**TASK 1**

**Subtask 1:**

Membuat database baru beserta tabel-tabelnya

1. Membuat Database **ecommerce**

create database ecommerce;

1. Membuat tabel dari data csv
2. Tabel Customer

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)

);

1. Tabel Geolocation

create table geolocation (

geo\_zip\_code\_prefix varchar(250),

geo\_lat varchar(250),

geo\_lng varchar(250),

geo\_city varchar(250),

geo\_state varchar(250)

);

1. Tabel Orders

**order\_purchase\_timestamp, order\_approved\_at,** **order\_delivered\_carrier\_date, order\_delivered\_customer\_date, order\_estimated\_delivered\_date** dirubah dulu menjadi text baru dirubah menjadi timestamp without time zone karena jika dibuat menjadi timestamp without time zone dulu maka terjadi error saat Import

create table orders (

order\_id varchar(250),

customers\_id varchar(250),

order\_status varchar(250),

order\_purchase\_timestamp text,

order\_approved\_at text,

order\_delivered\_carrier\_date text,

order\_delivered\_customer\_date text,

order\_estimated\_delivered\_date text

);

1. Tabel Order Item

**shipping\_limit\_date** dirubah dulu menjadi text baru dirubah menjadi timestamp without time zone karena jika dibuat menjadi timestamp without time zone dulu maka terjadi error saat Import

create table order\_item (

order\_id varchar(250),

order\_item\_id int,

product\_id varchar(250),

seller\_id varchar(250),

shipping\_limit\_date text,

price float,

freight\_value float

);

1. Tabel Payments

create table payments (

order\_id varchar(250),

payment\_sequential int,

payment\_type varchar(250),

payment\_installment int,

payment\_value float

);

1. Tabel Reviews

**review\_creation\_date, review\_answer** dirubah dulu menjadi text baru dirubah menjadi timestamp without time zone karena jika dibuat menjadi timestamp without time zone dulu maka terjadi error saat Import

create table 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 without time zone,

review\_answer timestamp without time zone

);

1. Table Product

create table products (

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

);

1. Tabel Sellers

create table sellers (

seller\_id varchar(250),

seller\_zip\_code int,

seller\_city varchar(250),

seller\_state varchar(250)

);

**Subtask 2:**

Importing data csv ke dalam database

Sebenarnya bias dilakukan Import data secara langsung tetapi jika ukuran data besar (lebih dari 8 MB) maka akan terjadi error, untuk itu dilakukan import data dengan cara **COPY** dan bias digunakan untuk ukuran data yang besar. Import data ini akan error jika ada data yang berbentuk tanda petik dua (“ ”) yang digunakan pemisah antar kolom

1. Import data customers

COPY public."customers" FROM 'D:\Mini Project Bootcamp Data Science\SQL\Dataset Filter\customers.csv'

DELIMITER ',' CSV HEADER;

1. Import data geolocation

COPY public."geolocation" FROM 'D:\Mini Project Bootcamp Data Science\SQL\Dataset Filter\geolocation.csv'

DELIMITER ',' CSV HEADER;

1. Import data order\_item

COPY public."order\_item" FROM 'D:\Mini Project Bootcamp Data Science\SQL\Dataset Filter\order\_items.csv'

DELIMITER ',' CSV HEADER;

* Membuat kolom baru terlebih dahulu sebelum mengganti type data pada kolom shipping\_limit\_date

ALTER TABLE order\_item

ADD COLUMN shipping timestamp without time zone

* Mengisi nilai kolom baru

UPDATE order\_item SET shipping = to\_timestamp(shipping\_limit\_date, 'MM/DD/YYYY HH24:MI:SS')

* Menghapus kolom lama dan mengubah nama kolom baru menjadi nama kolom lama

ALTER TABLE order\_item DROP COLUMN shipping\_limit\_date;

ALTER TABLE order\_item RENAME COLUMN shipping TO shipping\_limit\_date

1. Import data orders

COPY public."orders" FROM 'D:\Mini Project Bootcamp Data Science\SQL\Dataset Filter\orders.csv'

