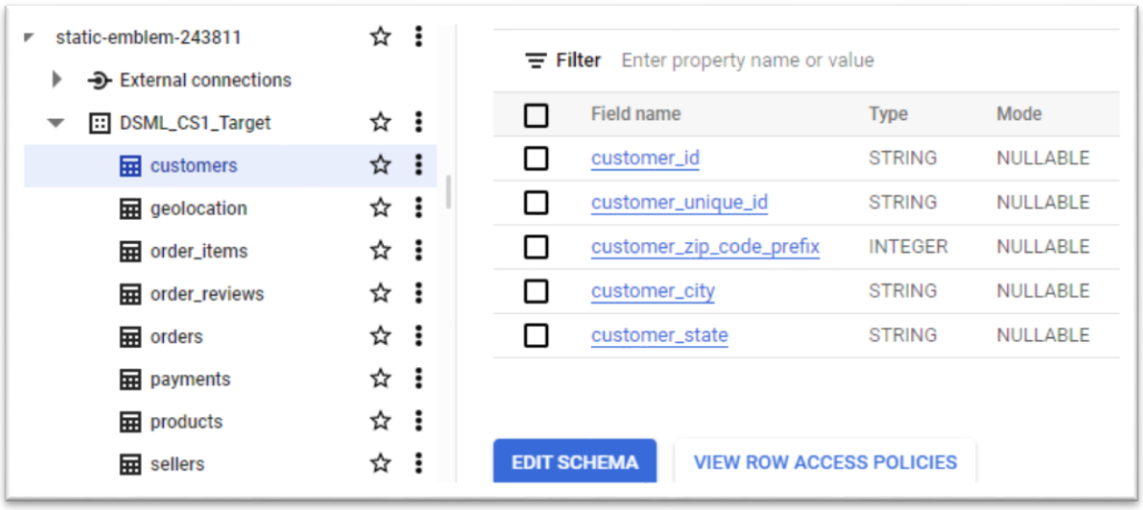
Q1. Business Case: Target SQL

What 'good' looks like?

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

    ➔ By Clicking on dataset & then on each table we can view the structure of that table which would also include the Data type of columns in that particular table e.g.



Alternatively, we can use the below command to view complete information schema for any particular dataset which would include all columns & their data types for all tables.

```
SELECT ddl FROM DSML_CS1_Target.INFORMATION_SCHEMA.TABLES;
```

    2. Time period for which the data is given

    ➔ In the orders table we can 4 different timestamps which are purchase, delivery to carrier, delivery to customer & estimated delivery to customer. We can figure out the first & last purchase ever made & also the first and last delivery ever done to understand the scope of Time period for which the data is given.

```
SELECT min(order_purchase_timestamp) First_Purchase_Date, max(order_purchase_timestamp) Last
_Purchase_Date,
min(order_delivered_customer_date) First_Delivery_Date, max(order_delivered_customer_date) L
ast_Delivery_Date
FROM `static-emblem-243811.DSML_CS1_Target.orders` LIMIT 1000;

select * from `DSML_CS1_Target.orders` where order_purchase_timestamp = '2018-10-
17 17:30:18 UTC';
```

Row	First_Purchase_Date	Last_Purchase_Date	First_Delivery_Date	Last_Delivery_Date
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC	2016-10-11 13:46:32 UTC	2018-10-17 13:22:46 UTC

Row	order_id	customer_id	order_status	order_purchase_timestamp
1	10a045cdf6a5650c21e9cfeb6...	a4b417188addbc05b26b72d5...	canceled	2018-10-17 17:30:18 UTC

We can see the Time period for which the data is given & also we can observe that the last purchase date time is more than the last delivery time which infers that the delivery never reached the customer. On further investigation it was clear that the order was cancelled.

3. Cities and States covered in the dataset

➔ To find all the States & Cities in the dataset we can make use of customers table from which we can get distinct count for both & also all the combinations for all the states and cities in the dataset

```
SELECT count(distinct customer_city) customer_city, count(distinct customer_state) customer_
state
FROM `static-emblem-243811.DSML_CS1_Target.customers`;

SELECT distinct customer_city, customer_state
FROM `static-emblem-243811.DSML_CS1_Target.customers`;
```

JOB INFORMATION | RESULTS | JSON

Row	customer_city	customer_state
1	4119	27

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

Row	customer_city	customer_state
1	acu	RN
2	ico	CE
3	ipe	RS
4	ipu	CE
5	ita	SC
6	itu	SP

Results per page: 50 1 – 50 of 4310

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?

