

LAPORAN PRAKTIKUM TEKNOLOGI BASIS DATA

MODUL 1

INSTALASI DAN PERANCANGAN SCHEMA



KELAS PRAKTIKUM TEKNOLOGI BASIS DATA – TIK1092_B

KELOMPOK 4 :

- | | |
|----------------------------|--------------|
| 1. HIKARU E JONES | 230211060107 |
| 2. VANCEL RENGKUNG | 230211060105 |
| 3. RISKY A. IMBAT | 230211060102 |
| 4. JONATHAN HENRY EMAN | 230211060041 |
| 5. CHRISTIAN D.A KAREPOWAN | 230211060095 |
| 6. GRACIANO P. TILAR | 230211060031 |

ASISTEN: MIRACLE SUMAJOW

**PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN ELEKTRO
FAKULTAS TEKNIK
UNIVERSITAS SAM RATULANGI
MANADO
2024**

LATIHAN

No. Latihan: 1**Soal Latihan:**

Buka sqlshell lalu isi sebagai berikut :

Server [localhost]: localhost

Database [postgres]: postgres

Port [5432]: 5432

Username [postgres]: postgres

Password for user postgres:

Statement SQL:

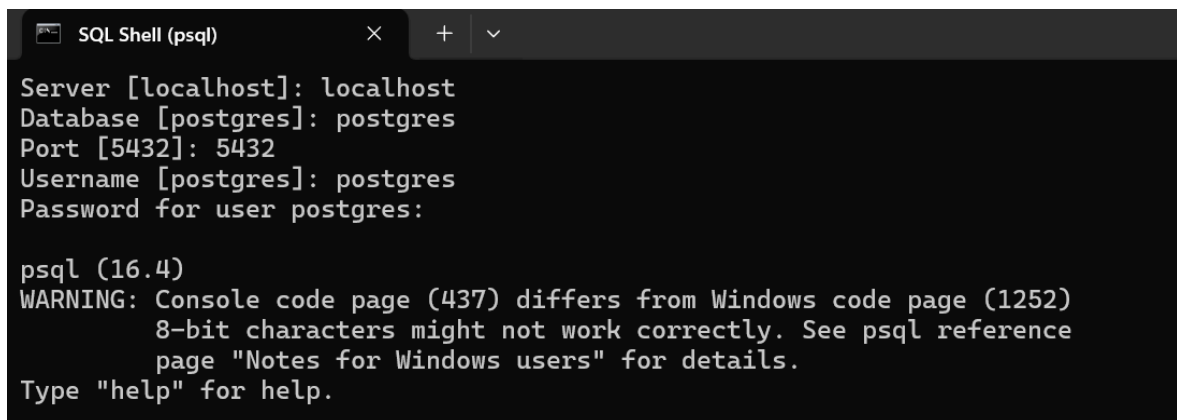
```
Server [localhost]:  
localhost  
Database [postgres]:  
postgres  
Port [5432]: 5432  
Username [postgres]:  
postgres  
Password for user  
postgres:
```

Tujuan/Penjelasan Query:

Tujuan utama query ini adalah untuk membuat koneksi ke database PostgreSQL dan mengelola data yang tersimpan di dalamnya.

Hasil Query/SQL:

SUCCESS



```
SQL Shell (psql)  x  +  v  
Server [localhost]: localhost  
Database [postgres]: postgres  
Port [5432]: 5432  
Username [postgres]: postgres  
Password for user postgres:  
  
psql (16.4)  
WARNING: Console code page (437) differs from Windows code page (1252)  
8-bit characters might not work correctly. See psql reference  
page "Notes for Windows users" for details.  
Type "help" for help.
```

Analisa Error:

Tidak terdapat error pada query ini.

Penerapan Solusi Akhir:


Tidak terdapat penerapan solusi akhir untuk query ini.

Penjelasan Statement:

Tidak terdapat penjelasan statement untuk query ini.

Hasil Penerapan Solusi Akhir:

Tidak terdapat hasil penerapan solusi akhir untuk query ini.

No. Latihan: 2	
Soal Latihan: Membuat database LATIHAN1	
Statement SQL: CREATE DATABASE latihan1;	Tujuan/Penjelasan Query:x Tujuan utama query ini adalah membuat database yang bernama latihan1.
Hasil Query/SQL: SUCCESS 	
Analisa Error: Tidak terdapat error pada query ini.	
Penerapan Solusi Akhir: Tidak terdapat penerapan solusi akhir untuk query ini.	Penjelasan Statement: Tidak terdapat penjelasan statement untuk query ini.
Hasil Penerapan Solusi Akhir: Tidak terdapat hasil penerapan solusi akhir untuk query ini.	

No. Latihan: 3**Soal Latihan:**

Masuk ke database latihan1 dan buatlah schema TOKO_ONLINE di dalam database latihan1. Kemudian periksa kembali apakah schema sudah ada.

Statement SQL:

```
\c latihan1;  
CREATE SCHEMA toko_online;  
\dn
```

Tujuan/Penjelasan Query:

Tujuan utama query ini adalah masuk ke database latihan1 serta membuat schema yang bernama toko_online dan memunculkannya.

Hasil Query/SQL:**SUCCESS**

```
postgres=# \c latihan1;  
You are now connected to database "latihan1" as user "postgres".  
latihan1=# CREATE SCHEMA toko_online;  
CREATE SCHEMA  
latihan1=# \dn  
              List of schemas  
  Name      | Owner  
-----+-----  
 public     | pg_database_owner  
 toko_online | postgres  
(2 rows)
```

Analisa Error:

Tidak terdapat error pada query ini.

Penerapan Solusi Akhir:

Tidak terdapat penerapan solusi akhir untuk query ini.

Penjelasan Statement:

Tidak terdapat penjelasan statement untuk query ini.

Hasil Penerapan Solusi Akhir:

Tidak terdapat hasil penerapan solusi akhir untuk query ini.

No. Latihan: 4

Soal Latihan:

Akses schema toko_online, kemudian buatlah sebuah tabel CUSTOMERS dengan atribut CUSTOMER_ID sebagai Primary Key.

Statement SQL:

```
1. SET search_path TO
   toko_online;
2. SHOW search_path;
3. CREATE TABLE customers (
   customer_id VARCHAR(4),
   customer_name
   VARCHAR(50),
   address VARCHAR(100),
   contact VARCHAR(100),
   CONSTRAINT cust_id_pk
   PRIMARY
   KEY(customer_id));
\dt
```

Tujuan/Penjelasan Query:

1. Tujuan query ini adalah untuk mengubah "search path" atau jalur pencarian schema, sehingga query-query berikutnya akan dieksekusi dalam schema toko_online.
2. Tujuan query ini adalah untuk menampilkan jalur pencarian schema yang sedang aktif.
3. Tujuan query ini adalah untuk membuat table customers yang didalamnya ada customer_id, customer_name, address dan contact. Setelah itu menetapkan customer_id sebagai primary key dan memunculkannya dengan \dt.