DELIMITER ',' CSV HEADER;

* Membuat kolom baru terlebih dahulu sebelum mengganti type data pada kolom **order\_purchase\_timestamp, order\_approved\_at,** **order\_delivered\_carrier\_date, order\_delivered\_customer\_date, order\_estimated\_delivered\_date**

ALTER TABLE orders

ADD COLUMN order\_purchase timestamp without time zone,

ADD COLUMN order\_approved timestamp without time zone,

ADD COLUMN order\_delivered\_carrier timestamp without time zone,

ADD COLUMN order\_delivered\_customer timestamp without time zone,

ADD COLUMN order\_estimated timestamp without time zone

* Mengisi nilai kolom baru

UPDATE orders SET order\_purchase = to\_timestamp(order\_purchase\_timestamp, 'MM/DD/YYYY HH24:MI:SS')

UPDATE orders SET order\_approved = to\_timestamp(order\_approved\_at, 'MM/DD/YYYY HH24:MI:SS')

UPDATE orders SET order\_delivered\_carrier = to\_timestamp(order\_delivered\_carrier\_date, 'MM/DD/YYYY HH24:MI:SS')

UPDATE orders SET order\_delivered\_customer = to\_timestamp(order\_delivered\_customer\_date, 'MM/DD/YYYY HH24:MI:SS')

UPDATE orders SET order\_estimated = to\_timestamp(order\_estimated\_delivered\_date, 'MM/DD/YYYY HH24:MI:SS')

* Menghapus kolom lama dan mengubah nama kolom baru menjadi nama kolom lama

ALTER TABLE orders DROP COLUMN order\_purchase\_timestamp;

ALTER TABLE orders RENAME COLUMN order\_purchase TO order\_purchase\_timestamp

ALTER TABLE orders DROP COLUMN order\_approved\_at;

ALTER TABLE orders RENAME COLUMN order\_approved TO order\_approved\_at

ALTER TABLE orders DROP COLUMN order\_delivered\_carrier\_date;

ALTER TABLE orders RENAME COLUMN order\_delivered\_carrier TO order\_delivered\_carrier\_date

ALTER TABLE orders DROP COLUMN order\_delivered\_customer\_date;

ALTER TABLE orders RENAME COLUMN order\_delivered\_customer TO order\_delivered\_customer\_date

ALTER TABLE orders DROP COLUMN order\_estimated\_delivered\_date;

ALTER TABLE orders RENAME COLUMN order\_estimated TO order\_estimated\_delivered\_date

1. Import data payments

COPY public."payments" FROM 'D:\Mini Project Bootcamp Data Science\SQL\Dataset Filter\payments.csv'

DELIMITER ',' CSV HEADER;

1. Import data products

COPY public."products" FROM 'D:\Mini Project Bootcamp Data Science\SQL\Dataset Filter\products.csv'

DELIMITER ',' CSV HEADER;

1. Import data reviews

Export data dari PostgreSQL ke data D

COPY public."reviews\_dataset" TO 'D:\reviews\_dataset.csv' DELIMITER ',' CSV HEADER

Import data

COPY public."reviews" FROM 'D:\Mini Project Bootcamp Data Science\SQL\Dataset Filter\reviews.csv'

DELIMITER ',' CSV HEADER;

1. Import data sellers

COPY public."sellers" FROM 'D:\Mini Project Bootcamp Data Science\SQL\Dataset Filter\sellers.csv'

DELIMITER ',' CSV HEADER;

**Subtask 3**

Membuat entity relationship antar table

Sebelum membuat relasi, perlu dibuat terlebih dahulu Primary Key pada table karena sebelumnya belum dibuat

1. Memuat Primary Key pada Tabel customers

ALTER TABLE customers

ADD CONSTRAINT pk\_customers PRIMARY KEY (customer\_id);

