Target

Business Case

Scaler

Topic: SQL

Duration: 1 week

Mindset:

1. Evaluation will be kept lenient, so make sure you attempt this case study.
2. It is understandable that you might struggle with getting started on this. Just brainstorm, discuss with peers, or get help from TAs.
3. Try to attempt this before it is discussed in the Live Case Discussion with the Instructor.
4. There is no right or wrong answer. We have to become comfortable dealing with uncertainty in business. This is exactly the skill we want to develop.

Context

Target is one of the world's most recognized brands and one of America's leading retailers. 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 business case has information of 100k orders from 2016 to 2018 made at Target in Brazil. Its features allows viewing an order from multiple dimensions: from order status, price, payment and freight performance to customer location, product attributes and finally reviews written by customers.

Dataset:
https://drive.google.com/drive/folders/1TGEc66YKbD443nslRi1bWgVd238gJCnb?usp=sharing

Data is available in 8 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

Each feature or columns of different CSV files are described below:

The customers.csv contain following features:

Features	**Description**
customer_id	Id of the consumer who made the purchase.
customer_unique_id	Unique Id of the consumer.
customer_zip_code_prefix	Zip Code of the location of the consumer.
customer_city	Name of the City from where order is made.
customer_state	State Code from where order is made(Ex- sao paulo-SP).

The geolocations.csv contain following features:

Features	**Description**
geolocation_zip_code_prefix	first 5 digits of zip code
geolocation_lat	latitude.
geolocation_lng	longitude.
geolocation_city	city name.
geolocation_state	state.

The sellers.csv contains following features:

Features	**Description**
seller_id	Unique Id of the seller registered
seller_zip_code_prefix	Zip Code of the location of the seller.
seller_city	Name of the City of the seller.
seller_state	State Code (Ex- são paulo-SP)

The order_items.csv contain following features:

Features	**Description**
order_id	A unique id of order made by the consumers.
order_item_id	A Unique id given to each item ordered in the order.
product_id	A unique id given to each product available on the site.
seller_id	Unique Id of the seller registered in Target.
shipping_limit_date	The date before which shipping of the ordered product must be completed.
price	Actual price of the products ordered .
freight_value	Price rate at which a product is delivered from one point to another.

The payments.csv contain following features:

Features	**Description**
order_id	A unique id of order made by the consumers.
payment_sequential	sequences of the payments made in case of EMI.
payment_type	mode of payment used.(Ex-Credit Card)
payment_installments	number of installments in case of EMI purchase.
payment_value	Total amount paid for the purchase order.

The orders.csv contain following features:

Features	**Description**
order_id	A unique id of order made by the consumers.

customer_id	Id of the consumer who made the purchase.
order_status	status of the order made i.e delivered, shipped etc.
order_purchase_timestamp	Timestamp of the purchase.
order_delivered_carrier_date	delivery date at which carrier made the delivery.
order_delivered_customer_date	date at which customer got the product.
order_estimated_delivery_date	estimated delivery date of the products.

The reviews.csv contain following features:

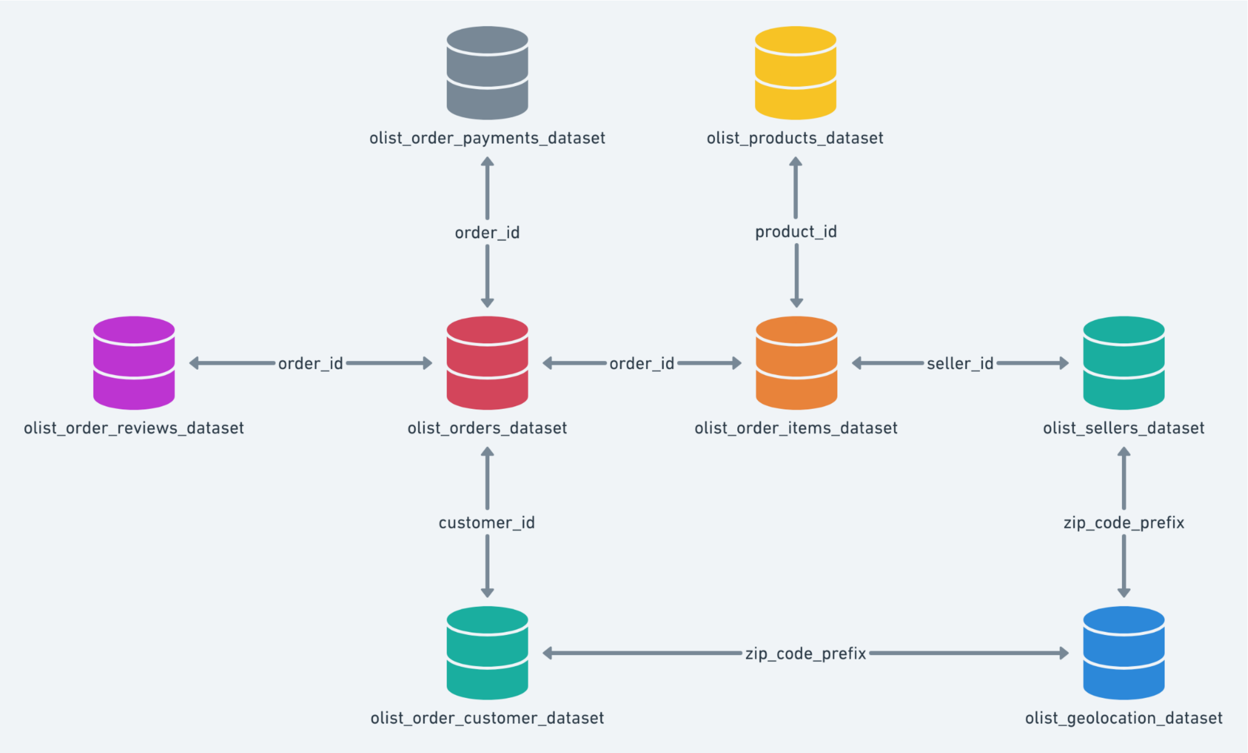
Features	**Description**
review_id	Id of the review given on the product ordered by the order id.
order_id	A unique id of order made by the consumers.
review_score	review score given by the customer for each order on the scale of 1–5.
review_comment_title	Title of the review
review_comment_message	Review comments posted by the consumer for each order.
review_creation_date	Timestamp of the review when it is created.
review_answer_timestamp	Timestamp of the review answered.

The products.csv contain following features:

Features	**Description**
product_id	A unique identifier for the proposed project.
product_category_name	Name of the product category

product_name_lenght	length of the string which specifies the name given to the products ordered.
product_description_lenght	length of the description written for each product ordered on the site.
product_photos_qty	Number of photos of each product ordered available on the shopping portal.
product_weight_g	Weight of the products ordered in grams.
product_length_cm	Length of the products ordered in centimeters.
product_height_cm	Height of the products ordered in centimeters.
product_width_cm	width of the product ordered in centimeters.

High level overview of relation between datasets:



Assume you are a data scientist at Target, and are given this data to analyze and provide some insights and recommendations from it.

What does ‘good’ look like?

1. Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset
    a. Data type of columns in a table- orders
    b. Time period for which the data is given
    c. Cities and States of **customers who ordered during the given period**
2. In-depth Exploration:
    a. 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?
    b. What time do Brazilian customers tend to buy (Dawn, Morning, Afternoon or Night)?
3. Evolution of E-commerce orders in the Brazil region:
    a. Get month on month orders by states
    b. Distribution of customers across the states in Brazil
4. Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight and others.
    a. 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*
    b. Mean & Sum of price and freight value by customer state
5. Analysis on sales, freight and delivery time
    a. Calculate days between purchasing, delivering and estimated delivery
    b. 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
    c. Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery
    d. Sort the data to get the following:
        a. Top 5 states with highest/lowest average freight value - sort in desc/asc limit 5
        b. Top 5 states with highest/lowest average time to delivery

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

6. Payment type analysis:
    1. Month over Month count of orders for different payment types
    2. Count of orders based on the no. of payment installments

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.

- Use Word doc to paste your SQL queries along with the screenshot of the first 10 rows
- Convert your solutions notebook into PDF, and upload the same on the platform
- Optionally, you may add images/graphs in the text editor by taking screenshots
- After submitting, you **will not be allowed to edit** your submission.