Hasil Query/SQL:

SUCCESS

```
latihan1=# SET search_path TO toko_online;
SET
latihan1=# SHOW search_path;
search_path
-----
toko_online
(1 row)

latihan1=# CREATE TABLE customers (
latihan1=# customer_id VARCHAR(4),
latihan1=# customer_name VARCHAR(50),
latihan1=# address VARCHAR(100),
latihan1=# contact VARCHAR(100),
latihan1=# CONSTRAINT cust_id_pk PRIMARY KEY(customer_id));
CREATE TABLE
latihan1=# \dt
          List of relations
Schema | Name      | Type  | Owner
-----+-----+-----+-----
toko_online | customers | table | postgres
(1 row)
```

Analisa Error: Tidak terdapat error pada query ini.	
Penerapan Solusi Akhir: Tidak terdapat penerapan solusi akhir untuk query ini.	Penjelasan Statement: Tidak terdapat penjelasan statement untuk query ini.
Hasil Penerapan Solusi Akhir: Tidak terdapat hasil penerapan solusi akhir untuk query ini.	

No. Latihan: 5

Soal Latihan: Buatlah tabel COMMODITY dengan atribut COMMODITY_ID sebagai Primary Key dan atribut UNIT_PRICE sebagai atribut non-null.

Statement SQL:

```
CREATE TABLE commodities (  
  commodity_id VARCHAR(4),  
  commodity_name VARCHAR(25),  
  unit_price NUMERIC(8,2) NOT  
  NULL,  
  CONSTRAINT comm_id_pk PRIMARY  
  KEY(commodity_id));  
\dt
```

Tujuan/Penjelasan Query:

Tujuan utama query ini adalah membuat tabel bernama commodities yang di dalamnya ada commodity_id, commodity_name dan unit_price.

Setelah itu menetapkan commodity_id sebagai primary key dan memunculkannya dengan \dt.

Hasil Query/SQL:**SUCCESS**

```
latihan1=# CREATE TABLE commodities (  
latihan1(# commodity_id VARCHAR(4),  
latihan1(# commodity_name VARCHAR(25),  
latihan1(# unit_price NUMERIC(8,2) NOT NULL,  
latihan1(# CONSTRAINT comm_id_pk PRIMARY KEY(commodity_id));  
CREATE TABLE  
latihan1=# \dt  
  
      List of relations  
Schema | Name | Type | Owner  
-----+-----+-----+-----  
toko_online | commodities | table | postgres  
toko_online | customers | table | postgres  
(2 rows)
```

Analisa Error:

Tidak terdapat error pada query ini.

Penerapan Solusi Akhir:

Tidak terdapat penerapan solusi akhir untuk query ini.

Penjelasan Statement:

Tidak terdapat penjelasan statement untuk query ini.

Hasil Penerapan Solusi Akhir:

Tidak terdapat hasil penerapan solusi akhir untuk query ini.

No. Latihan: 6

Soal Latihan: Buatlah tabel ORDERS dengan atribut ORDER_ID sebagai Primary Key, atribut COMMODITY_ID dan CUSTOMER_ID sebagai Foreign Key, atribut UNITS dan TOTAL_COST sebagai atribut non-null, dan menambahkan CONSTRAINT pada atribut numerik untuk menerima nilai lebih besar dari nol.

Statement SQL:

```
CREATE TABLE orders (  
  order_id VARCHAR(4),  
  customer_id VARCHAR(4),  
  commodity_id VARCHAR(4),  
  units NUMERIC(8,2) NOT NULL,  
  total_cost NUMERIC(8,2) NOT  
  NULL,  
  CONSTRAINT ord_r_id_pk PRIMARY  
  KEY(order_id),  
  CONSTRAINT ord_r_cust_fk  
  FOREIGN KEY (customer_id)  
  REFERENCES  
  customers(customer_id),  
  CONSTRAINT ord_r_comm_fk  
  FOREIGN KEY (commodity_id)  
  REFERENCES  
  commodities(commodity_id),  
  CONSTRAINT check_unit  
  CHECK(units > 0),  
  CONSTRAINT check_totl  
  CHECK(total_cost > 0));  
  
\dt
```

Tujuan/Penjelasan Query:

Tujuan utama query ini adalah membuat tabel bernama orders yang didalamnya ada order_id, customer_id, commodity_id, units dan total_cost.

Setelah itu menetapkan kolom order_id sebagai Primary Key.

Menambah Foreign Key ke kolom customer_id yang dihubungkan ke kolom customer_id di tabel customers.

Menambah Foreign Key ke kolom commodity_id yang dihubungkan ke kolom commodity_id di tabel commodities.

Lalu memastikan nilai di kolom units dan check_totl selalu lebih dari 0.

Menggunakan perintah \dt untuk menampilkan tabel.

Hasil Query/SQL:

SUCCESS

```
latihan1=# CREATE TABLE orders (  
latihan1(# order_id VARCHAR(4),  
latihan1(# customer_id VARCHAR(4),  
latihan1(# commodity_id VARCHAR(4),  
latihan1(# units NUMERIC(8,2) NOT NULL,  
latihan1(# total_cost NUMERIC(8,2) NOT NULL,  
latihan1(# CONSTRAINT ord_r_id_pk PRIMARY KEY(order_id),  
latihan1(# CONSTRAINT ord_r_cust_fk FOREIGN KEY (customer_id) REFERENCES customers(customer_id),  
latihan1(# CONSTRAINT ord_r_comm_fk FOREIGN KEY (commodity_id) REFERENCES commodities(commodity_id),  
latihan1(# CONSTRAINT check_unit CHECK(units > 0),  
latihan1(# CONSTRAINT check_totl CHECK(total_cost > 0));  
CREATE TABLE  
latihan1=# \dt  
      List of relations  
Schema | Name | Type | Owner  
-----+-----+-----+-----  
toko_online | commodities | table | postgres  
toko_online | customers | table | postgres  
toko_online | orders | table | postgres  
(3 rows)
```

Analisa Error:

Tidak terdapat error pada query ini.

Penerapan Solusi Akhir:

Tidak terdapat penerapan solusi akhir untuk query ini.

Penjelasan Statement:

Tidak terdapat penjelasan statement untuk query ini.

Hasil Penerapan Solusi Akhir:

Tidak terdapat hasil penerapan solusi akhir untuk query ini.

No. Latihan: 7

Soal Latihan: Tambahkan data dummy pada tabel-tabel yang telah dibuat dengan perintah INSERT sebagai berikut

Statement SQL:

- ```
1. INSERT INTO customers
(customer_id,
customer_name, address,
contact) VALUES ('C001',
'BDAVIS', 'Boston',
'650.551.4876');
INSERT INTO customers
(customer_id,
customer_name, address,
contact) VALUES ('C002',
'SSTEPHEN', 'ST.Louis',
'650.501.9321');
INSERT INTO customers
(customer_id,
customer_name, address,
contact) VALUES ('C003',
'DCARTER', 'California',
'650.507.6632');
SELECT * FROM customers;
```
- ```
2. INSERT INTO
commodities(commodity_id,
commodity_name,
unit_price)
VALUES ('M001', 'DVD
Player', 109);
INSERT INTO
commodities(commodity_id,
commodity_name,
unit_price)
```

Tujuan/Penjelasan Query:x

1. Tujuan utama query ini adalah memasukkan data dummy pada tabel “customers” dengan perintah INSERT INTO dan menampilkannya dengan perintah SELECT * FROM.
2. Tujuan utama query ini adalah memasukkan data dummy pada tabel “commodities” dengan perintah INSERT INTO dan menampilkannya dengan perintah SELECT * FROM.
3. Tujuan utama query ini adalah memasukkan data dummy pada tabel “orders” dengan perintah INSERT INTO dan menampilkannya dengan perintah SELECT * FROM.

