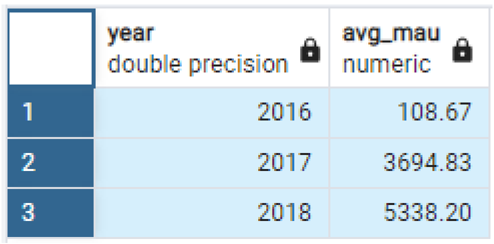
TASK 1

```
-- This script was generated by a beta version of the ERD tool in pgAdmin 4.
-- Please log an issue at https://redmine.postgresql.org/projects/pgadmin4/issues/new if
you find any bugs, including reproduction steps.
BEGIN;


CREATE TABLE IF NOT EXISTS public.customer_dataset
(
   customer_id character varying COLLATE pg_catalog."default" NOT NULL,
   customer_unique_id character varying COLLATE pg_catalog."default",
   customer_zip_code_prefix integer,
   customer_city character varying COLLATE pg_catalog."default",
   customer_state character varying COLLATE pg_catalog."default",
   CONSTRAINT customer_id_pkey PRIMARY KEY (customer_id)
);

CREATE TABLE IF NOT EXISTS public.geolocation_dataset
(
   geolocation_zip_code_prefix integer,
   geolocation_lat real,
   geolocation_lng real,
   geolocation_city character varying COLLATE pg_catalog."default",
   geolocation_state character varying COLLATE pg_catalog."default",
   CONSTRAINT geolocation_dataset PRIMARY KEY (geolocation_zip_code_prefix)
);

CREATE TABLE IF NOT EXISTS public.order_items_dataset
(
   order_id character varying COLLATE pg_catalog."default",
   order_item_id integer,
   product_id character varying COLLATE pg_catalog."default",
   seller_id character varying COLLATE pg_catalog."default",
   shipping_limit_date date,
   price real,
   freight_value real,
   CONSTRAINT order_items_dataset PRIMARY KEY (seller_id)
);

CREATE TABLE IF NOT EXISTS public.order_payments_dataset
(
   order_id character varying COLLATE pg_catalog."default",
   payment_sequential integer,
   payment_type character varying COLLATE pg_catalog."default",
   payment_installments integer,
   payment_value real,
   CONSTRAINT order_payments_dataset PRIMARY KEY (order_id)
);

CREATE TABLE IF NOT EXISTS public.order_reviews_dataset
(
   review_id character varying COLLATE pg_catalog."default",
   order_id character varying COLLATE pg_catalog."default",
   review_score integer,
   review_comment_title character varying COLLATE pg_catalog."default",
   review_comment_message character varying COLLATE pg_catalog."default",
   review_creation_date date,
   review_answer_timestamp date,
   CONSTRAINT order_reviews_dataset PRIMARY KEY (order_id)
);

CREATE TABLE IF NOT EXISTS public.orders_dataset
(
   order_id character varying COLLATE pg_catalog."default",
   customer_id character varying COLLATE pg_catalog."default",
   order_status character varying COLLATE pg_catalog."default",
   order_purchase_timestamp date,
   order_approved_at date,
   order_delivered_carrier_date date,
   order_delivered_customer_date date,
   order_estimated_delivery_date date,
   CONSTRAINT orders_dataset PRIMARY KEY (order_id)
);

CREATE TABLE IF NOT EXISTS public.product_dataset
(
   "Unnamed: 0" integer,
   product_id character varying COLLATE pg_catalog."default",
   product_category_name character varying COLLATE pg_catalog."default",
   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,
   CONSTRAINT product_dataset PRIMARY KEY (product_id)
);

CREATE TABLE IF NOT EXISTS public.sellers_dataset
(
   seller_id character varying COLLATE pg_catalog."default",
   seller_zip_code_prefix integer,
   seller_city character varying COLLATE pg_catalog."default",
   seller_state character varying COLLATE pg_catalog."default",
   CONSTRAINT sellers_dataset PRIMARY KEY (seller_id)
);

ALTER TABLE IF EXISTS public.customer_dataset
   ADD CONSTRAINT orders_dataset FOREIGN KEY (customer_id)
   REFERENCES public.orders_dataset (customer_id) MATCH SIMPLE
   ON UPDATE NO ACTION
   ON DELETE NO ACTION
   NOT VALID;


ALTER TABLE IF EXISTS public.geolocation_dataset
   ADD CONSTRAINT sellers_dataset FOREIGN KEY (geolocation_zip_code_prefix)
   REFERENCES public.sellers_dataset (seller_zip_code_prefix) MATCH SIMPLE
   ON UPDATE NO ACTION
   ON DELETE NO ACTION
   NOT VALID;


ALTER TABLE IF EXISTS public.geolocation_dataset
   ADD CONSTRAINT customer_dataset FOREIGN KEY (geolocation_zip_code_prefix)
   REFERENCES public.customer_dataset (customer_zip_code_prefix) MATCH SIMPLE
   ON UPDATE NO ACTION
   ON DELETE NO ACTION
   NOT VALID;


ALTER TABLE IF EXISTS public.order_items_dataset
   ADD CONSTRAINT sellers_dataset FOREIGN KEY (seller_id)
   REFERENCES public.sellers_dataset (seller_id) MATCH SIMPLE
   ON UPDATE NO ACTION
   ON DELETE NO ACTION
   NOT VALID;


ALTER TABLE IF EXISTS public.order_payments_dataset
   ADD CONSTRAINT orders_dataset FOREIGN KEY (order_id)
   REFERENCES public.orders_dataset (order_id) MATCH SIMPLE
   ON UPDATE NO ACTION
   ON DELETE NO ACTION
   NOT VALID;


ALTER TABLE IF EXISTS public.order_reviews_dataset
   ADD CONSTRAINT orders_dataset FOREIGN KEY (order_id)
   REFERENCES public.orders_dataset (order_id) MATCH SIMPLE
   ON UPDATE NO ACTION
   ON DELETE NO ACTION
   NOT VALID;


ALTER TABLE IF EXISTS public.orders_dataset
   ADD CONSTRAINT order_items_dataset FOREIGN KEY (order_id)
   REFERENCES public.order_items_dataset (order_id) MATCH SIMPLE
   ON UPDATE NO ACTION
   ON DELETE NO ACTION
   NOT VALID;


ALTER TABLE IF EXISTS public.product_dataset
   ADD CONSTRAINT order_items_dataset FOREIGN KEY (product_id)
   REFERENCES public.order_items_dataset (product_id) MATCH SIMPLE
   ON UPDATE NO ACTION
   ON DELETE NO ACTION
   NOT VALID;


ALTER TABLE IF EXISTS public.sellers_dataset
   ADD CONSTRAINT order_items_dataset FOREIGN KEY (seller_id)
   REFERENCES public.order_items_dataset (seller_id) MATCH SIMPLE
   ON UPDATE NO ACTION
   ON DELETE NO ACTION
   NOT VALID;

END;
```

