# ANALYZING ECOMMERCE BUSINESS PERFORMANCE QUERY

1. ***Generate table and define primary key on each table***

```
CREATE TABLE customers(
	customer_id VARCHAR PRIMARY KEY,
	customer_unique_id VARCHAR,
	customer_zip_code_prefix INT,
	customer_city VARCHAR,
	customer_state VARCHAR
)
COPY customers(customer_id, customer_unique_id, customer_zip_code_prefix,
customer_city, customer_state)
FROM 'E:\DIKA\DATA PRODUKTIF DIKA\DATA SCIENCE\Rakamin\Mini project\Analyzing
eCommerce Business Performance with SQL\Dataset\customers_dataset.csv'
DELIMITER ','
CSV HEADER;

CREATE TABLE geolocation(
	geolocation_zip_code_prefix INT,
	geolocation_lat DECIMAL,
	geolocation_lng DECIMAL,
	geolocation_city VARCHAR,
	geolocation_state VARCHAR
)
COPY geolocation(geolocation_zip_code_prefix, geolocation_lat, geolocation_lng,
geolocation_city, geolocation_state)
FROM 'E:\DIKA\DATA PRODUKTIF DIKA\DATA SCIENCE\Rakamin\Mini project\Analyzing
eCommerce Business Performance with SQL\Dataset\geolocation_dataset.csv'
DELIMITER ','
CSV HEADER;

CREATE TABLE order_items(
	order_id VARCHAR,
	order_item_id INT,
	product_id VARCHAR,
	seller_id VARCHAR,
	shipping_limit_date TIMESTAMP,
	price FLOAT,
	freight_value FLOAT
)
COPY order_items(order_id, order_item_id, product_id, seller_id, shipping_limit_date, price,
freight_value)
FROM 'E:\DIKA\DATA PRODUKTIF DIKA\DATA SCIENCE\Rakamin\Mini project\Analyzing
eCommerce Business Performance with SQL\Dataset\order_items_dataset.csv'
DELIMITER ','
```

```
CSV HEADER
ALTER TABLE order_items ADD FOREIGN KEY (order_id) REFERENCES orders
ALTER TABLE order_items ADD FOREIGN KEY (product_id) REFERENCES product
ALTER TABLE order_items ADD FOREIGN KEY (seller_id) REFERENCES sellers;

CREATE TABLE order_payments(
	order_id VARCHAR,
	payment_sequential INT,
	payment_type VARCHAR,
	payment_installments INT,
	payment_value FLOAT
)
COPY order_payments(order_id, payment_sequential, payment_type,
payment_installments, payment_value)
FROM 'E:\DIKA\DATA PRODUKTIF DIKA\DATA SCIENCE\Rakamin\Mini project\Analyzing
eCommerce Business Performance with SQL\Dataset\order_payments_dataset.csv'
DELIMITER ','
CSV HEADER
ALTER TABLE order_payments ADD FOREIGN KEY (order_id) REFERENCES
orders(order_id);

CREATE TABLE order_reviews(
	review_id VARCHAR,
	order_id VARCHAR,
	review_score INT,
	review_comment_title VARCHAR,
	review_comment_message VARCHAR,
	review_creation_date DATE,
	review_answer_timestamp TIMESTAMP
)
COPY order_reviews(review_id, order_id, review_score, review_comment_title,
review_comment_message, review_creation_date, review_answer_timestamp)
FROM 'E:\DIKA\DATA PRODUKTIF DIKA\DATA SCIENCE\Rakamin\Mini project\Analyzing
eCommerce Business Performance with SQL\Dataset\order_reviews_dataset.csv'
DELIMITER ','
CSV HEADER
ALTER TABLE order_reviews ADD FOREIGN KEY (order_id) REFERENCES
orders(order_id);

CREATE TABLE orders(
	order_id VARCHAR PRIMARY KEY,
	customer_id VARCHAR,
	order_status VARCHAR,
	order_purchase_timestamp TIMESTAMP,
	order_approved_at TIMESTAMP,
	order_delivered_carrier_date TIMESTAMP,
	order_delivered_customer_date TIMESTAMP,
	order_estimated_delivery_date DATE
)
COPY 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)
FROM 'E:\DIKA\DATA PRODUKTIF DIKA\DATA SCIENCE\Rakamin\Mini project\Analyzing
eCommerce Business Performance with SQL\Dataset\orders_dataset.csv'
DELIMITER ','
CSV HEADER
ALTER TABLE orders ADD FOREIGN KEY (customer_id) REFERENCES
customers(customer_id);

CREATE TABLE product(
        idx INT,
        product_id VARCHAR PRIMARY KEY,
        product_category_name VARCHAR,
        product_name_lenght FLOAT,
