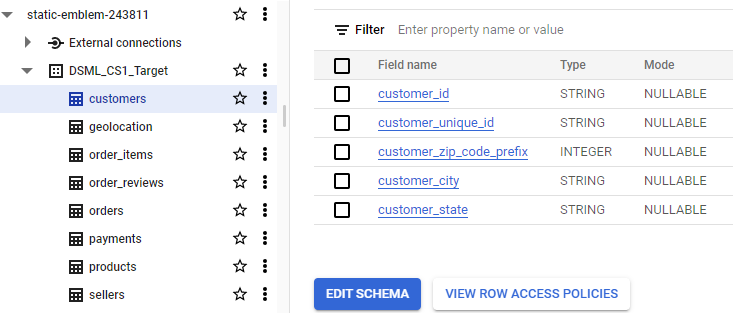
**Q1. Business Case: Target SQL**

**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; |

* 1. 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) Last\_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'; |

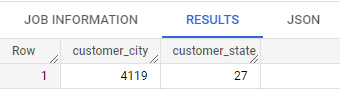




We can see the Time period for which the data is given & also we can observe that the last purchase date time is more than the last delivery time which infers that the delivery never reached the customer. On further investigation it was clear that the order was cancelled.

* 1. 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`; |



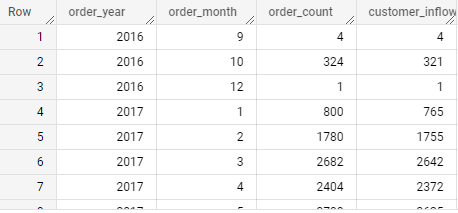


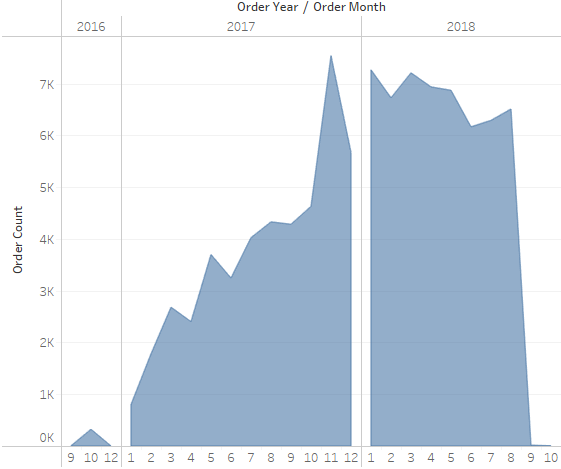
1. In-depth Exploration:  
   1. Is there a growing trend on e-commerce in Brazil? How can we describe a complete scenario? Can we see some seasonality with peaks at specific months?

* 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.



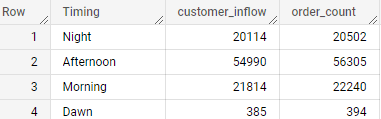


* 1. What time do Brazilian customers tend to buy (Dawn, Morning, Afternoon or Night)?
* Brazilian customers tend to buy the most in the ‘Afternoon’. We can first bucket the data on the basis of which hour of the day is the purchase made and then we can name that time period accordingly. Then we can 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: -

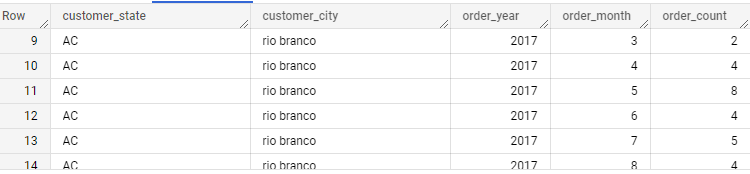
|  |
| --- |
| With timeofday as  (SELECT customer\_id, order\_purchase\_timestamp,  CASE  when extract(hour from order\_purchase\_timestamp) between 21 and 23 or  extract(hour from order\_purchase\_timestamp) between 0 and 3 then 'Night'  when extract(hour from order\_purchase\_timestamp) between 4 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 |



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

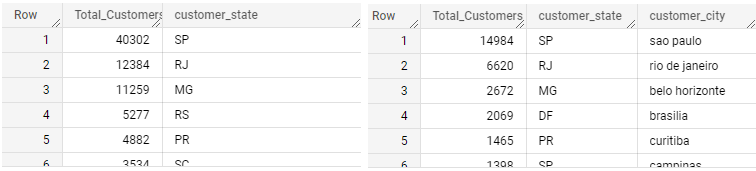
* We can get the count of no. orders w.r.t. months for 2016 to 2018 by joining customers and orders table.