TASK 2

a. **Rata-rata Monthly Active User (MAU) per tahun**

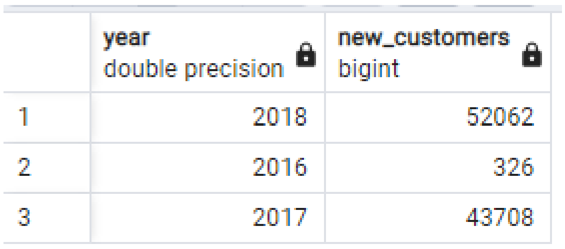
```
SELECT
        year,
        round(avg(mau), 2) as avg_mau
from (
        select
                date_part('year', o.order_purchase_timestamp) as year,
                date_part('month', o.order_purchase_timestamp) as month,
                count (distinct c.customer_unique_id) as mau
        from orders_dataset o
        join customer_dataset c on o.customer_id = c.customer_id
        group by 1,2
) subq
group by 1
```



	year double precision	avg_mau numeric
1	2016	108.67
2	2017	3694.83
3	2018	5338.20

b. **Total customer baru per tahun**

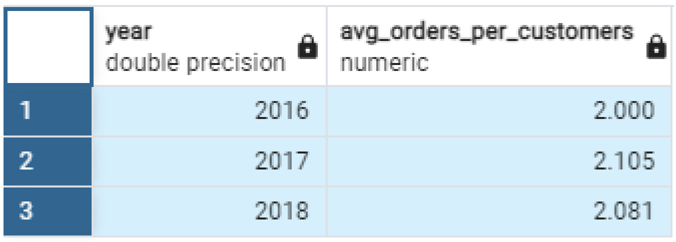
```
SELECT
        date_part('year', first_purchase_time) as year,
        count(1) as new_customers
from (
        select
                c.customer_unique_id,
                min(o.order_purchase_timestamp) as first_purchase_time
        from orders_dataset o
        join customer_dataset c on c.customer_id = o.customer_id
        group by 1
) subq
group by 1
```



