BUSINESS CASE: TARGET SQL

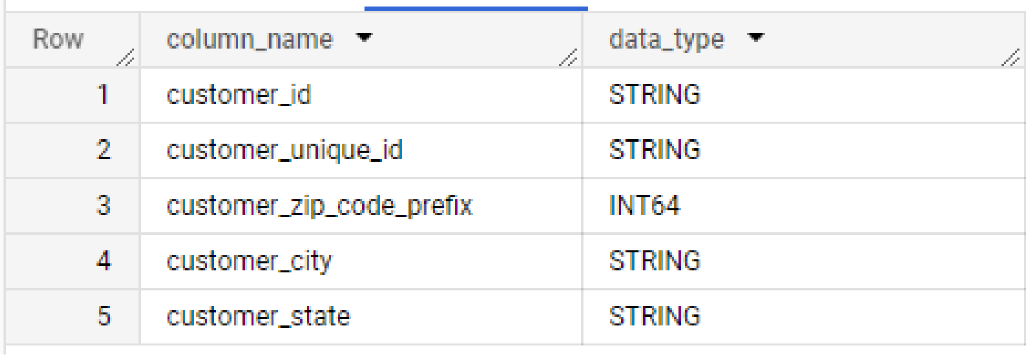
Target is a globally renowned brand and a prominent retailer in the United States. Target makes itself a preferred shopping destination by offering outstanding value, inspiration, innovation and an exceptional guest experience that no other retailer can deliver.
This particular business case focuses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

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.

ANSWER :

```
select
 column_name,
 data_type
from
`dsml-june23-beginner-390606.Business_CaseTarget_SQL`.INFORMATION_SCHEMA
.COLUMNS
where table_name = 'customers';
```



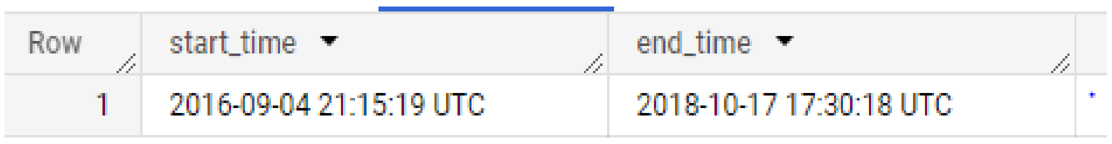
Row	column_name	data_type
1	customer_id	STRING
2	customer_unique_id	STRING
3	customer_zip_code_prefix	INT64
4	customer_city	STRING
5	customer_state	STRING

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

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

ANSWER :

```
select
 min(order_purchase_timestamp) as start_time,
 max(order_purchase_timestamp) as end_time
from `Business_CaseTarget_SQL.orders`
```



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

Insight:

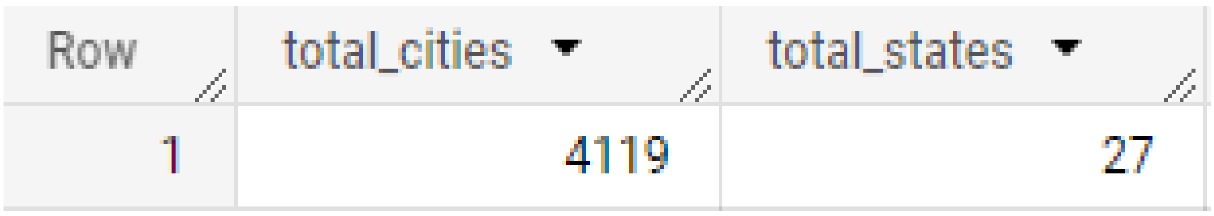
1. The first order placed on 2016-09-04.
2. The last order placed on 2018-10-17.
3. Orders were placed from 2016-09-04 to 2018-10-17.

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

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

ANSWER :

```
select
 count(distinct customer_city) as total_cities,
 count(distinct customer_state) as total_states
from `Business_CaseTarget_SQL.orders` as o
join `Business_CaseTarget_SQL.customers` as c
on o.customer_id = c.customer_id
```



Row	total_cities	total_states
1	4119	27

Insights:

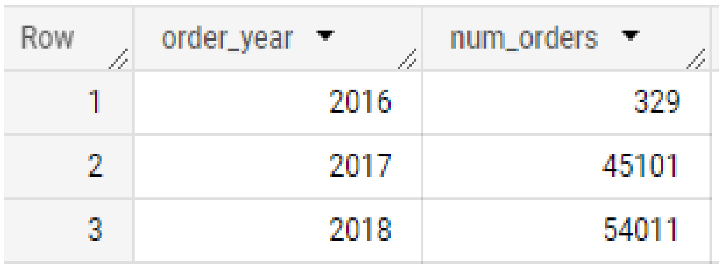
1. It was found that using 27 states there were 4119 cities who ordered in between 4-09-2016 to 17-10-2018.
2. It is recorded that highest count of customer's city in state named 'MG'
3. It is recorded that lowest count of customer city in state named 'RR'

