Tugas 1

Kita dapat membuat 9 tabel di database PostgreSQL menggunakan pengetikan query SQL dan memilih type data yang cocok untuk setiap kolomnya.

```
CREATE TABLE customers_dataset(
        customer_id VARCHAR PRIMARY KEY,
        customer_unique_id VARCHAR(240),
        customer_zip_code_prefix VARCHAR(240),
        customer_city VARCHAR(100),
        customer_state VARCHAR(100)
);

CREATE TABLE geolocation_dataset(
        geolocation_zip_code_prefix VARCHAR,
        geolocation_lat VARCHAR,
        geolocation_lng VARCHAR,
        geolocation_city VARCHAR,
        geolocation_state VARCHAR
);

CREATE TABLE order_items_dataset(
        order_id VARCHAR,
        order_item_id INT,
        product_id VARCHAR,
        seller_id VARCHAR,
        shipping_limit_date TIMESTAMP,
        price real,
        freight_value real
);

CREATE TABLE order_payments_dataset(
        order_id VARCHAR,
        payment_sequential INT,
        payment_type VARCHAR,
        payment_installments INT,
        payment_value real
);

CREATE TABLE order_reviews_dataset(
        review_id VARCHAR,
        order_id VARCHAR,
        review_score INT,
        review_comment_title VARCHAR,
        review_comment_message VARCHAR,
        review_creation_date TIMESTAMP,
        review_answer_timestamp TIMESTAMP
);
```

```
CREATE TABLE orders_dataset(
	order_id VARCHAR,
	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 TIMESTAMP
);

CREATE TABLE product_dataset(
	id INT,
	product_id VARCHAR,
	product_category_name VARCHAR,
	product_name_lenght REAL,
	product_description_lenght REAL,
	product_photos_qty REAL,
	product_weight_g REAL,
	product_length_cm REAL,
	product_height_cm REAL,
	product_width_cm REAL
);

CREATE TABLE sellers_dataset(
	seller_id VARCHAR,
	seller_zip_code_prefix VARCHAR,
	seller_city VARCHAR,
	seller_state VARCHAR
);
```

Mencoba untuk menampilkan data pada masing masing tabel

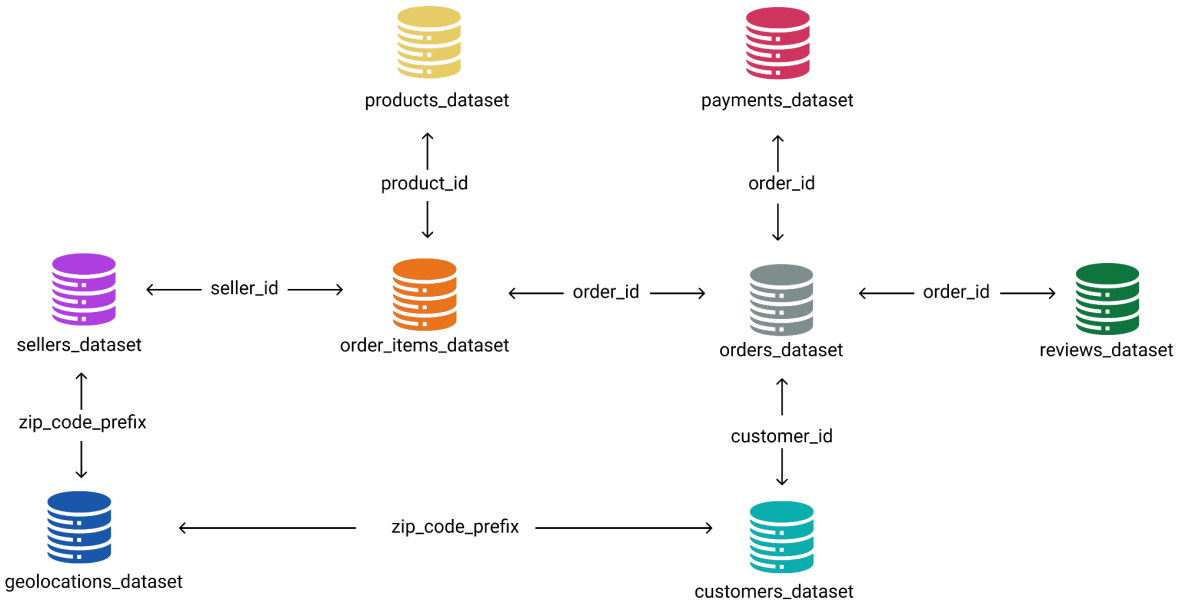
```
SELECT * FROM customers_dataset;
SELECT * FROM geolocation_dataset;
SELECT * FROM order_items_dataset;
SELECT * FROM order_payments_dataset;
SELECT * FROM order_reviews_dataset;
SELECT * FROM orders_dataset;
SELECT * FROM product_dataset;
SELECT * FROM sellers_dataset;
```