	year double precision	new_customers bigint
1	2018	52062
2	2016	326
3	2017	43708

c. **Jumlah customer yang melakukan repeat order per tahun**

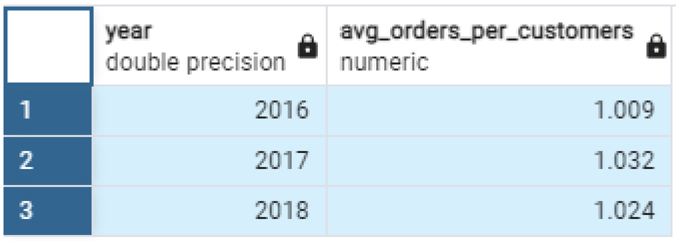
```
SELECT
        year,
        round(avg(frequency_purchase)::numeric, 3) as avg_orders_per_customers
FROM (
        SELECT
                date_part('year', o.order_purchase_timestamp) as year,
                c.customer_unique_id,
                count(1) as frequency_purchase
        FROM orders_dataset o
        JOIN customer_dataset c
        on c.customer_id = o.customer_id
        GROUP by 1,2
        HAVING count(1) > 1
) subq
group by 1
```



	year double precision	avg_orders_per_customers numeric
1	2016	2.000
2	2017	2.105
3	2018	2.081

d. **Rata-rata frekuensi order untuk setiap tahun**

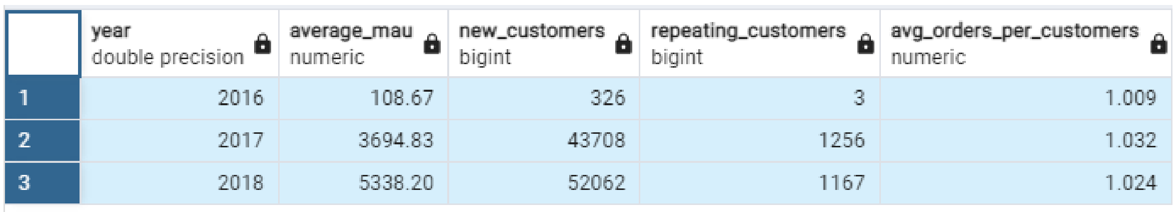
```
SELECT
        year,
        round(avg(frequency_purchase)::numeric, 3) as avg_orders_per_customers
FROM (
        SELECT
                date_part('year', o.order_purchase_timestamp) as year,
                c.customer_unique_id,
                count(1) as frequency_purchase
        FROM orders_dataset o
        JOIN customer_dataset c
        on c.customer_id = o.customer_id
        GROUP by 1,2
) subq
group by 1
```



	year double precision	avg_orders_per_customers numeric
1	2016	1.009
2	2017	1.032
3	2018	1.024

e. **<u>Menggabungkan ketiga metrik yang telah berhasil ditampilkan menjadi satu tampilan tabel</u>**

```
with
calc_mau as (
select
year,

round(avg(mau), 2) as average_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_dataset o
join customer_dataset c on o.customer_id = c.customer_id
group by 1,2
) subq
group by 1
),
calc_newcust 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_dataset o
join customer_dataset c on c.customer_id = o.customer_id
group by 1
) subq
group by 1
),
calc_repeat as (
select
year,
count(distinct customer_unique_id) as repeating_customers
from
(
select
date_part('year', o.order_purchase_timestamp) as year,
c.customer_unique_id,
count(1) as purchase_frequency
from orders_dataset o
join customer_dataset c on c.customer_id = o.customer_id
group by 1, 2
having count(1) > 1
) subq
group by 1
),

calc_avg_freq 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_dataset o
join customer_dataset c on c.customer_id = o.customer_id
group by 1, 2
) a
group by 1
)
select
mau.year,
mau.average_mau,
newc.new_customers,
rep.repeating_customers,
freq.avg_orders_per_customers
from calc_mau mau
join calc_newcust newc on mau.year = newc.year
join calc_repeat rep on rep.year = mau.year
join calc_avg_freq freq on freq.year = mau.year
```