2. In-depth Exploration

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

ANSWER :

```
select *
from(SELECT
  extract(year from order_purchase_timestamp) AS order_year,
  COUNT(order_id) AS num_orders
FROM `Business_CaseTarget_SQL.orders`
GROUP BY extract(year from order_purchase_timestamp))
order by order_year
```



Row	order_year	num_orders
1	2016	329
2	2017	45101
3	2018	54011

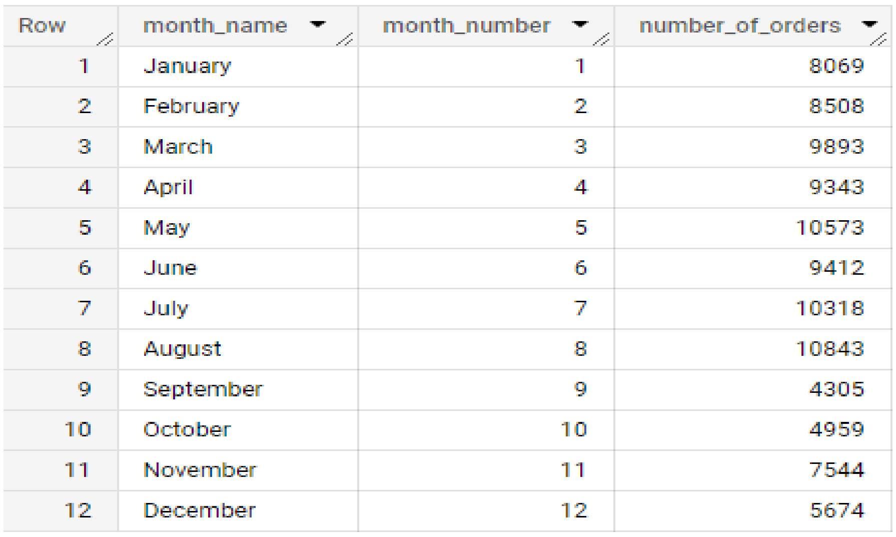
Insight: Yes count of order is increasing yearly where lowest order count was in 2016 and highest order count was in 2018

2. In-depth Exploration

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

ANSWER :

```
SELECT *
FROM
(SELECT FORMAT_DATETIME("%B",order_purchase_timestamp)
AS month_name, EXTRACT(MONTH FROM order_purchase_timestamp) AS
month_number, COUNT(order_id) AS number_of_orders
FROM `Business_CaseTarget_SQL.orders`
GROUP BY FORMAT_DATETIME("%B",order_purchase_timestamp),
EXTRACT(MONTH FROM order_purchase_timestamp))
ORDER BY month_number
```



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

Insights:

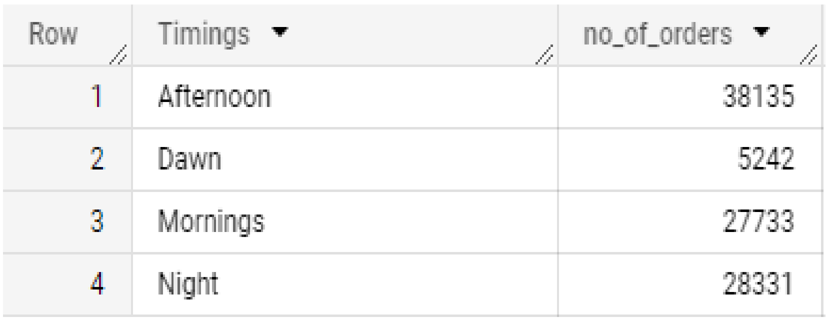
1. It is recorded that highest order were placed in August Month
2. It is recorded that lowest order were placed in September Month

2. In-depth Exploration

3. During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)
1. 0-6 hrs : Dawn
2. 7-12 hrs : Mornings
3. 13-18 hrs : Afternoon
4. 19-23 hrs : Night

ANSWER :

```
WITH CTE AS (
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 Timings, order_id
FROM `Business_CaseTarget_SQL.orders`)
SELECT Timings, COUNT(order_id) AS no_of_orders FROM CTE
GROUP BY Timings
ORDER BY no_of_orders
```



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

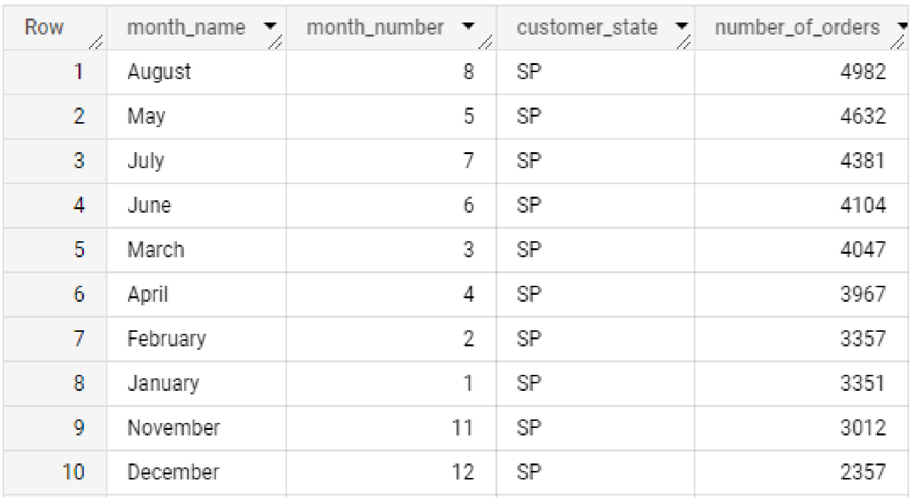
Insights :
1. Highest order were recorded at Afternoon time it is active hour
2. Dawn time is lazy hour for order placing