Melakukan relasi antar tabel menggunakan query

– Query relasi tabel antara orders_dataset dengan customers_dataset

```
ALTER TABLE orders_dataset
ADD CONSTRAINT customer_id
FOREIGN KEY (customer_id) REFERENCES customers_dataset(customer_id);


– Query relasi tabel antara customers_dataset dengan geolocation_dataset
ALTER TABLE customers_dataset
ADD CONSTRAINT geolocation_zip_code_prefix
FOREIGN KEY (customer_zip_code_prefix) REFERENCES
geolocation_dataset(geolocation_zip_code_prefix);

-- Query relasi tabel antara sellers_dataset dengan geolocation_dataset
ALTER TABLE sellers_dataset
ADD CONSTRAINT geolocation_zip_code_prefix
FOREIGN KEY (seller_zip_code_prefix) REFERENCES
geolocation_dataset(geolocation_zip_code_prefix);

-- Query relasi tabel antara order_items_dataset dengan sellers_dataset
ALTER TABLE order_items_dataset
ADD CONSTRAINT seller_id
FOREIGN KEY (seller_id) REFERENCES sellers_dataset(seller_id);

-- Query relasi tabel antara order_items_dataset dengan product_dataset
ALTER TABLE order_items_dataset
ADD CONSTRAINT product_id
FOREIGN KEY (product_id) REFERENCES product_dataset(product_id);

-- Query relasi tabel antara order_reviews_dataset dengan orders_dataset
ALTER TABLE order_reviews_dataset
ADD CONSTRAINT order_id
FOREIGN KEY (order_id) REFERENCES orders_dataset(order_id);

-- Query relasi tabel antara order_payments_dataset dengan orders_dataset
ALTER TABLE order_payments_dataset
ADD CONSTRAINT order_id
FOREIGN KEY (order_id) REFERENCES orders_dataset(order_id);

-- Query relasi tabel antara order_items_dataset dengan orders_dataset
ALTER TABLE order_items_dataset
ADD CONSTRAINT order_id
FOREIGN KEY (order_id) REFERENCES orders_dataset(order_id);
```

Setelah melakukan query, didapatkan bahwa geolocation_zip_code_index berisi data yang duplikat. Maka dari itu, kita dapat menambahkan line khusus pada ERG untuk membentuk seperti gambar yang diminta.


products_dataset
payments_dataset
product_id
order_id
seller_id
order_id
order_id
sellers_dataset
order_items_dataset
orders_dataset
reviews_dataset
zip_code_prefix
customer_id
zip_code_prefix
geolocations_dataset
customers_dataset

Tugas 2

1. Menampilkan rata-rata jumlah customer aktif bulanan (monthly active user) untuk setiap tahun (Hint: Perhatikan kesesuaian format tanggal)

```
SELECT
	tahun,
	ROUND(AVG(monthly_active_user), 2) AS MAU
FROM(
	SELECT
		date_part('year', orders.order_purchase_timestamp) AS tahun,
		date_part('month', orders.order_purchase_timestamp) AS bulan,
		COUNT(DISTINCT customers.customer_unique_id) AS monthly_active_user
	FROM
		orders_dataset orders
	JOIN customers_dataset customers
	ON customers.customer_id = orders.customer_id
	GROUP BY 1,2
) AS cnt_mau
GROUP BY 1;
```