1. Memuat Primary Key pada Tabel geolocation

* Hapus Data Duplikat

DELETE FROM geolocation

WHERE geo\_zip\_code\_prefix IN (

SELECT geo\_zip\_code\_prefix

FROM geolocation

GROUP BY geo\_zip\_code\_prefix

HAVING COUNT(\*) > 1

);

* Membuat Primay Key

ALTER TABLE geolocation

ADD CONSTRAINT pk\_geolocation PRIMARY KEY (geo\_zip\_code\_prefix);

1. Memuat Primary Key pada Tabel orders

ALTER TABLE orders

ADD CONSTRAINT pk\_orders PRIMARY KEY (order\_id);

1. Memuat Primary Key pada Tabel products

ALTER TABLE products

ADD CONSTRAINT pk\_products PRIMARY KEY (product\_id);

1. Memuat Primary Key pada Tabel reviews

ALTER TABLE reviews

ADD CONSTRAINT pk\_reviews PRIMARY KEY (review\_id);

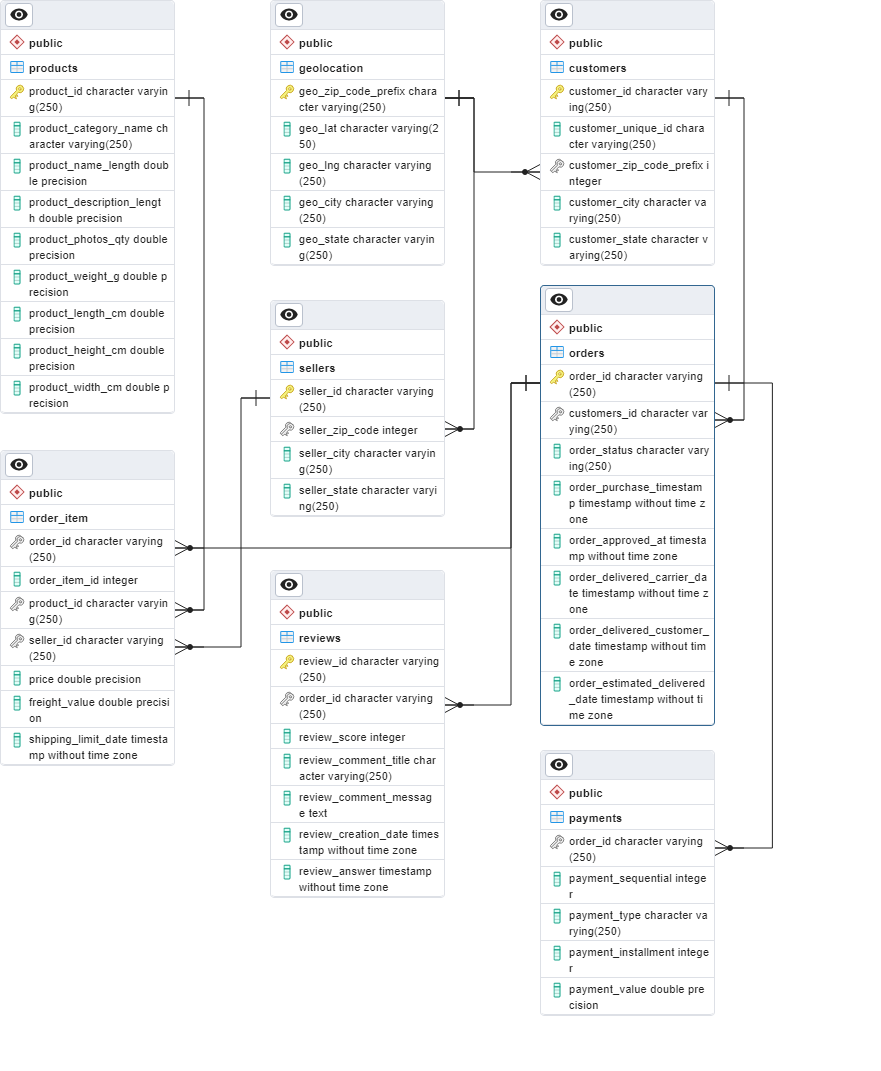
1. Memuat Primary Key pada Tabel sellers

