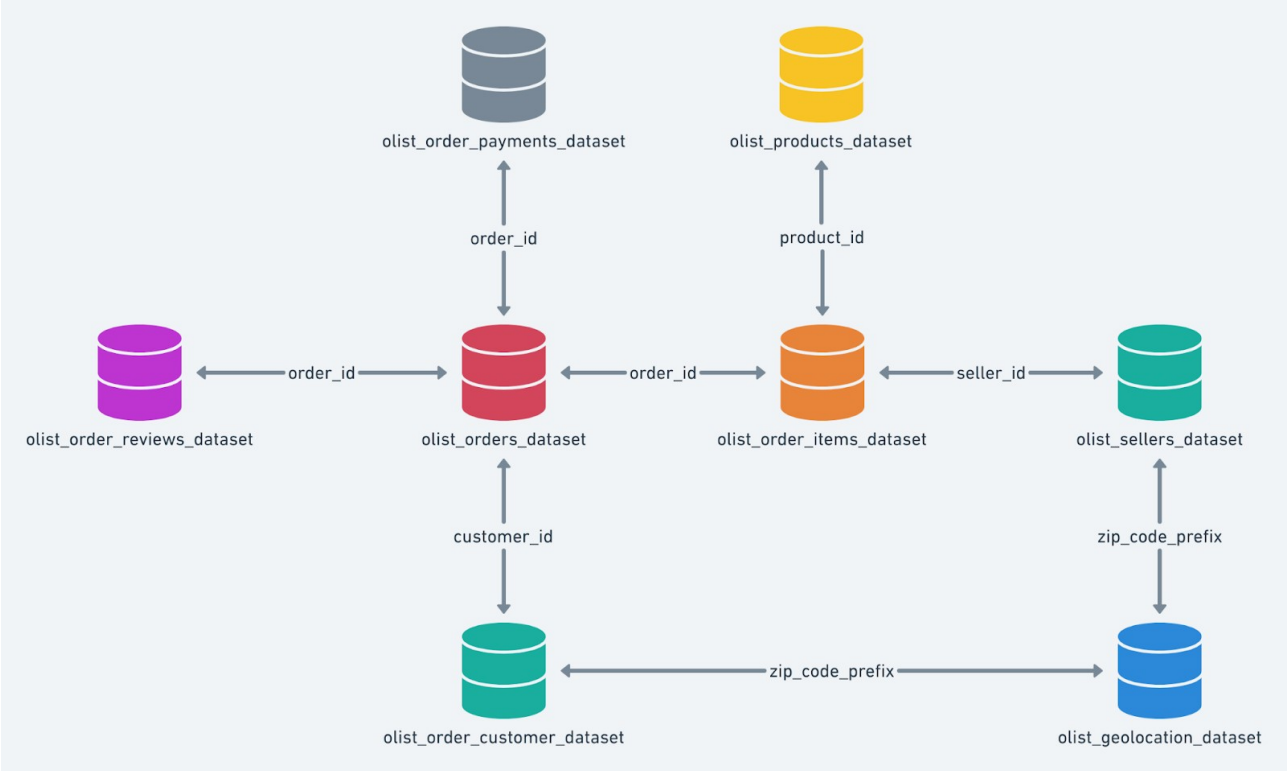
Context:

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.



Problem Statement:

Analyze 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.Data type of all columns in the "customers" table.
2.Get the time range between which the orders were placed.
3.Count the Cities & States of customers who ordered during the given period.

2.**In-depth Exploration:**

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

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

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

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

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

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

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.

