TARGET-BUSINESS CASE STUDY PROJECT

Introduction:

Target, the renowned and thriving retailer from the United States, has made its mark in Brazil, aiming to become the preferred shopping destination in the region. With a commitment to delivering exceptional value, innovation, and an unparalleled guest experience, Target seeks to differentiate itself from other retailers in the country.

This analysis delves into a dataset of 100,000 orders spanning 2016 to 2018, providing valuable insights into various aspects of Target's operations in Brazil. From order processing and pricing strategies to customer demographics and satisfaction levels, we aim to uncover crucial information that can optimize Target's performance and success in the Brazilian market. Let's explore the data to gain valuable business intelligence and enhance Target's presence in Brazil.

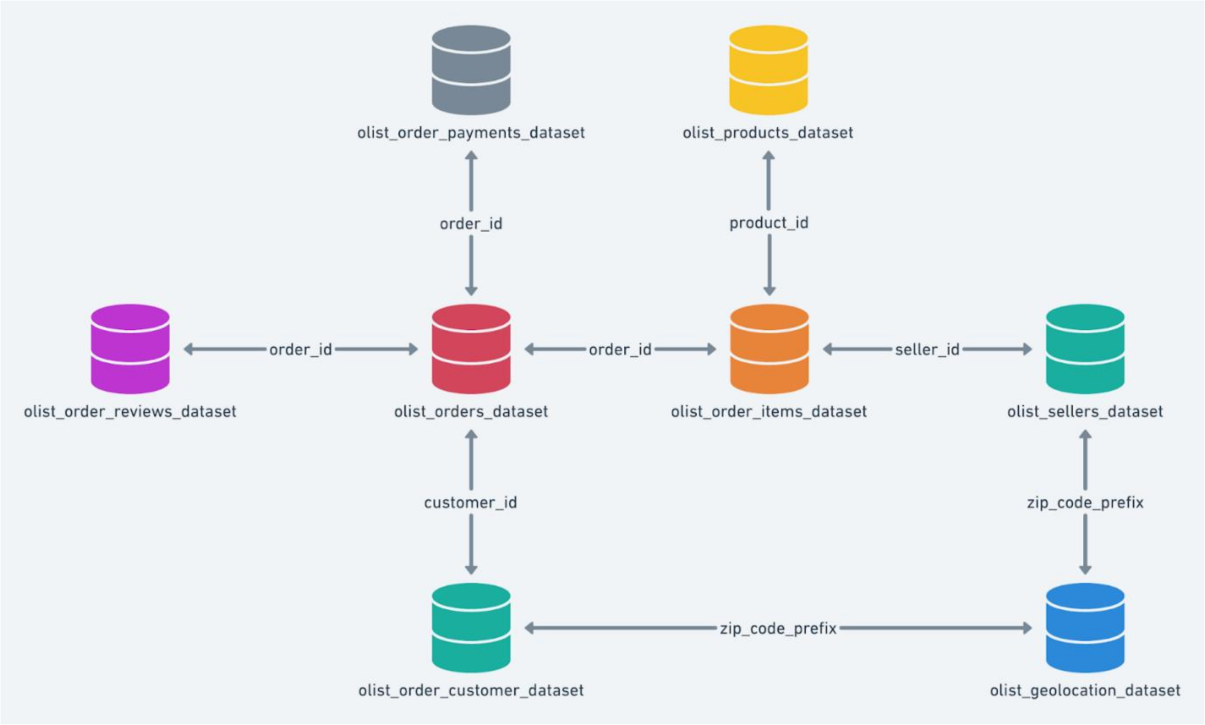
Datasets:

This case study utilizes data from the following eight CSV files:

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

These datasets provide essential information on various aspects of Target's operations in Brazil, such as customer profiles, product details, order processing, payment transactions, customer reviews, and seller attributes. By analyzing this data, we aim to gain valuable insights that can contribute to optimizing Target's performance and enhancing the overall customer experience in the Brazilian market.

All these tables are interrelated with each other. Attached below is the Entity-Relationship (ER) diagram depicting the relationships among these tables.



Data Analysis and Findings:

Now, let's embark on our journey of exploration and uncover valuable insights as we dive into the question session.

What does ‘good’ look like?

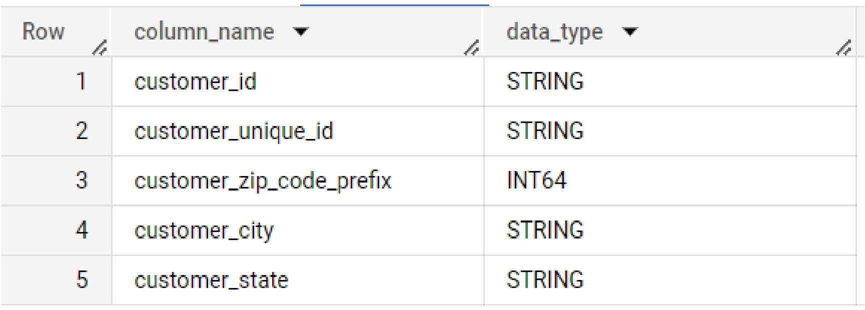
1.Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset:

1.1. Data type of all columns in the “customers” table

Query:

```
select
  column_name,
  data_type
from target.INFORMATION_SCHEMA.COLUMNS
where table_name = "customers"
```

Result:



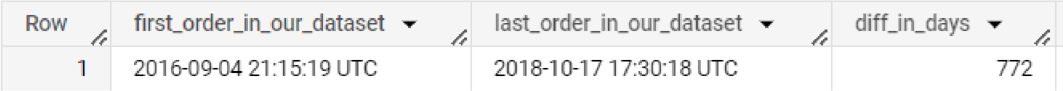
Row	column_name	data_type
1	customer_id	STRING
2	customer_unique_id	STRING
3	customer_zip_code_prefix	INT64
4	customer_city	STRING
5	customer_state	STRING

