TARGET: 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 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 analysing 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.

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

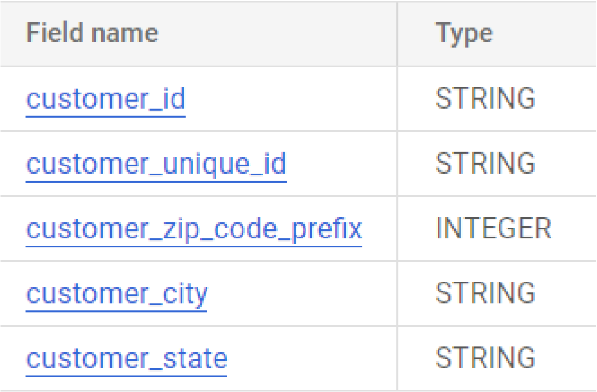
Problem Statement:

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

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.



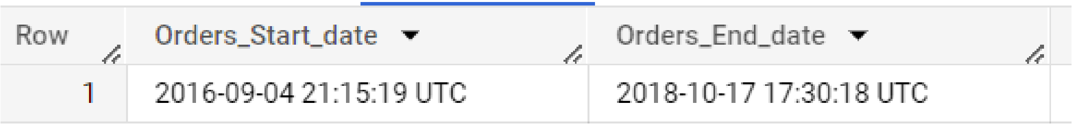
Field name	Type
customer_id	STRING
customer_unique_id	STRING
customer_zip_code_prefix	INTEGER
customer_city	STRING
customer_state	STRING

Insights:
From above table the date types of customers table columns are shown

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

```
select min(order_purchase_timestamp) as Orders_Start_date,
       max(order_purchase_timestamp) as Orders_End_date
from `Target_SQL.orders`
```

output:



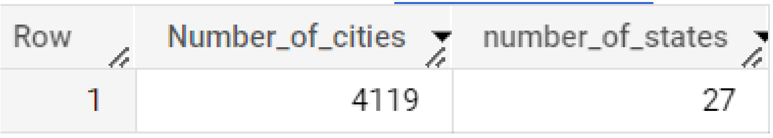
Row	Orders_Start_date	Orders_End_date
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC

Insights:
The Date range between which the orders were placed in between the dates are
Start Date from 4th Sept 2016
End Date from 17th Oct 2017

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

```
select count(Distinct customer_city) as Number_of_cities,
       count(Distinct customer_state) as number_of_states
from `Target_SQL.customers` as c
Join `Target_SQL.orders` as o
on c.customer_id=o.customer_id
where order_purchase_timestamp between
(Select min(order_purchase_timestamp) from `Target_SQL.orders`)
and (Select max(order_purchase_timestamp) from `Target_SQL.orders`);
```

output:



Row	Number_of_cities	number_of_states
1	4119	27

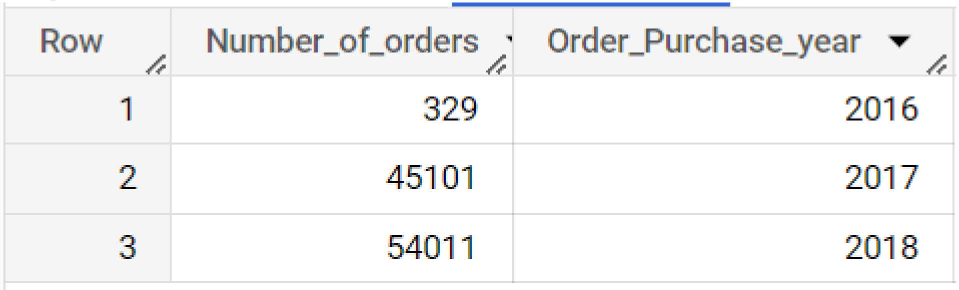
Insights:
Number of cities are 4119 and number of states are 27 of the customers who ordered during the given period.

2. **In-depth Exploration:**

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

```
select count(order_id) as Number_of_orders,
      extract(Year from order_purchase_timestamp) as Order_Purchase_year
from `Target_SQL.orders`
group by Order_Purchase_year
order by Order_Purchase_year;
```

output:



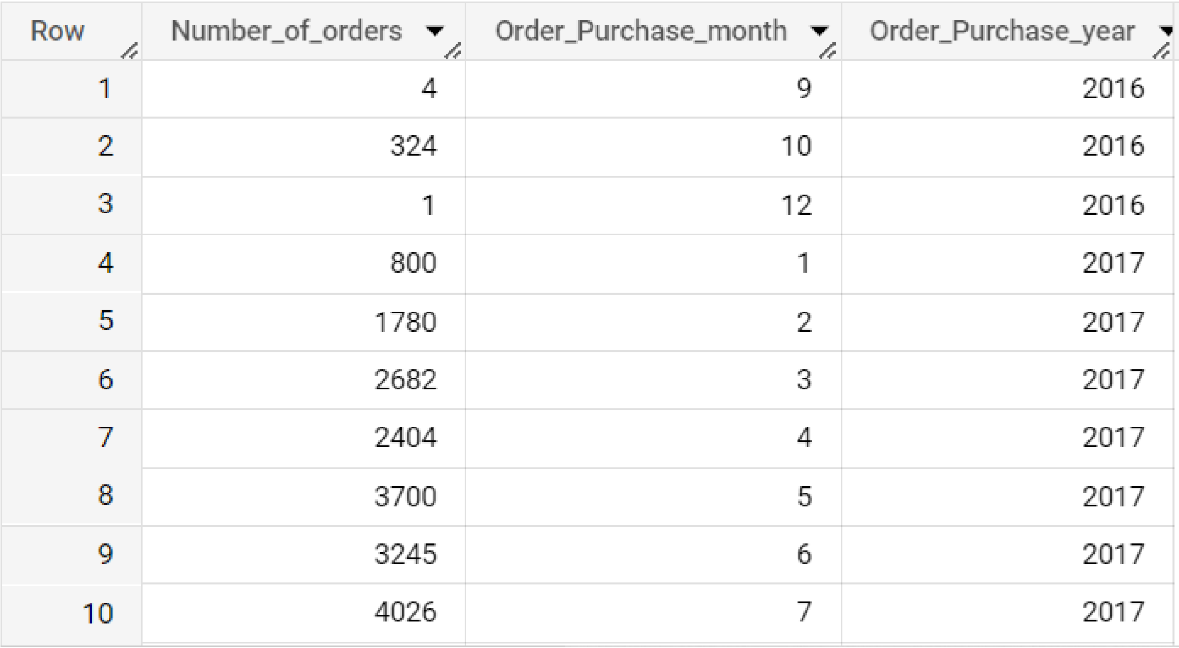
Row	Number_of_orders	Order_Purchase_year
1	329	2016
2	45101	2017
3	54011	2018

