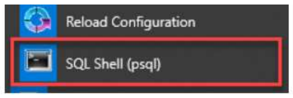
A. Instalasi

   I. Link referensi install [Install PostgreSQL (postgresqltutorial.com)]()

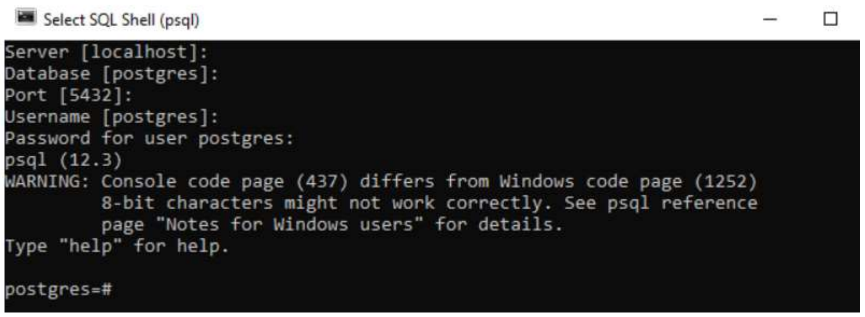
B. Terhubung ke Database

   I. Melalui terminal

      a. Pilih menu SQL Shell (psql)dari tombol start windows, seperti gambar berikut:

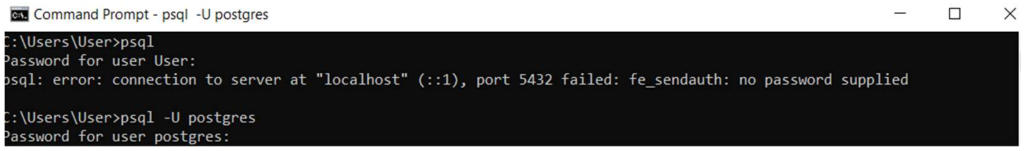


      b. Selanjutnya masukkan username dan password, dan jika berhasil maka akan tampil seperti gambar berikut:



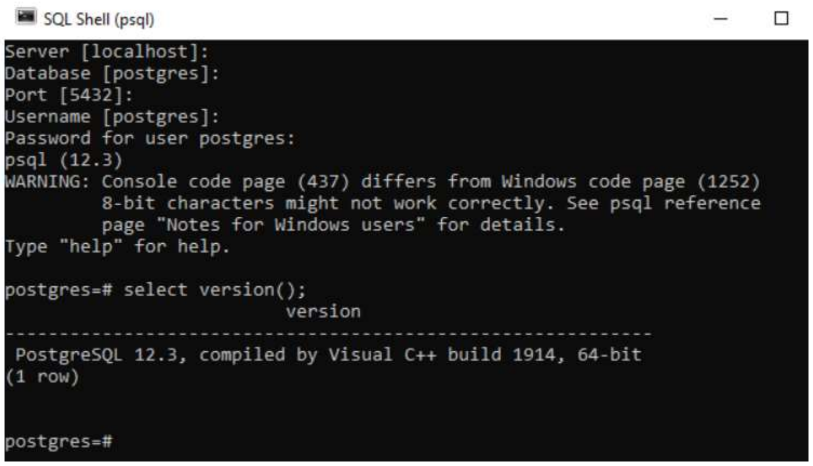
```
Server [localhost]:
Database [postgres]:
Port [5432]:
Username [postgres]:
Password for user postgres:
psql (12.3)
WARNING: Console code page (437) differs from Windows code page (1252)
         8-bit characters might not work correctly. See psql reference
         page "Notes for Windows users" for details.
Type "help" for help.

postgres=#
```

      c. Selain melalui SQL Shell bias dilakukan melalui command prompt, dengan mengetikkan psql -U username, lalu ketikkan password, seperti gambar berikut:



```
:\Users\User>psql
assword for user User:
sql: error: connection to server at "localhost" (::1), port 5432 failed: fe_sendauth: no password supplied

:\Users\User>psql -U postgres
assword for user postgres:
```

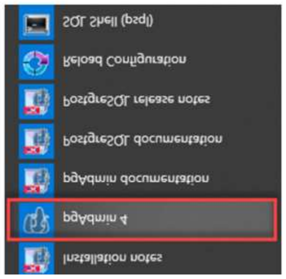
      d. Ketiikan perintah select version(), untuk melihat versi postgresql yang diinstall



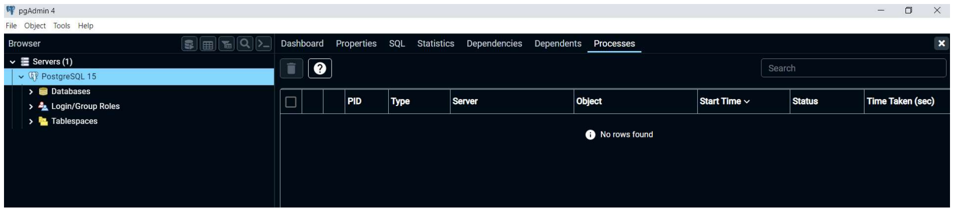
```
Server [localhost]:
Database [postgres]:
Port [5432]:
Username [postgres]:
Password for user postgres:
psql (12.3)
WARNING: Console code page (437) differs from Windows code page (1252)
         8-bit characters might not work correctly. See psql reference
         page "Notes for Windows users" for details.
Type "help" for help.

postgres=# select version();
                          version
------------------------------------------------------------
 PostgreSQL 12.3, compiled by Visual C++ build 1914, 64-bit
(1 row)


postgres=#
```

   II. Melalui pgAdmin

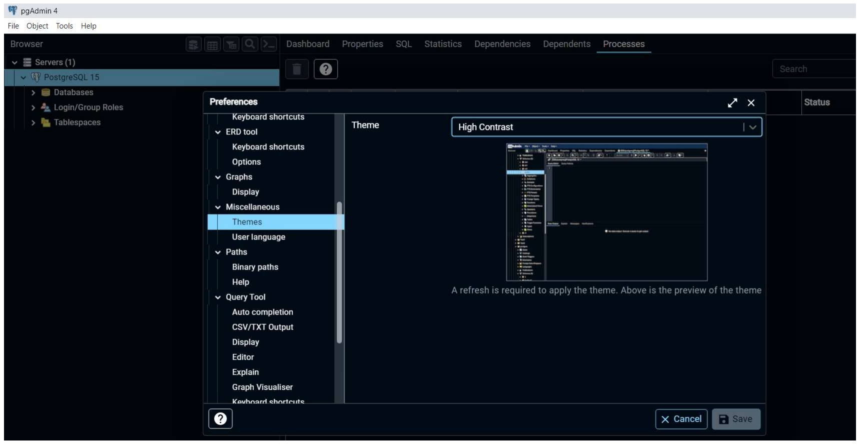
      a. Pilih menu pgAdmin dari tombol start windows, seperti gambar berikut:



b. Ketikkan password yang dibuat saat install, jika sudah terhubung maka akan tampil seperti berikut:


pgAdmin 4
File Object Tools Help
Browser
Servers (1)
PostgreSQL 15
Databases
Login/Group Roles
Tablespaces
Dashboard Properties SQL Statistics Dependencies Dependents Processes
Search
PID Type Server Object Start Time Status Time Taken (sec)
No rows found

c. Mengubah tampilan GUI pgAdmin, melalui menu file pada halaman pgAdmin lalu pilih preferences, seperti gambar berikut:

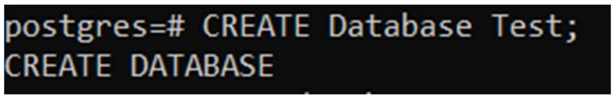


C. Database

I. Melalui terminal

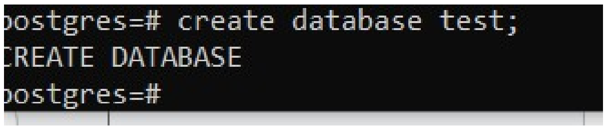
a. Lakukan koneksi ke database melalui terminal (lihat point B.I)

b. Jika berhasil melakukan koneksi, ketikkan perintah **create Database nama_database ;**, (bisa huruf kecil atau besar, tanda titik koma harus ada sebagai penanda akhir perintah) seperti gambar berikut:



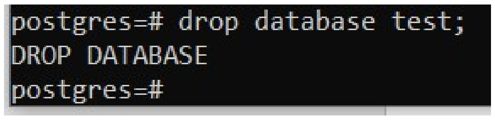
```
postgres=# CREATE Database Test;
CREATE DATABASE
```

c. Jika berhasil maka akan tampil seperti berikut:



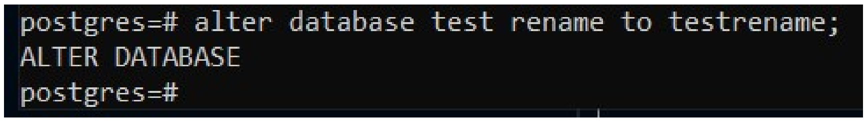
```
postgres=# create database test;
CREATE DATABASE
postgres=#
```

d. Untuk menghapus database yang dibuat ketikkan perintah **drop database nama_database ;**, (bisa huruf kecil atau besar, tanda titik koma harus ada sebagai penanda akhir perintah). Pastikan database yang akan dihapus sedang tidak digunakan, jika berhasil akan tampil seperti gambar berikut:



```
postgres=# drop database test;
DROP DATABASE
postgres=#
```

e. Untuk merename nama database ketikkan perintah **alter database nama_database_lama rename to nama_database_baru,** pastikan database yang akan direname sedang tidak digunakan, jika berhasil akan tampil seperti gambar berikut:

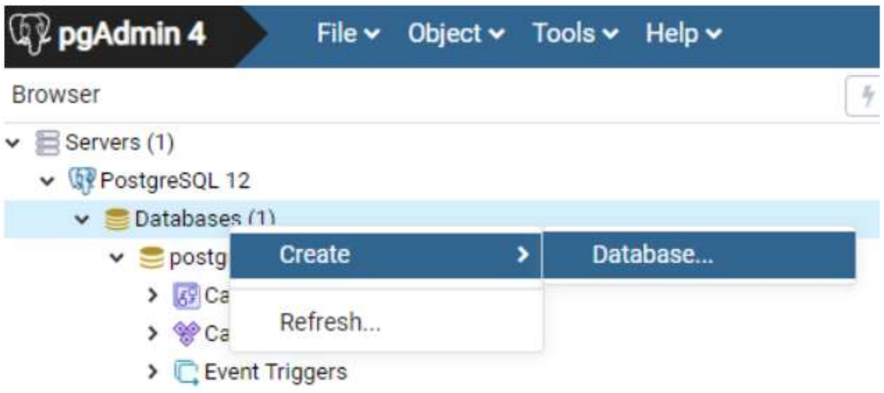


```
postgres=# alter database test rename to testrename;
ALTER DATABASE
postgres=#
```

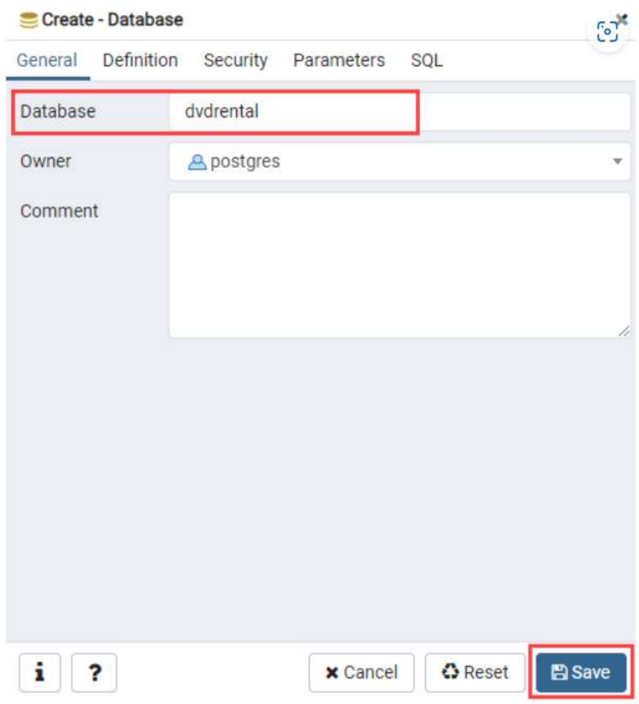
II. Melalui pgAdmin

a. Lakukan koneksi ke database melalui terminal (lihat point B.II)

b. Jika berhasil terhubung ke database, klik kanan pada direktori database lalu pilih menu create kemudian menu database, seperti gambar berikut:

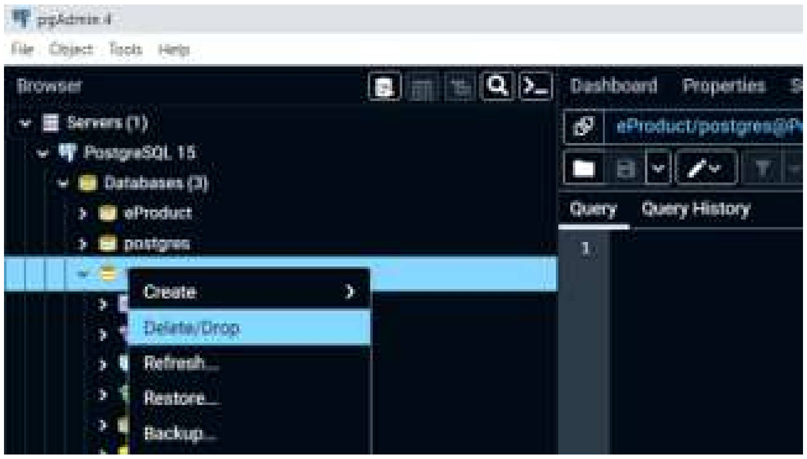


c. Ketikkan nama databse nya lalu tekan tombol save, seperti gambar barikut:



Database: dvdrental
Owner: postgres
Comment:

d. Untuk menghapus database, klik kanan pada nama databse yang akan dihapus lalu pilih menu delete/drop. Seperti gambar berikut:



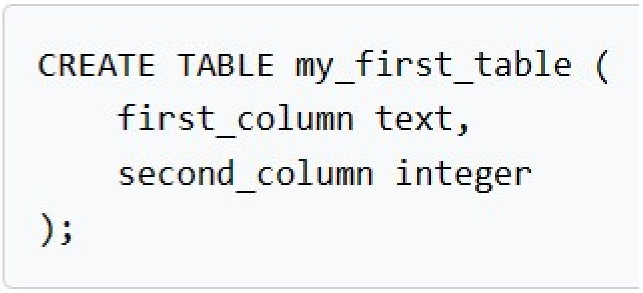
e. Jika berhasil maka database yang dihapus tidak akan tampil lagi di direktori databases

f. Mengakses database melalui shell atau terminal, ketikkan **\list** dan untuk melihat perintah laiinya ketikkan **\?**. Untuk masuk ke database tertentu ketikkan **\c nama_database**, selanjutnya bisa masukkan script sql misalnya select