ALTER TABLE sellers

ADD CONSTRAINT pk\_sellers PRIMARY KEY (seller\_id);

Entity Relationship Antar Tabel

[Link ERD](https://drive.google.com/file/d/1fW2CiLZ2ep7Q1usnkhrSyQCkN_XoTVEF/view?usp=sharing)



Berdasarkan gambar ERD diatas dapat dilihat bahwa setiap kolom pada suatu dataset memiliki relasi / hubungan dengan kolom di dataset lainnya. Hal tersebut menunjukkan bahwa, kolom tersebut menjadi kolom kunci yang menghubungkan dataset dengan dataset lainnya.

Sebagai contoh, dalam dataset order\_item terdapat kolom **product\_id** (Foreign Key / warna abu-abu) berhubungan dengan kolom **product\_id** di dataset product (Primary Key / warna kuning). Ini menunjukkan bahwa setiap pesanan item / order\_item terkait dengan product tertentu melalui **ID Product**.

**Primary Key (Kunci Utama)**

**Foreign Key (Kunci Asing)**

**TASK 2**

**Subtask 1:**

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

SELECT

YEAR,

round(AVG(rata\_rata), 2) AS average\_customer

FROM (

SELECT

EXTRACT(YEAR FROM order\_purchase\_timestamp) AS YEAR,

EXTRACT(MONTH FROM order\_purchase\_timestamp) AS MONTH,

count(distinct C.customer\_unique\_id) AS rata\_rata

FROM orders AS O

JOIN customers AS C

ON O.customers\_id = C.customer\_id

GROUP BY 1,2

) subq

GROUP BY 1

Langkah pertama yang dilakukan adalah membuat tabel sementara dengan *subquery* di *from* *clause* (diwarnai kuning). Tabel sementara tersebut menghitung jumlah customer unik yang aktif melakukan order setiap bulannya, jumlah customer unik inilah yang disebut dengan *Monthly Active User*. Fungsi **EXTRACT** digunakan untuk melakukan ekstraksi komponen *timestamp*, dalam kasus ini, komponen *year* (tahun) dan *month* (bulan) yang diekstrak sesuai dengan kebutuhan. Setelah mendapatkan angka *Monthly Active User* setiap bulan, dapat dilakukan agregasi yang lebih besar untuk mendapatkan rata-rata *Monthly Active User* untuk masing-masing tahun, karena itu dilakukan perhitungan *avg* (rata-rata) yang dikelompokkan berdasarkan *year* (tahun).

**Subtask 2:**

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

SELECT

date\_part('year', first\_purchase\_time) AS YEAR,

COUNT(1) AS new\_customers

FROM (

SELECT

C.customer\_unique\_id,

MIN(O.order\_purchase\_timestamp)AS first\_purchase\_time

FROM orders AS O

JOIN customers AS C

ON C.customer\_id = O.customers\_id

GROUP by 1

) subq

GROUP by 1

Langkah pertama yang dilakukan adalah membuat tabel sementara dengan *subquery* di *from* *clause* (diwarnai kuning). Tabel sementara tersebut berisi informasi tanggal pertama kali order (*first purchase timestamp*) untuk masing-masing customer menggunakan fungsi agregasi MIN pada kolom **order\_purchase\_timestamp** untuk mendapatkan tanggal terkecil dari masing-masing customer. Setelah mendapatkan tanggal order pertama untuk setiap customer, maka langkah berikutnya bisa dilakukan ekstraksi tahun menggunakan date\_part, kemudian menghitung ada berapa jumlah customer untuk masing-masing tahun tersebut menggunakan fungsi agregasi COUNT.

