SQL Business Case

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.

Data Dictionary :

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 **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- sao 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.

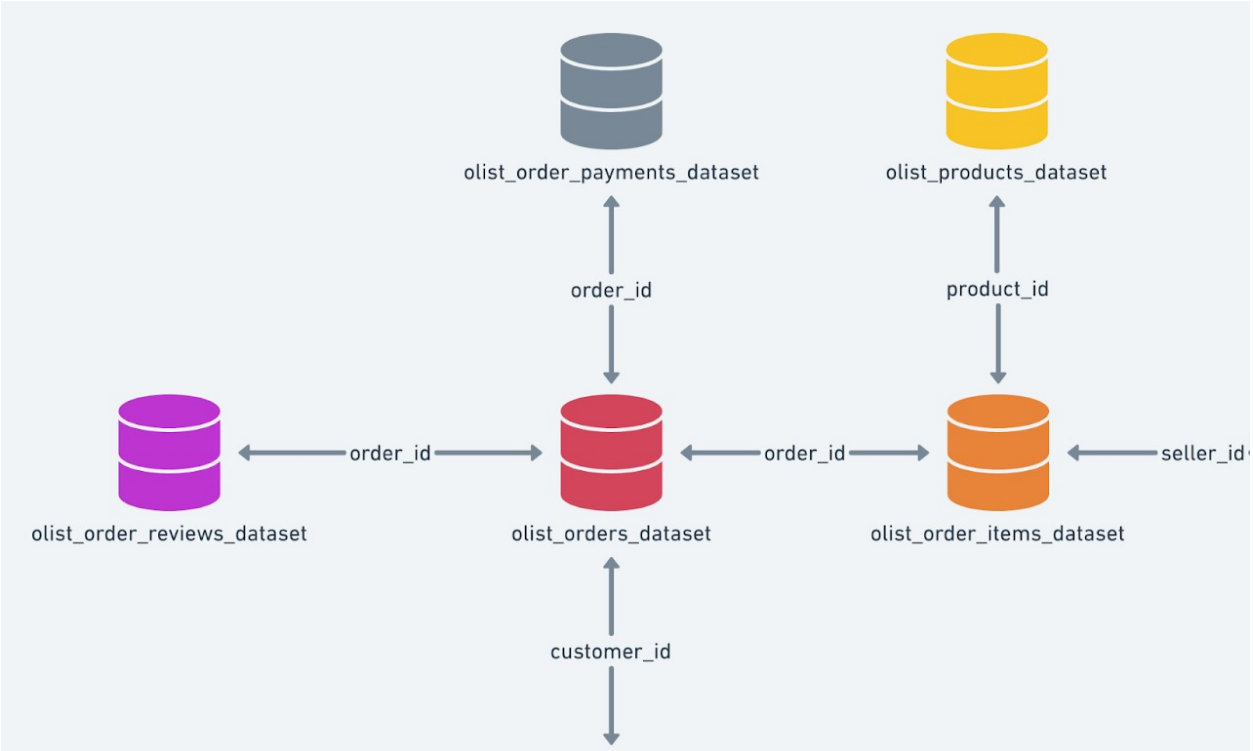
The **reviews.csv** contain following features:

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 relationship between datasets:



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

a. Get number of rows in the data

1. For Customers Table :

```
1  select count(*) from `scaler-dsml-target-sql.target11.custo
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAI

Row | f0

2. **For GeoLocation Table:**

```
1  select count(*) from `scaler-dsml-target-sql.target11
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION

Row	f0

3. **For order_items table:**

```
1  select count(*) from `scaler-dsml-target-sql.target1
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTIO

Row	f0

4. **For order_reviews table:**

```
1  select count(*) from `scaler-dsml-target-sql.target11.o
```

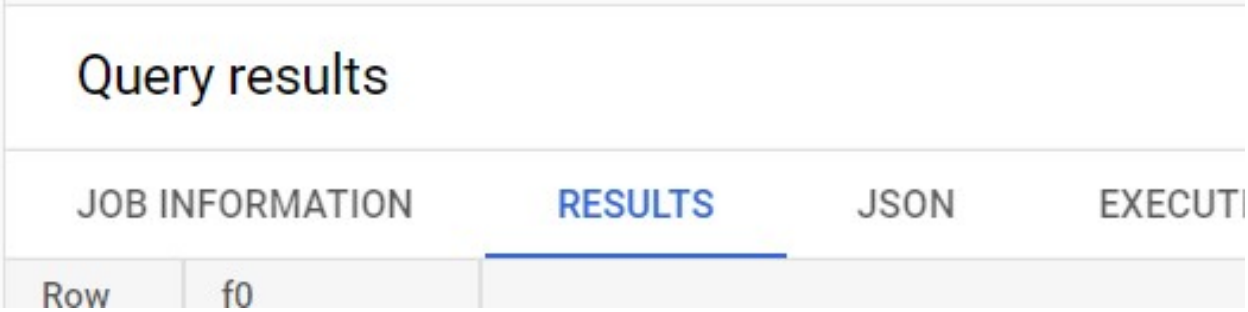
Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DE

Row	f0

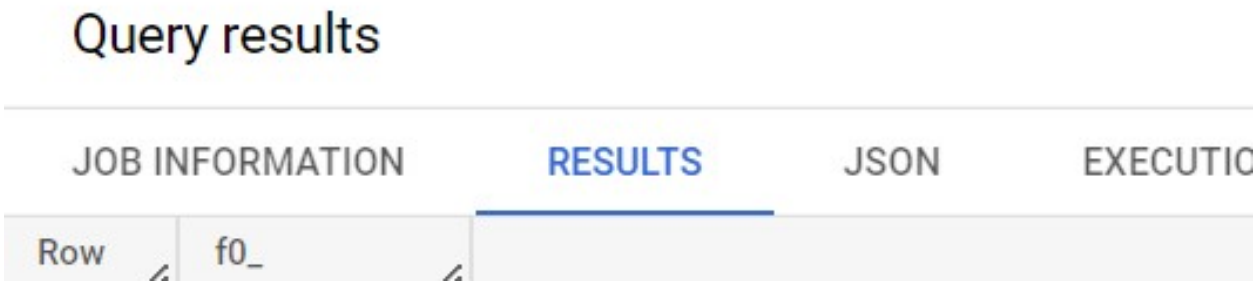
5. **For orders table :**