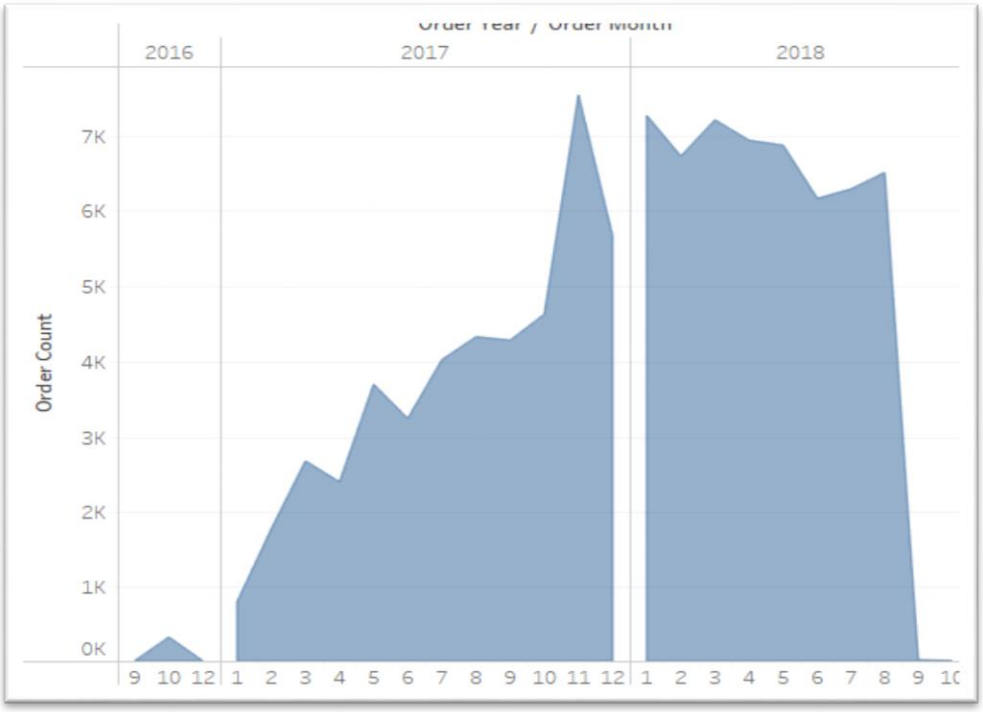
➔ If we get the count of all the orders purchased monthly for years 2016-18 then we can figure out if any trend exist for the given orders or not. This can be done using the following query: -

```
SELECT extract(year from order_purchase_timestamp) as order_year,
extract(month from order_purchase_timestamp) as order_month,
count(order_purchase_timestamp) order_count,
count(distinct customer_unique_id) customer_inflow,
FROM `static-emblem-243811.DSML_CS1_Target.orders` o
inner join `DSML_CS1_Target.customers` c on c.customer_id = o.customer_id
group by order_year, order_month
order by 1,2
```

We do not have adequate data for 2016 although we can see a spike in no. orders purchase in the month of October. In the year 2017 we can clearly see **growing** trend

in orders and a spike in the month of November. In the year 2018 there isn't any considerable trend although we can see a sharp drop from Aug-Sep.

Row	order_year	order_month	order_count	customer_inflov
1	2016	9	4	4
2	2016	10	324	321
3	2016	12	1	1
4	2017	1	800	765
5	2017	2	1780	1755
6	2017	3	2682	2642
7	2017	4	2404	2372



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

➔ Brazilian customers tend to buy the most in the 'Afternoon'. We can first bucket the data on the basis of which hour of the day is the purchase made and then we can name that time period accordingly. Then we can see how many unique customers are making orders in that particular period and how many orders are made.

Note: No. of orders made is different from number of unique customers making orders, as 1 person can make 10 orders (so, no. orders as 10 & no. customer as 1) and 3 people can make 3 orders (so, no. orders as 3 & no. customer as 3). So it's not necessary that higher no. orders mean that more customers are ordering.

