Retailer Analysis

Problem Statement

Analyzing 100k orders from 2016 to 2018 made at an American big box department store chain in Brazil by viewing orders from multiple dimensions such as order status, price, payment and freight performance to customer locations, product attributes and reviews written by customers for providing insights and recommendations to help improve the business.

The data is made available in 8 different tables namely customers, sellers, geolocation, products, orders, order_items, payments and reviews.

I. Initial Exploration of Dataset

1. Customers

	customer_id	customer_unique_id	customer_zip_code_prefix	customer_city	customer_state
1	0735e7e4298a2ebbb46649346570476a	fcb003b1bdc0df64b4d065d9bb94f8c4	59650	acu	RN
2	903b3d86e3990db01619a4ebe3edef4e	46824822b15da44e983b021d0e945379	59650	acu	RN
3	38c97666e962d4fea7fd6a83e69f20cd	b6108acc674ae5c99e29adc1047d1049	59650	acu	RN

➢ Number of rows in the data

```
SELECT COUNT(*) FROM `scaler-sql.retailer.customers`
```

- 99441

➢
```
SELECT COUNT(DISTINCT customer_unique_id) FROM `scaler-sql.retailer.customers`
```

- 96096

➢ Number of null or missing values

```
SELECT
 COUNT(*) - COUNT(customer_id) id_null_count,
 COUNT(*) - COUNT(customer_unique_id) unique_id_null_count,
 COUNT(*) - COUNT(customer_zip_code_prefix) zip_null_count,
 COUNT(*) - COUNT(customer_city) city_null_count,
 COUNT(*) - COUNT(customer_state) state_null_count
FROM
 `scaler-sql.retailer.customers`
```

	id_null_count	unique_id_null_count	zip_null_count	city_null_count	state_null_count
	0	0	0	0	0

➢ Data type of columns

Field name	Type
customer_id	STRING, Categorical
customer_unique_id	STRING, Categorical
customer_zip_code_prefix	INTEGER, Categorical
customer_city	STRING, Categorical
customer_state	STRING, Categorical

➢ Number of distinct zip codes/city/state

```
SELECT
 COUNT(DISTINCT customer_zip_code_prefix) zip_code_count,
 COUNT(DISTINCT customer_city) city_count,
 COUNT(DISTINCT customer_state) state_count
FROM
 `scaler-sql.retailer.customers`
```

Row	zip_code_count	city_count	state_count
1	14994	4119	27

2. Sellers

	seller_id	seller_zip_code_prefix	seller_city	seller_state
1	4be2e7f96b4fd749d52dff41f80e39dd	69900	rio branco	AC
2	327b89b872c14d1c0be7235ef4871685	69005	manaus	AM
3	4221a7df464f1fe2955934e30ff3a5a1	48602	bahia	BA

➢ Number of rows in the data

```
SELECT COUNT(*) FROM `scaler-sql.retailer.sellers`
```

- 3095

➢ Number of null or missing values

```
SELECT
 COUNT(*) - COUNT(seller_id) id_null_count,
 COUNT(*) - COUNT(seller_zip_code_prefix) zip_null_count,
 COUNT(*) - COUNT(seller_city) city_null_count,
 COUNT(*) - COUNT(seller_state) state_null_count
FROM
 `scaler-sql.retailer.sellers`
```

Row	id_null_count	zip_null_count	city_null_count	state_null_count
1	0	0	0	0

➢ Data type of columns

Field name	Type
seller_id	STRING, Categorical
seller_zip_code_prefix	INTEGER, Categorical
seller_city	STRING, Categorical
seller_state	STRING, Categorical

➢ Number of distinct zip codes/city/state

```
SELECT
 COUNT(DISTINCT seller_zip_code_prefix) zip_code_count,
 COUNT(DISTINCT seller_city) city_count,
 COUNT(DISTINCT seller_state) state_count
FROM
 `scaler-sql.retailer.sellers`
```

Row	zip_code_count	city_count	state_count
1	2246	611	23

3. Geolocation

	geolocation_zip_code_prefix	geolocation_lat	geolocation_lng	geolocation_city	geolocation_state
1	49010	-10.910514518754546	-37.052400776992329	aracaju	SE
2	49047	-10.9268145	-37.071063000000009	aracaju	SE
3	49030	-10.970164794304576	-37.061643830745815	aracaju	SE

➢ Number of rows in the data

```
SELECT COUNT(*) FROM `scaler-sql.retailer.geolocation`
```

- 1000163

➢ Number of null or missing values

```
SELECT
 COUNT(*) - COUNT(geolocation_zip_code_prefix) zip_null_count,
 COUNT(*) - COUNT(geolocation_lat) lat_null_count,
 COUNT(*) - COUNT(geolocation_lng) lng_null_count,
 COUNT(*) - COUNT(geolocation_city) city_null_count,
 COUNT(*) - COUNT(geolocation_state) state_null_count
FROM
 `scaler-sql.retailer.geolocation`
```

	zip_null_count	lat_null_count	lng_null_count	city_null_count	state_null_count
1	0	0	0	0	0

➢ Data type of columns

Field name	Type
geolocation_zip_code_prefix	INTEGER, Categorical
geolocation_lat	FLOAT, Continuous
geolocation_lng	FLOAT, Continuous
geolocation_city	STRING, Categorical
geolocation_state	STRING, Categorical

➢ Number of distinct zip codes/city/state

```
SELECT
 COUNT(DISTINCT geolocation_zip_code_prefix) zip_code_count,
 COUNT(DISTINCT geolocation_city) city_count,
 COUNT(DISTINCT geolocation_state) state_count
FROM
 `scaler-sql.retailer.geolocation`
```