```
1  select count(*) from `scaler-dsml-target-sql.target
```



6. **For payments table:**

```
1  select count(*) from `scaler-dsml-target-sql.target1
```



7. **For products table :**

```
1  select count(*) from `scaler-dsml-target-sql.target
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTI

Row	f0

8. **For sellers table :**

```
1  select count(*) from `scaler-dsml-target-sql.targe
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUT

Row	f0_

b. Number of null or missing values in a column

1. For Customers Table :

```sql
select count(*)-count(customer_id) as missing_customer_ids,
       count(*)-count(customer_unique_id) as missing_customer_unique_ids,
       count(*)-count(customer_zip_code_prefix) as missing_customer_zip_code_prefix,
       count(*)-count(customer_city) as missing_customer_city,
       count(*)-count(customer_state) as missing_customer_state,

    from `scaler-dsml-target-sql.target11.customers`
```

Query results SAVE RESULTS

JOB INFORMATION RESULTS JSON EXECUTION DETAILS

2. For GeoLocation Table:

```sql
select count(*)-count(geolocation_zip_code_prefix) as missing_location_zip_code_prefix,
       count(*)-count(geolocation_lat) as missing_geolocation_lat,
       count(*)-count(geolocation_lng) as missing_geolocation_lng,
       count(*)-count(geolocation_city) as missing_geolocation_city,
       count(*)-count(geolocation_state) as missing_geolocation_state

    from `scaler-dsml-target-sql.target11.geolocation`
```

Query results SAVE RESULTS

JOB INFORMATION RESULTS JSON EXECUTION DETAILS

3. For Order_items Table:

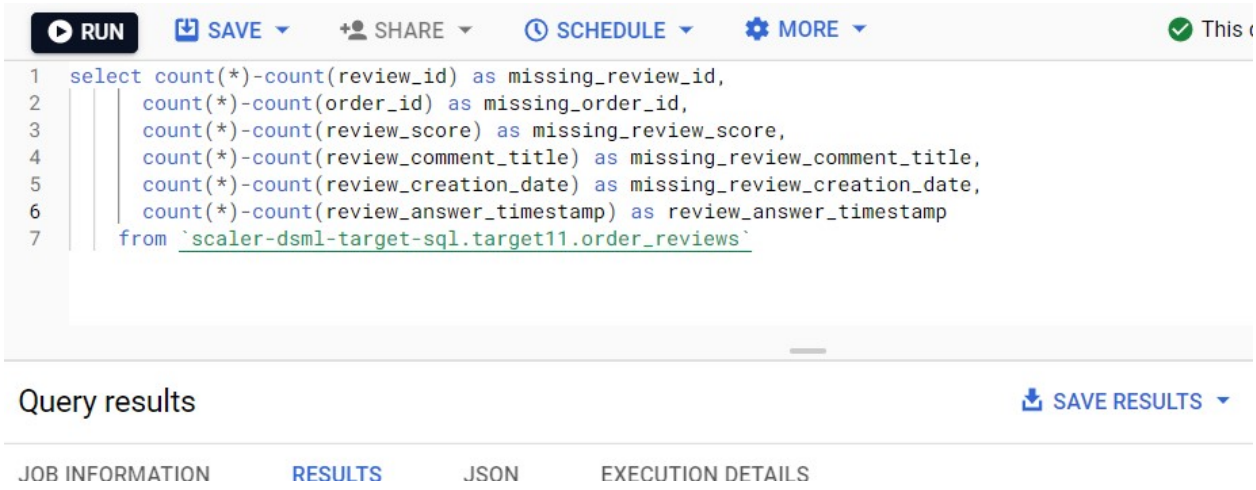
RUN SAVE SHARE SCHEDULE MORE This

```sql
select count(*)-count(order_id) as missing_order_id,
       count(*)-count(order_item_id) as missing_order_item_id,
       count(*)-count(product_id) as missing_product_id,
       count(*)-count(seller_id) as missing_seller_id,
       count(*)-count(shipping_limit_date) as missing_shipping_limit_date,
       count(*)-count(price) as missing_price,
       count(*)-count(freight_value) as missing_freight_valuee