Below is the query in order to achieve this: -

```sql
With timeofday as
(SELECT customer_id, order_purchase_timestamp,
CASE
when extract(hour from order_purchase_timestamp) between 21 and 23 or
extract(hour from order_purchase_timestamp) between 0 and 3 then 'Night'
when extract(hour from order_purchase_timestamp) between 4 and 5 then 'Dawn'
when extract(hour from order_purchase_timestamp) between 6 and 11 then 'Morning'
when extract(hour from order_purchase_timestamp) between 12 and 21 then 'Afternoon'
END Timing
FROM `static-emblem-243811.DSML_CS1_Target.orders`)

select Timing, count(distinct c.customer_unique_id) as customer_inflow,
count(order_purchase_timestamp) as order_count
from timeofday tod
inner join `DSML_CS1_Target.customers` as c on c.customer_id = tod.customer_id
group by Timing
```

Row	Timing	customer_inflow	order_count
1	Night	20114	20502
2	Afternoon	54990	56305
3	Morning	21814	22240
4	Dawn	385	394

3. Evolution of E-commerce orders in the Brazil region
    1. Get month on month orders by region, states

➔ We can get the count of no. orders w.r.t. months for 2016 to 2018 by joining customers and orders table.

```sql
SELECT c.customer_state, c.customer_city,
extract(year from o.order_purchase_timestamp) as order_year,
extract(month from o.order_purchase_timestamp) as order_month,
count(o.order_id) as order_count
FROM `static-emblem-243811.DSML_CS1_Target.customers` c
inner join `DSML_CS1_Target.orders` o on o.customer_id = c.customer_id
group by c.customer_state, c.customer_city, order_year, order_month
order by 1,2,3,4
```

Row	customer_state	customer_city	order_year	order_month	order_count
9	AC	rio branco	2017	3	2
10	AC	rio branco	2017	4	4
11	AC	rio branco	2017	5	8
12	AC	rio branco	2017	6	4
13	AC	rio branco	2017	7	5
14	AC	rio branco	2017	8	4

2. How are customers distributed in Brazil?

➔ We can view the customer's distribution across state & region.
Below query can get us the count of customer's state & city wise.

```sql
SELECT count(distinct customer_unique_id) as Total_Customers,
customer_state
FROM `static-emblem-243811.DSML_CS1_Target.customers`
group by customer_state
order by 1 desc;

SELECT count(distinct customer_unique_id) as Total_Customers,
customer_state, customer_city
FROM `static-emblem-243811.DSML_CS1_Target.customers`
group by customer_state, customer_city
order by 1 desc;
```

Row	Total_Customers	customer_state
1	40302	SP
2	12384	RJ
3	11259	MG
4	5277	RS
5	4882	PR
6	3534	SC

Row	Total_Customers	customer_state	customer_city
1	14984	SP	sao paulo
2	6620	RJ	rio de janeiro
3	2672	MG	belo horizonte
4	2069	DF	brasilia
5	1465	PR	curitiba
6	1398	SP	campinas

4. Impact on Economy: Analyse 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)

    ➔ Query for percent increase from 2017 – 2018 between Jan to Aug can be written by using sub-query by joining orders table to order items table in the following way: -

```sql
WITH temp as
(SELECT extract(year from o.order_purchase_timestamp) as order_year,
extract(month from o.order_purchase_timestamp) as order_month, sum(oi.price) price
FROM `static-emblem-243811.DSML_CS1_Target.order_items` oi
inner join `DSML_CS1_Target.orders` o on o.order_id = oi.order_id
group by order_year, order_month
order by 1,2,3)

select order_month, Price_2017, Price_2018,
(((Price_2018 - Price_2017)*100)/Price_2017) percent_inc from
(select t1.order_month, t1.price as Price_2017, t2.price as Price_2018
from temp t1
inner join (select order_month, price
from temp where order_year = 2018 and
order_month between 1 and 8) as t2 on t1.order_month = t2.order_month
where t1.order_year = 2017 and
t1.order_month between 1 and 8
order by 1) t
```

Row	order_month	Price_2017	Price_2018	percent_inc
1	1	120312.869...	950030.360...	689.633195...
2	2	247303.019...	844178.710...	241.353983...
3	3	374344.300...	983213.440...	162.649502...
4	4	359927.230...	996647.750...	176.902570...
5	5	506071.140...	996517.680...	96.9125684...
6	6	433038.600...	865124.310...	99.7799526...
7	7	498031.480...	895507.220...	79.8093606...
8	8	573971.680...	854686.330...	48.9074042...