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

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

ANSWER :

```
 WITH CTE AS (
   SELECT FORMAT_DATETIME("%B", o.order_purchase_timestamp) AS
month_name, EXTRACT(MONTH FROM o.order_purchase_timestamp) AS
month_number, c.customer_state, o.order_id
   FROM `Business_CaseTarget_SQL.customers` c
   INNER JOIN `Business_CaseTarget_SQL.orders` o
   ON c.customer_id = o.customer_id)
SELECT month_name,month_number,customer_state , COUNT(order_id) as
number_of_orders
FROM CTE
GROUP BY month_name, month_number, customer_state
ORDER BY number_of_orders DESC
```



Row	month_name	month_number	customer_state	number_of_orders
1	August	8	SP	4982
2	May	5	SP	4632
3	July	7	SP	4381
4	June	6	SP	4104
5	March	3	SP	4047
6	April	4	SP	3967
7	February	2	SP	3357
8	January	1	SP	3351
9	November	11	SP	3012
10	December	12	SP	2357

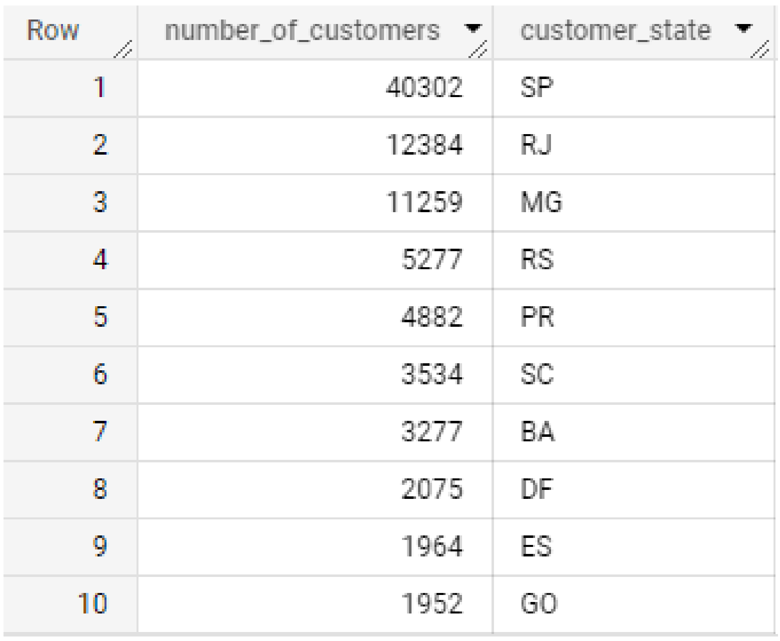
Insights: Highest order of each month were recorded in state named SP

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

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

ANSWER :

```
select
 count(distinct customer_unique_id) as
number_of_customers,
 customer_state
from `Business_CaseTarget_SQL.customers`
group by customer_state
order by number_of_customers desc
```



Row	number_of_customers	customer_state
1	40302	SP
2	12384	RJ
3	11259	MG
4	5277	RS
5	4882	PR
6	3534	SC
7	3277	BA
8	2075	DF
9	1964	ES
10	1952	GO

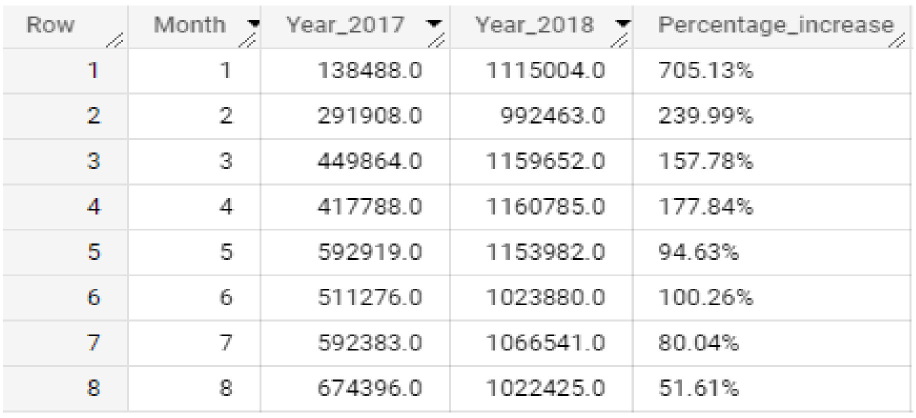
Insights: Highest count of customer is in state SP and lowest count of customer is in state RR

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). You can use the "payment_value" column in the payments table to get the cost of
orders.