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

Query:

```
select
  min(order_purchase_timestamp)first_order_in_our_dataset,
  max(order_purchase_timestamp)last_order_in_our_dataset,
  date_diff(max(order_purchase_timestamp),min(order_purchase_timestamp),day)diff_
in_days
from
  `target.orders`
```

Result:



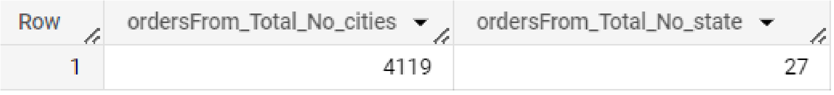
Row	first_order_in_our_dataset	last_order_in_our_dataset	diff_in_days
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC	772

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

Query:

```
select
  count(distinct c.customer_city)ordersFrom_Total_No_cities,
  count(distinct c.customer_state)ordersFrom_Total_No_state
from `target.orders` o
join
`target.customers` c
on o.customer_id = c.customer_id
```

Result:



Row	ordersFrom_Total_No_cities	ordersFrom_Total_No_state
1	4119	27

2. In-depth Exploration:

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

Query:

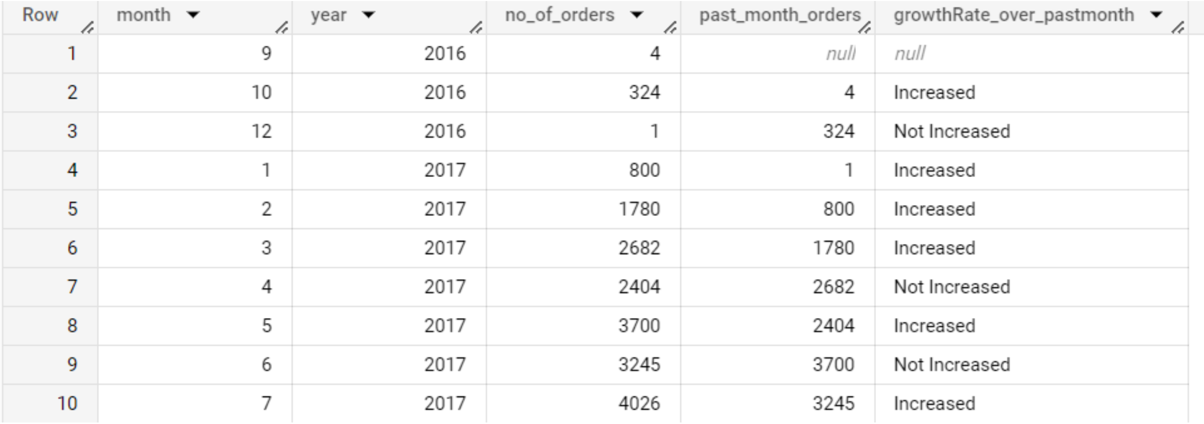
```
with monthsYear as

   (select order_id,
   extract(month from order_purchase_timestamp)month,
   extract(year from order_purchase_timestamp)year,
   from `target.orders`),

   ordersPerMonth as
   (select month,year,count(order_id)no_of_orders,
     from monthsYear
     group by month,year),
   growthRate as
   (select month,year,no_of_orders,
   lag(no_of_orders)over(order by year,month)past_month_orders
   from ordersPerMonth
   order by year,month
   )

 select * ,
 case
 when no_of_orders > past_month_orders then "Increased"
 when past_month_orders is null then null
 else "Not Increased"
 end growthRate_over_pastmonth
 from growthRate
```

Result:



Row	month	year	no_of_orders	past_month_orders	growthRate_over_pastmonth
1	9	2016	4	null	null
2	10	2016	324	4	Increased
3	12	2016	1	324	Not Increased
4	1	2017	800	1	Increased
5	2	2017	1780	800	Increased
6	3	2017	2682	1780	Increased
7	4	2017	2404	2682	Not Increased
8	5	2017	3700	2404	Increased
9	6	2017	3245	3700	Not Increased
10	7	2017	4026	3245	Increased

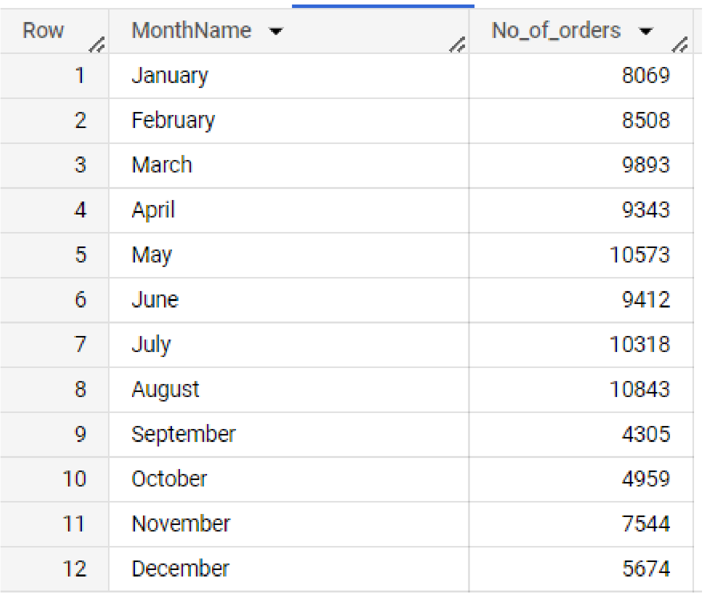
2.2. Can we see some kind of monthly seasonality in terms of the no. of orders being placed?