Insights:
There is a up trend in the Number_of_orders placed over the past years from 206 to 2018

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

```
select count(order_id) as Number_of_orders,
       extract(month from order_purchase_timestamp) as Order_Purchase_month,
       extract(Year from order_purchase_timestamp) as Order_Purchase_year
from `Target_SQL.orders`
group by Order_Purchase_year, Order_purchase_month
order by Order_Purchase_year, Order_purchase_month;
```

output:



Row	Number_of_orders	Order_Purchase_month	Order_Purchase_year
1	4	9	2016
2	324	10	2016
3	1	12	2016
4	800	1	2017
5	1780	2	2017
6	2682	3	2017
7	2404	4	2017
8	3700	5	2017
9	3245	6	2017
10	4026	7	2017

Insights:
W.r.t each year monthly seasonality in terms of the no. of orders being placed are

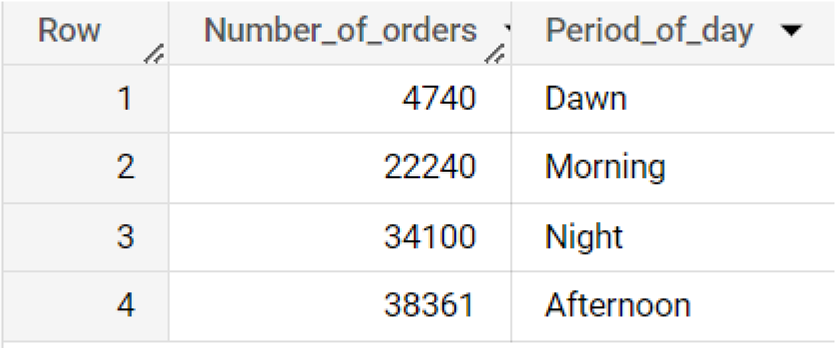
2016- Maximum orders placed in October
2017-Here we can see the uptrend and Maximum orders placed in November
2018-Maximum orders placed in Jan but suddenly after may month the orders started falling

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

2.3.1. 0-6 hrs : Dawn
2.3.2. 7-12 hrs : Mornings
2.3.3. 13-18 hrs : Afternoon
2.3.4. 19-23 hrs : Night

```
select count(order_id) as Number_of_orders,
       Case
       when extract(time from order_purchase_timestamp) between '00:00:00' and
'05:59:59' then 'Dawn'
       when extract(time from order_purchase_timestamp) between '06:00:00' and
'11:59:59' then 'Morning'
       when extract(time from order_purchase_timestamp) between '12:00:00' and
'17:59:59' then 'Afternoon'
       else 'Night'
       End as Period_of_day
  From `Target_SQL.orders`
  Group by Period_of_day
  Order by Number_of_orders;
```

output:



Row	Number_of_orders	Period_of_day
1	4740	Dawn
2	22240	Morning
3	34100	Night
4	38361	Afternoon

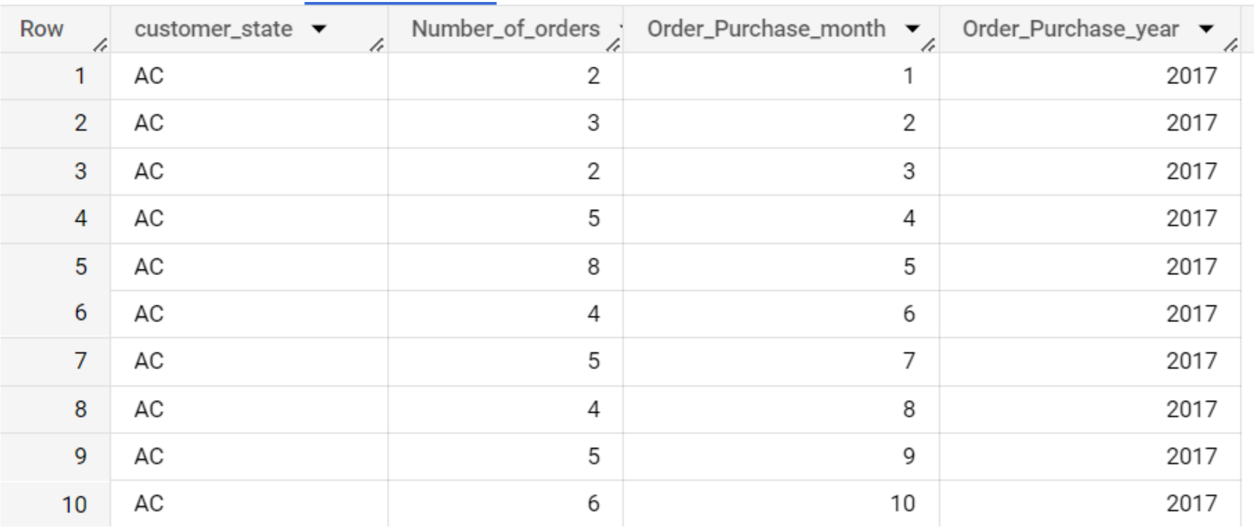
Insights:
Most of the Brazilian customers place orders in Afternoon time