ANSWER :

```
WITH CTE AS (
  select extract(year from o.order_purchase_timestamp) as year,
  EXTRACT(Month from o.order_purchase_timestamp) as Month,
round(sum(p.payment_value)) as total_payment
  from `Business_CaseTarget_SQL.orders` o
 inner join `Business_CaseTarget_SQL.payments` p
 on o.order_id = p.order_id
  group by extract(year from o.order_purchase_timestamp), EXTRACT(Month
from o.order_purchase_timestamp))
select t1.Month, t1.Year_2017, t2.Year_2018,concat(round(((t2.Year_2018 -
t1.Year_2017) / t1.Year_2017) * 100, 2) , '%') AS Percentage_increase
from(select total_payment as Year_2017, Month from cte
 where year = 2017 and Month between 1 and 8) t1
 inner join
 (select total_payment as Year_2018, Month from cte
 where year = 2018 and Month between 1 and 8) t2
 on t1.Month = t2.Month
 order by t1.Month
```



Row	Month	Year_2017	Year_2018	Percentage_increase
1	1	138488.0	1115004.0	705.13%
2	2	291908.0	992463.0	239.99%
3	3	449864.0	1159652.0	157.78%
4	4	417788.0	1160785.0	177.84%
5	5	592919.0	1153982.0	94.63%
6	6	511276.0	1023880.0	100.26%
7	7	592383.0	1066541.0	80.04%
8	8	674396.0	1022425.0	51.61%

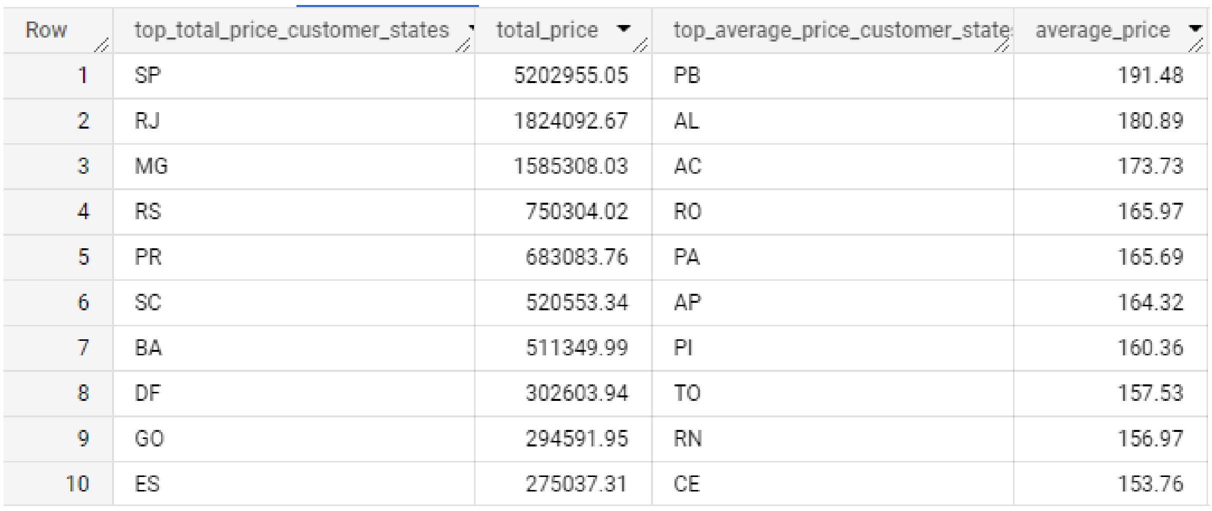
Insights : The overall percentage increase in the cost of orders from 2017 to 2018, including only the months from January to August, is 138.53%. Upon examining the month-wise increase, January shows the highest percentage increase, followed by February and April.

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

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

ANSWER :

```
WITH CTE AS
(SELECT c.customer_state, ROUND(SUM(i.price),2) AS total_price,
ROUND(AVG(i.price),2) AS average_price
FROM `Business_CaseTarget_SQL.orders` o
INNER JOIN `Business_CaseTarget_SQL.order_items` i
ON o.order_id = i.order_id
INNER JOIN `Business_CaseTarget_SQL.customers` c
ON c.customer_id = o.customer_id
GROUP BY c.customer_state)
SELECT t1.customer_state AS top_total_price_customer_states,
t1.total_price,t2.customer_state AS top_average_price_customer_states,
t2.average_price
FROM(SELECT ROW_NUMBER() OVER(ORDER BY total_price DESC) AS r1,
customer_state, total_price FROM CTE) t1
INNER JOIN
(SELECT ROW_NUMBER() OVER(ORDER BY average_price DESC) AS r2,
customer_state, average_price FROM CTE) t2
ON t1.r1 = t2.r2
```



Row	top_total_price_customer_states	total_price	top_average_price_customer_state	average_price
1	SP	5202955.05	PB	191.48
2	RJ	1824092.67	AL	180.89
3	MG	1585308.03	AC	173.73
4	RS	750304.02	RO	165.97
5	PR	683083.76	PA	165.69
6	SC	520553.34	AP	164.32
7	BA	511349.99	PI	160.36
8	DF	302603.94	TO	157.53
9	GO	294591.95	RN	156.97
10	ES	275037.31	CE	153.76