Query:

```
with extractedMonth as
  (select
    order_id,
    format_timestamp("%B",order_purchase_timestamp)MonthName,
   from target.orders
   order by extract(month from order_purchase_timestamp))

select
  MonthName,
  count(order_id)No_of_orders
from extractedMonth
group by MonthName
order by parse_date("%B", MonthName)
```

Result:



Row	MonthName	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

2.3. 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

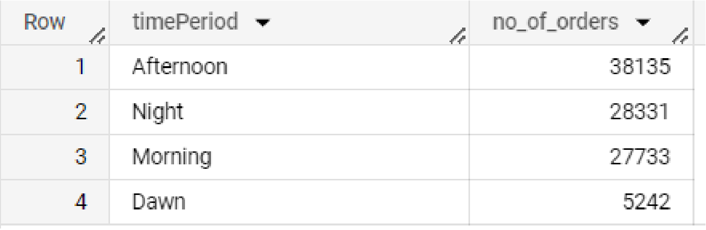
Query:

```
with timeInterval as

  (select
    order_id,
    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
"Morning"
      when extract(hour from order_purchase_timestamp) between 13 and 18 then
"Afternoon"
      when extract(hour from order_purchase_timestamp) between 19 and 23 then
"Night"
      else NULL
    end timePeriod
  from `target.orders`)

select
  timePeriod,
  count(order_id)no_of_orders
from timeInterval
where timePeriod is not null
group by timePeriod
order by no_of_orders desc
```

Result:



Row	timePeriod	no_of_orders
1	Afternoon	38135
2	Night	28331
3	Morning	27733
4	Dawn	5242

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

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

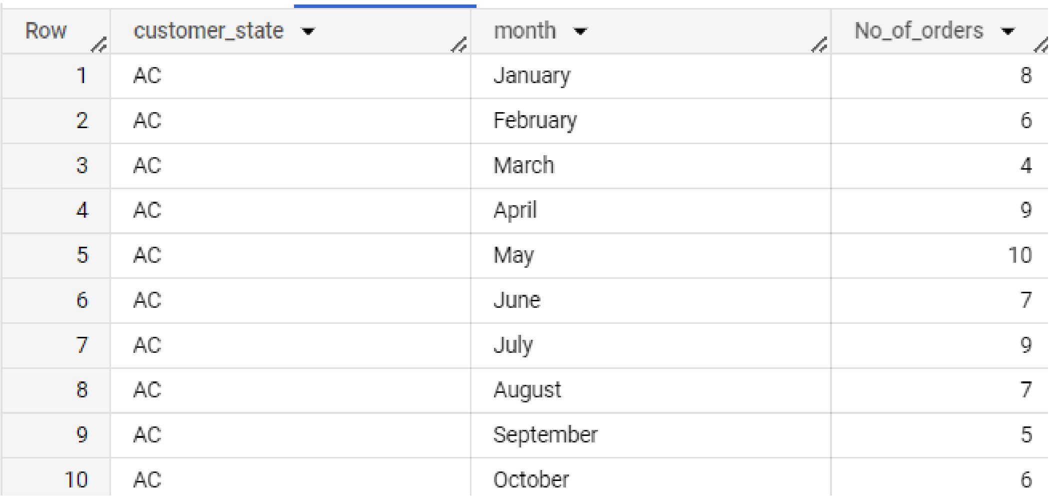
```
with customerandstate as

  (select
    o.order_id,
    format_timestamp("%B",o.order_purchase_timestamp)month,
    c.customer_state
  from `target.orders` o
join  `target.customers` c
on o.customer_id = c.customer_id)

select
  customer_state,
  month,
  count(order_id)No_of_orders
from customerandstate
group by customer_state,month
order by customer_state, parse_date("%B",month)
```

Result:

Note: The query output contains 322 rows, and for brevity, only a sample of 10 representative rows is presented below



Row	customer_state	month	No_of_orders
1	AC	January	8
2	AC	February	6
3	AC	March	4
4	AC	April	9
5	AC	May	10
6	AC	June	7
7	AC	July	9
8	AC	August	7
9	AC	September	5
10	AC	October	6

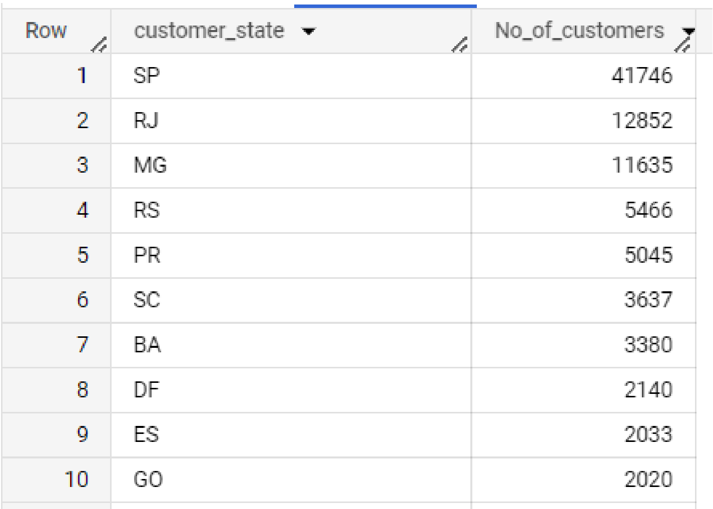
3.2 How are the customers distributed across all the states?