    from `scaler-dsml-target-sql.target11.order_items`
```

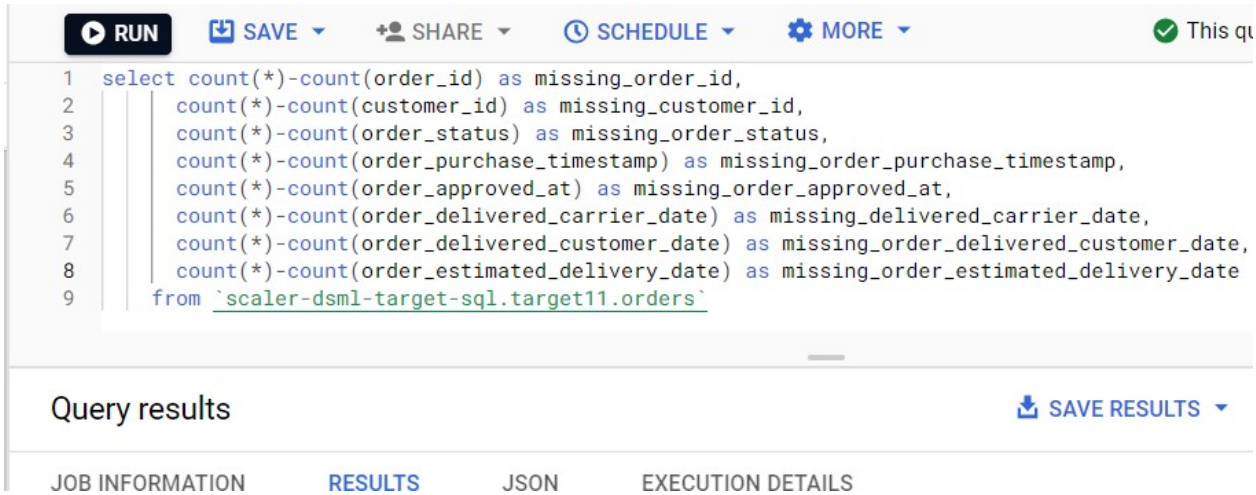
Query results SAVE RESULTS

4. For Order_reviews Table :



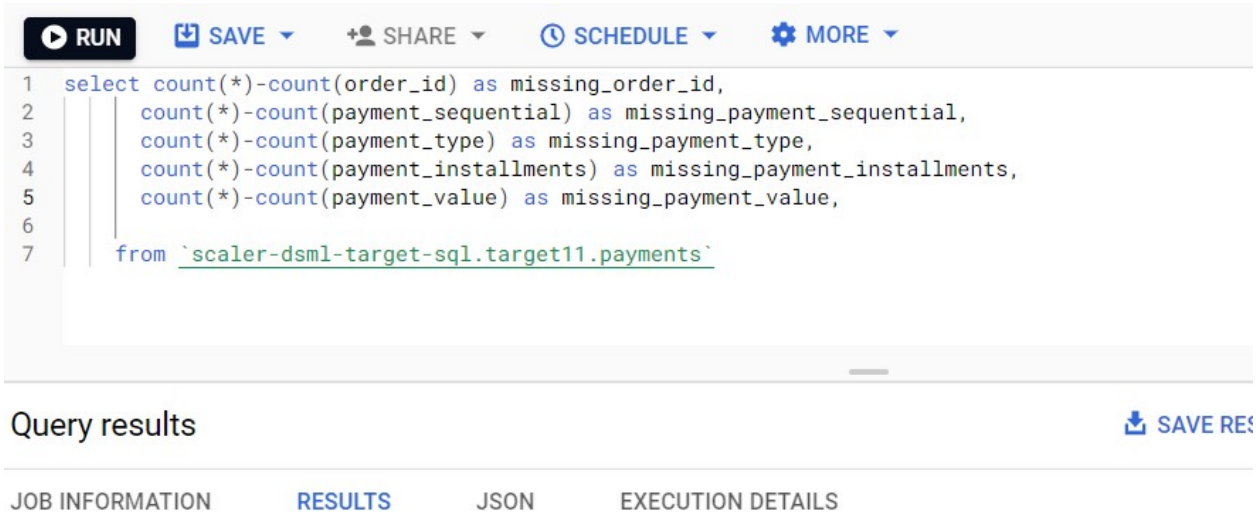
There are 87675 missing values of Review_comment_title

5. For Orders table :

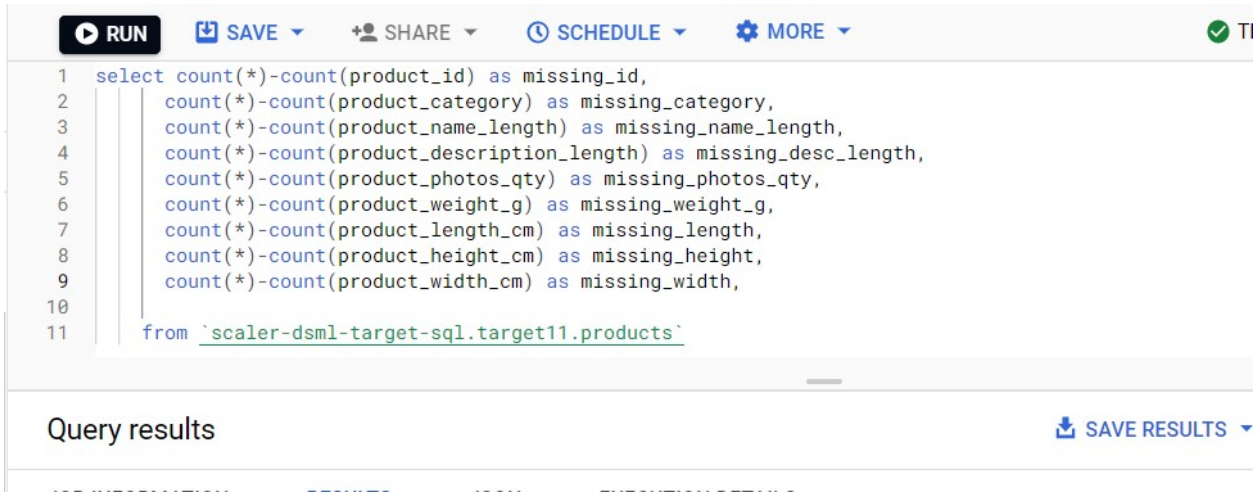


There are 160 missing values in order_approved_at column, 1783 missing values in order_delivered_carrier_date and 2965 missing values in order_estimated_delivery_date

6. For Payments table :

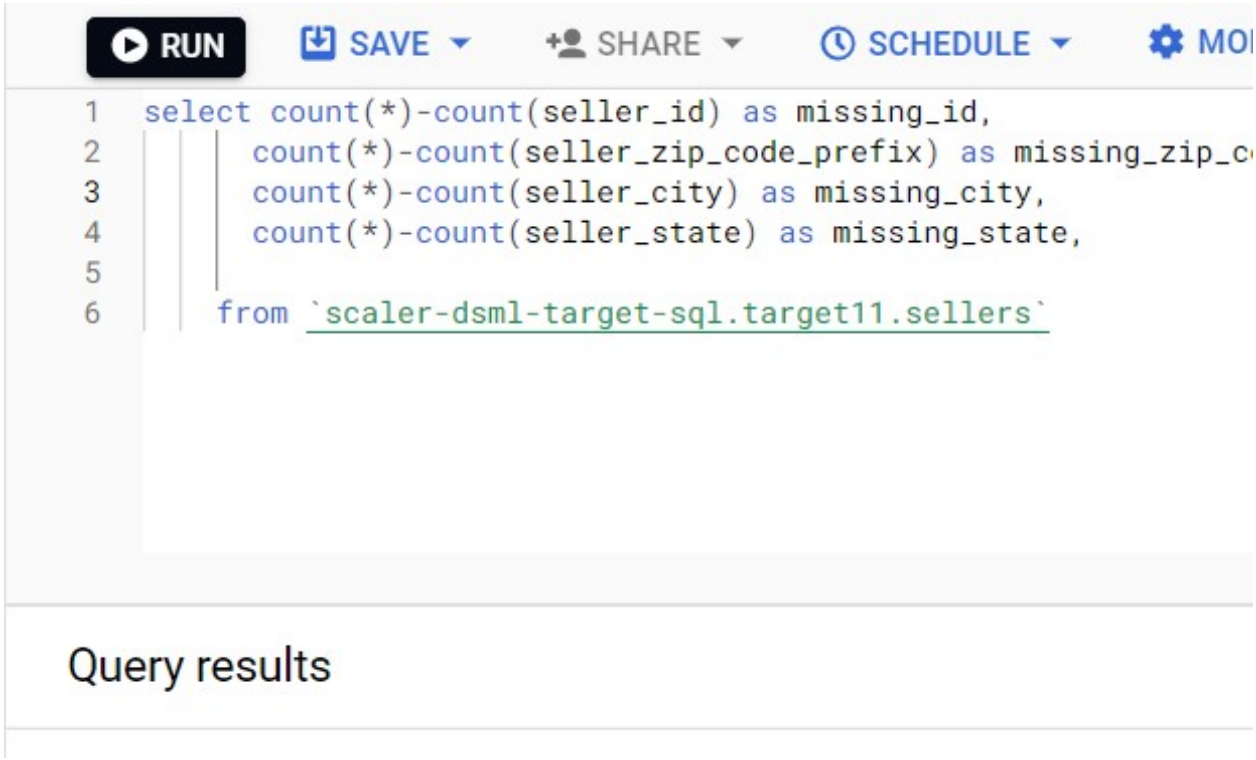


7. For Products table :



There are 610 missing values in each columns of product_category, product_name_length, product_description_length, product_photos_qty and 2 missing values in each of product_length_cm, product_height_cm and product_width_cm.

8. **For Sellers table :**



c. Data type of columns in a table

RUN SAVE SHARE SCHEDULE MORE