Langkah pertama dapat kita lakukan dengan mencari total customer yang aktif tiap bulan dan tahunnya pada bagian subquery. Kemudian kita dapat hitung total customer dengan fungsi COUNT() menggunakan kolom customer_unique_id di tabel customers. Oleh karena kolom customer_unique_id pada tabel customer tidak bersifat unik paka ditambahkan fungsi DISTINCT. Kemudian, kita dapat melakukan pencarian untuk jumlah rata-rata customer aktif tiap tahunnya menggunakan fungsi agregat AVG().

	tahun double precision	mau numeric
1	2016	108.67
2	2017	3694.83
3	2018	5338.20

2. Menampilkan jumlah customer baru pada masing-masing tahun (Hint: Pelanggan baru adalah pelanggan yang melakukan order pertama kali)

```
SELECT
	date_part('year', first_order_time) AS tahun,
	COUNT(*) AS total_customers
```

```
FROM(
      SELECT
            customers.customer_unique_id,
            MIN(orders.order_purchase_timestamp) AS
first_order_time
      FROM
            customers_dataset customers
      JOIN
            orders_dataset orders
      ON customers.customer_id = orders.customer_id
      GROUP BY 1
) cnt_first_order
GROUP BY 1
ORDER BY tahun;
```

Langkah awal cari customer yang melakukan order / pembelian pertama kali dan kapan tanggal pembelian pertama customer dilakukan pada bagian subquery. Untuk mencari tanggal customer melakukan pembelian dapat menggunakan fungsi MIN(). Kemudian Hitung total customer yang melakukan pembelian pertama kali tiap tahunnya. Untuk menghitung total jumlah customer melakukan pembelian pertama kali bisa menggunakan fungsi COUNT().

	tahun double precision	total_customers bigint
1	2016	326
2	2017	43708
3	2018	52062

3. Menampilkan jumlah customer yang melakukan pembelian lebih dari satu kali (repeat order) pada masing-masing tahun (Hint: Pelanggan yang melakukan repeat order adalah pelanggan yang melakukan order lebih dari 1 kali)

```
SELECT
      tahun,
      COUNT(customer_unique_id) AS total_repeat_customers
FROM(
      SELECT
            date_part('year', orders.order_purchase_timestamp)
AS tahun,
            customers.customer_unique_id,
            count(*) AS total_orders
      FROM
```

```
            orders_dataset orders
      JOIN
            customers_dataset customers
      ON orders.customer_id = customers.customer_id
      GROUP BY 1,2
      HAVING count(*) > 1
) repeat_order_total
GROUP BY 1
ORDER BY 1;
```

Langkah pertama mencari customer mana saja yang melakukan order lebih dari 1 kali pada bagian subquery. Untuk mencari order lebih dari 1 kali dapat menggunakan fungsi HAVING() dan beri perbandingan nilai lebih dari 1. Kemudian hitung total customer yang melakukan pembelian lebih dari satu kali tiap tahunnya.

	tahun double precision	total_repeat_customers bigint
1	2016	3
2	2017	1256
3	2018	1167

4. Menampilkan rata-rata jumlah order yang dilakukan customer untuk masing-masing tahun (Hint: Hitung frekuensi order (berapa kali order) untuk masing-masing customer terlebih dahulu)

```
SELECT
      tahun,
      ROUND(AVG(total_orders),2) AS average_order
FROM(
      SELECT
            date_part('year', orders.order_purchase_timestamp)
AS tahun,
            customers.customer_unique_id,
            count(*) AS total_orders
      FROM
            orders_dataset orders
      JOIN customers_dataset customers
      ON customers.customer_id = orders.customer_id
      GROUP BY 1,2
) AS cnt_total_orders
GROUP BY 1
ORDER BY 1;
```