<pre>VALUES ('M002', 'Cereal', 03); INSERT INTO commodities(commodity_id, commodity_name, unit_price) VALUES ('M003', 'Scrabble', 29); SELECT * FROM commodities; 3. INSERT INTO orders(order_id, customer_id, commodity_id, units,total_cost) VALUES ('R001', 'C003', 'M002', 50, 150); INSERT INTO orders(order_id, customer_id, commodity_id, units, total_cost) VALUES ('R002', 'C001', 'M003', 30, 87); INSERT INTO orders(order_id, customer_id, commodity_id, units, total_cost) VALUES ('R003', 'C003', 'M001', 6, 654); SELECT * FROM orders;</pre>	
--	--

Hasil Query/SQL:

1. SUCCESS

```
latihan1=# INSERT INTO customers (customer_id, customer_name, address, contact) VALUES ('C001', 'BDAVIS', 'Boston', '650.551.4876');
INSERT 0 1
latihan1=# INSERT INTO customers (customer_id, customer_name, address, contact) VALUES ('C002', 'SSTEPHEN', 'ST.Louis', '650.501.9321');
INSERT 0 1
latihan1=# INSERT INTO customers (customer_id, customer_name, address, contact) VALUES ('C003', 'DCARTER', 'California', '650.507.6632');
INSERT 0 1
latihan1=# SELECT * FROM customers;
 customer_id | customer_name | address | contact
-----+-----+-----+-----
C001         | BDAVIS        | Boston  | 650.551.4876
C002         | SSTEPHEN      | ST.Louis | 650.501.9321
C003         | DCARTER       | California | 650.507.6632
(3 rows)
```

2. SUCCESS

```
latihan1=# INSERT INTO commodities(commodity_id, commodity_name, unit_price) VALUES ('M001', 'DVD Player', 109);
INSERT 0 1
latihan1=# INSERT INTO commodities(commodity_id, commodity_name, unit_price) VALUES ('M002', 'Cereal', 03);
INSERT 0 1
latihan1=# INSERT INTO commodities(commodity_id, commodity_name, unit_price) VALUES ('M003', 'Scrabble', 29);
INSERT 0 1
latihan1=# SELECT * FROM commodities;
 commodity_id | commodity_name | unit_price
-----+-----+-----
M001          | DVD Player     | 109.00
M002          | Cereal         | 3.00
M003          | Scrabble       | 29.00
(3 rows)
```

3. SUCCESS

```
latihan1=# INSERT INTO orders(order_id, customer_id, commodity_id, units, total_cost) VALUES ('R001', 'C003', 'M002', 50, 150);
INSERT 0 1
latihan1=# INSERT INTO orders(order_id, customer_id, commodity_id, units, total_cost) VALUES ('R002', 'C001', 'M003', 30, 87);
INSERT 0 1
latihan1=# INSERT INTO orders(order_id, customer_id, commodity_id, units, total_cost) VALUES ('R003', 'C003', 'M001', 6, 654);
INSERT 0 1
latihan1=# SELECT * FROM orders;
 order_id | customer_id | commodity_id | units | total_cost
-----+-----+-----+-----+-----
R001     | C003        | M002         | 50.00 | 150.00
R002     | C001        | M003         | 30.00 | 87.00
R003     | C003        | M001         | 6.00  | 654.00
(3 rows)
```

Analisa Error:

Tidak terdapat error pada query ini.

Penerapan Solusi Akhir:

Tidak terdapat penerapan solusi akhir untuk query ini.

Penjelasan Statement:

Tidak terdapat penjelasan statement untuk query ini.

Hasil Penerapan Solusi Akhir:

Tidak terdapat hasil penerapan solusi akhir untuk query ini.

TUGAS

No. Tugas: 1.

Soal Tugas:

1. Buatlah sebuah database baru dengan nama TOKO_SEPEDA. Di dalam database tersebut, buatlah sebuah schema bernama SEPEDA_TOKO. Di dalam schema tersebut, buatlah dua tabel dengan spesifikasi berikut:

- a. Tabel CUSTOMERS:

- **customer_id** sebagai Primary Key dengan tipe data serial.
- **customer_name** dengan tipe data VARCHAR(100) yang tidak boleh kosong (NOT NULL).
- **address** dengan tipe data TEXT.
- **contact** dengan tipe data VARCHAR(15), harus unik (UNIQUE) dan menggunakan constraint format (regular expression) yang hanya menerima angka dan simbol (+) di awal nomor (contoh: +628123456789).

- b. Tabel PRODUCTS:

- **product_id** sebagai Primary Key dengan tipe data serial.
- **product_name** dengan tipe data VARCHAR(100), yang tidak boleh kosong (NOT NULL).
- **unit_price** dengan tipe data NUMERIC(10,2) (dua angka di belakang koma), dan harus lebih dari 0 (CHECK (unit_price > 0)).

Selain itu, buat relasi antara tabel CUSTOMERS dan tabel PRODUCTS dengan ketentuan:

- Setiap customer bisa memiliki banyak products (1 relationship). Tambahkan tabel ORDERS yang merepresentasikan pesanan sebagai tabel junction.

Detailnya:

- order_id sebagai Primary Key dengan tipe data serial.
- customer_id sebagai Foreign Key yang merujuk ke CUSTOMERS(customer_id).
- product_id sebagai Foreign Key yang merujuk ke PRODUCTS(product_id).
- order_date dengan tipe data TIMESTAMP yang menyimpan waktu pesanan dibuat.

Buatlah tiga constraint tambahan:

1. Pastikan bahwa order_date tidak boleh di masa depan (gunakan CHECK).
2. Tambahkan foreign key constraint dengan aturan ON DELETE CASCADE untuk customer_id di tabel ORDERS, sehingga ketika customer dihapus, pesanan terkait juga akan dihapus.
3. Pastikan combination customer_id dan product_id pada tabel ORDERS unik, agar tidak ada duplikasi pesanan yang sama dari customer yang sama.

Sertakan screenshot hasil pembuatan database, schema, tabel, dan constraint.

Lakukan query \d <table_name> untuk menampilkan struktur tabel.

Statement SQL:

```
postgres=# CREATE DATABASE
TOKO_SEPEDA;
\c toko_sepeda;
CREATE SCHEMA SEPEDA_TOKO;
SET search_path TO
sepeda_toko;
a. CREATE TABLE customers
(customer_id SERIAL,
customer_name
VARCHAR(100) NOT NULL,
address TEXT, contact
VARCHAR(15) UNIQUE,
CONSTRAINT cust_id_pk
PRIMARY KEY(customer_id),
CONSTRAINT contact_format
CHECK (contact ~
'^\+[0-9]{9,14}$'));
b. CREATE TABLE
products(product_id
SERIAL, product_name
VARCHAR(100) NOT NULL,
unit_price NUMERIC(10,2),
CONSTRAINT unit_pri
```

Tujuan/Penjelasan Query:

Membuat database 'TOKO_SEPEDA' dan masuk ke database yang baru dibuat, membuat schema 'SEPEDA_TOKO', lalu mengakses schema 'SEPEDA_TOKO'.

