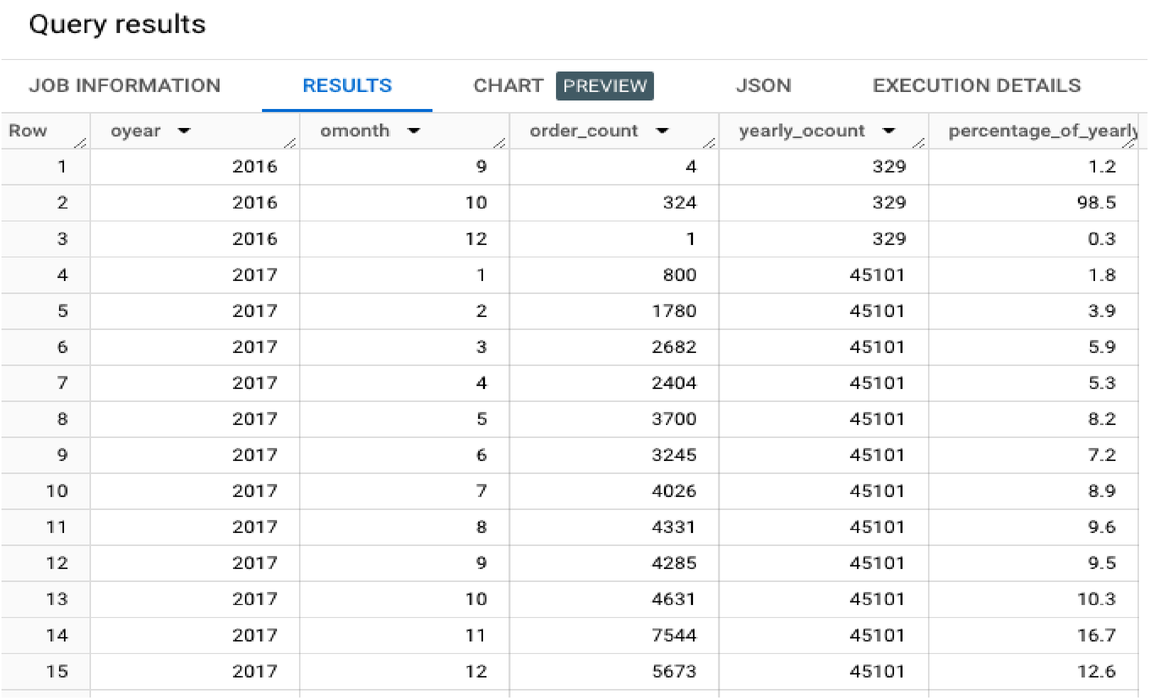
Row	year_of_order	order_count	prev_year_order_cou	percentage_increase
1	2016	329	*null*	*null*
2	2017	45101	329	99.27
3	2018	54011	45101	16.5

Insights : orders were highest in 2017 compared to 2016 and 2018 but there was not full year of sale sate was available so can't really compare 2017 sale with 2016 and 2018

Action : NA

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

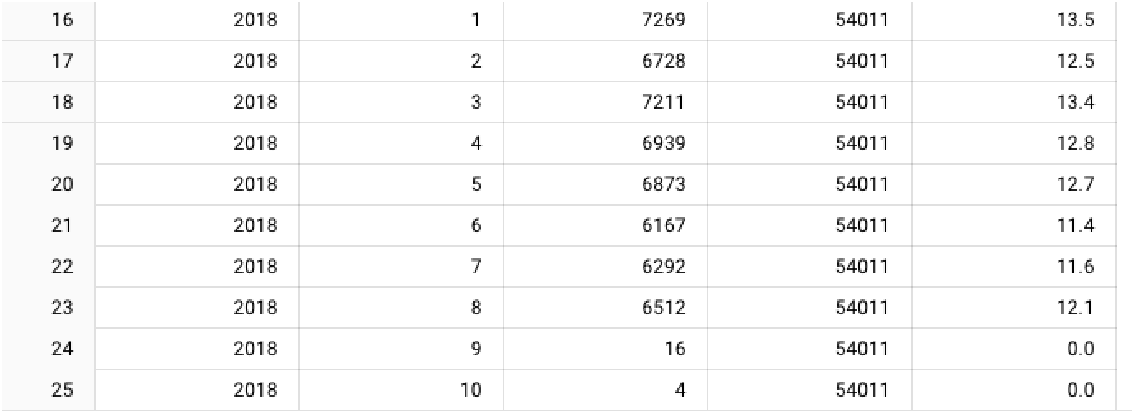
```
SELECT oyear, omonth, order_count,

SUM(order_count) OVER (PARTITION BY oyear) AS yearly_ocount,

ROUND (order_count*100/ (SUM(order_count) OVER (PARTITION BY oyear)),1)
AS percentage_of_yearly_orders

FROM

(

SELECT EXTRACT (YEAR FROM order_purchase_timestamp)AS oyear,

EXTRACT (MONTH FROM order_purchase_timestamp)AS omonth,

COUNT(order_id) as order_count

FROM `scalaer-dsml-sql-405918.business_case_target.orders`

GROUP BY oyear,omonth

ORDER BY oyear ASC, omonth ASC

)
```



Row	oyear	omonth	order_count	yearly_ocount	percentage_of_yearly
1	2016	9	4	329	1.2
2	2016	10	324	329	98.5
3	2016	12	1	329	0.3
4	2017	1	800	45101	1.8
5	2017	2	1780	45101	3.9
6	2017	3	2682	45101	5.9
7	2017	4	2404	45101	5.3
8	2017	5	3700	45101	8.2
9	2017	6	3245	45101	7.2
10	2017	7	4026	45101	8.9
11	2017	8	4331	45101	9.6
12	2017	9	4285	45101	9.5
13	2017	10	4631	45101	10.3
14	2017	11	7544	45101	16.7
15	2017	12	5673	45101	12.6

Insights : During last 3 months of the year order count in high in 2017 may be due to festive season

Action : inventory and supply chain to be planned accordingly keeping upcoming season in mind.



