# Table of Contents

# Import Database & Table

```
create database ecommerce;
```

### Create table from csv data

```
create table customers (
      customer_id varchar(250),
      customer_unique_id varchar(250),
      customer_zip_code_prefix int,
      customer_city varchar(250),
      customer_state varchar(250)
);

create table geolocation (
      geo_zip_code_prefix int,
      geo_lat float,
      geo_lng float,
      geo_city varchar(250),
      geo_state varchar(250)
);

create table order_items (
      order_id varchar(250),
      order_item_id int,
      product_id varchar(250),
      seller_id varchar(250),
      shipping_limit_date timestamp,
      price float,
      freight_value float
);

create table order_payments (
      order_id varchar(250),
      payment_sequential int,
      payment_type varchar(250),
      payment_installment int,
      payment_value float
);


create table order_reviews (
      review_id varchar(250),
      order_id varchar(250),
```

```
      review_score int,
      review_comment_title varchar(250),
      review_comment_message text,
      review_creation_date timestamp,
      review_answer timestamp
);

create table orders (
      order_id varchar(250),
      customers_id varchar(250),
      order_status varchar(250),
      order_purchase_timestamp timestamp,
      order_approved_at timestamp,
      order_delivered_carrier_date timestamp,
      order_delivered_customer_date timestamp,
      order_estimated_delivery_date timestamp
);

create table product (
      index int,
      product_id varchar(250),
      product_category_name varchar(250),
      product_name_length float,
      product_description_length float,
      product_photos_qty float,
      product_weight_g float,
      product_length_cm float,
      product_height_cm float,
      product_width_cm float
);

create table sellers (
      seller_id varchar(250),
      seller_zip_code int,
      seller_city varchar(250),
      seller_state varchar(250)
);
```

**Importing csv data in Database**

```
copy customers(
      customer_id,
      customer_unique_id,
      customer_zip_code_prefix,
```

```
      customer_city,
      customer_state
)
from 'D:\Document\File\Rakamin\Portfolio\mini project\1. Analyzing
eCommerce\customers_dataset.csv'
delimiter ','
csv header;

copy geolocation(
      geo_zip_code_prefix,
      geo_lat,
      geo_lng,
      geo_city,
      geo_state
)
from 'D:\Document\File\Rakamin\Portfolio\mini project\1. Analyzing
eCommerce\geolocation_dataset.csv'
delimiter ','
csv header;

copy order_items (
    order_id,
    order_item_id,
    product_id,
    seller_id,
    shipping_limit_date,
    price,
    freight_value)
from   D:\Document\File\Rakamin\Portfolio\mini   project\1.   Analyzing
eCommerce \order_items_dataset.csv'
delimiter ','
csv header;

copy order_payments (
    order_id,
    payment_sequential,
    payment_type,
    payment_installments,
    payment_value)
from   D:\Document\File\Rakamin\Portfolio\mini   project\1.   Analyzing
eCommerce\order_payments_dataset.csv'
delimiter ','
csv header;

copy order_reviews (
```

```
    review_id,
    order_id,
    review_score,
    review_comment_title,
    review_comment_message,
    review_creation_date,
    review_answer_time)
from  '  D:\Document\File\Rakamin\Portfolio\mini  project\1.  Analyzing
eCommerce\order_reviews_dataset.csv'
delimiter ','
csv header;

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  '  D:\Document\File\Rakamin\Portfolio\mini  project\1.  Analyzing
eCommerce\orders_dataset.csv'
delimiter ','
csv header;

copy product (
    indeks,
    product_id,
    product_category_name,
    product_name_length,
    product_description_length,
    product_photos_qty,
    product_weight_g,
    product_length_cm,
    product_height_cm,
    product_width_cm)
from  '  D:\Document\File\Rakamin\Portfolio\mini  project\1.  Analyzing
eCommerce\product_dataset.csv'
delimiter ','
csv header;

copy sellers (
    seller_id,
    seller_zip_code_prefix,
```

```
    seller_city,
    seller_state)
from  '  D:\Document\File\Rakamin\Portfolio\mini  project\1.  Analyzing
eCommerce\sellers_dataset.csv'
delimiter ','
csv header;
```

# ERD

**Create entity relationship table and export ERD to jpg**

Based on the schema image provided, it can be seen that between the arrows connecting each dataset there is a column name in the middle. This shows that, the column is a key column that connects the dataset with other datasets.

For example, the order_item dataset (orange color) corresponds to the product dataset (yellow) with the key column being the product_id column.