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

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

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:

- **time_to_deliver** = order_delivered_customer_date - order_purchase_timestamp
- **diff_estimated_delivery** = order_estimated_delivery_date - order_delivered_customer_date

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

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

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

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

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

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

1. Data type of columns in a table

```
SELECT column_name,DATA_TYPE
FROM `target-sql-382405`.Target_Dataset.INFORMATION_SCHEMA.COLUMNS
WHERE
TABLE_NAME = 'customers'
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

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

```
SELECT column_name,DATA_TYPE
FROM `target-sql-382405`.Target_Dataset.INFORMATION_SCHEMA.COLUMNS
WHERE
TABLE_NAME = 'geolocation'
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	column_name	DATA_TYPE
1	geolocation_zip_code_prefix	INT64
2	geolocation_lat	FLOAT64
3	geolocation_lng	FLOAT64
4	geolocation_city	STRING
5	geolocation_state	STRING

```
SELECT column_name,DATA_TYPE
FROM `target-sql-382405`.Target_Dataset.INFORMATION_SCHEMA.COLUMNS
WHERE
TABLE_NAME = 'order_items'
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	column_name	DATA_TYPE
1	order_id	STRING
2	order_item_id	INT64
3	product_id	STRING
4	seller_id	STRING
5	shipping_limit_date	TIMESTAMP
6	price	FLOAT64
7	freight_value	FLOAT64

```
SELECT column_name,DATA_TYPE
FROM `target-sql-382405`.Target_Dataset.INFORMATION_SCHEMA.COLUMNS
WHERE
TABLE_NAME = 'order_reviews'
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	column_name	DATA_TYPE
1	review_id	STRING
2	order_id	STRING
3	review_score	INT64
4	review_comment_title	STRING
5	review_creation_date	TIMESTAMP
6	review_answer_timestamp	TIMESTAMP

```
SELECT column_name,DATA_TYPE
FROM `target-sql-382405`.Target_Dataset.INFORMATION_SCHEMA.COLUMNS
WHERE
TABLE_NAME = 'orders'
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	column_name	DATA_TYPE
1	order_id	STRING
2	customer_id	STRING
3	order_status	STRING
4	order_purchase_timestamp	TIMESTAMP
5	order_approved_at	TIMESTAMP
6	order_delivered_carrier_date	TIMESTAMP
7	order_delivered_customer_date	TIMESTAMP
8	order_estimated_delivery_date	TIMESTAMP

```
SELECT column_name,DATA_TYPE
FROM `target-sql-382405`.Target_Dataset.INFORMATION_SCHEMA.COLUMNS
WHERE
TABLE_NAME = 'payments'
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	column_name	DATA_TYPE
1	order_id	STRING
2	payment_sequential	INT64
3	payment_type	STRING
4	payment_installments	INT64
5	payment_value	FLOAT64

```
SELECT column_name,DATA_TYPE
FROM `target-sql-382405`.Target_Dataset.INFORMATION_SCHEMA.COLUMNS
WHERE
TABLE_NAME = 'products'
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	column_name	DATA_TYPE
1	order_id	STRING
2	payment_sequential	INT64
3	payment_type	STRING
4	payment_installments	INT64
5	payment_value	FLOAT64

```
SELECT column_name,DATA_TYPE
FROM `target-sql-382405`.Target_Dataset.INFORMATION_SCHEMA.COLUMNS
WHERE
TABLE_NAME = 'sellers'
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	column_name	DATA_TYPE
1	seller_id	STRING
2	seller_zip_code_prefix	INT64
3	seller_city	STRING
4	seller_state	STRING

2. Time period for which the data is given:

```
SELECT min(order_purchase_timestamp) as start_time, max(order_purchase_timestamp) as
end_time
FROM `target-sql-382405.Target_Dataset.orders`
```

Here I have considered the time period of order_purchase_timestamp since there is nothing mentioned specifically in the question.

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

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

3. City and states of customers ordered during the given time period:

```
SELECT count(c.customer_id) as customer_count,customer_city,customer_state
FROM `target-sql-382405.Target_Dataset.customers` as c
join `Target_Dataset.orders` as o on c.customer_id=o.customer_id
where order_purchase_timestamp between (SELECT min(order_purchase_timestamp) FROM
`target-sql-382405.Target_Dataset.orders` ) and (select max(order_purchase_timestamp) FROM
`target-sql-382405.Target_Dataset.orders` )
group by customer_state,customer_city
```

Assumptions:

Here the time period is considered based on the order_purchase_timestamp.