Row	zip_code_count	city_count	state_count
1	19015	8011	27

4. Products

	product_id	product_category	product_name_length	product_description_length	product_photos_qty	product_weight_g	product_length_cm	product_height_cm	product_width_cm
1	5eb564652db742ff8f28759cd8d2652a	*null*	*null*	*null*	*null*	*null*	*null*	*null*	*null*
2	09ff539a621711667c43eba6a3bd8466	babies	60	865	3	*null*	*null*	*null*	*null*
3	2f763ba79d9cd987b2034aac7ceffe06	electronics	45	1198	2	595	8	6	6

➢ Number of rows in the data

```
SELECT COUNT(*) FROM `scaler-sql.retailer.products`
```

- 32951

➢ Number of null or missing values

```
SELECT
 COUNT(*) - COUNT(product_id) id_null_ct,
 COUNT(*) - COUNT(product_category) cat_null_ct,
 COUNT(*) - COUNT(product_name_length) name_null_ct,
 COUNT(*) - COUNT(product_description_length) desc_null_ct,
 COUNT(*) - COUNT(product_photos_qty) photo_null_ct,
 COUNT(*) - COUNT(product_weight_g) wgt_null_ct,
 COUNT(*) - COUNT(product_length_cm) len_null_ct,
 COUNT(*) - COUNT(product_height_cm) hgt_null_ct,
 COUNT(*) - COUNT(product_width_cm) wdt_null_ct
FROM
 `scaler-sql.retailer.products`
```

	id_null_ct	cat_null_ct	name_null_ct	desc_null_ct	photo_null_ct	wgt_null_ct	len_null_ct	hgt_null_ct	wdt_null_ct
1	0	610	610	610	610	2	2	2	2

➢ Data type of columns

Field name	Type
product_id	STRING, Categorical
product_category	STRING, Categorical
product_name_length	INTEGER, Discrete
product_description_length	INTEGER, Discrete
product_photos_qty	INTEGER, Discrete
product_weight_g	INTEGER, Continuous
product_length_cm	INTEGER, Continuous
product_height_cm	INTEGER, Continuous
product_width_cm	INTEGER, Continuous

➢ Number of distinct product categories

```
SELECT
 COUNT(DISTINCT product_category) product_category_count
FROM `scaler-sql.retailer.products`
```

- 73

5. Orders

	order_id	customer_id	order_status	order_purchase_timestamp	order_approved_at	order_delivered_carrier_date	order_delivered_customer_date	order_estimated_delivery_date
1	7a4df5d8cff4090e541401a20a22bb80	725e9c75605414b21fd8c8d5a1c2f1d6	created	2017-11-25 11:10:33 UTC	*null*	*null*	*null*	2017-12-12 00:00:00 UTC
2	35de4050331c6c644cddc86f4f2d0d64	4ee64f4bfc542546f422da0aeb462853	created	2017-12-05 01:07:58 UTC	*null*	*null*	*null*	2018-01-08 00:00:00 UTC
3	b5359909123fa03c50bdb0cfed07f098	438449d4af8980d107bf04571413a8e7	created	2017-12-05 01:07:52 UTC	*null*	*null*	*null*	2018-01-11 00:00:00 UTC

➢ Number of rows in the data

```
SELECT COUNT(*) FROM `scaler-sql.retailer.orders`
```

- 99441

➢ Number of null or missing values

```
SELECT
 COUNT(*) - COUNT(product_id) id_null_ct,
 COUNT(*) - COUNT(product_category) cat_null_ct,
 COUNT(*) - COUNT(product_name_length) name_null_ct,
 COUNT(*) - COUNT(product_description_length) desc_null_ct,
 COUNT(*) - COUNT(product_photos_qty) photo_null_ct,
 COUNT(*) - COUNT(product_weight_g) wgt_null_ct,
 COUNT(*) - COUNT(product_length_cm) len_null_ct,
 COUNT(*) - COUNT(product_height_cm) hgt_null_ct,
 COUNT(*) - COUNT(product_width_cm) wdt_null_ct
FROM
 `scaler-sql.retailer.products`
```

	id	cust_id	status	pur_time stamp	approved_at	deli_carrier _date	deli_cust _date	est_deli _date	
1	0	0	0	0	160	1783	2965	0	

➢ Data type of columns

Field name	Type
order_id	STRING, Categorical
customer_id	STRING, Categorical
order_status	STRING, Categorical
order_purchase_timestamp	TIMESTAMP, Timeseries
order_approved_at	TIMESTAMP, Timeseries
order_delivered_carrier_date	TIMESTAMP, Timeseries
order_delivered_customer_date	TIMESTAMP, TImeseries
order_estimated_delivery_date	TIMESTAMP, Timeseries

➢ Number of distinct order statuses

```
SELECT
 COUNT(DISTINCT order_status) order_status_count
FROM
  `scaler-sql.retailer.orders`
```

- 8

```
SELECT
  DISTINCT order_status
FROM
  `scaler-sql.retailer.orders`
```

- Created
- Shipped
- Approved
- Canceled
- Invoiced
- Delivered
- Processing
- Unavailable

