Query Analyzing eCommerce Business Performance with SQL

##Data Preparation

- create database ecommerce
  membuat tabel dari 9 data pada format csv dengan menyesuaikan tipe data setiap kolomnya.
- import data csv ke dalam database
- Membuat entity relationship antar tabel, berdasarkan skema di bawah ini. Kemudian export Entity Relationship Diagram (ERD) dalam bentuk gambar.

```
-- Customers_dataset
CREATE TABLE if not exists customers_dataset (
        customer_id varchar not null,
        customer_unique_id varchar,
        customer_zip_code_prefix int,
        customer_city varchar,
        customer_state char(5)
);
alter table customers_dataset add primary key (customer_id);
alter table customers_dataset add constraint customer_zip_code_prefix_uq unique
(customer_zip_code_prefix);

-- Geolocation_dataset
CREATE TABLE if not exists geolocation_dataset (
        geolocation_zip_code_prefix int not null,
        geolocation_lat double precision not null,
        geolocation_lng double precision not null,
        geolocation_city varchar,
        geolocation_state char(5)
);

-- Order_items_dataset
CREATE TABLE if not exists order_items_dataset (
        order_id varchar not null,
        order_item_id int not null,
        product_id varchar not null,
        seller_id varchar not null,
        shipping_limit_date TIMESTAMPTZ,
        price double precision,
        freight_value double precision
);
alter table order_items_dataset add foreign key (order_id) references orders_dataset(order_id);
alter table order_items_dataset add foreign key (product_id) references
product_dataset(product_id);
alter table order_items_dataset add foreign key (seller_id) references sellers_dataset(seller_id);
```

```sql
-- Order_payments_dataset
CREATE TABLE if not exists order_payments_dataset (
        order_id varchar not null,
        payment_sequential int,
        payment_type varchar not null,
        payment_installments int,
        payment_value double precision not null
);
alter table order_payments_dataset add foreign key (order_id) references orders_dataset(order_id);

-- Order_reviews_dataset
CREATE TABLE if not exists order_reviews_dataset (
        review_id varchar not null,
        order_id varchar,
        review_score int,
        review_comment_title varchar,
        review_comment_message varchar,
        review_creation_date TIMESTAMPTZ,
        review_answer_timestamp TIMESTAMPTZ
);
alter table order_reviews_dataset add foreign key (order_id) references orders_dataset(order_id);

-- Order_dataset
CREATE TABLE if not exists "orders_dataset" (
        order_id varchar not null,
        customer_id varchar,
        order_status varchar,
        order_purchase_timestamp TIMESTAMPTZ,
        order_approved_at TIMESTAMPTZ,
        order_delivered_carrier_date TIMESTAMPTZ,
        order_delivered_customer_date TIMESTAMPTZ,
        order_estimated_delivery_date TIMESTAMPTZ
);
alter table orders_dataset add primary key (order_id);
alter table "orders_dataset" add foreign key (customer_id) references
customers_dataset(customer_id);

-- Product_dataset
CREATE TABLE if not exists "product_dataset" (
        "index" int,
        product_id varchar,
        product_category_name varchar,
        product_name_lenght double precision,
        product_description_lenght double precision,
        product_photos_qty double precision,
        product_weight_g double precision,
        product_length_cm double precision,
        product_height_cm double precision,
        product_width_cm double precision
);
alter table "product_dataset" add primary key (product_id);
```

```
-- Sellers_dataset
CREATE TABLE if not exists "sellers_dataset" (
	seller_id varchar not null,
	seller_zip_code_prefix int,
	seller_city varchar,
	seller_state char(5)
);
alter table "sellers_dataset" add primary key (seller_id);
alter table sellers_dataset add constraint seller_zip_code_prefix_uq unique (seller_zip_code_prefix);
```

##Annual Customer Activity Growht Analysis

-- Task 1

Menampilkan rata-rata jumlah customer aktif bulanan (monthly active user) untuk setiap tahun

```
with act as (
	select Year, round(AVG(active), 0) as avg_active
	from (
		select date_part('year', od.order_purchase_timestamp) as Year,
			date_part('month', od.order_purchase_timestamp) as Month,
			count(distinct cd.customer_unique_id) as active
		from orders_dataset as od
		join customers_dataset as cd ON od.customer_id = cd.customer_id
		group by 1, 2
	) subq
	group by 1
	order by 1 ASC
),
```

-- Task 2

Menampilkan jumlah customer baru (pertama kali bertransaksi) pada masing-masing tahun

```
new_customer as(
	select
	date_part('year', first_order) as Year,
	count(1) as pelanggan_baru
	from (
		select
		cd.customer_unique_id,
		min(distinct od.order_purchase_timestamp) as first_order
		from orders_dataset as od
		join customers_dataset as cd ON od.customer_id = cd.customer_id
		group by 1
		) subq1
	group by 1
	order by 1 ASC
),
```

-- Task 3

Menampilkan jumlah customer yang melakukan pembelian lebih dari satu kali (*repeat order*) pada masing-masing tahun

```
order_customer as (
        select Year,
        count(Total_Customer) as Total_Repeat_Order
        from (
                select
                cd.customer_unique_id,
                count(1) as Total_Customer,
                date_part('year', od.order_purchase_timestamp) as Year
                from orders_dataset as od
                join customers_dataset as cd ON od.customer_id = cd.customer_id
                group by 1, 3
                having count(1) > 1
                ) subq2
        group by 1
        order by 1 ASC
),
```