16	2018	1	7269	54011	13.5
17	2018	2	6728	54011	12.5
18	2018	3	7211	54011	13.4
19	2018	4	6939	54011	12.8
20	2018	5	6873	54011	12.7
21	2018	6	6167	54011	11.4
22	2018	7	6292	54011	11.6
23	2018	8	6512	54011	12.1
24	2018	9	16	54011	0.0
25	2018	10	4	54011	0.0

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

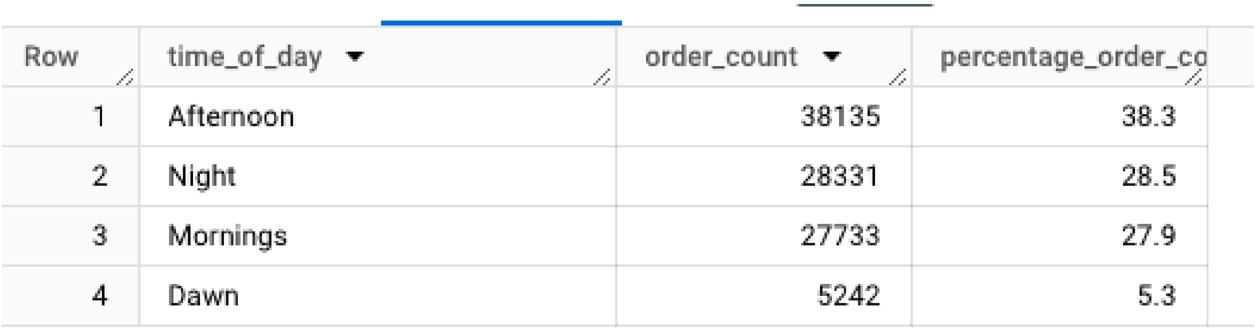
   i. 0-6 hrs : Dawn

   ii. 7-12 hrs : Mornings

   iii. 13-18 hrs : Afternoon

   iv. 19-23 hrs : Night

```
SELECT time_of_day,

order_count,

ROUND ((order_count*100/ SUM(order_count) OVER()),1) AS percentage_order_count

FROM

(

SELECT

CASE

WHEN EXTRACT (HOUR FROM order_purchase_timestamp) BETWEEN 0 AND 6 THEN 'Dawn'

WHEN EXTRACT (HOUR FROM order_purchase_timestamp) BETWEEN 7 AND 12 THEN
'Mornings'

WHEN EXTRACT (HOUR FROM order_purchase_timestamp) BETWEEN 13 AND 18 THEN
'Afternoon'

WHEN EXTRACT (HOUR FROM order_purchase_timestamp) BETWEEN 19 AND 24 THEN 'Night'

ELSE 'Other'

END AS time_of_day,

COUNT(*) as order_count

FROM `scalaer-dsml-sql-405918.business_case_target.orders`

GROUP BY time_of_day
ORDER BY time_of_day

)

ORDER BY percentage_order_count DESC
```



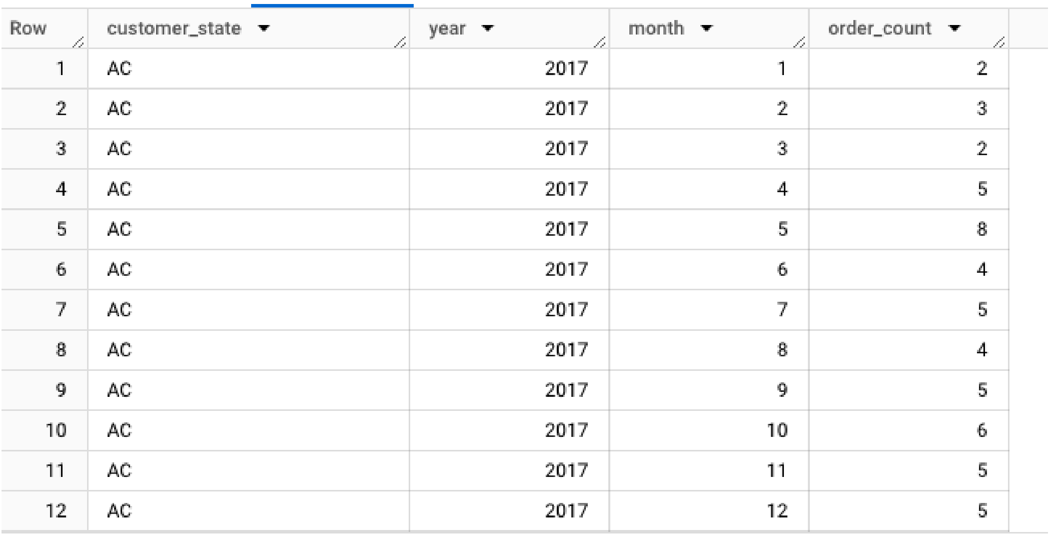
Row	time_of_day	order_count	percentage_order_co
1	Afternoon	38135	38.3
2	Night	28331	28.5
3	Mornings	27733	27.9
4	Dawn	5242	5.3

Insights : Orders placed in Afternoon is highest followed by Night and Mornings

Action : Accordingly inventory and staff in stores to be planned

3. **Evolution of E-commerce orders in the Brazil region:**
    a. Get the month-on-month no. of orders placed in each state.