```
1  SELECT table_name,column_name,ordinal_position,data_type FROM target11.INFORMATION
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	table_name	column_name	ordinal_posi...	data_type
1	order_items	order_id	1	STRING
2	order_items	order_item_id	2	INT64
3	order_items	product_id	3	STRING
4	order_items	seller_id	4	STRING
5	order_items	shipping_limit_date	5	TIMESTAMP
6	order_items	price	6	FLOAT64
11	sellers	seller_state	4	STRIN
12	geolocation	geolocation_zip_code_prefix	1	INT64
13	geolocation	geolocation_lat	2	FLOAT
14	geolocation	geolocation_lng	3	FLOAT
15	geolocation	geolocation_city	4	STRIN
16	geolocation	geolocation_state	5	STRIN
17	products	product_id	1	STRIN
18	products	product_category	2	STRIN

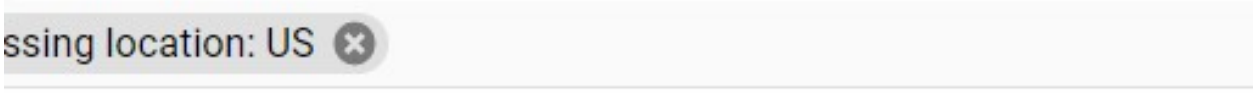
21	products	product_photos_qty	5	INT64
22	products	product_weight_g	6	INT64
23	products	product_length_cm	7	INT64
24	products	product_height_cm	8	INT64
25	products	product_width_cm	9	INT64
26	orders	order_id	1	STRIN
27	orders	customer_id	2	STRIN
28	orders	order_status	3	STRIN

31	orders	order_delivered_carrier_date	6	TIMEST
32	orders	order_delivered_customer_date	7	TIMEST
33	orders	order_estimated_delivery_date	8	TIMEST
34	payments	order_id	1	STRING
35	payments	payment_sequential	2	INT64
36	payments	payment_type	3	STRING
37	payments	payment_installments	4	INT64
38	payments	payment_value	5	FLOAT6

41	customers	customer_zip_code_prefix	3	INT64
42	customers	customer_city	4	STRING
43	customers	customer_state	5	STRING
44	order_reviews	review_id	1	STRING
45	order_reviews	order_id	2	STRING
46	order_reviews	review_score	3	INT64
47	order_reviews	review_comment_title	4	STRING

d. Get the time period for which the data is given

```
select
  concat(
   EXTRACT(MONTH FROM
  (SELECT MIN(o.order_purchase_timestamp)
  FROM `scaler-dsml-target-sql.target11.orders` o)),'/',
  EXTRACT(YEAR FROM
  (SELECT MIN(o.order_purchase_timestamp)
  FROM `scaler-dsml-target-sql.target11.orders` o)),' - '
   EXTRACT(MONTH FROM
  (SELECT MAX(o.order_purchase_timestamp)
  FROM `scaler-dsml-target-sql.target11.orders` o)),'/',
  EXTRACT(YEAR FROM
  (SELECT MAX(o.order_purchase_timestamp)
  FROM `scaler-dsml-target-sql.target11.orders` o))
  )
```



e. Number of cities in our dataset

```
1  select count(distinct geolocation_city) as no_of_cities from `scaler-dsml-target-sq
```

rocessing location: US

Query results

JOB INFORMATION RESULTS JSON EXECUTION DETAILS

f. Number of states in our dataset

```
1  select count(distinct geolocation_state) as no_of_states from `scaler-dsml-target-sql.
```

Processing location: US

Query results

JOB INFORMATION RESULTS JSON EXECUTION DETAILS

In-depth Exploration:

a. How many orders do we have for each order status?

```
1  select order_status,count(order_status) as no_of_o
2  from `scaler-dsml-target-sql.target11.orders` o
3  group by order_status
```

Processing location: US

Query results

JOB INFORMATION | RESULTS | JSON | EXECUT

ow	order_status	no_of_orders
1	created	5
2	shipped	1107
3	approved	2
4	canceled	625
5	invoiced	314

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

```
select *,lag(Num_of_orders_in_year,1) over (order by year) as prev_yearr_orders,
(Num_of_orders_in_year-lag(Num_of_orders_in_year,1) over (order by year)) as differenc
from(
  SELECT Extract(year from order_purchase_timestamp) as year,count(order_id) as Num_of
  FROM `scaler-dsml-target-sql.target11.orders`
  group by year order by year)
order by year
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	year	Num_of_ord...	prev_yearr_...	difference_i...
1	2016	329	null	null

c. On what day of week brazilians customers tend to do online purchasing?