- a. Membuat table 'customers' dengan kolom ('customer_id' tipe data SERIAL, 'customer_name' tipe data VARCHAR(100) dan tidak boleh kosong, 'address' tipe data TEXT, 'contact' tipe data VARCHAR(15) dan harus unik, menetapkan 'customer_id' sebagai PRIMARY KEY, dan menambah CONSTRAINT pada kolom 'contact' untuk memastikan bahwa nilai yang diisi hanya +9 sampai 14 digit angka saja).
- b. Membuat tabel "products" dengan kolom('product_id' tipe data SERIAL, 'product_name' tipe data VARCHAR(100) dan tidak kosong, 'unit_price' tipe data NUMERIC dua angka di belakang koma, dan

<pre> CHECK(unit_price > 0), CONSTRAINT prod_id_pk PRIMARY KEY(product_id)); c. CREATE TABLE orders(order_id SERIAL, customer_id SERIAL, product_id SERIAL, order_date TIMESTAMP, CONSTRAINT ord_r_cust__fk FOREIGN KEY (customer_id) REFERENCES customers(customer_id), CONSTRAINT ord_r_pro_fk FOREIGN KEY (product_id) REFERENCES products(product_id)); 1. ALTER TABLE orders ADD CONSTRAINT check_order_date CHECK (order_date <= CURRENT_DATE); 2. ALTER TABLE orders ADD CONSTRAINT fk_customer FOREIGN KEY (customer_id) REFERENCES customers(customer_id) ON DELETE CASCADE; 3. ALTER TABLE orders ADD CONSTRAINT unique_order UNIQUE (customer_id, product_id); </pre>	<p>memastikan bahwa nilai 'unit_price' harus lebih dari 0, lalu menetapkan 'product_id' sebagai PRIMARY KEY).</p> <p>c. Membuat tabel 'orders' dengan kolom('order_id' tipe data SERIAL, 'customer_id' tipe data SERIAL, 'product_id' tipe data SERIAL, 'order_date' tipe data TIMESTAMP, menetapkan FOREIGN KEY pada kolom 'customer_id' yang mengacu pada tabel 'customers(customer_id)', dan FOREIGN KEY pada kolom 'product_id' yang mengacu pada tabel 'products(product_id)').</p> <ol style="list-style-type: none"> 1. Menambahkan aturan pada tabel 'orders' untuk mengecek bahwa 'order_date' tidak boleh lebih besar dari tanggal saat ini. 2. Menambahkan FOREIGN KEY pada kolom 'customer_id' tabel 'orders' yang mengacu pada tabel 'customers(customer_id)' yang mana jika data pada tabel induk dihapus, maka data pada tabel anakpun ikut terhapus. 3. Menambahkan aturan pada tabel 'orders' bahwa kombinasi 'customer_id' dan 'product_id' harus unik.
Hasil Query/SQL:	

SUCCESS

```
SQL Shell (psql)
postgres=# CREATE DATABASE TOKO_SEPEDA;
CREATE DATABASE
postgres=# \c TOKO_SEPEDA
connection to server at "localhost" (::1), port 5432 failed: FATAL:  database "TOKO_SEPEDA" does not exist
Previous connection kept
postgres=# \c TOKO_SEPEDA;
connection to server at "localhost" (::1), port 5432 failed: FATAL:  database "TOKO_SEPEDA" does not exist
Previous connection kept
postgres=# \c TOKO_SEPEDA;
connection to server at "localhost" (::1), port 5432 failed: FATAL:  database "TOKO_SEPEDA" does not exist
Previous connection kept
postgres=# CREATE DATABASE TOKO_SEPEDA;
ERROR:  database "toko_sepeda" already exists
postgres=# \c toko_sepeda;
You are now connected to database "toko_sepeda" as user "postgres".
toko_sepeda=# CREATE SCHEMA SEPEDA_TOKO;
CREATE SCHEMA
toko_sepeda=# SET search_path TO sepeda_toko;
SET
toko_sepeda=# SET search_path TO sepeda_toko;
SET
toko_sepeda=# CREATE TABLE orders (customer_id SERIAL, customer_name VARCHAR(100) NOT NULL, address TEXT, contact VARCHAR(15) UNIQUE, CONSTRAINT cust_id_pk PRIMARY KEY(customer_id), CONSTRAINT contact_format CHECK (contact ~ '^+[0-9]{9,14}$'));
CREATE TABLE
toko_sepeda=# drop table orders;
DROP TABLE
toko_sepeda=# CREATE TABLE customers (customer_id SERIAL, customer_name VARCHAR(100) NOT NULL, address TEXT, contact VARCHAR(15) UNIQUE, CONSTRAINT cust_id_pk PRIMARY KEY(customer_id), CONSTRAINT contact_format CHECK (contact ~ '^+[0-9]{9,14}$'));
CREATE TABLE
toko_sepeda=# CREATE TABLE products (product_id SERIAL, product_name VARCHAR(100) NOT NULL, unit_price NUMERIC(10,2), CONSTRAINT unit_pri CHECK(unit_price > 0), CONSTRAINT prod_id_pk PRIMARY KEY(product_id));
CREATE TABLE
toko_sepeda=# drop table products;
DROP TABLE
toko_sepeda=# CREATE TABLE products (product_id SERIAL, product_name VARCHAR(100) NOT NULL, unit_price NUMERIC(10,2), CONSTRAINT unit_pri CHECK(unit_price > 0), CONSTRAINT prod_id_pk PRIMARY KEY(product_id));
CREATE TABLE
toko_sepeda=# CREATE TABLE orders (order_id SERIAL, customer_id SERIAL, product_id SERIAL, order_date TIMESTAMP, CONSTRAINT ord_r_cust_fk FOREIGN KEY (customer_id) REFERENCES customers(customer_id), CONSTRAINT ord_r_pro_fk FOREIGN KEY (product_id) REFERENCES products(product_id));
CREATE TABLE
toko_sepeda=# CREATE TABLE orders (order_id SERIAL, customer_id SERIAL, product_id SERIAL, order_date TIMESTAMP, CONSTRAINT ord_r_cust_fk FOREIGN KEY (customer_id) REFERENCES customers(customer_id), CONSTRAINT ord_r_pro_fk FOREIGN KEY (product_id) REFERENCES products(product_id));
ERROR:  relation "orders" already exists
```

```
toko_sepeda=# \d customers;

      Column      |      Type       | Table "sepeda_toko.customers" | Collation | Nullable | Default |
-----+-----+-----+-----+-----+-----+
customer_id       | integer         |                                |           | not null |          | nextval('customers_customer_id_seq'::regclass)
customer_name     | character varying(100) |                                |           | not null |          |
address           | text            |                                |           |          |          |
contact           | character varying(15) |                                |           |          |          |
Indexes:
    "cust_id_pk" PRIMARY KEY, btree (customer_id)
    "customers_contact_key" UNIQUE CONSTRAINT, btree (contact)
Check constraints:
    "contact_format" CHECK (contact::text ~ '^+[0-9]{9,14}$'::text)
Referenced by:
    TABLE "orders" CONSTRAINT "fk_customer" FOREIGN KEY (customer_id) REFERENCES customers(customer_id) ON DELETE CASCADE
    TABLE "orders" CONSTRAINT "ord_r_cust_fk" FOREIGN KEY (customer_id) REFERENCES customers(customer_id)

toko_sepeda=# \d products

      Column      |      Type       | Table "sepeda_toko.products" | Collation | Nullable | Default |
-----+-----+-----+-----+-----+-----+
product_id        | integer         |                                |           | not null |          | nextval('products_product_id_seq'::regclass)
product_name      | character varying(100) |                                |           | not null |          |
unit_price        | numeric(10,2)    |                                |           |          |          |
Indexes:
    "prod_id_pk" PRIMARY KEY, btree (product_id)
Check constraints:
    "unit_pri" CHECK (unit_price > 0::numeric)
Referenced by:
    TABLE "orders" CONSTRAINT "ord_r_pro_fk" FOREIGN KEY (product_id) REFERENCES products(product_id)
```

```
toko_sepeda=# \d orders

      Column      |      Type       | Table "sepeda_toko.orders" | Collation | Nullable | Default |
-----+-----+-----+-----+-----+-----+
order_id          | integer         |                                |           | not null |          | nextval('orders_order_id_seq'::regclass)
customer_id       | integer         |                                |           | not null |          | nextval('orders_customer_id_seq'::regclass)
product_id        | integer         |                                |           | not null |          | nextval('orders_product_id_seq'::regclass)
order_date        | timestamp without time zone |                                |           |          |          |
Indexes:
    "unique_order" UNIQUE CONSTRAINT, btree (customer_id, product_id)
Check constraints:
    "check_order_date" CHECK (order_date <= CURRENT_DATE)
Foreign-key constraints:
    "fk_customer" FOREIGN KEY (customer_id) REFERENCES customers(customer_id) ON DELETE CASCADE
    "ord_r_cust_fk" FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
    "ord_r_pro_fk" FOREIGN KEY (product_id) REFERENCES products(product_id)
```