Answer Sheet

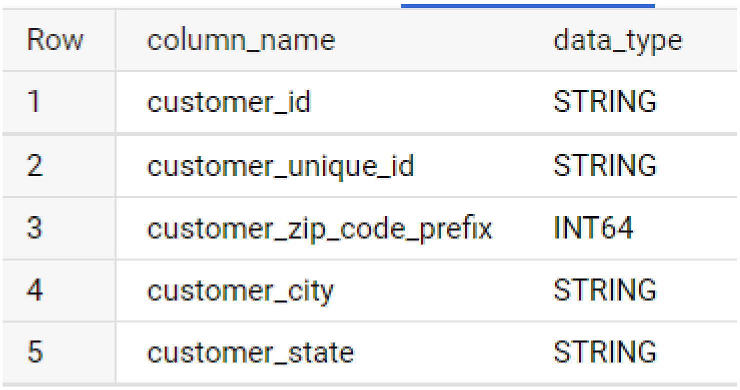
Questions:

1. **Initial exploratory questions:**

Here we'll first try to explore the data, understand it and answer some questions with simple queries

1.a. Data type of columns in a table

```
SELECT column_name, data_type
FROM sqlfreetest-353004.Ecommerce.INFORMATION_SCHEMA.COLUMNS
WHERE table_name = 'customers'
```

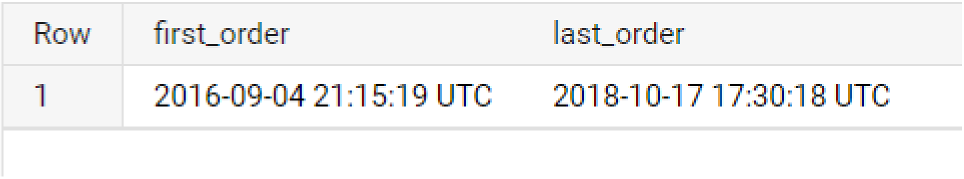


Row	column_name	data_type
1	customer_id	STRING
2	customer_unique_id	STRING
3	customer_zip_code_prefix	INT64
4	customer_city	STRING
5	customer_state	STRING

* INFORMATION_SCHEMA.COLUMNS is a special function in bigquery

1.b. Get the time period for which the data is given

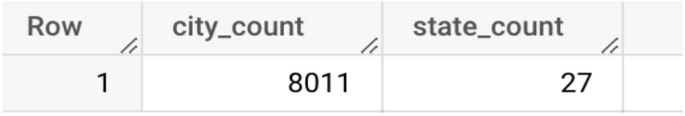
```
SELECT MIN(order_purchase_timestamp) AS first_order,
    MAX(order_purchase_timestamp) AS last_order
    from `sqlfreetest-353004.Ecommerce.orders`
```



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

1.c. Number of cities and states in our dataset

```
select count(distinct(geolocation_city)) as city_count,

count(distinct(geolocation_state)) as state_count

from `sqlfreetest-353004.Ecommerce.geolocations`;
```

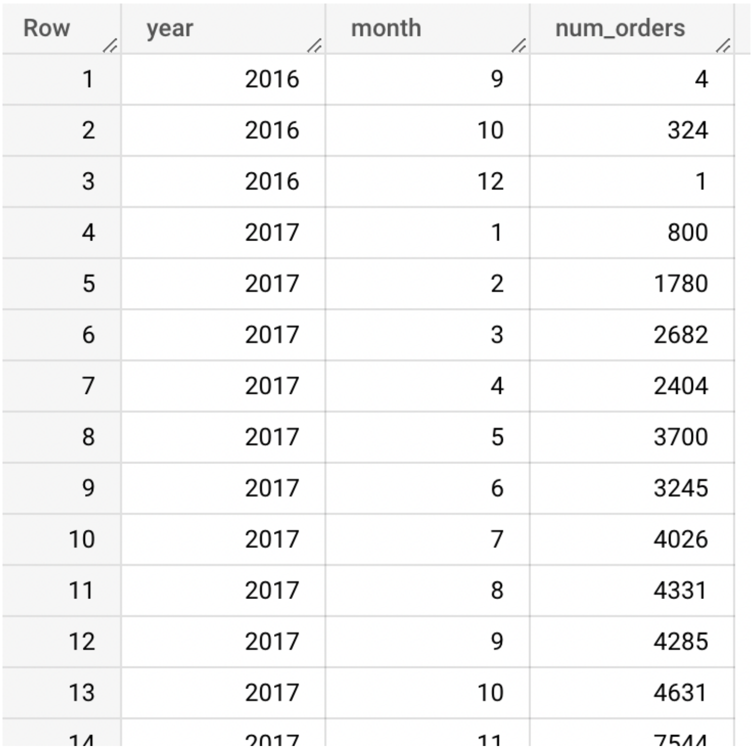


Row	city_count	state_count
1	8011	27

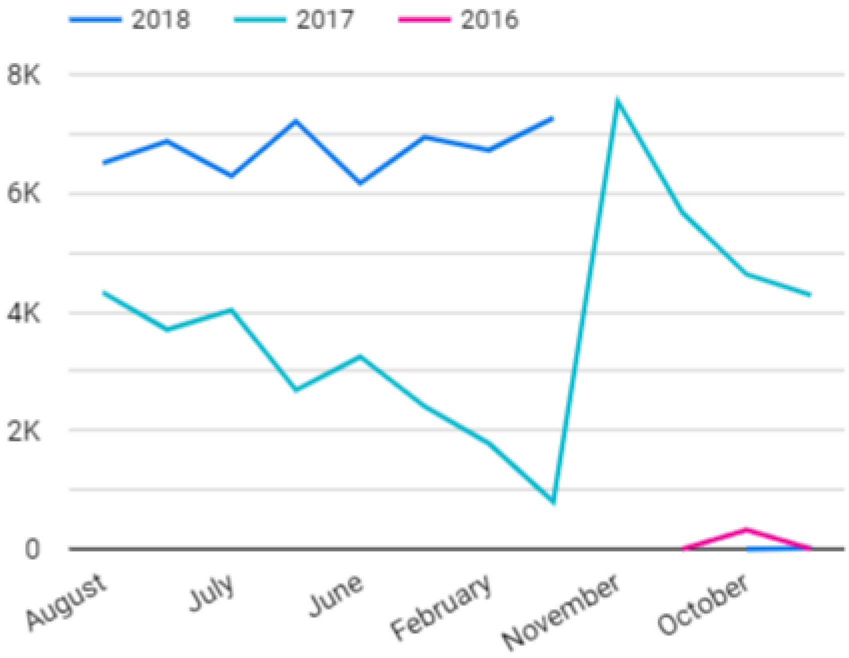
2.a. Is there a growing trend in e-commerce in Brazil? How can we describe a complete scenario?

Now we'll try to understand the trend in the data and see how things have changed for the data that we have over the course of time

```sql
SELECT Extract( year from order_purchase_timestamp) as year,
Extract( month from order_purchase_timestamp) as month,
COUNT(1)  as num_orders
FROM `sqlfreetest-353004.Ecommerce.orders`
GROUP BY year, month
ORDER BY year, month
```

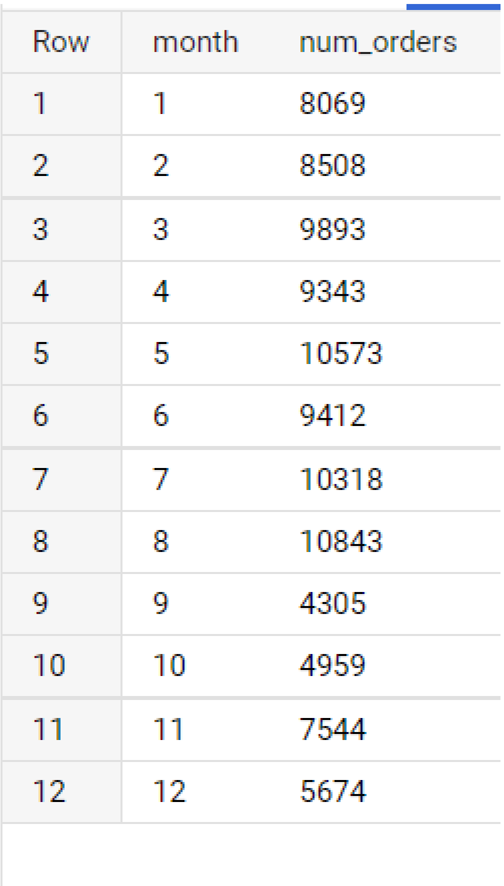


Row	year	month	num_orders
1	2016	9	4
2	2016	10	324
3	2016	12	1
4	2017	1	800
5	2017	2	1780
6	2017	3	2682
7	2017	4	2404
8	2017	5	3700
9	2017	6	3245
10	2017	7	4026
11	2017	8	4331
12	2017	9	4285
13	2017	10	4631
14	2017	11	7544



Question: Can we see some seasonality with peaks at specific months?

```
SELECT  Extract( month from order_purchase_timestamp) as month, COUNT(1)  as
num_orders
FROM `sqlfreetest-353004.Ecommerce.orders`
GROUP BY 1
ORDER BY 1
```

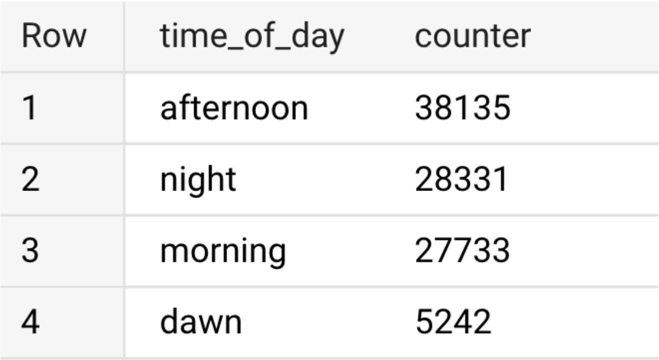


Row	month	num_orders
1	1	8069
2	2	8508
3	3	9893
4	4	9343
5	5	10573
6	6	9412
7	7	10318
8	8	10843
9	9	4305
10	10	4959
11	11	7544
12	12	5674

In general we can see clearly that customers are more prone to buy things online than before.