```
select * from
(select day,count(day) as no_of_orders_placed fro
(select FORMAT_DATE("%a",order_purchase_timestamp
from `scaler-dsml-target-sql.target11.orders` o)
group by day
order by count(day) desc
) as X
limit 1
```

Processing location: US

Query results

JOB INFORMATION | RESULTS | JSON | EXECU

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

```sql
select
case
when hour>=4 and hour<=6 then 'Dawn'
when hour>6 and hour<12 then 'Morning'
when hour>=12 and hour<16 then 'Afternoon'
when hour>=16 and hour<=23 then 'Night'
else 'Midnight'
end as Part_of_Day,count(order_id) as no_of_orders

from (SELECT Extract(hour from order_purchase_timestamp) as hour,order_id FROM `scaler-dsml-target-sc

group by Part_of_Day order by no_of_orders desc
limit 1
```

Processing location: US

Query results

e. **Feature Extraction**: Through order_purchase_timestamp in "orders" dataset extract

i. order_purchase_year
ii. order_purchase_month
iii. order_purchase_date
iv. order_purchase_day
v. order_purchase_dayofweek
vi. order_purchase_dayofweek_name
vii. order_purchase_hour
viii. order_purchase_time_day

```sql
select
  order_purchase_timestamp,
   EXTRACT(year from order_purchase_timestamp) as order_purchase_year,
  EXTRACT(month from order_purchase_timestamp) as order_purchase_month,
EXTRACT(date from order_purchase_timestamp) as order_purchase_date,
EXTRACT(day from order_purchase_timestamp) as order_purchase_day,
EXTRACT(dayofweek from order_purchase_timestamp) as order_purchase_dayofweek,
FORMAT_DATE("%a", order_purchase_timestamp) as order_purchase_dayofweek_name,
X.hour as order_purchase_hour,
case
when hour>=4 and hour<=6 then 'Dawn'
when hour>6 and hour<12 then 'Morning'
when hour>=12 and hour<16 then 'Afternoon'
when hour>=16 and hour<21 then 'Evening'
when hour>=21 or hour=0 then 'Night'
else 'Midnight'
end as order_purcahse_time_day
```

Query results

SAVE RESULT

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	order_purchase_timestamp	order_purch...	order_purch...	order_purch...	order_purch...	order_purch...	order_purchas...	order_
1	2017-12-01 11:09:24 UTC	2017	12	2017-12-01	1	6	Fri	
2	2017-12-01 14:38:00 UTC	2017	12	2017-12-01	1	6	Fri	
3	2017-08-01 20:26:19 UTC	2017	8	2017-08-01	1	3	Tue	
4	2018-04-01 11:30:07 UTC	2018	4	2018-04-01	1	1	Sun	
5	2017-07-01 16:48:42 UTC	2017	7	2017-07-01	1	7	Sat	
6	2017-04-01 18:26:29 UTC	2017	4	2017-04-01	1	7	Sat	
7	2018-03-01 18:19:28 UTC	2018	3	2018-03-01	1	5	Thu	

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

a. Get month on month orders by region

```
1  select
2  c.customer_state, EXTRACT(YEAR from o.order_purchase_timestamp) as year, EXTRACT(MONTH from o.order_purchase_timestamp)
   no_of_orders
3  from `scaler-dsml-target-sql.target11.orders`o join `scaler-dsml-target-sql.target11.customers` c
4  on o.customer_id = c.customer_id
5  group by c.customer_state, year, month
6  order by c.customer_state,year,month
```

Query results

SAVE RESULTS

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	year	month	no_of_orders
1	AC	2017	1	2
2	AC	2017	2	3
3	AC	2017	3	2
4	AC	2017	4	5
5	AC	2017	5	8
6	AC	2017	6	4

b. Total of customer orders by state

```sql
select
c.customer_state, count(order_id) as no_of_orders
from `scaler-dsml-target-sql.target11.orders`o join `scaler-dsml-target-sql.target11.customers`
on o.customer_id = c.customer_id
group by c.customer_state
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	no_of_orders
1	RJ	12852
2	RS	5466
3	SP	41746
4	DF	2140
5	PR	5045
6	MT	907
7	MA	747

c. Top 10 brazilian cities most no. of orders

```sql
select
c.customer_city, count(order_id) as no_of_orders
from `scaler-dsml-target-sql.target11.orders`o join `scaler-dsml-target-sql.ta
on o.customer_id = c.customer_id
group by c.customer_city
order by no_of_orders desc limit 10
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_city	no_of_orders
1	sao paulo	15540
2	rio de janeiro	6882
3	belo horizonte	2773
4	brasilia	2131
5	curitiba	1521
6	campinas	1444

d. How are customers distributed in Brazil

```
1  select
2  c.customer_state,c.customer_city, count(order_id) as no_of_orders
3  from `scaler-dsml-target-sql.target11.orders`o join `scaler-dsml-target-sql.ta
4  on o.customer_id = c.customer_id
5  group by c.customer_state,c.customer_city
6  order by c.customer_state, c.customer_city
7
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	customer_city	no_of_orders
1	AC	brasileia	1
2	AC	cruzeiro do sul	3
3	AC	epitaciolandia	1
4	AC	manoel urbano	1
5	AC	porto acre	1
6	AC	rio branco	70

e. City wise number of unique customers

```
1  select
2  c.customer_city, count(c.customer_unique_id) as no_of_custo
3  from `scaler-dsml-target-sql.target11.customers` c
4  group by c.customer_city
5  order by no_of_customers desc
6
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAI

Row	customer_city	no_of_custo...
1	sao paulo	15540
2	rio de janeiro	6882
3	belo horizonte	2773
4	brasilia	2131
5	curitiba	1521
6	campinas	1444

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

Step 1: Using CTE

1. “order_items” + “order” joined on order id where order_purchase timestamp is already divided into month & year

2. Group data by year and month, aggregation count(order_id), sum(price), sum(freight_value)

3. Create new columns:

price_per_order = sum(price) / count(order_id)

freight_per_order= sum(freight_value) / count(order_id)

Step 2: Answer the following questions:

a. Total amount sold in 2017 between Jan to August