No. Tugas: 2.**Soal Tugas:**

Tambahkan tabel ORDERS di dalam schema SEPEDA_TOKO dengan struktur sebagai berikut:

- order_id sebagai Primary Key (tipe data serial).
- customer_id sebagai Foreign Key yang merujuk ke CUSTOMERS(customer_id).
- product_id sebagai Foreign Key yang merujuk ke PRODUCTS(product_id).
- order_data dengan tipe data TIMESTAMP yang menyimpan waktu pesanan dibuat.
- quantity dengan tipe data INTEGER, tambahkan constraint CHECK bahwa nilai harus lebih dari nol.
- total_cost dengan tipe data NUMERIC(10,2), tambahkan constraint CHECK bahwa nilainya harus lebih dari nol.

Selain itu, tambahkan ketentuan-ketentuan berikut:

1. Buatlah trigger yang secara otomatis menghitung total_cost pada tabel ORDERS berdasarkan nilai quantity dikalikan dengan unit_price dari tabel PRODUCTS setiap kali pesanan baru ditambahkan.
2. Tambahkan constraint tambahan pada quantity, yang memastikan bahwa jumlah barang yang dipesan tidak boleh melebihi stok yang tersedia. Buat kolom stock di tabel PRODUCTS yang menyimpan jumlah stok barang yang ada.
3. Buatlah foreign key constraint dengan aturan ON DELETE SET NULL untuk product_id di tabel ORDERS, sehingga ketika produk dihapus, nilai product_id pada pesanan yang terkait akan diubah menjadi NULL, namun pesanan tersebut tetap disimpan.
4. Tampilkan struktur tabel beserta constraint yang sudah dibuat menggunakan query \d ORDERS.

Sertakan screenshot hasil pembuatan tabel dan trigger, serta contoh input data ke tabel ORDERS dengan perhitungan total_cost yang dilakukan secara otomatis.

Statement SQL:

```
ALTER TABLE orders ADD COLUMN  
quantity INTEGER, ADD COLUMN  
total_cost NUMERIC(10,2), ADD  
CONSTRAINT check_quantity
```

Tujuan/Penjelasan Query:

Menambah kolom 'quantity' tipe data INTEGER, dan 'total_cost' tipe data NUMERIC dua angka di belakang koma dan memastikan nilai di dua kolom tersebut

```
CHECK(quantity > 0), ADD
CONSTRAINT check_totlcost
CHECK(total_cost > 0);
```

```
1. CREATE OR REPLACE
FUNCTION
calculate_total_cost()
RETURNS TRIGGER AS $$
BEGIN SELECT unit_price
INTO NEW.total_cost FROM
products WHERE product_id
= NEW.product_id;
NEW.total_cost :=
NEW.quantity *
NEW.total_cost; RETURN
NEW; END; $$ LANGUAGE
plpgsql;
CREATE TRIGGER
set_total_cost BEFORE
INSERT ON orders FOR EACH
ROW EXECUTE FUNCTION
calculate_total_cost();
ALTER TABLE products ADD
COLUMN stock INTEGER;
```

```
2. CREATE OR REPLACE
FUNCTION
check_stock_quantity(p_pr
oduct_id INT, p_quantity
INT) RETURNS BOOLEAN AS
$$
DECLARE available_stock
INT; BEGIN SELECT stock
INTO available_stock FROM
```

selalu lebih dari 0 dengan ADD CONSTRAINT di dalam tabel orders.

1. Membuat fungsi dengan nama `calculate_total_cost`, lalu menggunakan `RETURNS TRIGGER` yang berfungsi sebagai aksi yang berarti ketika ada operasi yang dilakukan dalam tabel seperti `INSERT`, maka fungsi akan dieksekusi. `$$` menandai awal dan akhir fungsi yang menunjukkan kode di dalamnya ditulis dalam bahasa `plpgsql` (PostgreSQL procedural language). Membuat Trigger baru bernama `set_total_cost` yang berfungsi untuk memasukkan nilai `total_cost` pada setiap baris di dalam tabel sebelum data ditambahkan. Statement ini berfungsi untuk menghitung `total_cost` berdasarkan `unit_price` dan `quantity` dari tabel `products`. Menambah kolom `stock` pada tabel `products` dengan tipe data `INTEGER`.

2. Membuat fungsi dengan nama `check_stock_quantity` yang berisi parameter `p_product_id` untuk memeriksa jumlah stok berdasarkan `product_id` dan `p_quantity` untuk memeriksa jumlah yang dipesan

<pre> products WHERE product_id= p_product_id FOR UPDATE;IF p_quantity > available_stock THEN RETURN FALSE;ELSE RETURN TRUE;END IF;END;\$\$ LANGUAGE plpgsql; ALTER TABLE orders ADD CONSTRAINT check_quantity_stock CHECK (check_stock_quantity(pro duct_id, quantity)); 3. ALTER TABLE orders ADD CONSTRAINT deletenull_pro FOREIGN KEY (product_id) REFERENCES products(product_id) ON DELETE SET NULL; 4. \d ORDERS </pre>	<p>apakah cukup. RETURNS BOOLEAN berarti nilai yang akan dikembalikan antara TRUE yang berarti stok mencukupi jumlah yang diminta dan FALSE yang berarti sebaliknya. Statement ini berfungsi untuk mendeklarasikan variabel available_stock dengan tipe INT, yang nilainya diambil dari kolom 'stock' pada tabel products lalu memeriksa apakah jumlah p_quantity lebih besar atau lebih kecil dari available_stock.</p> <p>Menambah CONSTRAINT bernama check_quantity_stock ke tabel orders yang akan memeriksa apakah stok produk mencukupi jumlah pesanan.</p> <p>3. Menambah CONSTRAINT bernama deletenull_pro ke tabel orders dan menambahkan Foreign Key ke kolom product_id yang dihubungkan ke kolom product_id pada tabel products dengan aksi ON DELETE SET NULL.</p> <p>4. Menampilkan struktur tabel orders dengan lengkap dengan perintah \d.</p>
Hasil Query/SQL:	