        product_description_lenght FLOAT,
        product_photos_qty FLOAT,
        product_weight_g FLOAT,
        product_length_cm FLOAT,
        product_height_cm FLOAT,
        product_width_cm FLOAT
)
COPY product(idx, product_id, product_category_name, product_name_lenght,
        product_description_lenght,
        product_photos_qty,
        product_weight_g,
        product_length_cm,
        product_height_cm,
        product_width_cm)
FROM 'E:\DIKA\DATA PRODUKTIF DIKA\DATA SCIENCE\Rakamin\Mini project\Analyzing
eCommerce Business Performance with SQL\Dataset\product_dataset.csv'
DELIMITER ','
CSV HEADER;


CREATE TABLE sellers(
        seller_id VARCHAR PRIMARY KEY,
        seller_zip_code_prefix INT,
        seller_city VARCHAR,
        seller_state VARCHAR
)
COPY sellers(seller_id, seller_zip_code_prefix, seller_city, seller_state)
FROM 'E:\DIKA\DATA PRODUKTIF DIKA\DATA SCIENCE\Rakamin\Mini project\Analyzing
eCommerce Business Performance with SQL\Dataset\sellers_dataset.csv'
DELIMITER ','
CSV HEADER;
```

2. ***Calculate of MAU (Monthly Active User) per year, new customer per year, repeat order customer per year, and average of order frequency per year***

```
--MAU (Monthly Active User) per year
WITH mau AS(
SELECT year, round(AVG(mau), 2) AS avg_mau
FROM(
        SELECT      date_part('year', o.order_purchase_timestamp) AS year,
                    date_part ('month', o.order_purchase_timestamp) AS month,
                    count(distinct c.customer_unique_id) AS mau
        FROM orders AS o
        JOIN    customers AS c ON o.customer_id = c.customer_id
        GROUP BY 1, 2
) subq
GROUP BY 1
ORDER BY 1 ASC
),
--new customer per year
new_customer AS(
SELECT      date_part('year', first_order) AS year,
            COUNT(DISTINCT customer_unique_id) AS total_new_customer
FROM(
        SELECT      c.customer_unique_id,
                    min(o.order_purchase_timestamp) AS first_order
        FROM orders AS o
        JOIN    customers AS c ON o.customer_id = c.customer_id
        GROUP BY 1
) subq
GROUP BY 1
ORDER BY 1 ASC
),
--repeat order customer per year
repeat AS(
SELECT      year,
            COUNT(customer) AS total_repeat_customer
FROM(
        SELECT      c.customer_unique_id,
                    COUNT(1) AS customer,
                    date_part('year', o.order_purchase_timestamp) AS year
        FROM orders AS o
        JOIN    customers AS c ON o.customer_id = c.customer_id
        GROUP BY 1, 3
        HAVING COUNT (1) > 1
) subq
GROUP BY 1
ORDER BY 1 ASC
),
--average of order frequency per year
avg_freq AS(
SELECT        year,
              ROUND(AVG(total_order), 3) AS avg_total_order
FROM(
        SELECT  c.customer_unique_id,
                date_part('year', o.order_purchase_timestamp) AS year,
                COUNT(1) AS total_order
        FROM orders AS o
        JOIN    customers AS c ON o.customer_id = c.customer_id
        GROUP BY 1, 2
) subsq
GROUP BY 1
ORDER BY 1 ASC
)
--combine the new metrics to be one table
SELECT m.year, m.avg_mau, nc.total_new_customer, r.total_repeat_customer,
av.avg_total_order
FROM mau AS m
JOIN new_customer AS nc ON m.year = nc.year
JOIN repeat AS r ON m.year = r.year
JOIN avg_freq AS av ON m.year = av.year
```

3. ***Calculate total revenue, total canceled customer, top product category and top product revenue, most canceled product and total canceled product***

```
--Total Revenue
WITH total_revenues AS(
SELECT date_part('year', order_purchase_timestamp) AS year,
     ROUND(SUM(revenue)) AS total_revenue
FROM(
        SELECT        order_id,