6. Order Items

	order_id	order_item_id	product_id	seller_id	shipping_limit_date	price	freight_value
1	f09e36e2586 56850b9265 7ac5f67b6d5	1	44d53f1240d 6332232e439 3c06500475	b64d51f0435 e884e8de603 b1655155ae	2018-07-09 13:31:36 UTC	3.0	12.79
2	f9ccaff7267f d0cf076e795 b1fae8b69	1	44d53f1240d 6332232e439 3c06500475	b64d51f0435 e884e8de603 b1655155ae	2018-08-14 14:04:44 UTC	3.0	15.23
3	c79bdf061e2 2288609201e c60deb42fb	1	5304ff3fa358 56a156e1170 a6022d34d	cf6f6bc4df39 99b9c6440f1 24fb2f687	2017-05-12 19:05:20 UTC	3.5	8.72

➢ Number of rows in the data

```
SELECT COUNT(*) FROM `scaler-sql.retailer.order_items`
```

- 112650

➢ Number of null or missing values

```
SELECT
 COUNT(*) - COUNT(order_id) id,
 COUNT(*) - COUNT(order_item_id) item_id,
 COUNT(*) - COUNT(product_id) prod_id,
 COUNT(*) - COUNT(seller_id) seller_id,
 COUNT(*) - COUNT(shipping_limit_date) ship_lt_date,
 COUNT(*) - COUNT(price) price,
 COUNT(*) - COUNT(freight_value) freight_val
FROM
 `scaler-sql.retailer.order_items`
```

	id	item_id	prod_id	seller_id	ship_lt_date	price	freight_val
1	0	0	0	0	0	0	0

➢ Data type of columns

Field name	Type
order_id	STRING, Categorical
order_item_id	INTEGER, Categorical
product_id	STRING, Categorical
seller_id	STRING, Categorical
shipping_limit_date	TIMESTAMP, Timeseries
price	FLOAT, Continuous
freight_value	FLOAT, Continuous

7. Payments

	order_id	payment_sequential	payment_type	payment_installments	payment_value
1	1a57108394169c0b47d8f876acc9ba2d	2	credit_card	0	129.94
2	744bade1fcf9ff3f31d860ace076d422	2	credit_card	0	58.69
3	8bcbe01d44d147f901cd3192671144db	4	voucher	1	0.0

➢ Number of rows in the data

```
SELECT COUNT(*) FROM `scaler-sql.retailer.payments`
```

- 103886

➢ Number of null or missing values

```
SELECT
 COUNT(*) - COUNT(order_id) order_id_null_ct,
 COUNT(*) - COUNT(payment_sequential) pay_seq_null_ct,
 COUNT(*) - COUNT(payment_type) pay_type_null_ct,
 COUNT(*) - COUNT(payment_installments) pay_inst_null_ct,
 COUNT(*) - COUNT(payment_value) pay_val_null_ct
FROM `scaler-sql.retailer.payments`
```

Row	order_id_null_ct	pay_seq_null_ct	pay_type_null_ct	pay_inst_null_ct	pay_val_null_ct
1	0	0	0	0	0

➢ Data type of columns

Field name	Type
order_id	STRING, Categorical
payment_sequential	INTEGER, Discrete
payment_type	STRING, Categorical
payment_installments	INTEGER, Discrete
payment_value	FLOAT, Continuous

➢ Number of distinct payment types

```
SELECT
  COUNT(DISTINCT payment_type) payment_type_count
FROM `scaler-sql.retailer.payments`
```

- 5

➢ Distinct payment types

```
SELECT
  DISTINCT payment_type payment_type
FROM `scaler-sql.retailer.payments`
```

- Credit_card
- Voucher
- Debit_card
- UPI
- Not_defined

8. Reviews

	review_id	order_id	review_score	review_comment_title	review_creation_date	review_answer_timestamp
1	be7e2989673cb2a147b87ba73277da9e	777c67eab7c0712ccde8ffbb22715adb	1	*null*	0001-04-17 00:00:00 UTC	0001-04-17 07:40:00 UTC
2	e12151267e4594d69eda14a871e2a1ab	4338a4463f7f9193d2a03a83a4b68ba0	1	*null*	0001-04-17 00:00:00 UTC	0001-04-17 09:04:00 UTC

➢ Number of rows in the data

```
SELECT COUNT(*) FROM `scaler-sql.retailer.reviews`
```

- 99224

➢ Number of null or missing values

```
SELECT
 COUNT(*) - COUNT(review_id) id_null_ct,
 COUNT(*) - COUNT(order_id) order_id_null_ct,
 COUNT(*) - COUNT(review_score) score_null_ct,
 COUNT(*) - COUNT(review_comment_title) title_null_ct,
 COUNT(*) - COUNT(review_creation_date) date_null_ct,
 COUNT(*) - COUNT(review_answer_timestamp) timestamp_null_ct
FROM
 `scaler-sql.retailer.reviews`
```

	id_null_ct	order_id_null_ct	score_null_ct	title_null_ct	date_null_ct	timestamp_null_ct
1	0	0	0	87675	0	0

➢ Data type of columns

Field name	Type
review_id	STRING
order_id	STRING
review_score	INTEGER
review_comment_title	STRING
review_creation_date	TIMESTAMP
review_answer_timestamp	TIMESTAMP

➢ Review score count

```
SELECT
 review_score,
 COUNT(review_score) review_score_count
FROM
 `scaler-sql.retailer.reviews`
GROUP BY
 review_score
```

Row	review_score	review_score_count
1	1	11424
2	2	3151
3	3	8179
4	4	19142
5	5	57328