SUCCESS

```
toko_sepeda=# ALTER TABLE orders ADD COLUMN quantity INTEGER, ADD COLUMN total_cost NUMERIC(10,2), ADD CONSTRAINT check_quantity CHECK(quantity > 0), ADD CO
NSTRRAINT check_totlcost CHECK(total_cost > 0);
ALTER TABLE
toko_sepeda=# CREATE OR REPLACE FUNCTION calculate_total_cost() RETURNS TRIGGER AS $$ BEGIN SELECT unit_price INTO NEW.total_cost FROM products WHERE produ
ct_id = NEW.product_id; NEW.total_cost := NEW.quantity * NEW.total_cost; RETURN NEW; END; $$ LANGUAGE plpgsql;
CREATE FUNCTION
toko_sepeda=# CREATE TRIGGER set_total_cost BEFORE INSERT ON orders FOR EACH ROW EXECUTE FUNCTION calculate_total_cost();
CREATE TRIGGER
toko_sepeda=# ALTER TABLE products ADD COLUMN stock INTEGER; CREATE OR REPLACE FUNCTION check_stock_quantity(p_product_id INT, p_quantity INT) RETURNS BOOLE
AN AS $$ DECLARE available_stock INT; BEGIN SELECT stock INTO available_stock FROM products WHERE product_id = p_product_id; IF p_quantity > available_stock
THEN RETURN FALSE; ELSE RETURN TRUE; END IF; END; $$ LANGUAGE plpgsql;
ALTER TABLE
CREATE FUNCTION
toko_sepeda=# ALTER TABLE orders ADD CONSTRAINT check_quantity_stock CHECK (check_stock_quantity(product_id, quantity));

ALTER TABLE
toko_sepeda=# ALTER TABLE orders ADD CONSTRAINT deletenull_pro FOREIGN KEY (product_id) REFERENCES products(product_id) ON DELETE SET NULL);
ERROR:  syntax error at or near ")"
LINE 1: ..uct_id) REFERENCES products(product_id) ON DELETE SET NULL);
toko_sepeda=# ALTER TABLE orders ADD CONSTRAINT deletenull_pro FOREIGN KEY (product_id) REFERENCES products(product_id) ON DELETE SET NULL;
ALTER TABLE
```

```
toko_sepeda=# CREATE OR REPLACE FUNCTION check_stock_quantity(p_product_id INT, p_quantity INT) RETURNS BOOLEAN AS $$ DE
CLARE available_stock INT; BEGIN SELECT stock INTO available_stock FROM products WHERE product_id= p_product_id FOR UPDA
TE;IF p_quantity > available_stock THEN RETURN FALSE;ELSE RETURN TRUE;END IF;END;$$ LANGUAGE plpgsql;
CREATE FUNCTION
toko_sepeda=# ALTER TABLE orders ADD CONSTRAINT check_quantity_stock CHECK(check_stock_quantity(product_id, quantity))
```

```
toko_sepeda=# \d ORDERS

      Column |          Type          | Table "sepeda_toko.orders"
      -----|-----|-----|-----|-----|
      order_id | integer                | | not null | nextval('orders_order_id_seq'::regclass)
      customer_id | integer                | | not null | nextval('orders_customer_id_seq'::regclass)
      product_id | integer                | | not null | nextval('orders_product_id_seq'::regclass)
      order_date | timestamp without time zone |
      quantity | integer                |
      total_cost | numeric(10,2)          |

Indexes:
    "unique_order" UNIQUE CONSTRAINT, btree (customer_id, product_id)
Check constraints:
    "check_order_date" CHECK (order_date <= CURRENT_DATE)
    "check_quantity" CHECK (quantity > 0)
    "check_quantity_stock" CHECK (check_stock_quantity(product_id, quantity))
    "check_totlcost" CHECK (total_cost > 0::numeric)
Foreign-key constraints:
    "deletenull_pro" FOREIGN KEY (product_id) REFERENCES products(product_id) ON DELETE SET NULL
    "fk_customer" FOREIGN KEY (customer_id) REFERENCES customers(customer_id) ON DELETE CASCADE
    "ordr_cust_fk" FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
    "ordr_pro_fk" FOREIGN KEY (product_id) REFERENCES products(product_id)
Triggers:
    set_total_cost BEFORE INSERT ON orders FOR EACH ROW EXECUTE FUNCTION calculate_total_cost()
```

No. Tugas: 3**Soal Tugas:**

Masukkan data dummy ke dalam tabel CUSTOMERS, PRODUCTS, dan ORDERS dengan

ketentuan sebagai berikut:

- a. CUSTOMERS: Masukkan minimal 5 data. Pastikan salah satu customer memiliki nomor telepon yang tidak valid sehingga sistem harus menolak data tersebut karena constraint yang sudah dibuat.
- b. PRODUCTS: Masukkan minimal 5 data produk dengan jumlah stok masing-masing produk (stock) diatur. Satu produk harus memiliki harga (unit_price) kurang dari atau sama dengan nol, dan proses ini harus gagal karena constraint yang sudah ditentukan di tabel PRODUCTS.
- c. ORDERS: Masukkan minimal 3 data pesanan ke dalam tabel ORDERS. Salah satu pesanan harus memiliki jumlah barang (quantity) melebihi stok produk yang tersedia di tabel PRODUCTS, sehingga sistem menolak pesanan tersebut karena constraint stok.

Lalu tambahkan hal-hal berikut ini:

- a) Buatlah trigger tambahan yang secara otomatis mengurangi jumlah stok produk di tabel PRODUCTS setiap kali ada pesanan baru ditambahkan ke tabel ORDERS. Jika stok menjadi nol setelah pesanan, stok tidak boleh turun menjadi negatif.
- b) Tampilkan data pelanggan yang belum pernah melakukan pesanan menggunakan query yang menggabungkan tabel CUSTOMERS dan ORDERS (gunakan LEFT JOIN).
- c) Buatlah query untuk menghitung total pendapatan yang dihasilkan dari setiap produk (berdasarkan semua pesanan yang ada). Query ini harus menampilkan:
 - product_name
 - total_quantity_ordered (total jumlah produk yang dipesan)
 - total_revenue (total pendapatan dari produk tersebut, yaitu $\text{unit_price} * \text{quantity}$)
- d) Tampilkan data produk yang stoknya sudah habis (stok bernilai nol) menggunakan query yang memfilter tabel PRODUCTS.

Statement SQL:**Tujuan/Penjelasan Query:**

<p>a. INSERT INTO customers (customer_name, address, contact) VALUES ('John Doe', '123 Main St, Springfield', '+621234567890'), ('Jane Smith', '456 Elm St, Metropolis', '+628987654321'), ('Alice Johnson', '789 Maple St, Gotham', '+628123456789'), ('Bob Brown', '321 Oak St, Star City', '+629876543210'), ('Charlie White', '654 Pine St, Central City', '+628345678912'), ('Invalid Contact', '999 Unknown St, Nowhere', '12345');</p> <p>b. INSERT INTO products (product_name, unit_price, stock) VALUES ('Sepeda Gunung', 2500000.00, 12), ('Helm Sepeda', 300000.00, 25), ('Kunci Rantai', 80000.00, 40), ('Botol Minum', 50000.00, 60), ('Pelumas Rantai', -1000.00, 10);</p>	<p>a. Memasukkan data dummy ke dalam tabel 'customers', dimana salah satu data pada kolom 'contact' sengaja diberi nomor telepon yang tidak valid untuk memastikan CONSTRAINT yang telah dibuat itu bekerja.</p> <p>b. Memasukkan data dummy ke dalam tabel 'products', dimana salah satu data pada kolom 'unit_price' sengaja diberi harga ≤ 0 untuk memastikan CONSTRAINT yang telah dibuat itu bekerja.</p> <p>c. Memasukkan data dummy ke dalam tabel 'orders' dimana salah satu data pada kolom 'quantity' sengaja diberi nilai melebihi stok yang tersedia pada tabel 'products' untuk memastikan CONSTRAINT yang telah dibuat itu bekerja.</p> <p>a) Membuat fungsi bernama reduce_stock yang berfungsi untuk memeriksa stok dalam tabel products dan memperbarui jumlah stok di tabel products saat NEW.quantity yang menunjukkan jumlah yang akan dipesan dimasukkan ke tabel orders.</p> <p>Membuat Trigger baru bernama reduce_product_stock yang berfungsi untuk memasukkan nilai</p>
---	---