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

```
select
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'
end as time_of_day, count(distinct order_id) as counter
from `sqlfreetest.Ecommerce.orders`
group by 1
order by 2 desc
```



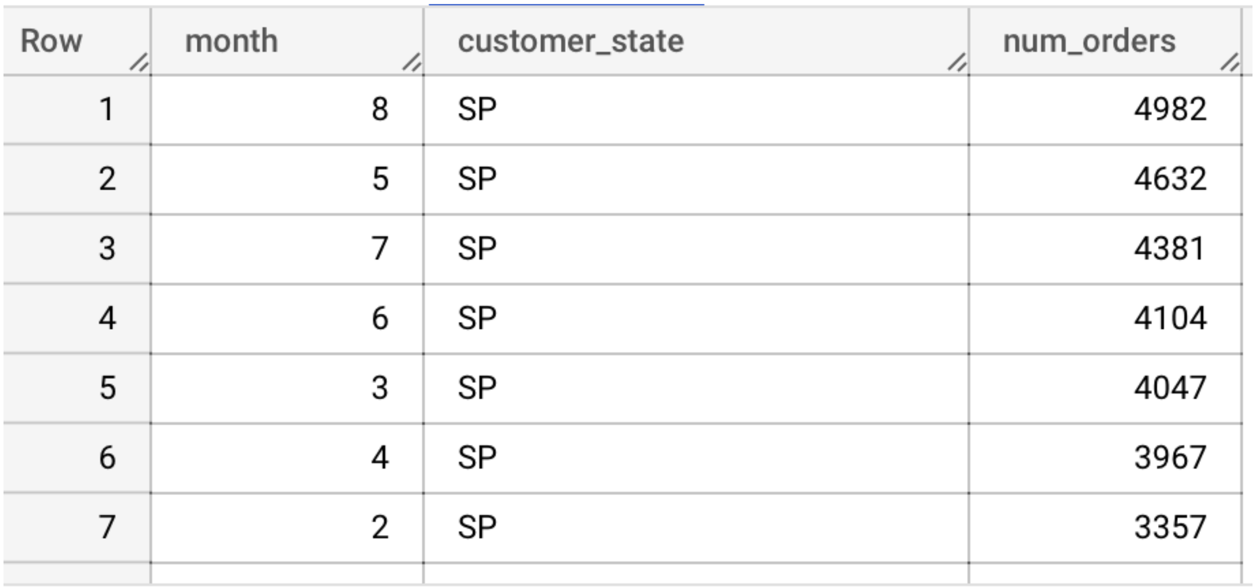
Row	time_of_day	counter
1	afternoon	38135
2	night	28331
3	morning	27733
4	dawn	5242

3. Evolution of E-commerce orders in Brazil region

Now we'll try to understand data based on state or city level and see what variations are present and how the people in various states order and receive deliveries.

3.a. Get month on month orders by states.

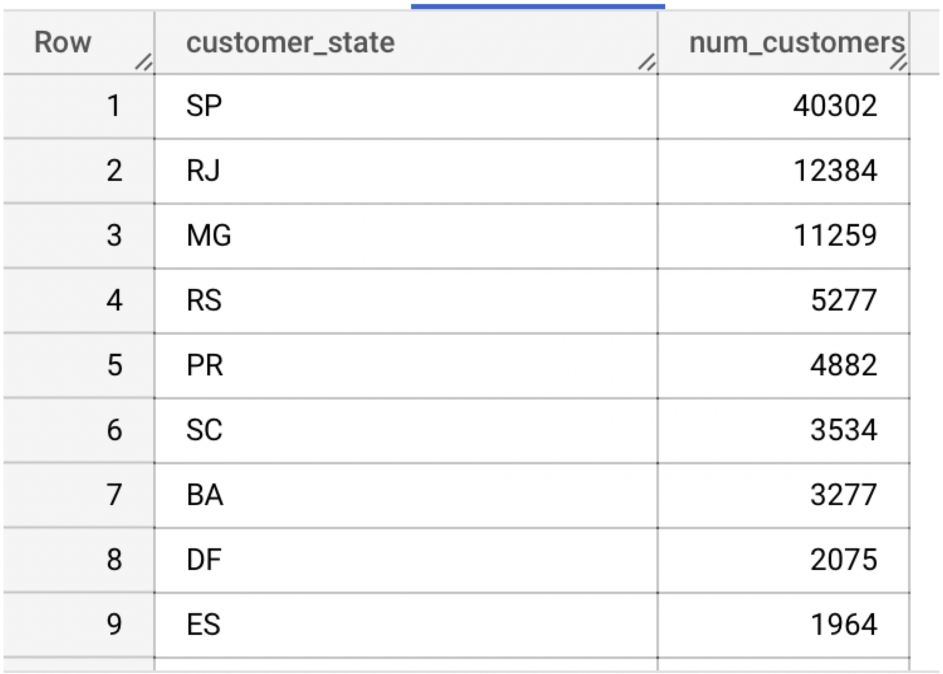
```
select Extract( month from order_purchase_timestamp) as month,
c.customer_state, COUNT(1) as num_orders
from `sqlfreetest-353004.Ecommerce.orders` o
inner join `sqlfreetest-353004.Ecommerce.customers` c
on o.customer_id = c.customer_id
group by c.customer_state, month
order by num_orders desc
```



Row	month	customer_state	num_orders
1	8	SP	4982
2	5	SP	4632
3	7	SP	4381
4	6	SP	4104
5	3	SP	4047
6	4	SP	3967
7	2	SP	3357

3.b. Distribution of customers across the states in Brazil

```
select customer_state, COUNT(distinct(customer_unique_id)) as

num_customers

from `sqlfreetest-353004.Ecommerce.customers`

group by customer_state

order by num_customers desc;
```



Row	customer_state	num_customers
1	SP	40302
2	RJ	12384
3	MG	11259
4	RS	5277
5	PR	4882
6	SC	3534
7	BA	3277
8	DF	2075
9	ES	1964

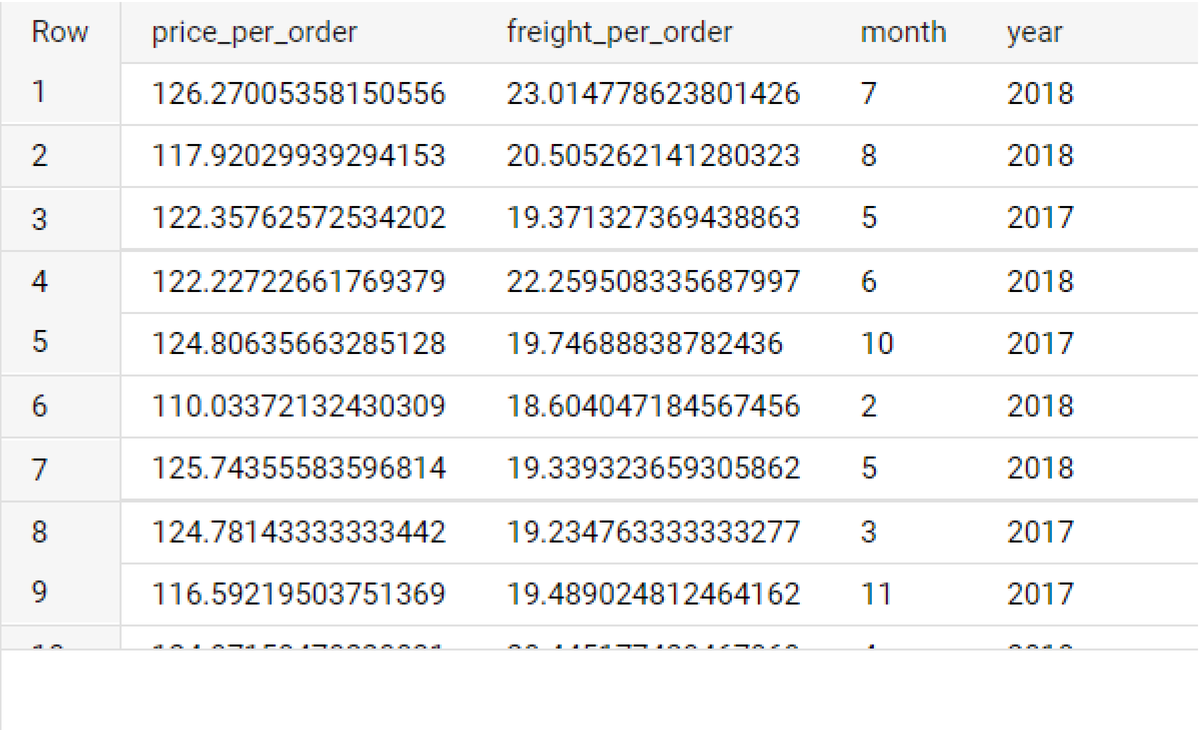
4. Impact on Economy

Until now, we just answered questions on the E-commerce scenario considering the number of orders received. We could see the volumetry by a month, day of week, time of the day and even the geolocation states.