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

➔ We can calculate the Sum & Mean of Price & freight value by state using following query. Note that I've rounded it off up to 2 decimal places.

```
SELECT c.customer_state, round(sum(price), 2) as price_sum,
round(sum(price)/count(price), 2) price_mean,
round(sum(freight_value), 2) as freight_value_sum,
round(sum(freight_value)/count(freight_value), 2) freight_value_mean
FROM `static-emblem-243811.DSML_CS1_Target.order_items` oi
inner join `DSML_CS1_Target.orders` o on o.order_id = oi.order_id
inner join `DSML_CS1_Target.customers` c on c.customer_id = o.customer_id
group by c.customer_state
order by 2 desc
```

Row	customer_state	price_sum	price_mean	freight_value_sum	freight_value_mean
1	SP	5202955.05	109.65	718723.07	15.15
2	RJ	1824092.67	125.12	305589.31	20.96
3	MG	1585308.03	120.75	270853.46	20.63
4	RS	750304.02	120.34	135522.74	21.74
5	PR	683083.76	119.0	117851.68	20.53

5. Analysis on sales, freight and delivery time
    1. Calculate days between purchasing, delivering and estimated delivery

    ➔ Below is the query to get the purchase delivery estimated gaps, so we can clearly see that how it is distributed across all the orders.

    purchase_to_esitmated_gap: gap between actual purchase date & estimated delivery date. It gives customer a fair idea as to when the product might get delivered. Although the actual delivery date might differ. If it differs too much than that's not a good sign.

    esitmated_to_delivery_gap: As discussed earlier this should be as low as possible if its in negative it means the customer is receiving the item beforehand i.e. before the estimated date which is still better than a higher number (gap) which would further frustrate the customer.

    purchase_to_delivery_gap: This data can also be verified by adding both purchase_to_esitmated_gap + esitmated_to_delivery_gap. It gives us insight as to how it is taking to deliver the product to the customer.

```sql
with waiting_time as
(SELECT order_id, order_purchase_timestamp, order_estimated_delivery_date, order_delivered_c
ustomer_date,
order_estimated_delivery_date - order_purchase_timestamp as purchase_to_esitmated_gap,
order_delivered_customer_date - order_estimated_delivery_date as esitmated_to_delivery_gap,
order_delivered_customer_date - order_purchase_timestamp as purchase_to_delivery_gap,
FROM `static-emblem-243811.DSML_CS1_Target.orders`
order by 4 desc),
t1 as
(select order_id,
round(extract(hour from purchase_to_esitmated_gap)/24,2) purchase_to_esitmated_gap,
round(extract(hour from esitmated_to_delivery_gap)/24,2) esitmated_to_delivery_gap,
round(extract(hour from purchase_to_delivery_gap)/24,2) purchase_to_delivery_gap
from waiting_time
order by 2,3,4)

-- select min(purchase_to_esitmated_gap) least_purchase_to_esitmated_gap,
-- max(purchase_to_esitmated_gap) max_purchase_to_esitmated_gap,
-- min(esitmated_to_delivery_gap) least_esitmated_to_delivery_gap,
-- max(esitmated_to_delivery_gap) max_esitmated_to_delivery_gap,
-- min(purchase_to_delivery_gap) least_purchase_to_delivery_gap,
-- max(purchase_to_delivery_gap) max_purchase_to_delivery_gap
-- from t1;

select * from t1;
```

Row	order_id	purchase_to_esitmated_gap	esitmated_to_delivery_gap	purchase_to_delivery_gap
6	a5ce51395d295fe2c816182f7…	2.0	4.96	7.0
7	91ba8089d2fac808ba2eebac6…	2.04	-0.46	1.54
8	62b4063077164817444ed917…	2.04	-0.25	1.79

Note: To get further insights into the given dataset we can get the min-max of these cols (for all orders combine) in order to get which are the orders with highest & least gaps in all 3 cols

Row	least_purchase_to_esitmat	max_purchase_to_esitmate	least_esitmated_to_delive	max_esitmated_to_deli	least_purchase_to_delive	max_purchase_to_deliver
1	1.63	155.13	-146.0	188.96	0.5	209.63

2. Create columns:
   - time_to_delivery = order_purchase_timestamp-order_delivered_customer_date
   - diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date

➔ These columns are already created in the above question under different names.