Pertama fokus pada bagian subquery, di sana kita akan mencari customer dan total order yang dilakukan per tahunnya. Kemudian hitung rata-rata order per customer menggunakan fungsi agregat AVG() dari data total_orders per tahunnya

	tahun double precision	average_order numeric
1	2016	1.01
2	2017	1.03
3	2018	1.02

5. Menggabungkan ketiga metrik yang telah berhasil ditampilkan menjadi satu tampilan tabel

```
WITH calculate_mau AS(
      SELECT
      tahun,
      ROUND(AVG(monthly_active_user), 2) AS MAU
      FROM(
            SELECT
                  date_part('year',
orders.order_purchase_timestamp) AS tahun,
                  date_part('month',
orders.order_purchase_timestamp) AS bulan,
                  COUNT(DISTINCT customers.customer_unique_id)
AS monthly_active_user
            FROM
                  orders_dataset orders
            JOIN customers_dataset customers
            ON customers.customer_id = orders.customer_id
            GROUP BY 1,2
      ) AS cnt_mau
      GROUP BY 1
),
calculate_new_customers AS (
      SELECT
            date_part('year', first_order_time) AS tahun,
            COUNT(*) AS total_customers
      FROM(
            SELECT
                  customers.customer_unique_id,
                  MIN(orders.order_purchase_timestamp) AS
first_order_time
            FROM
                  customers_dataset customers
            JOIN
                  orders_dataset orders
            ON customers.customer_id = orders.customer_id
```

```
            GROUP BY 1
      ) cnt_first_order
      GROUP BY 1
),
calculate_repeat_order AS (
      SELECT
            tahun,
            COUNT(customer_unique_id) AS
total_repeat_customers
      FROM(
            SELECT
                  date_part('year',
orders.order_purchase_timestamp) AS tahun,
                  customers.customer_unique_id,
                  count(*) AS total_orders
            FROM
                  orders_dataset orders
            JOIN
                  customers_dataset customers
            ON orders.customer_id = customers.customer_id
            GROUP BY 1,2
            HAVING count(*) > 1
      ) repeat_order_total
      GROUP BY 1
),
calculate_average_orders_customer AS (
      SELECT
            tahun,
            ROUND(AVG(total_orders),2) AS average_order
      FROM(
            SELECT
                  date_part('year',
orders.order_purchase_timestamp) AS tahun,
                  customers.customer_unique_id,
                  count(*) AS total_orders
            FROM
                  orders_dataset orders
            JOIN customers_dataset customers
            ON customers.customer_id = orders.customer_id
            GROUP BY 1,2
      ) AS cnt_total_orders
      GROUP BY 1
)

SELECT
      mau.tahun,
      mau.MAU,
      cal_new_cust.total_customers,
```

```
      cal_repeat_ord.total_repeat_customers,
      cal_avg_cust.average_order
FROM
      calculate_mau mau
JOIN calculate_new_customers cal_new_cust ON mau.tahun =
cal_new_cust.tahun
JOIN calculate_repeat_order cal_repeat_ord ON mau.tahun =
cal_repeat_ord.tahun
JOIN calculate_average_orders_customer cal_avg_cust ON
mau.tahun = cal_avg_cust.tahun
```

Pertama buatlah Common Table Experssion (CTE) dari hasil pengerjaan tugas 1 - 4. Untuk membuat CTE gunakan fungsi WITH. kemudian beri nama alias untuk masing-masing table temporarynya. Setelah itu tampilkan data tahun, MAU, total customer baru, total customer yang melakukan repeat order, dan rata-rata order per customer menggunakan fungsi JOIN dari beberapa table temporary.

	tahun double precision	avg_mau numeric	total_customers bigint	total_repeat_customers bigint	average_order numeric
1	2016	108.67	326	3	1.01
2	2017	3694.83	43708	1256	1.03
3	2018	5338.20	52062	1167	1.02