Observations

- There are 96096 unique customers and 3095 sellers in the dataset
- Customers are distributed across 4119 cities divide into 27 states whereas sellers are from 611 different cities in 23 states
- There are 32951 variety of products available, divided in 73 product categories
- A total 99441 order details are available in the dataset
- Every order has an associated status field with either of these 8 values Created, Shipped, Approved, Canceled, Invoiced, Delivered, Processing, Unavailable
- Price and freight values of every item in an order is available in the order items table
- Different payment types available in the dataset are Credit card, Voucher, Debit card, UPI or Not defined
- There are review scores available per order ranging from 1 to 5, 5 being the highest. Majority of the orders have a rating of 5

II. In-depth Exploration

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

```
SELECT
 order_status,
 COUNT(*) Number_of_orders
FROM
 `scaler-sql.retailer.orders`
GROUP BY
 Order_status;
```

Row	order_status	Number_of_orders
1	created	5
2	shipped	1107
3	approved	2
4	canceled	625
5	invoiced	314
6	delivered	96478
7	processing	301
8	unavailable	609

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

```
SELECT
    FORMAT_DATETIME('%Y/%m', order_purchase_timestamp) AS month,
    COUNT(order_id) AS total_orders
FROM `scaler-sql.retailer.orders`
GROUP BY month
ORDER BY month
```

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

```
SELECT
  FORMAT_DATE('%A', order_purchase_timestamp) AS day_of_week,
  COUNT(order_id) total_orders
FROM `scaler-sql.retailer.orders`
GROUP BY day_of_week
ORDER BY total_orders DESC
```

Row	day_of_week	total_orders
1	Monday	16196
2	Tuesday	15963
3	Wednesday	15552
4	Thursday	14761
5	Friday	14122
6	Sunday	11960
7	Saturday	10887

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

```
SELECT
 CASE
   WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 3 AND 6 THEN 'DAWN'
   WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 7 AND 12 THEN 'MORNING'
   WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 13 AND 18 THEN 'AFTERNOON'
   ELSE 'NIGHT'
 END AS time,
 COUNT(*) AS total_orders
FROM `scaler-sql.retailer.orders`
GROUP BY time
```

Row	time	total_orders
1	MORNING	27733
2	NIGHT	32405
3	AFTERNOON	38135
4	DAWN	1168

5. Feature Extraction: Through order_purchase_timestamp in “orders” dataset extract
    a. Order_purchase_year
    b. Order_purchase_month
    c. Order_purchase_date
    d. Order_purchase_day
    e. Order_purchase_dayofweek
    f. Order_purchase_dayofweek_name
    g. Order_purchase_hour
    h. Order_purchase_time_day

```
SELECT
  order_purchase_timestamp,
  EXTRACT(DATE FROM order_purchase_timestamp) AS date,
  EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
  EXTRACT(MONTH FROM order_purchase_timestamp) AS month,
  EXTRACT(DAY FROM order_purchase_timestamp) AS day,
  EXTRACT(DAYOFWEEK FROM order_purchase_timestamp) AS dayofweek,
  FORMAT_DATE('%A', order_purchase_timestamp) AS dow_name,
  EXTRACT(TIME FROM order_purchase_timestamp) AS time_day,
  EXTRACT(HOUR FROM order_purchase_timestamp) AS hour
FROM `scaler-sql.retailer.orders`
```

Row	order_purchase_timestamp	date	year	month	day	dayofweek	dow_name	time_day	hour
1	2017-12-01 11:09:24 UTC	2017-12-01	2017	12	1	6	Friday	11:09:24	11
2	2017-12-01 14:38:00 UTC	2017-12-01	2017	12	1	6	Friday	14:38:00	14
3	2017-08-01 20:26:19 UTC	2017-08-01	2017	8	1	3	Tuesday	20:26:19	20
4	2018-04-01 11:30:07 UTC	2018-04-01	2018	4	1	1	Sunday	11:30:07	11
5	2017-07-01 16:48:42 UTC	2017-07-01	2017	7	1	7	Saturday	16:48:42	16

Observations

- 97% of the orders are in status delivered and 609, that is 0.6% of the order statuses are unavailable
- Looking at the month on month data on orders, there is clearly a growing trend on e-commerce in Brazil but, the data from Jan 2018 onwards shows a flattened curve or a slight drop in the orders. Close to zero orders in Nov-Dec 2018 could be because of the unavailability of data
- Brazil customers tend to order the most on Mondays and Saturdays the least. However, the overall order distribution looks the same throughout the week
- A vast majority of the orders are placed in the afternoon between 1-6pm and very few orders are made at dawn 3-6am.