```sql
with cte as (

 select

  EXTRACT(year from orders.order_purchase_timestamp) as year,
  EXTRACT(month from orders.order_purchase_timestamp) as month,
  count(orders.order_id) as no_of_orders, sum(items.price) as total_price,sum(items.freight_value)
  sum(items.price)/count(orders.order_id) as price_per_order,
  sum(items.freight_value) as fright_per_order
 from `scaler-dsml-target-sql.target11.order_items` items join `scaler-dsml-target-sql.target11.or
 on items.order_id = orders.order_id
 group by year,month

)
select sum(total_price) from cte where year=2017 and month between 1 and 8
```

Query results

b. Total amount sold in 2018 between Jan to august

```sql
with cte as (

 select

  EXTRACT(year from orders.order_purchase_timestamp) as year,
  EXTRACT(month from orders.order_purchase_timestamp) as month,
  count(orders.order_id) as no_of_orders, sum(items.price) as total_price,sum(items.freight_value) a
  sum(items.price)/count(orders.order_id) as price_per_order,
  sum(items.freight_value) as fright_per_order
 from `scaler-dsml-target-sql.target11.order_items` items join `scaler-dsml-target-sql.target11.orde
 on items.order_id = orders.order_id
 group by year,month

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

Query results

c. % increase from 2017 to 2018

```sql
with cte as (
 select
  EXTRACT(year from orders.order_purchase_timestamp) as year,
  EXTRACT(month from orders.order_purchase_timestamp) as month,
  count(orders.order_id) as no_of_orders, sum(items.price) as total_price,sum(items.freight_value) as total_freight_val
  sum(items.price)/count(orders.order_id) as price_per_order,
  sum(items.freight_value) as fright_per_order
 from `scaler-dsml-target-sql.target11.order_items` items join `scaler-dsml-target-sql.target11.orders` orders
 on items.order_id = orders.order_id
 group by year,month
)
select X.year,X.price_this_year,X.price_prev_year,X.rate_of_increase
from
(select cte.year, round(sum(cte.total_price),2) as price_this_year,
round((lag(sum(cte.total_price),1) over (order by cte.year )),2) as price_prev_year,
round(((sum(cte.total_price)-(lag(sum(cte.total_price),1) over (order by cte.year )))/(lag(sum(cte.total_price),1) over
as rate_of_increase
from cte
group by cte.year
order by cte.year desc) as X
where X.year=2018
```

Query results SAVE RESULTS

Step 3: Join (orders+order_items) table from previous step with "customers" table on Customer_id and find:

a. Mean & Sum of price by customer state

```sql
with cte as (
 select
  customers.customer_state,
  count(orders.order_id) as no_of_orders, sum(items.price) as total_price,sum(items.freight_value) as total_freight_
  sum(items.price)/count(orders.order_id) as price_per_order,
  sum(items.freight_value) as freight_per_order
 from `scaler-dsml-target-sql.target11.order_items` items join `scaler-dsml-target-sql.target11.orders` orders
 on items.order_id = orders.order_id join `scaler-dsml-target-sql.target11.customers` customers on orders.customer_i
 group by customers.customer_state
)

select cte.customer_state,cte.total_price,cte.price_per_order as avg_price from cte
```

Query results SAVE RESULT

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	total_price	avg_price
1	MT	156453.529...	148.297184...
2	MA	119648.219...	145.204150...
3	AL	80314.81	180.889211...

b. Mean & Sum of freight value by customer state

```sql
with cte as (
 select
  customers.customer_state,
  count(orders.order_id) as no_of_orders, sum(items.price) as total_price,sum(items.freight_value) as total_freigh
  sum(items.price)/count(orders.order_id) as price_per_order,
  sum(items.freight_value)/count(orders.order_id) as freight_per_order
 from `scaler-dsml-target-sql.target11.order_items` items join `scaler-dsml-target-sql.target11.orders` orders
 on items.order_id = orders.order_id join `scaler-dsml-target-sql.target11.customers` customers on orders.customer
 group by customers.customer_state
)

select cte.customer_state,cte.total_freight_value,round(cte.freight_per_order,2) as avg_freight_value from cte
```

Query results SAVE RESU

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	total_freight...	avg_freight_...
1	MT	29715.4300...	28.1662843...
2	MA	31523.7700...	38.2570024...
3	AL	15914.5899	35.8436711

5. Analysis on sales, freight and delivery time

1. Calculating days between purchasing, delivering and estimated delivery

2. Create columns:

time_to_delivery = order_purchase_timestamp-order_delivered_customer_date

diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date

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

4. Sort the data to get the following:

a. Top 5 states with highest/lowest average freight value

States with Highest freight value :

```sql
select
X.customer_state, avg(X.freight_value) as avg_freight_value
from (select
c.customer_state,
DATE_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp,DAY) as time_to_delivery,
DATE_DIFF(o.order_estimated_delivery_date,o.order_delivered_customer_date, DAY) as diff_estimated_delivery,
oi.freight_value

from `scaler-dsml-target-sql.target11.orders` o join `scaler-dsml-target-sql.target11.order_items` oi
on o.order_id = oi.order_id
join `scaler-dsml-target-sql.target11.customers` c
on o.customer_id = c.customer_id
where o.order_delivered_customer_date is not null and o.order_purchase_timestamp is not null and o.order_estimate
) as X
group by X.customer_state
order by avg_freight_value desc limit 5
```

Query results

SAVE RES

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	avg_freight_...
1	PB	43.0916894...

States with Lowest freight Value :

RUN | SAVE | SHARE | SCHEDULE | MORE

```sql
select
X.customer_state, avg(X.freight_value) as avg_freight_value
from (select
c.customer_state,
DATE_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp,DAY) as time_to_delivery,
DATE_DIFF(o.order_estimated_delivery_date,o.order_delivered_customer_date, DAY) as diff_estimated_delivery,
oi.freight_value

from `scaler-dsml-target-sql.target11.orders` o join `scaler-dsml-target-sql.target11.order_items` oi
on o.order_id = oi.order_id
join `scaler-dsml-target-sql.target11.customers` c
on o.customer_id = c.customer_id
where o.order_delivered_customer_date is not null and o.order_purchase_timestamp is not null and o.order_estima
) as X
group by X.customer_state
order by avg_freight_value asc limit 5
```

Query results

SAVE R

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	avg_freight_...
1	SP	15.1149940...