**Subtask 3:**

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

SELECT

YEAR,

COUNT(distinct customer\_unique\_id) AS repeat\_orders

FROM (

SELECT

date\_part('year', O.order\_purchase\_timestamp)AS YEAR,

C.customer\_unique\_id,

COUNT(1)

FROM orders AS O

JOIN customers AS C

ON C.customer\_id = O.customers\_id

GROUP BY 1, 2

HAVING COUNT(1) > 1

) subq

GROUP BY 1

Langkah pertama, tabel sementara yang dibuat (query yang diwarnai kuning) menunjukkan banyak order yang dilakukan masing-masing customer pada tahun tertentu. Hal ini dilakukan dengan mengelompokkan perhitungan COUNT berdasarkan tahun dan customer\_unique\_id. Karena tujuannya adalah untuk mendapatkan customer yang melakukan *repeat order*, maka digunakan filter HAVING untuk mendapatkan customer yang melakukan order lebih dari 1 diikuti dengan agregat COUNT yang dikelompokkan berdasarkan tahun. Jadi **HAVING COUNT(1) > 1** dilakukan untuk menghitung ada berapa banyak customer yang melakukan repeat order (order lebih dari 1) pada masing-masing tahun.

**Subtask 4:**

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

SELECT

YEAR,

round(AVG(frequency\_purchase),3)AS avg\_orders\_per\_customers

FROM (

SELECT

date\_part('year', O.order\_purchase\_timestamp) AS YEAR,

C.customer\_unique\_id,

COUNT(1) AS frequency\_purchase

FROM orders AS O

JOIN customers C

ON C.customer\_id = O.customers\_id

GROUP BY 1,2

)subq

GROUP BY 1

Tabel sementara yang dibuat kali ini (query diwarnai kuning) mirip dengan yang telah dibuat pada subtask 3, hanya saja tidak dilakukan filter HAVING karena tujuannya adalah untuk mengitung rata-rata order yang dilakukan masing-masing customer setiap tahunnya. Setelah tabel yang berisi informasi frekuensi order untuk masing-masing customer setiap tahun, maka agregasi lebih lanjut bisa dilakukan dengan melakukan AVG pada kolom frekuensi yang kemudian dikelompokkan berdasarkan tahun. Dengan begitu, angka rata-rata frekuensi order untuk masing-masing customer di setiap tahunnya berhasil didapatkan.

**Subtask 5:**

Menggabungkan keempat metrik yang telah berhasil ditampilkan menjadi satu tampilan tabel

Menggabungkan keempat metrik menggunakan *Common Table Expression* (CTE). Kemudian, untuk menggabungkan semua hasilnya dapat dilakukan join berdasarkan year (tahun), karena semua *query* dari subtask 1-4 sudah dikelompokkan berdasarkan tahun.

with

avg\_customer AS (

SELECT

YEAR,

round(AVG(rata\_rata), 2) AS average\_customer

FROM (

SELECT

EXTRACT(YEAR FROM order\_purchase\_timestamp) AS YEAR,

EXTRACT(MONTH FROM order\_purchase\_timestamp) AS MONTH,

count(distinct C.customer\_unique\_id) AS rata\_rata

FROM orders AS O

JOIN customers AS C

ON O.customers\_id = C.customer\_id

GROUP BY 1,2

) subq

GROUP BY 1

),

new\_cust AS (

SELECT

date\_part('year', first\_purchase\_time) AS YEAR,

COUNT(1) AS new\_customers

FROM (

SELECT

C.customer\_unique\_id,

MIN(O.order\_purchase\_timestamp)AS first\_purchase\_time

FROM orders AS O

JOIN customers AS C

ON C.customer\_id = O.customers\_id

GROUP by 1

) subq

GROUP by 1

),

repeat\_ordr AS (

SELECT

YEAR,

COUNT(distinct customer\_unique\_id) AS repeat\_orders

FROM (

SELECT

date\_part('year', O.order\_purchase\_timestamp)AS YEAR,

C.customer\_unique\_id,

COUNT(1)

FROM orders AS O

JOIN customers AS C

ON C.customer\_id = O.customers\_id

GROUP BY 1, 2

HAVING COUNT(1) > 1

) subq

GROUP BY 1

),

avg\_orders AS (

SELECT

YEAR,

round(AVG(frequency\_purchase),3)AS avg\_orders\_per\_customers

FROM (

SELECT

date\_part('year', O.order\_purchase\_timestamp) AS YEAR,

C.customer\_unique\_id,

COUNT(1) AS frequency\_purchase

FROM orders AS O

JOIN customers C

ON C.customer\_id = O.customers\_id

GROUP BY 1,2

)subq

GROUP BY 1

)