III. Evolution of E-commerce orders in the Brazil region

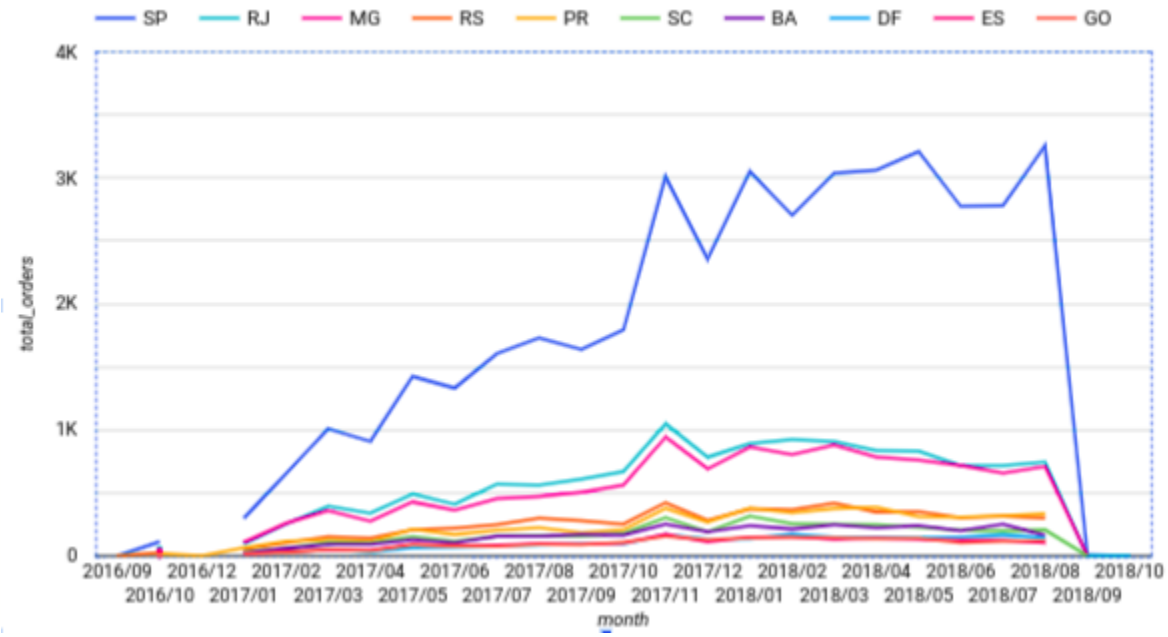
1. Get month on month orders by region

```
SELECT
    FORMAT_DATETIME('%Y/%m', order_purchase_timestamp) AS month,
    COUNT(order_id) AS total_orders
FROM `scaler-sql.retailer.orders`
GROUP BY month
ORDER BY month
```

Row	month	total_orders
1	2016/09	4
2	2016/10	324
3	2016/12	1
4	2017/01	800
5	2017/02	1780

```
SELECT
   c.customer_state,
   FORMAT_DATETIME('%Y/%m', o.order_purchase_timestamp) AS month,
   COUNT(DISTINCT o.order_id) AS total_orders
FROM `scaler-sql.retailer.orders` AS o
```

```
JOIN `scaler-sql.retailer.customers` AS c
 ON o.customer_id = c.customer_id
GROUP BY month, c.customer_state
ORDER BY month
```



2. Total of customer orders by state

```
SELECT
 c.customer_state,
 COUNT(o.order_id) total_customer_orders_per_state
FROM `scaler-sql.retailer.orders` AS o
JOIN `scaler-sql.retailer.customers` AS c
 ON o.customer_id = c.customer_id
GROUP BY c.customer_state
```

Row	customer_state	total_customer_orders_per_state
1	RJ	12852
2	RS	5466
3	SP	41746
4	DF	2140
5	PR	5045

3. Top 10 brazilian cities most no. of orders

```
SELECT
 c.customer_state,
 COUNT(o.order_id) total_customer_orders_per_state
FROM `scaler-sql.retailer.orders` AS o
JOIN `scaler-sql.retailer.customers` AS c
 ON o.customer_id = c.customer_id
GROUP BY c.customer_state
ORDER BY COUNT(o.order_id) DESC
LIMIT 10
```

Row	customer_state	total_customer_orders_per_state
1	SP	41746
2	RJ	12852
3	MG	11635
4	RS	5466
5	PR	5045
6	SC	3637
7	BA	3380
8	DF	2140
9	ES	2033
10	GO	2020

4. How are customers distributed in Brazil

```
SELECT
 customer_state,
 COUNT(customer_unique_id) number_of_customers
FROM `scaler-sql.retailer.customers`
GROUP BY customer_state
```

Row	customer_state	number_of_customers
1	RN	485
2	CE	1336
3	RS	5466
4	SC	3637
5	SP	41746

5. City wise number of unique customers

```
SELECT
 customer_city,
 COUNT(customer_unique_id) number_of_customers
FROM `scaler-sql.retailer.customers`
GROUP BY customer_city
ORDER BY COUNT(customer_unique_id) DESC
```

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

Observations

- The state Sao Paulo accounts for 42% of the total orders made and tops the list as the state with most number of orders
- Rio de Janeiro is a distant second with 13% of the total orders and Minas Gerais is a close third with 12% of the total orders. These 3 states together hold 65% share of all the orders made between 2016 and 2018
- Most of the customers are from city Sao Paulo and the city with second highest number of customers is Rio de Janeiro

IV. 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:
    a. price_per_order = sum(price) / count(order_id)
    b. freight_per_order= sum(freight_value) / count(order_id)

```
WITH order_details AS (
   SELECT
     EXTRACT(YEAR FROM o.order_purchase_timestamp) AS order_purchase_year,
     EXTRACT(MONTH FROM o.order_purchase_timestamp) AS order_purchase_month,
     COUNT(o.order_id) AS number_of_orders,
     ROUND(SUM(oi.price)) AS total_price,
     ROUND(SUM(oi.freight_value)) AS total_freight_value,
     ROUND(SUM(oi.price)/COUNT(o.order_id), 2) AS price_per_order,
     ROUND(SUM(oi.freight_value)/COUNT(o.order_id), 2) AS freight_per_order
   FROM `scaler-sql.retailer.orders` AS o
   JOIN `scaler-sql.retailer.order_items` AS oi
     ON o.order_id = oi.order_id
   GROUP BY order_purchase_year, order_purchase_month)
SELECT * FROM order_details
```

	order_purchase_year	order_purchase_month	number_of_orders	total_price	total_freight_value	price_per_order	freight_per_order
1	2018	7	7092	895507.0	163221.0	126.27	23.01
2	2018	8	7248	854686.0	148622.0	117.92	20.51
3	2017	5	4136	506071.0	80120.0	122.36	19.37