Tugas 3

1. Membuat tabel yang berisi informasi pendapatan/revenue perusahaan total untuk masing-masing tahun (Hint: Revenue adalah harga barang dan juga biaya kirim. Pastikan juga melakukan filtering terhadap order status yang tepat untuk menghitung pendapatan)

```
CREATE TABLE total_revenue_company_per_year AS
SELECT
        date_part('year', orders.order_purchase_timestamp) AS year,
        SUM(revenue_order) AS revenue
FROM(
        SELECT
                order_id,
                SUM(price+freight_value) AS revenue_order
        FROM
                order_items_dataset
        GROUP BY 1
) cal_revenue_order
JOIN
        orders_dataset AS orders
ON cal_revenue_order.order_id = orders.order_id
WHERE
        orders.order_status = 'delivered'
GROUP BY 1
ORDER BY 1;
```

	year double precision	revenue real
1	2016	46653.74
2	2017	6.9213715e+06
3	2018	8.451628e+06

Pengerjaan soal ini dimulai dari membuat table baru menggunakan subquery. Total revenue = price+freight value. Kita juga harus memfilter dimana status pengiriman = ‘delivered.’ Setelah itu, perlu dilakukan group by dan order by untuk mengurutkan tahun. Tabel yang sudah dibuat dapat dilihat dengan menggunakan select *.

2. Membuat tabel yang berisi informasi jumlah cancel order total untuk masing-masing tahun (Hint: Perhatikan filtering terhadap order status yang tepat untuk menghitung jumlah cancel order)

```
CREATE TABLE total_order_cancel_per_year AS
SELECT
        date_part('year', order_purchase_timestamp) AS year,
        COUNT(*) AS total_cancel_order
FROM
        orders_dataset
WHERE
```

```
        order_status = 'canceled'
GROUP BY 1
ORDER BY 1;
```

	year double precision	total_cancel_order bigint
1	2016	26
2	2017	265
3	2018	334

Pertama cari total order cancel yang status ordernya ‘canceled’ per tahunnya menggunakan fungsi COUNT(). Kemudian buat table total order cancel per year menggunakan fungsi CREATE TABLE (name_table) AS

3. Membuat tabel yang berisi nama kategori produk yang memberikan pendapatan total tertinggi untuk masing-masing tahun (Hint: Perhatikan penggunaan window function dan juga filtering yang dilakukan)

```
CREATE TABLE highest_product_category_per_year AS
SELECT
        year,
        product_category_name,
        revenue_product
FROM(
        SELECT
                date_part('year', orders.order_purchase_timestamp) AS year,
                products.product_category_name,
                SUM(order_items.price+order_items.freight_value) AS revenue_product,
                RANK() OVER (PARTITION BY date_part('year',
orders.order_purchase_timestamp) ORDER BY
SUM(order_items.price+order_items.freight_value) DESC) AS the_rank
        FROM
                order_items_dataset AS order_items
        JOIN
                orders_dataset AS orders ON order_items.order_id = orders.order_id
        JOIN
                product_dataset AS products ON order_items.product_id =
products.product_id
        WHERE orders.order_status = 'delivered'
        GROUP BY 1,2
) AS calt_revenue_product
WHERE the_rank = 1;
```

	year double precision	product_category_name character varying	revenue_product real
1	2016	furniture_decor	6899.3496
2	2017	bed_bath_table	580949.1
3	2018	health_beauty	866809.4

Pertama buat subquery terlebih dahulu untuk mencari total revenue product berdasarkan tahun dan nama productnya yang berstatus delivered. Untuk bisa mendapatkan nama product category dilakukan JOIN table product_dataset. Digunakan juga window function RANK untuk mendapatkan peringkat kategori produk berdasarkan tahun dan diurutkan dari terbesar ke terkecil nilai revenuenya. Tampilkan product category name, total revenue product tiap tahunnya. Buat table top product category per year menggunakan fungsi CREATE TABLE (name_table) AS