Query:

```
select
  customer_state,
  count(distinct customer_id)No_of_customers
from `target.customers`
group by customer_state
order by No_of_customers desc
```

Result:

Note: The query output contains 27rows, and for brevity, only a sample of 10 representative rows is presented below



Row	customer_state	No_of_customers
1	SP	41746
2	RJ	12852
3	MG	11635
4	RS	5466
5	PR	5045
6	SC	3637
7	BA	3380
8	DF	2140
9	ES	2033
10	GO	2020

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

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

Query:

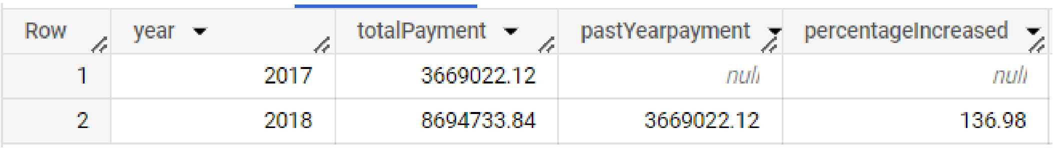
```
with orderNpayment as
(
    select
      p.payment_value,
      extract(year from o.order_purchase_timestamp )year
    from `target.orders` o
    join
    `target.payments` p
    on o.order_id=p.order_id

    where
    extract(year from o.order_purchase_timestamp) in (2017,2018)
    and
    extract(month from o.order_purchase_timestamp) between 1 and 8
),
totalPayment as
(
    select
      year,
      round(sum(payment_value),2)totalPayment,
      lag(round(sum(payment_value),2)) over(order by year)as pastYearpayment

    from orderNpayment
    group by year
    order by year
)
select *,
if(tp.pastYearpayment is not null,
round(((tp.totalPayment -tp. pastYearpayment) / tp.pastYearpayment) * 100, 2)
,null)AS percentageIncreased
```

```
from totalPayment as tp
```

Result:



Row	year	totalPayment	pastYearpayment	percentageIncreased
1	2017	3669022.12	null	null
2	2018	8694733.84	3669022.12	136.98

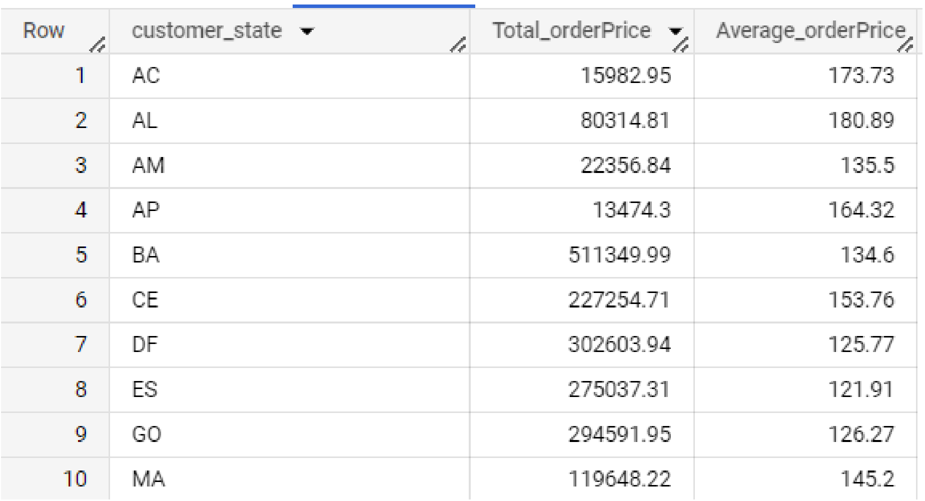
4.2. Calculate the Total & Average value of order price for each state.

Query:

```
select c.customer_state,
  round(sum(oi.price),2)Total_orderPrice,
  round(avg(oi.price),2)Average_orderPrice
from `target.orders` o
join
`target.customers` c
on o.customer_id = c.customer_id
join
`target.order_items` oi
on o.order_id = oi.order_id
group by c.customer_state
order by c.customer_state
```

Result:

Note: The query output contains 27rows, and for brevity, only a sample of 10 representative rows is presented below



Row	customer_state	Total_orderPrice	Average_orderPrice
1	AC	15982.95	173.73
2	AL	80314.81	180.89
3	AM	22356.84	135.5
4	AP	13474.3	164.32
5	BA	511349.99	134.6
6	CE	227254.71	153.76
7	DF	302603.94	125.77
8	ES	275037.31	121.91
9	GO	294591.95	126.27
10	MA	119648.22	145.2

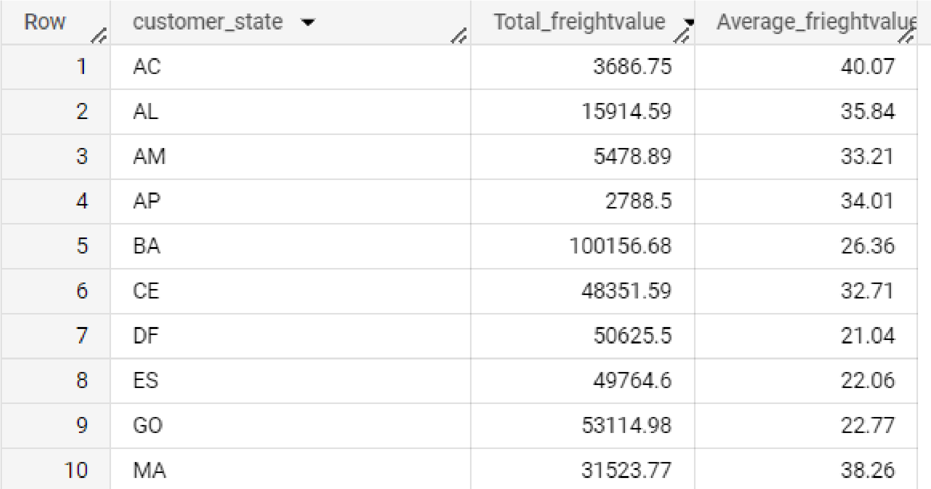
4.3 Calculate the Total & Average value of order freight for each state