<pre> c. INSERT INTO orders (customer_id, product_id, order_date, quantity, total_cost) VALUES (8, 6, '2024-10-13', 10, 2500000.00), (9, 7, '2024-10-13', 20, 6000000.00), (10, 8, '2024-10-13', 50, 2000000.00); a) CREATE OR REPLACE FUNCTION reduce_stock() RETURNS TRIGGER AS \$\$ BEGIN IF (NEW.quantity > (SELECT stock FROM products WHERE product_id = NEW.product_id)) THEN RAISE EXCEPTION 'Quantity exceeds available stock for product_id = %', NEW.product_id; END IF; UPDATE products SET stock = stock - NEW.quantity WHERE product_id = NEW.product_id; RETURN NEW; END; \$\$ LANGUAGE plpgsql; CREATE TRIGGER reduce_product_stock AFTER INSERT ON orders FOR EACH ROW EXECUTE FUNCTION reduce_stock(); </pre>	<p>stock yang sudah dikurangi dengan NEW.quantity pada setiap baris di dalam tabel sebelum data ditambahkan.</p> <p>b) Query ini menggunakan LEFT JOIN untuk mencari data pelanggan yang belum pernah melakukan pemesanan.</p> <p>c) SUM(orders.quantity): Menghitung total kuantitas produk yang dipesan. SUM(orders.quantity * products.unit_price): Menghitung total pendapatan yang diperoleh dari penjualan produk dengan mengalikan 'orders.quantity' dengan 'products.unit_price'. GROUP BY products.product_name: Mengelompokkan hasil berdasarkan nama produk, sehingga setiap baris dalam hasil mewakili satu produk unik.</p> <p>d) Menampilkan data dari tabel 'produk' yang stoknya sudah habis.</p>
--	---

```

b) SELECT customers. * FROM
    customers LEFT JOIN
    orders ON
    customers.customer_id =
    orders.customer_id WHERE
    orders.order_id IS NULL;

c) SELECT
    products.product_name,
    SUM(orders.quantity) AS
    total_quantity_ordered,
    SUM(orders.quantity *
    products.unit_price) AS
    total_revenue FROM
    products JOIN orders ON
    products.product_id =
    orders.product_id GROUP
    BY products.product_name;

d) SELECT * FROM products
    WHERE stock = 0;

```

Hasil Query/SQL:

SUCCESS

- a. `toko_sepeda=# INSERT INTO customers (customer_name, address, contact) VALUES ('John Doe', '123 Main St, Springfield', '+621234567890'), ('Jane Smith', '456 Elm St, Metropolis', '+628987654321'), ('Alice Johnson', '789 Maple St, Gotham', '+628123456789'), ('Bob Brown', '321 Oak St, Star City', '+629876543210'), ('Charlie White', '654 Pine St, Central City', '+628345678912'), ('Invalid Contact', '999 Unknown St, Nowhere', '12345');
ERROR: new row for relation "customers" violates check constraint "contact_format"
DETAIL: Failing row contains (6, Invalid Contact, 999 Unknown St, Nowhere, 12345).`
- b. `toko_sepeda=# INSERT INTO products (product_name, unit_price, stock) VALUES ('Sepeda Gunung', 2500000.00, 12), ('Helm Sepeda', 300000.00, 25), ('Kunci Rantai', 80000.00, 40), ('Botol Minum', 50000.00, 60), ('Pelumas Rantai', -1000.00, 10);
ERROR: new row for relation "products" violates check constraint "unit_price"
DETAIL: Failing row contains (5, Pelumas Rantai, -1000.00, 10).`
- c. `toko_sepeda=# INSERT INTO orders (customer_id, product_id, order_date, quantity, total_cost) VALUES (8, 6, '2024-10-13', 10, 2500000.00), (9, 7, '2024-10-13', 20, 6000000.00), (10, 8, '2024-10-13', 50, 2000000.00);
ERROR: new row for relation "orders" violates check constraint "check_quantity_stock"
DETAIL: Failing row contains (6, 10, 8, 2024-10-13 00:00:00, 50, 4000000.00).`
- a) `toko_sepeda=# CREATE OR REPLACE FUNCTION reduce_stock() RETURNS TRIGGER AS $$ BEGIN IF (NEW.quantity > (SELECT stock FROM products WHERE product_id = NEW.product_id)) THEN RAISE EXCEPTION 'Quantity exceeds available stock for product_id = %', NEW.product_id; END IF; UPDATE products SET stock = stock - NEW.quantity WHERE product_id = NEW.product_id; RETURN NEW; END; $$ LANGUAGE plpgsql;
CREATE FUNCTION`
- `toko_sepeda=# CREATE TRIGGER reduce_product_stock BEFORE INSERT ON orders FOR EACH ROW EXECUTE FUNCTION reduce_stock();
CREATE TRIGGER`

b)

```
toko_sepeda=# SELECT customers.* FROM customers LEFT JOIN orders ON customers.customer_id = orders.customer_id WHERE orders.order_id IS NULL;
customer_id | customer_name | address | contact
-----
11 | Charlie White | 654 Pine St, Central City | +628345678912
10 | Bob Brown | 321 Oak St, Star City | +629876543210
7 | John Doe | 123 Main St, Springfield | +621234567890
(3 rows)
```

c)

```
toko_sepeda=# SELECT products.product_name, SUM(orders.quantity) AS total_quantity_ordered, SUM(orders.quantity * products.unit_price) AS total_revenue FROM
products JOIN orders ON products.product_id = orders.product_id GROUP BY products.product_name;
product_name | total_quantity_ordered | total_revenue
-----
Sepeda Gunung | 10 | 25000000.00
Helm Sepeda | 20 | 6000000.00
(2 rows)
```

d)

```
toko_sepeda=# SELECT * FROM products WHERE stock = 0;
product_id | product_name | unit_price | stock
-----
8 | Kunci Rantai | 80000.00 | 0
(1 row)
```

No. Tugas: 4**Soal Tugas**

Lakukan modifikasi pada tabel ORDERS dengan menambahkan kolom baru bernama status yang digunakan untuk menyimpan status pesanan, dengan ketentuan sebagai berikut :

- Kolom status menggunakan tipe data VARCHAR(20) dan hanya boleh berisi salah satu dari tiga nilai: "Pending", "Shipped", atau "Delivered".
- Gunakan constraint CHECK untuk memastikan bahwa nilai status yang dimasukkan hanya valid jika merupakan salah satu dari tiga status tersebut.

Lalu buatlah trigger yang secara otomatis memperbarui status menjadi "Delivered" jika pesanan sudah lebih dari 7 hari sejak order_date. Pesanan yang masih dalam status "Pending" atau "Shipped" akan diubah menjadi "Delivered" secara otomatis setelah 7 hari.

Setelah itu, tambahkan data baru ke dalam tabel ORDERS dengan status awal "Pending".

Dan perbarui status salah satu pesanan menjadi "Shipped" secara manual. Sertakan screenshot hasil modifikasi.