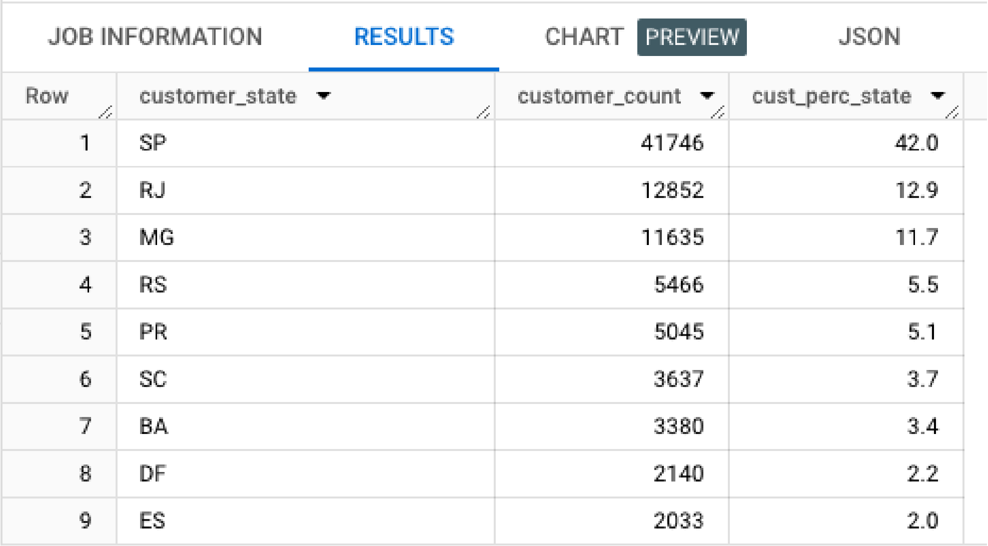
```
SELECT

          c.customer_state,

          EXTRACT (YEAR FROM o.order_purchase_timestamp) as year,

          EXTRACT (MONTH FROM o.order_purchase_timestamp) as month,

          COUNT(*) as order_count

          FROM `scalaer-dsml-sql-405918.business_case_target.orders` AS o

          JOIN `scalaer-dsml-sql-405918.business_case_target.customers` AS c

          ON o.customer_id = c.customer_id

          GROUP BY c.customer_state, year ,month

          ORDER BY c.customer_state ASC, year ASC, month ASC
```



Row	customer_state	year	month	order_count
1	AC	2017	1	2
2	AC	2017	2	3
3	AC	2017	3	2
4	AC	2017	4	5
5	AC	2017	5	8
6	AC	2017	6	4
7	AC	2017	7	5
8	AC	2017	8	4
9	AC	2017	9	5
10	AC	2017	10	6
11	AC	2017	11	5
12	AC	2017	12	5

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

```
SELECT customer_state,customer_count,

ROUND ((customer_count*100/ SUM(customer_count) OVER()),1) AS
cust_perc_state

FROM

(

SELECT c.customer_state, COUNT(DISTINCT o.customer_id) as customer_count

FROM `scalaer-dsml-sql-405918.business_case_target.orders` AS o

JOIN `scalaer-dsml-sql-405918.business_case_target.customers` AS c

ON o.customer_id = c.customer_id

GROUP BY c.customer_state

ORDER BY customer_count DESC

)

ORDER BY cust_perc_state DESC
```



Row	customer_state	customer_count	cust_perc_state
1	SP	41746	42.0
2	RJ	12852	12.9
3	MG	11635	11.7
4	RS	5466	5.5
5	PR	5045	5.1
6	SC	3637	3.7
7	BA	3380	3.4
8	DF	2140	2.2
9	ES	2033	2.0

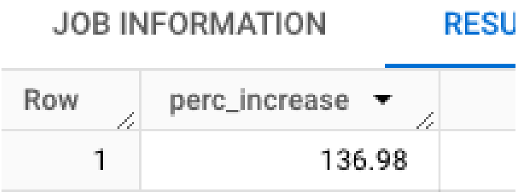
Insights :highest customers are from SP and RJ and so on

Action : according to the customer base new expansion can be planned and with states with low customer base some extra discount to be given to increase the footfall in stores.