|  |
| --- |
| 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 |



* 1. 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.

|  |
| --- |
| 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; |



1. 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)

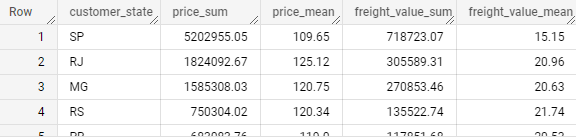
* Query for percent increase from 2017 – 2018 between Jan to Aug can be written by using sub-query by joining orders table to order items table in the following way: -

|  |
| --- |
| 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 |



* 1. 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 |



1. 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.

|  |
| --- |
| with waiting\_time as  (SELECT order\_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\_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; |



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



* 1. 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.

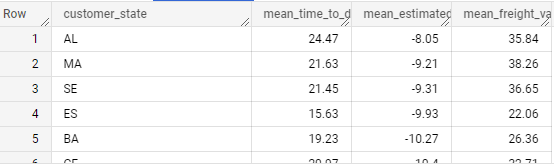
1. time\_to\_delivery = purchase\_to\_delivery\_gap
2. diff\_estimated\_delivery = esitmated\_to\_delivery\_gap

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

|  |
| --- |
| ALTER TABLE mydataset.mytable  ADD COLUMN new\_column STRING; |

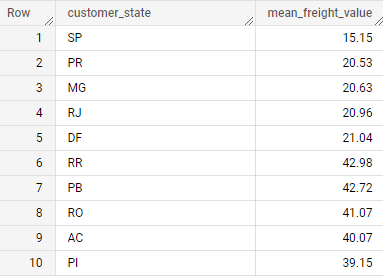
* 1. Group data by state, take mean of freight value, time\_to\_delivery, diff\_estimated\_delivery
* We can group by the state data in order to get the mean or the average of freight value, time\_to\_delivery, diff\_estimated\_delivery by joining orders, order items & customers table (which is modified).

|  |
| --- |
| with waiting\_time as  (SELECT order\_id, customer\_id, order\_purchase\_timestamp, order\_estimated\_delivery\_date, 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\_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 |

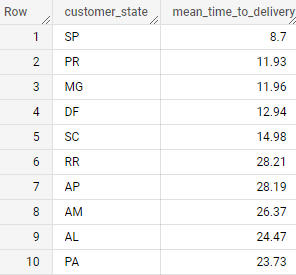


* 1. Sort the data to get the following:  
     + Top 5 states with highest/lowest average freight value - sort in desc/asc limit 5
* Let’s get the highest/lowest average freight value - sort in desc/asc limit 5 and union all to get it all into one table.

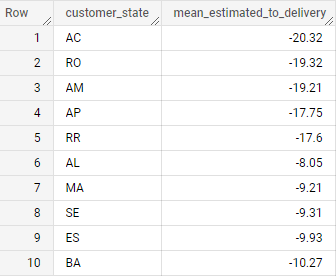
|  |
| --- |
| with waiting\_time as  (SELECT order\_id, customer\_id, order\_purchase\_timestamp, order\_estimated\_delivery\_date, order\_delivered\_customer\_date,  order\_estimated\_delivery\_date - order\_purchase\_timestamp as purchase\_to\_esitmated\_gap,  order\_delivered\_customer\_date - order\_estimated\_delivery\_date as esitmated\_to\_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 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) |



* + - 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



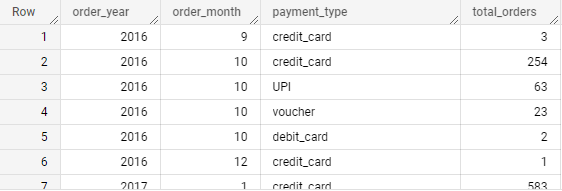
* + - Top 5 states where delivery is really fast/ not so fast compared to estimated date
* Similarly, by commenting out the mean freight value & average time to delivery we can get the estimated time to deliver which we can sort asc/desc and union all to get it all in one table



1. 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 |



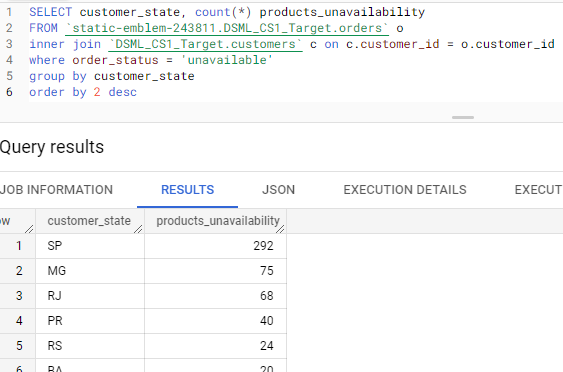
* 1. Distribution of payment instalments and count of orders
* Below is the query

|  |
| --- |
| 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 |



1. 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. 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.