Step 2: Answer the following questions:

1. Total amount sold in 2017 between Jan to August

```
SELECT
 SUM(total_price) AS total_amount_sold
FROM order_details
WHERE order_purchase_year = 2017
 AND order_purchase_month BETWEEN 1 AND 8
```

   - Total_amount_sold = 3113000.0

2. Total amount sold in 2018 between Jan to august

```
SELECT
 SUM(total_price) AS total_amount_sold
FROM order_details
WHERE order_purchase_year = 2018
 AND order_purchase_month BETWEEN 1 AND 8
```

   - Total_amount_sold = 7385905.0

3. % increase from 2017 to 2018 = (7385905 - 3113000) * 100 / (3113000)
   - 137.26 %

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

```
WITH order_details AS (
  SELECT
    c.customer_state,
    ROUND(SUM(oi.price)) AS total_price,
    ROUND(AVG(oi.price), 2) AS mean_price,
    ROUND(SUM(oi.freight_value)) AS total_freight_value,
    ROUND(AVG(oi.freight_value), 2) AS mean_freight_value
  FROM `scaler-sql.retailer.orders` AS o
  JOIN `scaler-sql.retailer.order_items` AS oi
    ON o.order_id = oi.order_id
  JOIN `scaler-sql.retailer.customers` AS c
    ON o.customer_id = c.customer_id
  GROUP BY c.customer_state)

SELECT * FROM order_details
```

	customer_state	total_price	mean_price	total_freight_value	mean_freight_value
1	MT	156454.0	148.3	29715.0	28.17
2	MA	119648.0	145.2	31524.0	38.26
3	AL	80315.0	180.89	15915.0	35.84
4	SP	5202955.0	109.65	718723.0	15.15
5	MG	1585308.0	120.75	270853.0	20.63

1. Mean & Sum of price by customer state

```
SELECT
  customer_state,
  total_price,
  mean_price
FROM order_details
ORDER BY mean_price DESC
```

Row	customer_state	total_price	mean_price
1	PB	115268.0	191.48
2	AL	80315.0	180.89
3	AC	15983.0	173.73
4	RO	46141.0	165.97
5	PA	178948.0	165.69

2. Mean & Sum of freight value by customer state

```
SELECT
  customer_state,
  total_freight_value,
  mean_freight_value
FROM order_details
ORDER BY mean_freight_value DESC
```

Row	customer_state	total_freight_value	mean_freight_value
1	RR	2235.0	42.98
2	PB	25720.0	42.72
3	RO	11417.0	41.07
4	AC	3687.0	40.07
5	PI	21218.0	39.15

Observations

- There is a 137% increase in a year in the total amount sold in Jan-Aug 2018 compared Jan-Aug 2017
- States with the highest mean_price are PB, AL and AC
- States with highest mean_freight_value are RR, PB and RO

V. Analysis on sales, freight and delivery time

1. Calculating days between purchasing, delivering and estimated delivery

```
SELECT
 order_purchase_timestamp,
 order_delivered_customer_date,
 order_estimated_delivery_date,
 DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY)
AS time_to_delivery,
 DATE_DIFF(order_delivered_customer_date, order_estimated_delivery_date,
DAY) AS diff_estimated_delivery
FROM `scaler-sql.retailer.orders`
WHERE order_delivered_customer_date IS NOT NULL
```

	order_purchase_timestamp	order_delivered_customer_date	order_estimated_delivery_date	time_to_delivery	diff_estimated_delivery
1	2016-10-07 14:52:30 UTC	2016-10-14 15:07:11 UTC	2016-11-29 00:00:00 UTC	7	-45
2	2018-02-19 19:48:52 UTC	2018-03-21 22:03:51 UTC	2018-03-09 00:00:00 UTC	30	12
3	2016-10-09 15:39:56 UTC	2016-11-09 14:53:50 UTC	2016-12-08 00:00:00 UTC	30	-28
4	2016-10-09 00:56:52 UTC	2016-10-16 14:36:59 UTC	2016-11-30 00:00:00 UTC	7	-44
5	2016-10-08 20:17:50 UTC	2016-10-19 18:47:43 UTC	2016-11-30 00:00:00 UTC	10	-41

2. Create columns:

a. time_to_delivery = order_purchase_timestamp-order_delivered_customer_date
b. diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date

```
SELECT
 order_purchase_timestamp,
 DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY) AS
time_to_delivery,
 DATE_DIFF(order_delivered_customer_date, order_estimated_delivery_date, DAY)
AS diff_estimated_delivery
FROM `scaler-sql.retailer.orders` WHERE order_delivered_customer_date IS NOT
NULL
```

Row	order_purchase_timestamp	time_to_delivery	diff_estimated_delivery
1	2016-10-07 14:52:30 UTC	7	-45
2	2018-02-19 19:48:52 UTC	30	12
3	2016-10-09 15:39:56 UTC	30	-28
4	2016-10-09 00:56:52 UTC	7	-44
5	2016-10-08 20:17:50 UTC	10	-41