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

   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.

```
SELECT perc_increase

FROM

(

SELECT year,SUM(order_amount) as total_order_amount,

LAG(SUM(order_amount)) OVER(ORDER BY year ASC) AS total_order_amount_lag,

ROUND(((SUM(order_amount) - LAG(sum(order_amount)) OVER(ORDER BY year
ASC) )*100 / LAG(sum(order_amount)) OVER(ORDER BY year ASC)),2) AS
Perc_increase

FROM

(

SELECT EXTRACT(YEAR from o.order_purchase_timestamp) AS year,

EXTRACT(MONTH from o.order_purchase_timestamp) AS month,
SUM(p.payment_value) AS order_amount
FROM `scalaer-dsml-sql-405918.business_case_target.orders` AS o
JOIN `scalaer-dsml-sql-405918.business_case_target.payments` AS p
ON o.order_id = p.order_id
WHERE
EXTRACT(YEAR from o.order_purchase_timestamp) IN (2017,2018) AND
EXTRACT(MONTH from o.order_purchase_timestamp) IN (1,2,3,4,5,6,7,8)
GROUP BY year, month
ORDER BY year ASC, month ASC
)
GROUP BY year
ORDER BY year ASC
)
WHERE perc_increase IS NOT NULL
```

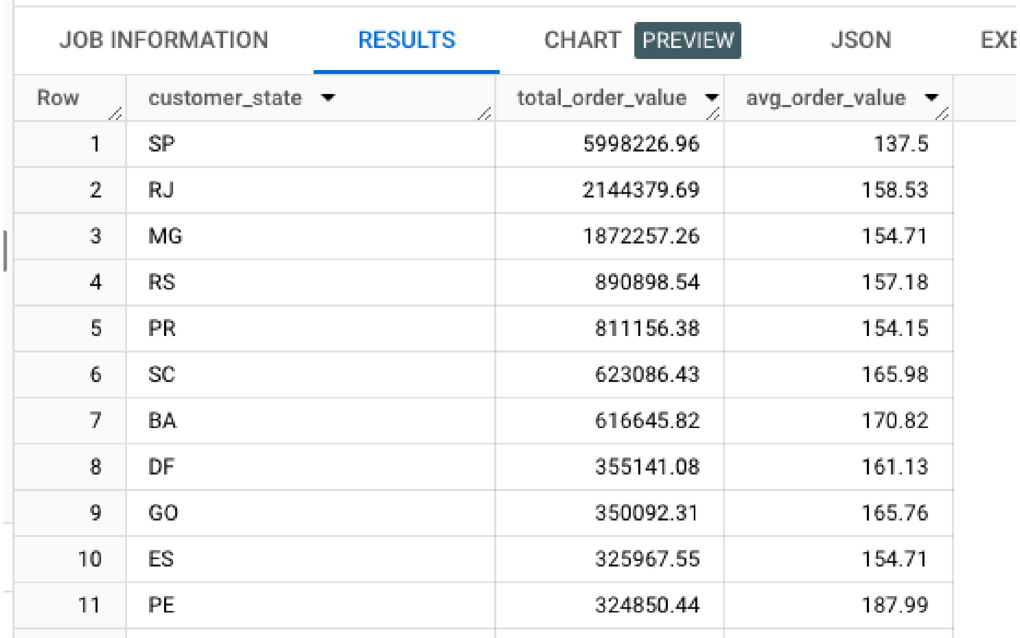


Row	perc_increase
1	136.98

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

```
SELECT c.customer_state,
ROUND (SUM(p.payment_value),2) AS total_order_value,
ROUND(AVG(p.payment_value),2) AS avg_order_value
FROM `scalaer-dsml-sql-405918.business_case_target.orders` AS o
JOIN `scalaer-dsml-sql-405918.business_case_target.customers` AS c
ON o.customer_id = c.customer_id
JOIN `scalaer-dsml-sql-405918.business_case_target.payments` AS p
ON o.order_id = p.order_id
GROUP BY c.customer_state
```

```
ORDER BY total_order_value DESC
```



Row	customer_state	total_order_value	avg_order_value
1	SP	5998226.96	137.5
2	RJ	2144379.69	158.53
3	MG	1872257.26	154.71
4	RS	890898.54	157.18
5	PR	811156.38	154.15
6	SC	623086.43	165.98
7	BA	616645.82	170.82
8	DF	355141.08	161.13
9	GO	350092.31	165.76
10	ES	325967.55	154.71
11	PE	324850.44	187.99

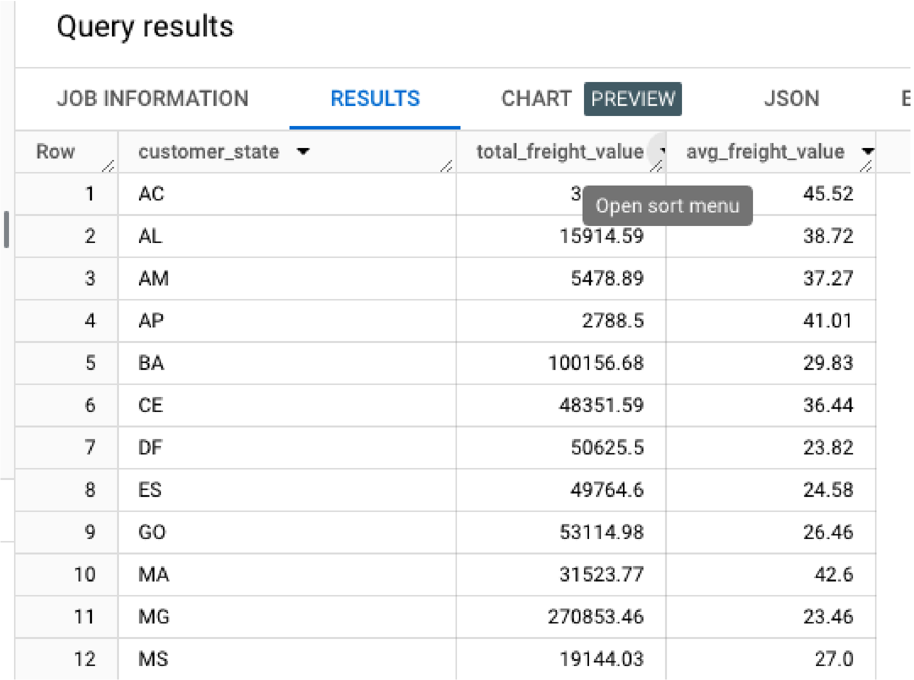
Insights : Avg order value is low in SP states with highest customers