Query:

```
select c.customer_state,
  round(sum(oi.freight_value),2)Total_freightvalue,
  round(avg(oi.freight_value),2)Average_frieghtvalue
from `target.orders` o
join
`target.customers` c
on o.customer_id = c.customer_id
join
`target.order_items` oi
on o.order_id = oi.order_id
group by c.customer_state
order by c.customer_state
```

Result:

Note: The query output contains 27rows, and for brevity, only a sample of 10 representative rows is presented below



Row	customer_state	Total_freightvalue	Average_frieghtvalue
1	AC	3686.75	40.07
2	AL	15914.59	35.84
3	AM	5478.89	33.21
4	AP	2788.5	34.01
5	BA	100156.68	26.36
6	CE	48351.59	32.71
7	DF	50625.5	21.04
8	ES	49764.6	22.06
9	GO	53114.98	22.77
10	MA	31523.77	38.26

5. Analysis based on sales, freight and delivery time

5.1 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.

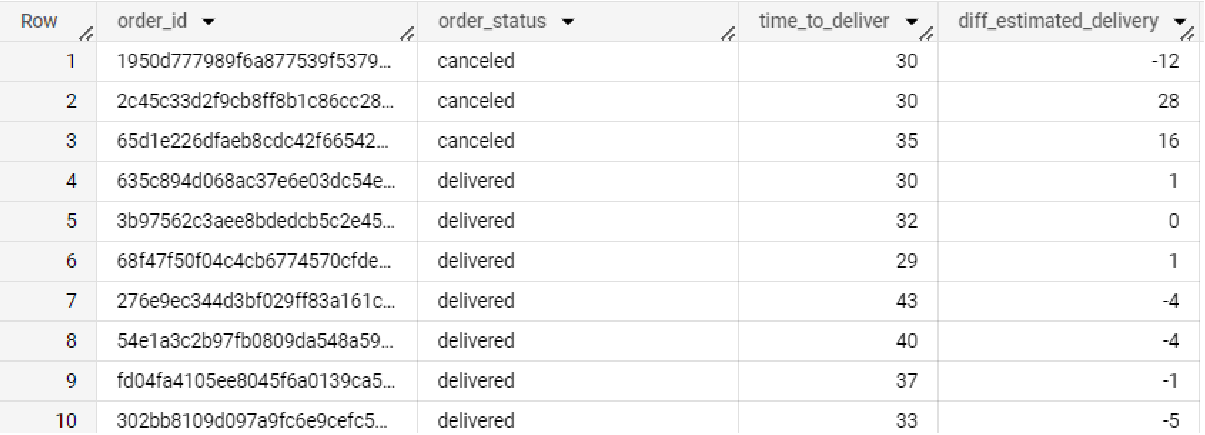
Do this in a single query.

Query:

```
select
  order_id,
  order_status,
  date_diff(order_delivered_customer_date,order_purchase_timestamp,day)time_to_de
liver,
  date_diff(order_estimated_delivery_date,order_delivered_customer_date,day)diff_
estimated_delivery

from `target.orders`
where order_delivered_customer_date is not null
```

Result:



Row	order_id	order_status	time_to_deliver	diff_estimated_delivery
1	1950d777989f6a877539f5379...	canceled	30	-12
2	2c45c33d2f9cb8ff8b1c86cc28...	canceled	30	28
3	65d1e226dfaeb8cdc42f66542...	canceled	35	16
4	635c894d068ac37e6e03dc54e...	delivered	30	1
5	3b97562c3aee8bdedcb5c2e45...	delivered	32	0
6	68f47f50f04c4cb6774570cfde...	delivered	29	1
7	276e9ec344d3bf029ff83a161c...	delivered	43	-4
8	54e1a3c2b97fb0809da548a59...	delivered	40	-4
9	fd04fa4105ee8045f6a0139ca5...	delivered	37	-1
10	302bb8109d097a9fc6e9cefc5...	delivered	33	-5