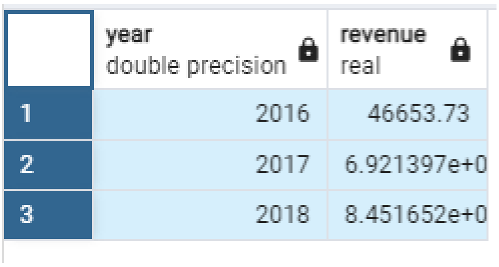
	year double precision	average_mau numeric	new_customers bigint	repeating_customers bigint	avg_orders_per_customers numeric
1	2016	108.67	326	3	1.009
2	2017	3694.83	43708	1256	1.032
3	2018	5338.20	52062	1167	1.024

Task III

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

1. create table total_revenue_per_year as

```
select
date_part('year', 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_items_dataset
group by 1
) subq
join orders_dataset o on subq.order_id = o.order_id
where o.order_status = 'delivered'
group by 1
```

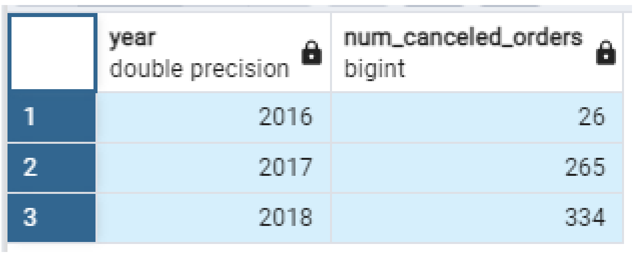


	year double precision	revenue real
1	2016	46653.73
2	2017	6.921397e+0
3	2018	8.451652e+0

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

2. create table total_cancel_per_year as

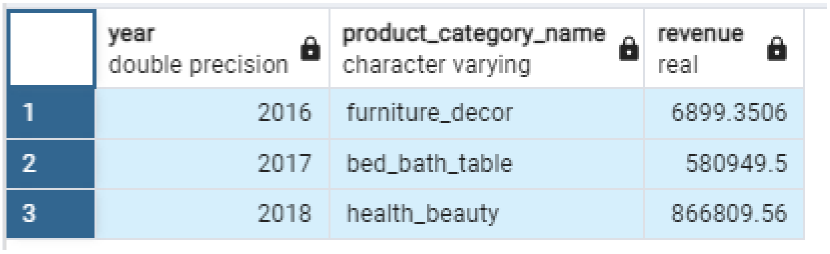
```
select
    date_part('year', order_purchase_timestamp) as year,
    count(1) as num_canceled_orders
from orders_dataset
where order_status = 'canceled'
group by 1
```



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

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

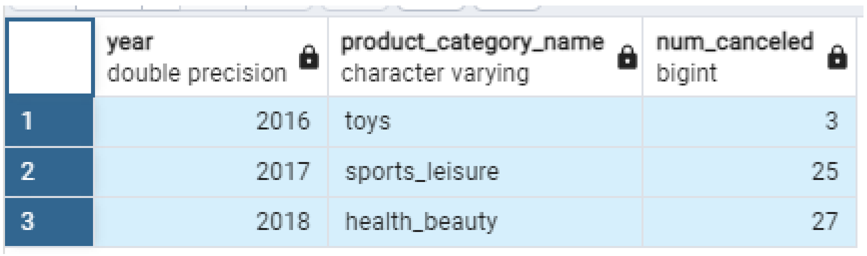
3.
```
create table top_product_category_by_revenue_per_year as
select
    year,
    product_category_name,
    revenue
from (
    select
            date_part('year', o.order_purchase_timestamp) as year,
            p.product_category_name,
            sum(oi.price + oi.freight_value) as revenue,
            rank() over(partition by

            date_part('year', o.order_purchase_timestamp)
            order by
            sum(oi.price + oi.freight_value) desc) as rk

            FROM order_items_dataset oi
            join orders_dataset o on o.order_id = oi.order_id
            join product_dataset p on p.product_id = oi.product_id
            where o.order_status = 'delivered'
            group by 1,2) sq
            where rk = 1
```



	year double precision	product_category_name character varying	revenue real
1	2016	furniture_decor	6899.3506
2	2017	bed_bath_table	580949.5
3	2018	health_beauty	866809.56