-- Task 4

Menampilkan rata-rata jumlah order yang dilakukan customer untuk masing-masing tahun

```
average_order as (
        select Year,
        round(AVG(Total_Order), 2) as Avg_Order_Customer
        from (
                select
                cd.customer_unique_id,
                date_part('year', od.order_purchase_timestamp) as Year,
                count(1) as Total_Order
                from orders_dataset as od
                join customers_dataset as cd ON od.customer_id = cd.customer_id
                group by 1, 2
                ) act
        group by 1
        order by 1 ASC
)
```

--Task 5

Menggabungkan ketiga metrik yang telah berhasil ditampilkan menjadi satu tampilan tabel

```
select a.Year, a.avg_active, b.pelanggan_baru, c.Total_Repeat_Order, d.Avg_Order_Customer
from act as a
join new_customer as b on a.Year = b.Year
join order_customer as c on a.Year = c.Year
join average_order as d on a.Year = d.Year
```

##Annual Product Category Quality Analysis

-- Task 1

Membuat tabel yang berisi informasi pendapatan/revenue perusahaan total untuk masing-masing tahun

```
create table if not exists all_revenue as (
        select Year,
        round(sum(revenue)::numeric, 2) as total_revenue
        from (
                select
                od.order_status,
                ((oi.price*oi.order_item_id)+oi.freight_value) as revenue,
                date_part('year', od.order_purchase_timestamp ) as Year
                from order_items_dataset as oi
                full outer join orders_dataset as od ON oi.order_id = od.order_id
                where od.order_status != 'canceled'
                ) subq1
        group by 1
        order by 1 ASC
);
```

-- Task 2

Membuat tabel yang berisi informasi jumlah cancel order total untuk masing-masing tahun

```
create table if not exists order_canceled as (
        select
                date_part('year', order_purchase_timestamp ) as Year,
                count(order_status) as total_cancel
                from orders_dataset
                where order_status = 'canceled'
                group by 1
                order by 1 ASC
);
```

-- Task 3

Membuat tabel yang berisi nama kategori produk yang memberikan pendapatan total tertinggi untuk masing-masing tahun

```
create table if not exists max_product_category as (
        select
                Year,
                product_category_name,
                revenue
        from (
        select
                date_part('year', od.order_purchase_timestamp) as Year,
                pd.product_category_name,
                round(sum((oi.price * oi.order_item_id) + oi.freight_value)::numeric, 2) as revenue,
                rank() over(partition by date_part('year', od.order_purchase_timestamp)
                           order by sum((oi.price * oi.order_item_id) + oi.freight_value) desc) as tmp
                from order_items_dataset as oi
```

```
			join orders_dataset as od ON od.order_id = oi.order_id
			join product_dataset as pd ON pd.product_id = oi.product_id
			where od.order_status != 'canceled'
			group by 1, 2
			) sbq
		where tmp = 1
);

-- Task 4
```

Membuat tabel yang berisi nama kategori produk yang memiliki jumlah cancel order terbanyak untuk masing-masing tahun

```
create table if not exists max_product_cancel as (
		select
			Year,
			product_category_name,
			total_cancel
		from (
		select
			date_part('year', od.order_purchase_timestamp) as Year,
			pd.product_category_name,
			count(order_status) as total_cancel,
			rank() over(partition by date_part('year', od.order_purchase_timestamp)
					order by count(order_status) desc) as tmp
			from order_items_dataset as oi
			join orders_dataset as od ON od.order_id = oi.order_id
			join product_dataset as pd ON pd.product_id = oi.product_id
			where od.order_status = 'canceled'
			group by 1, 2
			) sbq
		where tmp = 1
);

-- Task 5
```

Menggabungkan informasi-informasi yang telah didapatkan ke dalam satu tampilan tabel

```
select a.Year,
		a.total_revenue,
		c.revenue,
		b.total_cancel,
		c.product_category_name as top_category_product,
		d.product_category_name as canceled_category_product
from max_product_category as c
join all_revenue as a on c.Year = a.Year
join max_product_cancel as d on c.Year = d.Year
join order_canceled as b on c.Year = b.Year
```

##Annual Payment Type Usage Analysis

– Task 1

Menampilkan jumlah penggunaan masing-masing tipe pembayaran secara all time diurutkan dari yang terfavorit

```
select
	op.payment_type,
	count(1) as num_used
from order_payments_dataset op
join orders_dataset o on o.order_id = op.order_id
group by 1
order by 2 desc
;
```

– Task 2

Menampilkan detail informasi jumlah penggunaan masing-masing tipe pembayaran untuk masing-masing tahun

```
with
tmp as (
select
	date_part('year', o.order_purchase_timestamp) as year,
	op.payment_type,
	count(1) as num_used
from order_payments_dataset op
join orders_dataset o on o.order_id = op.order_id
group by 1, 2
)

select *,
	case when year_2017 = 0 then NULL
	else round((year_2018 - year_2017) / year_2017, 2)
	end as pct_change_2017_2018
from (
select
  payment_type,
  sum(case when year = '2016' then num_used else 0 end) as year_2016,
  sum(case when year = '2017' then num_used else 0 end) as year_2017,
  sum(case when year = '2018' then num_used else 0 end) as year_2018
from tmp
group by 1) subq
order by 5 desc
```