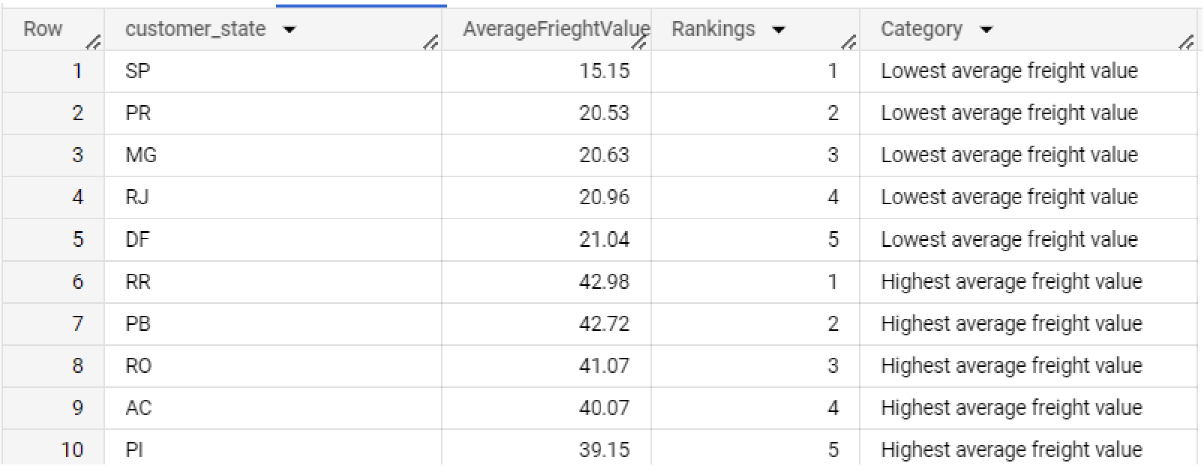
5.2 Find out the top 5 states with the highest & lowest average freight value

```
(select
  c.customer_state,
  round(avg(oi.freight_value),2)AverageFrieghtValue,
  dense_rank()over(order by avg(oi.freight_value)desc)Rankings,
  "Highest average freight value" Category
from `target.orders` o
join
`target.customers` c
on o.customer_id = c.customer_id
join
`target.order_items` oi
on o.order_id = oi.order_id

group by c.customer_state
order by AverageFrieghtValue desc
limit 5)
union all
```

```
(select
  c.customer_state,
  round(avg(oi.freight_value),2)AverageFrieghtValue,
  dense_rank()over(order by avg(oi.freight_value)asc)Rankings,
  "Lowest average freight value"Category
from `target.orders` o
join
`target.customers` c
on o.customer_id = c.customer_id
join
`target.order_items` oi
on o.order_id = oi.order_id
group by c.customer_state
order by AverageFrieghtValue desc
limit 5)
```

Result:



Row	customer_state	AverageFrieghtValue	Rankings	Category
1	SP	15.15	1	Lowest average freight value
2	PR	20.53	2	Lowest average freight value
3	MG	20.63	3	Lowest average freight value
4	RJ	20.96	4	Lowest average freight value
5	DF	21.04	5	Lowest average freight value
6	RR	42.98	1	Highest average freight value
7	PB	42.72	2	Highest average freight value
8	RO	41.07	3	Highest average freight value
9	AC	40.07	4	Highest average freight value
10	PI	39.15	5	Highest average freight value

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

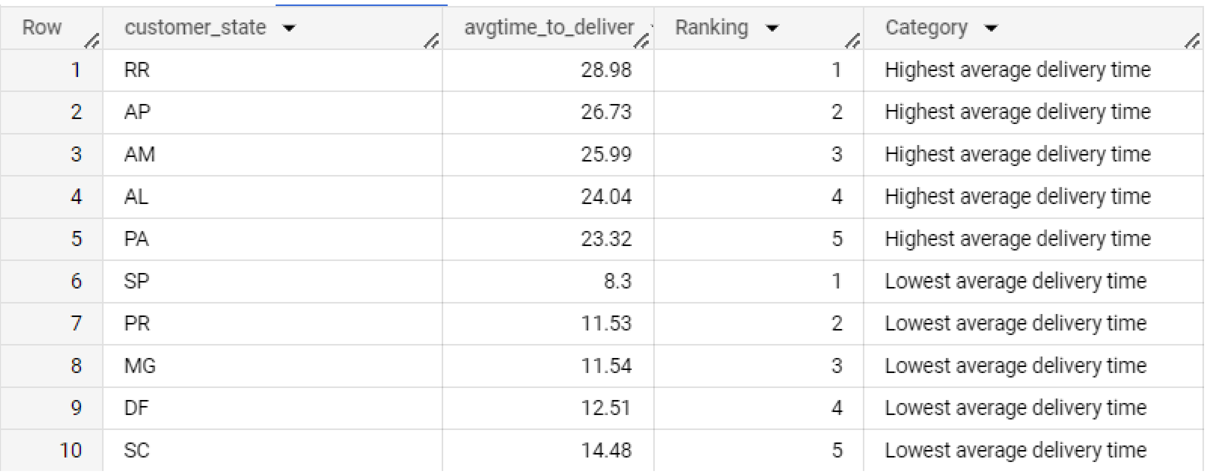
Query:

```
(select
  c.customer_state,
  round(avg(date_diff(o.order_delivered_customer_date,o.order_purchase_timestamp,
day)),2)avgtime_to_deliver,
  dense_rank()over(order by
round(avg(date_diff(o.order_delivered_customer_date,o.order_purchase_timestamp,da
y)),2)desc)Ranking ,
  "Highest average delivery time" Category
from `target.orders` o
join
`target.customers` c
on o.customer_id = c.customer_id
group by c.customer_state
order by avgtime_to_deliver desc
limit 5)

union all

(select
  c.customer_state,
  round(avg(date_diff(o.order_delivered_customer_date,o.order_purchase_timestamp,
day)),2)avgtime_to_deliver,
  dense_rank()over(order by
round(avg(date_diff(o.order_delivered_customer_date,o.order_purchase_timestamp,da
y)),2))Ranking ,
  "Lowest average delivery time" Category
from `target.orders` o
join
`target.customers` c
on o.customer_id = c.customer_id
group by c.customer_state
order by avgtime_to_deliver
limit 5)
```

Result:



Row	customer_state	avgtime_to_deliver	Ranking	Category
1	RR	28.98	1	Highest average delivery time
2	AP	26.73	2	Highest average delivery time
3	AM	25.99	3	Highest average delivery time
4	AL	24.04	4	Highest average delivery time
5	PA	23.32	5	Highest average delivery time
6	SP	8.3	1	Lowest average delivery time
7	PR	11.53	2	Lowest average delivery time
8	MG	11.54	3	Lowest average delivery time
9	DF	12.51	4	Lowest average delivery time
10	SC	14.48	5	Lowest average delivery time