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

States with highest avg delivery time :

```sql
select
X.customer_state, avg(X.time_to_delivery) as avg_delivery_time
from (select
c.customer_state,
DATE_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp,DAY) as time_to_delivery,
DATE_DIFF(o.order_estimated_delivery_date,o.order_delivered_customer_date, DAY) as diff_estimated_delivery,
oi.freight_value
from `scaler-dsml-target-sql.target11.orders` o join `scaler-dsml-target-sql.target11.order_items` oi
on o.order_id = oi.order_id
join `scaler-dsml-target-sql.target11.customers` c
on o.customer_id = c.customer_id
where o.order_delivered_customer_date is not null and o.order_purchase_timestamp is not null and o.order_estimated_d
) as X
group by X.customer_state
order by avg_delivery_time desc limit 5
```

Query results

SAVE RESULT

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	avg_delivery...
1	RR	27.8260869...

```sql
select
X.customer_state, avg(X.time_to_delivery) as avg_delivery_time
from (select
c.customer_state,
DATE_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp,DAY) as time_to_delivery,
DATE_DIFF(o.order_estimated_delivery_date,o.order_delivered_customer_date, DAY) as diff_estimated_delivery,
oi.freight_value
from `scaler-dsml-target-sql.target11.orders` o join `scaler-dsml-target-sql.target11.order_items` oi
on o.order_id = oi.order_id
join `scaler-dsml-target-sql.target11.customers` c
on o.customer_id = c.customer_id
where o.order_delivered_customer_date is not null and o.order_purchase_timestamp is not null and o.order_estimated_
) as X
group by X.customer_state
order by avg_delivery_time asc limit 5
```

Query results

SAVE RESU

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	avg_delivery...
1	SP	8.25960855...

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

states with delivery fast compared to estimated delivery :

```
select
X.customer_state, avg(X.time_to_delivery-X.diff_estimated_delivery) as avg_delay
from (select
c.customer_state,
DATE_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp,DAY) as time_to_delivery,
DATE_DIFF(o.order_estimated_delivery_date,o.order_delivered_customer_date, DAY) as diff_estimated_delivery,
oi.freight_value
from `scaler-dsml-target-sql.target11.orders` o join `scaler-dsml-target-sql.target11.order_items` oi
on o.order_id = oi.order_id
join `scaler-dsml-target-sql.target11.customers` c
on o.customer_id = c.customer_id
where o.order_delivered_customer_date is not null and o.order_purchase_timestamp is not null and o.order_estimated_
) as X
group by X.customer_state
order by avg_delay asc limit 5
```

Query results

SAVE RESUL

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	avg_delay
1	SP	-2.0059858...

states with late delivery compared to estimated delivery :

```
select
X.customer_state, avg(X.time_to_delivery-X.diff_estimated_delivery) as avg_delay
from (select
c.customer_state,
DATE_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp,DAY) as time_to_delivery,
DATE_DIFF(o.order_estimated_delivery_date,o.order_delivered_customer_date, DAY) as diff_estimated_delivery,
oi.freight_value
from `scaler-dsml-target-sql.target11.orders` o join `scaler-dsml-target-sql.target11.order_items` oi
on o.order_id = oi.order_id
join `scaler-dsml-target-sql.target11.customers` c
on o.customer_id = c.customer_id
where o.order_delivered_customer_date is not null and o.order_purchase_timestamp is not null and o.order_estimatec
) as X
group by X.customer_state
order by avg_delay desc limit 5
```

Query results

SAVE RES

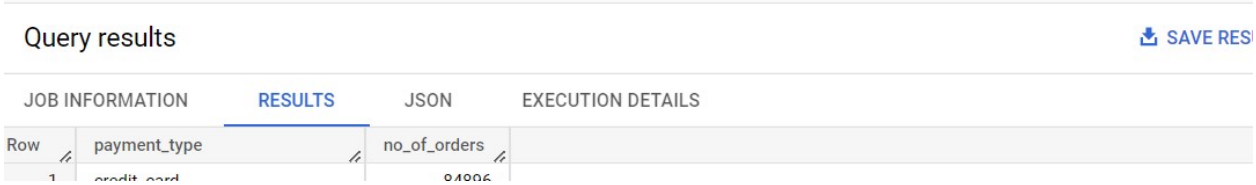
JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	avg_delay
1	AL	16.0163934...

6. Payment type analysis: Join “payments” dataset with the existing data on order_id

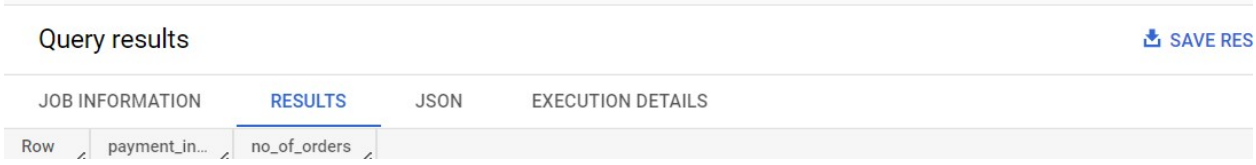
a. Count of orders for different payment types

```sql
select
X.payment_type, count(X.order_id) as no_of_orders
from (select
c.customer_state,
p.payment_type,o.order_id,p.payment_installments,
DATE_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp,DAY) as time_to_delivery,
DATE_DIFF(o.order_estimated_delivery_date,o.order_delivered_customer_date, DAY) as diff_estimated_delivery,
oi.freight_value
from `scaler-dsml-target-sql.target11.orders` o join `scaler-dsml-target-sql.target11.order_items` oi
on o.order_id = oi.order_id
join `scaler-dsml-target-sql.target11.customers` c
on o.customer_id = c.customer_id
join `scaler-dsml-target-sql.target11.payments` p
on p.order_id = o.order_id
where o.order_delivered_customer_date is not null and o.order_purchase_timestamp is not null and o.order_estimated
) as X
```



b. Distribution of payment installments and count of orders

```sql
select
X.payment_installments, count(X.order_id) as no_of_orders
from (select
c.customer_state,p.payment_type,o.order_id,p.payment_installments,
DATE_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp,DAY) as time_to_delivery,
DATE_DIFF(o.order_estimated_delivery_date,o.order_delivered_customer_date, DAY) as diff_estimated_delivery,
oi.freight_value
from `scaler-dsml-target-sql.target11.orders` o join `scaler-dsml-target-sql.target11.order_items` oi
on o.order_id = oi.order_id
join `scaler-dsml-target-sql.target11.customers` c
on o.customer_id = c.customer_id
join `scaler-dsml-target-sql.target11.payments` p
on p.order_id = o.order_id
where o.order_delivered_customer_date is not null and o.order_purchase_timestamp is not null and o.order_estimated
) as X
group by X.payment_installments order by no_of_orders desc
```