4. Membuat tabel yang berisi nama kategori produk yang memiliki jumlah cancel order terbanyak untuk masing-masing tahun (Hint: Perhatikan penggunaan window function dan juga filtering yang dilakukan)

```
CREATE TABLE highest_product_category_canceled AS
SELECT
        year,
        product_category_name,
        total_product_cancel
FROM(
        SELECT
                date_part('year', orders.order_purchase_timestamp) AS year,
                products.product_category_name,
                COUNT(*) AS total_product_cancel,
                RANK() OVER (PARTITION BY date_part('year',
orders.order_purchase_timestamp) ORDER BY COUNT(*) DESC) AS the_rank2
        FROM
                order_items_dataset AS order_items
        JOIN
                orders_dataset AS orders ON order_items.order_id = orders.order_id
        JOIN
                product_dataset AS products ON order_items.product_id =
products.product_id
        WHERE orders.order_status = 'canceled'
        GROUP BY 1,2
) AS calt_revenue_cancel_product
WHERE the_rank2 = 1;
```

	year double precision	product_category_name character varying	total_product_cancel bigint
1	2016	toys	3
2	2017	sports_leisure	25
3	2018	health_beauty	27

Langkah pertama buatlah subquery untuk mencari jumlah product category yang berstatus canceled berdasarkan setiap nama product cateogry dan tahun yang querynya hampir sama dengan query soal sebelumnya. Digunakan juga window function RANK untuk mendapatkan peringkat kategori produk berdasarkan tahun dan diurutkan dari terbesar ke terkecil berdasarkan jumlah produk tercanceled. Tampilkan produk kategori dan total produk cancel tiap tahunnya. Buat table top product category canceled.

5. Menggabungkan informasi-informasi yang telah didapatkan ke dalam satu tampilan tabel (Hint: Perhatikan teknik join yang dilakukan serta kolom-kolom yang dipilih)

```
SELECT
        a.year,
        a.revenue AS total_revenue,
        b.total_cancel_order AS number_cancel_order,
        c.product_category_name AS top_product_category_name,
        c.revenue_product AS revenue_top_product,
        d.product_category_name AS top_product_category_name_canceled,
        d.total_product_cancel AS num_top_product_category_canceled
FROM
        total_revenue_company_per_year AS a
JOIN
        total_cancel_per_year b on b.year = a.year
JOIN
        highest_product_category_per_year c ON a.year = c.year
JOIN
        highest_product_category_cancelled_per_year d ON a.year = d.year
```

	year double precision	total_revenue real	number_cancel_order bigint	top_product_category_name character varying (250)	revenue_top_product real	top_product_category_name_canceled character varying (250)	num_top_product_category_canceled bigint
1	2016	46653.74	26	furniture_decor	6899.3496	toys	3
2	2017	6.9213715e+06	265	bed_bath_table	580949.75	sports_leisure	25
3	2018	8.451628e+06	334	health_beauty	866810	health_beauty	27

Pada soal ini kita dapat menggabungkan semua tabel di atas yang sudah dibuat dengan menggunakan fungsi JOIN saja.

Tugas 4

1. Menampilkan jumlah penggunaan masing-masing tipe pembayaran secara all time diurutkan dari yang terfavorit (Hint: Perhatikan struktur (kolom-kolom apa saja) dari tabel akhir yang ingin didapatkan)

```
SELECT
        payment_type,
        COUNT(*) AS total_users
FROM
        order_payments_dataset
GROUP BY 1
ORDER BY 2 DESC;
```

	payment_type character varying	total_users bigint
1	credit_card	76795
2	boleto	19784
3	voucher	5775
4	debit_card	1529
5	not_defined	3

2. Menampilkan detail informasi jumlah penggunaan masing-masing tipe pembayaran untuk setiap tahun (Hint: Perhatikan struktur (kolom-kolom apa saja) dari tabel akhir yang ingin didapatkan)

```
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;
```

	payment_type character varying (250)	year_2016 numeric	year_2017 numeric	year_2018 numeric	pct_change_2017_2018 numeric
1	not_defined	0	0	3	[null]
2	debit_card	2	422	1105	1.62
3	credit_card	258	34568	41969	0.21
4	boleto	63	9508	10213	0.07
5	voucher	23	3027	2725	-0.10