Now, **we will analyze the money movement by e-commerce by looking at order prices, freight and others.**

- **Create CTE Table and new columns:**
    - **price_per_order = sum(price)/count(order_id)**
    - **freight_per_order= sum(freight_value)/count(order_id)**
    - **Group the data on yearly and monthly level**

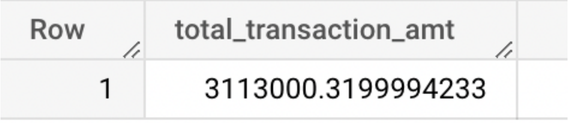
```
with cte_table as (

select Extract( month from o.order_purchase_timestamp) as month,
Extract( year from o.order_purchase_timestamp) as year,
(sum(price)/count(o.order_id)) as price_per_order,
(sum(freight_value)/count(o.order_id)) as freight_per_order
from `sqlfreetest-353004.Ecommerce.orders` o
inner join `sqlfreetest-353004.Ecommerce.order_items` i
on o.order_id= i.order_id
group by year,month

)
select (price_per_order), (freight_per_order), month , year
from cte_table
```



Row	price_per_order	freight_per_order	month	year
1	126.27005358150556	23.014778623801426	7	2018
2	117.92029939294153	20.505262141280323	8	2018
3	122.35762572534202	19.371327369438863	5	2017
4	122.22722661769379	22.259508335687997	6	2018
5	124.80635663285128	19.74688838782436	10	2017
6	110.03372132430309	18.604047184567456	2	2018
7	125.74355583596814	19.339323659305862	5	2018
8	124.78143333333442	19.234763333333277	3	2017
9	116.59219503751369	19.489024812464162	11	2017

4.a. Total amount sold in 2017 between Jan to august (Jan to Aug because data is available starting 2017 01 to 2018 08) and we can only compare cycles with cycles

```
with cte_table as (

select

Extract( month from order_purchase_timestamp) as month,

Extract( year from order_purchase_timestamp) as year,

sum(price) as total_price,

sum(freight_value) as total_freight

from `sqlfreetest-353004.Ecommerce.orders` o

inner join `sqlfreetest-353004.Ecommerce.order_items` i

on o.order_id= i.order_id

group by year, month
)
select sum(total_price) as total_transaction_amt
from cte_table
where year =2017 and month between 1 and 8
```

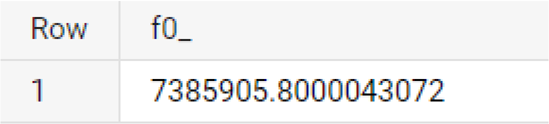


Row	total_transaction_amt
1	3113000.3199994233

(3.9M)

4.a. Total amount sold in 2018 between Jan to august

```
with cte_table as (
select
Extract( month from order_purchase_timestamp) as month,
Extract( year from order_purchase_timestamp) as year,
sum(price) as total_price,
sum(freight_value) as total_freight
from `sqlfreetest-353004.Ecommerce.orders` o
inner join `sqlfreetest-353004.Ecommerce.order_items` i
on o.order_id= i.order_id
group by year, month

)
select sum(total_price)
from cte_table
where year =2018 and month between 1 and 8
```

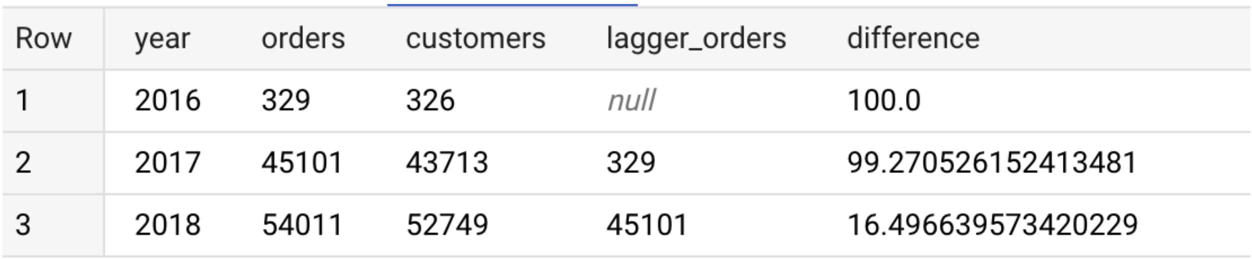


Row	f0_
1	7385905.8000043072

4.a. % increase from 2017 to 2018

Using another example (using orders and customers table)

```
select *, (orders-coalesce(lagger_orders,0))/coalesce(orders,1)*100 as difference from (

select *, lag (orders,1) over (order by year asc) as lagger_orders from (
select extract(year from a.order_purchase_timestamp) as year,
count(distinct a.order_id) as orders,
count(distinct b.customer_unique_id) as customers
from `sqlfreetest.Ecommerce.orders` a
left join `sqlfreetest.Ecommerce.customers` b
on a.customer_id=b.customer_id
group by 1
)base) base_2
order by year asc
```



Row	year	orders	customers	lagger_orders	difference
1	2016	329	326	*null*	100.0
2	2017	45101	43713	329	99.270526152413481
3	2018	54011	52749	45101	16.496639573420229

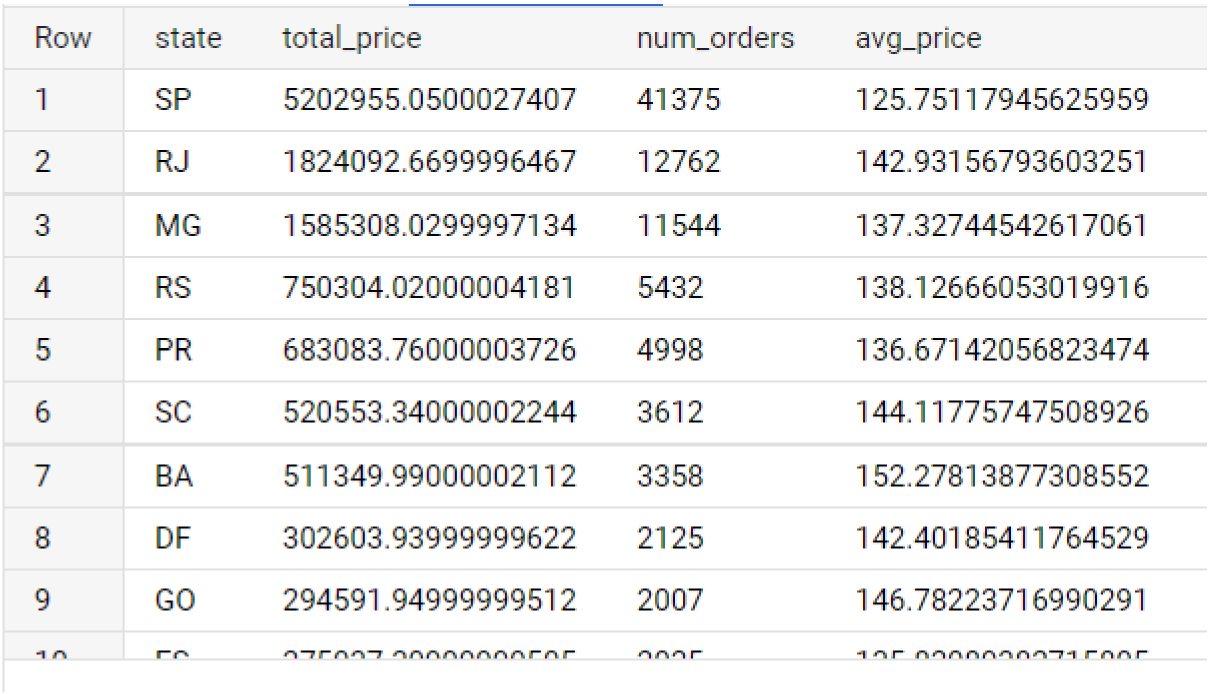
% Increase from below query

```
with base as
(
select * from Ecommerce.orders a
inner join
Ecommerce.payments b
on a.order_id = b.order_id
where
extract(year from a.order_purchase_timestamp) between 2017 and 2018
and
extract(month from a.order_purchase_timestamp) between 1 and 8
),
base_2 as
(
select extract(year from order_purchase_timestamp) as year, sum(payment_value) as cost from
base
group by 1
order by 1 asc
),
base_3 as
```

```
(
select *, lead(cost, 1) over (order by year) as next_year_cost from base_2
)
select *, (next_year_cost - cost)/ cost *100 as per_inc from base_3
```