Action : Cross selling to be done increase the ag order value to improve profitability

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

```
SELECT customer_state,

ROUND(SUM(order_freight_value),2) AS total_freight_value,

ROUND(AVG(order_freight_value),2) AS avg_freight_value

FROM

(

      SELECT distinct o.order_id,c.customer_state, SUM(oi.freight_value)
      AS order_freight_value

      FROM `scalaer-dsml-sql-405918.business_case_target.orders` AS o

      JOIN `scalaer-dsml-sql-405918.business_case_target.customers` AS c

      ON o.customer_id = c.customer_id

      JOIN `scalaer-dsml-sql-405918.business_case_target.order_items` AS
      oi

      ON o.order_id = oi.order_id

      GROUP BY o.order_id, c.customer_state
```

```
    ORDER BY o.order_id
    )
GROUP BY customer_state
ORDER BY customer_state ASC
```



Row	customer_state	total_freight_value	avg_freight_value
1	AC	3	45.52
2	AL	15914.59	38.72
3	AM	5478.89	37.27
4	AP	2788.5	41.01
5	BA	100156.68	29.83
6	CE	48351.59	36.44
7	DF	50625.5	23.82
8	ES	49764.6	24.58
9	GO	53114.98	26.46
10	MA	31523.77	42.6
11	MG	270853.46	23.46
12	MS	19144.03	27.0

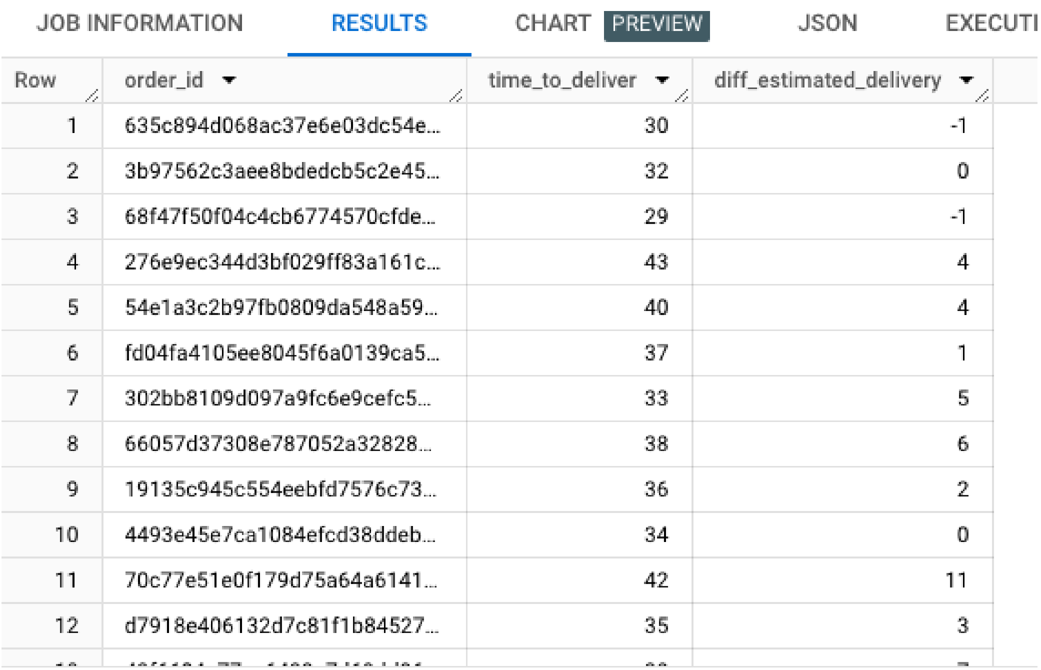
Insights : Avg freight cost decreases with high order count

Action : try to increase the total order value so to benefit both customer and Target

5. **Analysis based on sales, freight and delivery time.**
   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. Do this in a single query.

      You can calculate the delivery time and the difference between the estimated & actual delivery date using the given formula:
      - **time_to_deliver** = order_delivered_customer_date - order_purchase_timestamp
      - **diff_estimated_delivery** = order_delivered_customer_date - order_estimated_delivery_date

```
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 `scalaer-dsml-sql-405918.business_case_target.orders`

WHERE order_status = 'delivered'
```