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

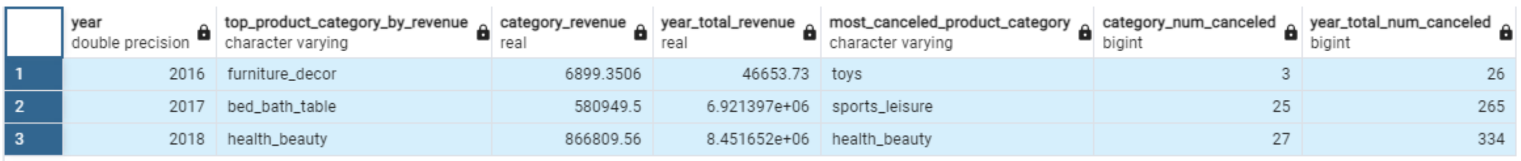
4.
```
create table most_canceled_product_category_per_year as
select
    year,
    product_category_name,
    num_canceled
from (
    select
            date_part('year', o.order_purchase_timestamp) as year,
            p.product_category_name,
            count(1) as num_canceled,
            rank() over(partition by

            date_part('year', o.order_purchase_timestamp)
            order by count(1) desc) as rk

            FROM order_items_dataset oi
            join orders_dataset o on o.order_id = oi.order_id
            join product_dataset p on p.product_id = oi.product_id
            where o.order_status = 'canceled'
            group by 1,2) sq
            where rk = 1
```



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

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

```
4. select
        a.year,
        a.product_category_name as top_product_category_by_revenue,
        a.revenue as category_revenue,
        b.revenue as year_total_revenue,
        c.product_category_name as most_canceled_product_category,
        c.num_canceled as category_num_canceled,
        d.num_canceled_orders as year_total_num_canceled
from top_product_category_by_revenue_per_year a
join total_revenue_per_year b on a.year = b.year
join most_canceled_product_category_per_year c on a.year = c.year
join total_cancel_per_year d on d.year = a.year
```

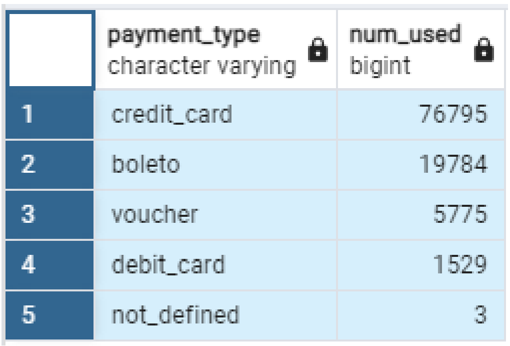


	year double precision	top_product_category_by_revenue character varying	category_revenue real	year_total_revenue real	most_canceled_product_category character varying	category_num_canceled bigint	year_total_num_canceled bigint
1	2016	furniture_decor	6899.3506	46653.73	toys	3	26
2	2017	bed_bath_table	580949.5	6.921397e+06	sports_leisure	25	265
3	2018	health_beauty	866809.56	8.451652e+06	health_beauty	27	334

Task IV

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

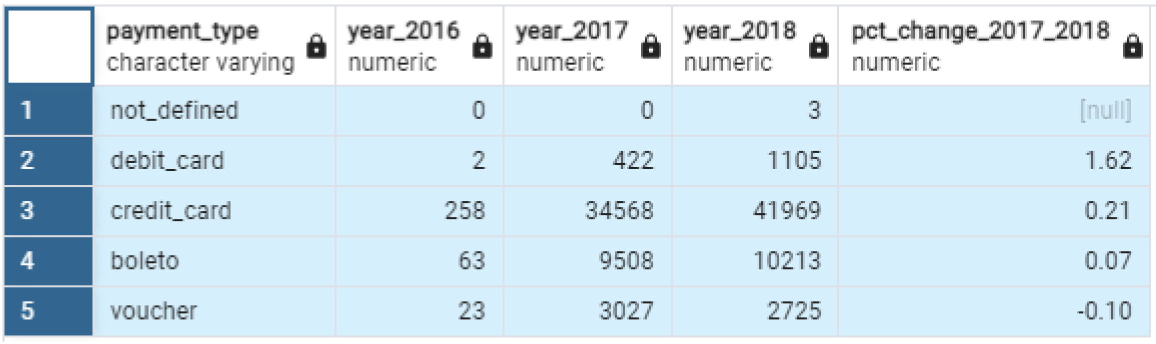
```
select
	op.payment_type,
	count(1) as num_used
FROM order_payments_dataset op
JOIN  orders_dataset o
on o.order_id = op.order_id
group by 1
order by 2 desc
```



	payment_type character varying	num_used 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 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
```



	payment_type character varying	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