D. Table

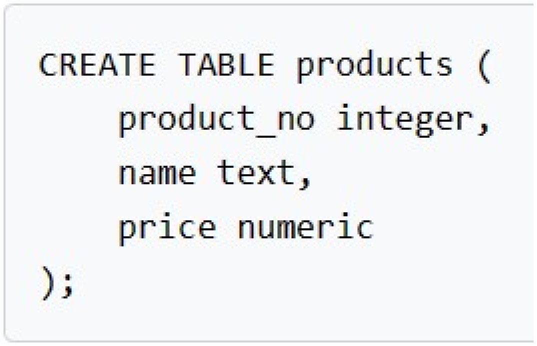
I. Membuat Table

a. Penamaan sebuah table sebaiknya konsisten, bisa menggunakan single atau jamak



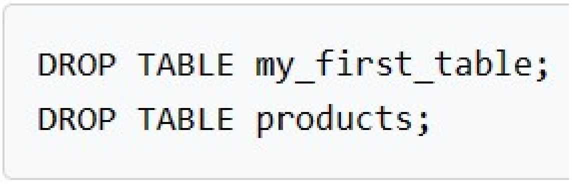
```
CREATE TABLE my_first_table (
    first_column text,
    second_column integer
);
```

Contoh: Membuat sebuah table bernama products dengam 3 (tiga) kolom yaitu product_no bertipe integer, name bertipe text, dan price bertipe numeric



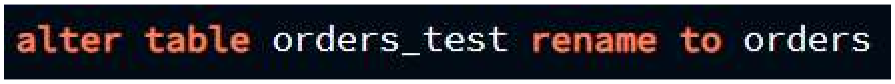
```
CREATE TABLE products (
    product_no integer,
    name text,
    price numeric
);
```

b. Batasan jumlah kolom yang dapat dimiliki senuah table antara 250 sampai 1600. Jika table sudah tidak diperlukan bisa dihapus dengan cara:



```
DROP TABLE my_first_table;
DROP TABLE products;
```

II. Rename Table

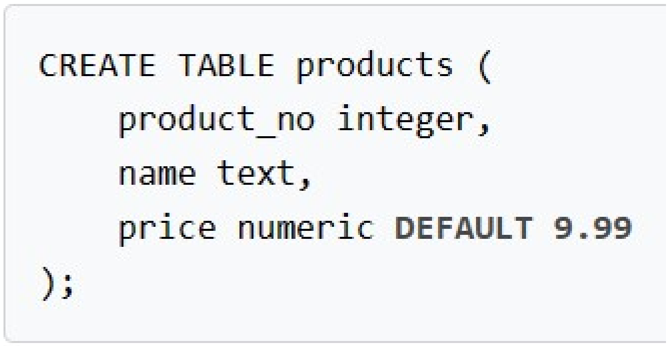


```
alter table orders_test rename to orders
```

III. Menambah/menghapus Kolom Table

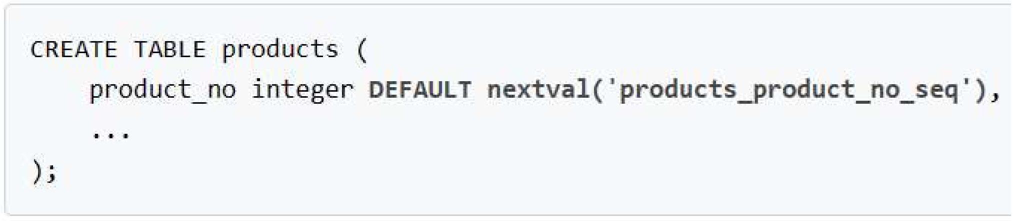
E. Column

a. Memberikan nilai default kolom



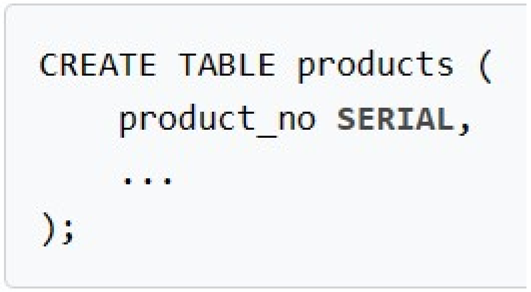
```
CREATE TABLE products (
    product_no integer,
    name text,
    price numeric DEFAULT 9.99
);
```

Contoh lain: nilai default kolom bisa dengan sebuah ekspresi selain memberikan nilai statik seperti contoh diatas. Biasanya ketika memberikan nilai default type waktu yaitu current_timestamp, atau seperti contoh mberikut membuat kolom product_no memiliki nilai default yang auto increment.



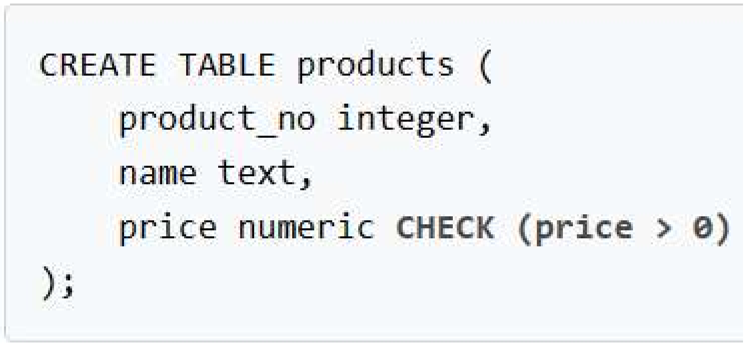
```
CREATE TABLE products (
    product_no integer DEFAULT nextval('products_product_no_seq'),
    ...
);
```

Contoh ini sama dengan penulisan singkat sebagai berikut:

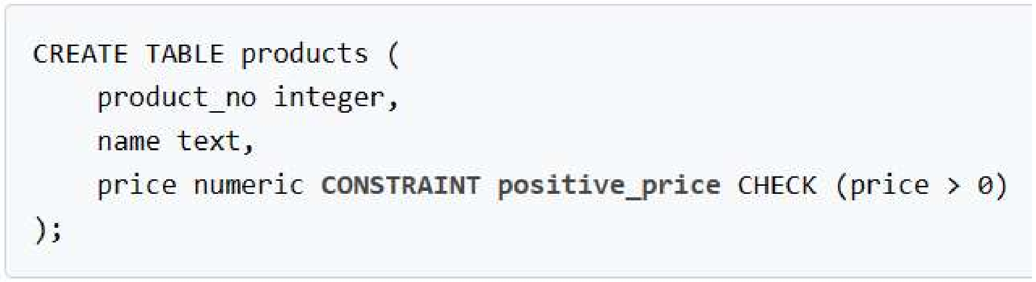


```
CREATE TABLE products (
    product_no SERIAL,
    ...
);
```

