```sql
--------------------------------- PART III DML

-- 1a. Insert 3 product_type
insert into product_type (type_name)
values ('Elektronik'), ('Pakaian'), ('Buku');

-- 1b. Insert 2 product dengan product_type_id = 1
insert into product (product_name, product_typeid, merk)
values ('Laptop', 1, 'Samsung'), ('Smartphone', 1, 'Iphone');

select * from product

-- 1c. Insert 3 product dengan product_type_id = 2
insert into product (product_name, product_typeid, merk)
values
('Kemeja', 2, 'Dust'),
('Jeans', 2, 'Levis'),
('Jaket', 2, 'Adidas');

-- 1d. Insert 3 product dengan product_type_id = 3
insert into product (product_name, product_typeid, merk)
values
('Novel', 3, 'Gramedia'),
('Buku Pelajaran', 3, 'Airlangga'),
('Majalah', 3, 'Tempo');

-- 1e. Insert product_description untuk setiap product
insert into product_description (product_descriptionid, description)
values
(1, 'barang baru dan bagus'),
(2, 'barang bekas dan masih bagus'),
(3, 'Kemeja katun nyaman'),
(4, 'Jeans denim gaya'),
(5, 'Jaket musim dingin hangat'),
(6, 'Novel terlaris'),
(7, 'Buku pelajaran edukatif'),
(8, 'Majalah populer');

-- 1f. Insert 3 payment_method
insert into payment_method (payment_methodid, method_name)
values
(1, 'Kartu Kredit'),
(2, 'Paypal'),
(3, 'Bayar di Tempat');

-- 1g. Insert 5 user
insert into users (name, address_id, dob, status_user, gender, created_at,
updated_at)
values
('Alisa', 1, '1990-01-01', 'aktif', 'F', NOW(), NOW()),
('Boby', 2, '1985-05-05', 'tidak aktif', 'M', NOW(), NOW()),
('Charly', 3, '1992-12-12', 'aktif', 'M', NOW(), NOW()),
('Davidson', 4, '1988-03-03', 'aktif', 'M', NOW(), NOW()),
('Eva', 5, '1995-07-07', 'aktif', 'F', NOW(), NOW());
```

```sql
-- 1h. Insert 3 transaksi untuk minimal 3 user
insert into transaction (userid, transaction_date)
values
(1, NOW()),
(2, NOW()),
(3, NOW()),
(1, NOW() - interval '2 day'),
(2, NOW() - interval '5 days'),
(3, NOW() - interval '7 days');

-- 1i. Insert 3 product dalam setiap transaksi
insert into transaction_detail (transactionid, productid, quantity, price,
payment_methodid)
values
(1, 7, 1, 10000000, 2),
(1, 8, 1, 5000000, 1),
(1, 15, 1, 300000, 1),
(2, 16, 1, 500000, 3),
(2, 17, 1, 400000, 1),
(2, 18, 1, 40000, 3),
(3, 19, 1, 30000, 3),
(3, 20, 1, 40000, 3),
(3, 18, 1, 60000, 1);


-- 2a.  Pilih nama user dengan gender 'M'
select name from users where gender = 'M';

-- 2b. Pilih product dengan id = 3
select * from product
where productid = 3;

-- 2c. Pilih data pelanggan yang created_at dalam 7 hari terakhir dan nama
mengandung 'a'
select * from users
where created_at >= NOW() - interval '7 days' AND name LIKE '%a%';

-- 2d. Hitung jumlah pelanggan dengan gender Perempuan (Female/F)
select count(*) from users where gender = 'F';

-- 2e. Tampilkan data pelanggan dalam urutan nama abjad
select * from users
order by name asc;

-- 2f. Tampilkan 5 data transaksi dengan product_id = 3
select * from transaction_detail
where productid = 3
limit 5;


-- 3a. Update nama product dimana product_id = 1 menjadi 'product_dummy'
update product
set product_name = 'product_dummy'
where productid = 1;

-- 3b. Update quantity menjadi 3 dalam transaction_detail dimana product_id = 1
update transaction_detail
set quantity = 3
where productid = 1;
```

```sql
-- 4a. Hapus data dalam tabel product dimana product_id = 1
delete from product
where productid=1;

-- 4b. Hapus data dalam tabel product dimana product_type_id = 1
delete from product
where product_typeid=1;


------------------------------- PART IV

-- 1. Gabungkan data transaksi dari user_id = 1 dan user_id = 2
select * from transaction
where userid IN (1, 2);

-- 2. Tampilkan jumlah harga transaksi user_id = 1
select sum(price * quantity)
as total_price from transaction_detail td
join transaction t ON td.transactionid = t.transactionid
where t.userid = 1;

select * from users
select * from product_type
select * from transaction
select * from transaction_detail

select transaction.transactionid , transaction_detail.price from
transaction_detail
inner join transaction on
transaction.transactionid=transaction_detail.transaction_detailid


-- 3. Tampilkan total transaksi untuk product_type = 2
select sum(price * quantity)
as total_transaction from transaction_detail td
join product p ON td.productid = p.productid
where p.product_typeid = 2;

-- 4. Tampilkan semua field dari tabel product dan field name dari tabel
product_type yang berhubungan
select p.*, pt.type_name
from product p
join product_type pt ON p.product_typeid = pt.product_typeid;

-- 5. Tampilkan semua field dari tabel transaction, field nama dari tabel product
dan field nama dari tabel user
select t.*, p.product_name, u.name
from transaction t
join transaction_detail td ON t.transactionid = td.transactionid
join product p ON td.productid = p.productid
join users u ON t.userid = u.userid;

--6. Tampilkan product yang tidak pernah ada di tabel transaction_detail
menggunakan subquery
select * from product
where productid NOT IN (select productid from transaction_detail);
```

```sql
-- 7. Buat fungsi dan trigger untuk mengupdate otomatis kolom updated_at
create or replace function update_updated_at_column()
returns trigger as $$
begin
   new.updated_at = NOW();
   return new;
end;
$$ language plpgsql;

-- Terapkan trigger ke semua tabel
create trigger update_user_updated_at before update ON users
for each row execute function update_updated_at_column();

create trigger update_product_updated_at before update ON product
for each row execute function update_updated_at_column();

create trigger update_transaction_updated_at before update ON product_type
for each row execute function update_updated_at_column();

create trigger update_transaction_updated_at before update ON product_description
for each row execute function update_updated_at_column();

create trigger update_transaction_updated_at before update ON transaction
for each row execute function update_updated_at_column();

create trigger update_transaction_detail_updated_at before update ON
transaction_detail
for each row execute function update_updated_at_column();

create trigger update_transaction_updated_at before update ON payment_method
for each row execute function update_updated_at_column();

create trigger update_transaction_updated_at before update ON address
for each row execute function update_updated_at_column();

-- 8. Buat view untuk menampilkan semua field dari product dan product_type
create view product_with_type AS
select p.*, pt.type_name
from product p
join product_type pt ON p.product_typeid = pt.product_typeid;
```