- **4.b. Sum and mean price by customer state**

```
with cte_table as (
select
c.customer_state as state,
sum(price) as total_price,
count(distinct(o.order_id)) as num_orders
from `sqlfreetest-353004.Ecommerce.orders` o
inner join `sqlfreetest-353004.Ecommerce.order_items` i
on o.order_id= i.order_id
inner join `sqlfreetest-353004.Ecommerce.customers` c
on o.customer_id=c.customer_id
group by state

)
select state, total_price,  num_orders,(total_price/num_orders) as avg_price
from cte_table
order by total_price desc
```

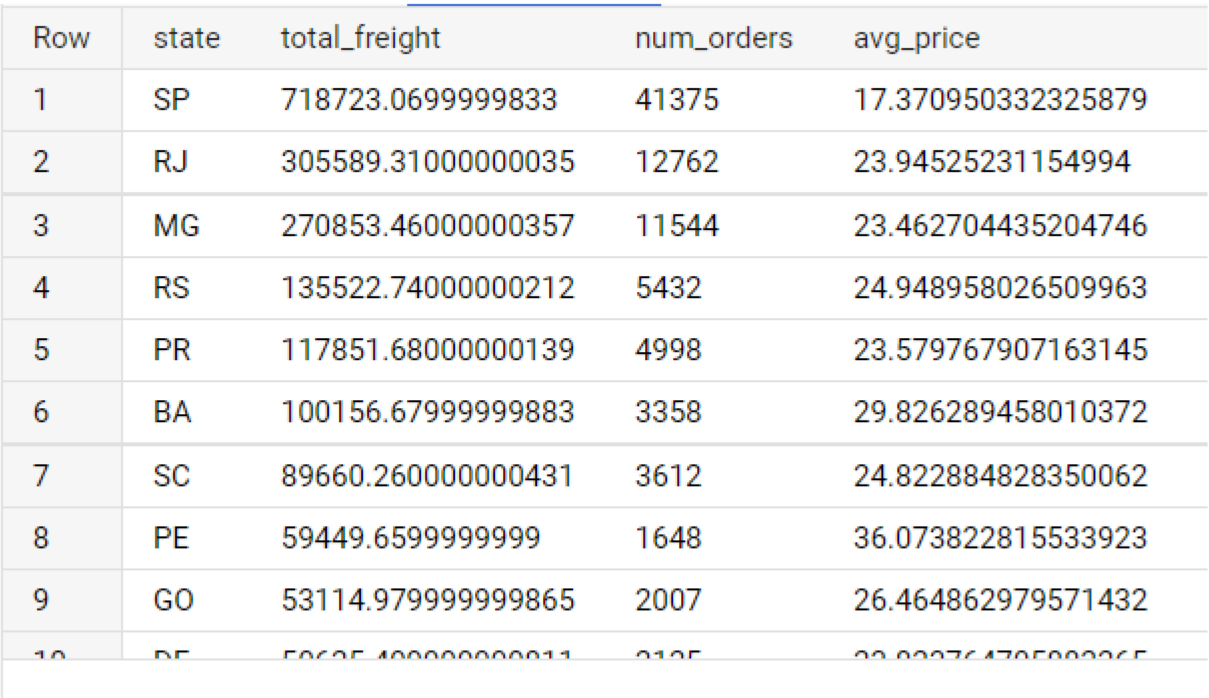


Row	state	total_price	num_orders	avg_price
1	SP	5202955.0500027407	41375	125.75117945625959
2	RJ	1824092.6699996467	12762	142.93156793603251
3	MG	1585308.0299997134	11544	137.32744542617061
4	RS	750304.02000004181	5432	138.12666053019916
5	PR	683083.76000003726	4998	136.67142056823474
6	SC	520553.34000002244	3612	144.11775747508926
7	BA	511349.99000002112	3358	152.27813877308552
8	DF	302603.93999999622	2125	142.40185411764529
9	GO	294591.94999999512	2007	146.78223716990291
10	ES	275037.30999999505	2025	[illegible]

It's very interesting to see how some states have a high total amount sold and a low price per order. If we look at SP (São Paulo) for example, it's possible to see that it is the state with most valuable state for e-commerce (5202955 sold) but it is also where customers pay less per order (125.75 per order)

4.b. Sum and mean freight by customer state

```
with cte_table as (
select
c.customer_state as state,
sum(freight_value) as total_freight,count(distinct(o.order_id)) as num_orders
from `sqlfreetest-353004.Ecommerce.orders` o
inner join `sqlfreetest-353004.Ecommerce.order_items` i
on o.order_id= i.order_id
inner join `sqlfreetest-353004.Ecommerce.customers` c
on o.customer_id=c.customer_id
group by state

)
select state, total_freight,  num_orders,(total_freight/num_orders) as avg_price
from cte_table
order by total_freight desc
```

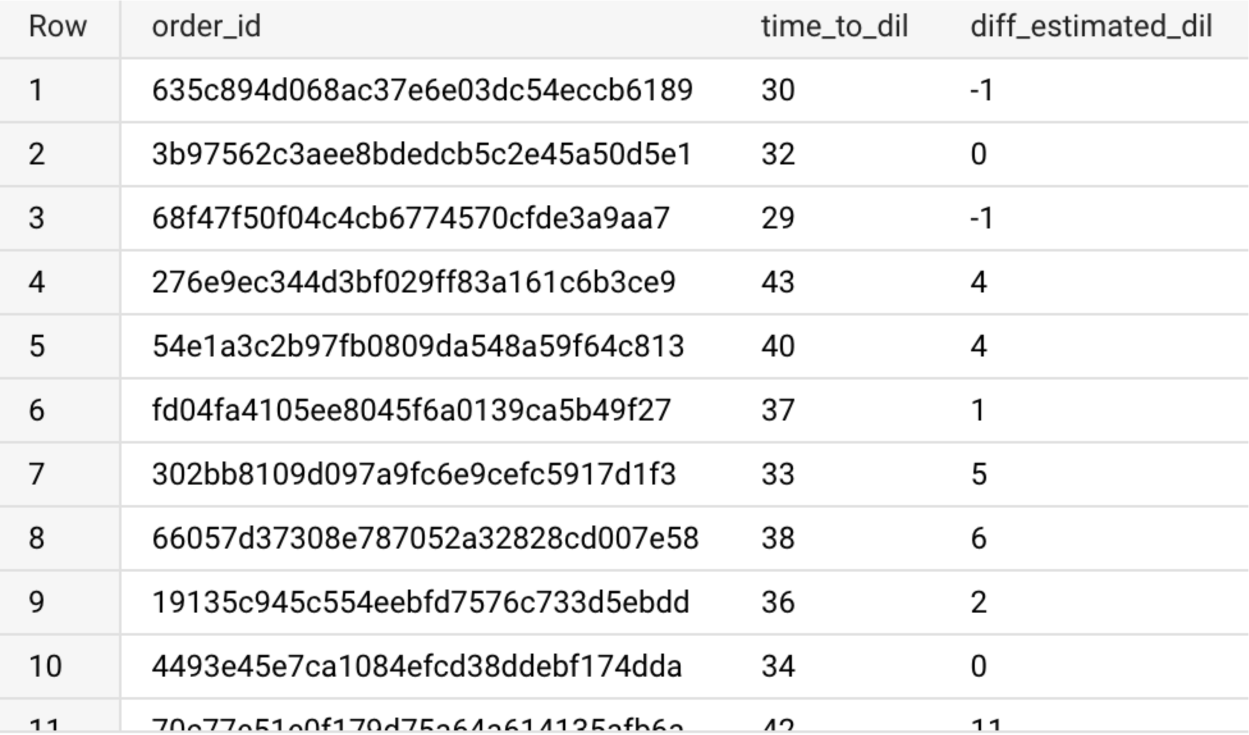


Row	state	total_freight	num_orders	avg_price
1	SP	718723.0699999833	41375	17.370950332325879
2	RJ	305589.31000000035	12762	23.94525231154994
3	MG	270853.46000000357	11544	23.462704435204746
4	RS	135522.74000000212	5432	24.948958026509963
5	PR	117851.68000000139	4998	23.579767907163145
6	BA	100156.67999999883	3358	29.826289458010372
7	SC	89660.260000000431	3612	24.822884828350062
8	PE	59449.6599999999	1648	36.073822815533923
9	GO	53114.979999999865	2007	26.464862979571432
10	DF	50625.499999999911	2135	23.823764705882365

5. Analysis on sales, freight and delivery time

create new columns for time to delivery and difference in estimated vs actual delivery

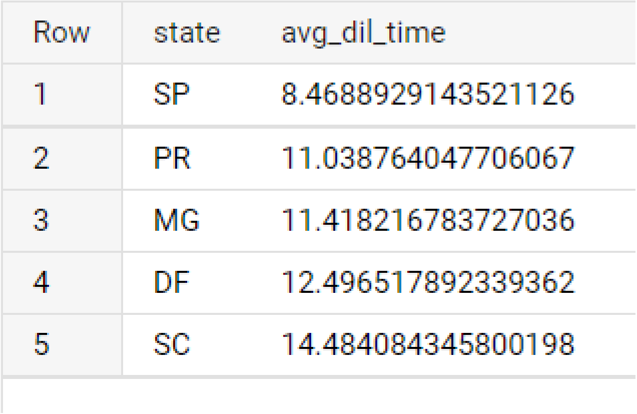
```
select order_id,TIMESTAMP_DIFF(
order_delivered_customer_date,order_purchase_timestamp, DAY) as time_to_dil,
TIMESTAMP_DIFF(  order_delivered_customer_date,order_estimated_delivery_date ,
DAY) as diff_estimated_dil
from `sqlfreetest-353004.Ecommerce.orders`
where order_status='delivered'
```