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

```
WITH state_wise_delivery_details AS
 (SELECT
   c.customer_state,
   ROUND(AVG(oi.freight_value), 2) AS mean_freight_value,
   ROUND(AVG(DATE_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp,
DAY)), 2) AS mean_time_to_delivery,
   ROUND(AVG(DATE_DIFF(o.order_delivered_customer_date,
o.order_estimated_delivery_date, DAY)), 2) AS mean_diff_estimated_delivery
 FROM `scaler-sql.retailer.orders` AS o
 JOIN `scaler-sql.retailer.customers` AS c
   ON o.customer_id = c.customer_id
 JOIN `scaler-sql.retailer.order_items` AS oi
   ON o.order_id = oi.order_id
 WHERE order_delivered_customer_date IS NOT NULL
 GROUP BY c.customer_state)

SELECT * FROM state_wise_delivery_details
```

	customer_state	mean_freight_value	mean_time_to_delivery	mean_diff_estimated_delivery
1	RJ	20.91	14.69	-11.14
2	MG	20.63	11.52	-12.4
3	SC	21.51	14.52	-10.67
4	SP	15.11	8.26	-10.27
5	GO	22.56	14.95	-11.37

4. Sort the data to get the following:

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

```
SELECT
 customer_state,
 mean_freight_value
FROM state_wise_delivery_details
ORDER BY mean_freight_value
LIMIT 5
```

Row	customer_state	mean_freight_value
1	SP	15.11
2	PR	20.47
3	MG	20.63
4	RJ	20.91
5	DF	21.07

```
SELECT
 customer_state,
 mean_freight_value
FROM state_wise_delivery_details
ORDER BY mean_freight_value DESC
LIMIT 5
```

Row	customer_state	mean_freight_value
1	PB	43.09
2	RR	43.09
3	RO	41.33
4	AC	40.05
5	PI	39.12

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

```
SELECT
 customer_state,
 mean_time_to_delivery
FROM state_wise_delivery_details
ORDER BY mean_time_to_delivery
LIMIT 5
```

Row	customer_state	mean_time_to_delivery
1	SP	8.26
2	PR	11.48
3	MG	11.52

4	DF	12.5
5	SC	14.52

```
SELECT
 customer_state,
 mean_time_to_delivery
FROM state_wise_delivery_details
ORDER BY mean_time_to_delivery DESC
LIMIT 5
```

Row	customer_state	mean_time_to_delivery
1	RR	27.83
2	AP	27.75
3	AM	25.96
4	AL	23.99
5	PA	23.3

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

```
SELECT
 customer_state,
 mean_diff_estimated_delivery
FROM state_wise_delivery_details
ORDER BY mean_diff_estimated_delivery
LIMIT 5
```

Row	customer_state	mean_diff_estimated_delivery
1	AC	-20.01
2	RO	-19.08
3	AM	-18.98
4	AP	-17.44
5	RR	-17.43

```
SELECT
 customer_state,
 mean_diff_estimated_delivery
```

```
FROM state_wise_delivery_details
ORDER BY mean_diff_estimated_delivery DESC
LIMIT 5
```

Row	customer_state	mean_diff_estimated_delivery
1	AL	-7.98
2	MA	-9.11
3	SE	-9.17
4	ES	-9.77
5	BA	-10.12

Observations

- States with lowest average freight value are SP, PR & MG and states with highest freight value are PB, RR & RO
- States with lowest average time to delivery are SP, PR & MG and states with highest average time to delivery are RR, AP & AM
- States with fastest delivery are AC, RO & AM and states with relatively less fast delivery are AL, MA & SE

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

1. Count of orders for different payment types

```
SELECT
 p.payment_type,
 count(*) number_of_orders
FROM `scaler-sql.retailer.orders` AS o
JOIN `scaler-sql.retailer.payments` AS p
 ON o.order_id = p.order_id
GROUP BY p.payment_type
ORDER BY number_of_orders DESC
```

Row	payment_type	number_of_orders
1	credit_card	76795
2	UPI	19784
3	voucher	5775
4	debit_card	1529
5	not_defined	3

2. Distribution of payment installments and count of orders

```
SELECT
 p.payment_installments,
 count(*) number_of_orders
FROM `scaler-sql.retailer.orders` AS o
JOIN `scaler-sql.retailer.payments` AS p
 ON o.order_id = p.order_id
GROUP BY p.payment_installments
ORDER BY number_of_orders DESC
```

Row	payment_installments	number_of_orders
1	1	52546
2	2	12413
3	3	10461
4	4	7098
5	10	5328
6	5	5239
7	8	4268
8	6	3920
9	7	1626
10	9	644
11	12	133
12	15	74