If we examine it again, in the order_item and product dataset there is a product_id column. However, in the product dataset all the values of the product_id column are unique (single), while in the dataset order_item the values of the product_id column are not unique (there are the same values). Therefore, the product_id column is the primary key of the product dataset and is the foreign key for the order_item dataset. Query:

```
alter table products add constraint pk_products primary key (product_id);
alter table order_items add foreign key (product_id) references products;
```

For relationships between other datasets, you can use the same method as the previous example in determining the primary key and foreign key, so that the right query is obtained as follows:

```
Primary key untuk tabel lainnya
alter table customers add constraint pk_cust primary key (customer_id);
alter table geolocation add constraint pk_geo primary key (geo_zip_code_prefix);
alter table orders add constraint pk_orders primary key (order_id);
alter table sellers add constraint pk_seller primary key (seller_id);
```

Foreign key for relationships between other tables

```
alter table customers add foreign key (customer_zip_code_prefix) references geolocation;
alter table orders add foreign key (customer_id) references customers;
alter table order_items add foreign key (order_id) references orders;
alter table order_items add foreign key (seller_id) references sellers;
alter table sellers add foreign key (seller_zip_code_prefix) references geolocation;
alter table payments add foreign key (order_id) references orders;
alter table order_items add foreign key (product_id) references products;
alter table reviews add foreign key (order_id) references orders;
```

# Annual Customer Activity Growth Analysis

--Amount of average monthly active (MAU) user per year.

```
WITH mau AS (
     SELECT
          EXTRACT (year from o.order_purchase_timestamp) AS
year,
          EXTRACT (month from o.order_purchase_timestamp) AS
month,
          COUNT(DISTINCT c.customer_unique_id) AS
montly_active_user
     FROM
          customers c
     JOIN
          orders o ON
          c.customer_id = o.customer_id
     GROUP BY 1,2
)

SELECT
     year,
     ROUND(AVG(montly_active_user), 2) as
average_montly_active_user
FROM mau
GROUP BY 1
ORDER BY 1;
```

--New customer (first time transaction) per year.

```
WITH new_customer AS (
     SELECT
          MIN(o.order_purchase_timestamp) AS first_order,
          c.customer_unique_id
     FROM
          customers c
     JOIN
          orders o ON
          c.customer_id = o.customer_id
     GROUP BY 2
```

```
)

SELECT
     EXTRACT(year from first_order) AS year,
     COUNT(1) AS count_new_customer
FROM
     new_customer
GROUP BY 1
ORDER BY 1;
```

--Amount of customer who orders more than one (repeat order) per year.

```
WITH repeat_order AS (
     SELECT
          EXTRACT (year from o.order_purchase_timestamp) AS
year,
          c.customer_unique_id,
          count(o.order_id) AS total_order
     FROM
          customers c
     JOIN
          orders o ON
          c.customer_id = o.customer_id
     GROUP BY 1,2
     HAVING count(o.order_id) > 1
)

SELECT
     year,
     COUNT(customer_unique_id) AS customer_repeat_order
FROM
     repeat_order
GROUP BY 1;
```

--Average orders of customers per year.

```
WITH freq_order AS (
     SELECT
          EXTRACT (year from o.order_purchase_timestamp) AS
year,
```

```
          c.customer_unique_id AS customer,
          count(2) AS frequency_order
     FROM
          customers c
     JOIN
          orders o ON
          c.customer_id = o.customer_id
     GROUP BY 1,2
)

SELECT
     year,
     ROUND(AVG(frequency_order), 2) AS average_order
FROM
     freq_order
GROUP BY 1
ORDER BY 1;
```

--Group 3 metrics in one display table

```
WITH avg_mau AS (
     SELECT
          year,
          ROUND(AVG(montly_active_user), 2) as average_mau
     FROM (
          SELECT
               EXTRACT (year from o.order_purchase_timestamp) AS
year,
               EXTRACT (month from o.order_purchase_timestamp)
AS month,
               COUNT(DISTINCT c.customer_unique_id) AS
montly_active_user
          FROM
               customers c
          JOIN
               orders o ON
               c.customer_id = o.customer_id
          GROUP BY 1,2
) AS mau
     GROUP BY 1
     ORDER BY 1
),
```

```
new_customers AS (
     SELECT
          EXTRACT(year from first_order) AS year,
          COUNT(1) AS new_customer
     FROM
      (
          SELECT
               MIN(o.order_purchase_timestamp) AS first_order,
               c.customer_unique_id
          FROM
               customers c
          JOIN
               orders o ON
               c.customer_id = o.customer_id
          GROUP BY 2
      )AS new_customer
     GROUP BY 1
     ORDER BY 1

),
repeat_orders AS (
     SELECT
     year,
     COUNT(customer_unique_id) AS customer_repeat_order
FROM
     ( SELECT
          EXTRACT (year from o.order_purchase_timestamp) AS
year,
          c.customer_unique_id,
          count(o.order_id) AS total_order
     FROM
          customers c
     JOIN
          orders o ON
          c.customer_id = o.customer_id
     GROUP BY 1,2
     HAVING count(o.order_id) > 1
)as repeat_order
GROUP BY 1

),
```

```
avg_order AS (
     SELECT
          year,
          ROUND(AVG(frequency_order), 2) AS average_order
     FROM
          (
               SELECT
                    EXTRACT (year from
o.order_purchase_timestamp) AS year,
                    c.customer_unique_id AS customer,
                    count(2) AS frequency_order
               FROM
                    customers c
               JOIN
                    orders o ON
                    c.customer_id = o.customer_id
               GROUP BY 1,2
) AS freq_order
     GROUP BY 1
     ORDER BY 1
)

SELECT
     m.year,
     average_mau,
     new_customer,
     customer_repeat_order,
     average_order
FROM
     avg_mau m
JOIN
     new_customers n ON
     n.year = m.year
JOIN
     repeat_orders ro ON
     ro.year = m.year
JOIN
     avg_order ao ON
     ao.year = m.year
GROUP BY 1, 2, 3, 4, 5;
```