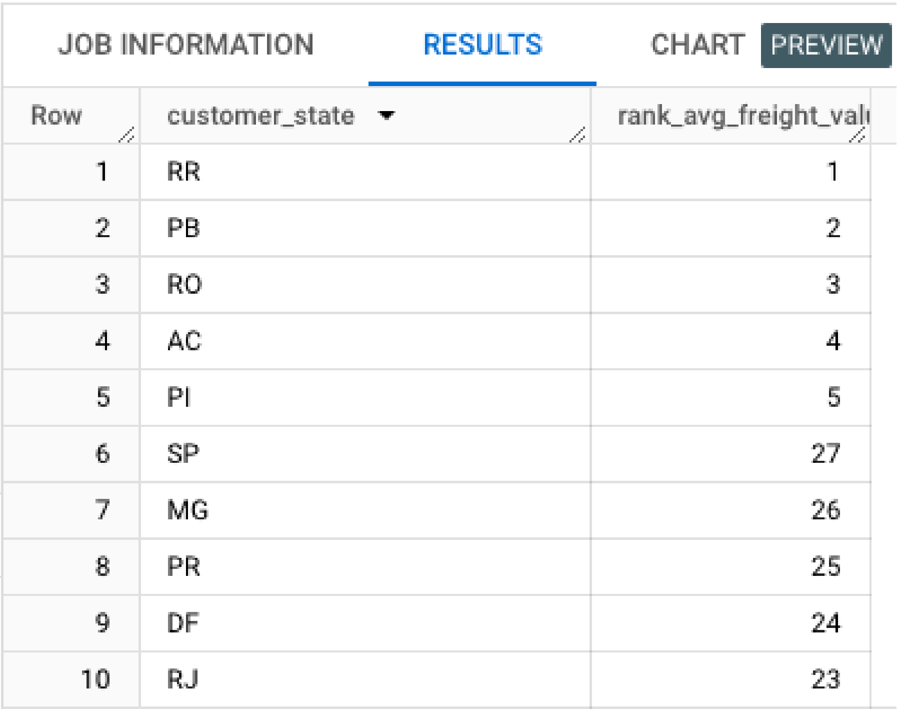
Row	order_id	time_to_deliver	diff_estimated_delivery
1	635c894d068ac37e6e03dc54e...	30	-1
2	3b97562c3aee8bdedcb5c2e45...	32	0
3	68f47f50f04c4cb6774570cfde...	29	-1
4	276e9ec344d3bf029ff83a161c...	43	4
5	54e1a3c2b97fb0809da548a59...	40	4
6	fd04fa4105ee8045f6a0139ca5...	37	1
7	302bb8109d097a9fc6e9cefc5...	33	5
8	66057d37308e787052a32828...	38	6
9	19135c945c554eebfd7576c73...	36	2
10	4493e45e7ca1084efcd38ddeb...	34	0
11	70c77e51e0f179d75a64a6141...	42	11
12	d7918e406132d7c81f1b84527...	35	3

Insights : value in minus for diff_estimated_delivery is early delivery cases

Action : feedback of customer incase of early delivery and can be highlighted in the reviews section

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

```
WITH CTE AS

(

SELECT customer_state,

ROUND(AVG(order_freight_value),2) AS avg_freight_value,

DENSE_RANK() OVER (ORDER BY ROUND(AVG(order_freight_value),2) DESC) AS
rank_avg_freight_value

FROM

(

SELECT distinct o.order_id,c.customer_state, SUM(oi.freight_value) AS
order_freight_value
FROM `scalaer-dsml-sql-405918.business_case_target.orders` AS o
JOIN `scalaer-dsml-sql-405918.business_case_target.customers` AS c
ON o.customer_id = c.customer_id
JOIN `scalaer-dsml-sql-405918.business_case_target.order_items` AS oi
ON o.order_id = oi.order_id
GROUP BY o.order_id, c.customer_state
ORDER BY o.order_id
)
GROUP BY customer_state
ORDER BY rank_avg_freight_value ASC
)

(
SELECT *
FROM CTE
ORDER BY rank_avg_freight_value DESC
LIMIT 5 )

UNION ALL
(
SELECT *
FROM CTE
ORDER BY rank_avg_freight_value ASC
LIMIT 5
)
```



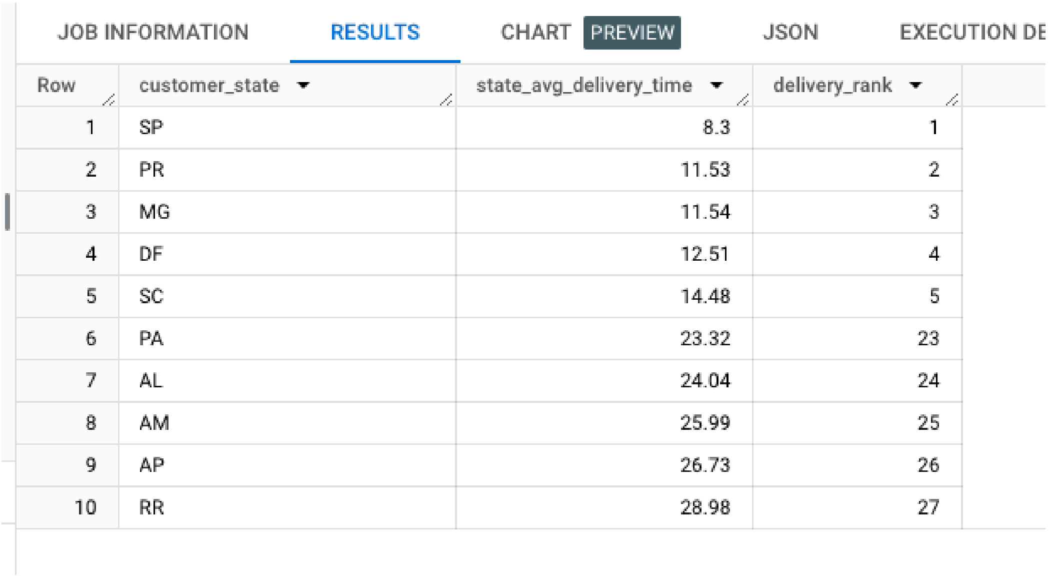
Row	customer_state	rank_avg_freight_val
1	RR	1
2	PB	2
3	RO	3
4	AC	4
5	PI	5
6	SP	27
7	MG	26
8	PR	25
9	DF	24
10	RJ	23