Insights:

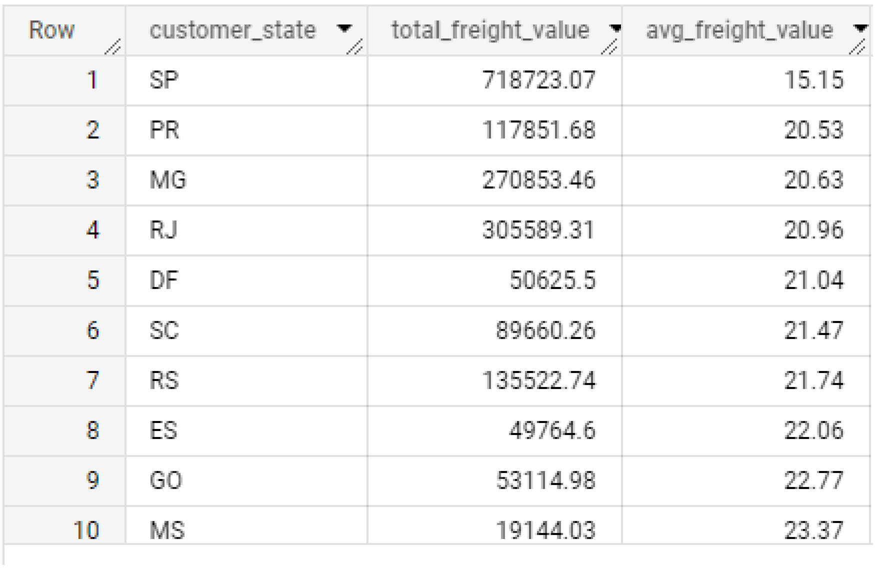
1. Highest total price of order is with state PB
2. Lowest Total price is with SP

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

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

ANSWER :

```
SELECT
c.customer_state,
ROUND(SUM(i.freight_value), 2) AS total_freight_value,
ROUND(AVG(i.freight_value), 2) AS avg_freight_value
FROM `Business_CaseTarget_SQL.orders` as o
JOIN `Business_CaseTarget_SQL.order_items` as i
ON o.order_id = i.order_id
JOIN `Business_CaseTarget_SQL.customers` as c
ON o.customer_id = c.customer_id
GROUP BY c.customer_state
ORDER BY avg_freight_value
```



Row	customer_state	total_freight_value	avg_freight_value
1	SP	718723.07	15.15
2	PR	117851.68	20.53
3	MG	270853.46	20.63
4	RJ	305589.31	20.96
5	DF	50625.5	21.04
6	SC	89660.26	21.47
7	RS	135522.74	21.74
8	ES	49764.6	22.06
9	GO	53114.98	22.77
10	MS	19144.03	23.37

Insights:

1. Highest Total freight value for order with state SP and lowest Total freight value is with RR.

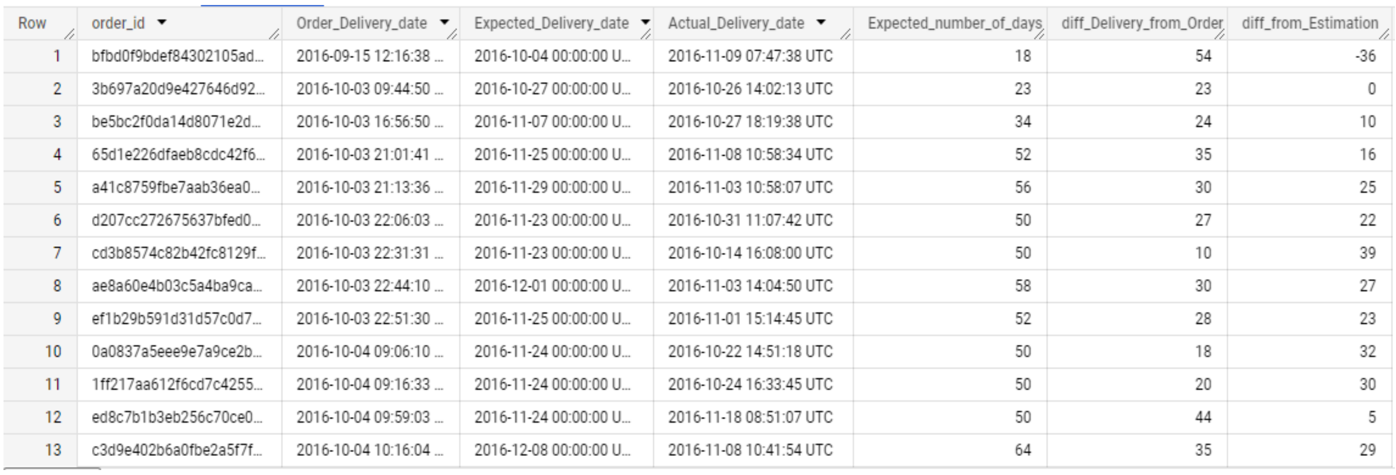
5. Analysis based on sales, freight and delivery time.

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.