Query results

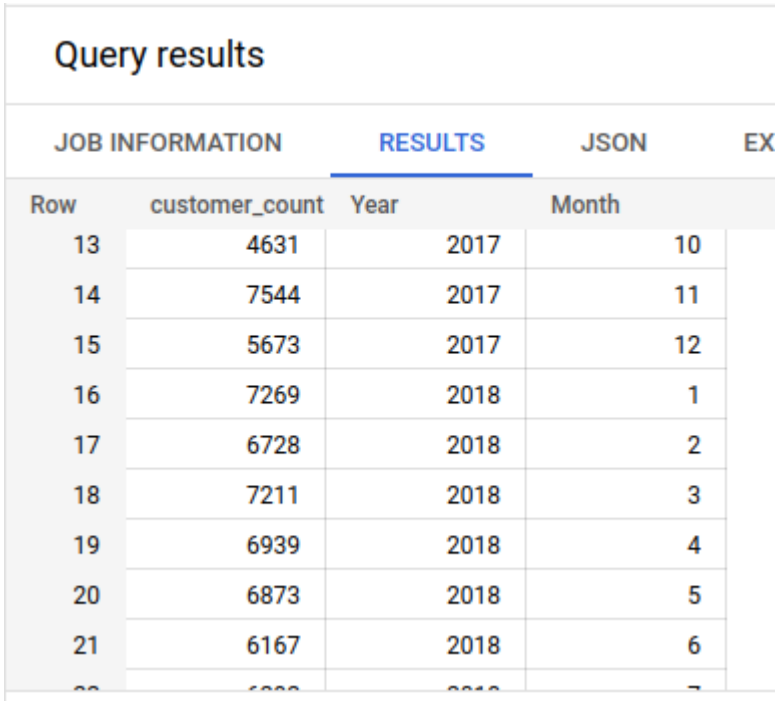
JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXE

Row	customer_count	customer_city	customer_state
1	3	acu	RN
2	8	ico	CE
3	2	ipe	RS
4	4	ipu	CE
5	3	ita	SC
6	136	itu	SP
7	74	jau	SP
8	2	luz	MG
9	85	poa	SP
10	53	uba	MG
11	5	una	BA

2. In-depth Exploration:

1. **Is there a growing trend on e-commerce in Brazil? How can we describe a complete scenario? Can we see some seasonality with peaks at specific months?**

```
SELECT count(customer_id) as customer_count,
extract(year FROM order_purchase_timestamp) as Year,
extract(month FROM order_purchase_timestamp) as Month
FROM `target-sql-382405.Target_Dataset.orders`
group by Year,Month
order by Year,Month
```



Query results

JOB INFORMATION | RESULTS | JSON | EX

Row	customer_count	Year	Month
13	4631	2017	10
14	7544	2017	11
15	5673	2017	12
16	7269	2018	1
17	6728	2018	2
18	7211	2018	3
19	6939	2018	4
20	6873	2018	5
21	6167	2018	6

From the year 2016 month of september till 2018 month of october we have the data. Among that we could see the count of customer increase month over month. From 11th month of 2017 to 3rd month of 2018 there is some increase in between for the months november 2017,January 2018,march 2018. So the seasonality could fall under these months. Mostly late fall and winter season.

2. What time do Brazilian customers tend to buy (Dawn, Morning, Afternoon or Night)?

```
SELECT count(customer_id) as customer_count,
case
when extract(hour FROM order_purchase_timestamp) between 3 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'
else 'Night'
end as Purchase_time
from `Target_Dataset.orders` group by Purchase_time order by Purchase_time
```

Assumptions:

I assume that the order_purchase_timestamp is in brazilian timestamp and based on that calculated the time customers tend to buy.

Orders are counted irrespective of delivered or undelivered, cancelled or paid partially or completely

Query results

JOB INFORMATION | RESULTS | JSON

Row	customer_count	Purchase_time
1	38135	Afternoon
2	1168	Dawn
3	27733	Morning
4	32405	Night

From the above result, it is observed that there is more customer traffic in the afternoon, that is between 13 hrs and 18 hrs. 'Night' also is have second less customer traffic.