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

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

```
select c.customer_state,count(o.order_id) as Number_of_orders,
      extract(month from o.order_purchase_timestamp) as Order_Purchase_month,
      extract(Year from o.order_purchase_timestamp) as Order_Purchase_year
from `Target_SQL.orders` as o
Join `Target_SQL.customers` as c
on c.customer_id=o.customer_id
group by Order_Purchase_year,Order_Purchase_month, c.customer_state
order by c.customer_state, Order_Purchase_year,Order_Purchase_month;
```

output:



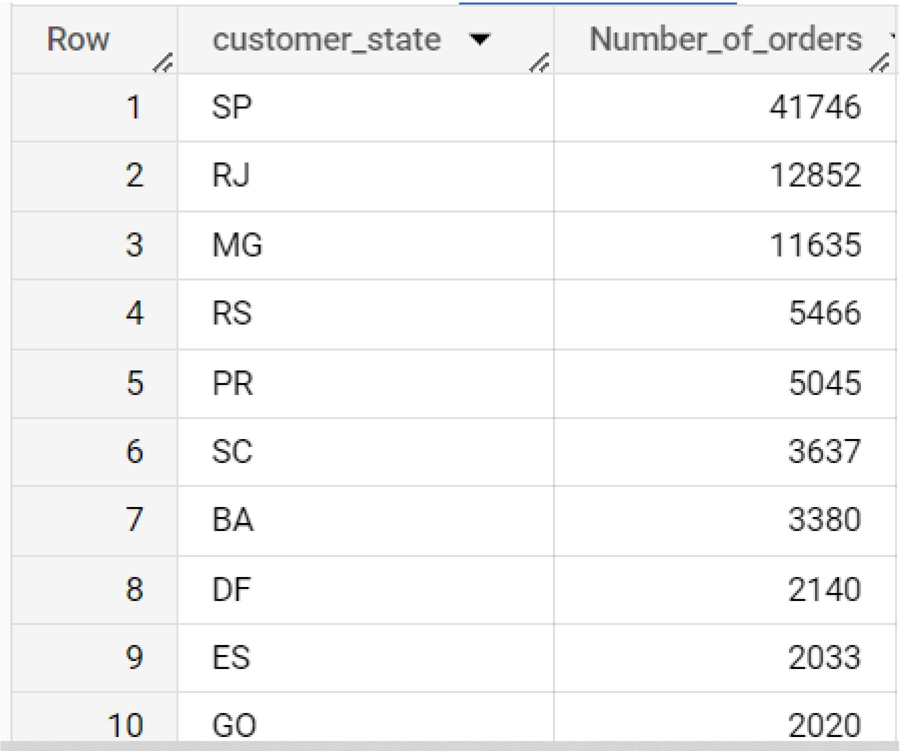
Row	customer_state	Number_of_orders	Order_Purchase_month	Order_Purchase_year
1	AC	2	1	2017
2	AC	3	2	2017
3	AC	2	3	2017
4	AC	5	4	2017
5	AC	8	5	2017
6	AC	4	6	2017
7	AC	5	7	2017
8	AC	4	8	2017
9	AC	5	9	2017
10	AC	6	10	2017

Insights:
From above output we can get information about the trends of orders placed in each state over the years w.r.t the months from this we can know the state wise seasonality from the past orders and we can predict the future order trends.

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

```
select c.customer_state,count(o.order_id) as Number_of_orders,
from `Target_SQL.orders` as o
Join `Target_SQL.customers` as c
on c.customer_id=o.customer_id
group by c.customer_state
order by Number_of_orders DESC;
```

output:



Row	customer_state	Number_of_orders
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

Insights:

Distribution of customers across the Brazil States
Highest number of customers are from SP state
Lowest number of Customers are from RR state

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

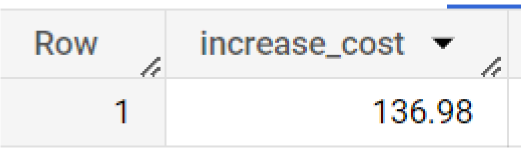
4.1. Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only).
You can use the "payment_value" column in the payments table to get the cost of orders.

```
 select Round((sum(CASE When extract(Year from o.order_purchase_timestamp)=2018
then p.payment_value else 0 end)-

  sum(CASE When extract(Year from o.order_purchase_timestamp)=2017 then
p.payment_value else 0 end))/
  sum(case when extract(Year from o.order_purchase_timestamp)=2017 then
p.payment_value else 0 end)*100,2) as  increase_cost
from `Target_SQL.orders` as o
```

```
Join `Target_SQL.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) between 1 and 8;
```

output:



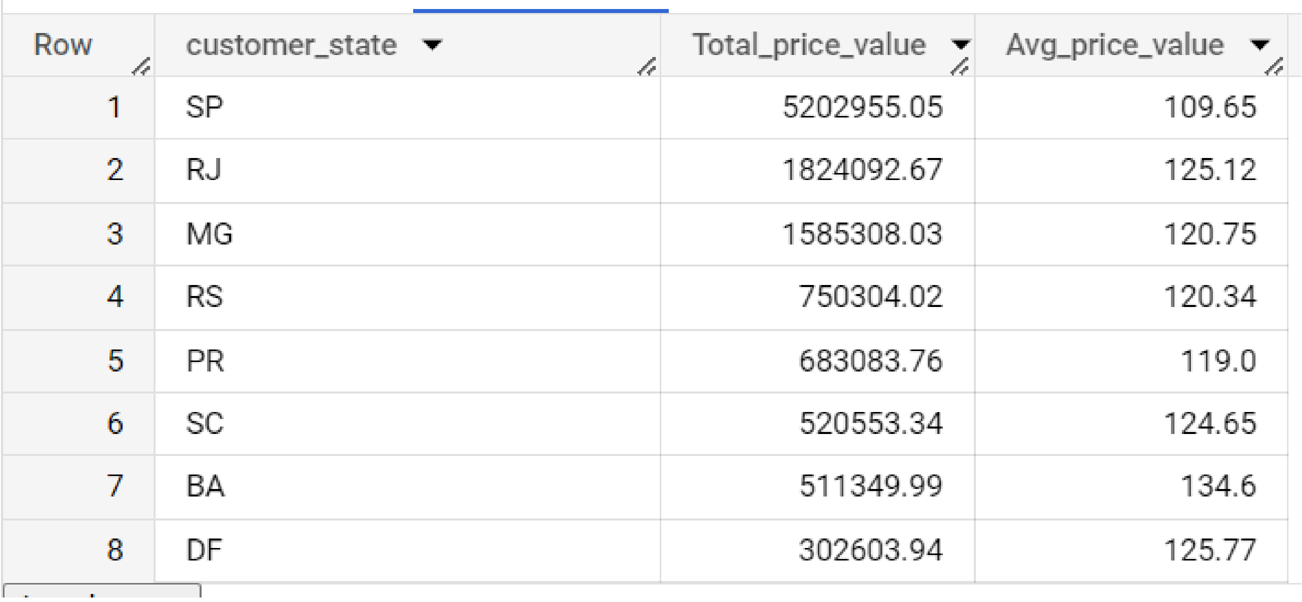
Row	increase_cost
1	136.98

Insights:
The percentage of increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only) is 136.98.

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

```
select distinct c.customer_state,
round(sum(price),2) as Total_price_value,
round(avg(price),2) as Avg_price_value
from `Target_SQL.orders` as o
Join `Target_SQL.customers` as c
on c.customer_id=o.customer_id
Join `Target_SQL.order_items` as io
on o.order_id=io.order_id
Group by c.customer_state
order by Total_price_value DESC, Avg_price_value DESC;
```

output:



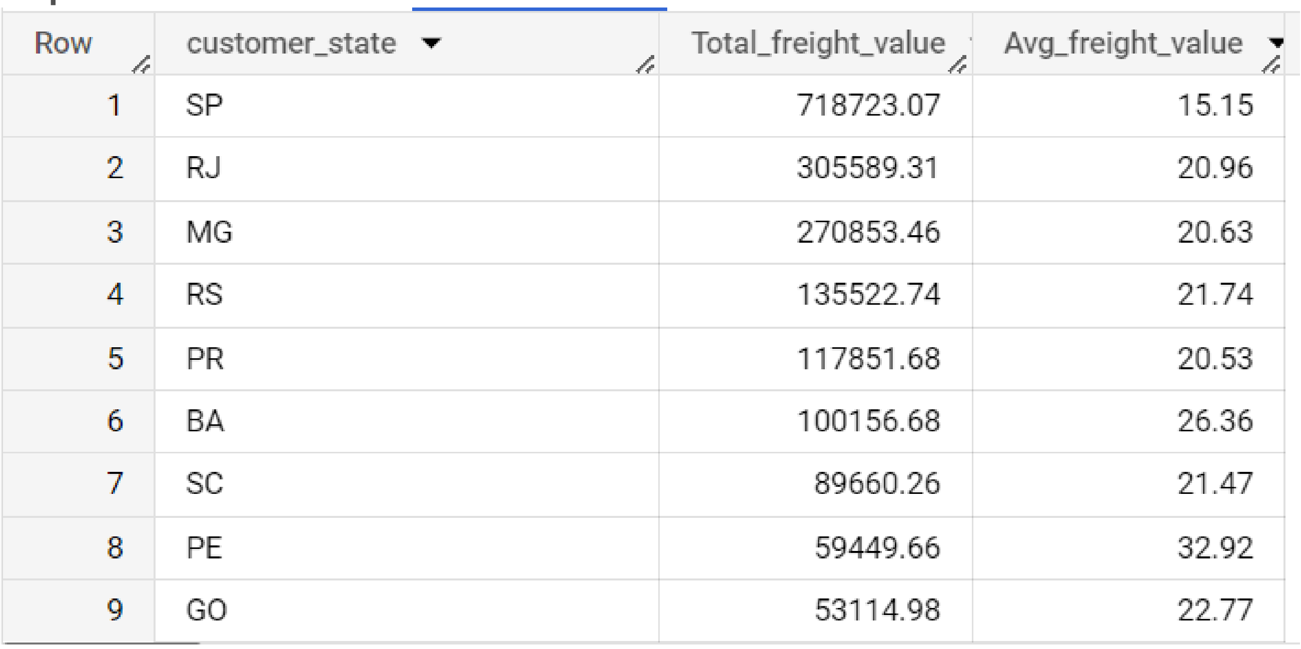
Row	customer_state	Total_price_value	Avg_price_value
1	SP	5202955.05	109.65
2	RJ	1824092.67	125.12
3	MG	1585308.03	120.75
4	RS	750304.02	120.34
5	PR	683083.76	119.0
6	SC	520553.34	124.65
7	BA	511349.99	134.6
8	DF	302603.94	125.77