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

```sql
select
X.year,X.MONTH, X.payment_type,count(X.order_id) as no_of_orders
from (select
c.customer_state,p.payment_type,o.order_id,p.payment_installments,
EXTRACT(MONTH from o.order_purchase_timestamp) as month,
EXTRACT(YEAR from o.order_purchase_timestamp) as year,
oi.freight_value
from `scaler-dsml-target-sql.target11.orders` o join `scaler-dsml-target-sql.target11.order_items` oi
on o.order_id = oi.order_id
join `scaler-dsml-target-sql.target11.customers` c
on o.customer_id = c.customer_id
join `scaler-dsml-target-sql.target11.payments` p
on p.order_id = o.order_id
where o.order_delivered_customer_date is not null and o.order_purchase_timestamp is not null and o.order_estimate
) as X
group by X.year,X.month,X.payment_type order by X.year,X.month asc
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	year	MONTH	payment_type	no_of_orders
1	2016	10	credit_card	253
2	2016	10	UPI	61
3	2016	10	voucher	20
4	2016	10	debit_card	2
5	2016	12	credit_card	1
6	2017	1	credit_card	659
7	2017	1	voucher	68
8	2017	1	UPI	226
9	2017	1	debit_card	10

Insights :

- There are 99441 orders placed in the given data
- There are 32951 different types of products which the customers ordered
- There are 87675 reviews where title is missing for review_comments.
- There are 160 missing values in order_approved_at, 1783 missing values in order_delivered_delivered_carrier_date and 2965 missing values in order_estimated_delivery_date in Orders table
- There are 610 missing values in each columns of product _category, product name product description_length, product photos qty and 2 missing values in each of product length_ cm, product_height _cm and product width_cm.
- The data is between September 2016 and October 2018.
- There are 27 states and 8011 cities in the data

- There is growing trend in ecommerce usage over the years.

```
select *,lag(Num_of_orders_in_year,1) over (order by year) as prev_yearr_orders,
(Num_of_orders_in_year-lag(Num_of_orders_in_year,1) over (order by year)) as differenc
from(
  SELECT Extract(year from order_purchase_timestamp) as year,count(order_id) as Num_of
  FROM `scaler-dsml-target-sql.target11.orders`
  group by year order by year)
order by year
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	year	Num_of_ord...	prev_yearr_...	difference_i...
1	2016	329	null	null

- More number of orders are placed on Monday

```
select * from
(select day,count(day) as no_of_orders_placed fro
(select FORMAT_DATE("%a",order_purchase_timestamp
from `scaler-dsml-target-sql.target11.orders` o)
group by day
order by count(day) desc
) as X
limit 1
```

Processing location: US

Query results

JOB INFORMATION | RESULTS | JSON | EXECU

- The states Sp, RJ and Mg have more number of orders while the states AC,AP and AR have least number of orders
- The “sau palo” city has more number of orders
- There is 19.99% of revenue increase from 2017 to 2018 even when the data of 2018 is till October only
- The states PB and SP have highest and lowest avg freight rates respectively.
- The states RR and SP have highest and lowest avg delivery times.
- SP state has the lowest delay where as AL state has highest order delay between estimated and delivered time

- More number of orders are placed through credit card

```
select
X.payment_type, count(X.order_id) as no_of_orders
from (select
c.customer_state,
p.payment_type,o.order_id,p.payment_installments,
DATE_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp,DAY) as time_to_delivery,
DATE_DIFF(o.order_estimated_delivery_date,o.order_delivered_customer_date, DAY) as diff_estimated_delivery,
oi.freight_value
from `scaler-dsml-target-sql.target11.orders` o join `scaler-dsml-target-sql.target11.order_items` oi
on o.order_id = oi.order_id
join `scaler-dsml-target-sql.target11.customers` c
on o.customer_id = c.customer_id
join `scaler-dsml-target-sql.target11.payments` p
on p.order_id = o.order_id
where o.order_delivered_customer_date is not null and o.order_purchase_timestamp is not null and o.order_estimated
) as X
```

Query results

SAVE RES

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	payment_type	no_of_orders
1	credit_card	84896

- More number of orders are paid in one installment

```
select
X.payment_type, count(X.order_id) as no_of_orders
from (select
c.customer_state,
p.payment_type,o.order_id,p.payment_installments,
DATE_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp,DAY) as time_to_delivery,
DATE_DIFF(o.order_estimated_delivery_date,o.order_delivered_customer_date, DAY) as diff_estimated_delivery,
oi.freight_value
from `scaler-dsml-target-sql.target11.orders` o join `scaler-dsml-target-sql.target11.order_items` oi
on o.order_id = oi.order_id
join `scaler-dsml-target-sql.target11.customers` c
on o.customer_id = c.customer_id
join `scaler-dsml-target-sql.target11.payments` p
on p.order_id = o.order_id
where o.order_delivered_customer_date is not null and o.order_purchase_timestamp is not null and o.order_estimated
) as X
```

Query results

SAVE RES

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	payment_type	no_of_orders
1	credit_card	84896

Recommendations :

- Every time an order is placed by a customer, each instance is treated as different customer but there will be a high possibility that there are repeated customers. If each instance of order is treated as different customer, it is difficult to analyze the customer behavior.
- There are 2965 missing values in estimated_delivery_time which is not a good experience for a customer to trace or expect the order
- The data is available for only four months i.e.(September to December) of data in 2016 and only ten months i.e.(January to October) in 2018. It's better to have data for whole year so that we can have proper estimation of trends over years.
- The states AC, AP and AR has least number of orders. We can attract more customers by having more campaigns and offers specific to those states.
- The RR state has more avg time taken for delivery. We can decrease the time by having more physical stores and delivery partners.
- The AL state has more difference in delay between the expected delivery and actual delivery time. Need to trace out the reasons for that.
- More number of orders are placed in single installments. We can have more number of orders if we offer no cost EMI