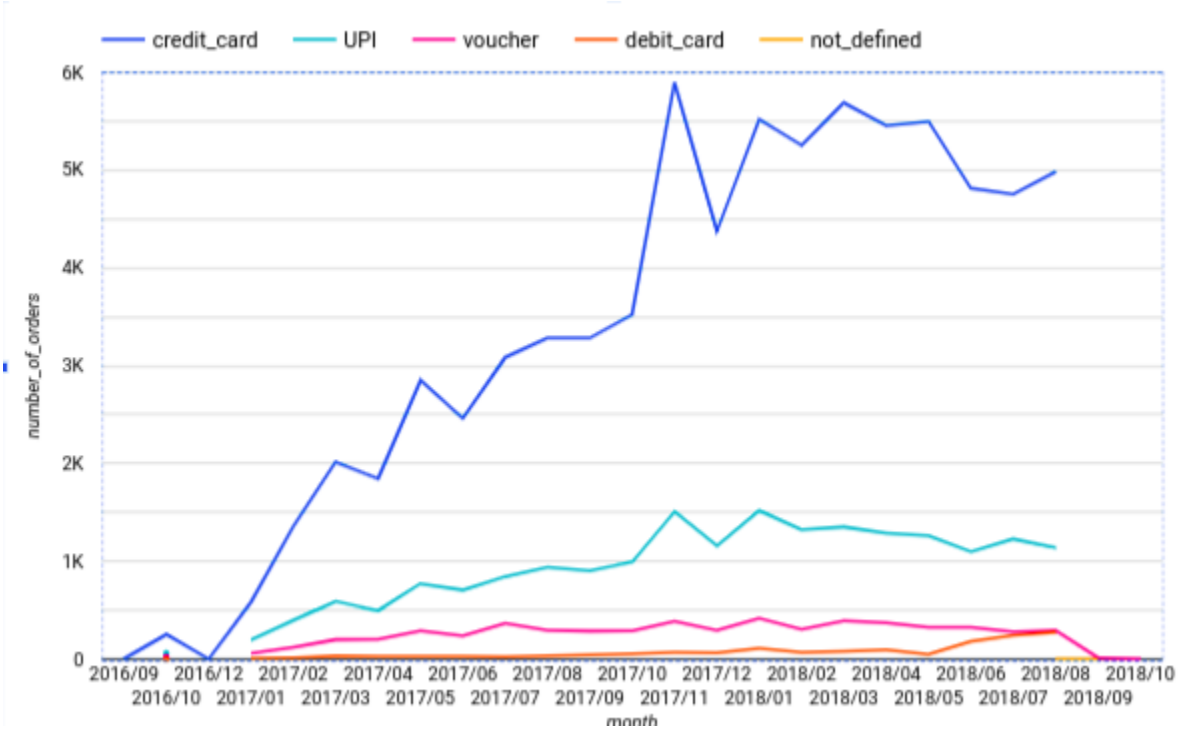
13	18	27
14	11	23
15	24	18
16	20	17
17	13	16
18	14	15
19	17	8
20	16	5
21	21	3
22	0	2
23	22	1
24	23	1

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

```
SELECT
 FORMAT_DATETIME('%Y/%m', o.order_purchase_timestamp) AS month,
 p.payment_type,
 count(*) number_of_orders
FROM `scaler-sql.retailer.orders` AS o
JOIN `scaler-sql.retailer.payments` AS p
 ON o.order_id = p.order_id
GROUP BY month, p.payment_type
ORDER BY month, number_of_orders DESC
```

Row	month	payment_type	number_of_orders
1	2016/09	credit_card	3
2	2016/10	credit_card	254
3	2016/10	UPI	63
4	2016/10	voucher	23
5	2016/10	debit_card	2

6	2016/12	credit_card	1
7	2017/01	credit_card	583
8	2017/01	UPI	197
9	2017/01	voucher	61
10	2017/01	debit_card	9



Observations

- Most popular payment type is using a credit card which accounts for over 77% of the total payments and UPI holds the second most share of purchases made
- Most of the payments are one time and less than half of the payments are through multiple installments
- Most popular number of installments are 1, 2, 3, 4, 10, 5 in that order
- Only 343, that is less than 1% of the payments have opted for more than 10 installments
- There is a high growth in number of credit card payments from 2016-2018 and UPI is also growing but, at a slower rate

VII. Actionable Insights

1. Looking at the month on month data on orders, there is clearly a growing trend on e-commerce in Brazil although, the data from Jan 2018 onwards shows a flattened curve or a slight drop in the orders.
2. There is around 2.5x increase in a year in the total amount sold in Jan-Aug 2018 compared to Jan-Aug 2017.
3. Brazil customers tend to order the most on Mondays and Saturdays the least.
4. A vast majority of the orders are placed in the afternoon between 1-6pm and very few orders are made at dawn 3-6am.
5. The state Sao Paulo accounts for 42% of the total orders made and tops the list as the state with the most number of orders.
6. Most of the customers are from city Sao Paulo and the city with second highest number of customers is Rio de Janeiro.
7. States with the highest mean_price are PB, AL and AC and highest mean_freight_value are RR, PB and RO.
8. States with lowest average time to delivery are SP, PR & MG and states with highest average time to delivery are RR, AP & AM. This may be because of the transportation infrastructure and proximity to sellers in these states.
9. States with fastest delivery are AC, RO & AM and states with relatively less fast delivery are AL, MA & SE.
10. Most popular payment type is using a credit card which accounts for over 77% of the total payments and UPI holds the second most share of purchases made.
11. Most popular choice of number of installments in case of EMI is 2 and only less than 1% of the payments have opted for more than 10 installments

VIII. Recommendations

1. Coupons can be given on Saturdays to improve the business as it is the week day with the least number of orders placed.
2. Discount offers should be given to the customers from states PB, AL and AC. Even a slightest increase in the orders by customers from these states will have a better reflection in the revenue as they tend to have the highest average price per order.
3. Best practices used for the delivery in states AC, RO and AM can be employed in AL, MA and SE if feasible to improve the delivery time.
4. UPI payments show a rising trend although the penetration is still very low compared to Credit Card payments could be used as an opportunity to incorporate growing UPI service providers with a revenue sharing model.
5. EMI options with less than or equal to 10 installments for more orders will have a positive impact on the business compared to more than 10 installments as they have less than 1% share of total orders.
6. Customers from the state of Sao Paulo must be maintained by giving incentives as their contributions account for nearly 50% of the total orders.
7. The e-commerce growth rate is stagnant in the year 2018 compared to the rapid growth in 2017. Further analysis with various economic factors may be required to understand this trend.