SELECT

rata\_rata.YEAR,

rata\_rata.average\_customer,

newcust.new\_customers,

rept.repeat\_orders,

avg\_odr.avg\_orders\_per\_customers

FROM avg\_customer AS rata\_rata

JOIN new\_cust AS newcust

ON rata\_rata.YEAR = newcust.YEAR

JOIN repeat\_ordr AS rept

ON rept.YEAR = rata\_rata.YEAR

JOIN avg\_orders AS avg\_odr

ON avg\_odr.YEAR = rata\_rata.year

**TASK 3**

**Subtask 1:**

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

CREATE TABLE total\_revenue\_per\_year AS

SELECT

EXTRACT (YEAR FROM O.order\_purchase\_timestamp) AS YEAR,

SUM(revenue\_per\_order) AS revenue

FROM (

SELECT

order\_id,

SUM(price+freight\_value) AS revenue\_per\_order

FROM order\_item

GROUP BY 1

) subq

JOIN orders AS O

ON subq.order\_id = O.order\_id

WHERE O.order\_status = 'delivered'

GROUP BY 1

Untuk membuat tabel baru berdasarkan query tertentu dapat digunakan command create table <nama\_tabel> as <query>. Untuk menghitung pendapatan total (*revenue*) dilakukan penjumlahan sederhana terhadap harga barang (price) dan juga biaya kirim (freight\_value), tidak lupa juga dilakukan filter untuk mendapatkan pesanan-pesanan yang benar-benar sudah terkirim (order\_status = ‘delivered’). Setelah itu, dilakukan pengelompokkan berdasarkan tahun.

**Subtask 2:**

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

create table total\_cancel\_per\_year as

select

EXTRACT (YEAR FROM order\_purchase\_timestamp) AS YEAR,

count(1) as total\_canceled\_orders

from orders

where order\_status = 'canceled'

group by 1

**Subtask 3:**

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

**Cara 1:**

Membuat Table Baru **highest\_revenue\_per\_year**

CREATE TABLE highest\_revenue\_per\_year (

year float,

category\_name character varying,

total\_revenue float

)

Mengisi nilai Tabel **highest\_revenue\_per\_year** dan melakukan JOIN Tabel

INSERT INTO highest\_revenue\_per\_year (YEAR, category\_name, total\_revenue)

SELECT

EXTRACT(YEAR FROM Oi.shipping\_limit\_date) AS YEAR,

P.product\_category\_name AS category\_name,

SUM(Oi.price + Oi.freight\_value) AS total\_revenue

FROM order\_item AS Oi

JOIN products AS P

ON Oi.product\_id = P.product\_id

GROUP BY 1, 2

ORDER BY 1, 3 DESC

Melakukan Filtering untuk mendapatkan Kategori Produk dengan Pendapatan Tertinggi untuk masing-masing Tahun

SELECT year, category\_name, total\_revenue

FROM (

SELECT year, category\_name, total\_revenue,

ROW\_NUMBER() OVER (PARTITION BY year ORDER BY total\_revenue DESC) AS rn

FROM highest\_revenue\_per\_year

) sub

WHERE rn = 1

**Cara 2:**

CREATE TABLE top\_product\_category\_by\_revenue\_per\_year AS

SELECT

YEAR,

product\_category\_name,

revenue

FROM (

SELECT

EXTRACT (YEAR FROM order\_purchase\_timestamp) AS YEAR,

P.product\_category\_name,

SUM(Oi.price + Oi.freight\_value) AS revenue,

RANK() OVER(PARTITION BY

EXTRACT (YEAR FROM order\_purchase\_timestamp)