                      SUM(price + freight_value) AS revenue
        FROM order_items
   GROUP BY 1
) subsq
JOIN orders o
ON subsq.order_id = o.order_id
WHERE order_status = 'delivered'
GROUP BY 1
ORDER BY 1 ASC
),
--Total Canceled Customer
canceled_customers AS(
SELECT date_part('year', order_purchase_timestamp) AS year,
     SUM(cust) AS canceled_customer
FROM(
        SELECT        order_id,
                              COUNT(*) AS cust
          FROM order_items
   GROUP BY 1
) subsq
JOIN orders o
ON subsq.order_id = o.order_id
WHERE order_status = 'canceled'
GROUP BY 1
ORDER BY 1 ASC
),
```
*--Top Product Category and Top Product Revenue*
```
top_product AS(
SELECT year, product_category_name AS top_product_category, ROUND(total_revenue)
AS top_product_revenue
FROM (SELECT year, p.product_category_name,
           SUM(t1.revenue) AS total_revenue,
           RANK() OVER (PARTITION BY year ORDER BY SUM(t1.revenue) DESC)
           AS value_rank
     FROM (SELECT order_id, date_part('year', order_purchase_timestamp) AS year
          FROM orders
          WHERE order_status = 'delivered') o
     JOIN (SELECT order_id, product_id,
                SUM(price + freight_value)
                AS revenue
          FROM order_items
          GROUP BY order_id, product_id) t1
     ON o.order_id = t1.order_id
           JOIN product p
     ON t1.product_id = p.product_id
     GROUP BY year, p.product_category_name) t3
WHERE value_rank = 1
),
```
*--Most Canceled product and Total Canceled Product*
```
canceled_product AS(
SELECT year, product_category_name AS most_canceled_product, total_canceled_orders
FROM (SELECT year, p.product_category_name,
           SUM(t1.num_canceled_orders) AS total_canceled_orders,
           RANK() OVER (PARTITION BY year ORDER BY SUM(t1.num_canceled_orders)
DESC)
           AS value_rank
     FROM (SELECT order_id, date_part('year', order_purchase_timestamp) AS year
          FROM orders
          WHERE order_status = 'canceled') o
     JOIN (SELECT order_id, product_id,
                COUNT(order_id)
                AS num_canceled_orders
          FROM order_items
          GROUP BY order_id, product_id) t1
```

```
    ON o.order_id = t1.order_id
    JOIN product p
    ON t1.product_id = p.product_id
    GROUP BY year, p.product_category_name) t3
WHERE value_rank = 1
)
SELECT tr.year, tr.total_revenue, cc.canceled_customer, tp.top_product_category,
tp.top_product_revenue, cp.most_canceled_product, cp.total_canceled_orders
FROM total_revenues AS tr
JOIN canceled_customers AS cc ON tr.year = cc.year
JOIN top_product AS tp ON tr.year = tp.year
JOIN canceled_product AS cp ON tr.year = cp.year
```

4. ***Payment Type by Customers of All time, Payment Type by Customers each Year***

```
--Payment Type by Customers of All time
WITH num_payment AS(
SELECT payment_type,
     COUNT(order_id) AS num_payments
FROM order_payments
GROUP BY payment_type
ORDER BY num_payments DESC
),
--Payment Type by Customers each Year
type_payment AS(
SELECT payment_type,
     SUM(CASE WHEN(date_part('year', order_purchase_timestamp)) = 2016 THEN 1
ELSE 0 END) AS year_2016,
     SUM(CASE WHEN(date_part('year', order_purchase_timestamp)) = 2017 THEN 1
ELSE 0 END) AS year_2017,
     SUM(CASE WHEN(date_part('year', order_purchase_timestamp)) = 2018 THEN 1
ELSE 0 END) AS year_2018
FROM order_payments AS op
JOIN orders o ON op.order_id = o.order_id
GROUP BY 1
ORDER BY 4 DESC
)
SELECT       np.payment_type, tp.year_2016, tp.year_2017, tp.year_2018
FROM num_payment AS np
JOIN type_payment AS tp ON np.payment_type = tp.payment_type
```