Row	order_id	time_to_dil	diff_estimated_dil
1	635c894d068ac37e6e03dc54eccb6189	30	-1
2	3b97562c3aee8bdedcb5c2e45a50d5e1	32	0
3	68f47f50f04c4cb6774570cfde3a9aa7	29	-1
4	276e9ec344d3bf029ff83a161c6b3ce9	43	4
5	54e1a3c2b97fb0809da548a59f64c813	40	4
6	fd04fa4105ee8045f6a0139ca5b49f27	37	1
7	302bb8109d097a9fc6e9cefc5917d1f3	33	5
8	66057d37308e787052a32828cd007e58	38	6
9	19135c945c554eebfd7576c733d5ebdd	36	2
10	4493e45e7ca1084efcd38ddebf174dda	34	0
11	70c77e51e0f179d75a64a614135afb6a	42	11

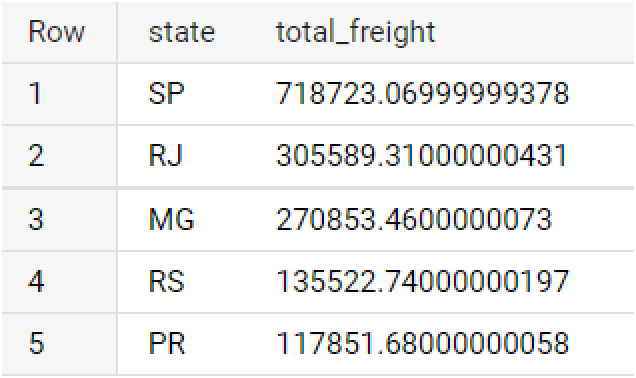
5.4.a Top 5 states with highest/lowest average time to delivery

```
select g.geolocation_state as state,

SUM(TIMESTAMP_DIFF(  order_delivered_customer_date,order_purchase_timestamp,

DAY))/COUNT(ORDER_ID) AS avg_dil_time,

from `sqlfreetest-353004.Ecommerce.orders` o

inner join `sqlfreetest-353004.Ecommerce.customers` c

on o.customer_id=c.customer_id

inner join `sqlfreetest-353004.Ecommerce.geolocations` g

on c.customer_zip_code_prefix=g.geolocation_zip_code_prefix

where order_status='delivered'

group by state

order by avg_dil_time

limit 5
```



Row	state	avg_dil_time
1	SP	8.4688929143521126
2	PR	11.038764047706067
3	MG	11.418216783727036
4	DF	12.496517892339362
5	SC	14.484084345800198

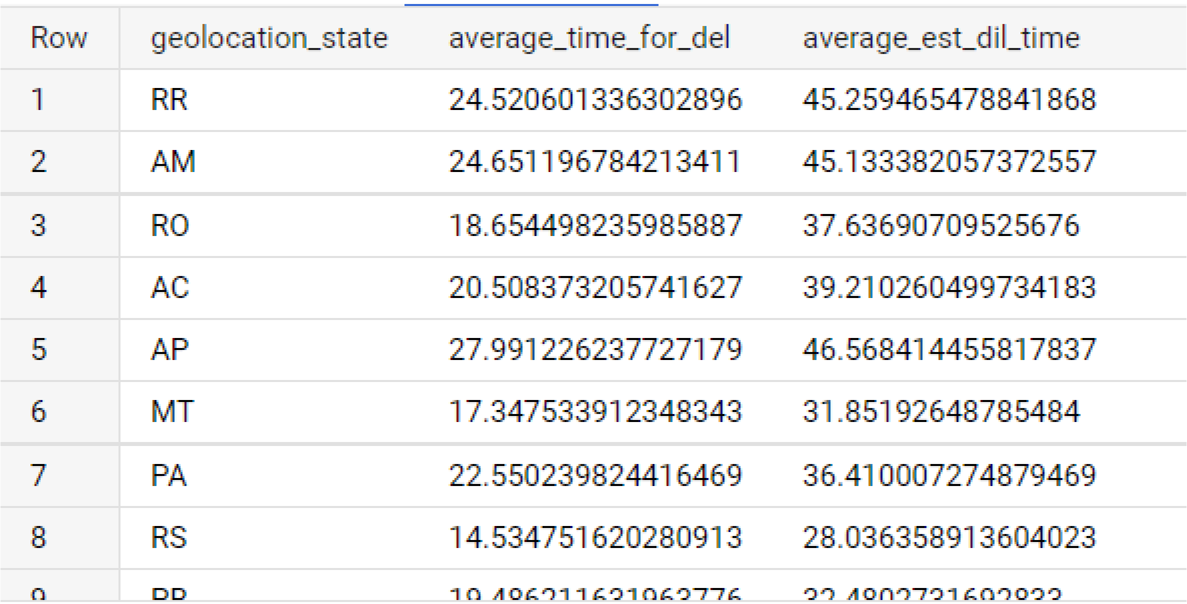
```
with cte_table as (
select
c.customer_state as state,
sum(freight_value) as total_freight,count(distinct(o.order_id)) as num_orders
from `sqlfreetest-353004.Ecommerce.orders` o
inner join `sqlfreetest-353004.Ecommerce.order_items` i
on o.order_id= i.order_id
inner join `sqlfreetest-353004.Ecommerce.customers` c
on o.customer_id=c.customer_id
group by state

)
select state, total_freight,
from cte_table
order by total_freight desc
limit 5
```



Row	state	total_freight
1	SP	718723.06999999378
2	RJ	305589.31000000431
3	MG	270853.4600000073
4	RS	135522.74000000197
5	PR	117851.68000000058

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

```
select geolocation_state,
SUM(TIMESTAMP_DIFF(order_delivered_customer_date,order_purchase_timestamp,
DAY))/COUNT(ORDER_ID) AS average_time_for_del,
SUM(TIMESTAMP_DIFF(order_estimated_delivery_date, order_purchase_timestamp,
DAY))/COUNT(ORDER_ID) AS average_est_dil_time,
from `sqlfreetest-353004.Ecommerce.orders` o
inner join `sqlfreetest-353004.Ecommerce.customers` c
on o.customer_id=c.customer_id
inner join `sqlfreetest-353004.Ecommerce.geolocations` g
on c.customer_zip_code_prefix=g.geolocation_zip_code_prefix
where order_status='delivered'
group by geolocation_state
order by (average_time_for_del-average_est_dil_time)
```

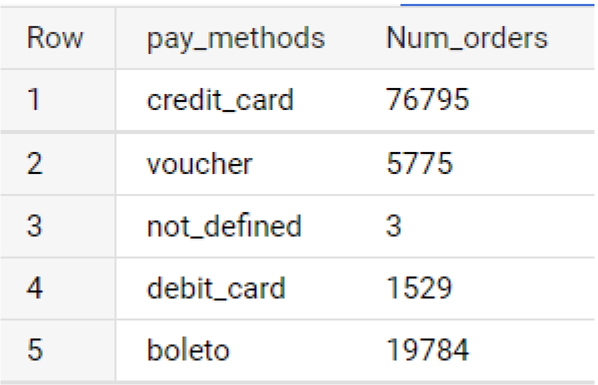


Row	geolocation_state	average_time_for_del	average_est_dil_time
1	RR	24.520601336302896	45.259465478841868
2	AM	24.651196784213411	45.133382057372557
3	RO	18.654498235985887	37.63690709525676
4	AC	20.508373205741627	39.210260499734183
5	AP	27.991226237727179	46.568414455817837
6	MT	17.347533912348343	31.85192648785484
7	PA	22.550239824416469	36.410007274879469
8	RS	14.534751620280913	28.036358913604023
9	PB	19.486211631963776	32.4802731692833