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

1. Get month on month orders by states

```
SELECT *,
ROUND((z.order_id_count - (LAG(z.order_id_count,1) OVER(PARTITION BY z.customer_state
ORDER BY z.customer_state, Year, Month)))*100/LAG(z.order_id_count,1) OVER(PARTITION
BY z.customer_state ORDER BY z.customer_state, Year, Month),2) AS m_o_m_per100
FROM
(SELECT DISTINCT c.customer_state,COUNT(o.order_id) AS order_id_count,
EXTRACT(year from order_purchase_timestamp) AS Year, EXTRACT(month FROM
order_purchase_timestamp) AS Month
FROM `Target_Dataset.orders` AS o
JOIN `Target_Dataset.customers` AS c
USING(customer_id)
GROUP BY c.customer_state, Year, Month
ORDER BY c.customer_state, Year, Month) AS z
ORDER BY z.customer_state, Year, Month
```

Not exactly, there is a growing trend in orders as it initially (2016) and then later (2018) we see a sharp dip in orders. Initially we see 2017 March. We get to see a peak in 2017 March. Also, we can find peaks during 2017 Nov and 2018 Jan, March. With given data we find peak in March repeated, indicating seasonality in the region.

Query results

SAVE RESUL

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	customer_state	order_id_count	Year	Month	m_o_m_per100
1	AC	2	2017	1	null
2	AC	3	2017	2	50.0
3	AC	2	2017	3	-33.33
4	AC	5	2017	4	150.0
5	AC	8	2017	5	60.0
6	AC	4	2017	6	-50.0
7	AC	5	2017	7	25.0
8	AC	4	2017	8	-20.0
9	AC	5	2017	9	25.0
10	AC	6	2017	10	20.0
11	AC	5	2017	11	-16.67

2. Distribution of customers across the states in Brazil

```
SELECT v.customer_state,concat(ROUND(v.c_count*100/SUM(v.c_count) OVER(),2),'%') AS percentage_distribution
FROM(
SELECT DISTINCT COUNT(customer_id) AS c_count, customer_state
FROM `Target_Dataset.customers`
GROUP BY customer_state) AS v
ORDER BY percentage_distribution DESC
```

Assumptions and insights:
From the result obtained, it is found that the maximum customer distribution is from the state named “**SP**” .

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DET.

Row	customer_state	percentage_distribution
1	RS	5.5%
2	PR	5.07%
3	SP	41.98%
4	SC	3.66%
5	BA	3.4%
6	DF	2.15%
7	ES	2.04%
8	GO	2.03%
9	RJ	12.92%
10	MG	11.7%
11	PE	1.66%

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

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

Assumptions and insights:

* But when we do see rise in payment value between 2016 to 2017 only considering Jan to Aug month on month even with oscillating growth percentages

```
SELECT *,
CONCAT(ROUND((x.tot_pay_val - (LAG(x.tot_pay_val,1) OVER(ORDER BY year, month)))*100/LAG(x.tot_pay_val,1) OVER(ORDER BY year, month),2),"%") AS m_o_m_per100
FROM
(SELECT EXTRACT(YEAR FROM DATE (order_purchase_timestamp)) AS year,
EXTRACT(MONTH FROM DATE (order_purchase_timestamp)) AS month,
ROUND(SUM(p.payment_value),2) AS tot_pay_val,
FROM `Target_Dataset.orders` AS o
JOIN `Target_Dataset.payments` AS p
ON o.order_id = p.order_id
GROUP BY year,month
ORDER BY year,month) AS x
WHERE x.year BETWEEN 2017 AND 2018 AND x.month BETWEEN 1 AND 8
ORDER BY year, month
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXE

Row	year	month	tot_pay_val	m_o_m_per100
1	2017	1	138488.04	*null*
2	2017	2	291908.01	110.78%
3	2017	3	449863.6	54.11%
4	2017	4	417788.03	-7.13%
5	2017	5	592918.82	41.92%
6	2017	6	511276.38	-13.77%
7	2017	7	592382.92	15.86%
8	2017	8	674396.32	13.84%
9	2018	1	1115004.18	65.33%
10	2018	2	992463.34	-10.99%
11	2018	3	1159652.12	16.85%
12	2018	4	1160785.48	0.1%
13	2018	5	1153982.15	-0.59%

2. Mean & Sum of price and freight value by customer state