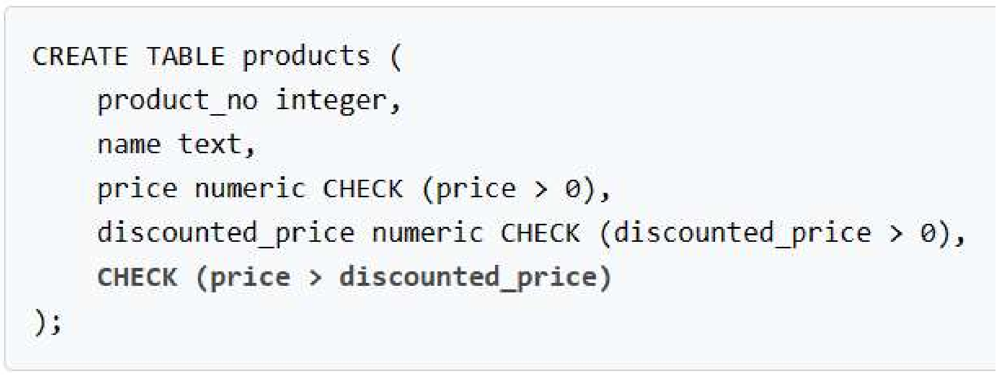
b. Pemberian tipe data adalah cara untuk membatasi jenis data yang disimpan di table. Setiap tipe data bisa diberikan constraint, pemberian constraint bisa dilakukan pada level kolom atau level table. CHECK



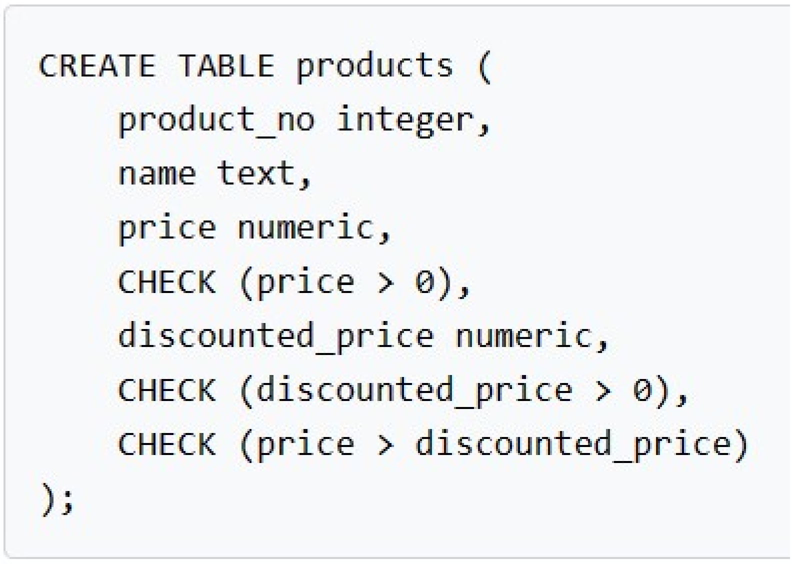
```
CREATE TABLE products (
    product_no integer,
    name text,
    price numeric CHECK (price > 0)
);
```



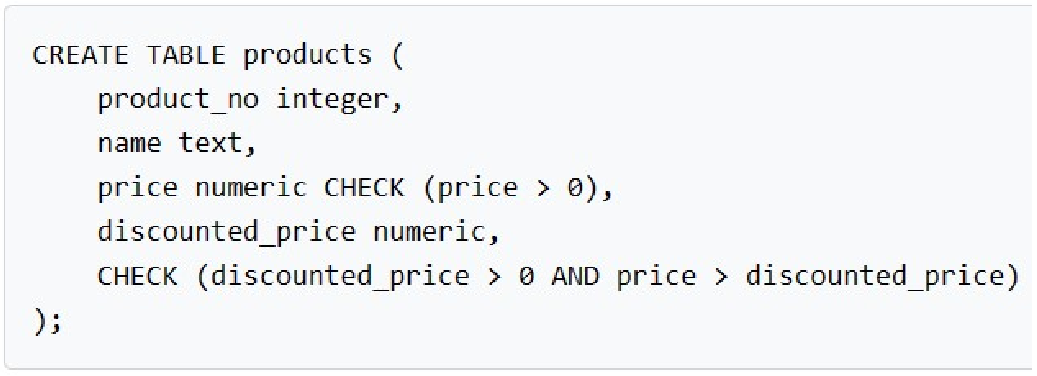
```
CREATE TABLE products (
    product_no integer,
    name text,
    price numeric CONSTRAINT positive_price CHECK (price > 0)
);
```



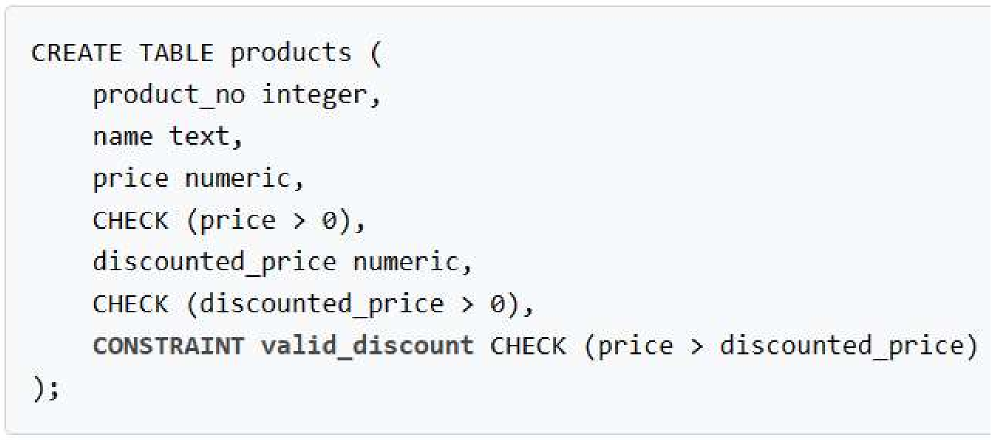
```
CREATE TABLE products (
    product_no integer,
    name text,
    price numeric CHECK (price > 0),
    discounted_price numeric CHECK (discounted_price > 0),
    CHECK (price > discounted_price)
);
```



```
CREATE TABLE products (
    product_no integer,
    name text,
    price numeric,
    CHECK (price > 0),
    discounted_price numeric,
    CHECK (discounted_price > 0),
    CHECK (price > discounted_price)
);
```

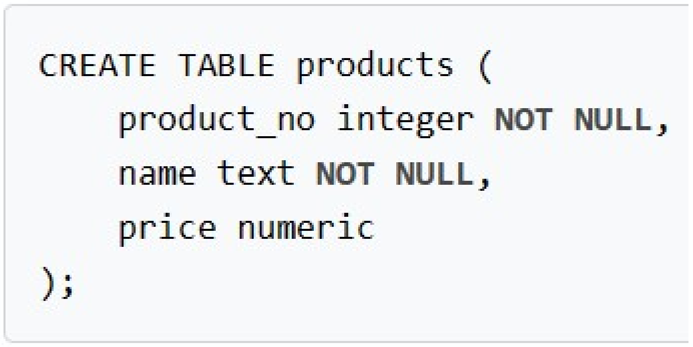


```
CREATE TABLE products (
    product_no integer,
    name text,
    price numeric CHECK (price > 0),
    discounted_price numeric,
    CHECK (discounted_price > 0 AND price > discounted_price)
);
```