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

ANSWER :

```
select
 order_id,
 date_diff (order_delivered_customer_date, order_purchase_timestamp,day) as
delivered_date,
date_diff(order_estimated_delivery_date,order_delivered_customer_date, day) as
estimated_minus_actual_delivery_days
from `Business_CaseTarget_SQL.orders`
where date_diff(order_delivered_customer_date,order_purchase_timestamp, day)
is not null
order by delivered_date desc, estimated_minus_actual_delivery_days
```



Row	order_id	Order_Delivery_date	Expected_Delivery_date	Actual_Delivery_date	Expected_number_of_days	diff_Delivery_from_Order	diff_from_Estimation
1	bfbd0f9bdef84302105ad...	2016-09-15 12:16:38 ...	2016-10-04 00:00:00 U...	2016-11-09 07:47:38 UTC	18	54	-36
2	3b697a20d9e427646d92...	2016-10-03 09:44:50 ...	2016-10-27 00:00:00 U...	2016-10-26 14:02:13 UTC	23	23	0
3	be5bc2f0da14d8071e2d...	2016-10-03 16:56:50 ...	2016-11-07 00:00:00 U...	2016-10-27 18:19:38 UTC	34	24	10
4	65d1e226dfaeb8cdc42f6...	2016-10-03 21:01:41 ...	2016-11-25 00:00:00 U...	2016-11-08 10:58:34 UTC	52	35	16
5	a41c8759fbe7aab36ea0...	2016-10-03 21:13:36 ...	2016-11-29 00:00:00 U...	2016-11-03 10:58:07 UTC	56	30	25
6	d207cc272675637bfed0...	2016-10-03 22:06:03 ...	2016-11-23 00:00:00 U...	2016-10-31 11:07:42 UTC	50	27	22
7	cd3b8574c82b42fc8129f...	2016-10-03 22:31:31 ...	2016-11-23 00:00:00 U...	2016-10-14 16:08:00 UTC	50	10	39
8	ae8a60e4b03c5a4ba9ca...	2016-10-03 22:44:10 ...	2016-12-01 00:00:00 U...	2016-11-03 14:04:50 UTC	58	30	27
9	ef1b29b591d31d57c0d7...	2016-10-03 22:51:30 ...	2016-11-25 00:00:00 U...	2016-11-01 15:14:45 UTC	52	28	23
10	0a0837a5eee9e7a9ce2b...	2016-10-04 09:06:10 ...	2016-11-24 00:00:00 U...	2016-10-22 14:51:18 UTC	50	18	32
11	1ff217aa612f6cd7c4255...	2016-10-04 09:16:33 ...	2016-11-24 00:00:00 U...	2016-10-24 16:33:45 UTC	50	20	30
12	ed8c7b1b3eb256c70ce0...	2016-10-04 09:59:03 ...	2016-11-24 00:00:00 U...	2016-11-18 08:51:07 UTC	50	44	5
13	c3d9e402b6a0fbe2a5f7f...	2016-10-04 10:16:04 ...	2016-12-08 00:00:00 U...	2016-11-08 10:41:54 UTC	64	35	29

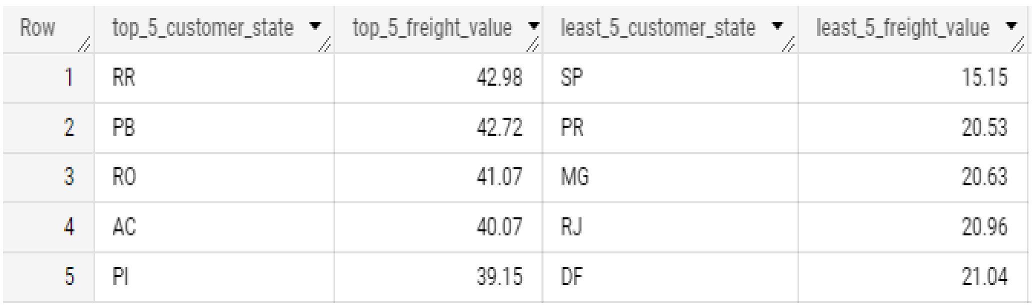
Insites :. Here negative Values in column diff_from_estimation represent delay in days from expected and positive values presented before estimated and 0 present deliver as expected.

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

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

ANSWER :

```
WITH CTE AS
(SELECT  c.customer_state, ROUND(AVG(i.freight_value),2) AS
avg_value
FROM `Business_CaseTarget_SQL.order_items` i
INNER JOIN `Business_CaseTarget_SQL.orders` o
ON i.order_id = o.order_id
INNER JOIN `Business_CaseTarget_SQL.customers` c
ON c.customer_id = o.customer_id
group by c.customer_state)
SELECT c1.customer_state AS top_5_customer_state, c1.avg_value
AS top_5_freight_value, c2.customer_state AS
least_5_customer_state, c2.avg_value AS least_5_freight_value
FROM(SELECT ROW_NUMBER() OVER(ORDER BY avg_value DESC) AS r1,
customer_state, CTE.avg_value FROM CTE) c1
INNER JOIN
(SELECT ROW_NUMBER() OVER(ORDER BY avg_value ASC) AS r2,
customer_state, CTE.avg_value FROM CTE) c2
ON c1.r1 = c2.r2
```