Insights:
The above output gives the information about Total price and average price value in each state

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

```
select distinct c.customer_state,
round(sum(freight_value),2) as Total_freight_value,
round(avg(freight_value),2) as Avg_freight_value
from `Target_SQL.orders` as o
Join `Target_SQL.customers` as c
on c.customer_id=o.customer_id
Join `Target_SQL.order_items` as io
on o.order_id=io.order_id
Group by c.customer_state
order by Total_freight_value DESC, Avg_freight_value DESC;
```

output:



Row	customer_state	Total_freight_value	Avg_freight_value
1	SP	718723.07	15.15
2	RJ	305589.31	20.96
3	MG	270853.46	20.63
4	RS	135522.74	21.74
5	PR	117851.68	20.53
6	BA	100156.68	26.36
7	SC	89660.26	21.47
8	PE	59449.66	32.92
9	GO	53114.98	22.77

Insights:
The above output gives the information about Total freight value and average freight value in each state.

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.

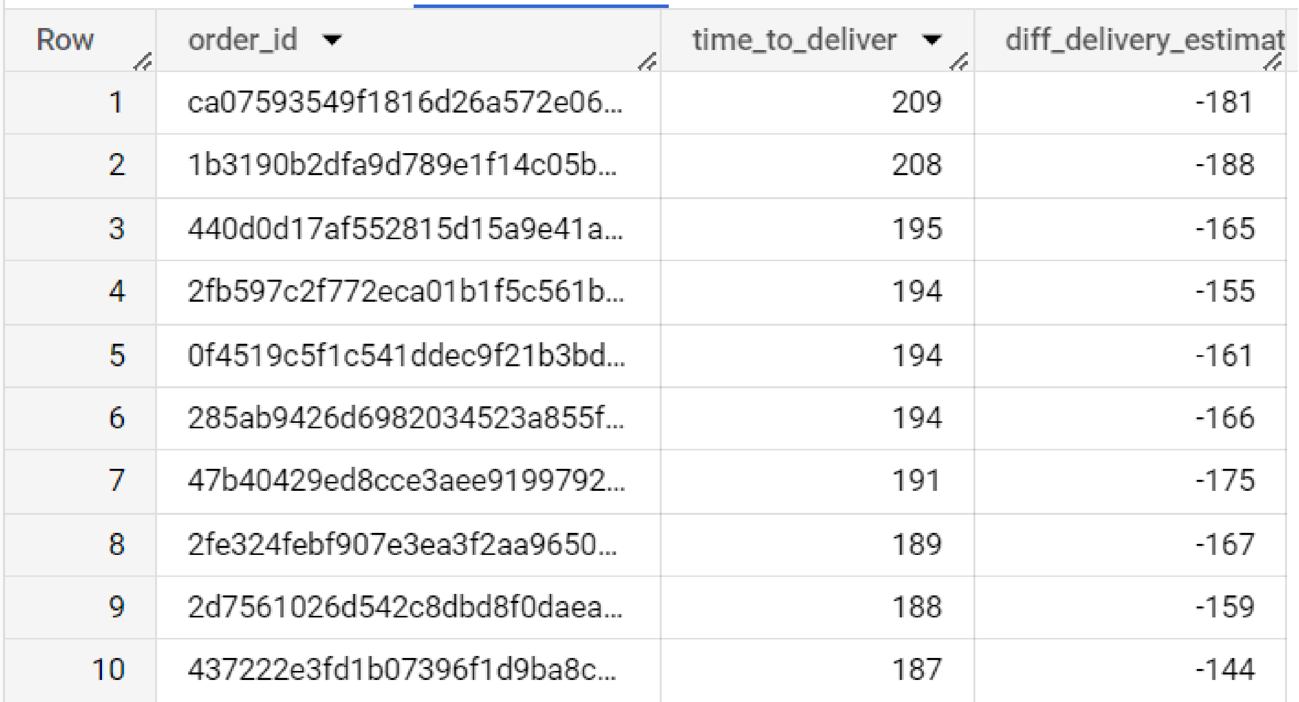
You can calculate the delivery time and the difference between the estimated & actual delivery date using the given formula:

- **time_to_deliver** = order_delivered_customer_date - order_purchase_timestamp

- **diff_estimated_delivery** = order_estimated_delivery_date - order_delivered_customer_date

```
select order_id,
Date_diff(order_delivered_customer_date, order_purchase_timestamp, day) as
time_to_deliver,
Date_diff(order_estimated_delivery_date, order_delivered_customer_date,day) as
diff_delivery_estimated
from `Target_SQL.orders`
where Date_diff(order_delivered_customer_date, order_purchase_timestamp, day) is
not null
order by time_to_deliver DESC, diff_delivery_estimated DESC;
```

output:

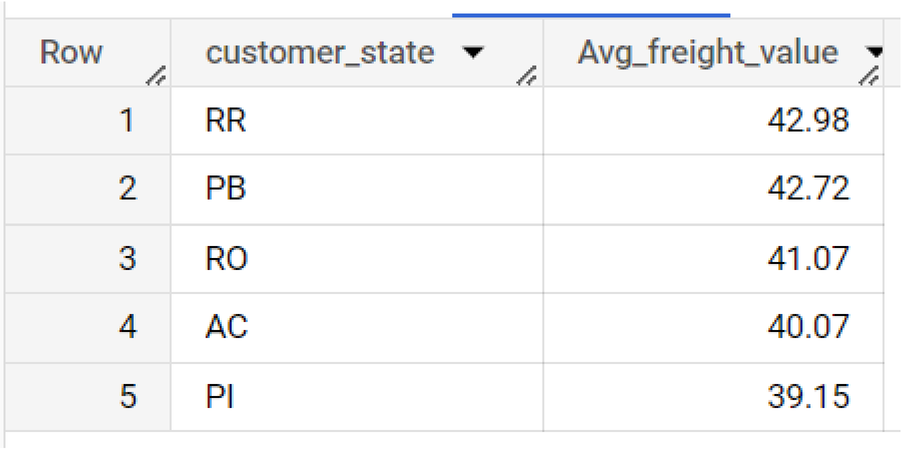


Row	order_id	time_to_deliver	diff_delivery_estimat
1	ca07593549f1816d26a572e06...	209	-181
2	1b3190b2dfa9d789e1f14c05b...	208	-188
3	440d0d17af552815d15a9e41a...	195	-165
4	2fb597c2f772eca01b1f5c561b...	194	-155
5	0f4519c5f1c541ddec9f21b3bd...	194	-161
6	285ab9426d6982034523a855f...	194	-166
7	47b40429ed8cce3aee9199792...	191	-175
8	2fe324febf907e3ea3f2aa9650...	189	-167
9	2d7561026d542c8dbd8f0daea...	188	-159
10	437222e3fd1b07396f1d9ba8c...	187	-144

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

```
select distinct c.customer_state,
round(avg(freight_value),2) as Avg_freight_value
from `Target_SQL.orders` as o
Join `Target_SQL.customers` as c
on c.customer_id=o.customer_id
Join `Target_SQL.order_items` as io
on o.order_id=io.order_id
Group by c.customer_state
order by Avg_freight_value DESC
limit 5;
```

output:



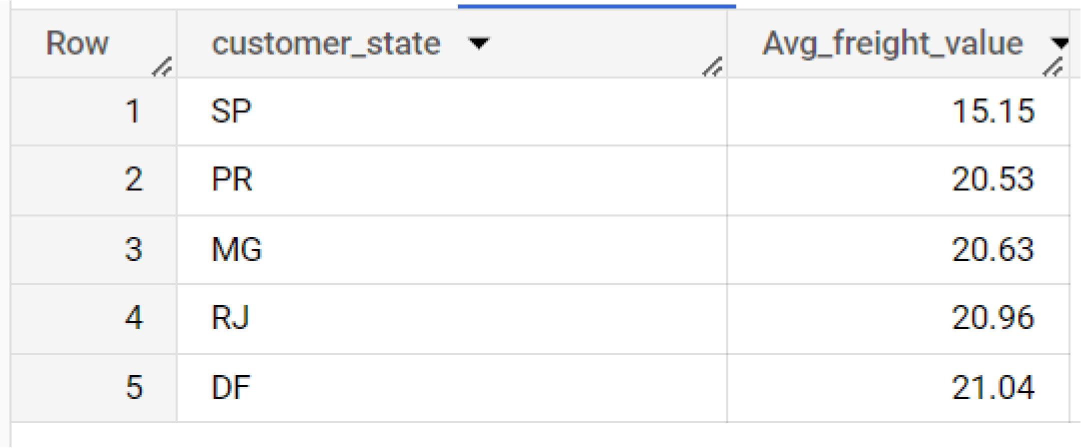
Row	customer_state	Avg_freight_value
1	RR	42.98
2	PB	42.72
3	RO	41.07
4	AC	40.07
5	PI	39.15

Insights:

Top 5 States Highest average freight value

```
select distinct c.customer_state,
round(avg(freight_value),2) as Avg_freight_value
from `Target_SQL.orders` as o
Join `Target_SQL.customers` as c
on c.customer_id=o.customer_id
Join `Target_SQL.order_items` as io
on o.order_id=io.order_id
Group by c.customer_state
order by Avg_freight_value
limit 5;
```

output:



Row	customer_state	Avg_freight_value
1	SP	15.15
2	PR	20.53
3	MG	20.63
4	RJ	20.96
5	DF	21.04

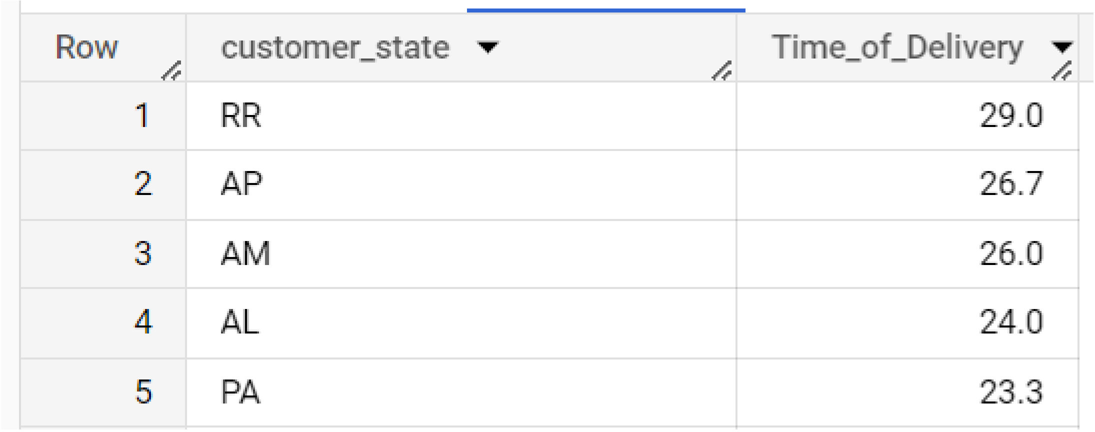
Insights:

Bottom 5 States average freight value

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

```
select c.customer_state,
round(avg(date_diff(order_delivered_customer_date, order_purchase_timestamp,
day)),1) as Time_of_Delivery
from `Target_SQL.orders` as o
Join `Target_SQL.customers` as c
on c.customer_id=o.customer_id
Group by c.customer_state
order by Time_of_Delivery DESC
limit 5;
```

output:



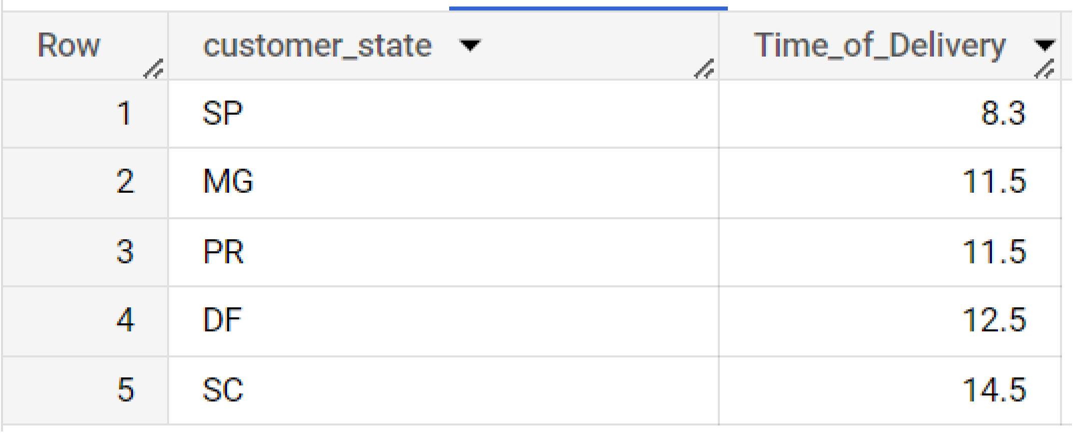
Row	customer_state	Time_of_Delivery
1	RR	29.0
2	AP	26.7
3	AM	26.0
4	AL	24.0
5	PA	23.3

Insights:

Top 5 states highest average delivery time

```
select c.customer_state,
round(avg(date_diff(order_delivered_customer_date, order_purchase_timestamp,
day)),1) as Time_of_Delivery
from `Target_SQL.orders` as o
Join `Target_SQL.customers` as c
on c.customer_id=o.customer_id
Group by c.customer_state
order by Time_of_Delivery
limit 5;
```

output:



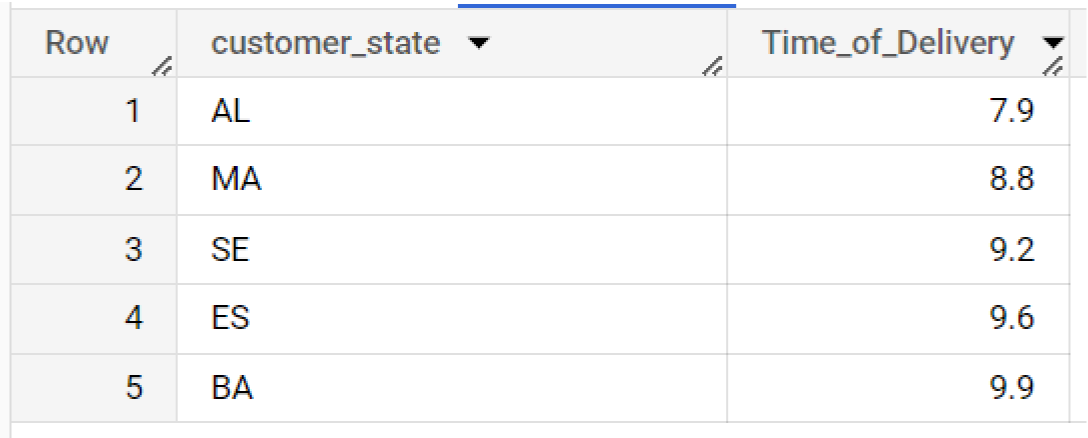
Row	customer_state	Time_of_Delivery
1	SP	8.3
2	MG	11.5
3	PR	11.5
4	DF	12.5
5	SC	14.5

Insights:
Top 5 states 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.
You can use the difference between the averages of actual & estimated delivery date to figure out how fast the delivery was for each state.

```
select c.customer_state,
round(avg(date_diff(order_estimated_delivery_date,
order_delivered_customer_date,day)),1) as Time_of_Delivery
from `Target_SQL.orders` as o
Join `Target_SQL.customers` as c
on c.customer_id=o.customer_id
Group by c.customer_state
order by Time_of_Delivery
limit 5;
```

output:



Row	customer_state	Time_of_Delivery
1	AL	7.9
2	MA	8.8
3	SE	9.2
4	ES	9.6
5	BA	9.9

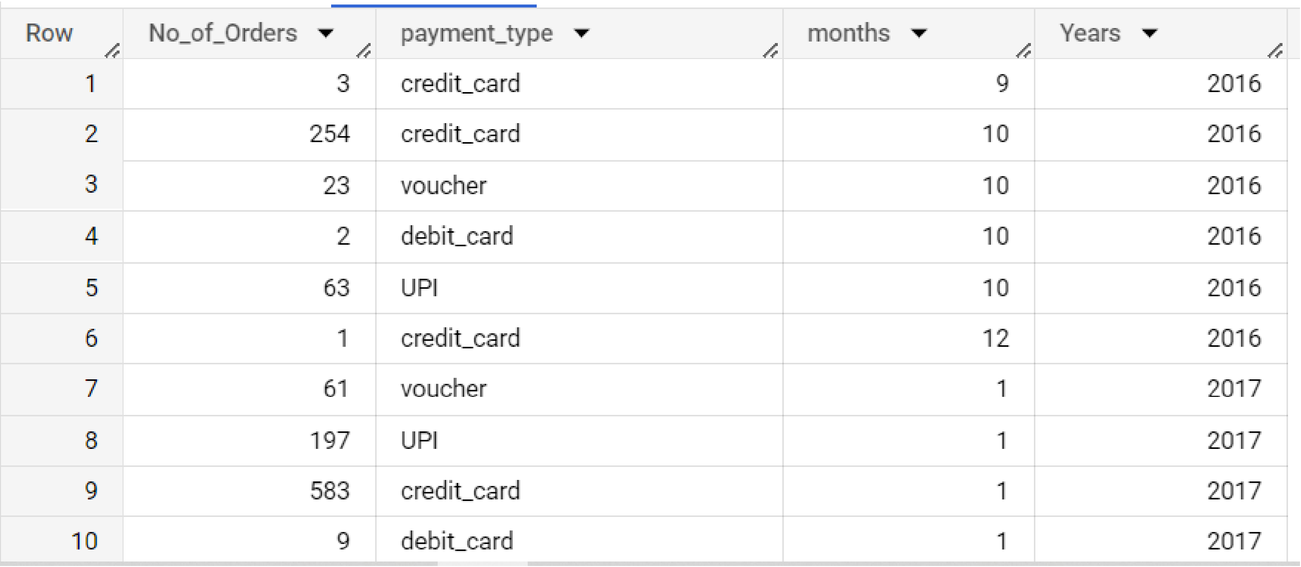
Insights:
Top 5 states where the order delivery is really fast as compared to the estimated date of delivery

6. **Analysis based on the payments:**

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

```
select count(o.order_id) as No_of_Orders,p.payment_type,
extract(month from order_purchase_timestamp) as months,
extract(Year from order_purchase_timestamp) as Years
from `Target_SQL.orders` as o
Join `Target_SQL.payments` as p
on o.order_id=p.order_id
group by months, years, p.payment_type
order by Years, months;
```

output:

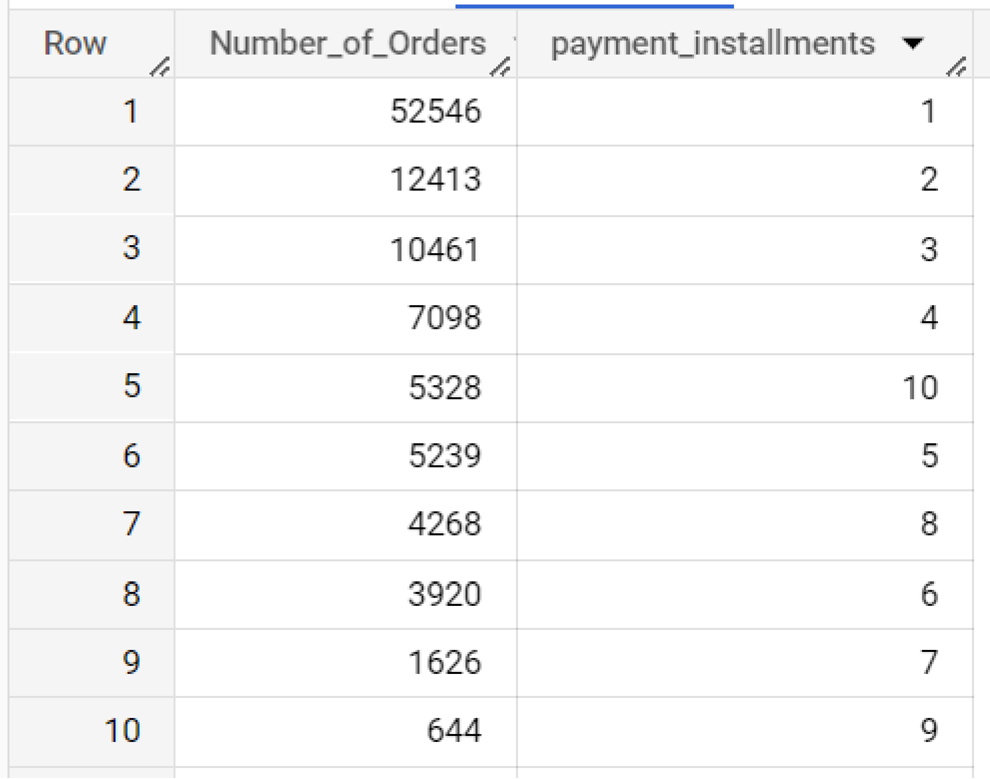


Row	No_of_Orders	payment_type	months	Years
1	3	credit_card	9	2016
2	254	credit_card	10	2016
3	23	voucher	10	2016
4	2	debit_card	10	2016
5	63	UPI	10	2016
6	1	credit_card	12	2016
7	61	voucher	1	2017
8	197	UPI	1	2017
9	583	credit_card	1	2017
10	9	debit_card	1	2017

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

```
select count(order_id) as Number_of_Orders, payment_installments
from `Target_SQL.payments`
group by payment_installments
order by Number_of_Orders DESC, payment_installments DESC;
```

output:



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

Insights:
Above output shows the no. of orders placed on the basis of the payment instalments that have been paid