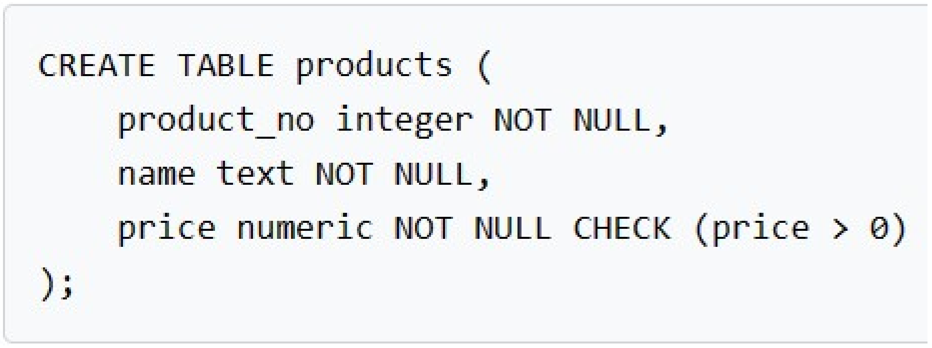


```
CREATE TABLE products (
    product_no integer,
    name text,
    price numeric,
    CHECK (price > 0),
    discounted_price numeric,
    CHECK (discounted_price > 0),
    CONSTRAINT valid_discount CHECK (price > discounted_price)
);
```

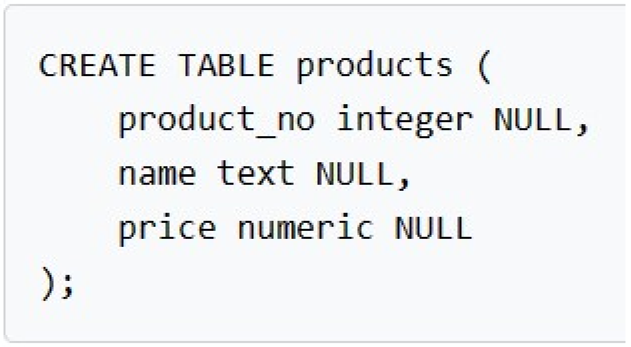
c. Not-Null Constraint



```
CREATE TABLE products (
    product_no integer NOT NULL,
    name text NOT NULL,
    price numeric
);
```

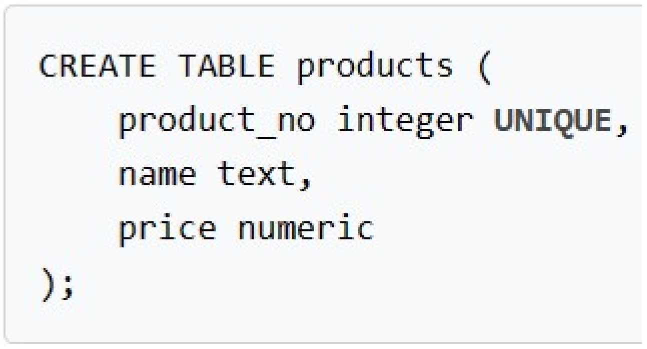


```
CREATE TABLE products (
    product_no integer NOT NULL,
    name text NOT NULL,
    price numeric NOT NULL CHECK (price > 0)
);
```

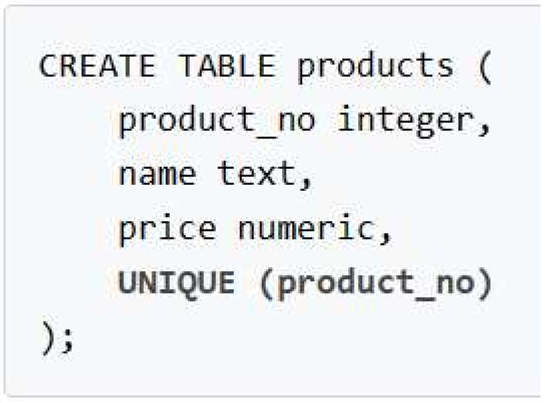


```
CREATE TABLE products (
    product_no integer NULL,
    name text NULL,
    price numeric NULL
);
```

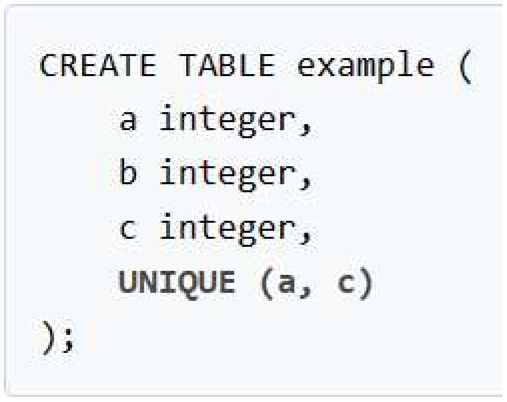
d. Unique Constraint, memastikan tidak ada nilai yang sama disemua row



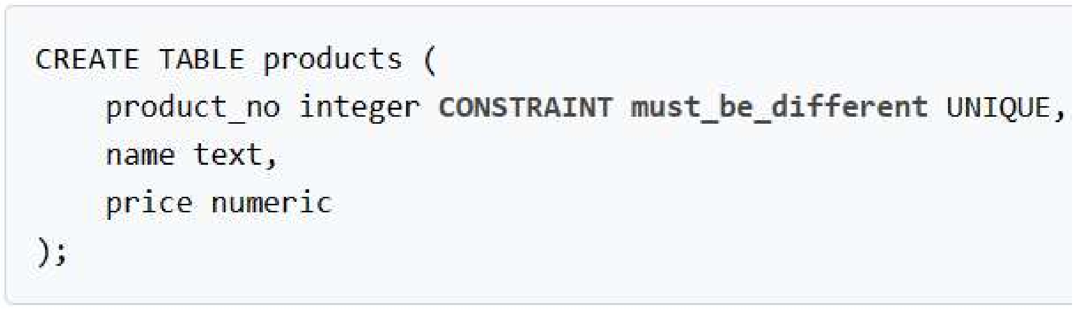
```
CREATE TABLE products (
    product_no integer UNIQUE,
    name text,
    price numeric
);
```



```
CREATE TABLE products (
    product_no integer,
    name text,
    price numeric,
    UNIQUE (product_no)
);
```



```
CREATE TABLE example (
    a integer,
    b integer,
    c integer,
    UNIQUE (a, c)
);
```



```
CREATE TABLE products (
    product_no integer CONSTRAINT must_be_different UNIQUE,
    name text,
    price numeric
);
```

Secara default null bisa duplikat walaupun diberi unique pada kolom tersebut, untuk mengubah spya null juga tidak bisa duplikat maka seperti berikut NULL NOT DISTINCT karena defaultnya NULL DISTINCT:

```
CREATE TABLE products (
    product_no integer UNIQUE NULLS NOT DISTINCT,
    name text,
    price numeric
);
```

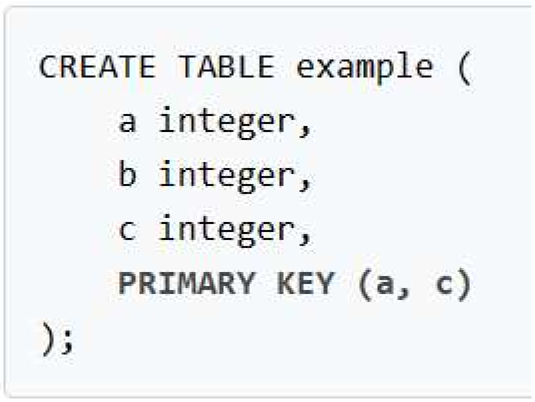
```
CREATE TABLE products (
    product_no integer,
    name text,
    price numeric,
    UNIQUE NULLS NOT DISTINCT (product_no)
);
```

e. Primary key Constraint, prinsipnya harus unique dan tidak bisa null

```
CREATE TABLE products (
    product_no integer UNIQUE NOT NULL,
    name text,
    price numeric
);
```