6. Payment type analysis

6.1. Count of orders for different payment types

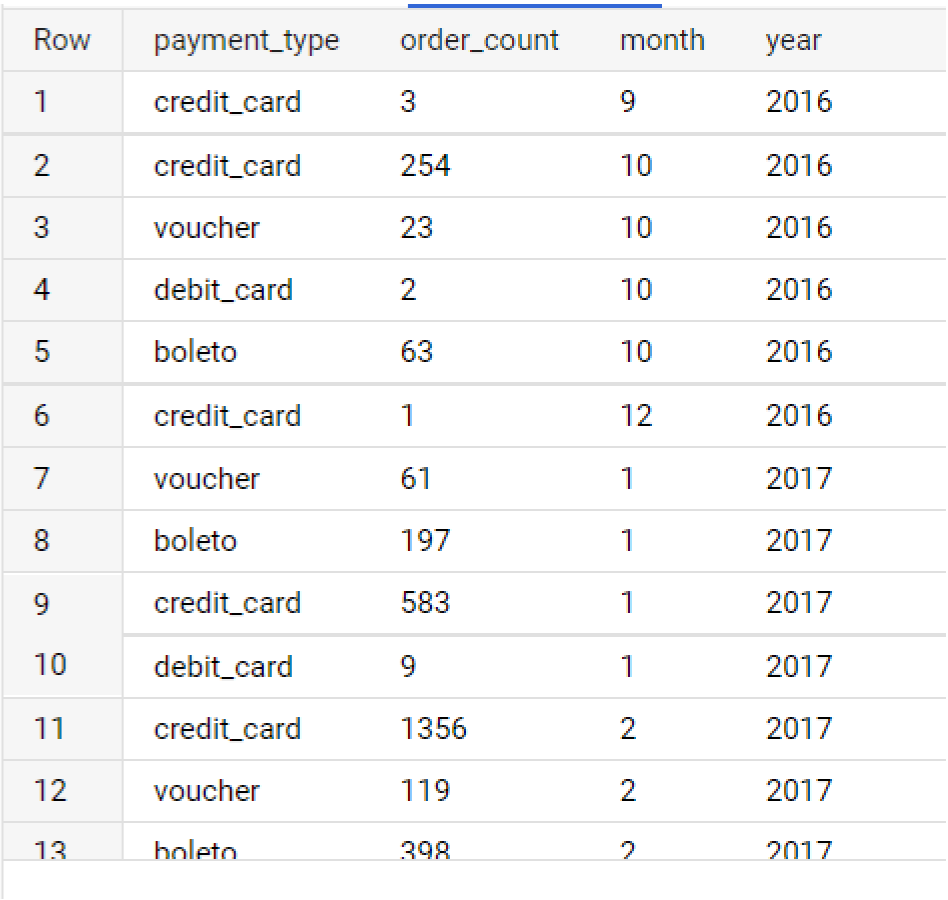
```
SELECT distinct(payment_type) as pay_methods, count(order_id) as Num_orders
FROM `sqlfreetest-353004.Ecommerce.payments`
GROUP BY payment_type;
```



Row	pay_methods	Num_orders
1	credit_card	76795
2	voucher	5775
3	not_defined	3
4	debit_card	1529
5	boleto	19784

6.2. Count of orders for different payment types Month over Month

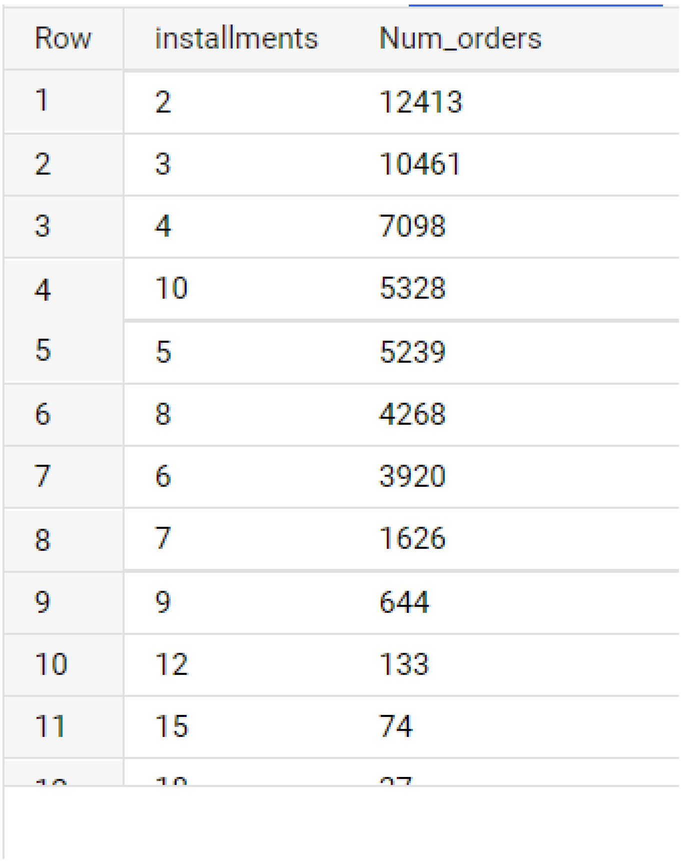
```
SELECT payment_type, COUNT(o.order_id) as order_count,
Extract( month from order_purchase_timestamp) as month,
Extract( year from order_purchase_timestamp) as year,
FROM `sqlfreetest-353004.Ecommerce.payments` p
join `sqlfreetest-353004.Ecommerce.orders` o
on o.order_id=p.order_id
GROUP BY payment_type, year, month
order by year, month;
```



Row	payment_type	order_count	month	year
1	credit_card	3	9	2016
2	credit_card	254	10	2016
3	voucher	23	10	2016
4	debit_card	2	10	2016
5	boleto	63	10	2016
6	credit_card	1	12	2016
7	voucher	61	1	2017
8	boleto	197	1	2017
9	credit_card	583	1	2017
10	debit_card	9	1	2017
11	credit_card	1356	2	2017
12	voucher	119	2	2017
13	boleto	398	2	2017

6.3. Count of orders based on the no. of payment installments

```
SELECT distinct(payment_installments) as installments, count(order_id) as Num_orders,
FROM `sqlfreetest-353004.Ecommerce.payments`
where payment_installments>1
GROUP BY payment_installments
order by Num_orders desc;
```

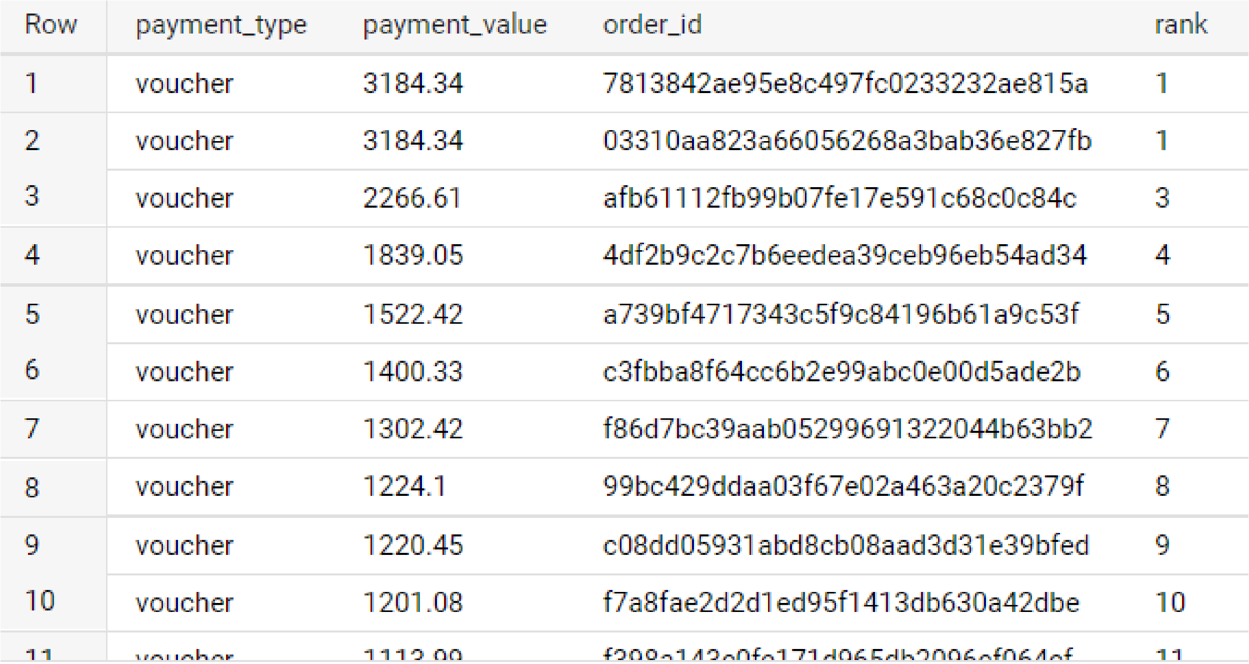


Row	installments	Num_orders
1	2	12413
2	3	10461
3	4	7098
4	10	5328
5	5	5239
6	8	4268
7	6	3920
8	7	1626
9	9	644
10	12	133
11	15	74
12	18	27