1. time_to_delivery = purchase_to_delivery_gap
2. diff_estimated_delivery = esitmated_to_delivery_gap

Note: - Now if we want to permanently update or add those columns in the table than it can be done using the below query

```sql
ALTER TABLE mydataset.mytable
ADD COLUMN new_column STRING;
```

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

➔ We can group by the state data in order to get the mean or the average of freight value, time_to_delivery, diff_estimated_delivery by joining orders, order items & customers table (which is modified).

```
with waiting_time as
(SELECT order_id, customer_id, order_purchase_timestamp, order_estimated_delivery_date, orde
r_delivered_customer_date,
order_estimated_delivery_date - order_purchase_timestamp as purchase_to_esitmated_gap,
order_delivered_customer_date - order_estimated_delivery_date as esitmated_to_delivery_gap,
order_delivered_customer_date - order_purchase_timestamp as purchase_to_delivery_gap,
FROM `static-emblem-243811.DSML_CS1_Target.orders`
order by 4 desc),
t1 as
(select order_id, customer_id,
round(extract(hour from purchase_to_esitmated_gap)/24,2) purchase_to_esitmated_gap,
round(extract(hour from esitmated_to_delivery_gap)/24,2) esitmated_to_delivery_gap,
round(extract(hour from purchase_to_delivery_gap)/24,2) purchase_to_delivery_gap
from waiting_time
order by 2,3,4)

select customer_state, round(avg(o.purchase_to_delivery_gap), 2) mean_time_to_delivery,
round(avg(o.esitmated_to_delivery_gap), 2) mean_estimated_to_delivery,
round(avg(oi.freight_value), 2) mean_freight_value
from `DSML_CS1_Target.customers` c
inner join t1 o on c.customer_id = o.customer_id
inner join `DSML_CS1_Target.order_items` oi on oi.order_id = o.order_id
group by customer_state
order by 3 desc
```

Row	customer_state	mean_time_to_d	mean_estimatec	mean_freight_va
1	AL	24.47	-8.05	35.84
2	MA	21.63	-9.21	38.26
3	SE	21.45	-9.31	36.65
4	ES	15.63	-9.93	22.06
5	BA	19.23	-10.27	26.36
6	CE	20.07	-10.4	32.71

4. Sort the data to get the following:

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

➔ Let's get the highest/lowest average freight value - sort in desc/asc limit 5 and union all to get it all into one table.

```
with waiting_time as
(SELECT order_id, customer_id, order_purchase_timestamp, order_estimated_delivery
_date, order_delivered_customer_date,
order_estimated_delivery_date - order_purchase_timestamp as purchase_to_esitmated
_gap,
order_delivered_customer_date - order_estimated_delivery_date as esitmated_to_del
ivery_gap,
order_delivered_customer_date - order_purchase_timestamp as purchase_to_delivery_
gap,
```

```
FROM `static-emblem-243811.DSML_CS1_Target.orders`
order by 4 desc),
t1 as
(select order_id, customer_id,
round(extract(hour from purchase_to_esitmated_gap)/24,2) purchase_to_esitmated_ga
p,
round(extract(hour from esitmated_to_delivery_gap)/24,2) esitmated_to_delivery_ga
p,
round(extract(hour from purchase_to_delivery_gap)/24,2) purchase_to_delivery_gap
from waiting_time
order by 2,3,4)

(select customer_state,
-- round(avg(o.purchase_to_delivery_gap), 2) mean_time_to_delivery,
-- round(avg(o.esitmated_to_delivery_gap), 2) mean_estimated_to_delivery,
round(avg(oi.freight_value), 2) mean_freight_value
from `DSML_CS1_Target.customers` c
inner join t1 o on c.customer_id = o.customer_id
inner join `DSML_CS1_Target.order_items` oi on oi.order_id = o.order_id
group by customer_state
order by 2 desc
limit 5)
union all
(select customer_state,
-- round(avg(o.purchase_to_delivery_gap), 2) mean_time_to_delivery,
-- round(avg(o.esitmated_to_delivery_gap), 2) mean_estimated_to_delivery,
round(avg(oi.freight_value), 2) mean_freight_value
from `DSML_CS1_Target.customers` c
inner join t1 o on c.customer_id = o.customer_id
inner join `DSML_CS1_Target.order_items` oi on oi.order_id = o.order_id
group by customer_state
order by 2
limit 5)
```