```
select
customer_state,
sum(price) as Sum_price_per_state,
sum(freight_value) as Sum_freight_per_state,
avg(price) as Mean_price_per_state,
avg(freight_value) as Mean_freight_per_state
from `Target_Dataset.order_items`,`Target_Dataset.customers`
group by customer_state
```

Query results

SAVE RE

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH | PREVIEW

Row	customer_state	Sum_price_per_state	Sum_freight_per_state	Mean_price_per_	Mean_freight_pe
1	SP	567396757896.37	94008215657.29	120.653739...	19.9903199...
2	MG	158138774450.8	26200967497.88	120.653739...	19.9903199...
3	RS	74291924463.78	12308937545.64	120.653739...	19.9903199...
4	BA	45939755705.81	7611454245.2	120.653739...	19.9903199...
5	RJ	174679804833.93	28941541408.07	120.653739...	19.9903199...
6	SC	49432808136.66	8190194996.98	120.653739...	19.9903199...
7	ES	27631811642.12	4578132094.82	120.653739...	19.9903199...
8	PB	7285121023.2	1207023513.44	120.653739...	19.9903199...
9	PA	13251852607.52	2195611801.5	120.653739...	19.9903199...
10	GO	27455120274.02	4548857270.8	120.653739...	19.9903199...
11	PR	68569842466.07	11360883629.3	120.653739...	19.9903199...
12	MT	12327620835.91	2042481952.78	120.653739...	19.9903199...

5. Analysis on sales, freight and delivery time

1. **Calculate days between purchasing, delivering and estimated delivery**

```
select
date_diff(order_purchase_timestamp, order_delivered_customer_date, day) as
purchase_delivery_timeperiod,
date_diff(order_estimated_delivery_date,order_purchase_timestamp, day) as
purchase_estimated_delivery,
date_diff(order_delivered_customer_date,order_estimated_delivery_date, day) as
delivery_estimated_delivery
from `Target_Dataset.orders`
order by purchase_delivery_timeperiod
```

Query results

JOB INFORMATION | RESULTS | JSON

Row	purchase_delive	purchase_estima	delivery_estimat
1	null	16	null
2	null	33	null
3	null	36	null
4	null	25	null
5	null	24	null
6	null	27	null
7	null	31	null
8	null	33	null
9	null	15	null
10	null	31	null
11	null	25	null

2. Find time_to_delivery & diff_estimated_delivery. Formula for the same given below:

time_to_delivery = order_purchase_timestamp-order_delivered_customer_date

diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date

```
select
date_diff(order_purchase_timestamp, order_delivered_customer_date, day) as purchase_delivery_timeperiod,
date_diff(order_estimated_delivery_date,order_delivered_customer_date, day) as delivery_estimated_deliveryy
from `Target_Dataset.orders`
```

Query results

JOB INFORMATION | RESULTS

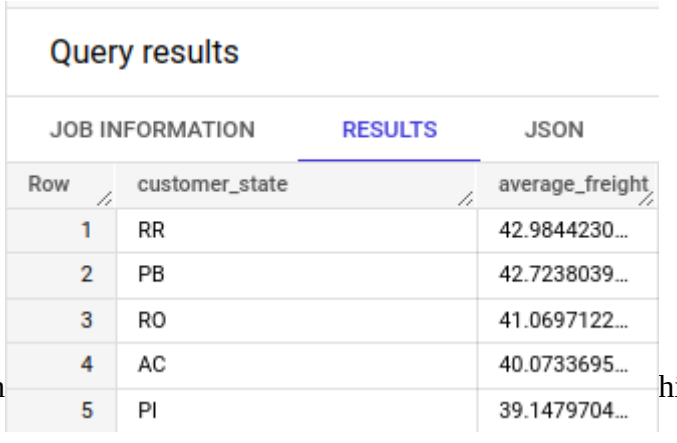
Row	purchase_delive	delivery_estimat
1	-30	-12
2	-30	28
3	-35	16
4	-30	1
5	-32	0
6	-29	1
7	-43	-4
8	-40	-4
9	-37	-1
10	-33	-5
11	-38	-6
12	-36	-2
13	-34	0

3. Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery

4. Sort the data to get the following:

Top 5 states with highest average freight value - sort in desc/asc limit 5