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

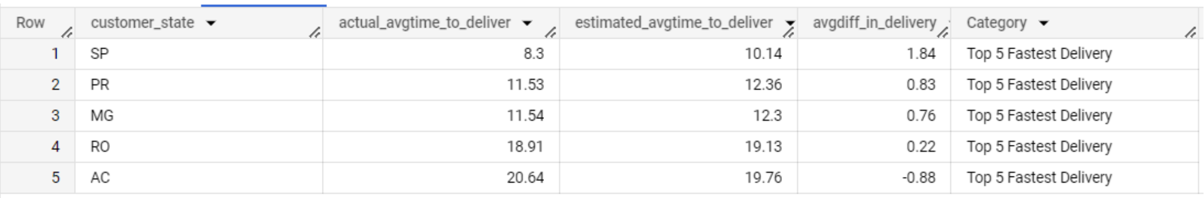
Query:

```
with avgtime_to_deliver as

  (select
    c.customer_state,
    round(avg(date_diff(o.order_delivered_customer_date,o.order_purchase_timestam
p,day)),2)actual_avgtime_to_deliver,
    round(avg(date_diff(o.order_estimated_delivery_date,o.order_delivered_custome
r_date,day)),2)estimated_avgtime_to_deliver,
  from `target.orders` o
  join
  `target.customers` c
  on o.customer_id = c.customer_id
  where o.order_delivered_customer_date is not null
  group by c.customer_state)

select
  *,
  round((estimated_avgtime_to_deliver -
actual_avgtime_to_deliver),2)avgdiff_in_delivery,
  "Top 5 Fastest Delivery" as Category
from avgtime_to_deliver
order by avgdiff_in_delivery desc
limit 5
```

Result:



Row	customer_state	actual_avgtime_to_deliver	estimated_avgtime_to_deliver	avgdiff_in_delivery	Category
1	SP	8.3	10.14	1.84	Top 5 Fastest Delivery
2	PR	11.53	12.36	0.83	Top 5 Fastest Delivery
3	MG	11.54	12.3	0.76	Top 5 Fastest Delivery
4	RO	18.91	19.13	0.22	Top 5 Fastest Delivery
5	AC	20.64	19.76	-0.88	Top 5 Fastest Delivery

6.Analysis based on the payments:

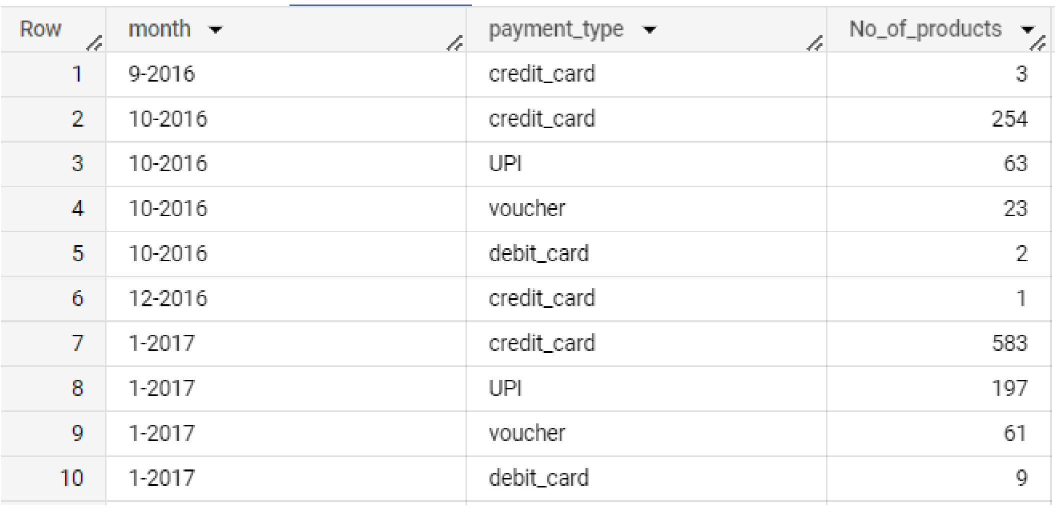
6.1. Find the month-on-month no. of orders placed using different payment types.

Query:

```
select
    concat(x.month,"-",x.year)month,
    x.payment_type,
    x.No_of_products
from(
select
    extract(month from o.order_purchase_timestamp)month,
    extract(year from o.order_purchase_timestamp)year,
    p.payment_type,
    count(o.order_id)No_of_products
from `target.orders` o
join
`target.payments` p
on o.order_id = p.order_id
group by
```

```
    extract(month from o.order_purchase_timestamp),
    extract(year from o.order_purchase_timestamp),
    p.payment_type)x
order by x.year,x.month
```

Result:



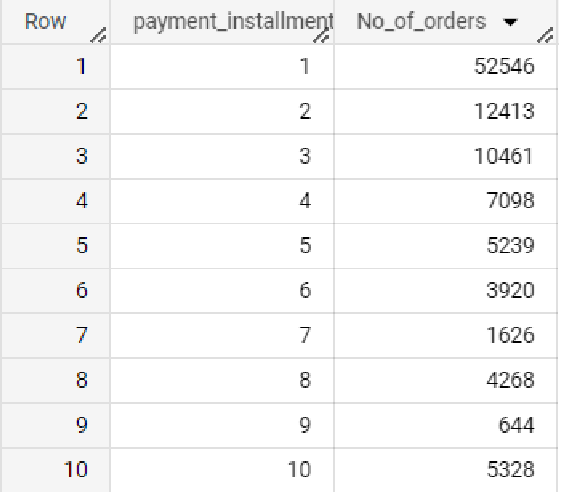
Row	month	payment_type	No_of_products
1	9-2016	credit_card	3
2	10-2016	credit_card	254
3	10-2016	UPI	63
4	10-2016	voucher	23
5	10-2016	debit_card	2
6	12-2016	credit_card	1
7	1-2017	credit_card	583
8	1-2017	UPI	197
9	1-2017	voucher	61
10	1-2017	debit_card	9

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

Query:

```
select p.payment_installments,count(o.order_id)No_of_orders
from `target.orders` o
join
`target.payments` p
on o.order_id = p.order_id
where p.payment_installments <> 0
group by p.payment_installments
order by p.payment_installments
```

Result:



Row	payment_installment	No_of_orders
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:

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

The output indicates varying order counts from month to month. However, upon closer examination, the first three months of 2018 consistently show an increasing order count. Concurrently, year-end order counts appear slightly lower. This trend suggests that a significant number of customers made purchases at Target predominantly during January, February, and March.