# Annual Product Category Quality Analysis

--Total revenue each year.

```
CREATE TABLE revenue_per_year AS
SELECT
     EXTRACT (year from o.order_purchase_timestamp) AS year,
     SUM(oi.price+oi.freight_value) as revenue
FROM
     order_items oi
JOIN
     orders o ON
     oi.order_id = o .order_id
WHERE
     o.order_status = 'delivered'
GROUP BY 1
ORDER BY year;

SELECT * FROM revenue_per_year
```

--Total canceled order each year.

```
CREATE TABLE canceled_per_year AS
SELECT
     EXTRACT (year from o.order_purchase_timestamp) AS year,
     count(*) as canceled_order
FROM
     order_items oi
JOIN
     orders o ON
     oi.order_id = o .order_id
WHERE
     o.order_status = 'canceled'
GROUP BY 1
ORDER BY year;

SELECT * FROM canceled_per_year;
```

--Product category that give total most revenue each year.

```
CREATE TABLE top_product_category_by_revenue_per_year AS
SELECT
     year,
     top_product_category_by_revenue,
     revenue
FROM
     (
          SELECT
               date_part('year', o.order_purchase_timestamp) AS
year,
               p.product_category_name AS
top_product_category_by_revenue,
               SUM(oi.price + oi.freight_value) as revenue,
               RANK() OVER (PARTITION BY date_part('year',
o.order_purchase_timestamp)
                                   ORDER BY SUM(oi.price +
oi.freight_value) DESC) AS rank
          FROM
               order_items oi
          JOIN
               orders o ON
               oi.order_id = o.order_id
          JOIN
               products p ON
               oi.product_id = p.product_id
          WHERE
               o.order_status = 'delivered'
          GROUP BY 1,2
     ) as subq
WHERE
     rank = 1;

SELECT * FROM top_product_category_by_revenue_per_year;
```

--Product category name with total most cancel order each year.

```
CREATE TABLE top_product_category_by_canceled_per_year AS
SELECT
     year,
     top_product_category_by_canceled,
```

```
     canceled
FROM
     (
          SELECT
               date_part('year', o.order_purchase_timestamp) AS
year,
               p.product_category_name AS
top_product_category_by_canceled,
               COUNT(*) as canceled,
               RANK() OVER (PARTITION BY date_part('year',
o.order_purchase_timestamp)
                                   ORDER BY COUNT(*) DESC) AS rank
          FROM
               order_items oi
          JOIN
               orders o ON
               oi.order_id = o.order_id
          JOIN
               products p ON
               oi.product_id = p.product_id
          WHERE
               o.order_status = 'canceled'
          GROUP BY 1,2
     ) as subq
WHERE
     rank = 1;

SELECT * FROM top_product_category_by_canceled_per_year;

--All table
SELECT
     r.year,
     tp.top_product_category_by_revenue,
     ROUND( tp.revenue ) AS revenue,
     ROUND(r.revenue) AS total_revenue,
     tc.top_product_category_by_canceled,
     tc.canceled,
     co.canceled_order AS total_canceled_order
FROM revenue_per_year AS r
JOIN canceled_per_year AS co ON co.year = r.year
```

```
JOIN top_product_category_by_revenue_per_year AS tp ON tp.year =
r.year
JOIN top_product_category_by_canceled_per_year AS tc ON tc.year
= co.year;
```

# Analysis of Annual Payment Type Usage

--Total payment type usage of all the time sorted by the most favorite.

```
SELECT
     op.payment_type,
     date_part('year', o.order_purchase_timestamp) as year,
     COUNT(*) as total
FROM
     order_payments op
JOIN
     orders o ON
     op.order_id = o.order_id
GROUP BY 1,2
ORDER BY 2 ASC, 3 DESC;
```

--Total payment type usage each year.

```
SELECT
     payment_type,
     SUM(CASE WHEN year = 2016 THEN payment_type_usage ELSE 0
END) AS year_2016,
     SUM(CASE WHEN YEAR = 2017 THEN payment_type_usage ELSE 0
END) AS year_2017,
     SUM(CASE WHEN YEAR = 2018 THEN payment_type_usage ELSE 0
END) AS year_2018
FROM (
     SELECT
          op.payment_type,
          date_part('year', o.order_purchase_timestamp) AS year,
          COUNT(*) AS payment_type_usage
     FROM
          order_payments op
     JOIN
          orders o ON
          op.order_id = o.order_id
     GROUP BY 1,2
     ORDER BY 2 ASC, 3 DESC
     ) AS subq
```

```
GROUP BY 1
ORDER BY 2 DESC,3 DESC,4 DESC;
```