```
select
customer_state,
avg(ot.freight_value) average_freight
from `Target_Dataset.order_items` as ot
join `Target_Dataset.orders` as o on ot.order_id=o.order_id
join `Target_Dataset.customers` as c on o.customer_id=c.customer_id
group by c.customer_state
order by average_freight desc
```



Top 5 states with highest average time to delivery

```
select
customer_state,
avg(DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp,day)) as
average_delivery
from `Target_Dataset.order_items` as ot
join `Target_Dataset.orders` as o on ot.order_id=o.order_id
join `Target_Dataset.customers` as c on o.customer_id=c.customer_id
group by c.customer_state
order by average_delivery desc limit 5
```

Query results

JOB INFORMATION | RESULTS | JSON

Row	customer_state	average_delivery
1	RR	27.8260869...
2	AP	27.7530864...
3	AM	25.9631901...
4	AL	23.9929742...
5	PA	23.3017077...

Top 5 states where delivery is really fast/ not so fast compared to estimated date

```
select *,
(average_estimated_delivery-average_delivery) as delivery_difference
from
(
select
customer_state,
avg(DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp,day)) as
average_delivery,
avg(DATE_DIFF(order_estimated_delivery_date,order_purchase_timestamp,day)) as
average_estimated_delivery
from `Target_Dataset.order_items` as ot
join `Target_Dataset.orders` as o on ot.order_id=o.order_id
join `Target_Dataset.customers` as c on o.customer_id=c.customer_id
group by c.customer_state
)
order by delivery_difference desc limit 5
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXE

Row	customer_state	average_delivery	average_estimat	delivery_differen
1	AC	20.3296703...	40.6956521...	20.3659818...
2	RO	19.2820512...	38.6510791...	19.3690278...
3	AM	25.9631901...	45.2060606...	19.2428704...
4	RR	27.8260869...	45.9807692...	18.1546822...
5	AP	27.7530864...	45.4878048...	17.7347184...

6. Payment type analysis:

1. Month over Month count of orders for different payment types

```
select *,
round(((x.order_count-lag(x.order_count) over(partition by payment_type order by Year,Month))*100)/lag(x.order_count) over(partition by payment_type order by Year,Month),2) as Month_on_Month_orders
from
(
select
count(o.order_id) as order_count,
payment_type,
extract(month from order_purchase_timestamp) as Month,
extract(year from order_purchase_timestamp) as Year
from `Target_Dataset.orders` as o join `Target_Dataset.payments` as p
on o.order_id=p.order_id
group by payment_type,Year,Month
order by payment_type,Year,Month
) as x
```

Assumptions and Insights:

- We see customers prefer credit card payments more than UPI and debit card.

- For those customers who pay by installments, we see high number for less than 10 installments and few for more than 10 installments

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	order_count	payment_type	Month	Year	Month_on_Mont
17	1287	UPI	4	2018	-4.81
18	1263	UPI	5	2018	-1.86
19	1100	UPI	6	2018	-12.91
20	1229	UPI	7	2018	11.73
21	1139	UPI	8	2018	-7.32
22	3	credit_card	9	2016	*null*
23	254	credit_card	10	2016	8366.67
24	1	credit_card	12	2016	-99.61
25	583	credit_card	1	2017	58200.0
26	1356	credit_card	2	2017	132.59
27	2016	credit_card	3	2017	48.67
28	1846	credit_card	4	2017	-8.43
29	2853	credit_card	5	2017	54.55

2. **Count of orders based on the no. of payment installments**

```
SELECT p.payment_installments, COUNT(p.order_id) AS orders_count
FROM `SQL_Target_Project.payments` AS p
JOIN `SQL_Target_Project.orders` AS o
USING(order_id)
GROUP BY p.payment_installments;
```

Query results

JOB INFORMATION | RESULTS

Row	payment_installr	orders_count
1	0	2
2	1	52546
3	2	12413
4	3	10461
5	4	7098
6	5	5239
7	6	3920
8	7	1626
9	8	4268
10	9	644
11	10	5328