Furthermore, if we analyze the order counts between August and September, a noticeable sharp decrease is evident. This could lead to the inference that by efficiently managing their inventory, Target can mitigate potential losses.

Key Insights:

- Product Expansion: Implementing strategies to expand their product range could potentially drive higher profits.

- Year-End Marketing: Concentrating on marketing efforts and offering promotions, especially in the months from September to December, might boost sales. Incorporating clearance sales and innovative ideas during this period could yield improved year-end performance.

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

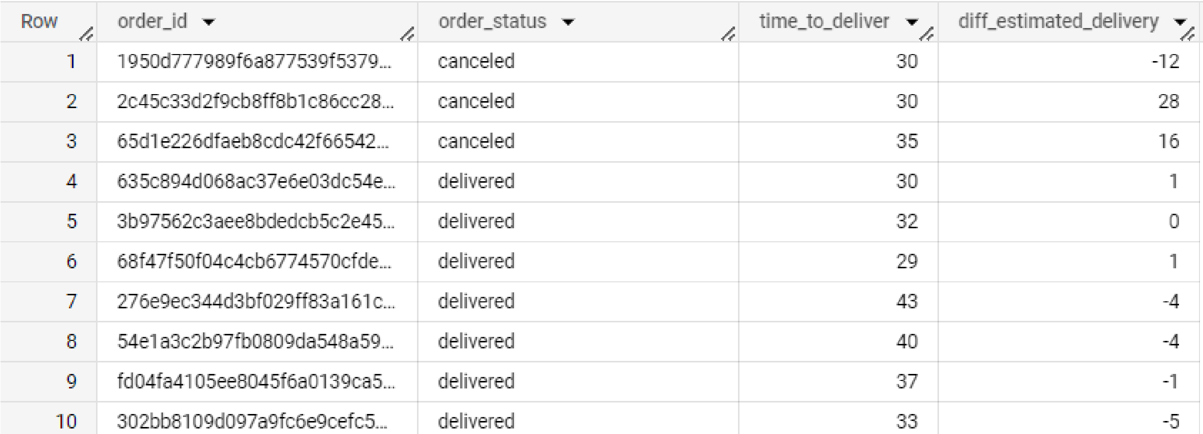
The majority of orders were placed during the afternoon hours, specifically between 13:00 and 18:00. This data suggests that opening the shop during these peak hours could be a strategic decision.

5.1 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.



Row	order_id	order_status	time_to_deliver	diff_estimated_delivery
1	1950d777989f6a877539f5379...	canceled	30	-12
2	2c45c33d2f9cb8ff8b1c86cc28...	canceled	30	28
3	65d1e226dfaeb8cdc42f66542...	canceled	35	16
4	635c894d068ac37e6e03dc54e...	delivered	30	1
5	3b97562c3aee8bdedcb5c2e45...	delivered	32	0
6	68f47f50f04c4cb6774570cfde...	delivered	29	1
7	276e9ec344d3bf029ff83a161c...	delivered	43	-4
8	54e1a3c2b97fb0809da548a59...	delivered	40	-4
9	fd04fa4105ee8045f6a0139ca5...	delivered	37	-1
10	302bb8109d097a9fc6e9cefc5...	delivered	33	-5

In the output, I can observe instances where cancelled orders were still delivered to customers. Based on my understanding, these orders should have been cancelled before the delivery day. Additionally, many orders were delivered after the estimated delivery date. It seems there's a need for improvement in managing cancellations and delivery timelines.

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

Over 40% of the customers used their credit cards to purchase the products. This observation clearly indicates a significant presence of working individuals in Brazil. Additionally, this trend has its own benefits.

From my perspective, the insights garnered from the data show that Target has effectively documented and organized every customer's information in a comprehensive and structured manner. This meticulous record-keeping proves to be a significant advantage, as it enables seamless retrieval of any required details. This organized approach can provide insights into various aspects of customer behavior and preferences.

Key Points:

1. **Comprehensive Customer Records:** The data demonstrates that Target has captured and cataloged details about each customer. This systematic approach makes it easy to access comprehensive information, fostering a deeper understanding of customers' purchasing patterns and tendencies.

2. **Identifying Customer Concentrations**: With well-organized data, it's feasible to pinpoint which states have the highest concentration of customers. This geographical insight can influence targeted marketing campaigns and tailored strategies for specific regions.

3. **High-Value Customer Identification:** The data allows for the identification of customers who have made significant purchases, perhaps even acquiring the costliest products. This information is invaluable for recognizing and rewarding loyal, high-value customers.

4. **Frequent Shoppers Identification:** By leveraging the organized data, it becomes possible to identify customers who consistently make frequent purchases. Recognizing these repeat shoppers can enable personalized offers and incentives to further encourage their loyalty.

-Analysed by

HARIHARAN.M

Batch:DSML June23
Beginner Tue