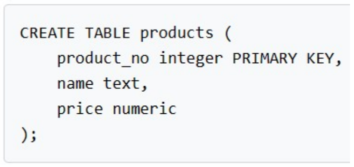
Sama dengan penulisan dibawah berikut ini:

```
CREATE TABLE products (
    product_no integer PRIMARY KEY,
    name text,
    price numeric
);
```

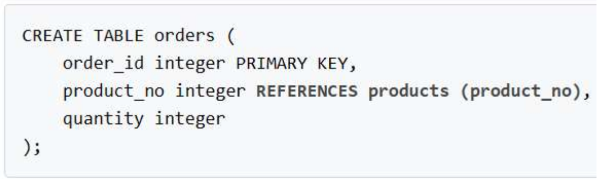


```
CREATE TABLE example (
    a integer,
    b integer,
    c integer,
    PRIMARY KEY (a, c)
);
```

f. Foreign Key Constraint

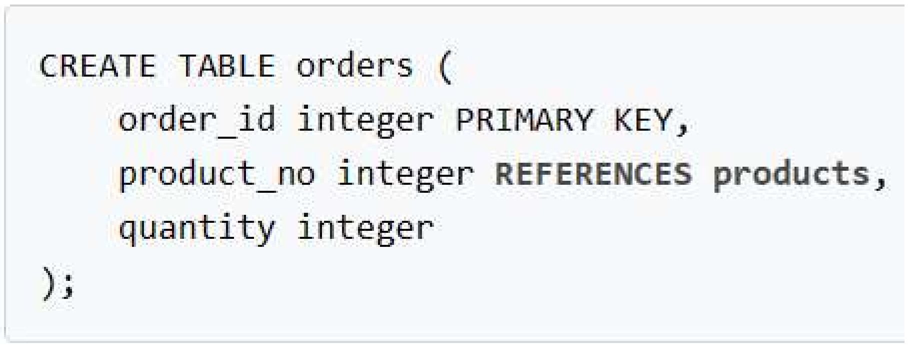


```
CREATE TABLE products (
    product_no integer PRIMARY KEY,
    name text,
    price numeric
);
```

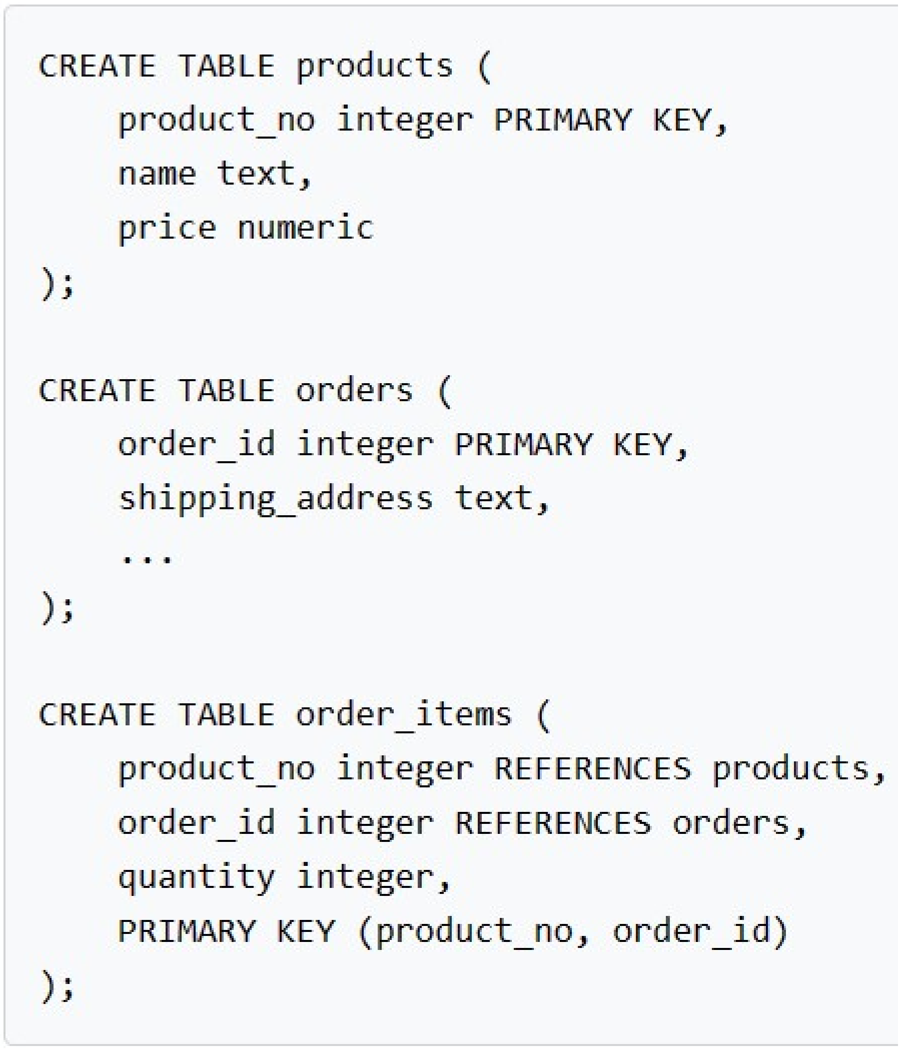


```
CREATE TABLE orders (
    order_id integer PRIMARY KEY,
    product_no integer REFERENCES products (product_no),
    quantity integer
);
```

Jika nama kolom diparentnya sama:



```
CREATE TABLE orders (
    order_id integer PRIMARY KEY,
    product_no integer REFERENCES products,
    quantity integer
);
```



```
CREATE TABLE products (
    product_no integer PRIMARY KEY,
    name text,
    price numeric
);

CREATE TABLE orders (
    order_id integer PRIMARY KEY,
    shipping_address text,
    ...
);

CREATE TABLE order_items (
    product_no integer REFERENCES products,
    order_id integer REFERENCES orders,
    quantity integer,
    PRIMARY KEY (product_no, order_id)
);
```

Dengan adanya Foreign Key tidak akan mengijinkan pembuatan order_items jika productnya tidak ada atau productnya tidak relate ditable product. Bagaimana jika product yang sudah disorder disorder_item dihapus dari table product. SQL bisa handle dengan restrict, no action atau cascade.

```
CREATE TABLE products (
    product_no integer PRIMARY KEY,
    name text,
    price numeric
);

CREATE TABLE orders (
    order_id integer PRIMARY KEY,
    shipping_address text,
    ...
);

CREATE TABLE order_items (
    product_no integer REFERENCES products ON DELETE RESTRICT,
    order_id integer REFERENCES orders ON DELETE CASCADE,
    quantity integer,
    PRIMARY KEY (product_no, order_id)
);
```

g. Modifikasi table tanpa harus delete table menggunakan perintah ALTER TABLE.
Berikut untuk menambah kolom

```
ALTER TABLE products ADD COLUMN description text;
```

```
ALTER TABLE products ADD COLUMN description text CHECK (description <> '');
```

h. Menghapus kolom

```
ALTER TABLE products DROP COLUMN description;
```

```
ALTER TABLE products DROP COLUMN description CASCADE;
```

i. Menambah constraint

```
ALTER TABLE products ADD CHECK (name <> '');
ALTER TABLE products ADD CONSTRAINT some_name UNIQUE (product_no);
ALTER TABLE products ADD FOREIGN KEY (product_group_id) REFERENCES product_groups;
ALTER TABLE products ALTER COLUMN product_no SET NOT NULL;
```

j. Menghapus Constraint

```
ALTER TABLE products DROP CONSTRAINT some_name;
```

```
ALTER TABLE products ALTER COLUMN product_no DROP NOT NULL;
```

k. Mengubah Default value kolom

```
ALTER TABLE products ALTER COLUMN price SET DEFAULT 7.77;
```

```
ALTER TABLE products ALTER COLUMN price DROP DEFAULT;
```

l. Mengubah type data kolom

```
ALTER TABLE products ALTER COLUMN price TYPE numeric(10,2);
```

m. Rename nama kolom

```
ALTER TABLE products RENAME COLUMN product_no TO product_number;
```