Statement SQL:

```
ALTER TABLE orders ADD COLUMN
status VARCHAR(20);

ALTER TABLE orders ADD
CONSTRAINT check_order_status
CHECK (status IN ('Pending',
'Shipped', 'Delivered'));

CREATE OR REPLACE FUNCTION
update_order_status() RETURNS
TRIGGER AS $$ BEGIN IF
NEW.order_date <= CURRENT_DATE
- INTERVAL '7 days' THEN
NEW.status := 'Delivered'; END
IF; RETURN NEW; END; $$
LANGUAGE plpgsql;
```

Tujuan/Penjelasan Query:

ALTER TABLE orders ADD COLUMN status VARCHAR(20); digunakan untuk menambahkan kolom "status" dengan tipe data VARCHAR(20).

ALTER TABLE orders ADD CONSTRAINT check_order_status CHECK (status IN ('Pending', 'Shipped', 'Delivered')); digunakan untuk menambahkan constraint "check_order_status". Constraint ini mengatur agar status nilai hanya bisa diisi dengan 'Pending', 'Shipped', dan 'Delivered'.

```

CREATE TRIGGER
after_order_insert_status
AFTER INSERT ON orders FOR
EACH ROW EXECUTE FUNCTION
update_order_status();

INSERT INTO orders
(customer_id, product_id,
order_date, quantity, status)
VALUES (10, 9, CURRENT_DATE,
2, 'Pending');

UPDATE orders SET status =
'Shipped' WHERE order_id = 30;

```

CREATE OR REPLACE FUNCTION
update_order_status() RETURNS
TRIGGER AS \$\$ BEGIN IF
NEW.order_date <= CURRENT_DATE -
INTERVAL '7 days' THEN NEW.status :=
'Delivered'; END IF; RETURN NEW;
END; \$\$ LANGUAGE plpgsql; digunakan
untuk membuat fungsi
“update_order_status”, fungsi ini membuat
nilai status setelah 7 hari menjadi
“Delivered”.

CREATE TRIGGER
after_order_insert_status AFTER INSERT
ON orders FOR EACH ROW EXECUTE
FUNCTION update_order_status();
digunakan untuk membuat trigger
“after_order_insert_status”, trigger ini
menggunakan fungsi
“update_order_status”. Trigger ini
dijalankan setelah baris baru ditambahkan,
memastikan bahwa setiap insert ada fungsi
yang dijalankan, dan ini bekerja pada
setiap baris yang dimasukkan nilai.

INSERT INTO orders (customer_id,
product_id, order_date, quantity, status)
VALUES (10, 9, CURRENT_DATE, 2,
'Pending'); memasukkan nilai kedalam
tabel orders,

UPDATE orders SET status = 'Shipped'
WHERE order_id = 30; digunakan untuk

	mengubah nilai yang ada pada tabel orders kolom status dengan order_id = 30, nilai status menjadi 'Shipped' yang sebelumnya 'Pending'.
--	--

Hasil Query/SQL:

SUCCESS

```
toko_sepeda=# ALTER TABLE orders ADD COLUMN status VARCHAR(20);
ALTER TABLE
toko_sepeda=# ALTER TABLE orders ADD CONSTRAINT check_order_status CHECK (status IN ('Pending', 'Shipped', 'Delivered'));
ALTER TABLE
```

```
toko_sepeda=# CREATE OR REPLACE FUNCTION update_order_status() RETURNS TRIGGER AS $$ BEGIN IF NEW.order_date <= CURRENT_DATE - INTERVAL '7 days' THEN NEW.st
atus := 'Delivered'; END IF; RETURN NEW; END; $$ LANGUAGE plpgsql;
CREATE FUNCTION
toko_sepeda=# CREATE TRIGGER after_order_insert AFTER INSERT ON orders FOR EACH ROW EXECUTE FUNCTION update_order_status();
ERROR: trigger "after_order_insert" for relation "orders" already exists
toko_sepeda=# CREATE TRIGGER after_order_insert_status AFTER INSERT ON orders FOR EACH ROW EXECUTE FUNCTION update_order_status();
CREATE TRIGGER
toko_sepeda=# INSERT INTO orders (customer_id, product_id, order_date, quantity, status) VALUES (7, 9, CURRENT_DATE, 2, 'Pending');
ERROR: duplicate key value violates unique constraint "unique_order"
DETAIL: Key (customer_id, product_id)=(7, 9) already exists.
toko_sepeda=# INSERT INTO orders (customer_id, product_id, order_date, quantity, status) VALUES (10, 9, CURRENT_DATE, 2, 'Pending');
INSERT 0 1
toko_sepeda=# SELECT * FROM orders;
 order_id | customer_id | product_id | order_date | quantity | total_cost | status
-----
7 | 8 | 6 | 2024-10-13 00:00:00 | 10 | 25000000.00 |
8 | 9 | 7 | 2024-10-13 00:00:00 | 20 | 60000000.00 |
21 | 7 | 9 | 2024-10-13 00:00:00 | 12 | 6000000.00 |
23 | 8 | 9 | 2024-10-13 00:00:00 | 12 | 6000000.00 |
24 | 7 | 6 | 2024-10-13 00:00:00 | 12 | 30000000.00 |
26 | 10 | 6 | 2024-10-13 00:00:00 | 11 | 27500000.00 |
27 | 7 | 7 | 2024-10-13 00:00:00 | 20 | 60000000.00 |
28 | 8 | 8 | 2024-10-13 00:00:00 | 40 | 32000000.00 |
30 | 10 | 9 | 2024-10-13 00:00:00 | 2 | 1000000.00 | Pending
(9 rows)
```

```
toko_sepeda=# UPDATE orders SET status = 'Shipped' WHERE order_id = 30;
UPDATE 1
toko_sepeda=# SELECT * FROM orders;
 order_id | customer_id | product_id | order_date | quantity | total_cost | status
-----
7 | 8 | 6 | 2024-10-13 00:00:00 | 10 | 25000000.00 |
8 | 9 | 7 | 2024-10-13 00:00:00 | 20 | 60000000.00 |
21 | 7 | 9 | 2024-10-13 00:00:00 | 12 | 6000000.00 |
23 | 8 | 9 | 2024-10-13 00:00:00 | 12 | 6000000.00 |
24 | 7 | 6 | 2024-10-13 00:00:00 | 12 | 30000000.00 |
26 | 10 | 6 | 2024-10-13 00:00:00 | 11 | 27500000.00 |
27 | 7 | 7 | 2024-10-13 00:00:00 | 20 | 60000000.00 |
28 | 8 | 8 | 2024-10-13 00:00:00 | 40 | 32000000.00 |
30 | 10 | 9 | 2024-10-13 00:00:00 | 2 | 1000000.00 | Shipped
(9 rows)
```

No. Tugas: 5**Soal Tugas:**

Buat tiga skema dalam PostgreSQL yang mewakili entitas bisnis yang berbeda. Setiap skema harus memiliki setidaknya tiga tabel yang dirancang berdasarkan Entity-Relationship Diagram (ERD). Setelah itu, buatlah hubungan antar skema dengan menghubungkan tabel-tabel yang relevan menggunakan Foreign Key.

Statement SQL:

```
SELECT * FROM employee
WHERE
employee.salary=1000000;
...
```

Tujuan/Penjelasan Query:

Lorem ipsum dolor sit amet

Hasil Query/SQL:

Capture Hasil Query (sukses/error/nilai yang tidak sesuai)

<Screenshoot hasil query>