ORDER BY

SUM(Oi.price + Oi.freight\_value) DESC) AS rk

FROM order\_item AS Oi

JOIN orders AS O

ON O.order\_id = Oi.order\_id

JOIN products AS P

ON P.product\_id = Oi.product\_id

WHERE O.order\_status = 'delivered'

GROUP BY 1,2) sq

WHERE rk = 1

Langkah pertama pada **Cara 2** adalah membuat subquery yang berisi perhitungan total revenue yang dikelompokkan berdasarkan tahun dan juga kategori produk. Untuk mendapatkan nama kategori produk, dilakukan join ke tabel products. Tidak lupa juga digunakan window function **RANK** untuk mendapatkan kolom rk yang memberikan peringkat (RANK) pada setiap baris berdasarkan pendapatan (revenue) yang dihitung sebelumnya. Peringkat diberikan berdasarkan pendapat terbesar (descending) untuk setiap tahun. Setelah itu, dilakukan filter untuk mendapatkan kategori-kategori produk yang mempunyai peringkat 1 (artinya paling memberikan banyak revenue) untuk masing-masing tahun.

**Subtask 4:**

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

**Cara 1:**

Membuat Table **Baru most\_cancelled\_orders** yang berisi Nama Kategori Produk yang Memiliki Jumlah Order dibatalkan (cancel order)

CREATE TABLE most\_cancelled\_orders (

year float,

category\_name character varying,

total\_cancelled\_order bigint

);

Mengisi nilai Tabel **most\_cancelled\_orders** dan melakukan JOIN Tabel

INSERT INTO most\_cancelled\_orders (YEAR, category\_name, total\_cancelled\_order)

SELECT EXTRACT(YEAR FROM O.order\_purchase\_timestamp) AS YEAR,

P.product\_category\_name AS category\_name,

COUNT(\*) AS total\_cancelled\_order

FROM orders AS O

JOIN order\_item AS Oi

ON O.order\_id = Oi.order\_id

JOIN products AS P

ON Oi.product\_id = P.product\_id

WHERE O.order\_status = 'canceled'

GROUP BY 1,2

ORDER BY 1,3 DESC;

Melakukan Filtering untuk mendapatkan Kategori Produk dengan Jumlah Cancel Order terbanyak untuk masing-masing Tahun

SELECT year, category\_name, total\_cancelled\_order

FROM (

SELECT year, category\_name, total\_cancelled\_order,

ROW\_NUMBER() OVER (PARTITION BY year ORDER BY total\_cancelled\_order DESC) AS rn

FROM most\_cancelled\_orders

) sub

WHERE rn = 1;

**Cara 2:**

CREATE TABLE most\_canceled\_product\_category\_per\_year as

SELECT

YEAR,

product\_category\_name,

total\_canceled

FROM (

SELECT

EXTRACT (YEAR FROM order\_purchase\_timestamp) AS YEAR,

P.product\_category\_name,

COUNT(1) as total\_canceled,

RANK() OVER(PARTITION BY

EXTRACT (YEAR FROM order\_purchase\_timestamp)

ORDER BY COUNT(1) DESC) AS rk

FROM order\_item AS Oi

JOIN orders AS O

ON O.order\_id = Oi.order\_id

JOIN products AS p

ON P.product\_id = Oi.product\_id

WHERE O.order\_status = 'canceled'

GROUP BY 1,2) sq

WHERE rk = 1

Langkah-langkah query pada subtask **Cara 2** ini sama seperti subtask 3. Perbedaan hanya terletak pada window function **RANK**. Pada subtask 3 diberikan peringkat berdasarkan total revenue terbanyak, namun pada subtask kali ini pemberian peringkat (RANK) pada setiap baris berdasarkan jumlah pembatalan yang dihitung sebelumnya . Peringkat diberikan berdasarkan jumlah pembatalan terbanyak (descending) untuk setiap tahun. Tentu saja tidak lupa dilakukan filter untuk mendapatkan order yang dicancel (order\_status = ‘canceled’). Dengan cara seperti ini, berhasil didapatkan jumlah cancel untuk masing-masing kategori produk. Setelah itu filter untuk mendapatkan peringkat 1 (artinya jumlah cancel terbanyak untuk tahun tersebut) juga bisa dilakukan untuk mendapatkan hasil data sesuai keinginan.