Row	customer_state	mean_freight_value
1	SP	15.15
2	PR	20.53
3	MG	20.63
4	RJ	20.96
5	DF	21.04
6	RR	42.98
7	PB	42.72
8	RO	41.07
9	AC	40.07
10	PI	39.15

2. Top 5 states with highest/lowest average time to delivery

➔ Similarly, by commenting out the mean freight value & estimated time to deliver we can get the average time to delivery which we can sort asc/desc and union all to get it all in one table

Row	customer_state	mean_time_to_delivery
1	SP	8.7
2	PR	11.93
3	MG	11.96
4	DF	12.94
5	SC	14.98
6	RR	28.21
7	AP	28.19
8	AM	26.37
9	AL	24.47
10	PA	23.73

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

➔ Similarly, by commenting out the mean freight value & average time to delivery we can get the estimated time to deliver which we can sort asc/desc and union all to get it all in one table

Row	customer_state	mean_estimated_to_delivery
1	AC	-20.32
2	RO	-19.32
3	AM	-19.21
4	AP	-17.75
5	RR	-17.6
6	AL	-8.05
7	MA	-9.21
8	SE	-9.31
9	ES	-9.93
10	BA	-10.27

6. Payment type analysis:
    1. Month over Month count of orders for different payment types

➔ Below is the query

```
SELECT extract(year from o.order_purchase_timestamp) as order_year,
extract(month from o.order_purchase_timestamp) as order_month,
p.payment_type,
count(o.order_id) total_orders
FROM `static-emblem-243811.DSML_CS1_Target.orders` o
inner join `DSML_CS1_Target.payments` p on p.order_id = o.order_id
group by order_year, order_month, p.payment_type
order by 1,2
```

Row	order_year	order_month	payment_type	total_orders
1	2016	9	credit_card	3
2	2016	10	credit_card	254
3	2016	10	UPI	63
4	2016	10	voucher	23
5	2016	10	debit_card	2
6	2016	12	credit_card	1
7	2017	1	credit_card	583

2. Distribution of payment instalments and count of orders

➔ Below is the query

```
SELECT p.payment_installments,
count(o.order_id) total_orders
FROM `static-emblem-243811.DSML_CS1_Target.orders` o
inner join `DSML_CS1_Target.payments` p on p.order_id = o.order_id
group by p.payment_installments
order by 1
```

Row	payment_installments	total_orders
1	0	2
2	1	52546
3	2	12413
4	3	10461
5	4	7098
6	5	5239
7	6	2020

7. Actionable Insights:

➔ Unavailability of products (State / Region wise): We can group the data region wise and sort it based on the unavailability of the products. By this we'll get which products are mostly unavailability yet in demand region wise. So we can make them available in order to increase profit.

```
1  SELECT customer_state, count(*) products_unavailability
2  FROM `static-emblem-243811.DSML_CS1_Target.orders` o
3  inner join `DSML_CS1_Target.customers` c on c.customer_id = o.customer_id
4  where order_status = 'unavailable'
5  group by customer_state
6  order by 2 desc
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUT

w	customer_state	products_unavailability
1	SP	292
2	MG	75
3	RJ	68
4	PR	40
5	RS	24
6	BA	20

8. Recommendations:

➔ We can improve the customers experience by reducing the estimated delivery time & the actual delivery time. From the given data we can infer that maximum a person is waiting is for 12.92 days ahead of estimation date after which he'll cancel the order. So it recommended to make this more efficient by reducing the gap between estimated delivery time & the actual delivery time.

We can also see that most people are comfortable using credit card as supposed to other mode of payments like debit card, UPI etc. So it is recommended to give offers like annual membership or discounts related to credit cards so as to give customers a push in order to buy products which are in most demand or which are not selling that much. By doing this we can improve the overall sales of the company.

Evaluation Criteria (80 points)

9. Initial exploration of dataset like checking the characteristics of data (10 points)
10. In-depth Exploration (10 points)
11. Evolution of E-commerce orders in the Brazil region (10 points)
12. Impact on Economy (10 points)
13. Analysis on sales, freight and delivery time (10 points)
14. Payment type analysis (10 points)
15. Actionable Insights (10 points)
16. Recommendations (10 points)

Submission Process:

- Type your insights and recommendations in the text editor
- Convert your solutions notebook into PDF, upload it on the dashboard
- Optionally, you may add images/graphs in the text editor by taking screenshots
- After submitting, you will not be allowed to edit your submission