Insights : states with high fright value are higher in rank

Action : action to be taken to reduce freight cost somehow

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

```
WITH CTE AS

(

SELECT customer_state,

ROUND(AVG(time_to_deliver),2) AS state_avg_delivery_time,

DENSE_RANK() OVER (ORDER BY ROUND(AVG(time_to_deliver),2) ASC) AS
delivery_rank #less rank for early delivery

FROM

(

SELECT o.order_id, c.customer_state,

DATE_DIFF (o.order_delivered_customer_date, o.order_purchase_timestamp,
DAY) AS time_to_deliver

FROM `scalaer-dsml-sql-405918.business_case_target.orders` AS o
JOIN `scalaer-dsml-sql-405918.business_case_target.customers` AS c
ON o.customer_id = c.customer_id
WHERE o.order_status = 'delivered'
ORDER BY time_to_deliver
)
GROUP BY customer_state
ORDER BY delivery_rank ASC
)

(
(
SELECT *
FROM CTE
ORDER BY delivery_rank ASC
LIMIT 5
)

UNION ALL

(
SELECT *
FROM CTE
ORDER BY delivery_rank DESC
LIMIT 5
)
ORDER BY delivery_rank ASC
)
```



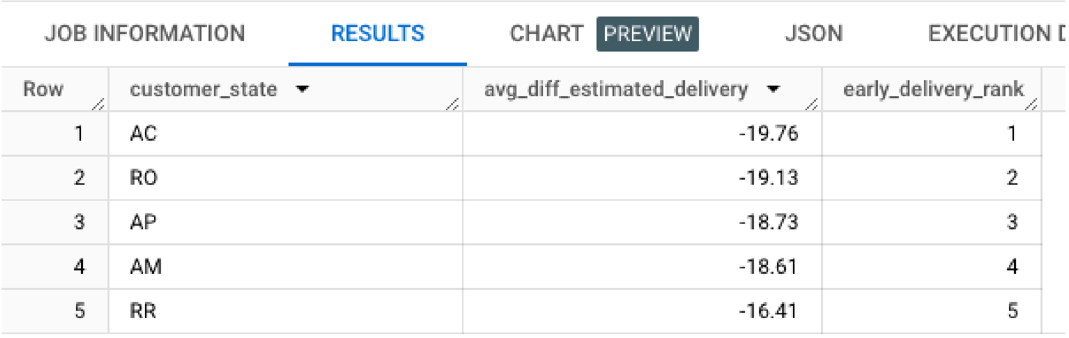
Row	customer_state	state_avg_delivery_time	delivery_rank
1	SP	8.3	1
2	PR	11.53	2
3	MG	11.54	3
4	DF	12.51	4
5	SC	14.48	5
6	PA	23.32	23
7	AL	24.04	24
8	AM	25.99	25
9	AP	26.73	26
10	RR	28.98	27

Insights : states with lowest and highest delivery time

Action : action to be taken to reduce delivery time to increase customer satisfaction

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.

```
WITH CTE AS
(
SELECT customer_state,
ROUND(AVG(diff_estimated_delivery ),2) AS avg_diff_estimated_delivery,
DENSE_RANK() OVER (ORDER BY ROUND(AVG(diff_estimated_delivery ),2)) AS early_delivery_rank #less rank for early delivery
FROM
(
SELECT o.order_id, c.customer_state,
DATE_DIFF (order_delivered_customer_date, order_estimated_delivery_date, DAY) AS diff_estimated_delivery
FROM `scalaer-dsml-sql-405918.business_case_target.orders` AS o
JOIN `scalaer-dsml-sql-405918.business_case_target.customers` AS c
ON o.customer_id = c.customer_id
WHERE

o.order_status = 'delivered'

AND

DATE_DIFF (order_delivered_customer_date, order_estimated_delivery_date,
DAY) IS NOT NULL

ORDER BY diff_estimated_delivery

)

GROUP BY customer_state

ORDER BY early_delivery_rank ASC

)

SELECT *

FROM CTE

ORDER BY early_delivery_rank ASC

LIMIT 5
```