**Subtask 5:**

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

SELECT

A.year,

B.revenue AS total\_revenue\_year,

D.total\_canceled\_orders as total\_canceled\_orders,

A.product\_category\_name AS top\_product\_category\_by\_revenue,

A.revenue AS total\_product\_category\_revenue,

C.product\_category\_name AS most\_canceled\_product,

C.total\_canceled AS total\_most\_canceled\_product

FROM top\_product\_category\_by\_revenue\_per\_year AS A

JOIN total\_revenue\_per\_year AS B

ON A.year = B.year

JOIN most\_canceled\_product\_category\_per\_year AS C

ON A.year = C.year

JOIN total\_cancel\_per\_year AS D

ON D.year = A.year

**TASK 4**

**Subtask 1:**

Menampilkan jumlah penggunaan masing-masing tipe pembayaran secara all time diurutkan dari yang terfavorit (kalau diurutkan dari yang terfavorit berarti sama dengan nilai terbesar)

**Cara 1:**

SELECT payment\_type,

COUNT(\*) AS total

FROM payments

GROUP BY 1

ORDER BY 2 DESC;

**Cara 2:**

SELECT

P.payment\_type,

COUNT(1) AS total

FROM payments AS P

JOIN orders AS O

ON O.order\_id = P.order\_id

GROUP BY 1

ORDER BY 2 DESC

**Subtask 2:**

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

**Cara 1:**

SELECT EXTRACT(YEAR FROM O.order\_purchase\_timestamp) AS YEAR,

P.payment\_type,

COUNT(\*) AS total

FROM payments AS P

JOIN orders AS O

ON P.order\_id = O.order\_id

GROUP BY 1,2

ORDER BY 1,3 DESC;

**Cara 2:**

with

tmp AS (

SELECT

EXTRACT(YEAR FROM O.order\_purchase\_timestamp) AS YEAR,

P.payment\_type,

COUNT(1) AS total

FROM payments AS P

JOIN orders AS O

ON O.order\_id = P.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 total ELSE 0 END) AS year\_2016,

SUM(CASE WHEN YEAR = '2017' THEN total ELSE 0 END) AS year\_2017,

sum(CASE WHEN YEAR = '2018' THEN total ELSE 0 END) AS year\_2018

FROM tmp

GROUP BY 1

) subq

ORDER BY 5 DESC

Langkah Pertama menggunakan subquery yang disebut "tmp" untuk menghitung jumlah pembayaran (total) berdasarkan tahun dan jenis pembayaran (payment\_type). Subquery ini menggunakan tabel "payments" yang di-join dengan tabel "orders" berdasarkan order\_id. Hasilnya kemudian dikelompokkan berdasarkan tahun dan jenis pembayaran, dan menghitung jumlah pembayaran untuk setiap grup.

Bagian kedua adalah subquery utama yang menggunakan subquery "tmp" untuk menghitung total pembayaran untuk setiap jenis pembayaran (payment\_type) pada tahun 2016, 2017, dan 2018. Hasilnya kemudian dikelompokkan berdasarkan jenis pembayaran dan menghitung jumlah pembayaran untuk setiap tahun secara terpisah. Subquery ini juga menambahkan kolom "pct\_change\_2017\_2018" yang menghitung persentase perubahan (growth) dari total pembayaran antara tahun 2017 dan 2018. Jika total pembayaran pada tahun 2017 adalah 0, maka nilai "pct\_change\_2017\_2018" akan diatur menjadi NULL untuk menghindari pembagian dengan nol.

Terakhir, hasil subquery utama diurutkan (ORDER BY) berdasarkan kolom "pct\_change\_2017\_2018" secara menurun (DESC), yang berarti jenis pembayaran dengan persentase pertumbuhan terbesar dari tahun 2017 ke tahun 2018 akan muncul di atas.