ADDITIONAL QUESTIONS

- Rank payment_value partitioned by payment_type

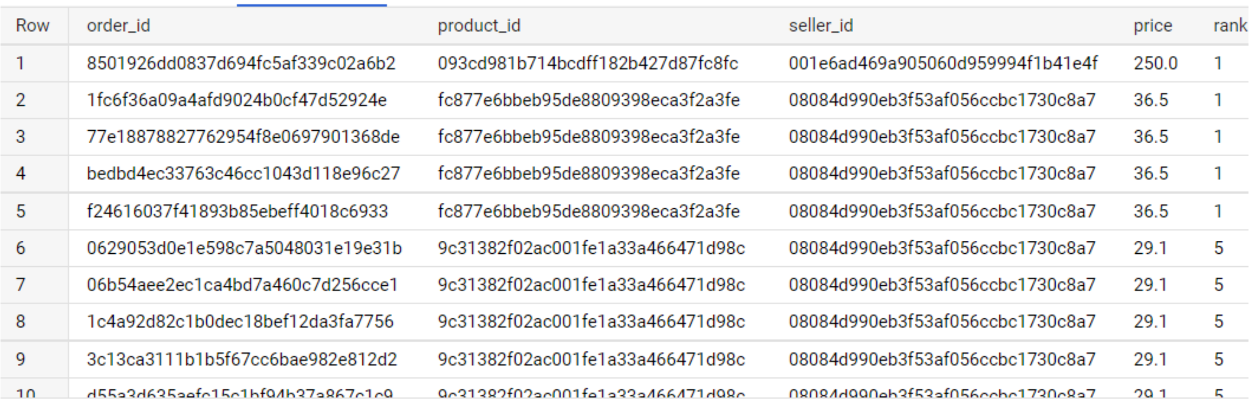
```
select payment_type, payment_value, order_id,
rank() over(partition by payment_type order by payment_value desc ) as rank
from `sqlfreetest-353004.Ecommerce.payments`
```



Row	payment_type	payment_value	order_id	rank
1	voucher	3184.34	7813842ae95e8c497fc0233232ae815a	1
2	voucher	3184.34	03310aa823a66056268a3bab36e827fb	1
3	voucher	2266.61	afb61112fb99b07fe17e591c68c0c84c	3
4	voucher	1839.05	4df2b9c2c7b6eedea39ceb96eb54ad34	4
5	voucher	1522.42	a739bf4717343c5f9c84196b61a9c53f	5
6	voucher	1400.33	c3fbba8f64cc6b2e99abc0e00d5ade2b	6
7	voucher	1302.42	f86d7bc39aab05299691322044b63bb2	7
8	voucher	1224.1	99bc429ddaa03f67e02a463a20c2379f	8
9	voucher	1220.45	c08dd05931abd8cb08aad3d31e39bfed	9
10	voucher	1201.08	f7a8fae2d2d1ed95f1413db630a42dbe	10
11	voucher	1113.99	f398a143c0fe171d965db2096cf064af	11

- **For each seller rank the items by price**

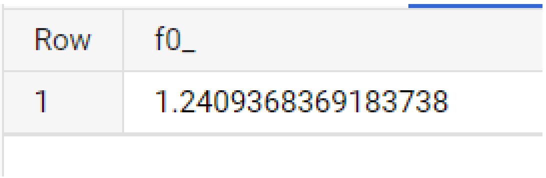
```
select  order_id,product_id, seller_id, price,
rank() over(partition by seller_id order by price desc  ) as rank
from `sqlfreetest-353004.Ecommerce.order_items`
```



Row	order_id	product_id	seller_id	price	rank
1	8501926dd0837d694fc5af339c02a6b2	093cd981b714bcdff182b427d87fc8fc	001e6ad469a905060d959994f1b41e4f	250.0	1
2	1fc6f36a09a4afd9024b0cf47d52924e	fc877e6bbeb95de8809398eca3f2a3fe	08084d990eb3f53af056ccbc1730c8a7	36.5	1
3	77e18878827762954f8e0697901368de	fc877e6bbeb95de8809398eca3f2a3fe	08084d990eb3f53af056ccbc1730c8a7	36.5	1
4	bedbd4ec33763c46cc1043d118e96c27	fc877e6bbeb95de8809398eca3f2a3fe	08084d990eb3f53af056ccbc1730c8a7	36.5	1
5	f24616037f41893b85ebeff4018c6933	fc877e6bbeb95de8809398eca3f2a3fe	08084d990eb3f53af056ccbc1730c8a7	36.5	1
6	0629053d0e1e598c7a5048031e19e31b	9c31382f02ac001fe1a33a466471d98c	08084d990eb3f53af056ccbc1730c8a7	29.1	5
7	06b54aee2ec1ca4bd7a460c7d256cce1	9c31382f02ac001fe1a33a466471d98c	08084d990eb3f53af056ccbc1730c8a7	29.1	5
8	1c4a92d82c1b0dec18bef12da3fa7756	9c31382f02ac001fe1a33a466471d98c	08084d990eb3f53af056ccbc1730c8a7	29.1	5
9	3c13ca3111b1b5f67cc6bae982e812d2	9c31382f02ac001fe1a33a466471d98c	08084d990eb3f53af056ccbc1730c8a7	29.1	5
10	d55a3d635aefc15c1bf94b37a867c1c9	9c31382f02ac001fe1a33a466471d98c	08084d990eb3f53af056ccbc1730c8a7	29.1	5

- **What percentage of orders were canceled or unavailable**

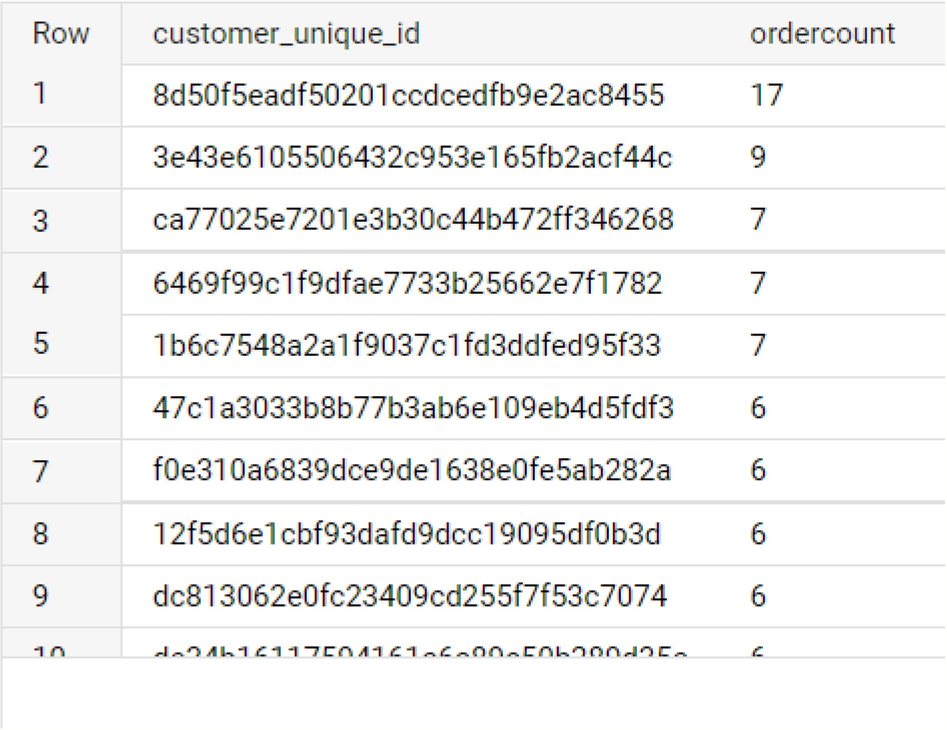
```
select counter/total*100  from (
select sum(case when order_status in ('canceled','unavailable') then 1 else 0 end) as counter,
count (1) as total from `sqlfreetest-353004.Ecommerce.orders`
);
```



Row	f0_
1	1.2409368369183738

- **Find customers with more than one order**

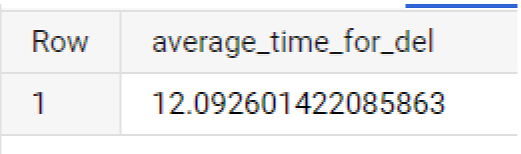
```
SELECT customer_unique_id, count(1) as ordercount FROM
`sqlfreetest-353004.Ecommerce.customers` c
JOIN  `sqlfreetest-353004.Ecommerce.orders` o ON c.customer_id = o.customer_id
GROUP BY customer_unique_id
HAVING COUNT(order_purchase_timestamp) >1
ORDER BY COUNT(order_purchase_timestamp) DESC ;
```



Row	customer_unique_id	ordercount
1	8d50f5eadf50201ccdcedfb9e2ac8455	17
2	3e43e6105506432c953e165fb2acf44c	9
3	ca77025e7201e3b30c44b472ff346268	7
4	6469f99c1f9dfae7733b25662e7f1782	7
5	1b6c7548a2a1f9037c1fd3ddfed95f33	7
6	47c1a3033b8b77b3ab6e109eb4d5fdf3	6
7	f0e310a6839dce9de1638e0fe5ab282a	6
8	12f5d6e1cbf93dafd9dcc19095df0b3d	6
9	dc813062e0fc23409cd255f7f53c7074	6
10	de34b16117594161a6a89c50b289d35a	6

Avg time for delivery

```
select SUM(TIMESTAMP_DIFF(order_delivered_customer_date,order_purchase_timestamp,
DAY))/COUNT(ORDER_ID) AS average_time_for_del
from `sqlfreetest-353004.Ecommerce.orders`where order_status='delivered' limit 1
```



Row	average_time_for_del
1	12.092601422085863