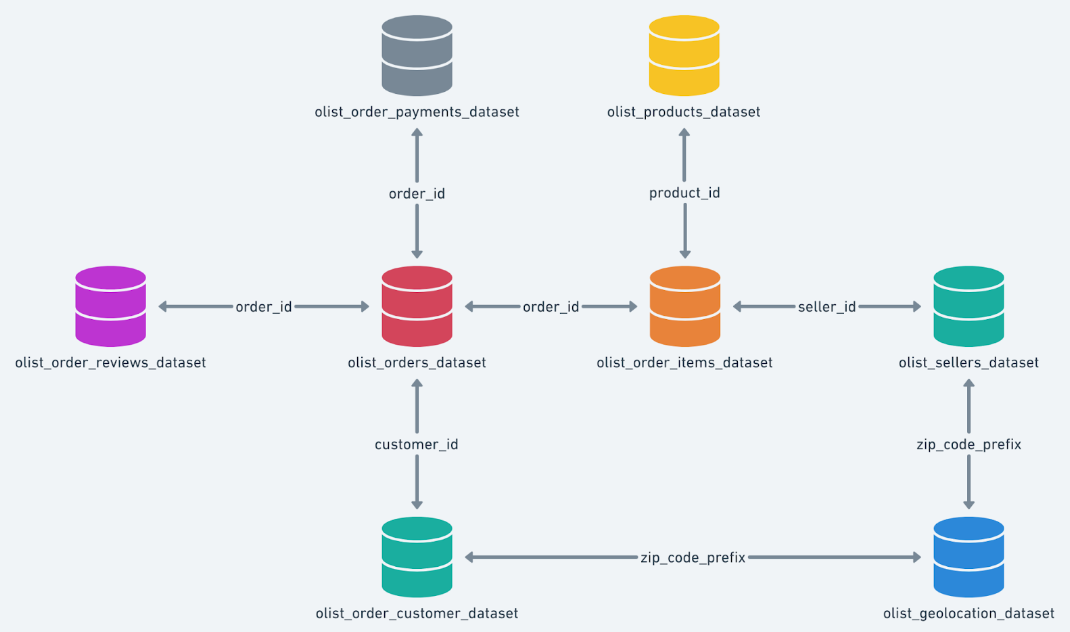
We can also see that most people are comfortable using credit card as supposed to other mode of payments like debit card, UPI etc. So it is recommended to give offers like annual membership or discounts related to credit cards so as to give customers a push in order to buy products which are in most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

**Evaluation Criteria (80 points)**

1. Initial exploration of dataset like checking the characteristics of data (10 points)
2. In-depth Exploration (10 points)
3. Evolution of E-commerce orders in the Brazil region (10 points)
4. Impact on Economy (10 points)
5. Analysis on sales, freight and delivery time (10 points)
6. Payment type analysis (10 points)
7. Actionable Insights (10 points)
8. 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



Other student’s solution: -

7. Actionable Insights -

1) Target project contains 6 tables i.e. customers, products, sellers, orders, order\_items, and payments

2) Time period started from 4th SEP 2016 to 17th OCT 2018

3) Seasonable trend is generally MAY - AUG

4) Most of the orders are place in afternoon and night timings. Dawn time has less number of orders

5) Percentage of increase in orders in 2018 is 17% but the avg cost price is increase by 0.63%

6) Most of the orders are delivered after the estimated time of delivery which is around 11%.

7) Highest freight value is 410 for customer state PI and time\_to\_delivery 11 while estimated delivery is 30

8) Most of the customer are using credit cards for the payment

8. Recommendation

1) if there is any down time require for changes in database level target can do in dawn time.

2) Since there is low volume in some of the months except MAY - AUG. Target can launch exciting offers to attract the customers

3) 11% of orders are getting delivered late to to customer this can be minimize