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

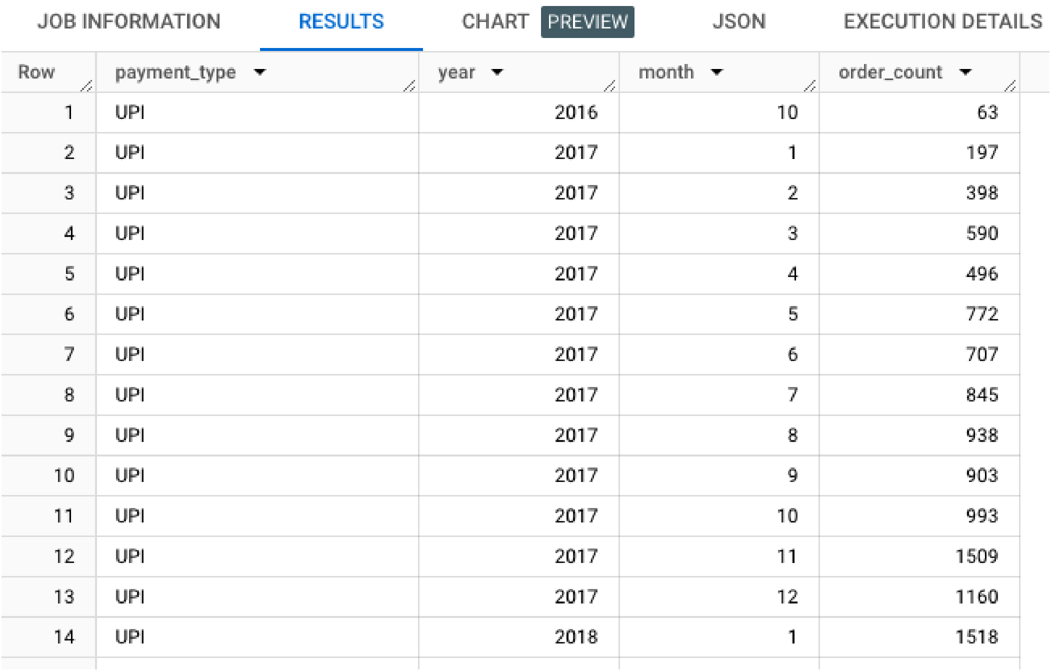
Insights : states with lowest and highest delivery time

Action : action to be taken to reduce delivery time to increase customer satisfaction

9. Analysis based on the payments:

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

```
SELECT payment_type, year, month, COUNT(DISTINCT order_id) as order_count

FROM

(

SELECT DISTINCT o.order_id, p.payment_type,

EXTRACT (YEAR from o.order_purchase_timestamp) AS year,

EXTRACT (MONTH from o.order_purchase_timestamp) AS month

FROM `scalaer-dsml-sql-405918.business_case_target.orders` AS o

JOIN `scalaer-dsml-sql-405918.business_case_target.payments` AS p

ON o.order_id = p.order_id

)

GROUP BY payment_type, year, month

ORDER BY payment_type ASC, year ASC, month ASC
```



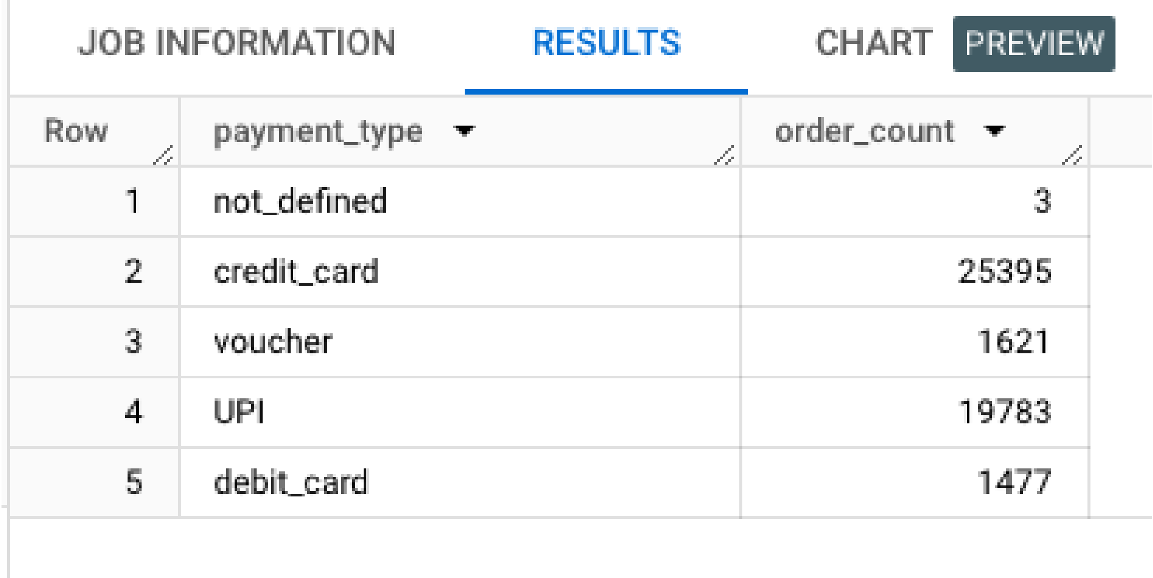
Row	payment_type	year	month	order_count
1	UPI	2016	10	63
2	UPI	2017	1	197
3	UPI	2017	2	398
4	UPI	2017	3	590
5	UPI	2017	4	496
6	UPI	2017	5	772
7	UPI	2017	6	707
8	UPI	2017	7	845
9	UPI	2017	8	938
10	UPI	2017	9	903
11	UPI	2017	10	993
12	UPI	2017	11	1509
13	UPI	2017	12	1160
14	UPI	2018	1	1518

Insights : order count visibility by payment type to show preference of customer for payment type overt the time

Action : more facility for high preferred payment mode with lower payment failure to be made available

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

```
SELECT payment_type, COUNT (DISTINCT order_id) as order_count

FROM `scalaer-dsml-sql-405918.business_case_target.payments`

WHERE payment_sequential = payment_installments

#if these two are equal then we can say that all the installments have been paid

GROUP BY payment_type
```



Row	payment_type	order_count
1	not_defined	3
2	credit_card	25395
3	voucher	1621
4	UPI	19783
5	debit_card	1477

Insights : CREDIT card and UPI are hit in payment mode with completed payment and then

Action : these two payment modes to be further strengthened so that we can have low payment failure to improve customer experience