Row	top_5_customer_state	top_5_freight_value	least_5_customer_state	least_5_freight_value
1	RR	42.98	SP	15.15
2	PB	42.72	PR	20.53
3	RO	41.07	MG	20.63
4	AC	40.07	RJ	20.96
5	PI	39.15	DF	21.04

Insights:

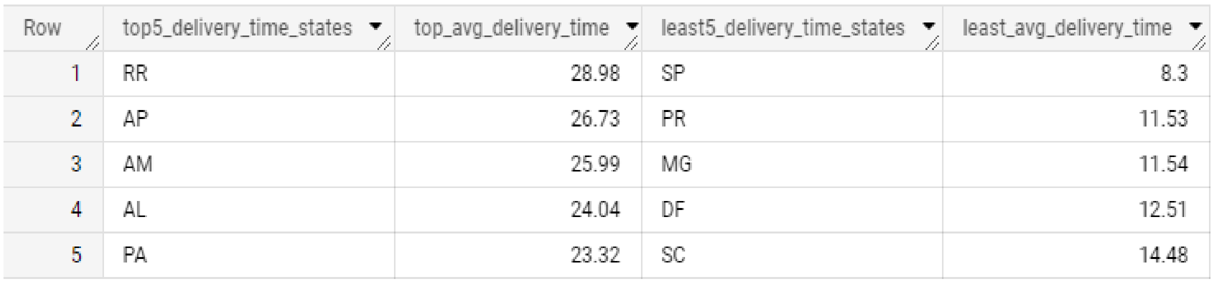
1. Top 5 states with the highest average freight value are RR, PB, RO, AC and PI.
2. Top 5 states with the highest average freight value are SP, PR, MG, RJ and DF.

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

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

ANSWER :

```
WITH CTE AS
(SELECT  c.customer_state, ROUND(AVG(TIMESTAMP_DIFF
(order_delivered_customer_date, order_purchase_timestamp, day)),2)
AS delivery_timestamp_in_days
FROM `Business_CaseTarget_SQL.orders` o
INNER JOIN `Business_CaseTarget_SQL.customers` c
ON c.customer_id = o.customer_id
GROUP BY c.customer_state
)
SELECT c1.customer_state AS top5_delivery_time_states,
c1.delivery_timestamp_in_days AS top_avg_delivery_time,
c2.customer_state AS least5_delivery_time_states,
c2.delivery_timestamp_in_days AS least_avg_delivery_time
FROM
(SELECT ROW_NUMBER() OVER(ORDER BY delivery_timestamp_in_days DESC)
AS r1, customer_state, CTE.delivery_timestamp_in_days FROM CTE) c1
INNER JOIN
(SELECT ROW_NUMBER() OVER(ORDER BY delivery_timestamp_in_days ASC)
AS r2, customer_state, CTE.delivery_timestamp_in_days FROM CTE) c2
ON c1.r1 = c2.r2
WHERE r1 <=5 AND r2 <=5
```



Row	top5_delivery_time_states	top_avg_delivery_time	least5_delivery_time_states	least_avg_delivery_time
1	RR	28.98	SP	8.3
2	AP	26.73	PR	11.53
3	AM	25.99	MG	11.54
4	AL	24.04	DF	12.51
5	PA	23.32	SC	14.48

Insights:

1. Top 5 states with the highest average delivery time are RR, AP, AM, AL, PA.
2. Top 5 states with the least average delivery time are AP, PR, MG, DF, SC.

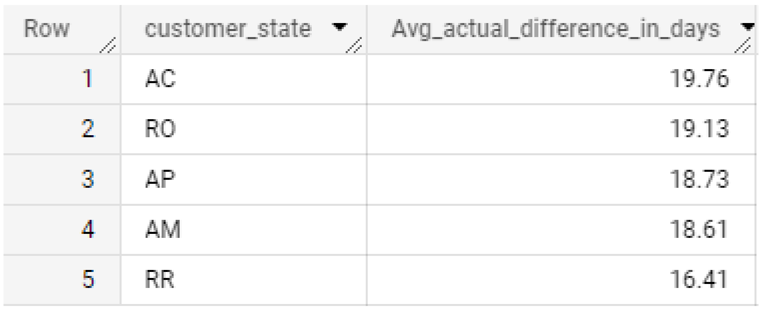
5. Analysis based on sales, freight and delivery time.

4. 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.

ANSWER :

```
WITH CTE AS
(SELECT c.customer_state,
ROUND(AVG(TIMESTAMP_DIFF(o.order_estimated_delivery_date,
o.order_delivered_customer_date, DAY)), 2) AS
Avg_actual_difference_in_days
FROM `Business_CaseTarget_SQL.orders` AS o
INNER JOIN `Business_CaseTarget_SQL.customers` AS c
ON o.customer_id = c.customer_id
WHERE order_delivered_customer_date is NOT NULL AND
order_estimated_delivery_date is NOT NULL
GROUP BY c.customer_state)
SELECT customer_state, Avg_actual_difference_in_days
FROM (SELECT *, ROW_NUMBER() OVER(ORDER BY
Avg_actual_difference_in_days DESC) AS numbering FROM CTE)
WHERE numbering <=5
```



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

Insights:

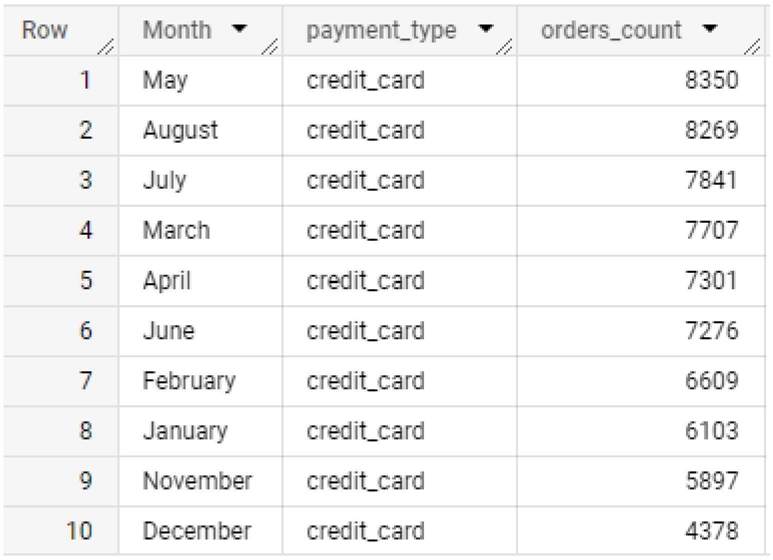
1. Top 5 state where orders are deliver on average of 16-19 days before are AC,RO,AP,AM,RR

6. Analysis based on the payments:

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

ANSWER :

```
SELECT * FROM
(SELECT FORMAT_DATETIME("%B",o.order_purchase_timestamp) AS Month,
p.payment_type,
COUNT(o.order_id) AS orders_count
FROM `Business_CaseTarget_SQL.orders` o
INNER JOIN `Business_CaseTarget_SQL.payments` p
ON o.order_id = p.order_id
GROUP BY FORMAT_DATETIME("%B",o.order_purchase_timestamp),p.payment_type)
ORDER BY orders_count
```



Row	Month	payment_type	orders_count
1	May	credit_card	8350
2	August	credit_card	8269
3	July	credit_card	7841
4	March	credit_card	7707
5	April	credit_card	7301
6	June	credit_card	7276
7	February	credit_card	6609
8	January	credit_card	6103
9	November	credit_card	5897
10	December	credit_card	4378

Insights:

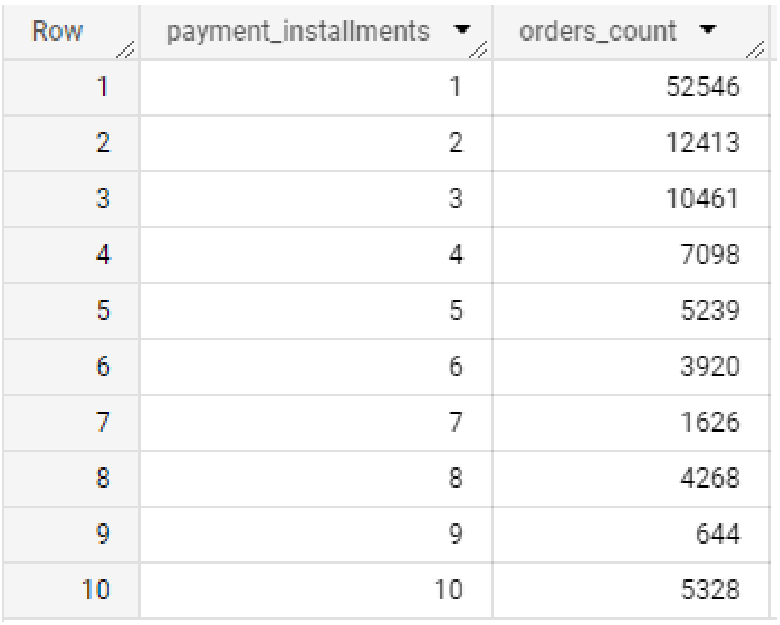
1. It is recorded that in every month maximum order's payments are done with credit card.

6. Analysis based on the payments:

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

ANSWER :

```
SELECT
 payment_installments,
 COUNT(o.order_id) as orders_count
FROM `Business_CaseTarget_SQL.orders` o
INNER JOIN `Business_CaseTarget_SQL.payments` p
ON o.order_id = p.order_id
WHERE payment_installments >=1
GROUP BY p.payment_installments
ORDER BY p.payment_installments
```



Row	payment_installments	orders_count
1	1	52546
2	2	12413
3	3	10461
4	4	7098
5	5	5239
6	6	3920
7	7	1626
8	8	4268
9	9	644
10	10	5328

Insights:

1. 1st Payment installment has been paid for 52546 orders.
2. 2nd Payment has been paid for 12413 orders.
3. At last, in the dataset 18 orders have the 24 payment installment which has been paid.