F. Generate ERD

G. Manipulasi Data

a. Insert

```
CREATE TABLE products (
    product_no integer,
    name text,
    price numeric
);
```

```sql
INSERT INTO products VALUES (1, 'Cheese', 9.99);
```

```sql
INSERT INTO products (product_no, name, price) VALUES (1, 'Cheese', 9.99);
INSERT INTO products (name, price, product_no) VALUES ('Cheese', 9.99, 1);
```

```sql
INSERT INTO products (product_no, name) VALUES (1, 'Cheese');
INSERT INTO products VALUES (1, 'Cheese');
```

Insert multiple row

```sql
INSERT INTO products (product_no, name, price) VALUES
    (1, 'Cheese', 9.99),
    (2, 'Bread', 1.99),
    (3, 'Milk', 2.99);
```

Insert dari hasil query

```sql
INSERT INTO products (product_no, name, price)
  SELECT product_no, name, price FROM new_products
    WHERE release_date = 'today';
```

b. Update

```sql
UPDATE products SET price = 10 WHERE price = 5;
```

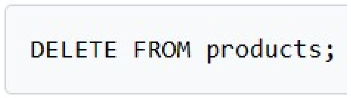
```sql
UPDATE products SET price = price * 1.10;
```

```sql
UPDATE mytable SET a = 5, b = 3, c = 1 WHERE a > 0;
```

c. Delete

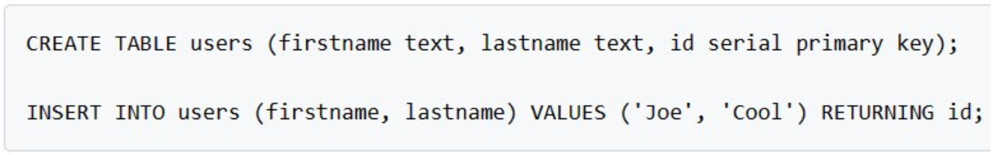
```sql
DELETE FROM products WHERE price = 10;
```

Delete semua row

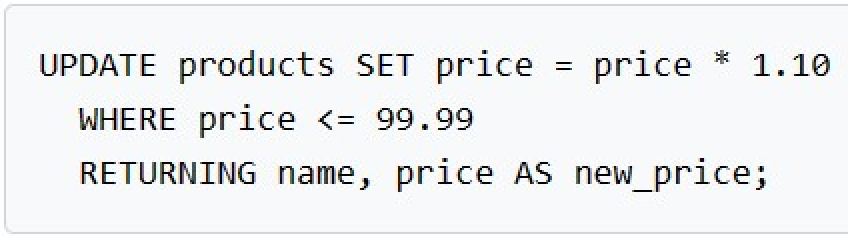


```
DELETE FROM products;
```

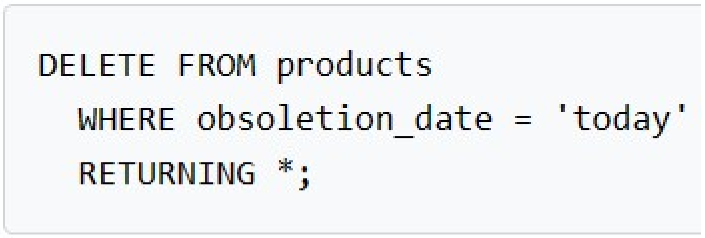
d. Returning



```
CREATE TABLE users (firstname text, lastname text, id serial primary key);

INSERT INTO users (firstname, lastname) VALUES ('Joe', 'Cool') RETURNING id;
```



```
UPDATE products SET price = price * 1.10
  WHERE price <= 99.99
  RETURNING name, price AS new_price;
```



```
DELETE FROM products
  WHERE obsoletion_date = 'today'
  RETURNING *;
```

H. Query

a. `FROM` clause that is optionally followed by `WHERE`, `GROUP BY`, and `HAVING` clauses

b. From bisa lebih dari 1 table menggunakan join

To put this together, assume we have tables `t1`:

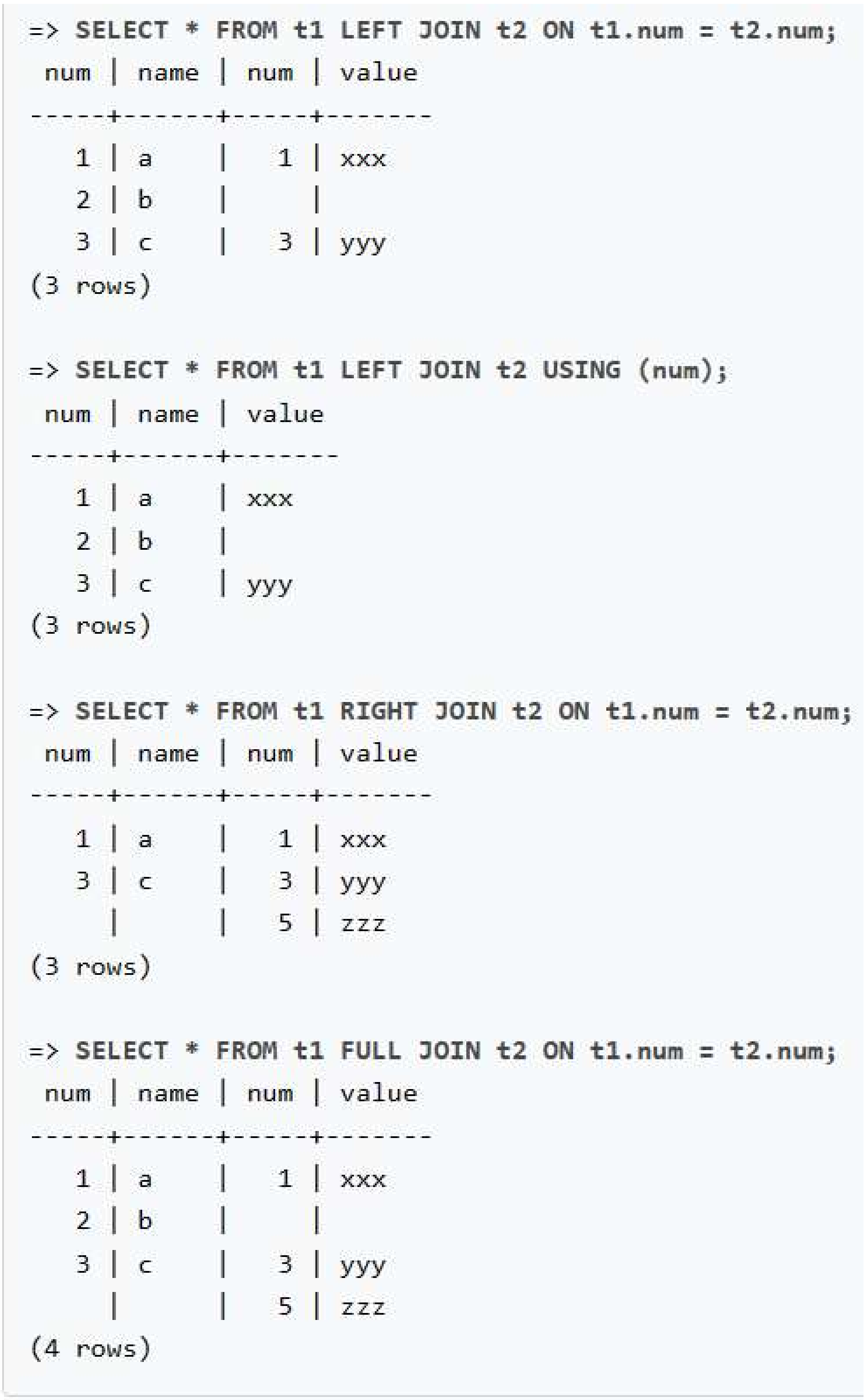
```
 num | name
-----+------
   1 | a
   2 | b
   3 | c
```

and `t2`:

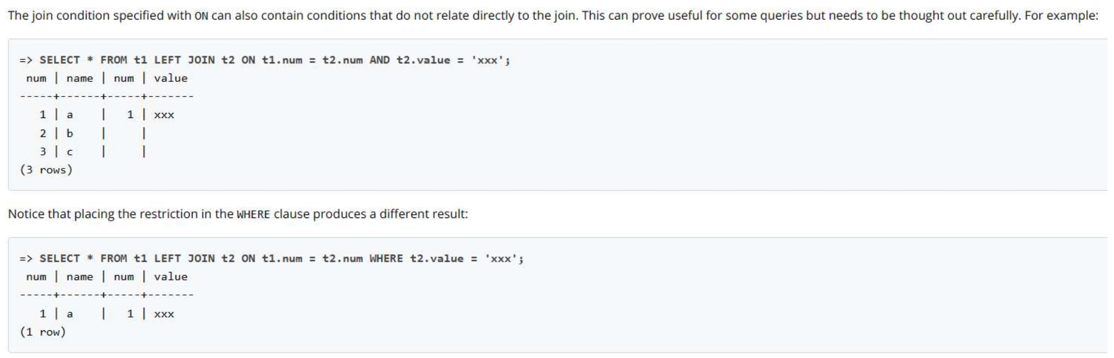
```
 num | value
-----+-------
   1 | xxx
   3 | yyy
   5 | zzz
```

then we get the following results for the various joins:

```
=> SELECT * FROM t1 CROSS JOIN t2;
 num | name | num | value
-----+------+-----+-------
   1 | a    |   1 | xxx
   1 | a    |   3 | yyy
   1 | a    |   5 | zzz
   2 | b    |   1 | xxx
   2 | b    |   3 | yyy
   2 | b    |   5 | zzz
   3 | c    |   1 | xxx
   3 | c    |   3 | yyy
   3 | c    |   5 | zzz
(9 rows)

=> SELECT * FROM t1 INNER JOIN t2 ON t1.num = t2.num;
 num | name | num | value
-----+------+-----+-------
   1 | a    |   1 | xxx
   3 | c    |   3 | yyy
(2 rows)

=> SELECT * FROM t1 INNER JOIN t2 USING (num);
 num | name | value
-----+------+-------
   1 | a    | xxx
   3 | c    | yyy
(2 rows)

=> SELECT * FROM t1 NATURAL INNER JOIN t2;
 num | name | value
-----+------+-------
   1 | a    | xxx
   3 | c    | yyy
(2 rows)
```



```
=> SELECT * FROM t1 LEFT JOIN t2 ON t1.num = t2.num;
 num | name | num | value
-----+------+-----+-------
   1 | a    |   1 | xxx
   2 | b    |     |
   3 | c    |   3 | yyy
(3 rows)

=> SELECT * FROM t1 LEFT JOIN t2 USING (num);
 num | name | value
-----+------+-------
   1 | a    | xxx
   2 | b    |
   3 | c    | yyy
(3 rows)

=> SELECT * FROM t1 RIGHT JOIN t2 ON t1.num = t2.num;
 num | name | num | value
-----+------+-----+-------
   1 | a    |   1 | xxx
   3 | c    |   3 | yyy
     |      |   5 | zzz
(3 rows)

=> SELECT * FROM t1 FULL JOIN t2 ON t1.num = t2.num;
 num | name | num | value
-----+------+-----+-------
   1 | a    |   1 | xxx
   2 | b    |     |
   3 | c    |   3 | yyy
     |      |   5 | zzz
(4 rows)
```

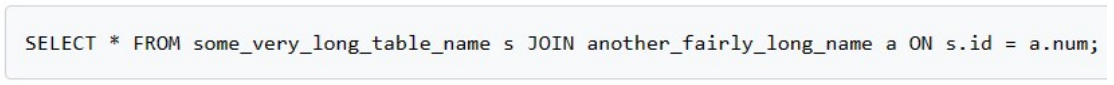

The join condition specified with ON can also contain conditions that do not relate directly to the join. This can prove useful for some queries but needs to be thought out carefully. For example:

```
=> SELECT * FROM t1 LEFT JOIN t2 ON t1.num = t2.num AND t2.value = 'xxx';
 num | name | num | value
-----+------+-----+-------
   1 | a    |   1 | xxx
   2 | b    |     |
   3 | c    |     |
(3 rows)
```

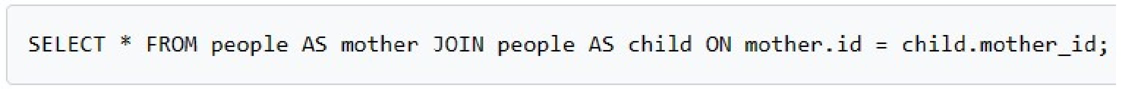
Notice that placing the restriction in the WHERE clause produces a different result:

```
=> SELECT * FROM t1 LEFT JOIN t2 ON t1.num = t2.num WHERE t2.value = 'xxx';
 num | name | num | value
-----+------+-----+-------
   1 | a    |   1 | xxx
(1 row)
```

c. Penggunaan alias



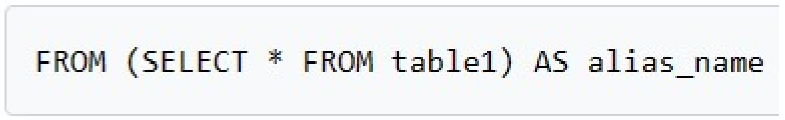
```
SELECT * FROM some_very_long_table_name s JOIN another_fairly_long_name a ON s.id = a.num;
```



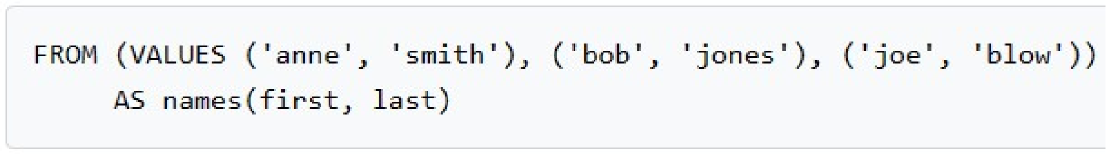
```
SELECT * FROM people AS mother JOIN people AS child ON mother.id = child.mother_id;
```

d. Subquery

Subqueries specifying a derived table must be enclosed in parentheses and *must* be assigned a table alias name.

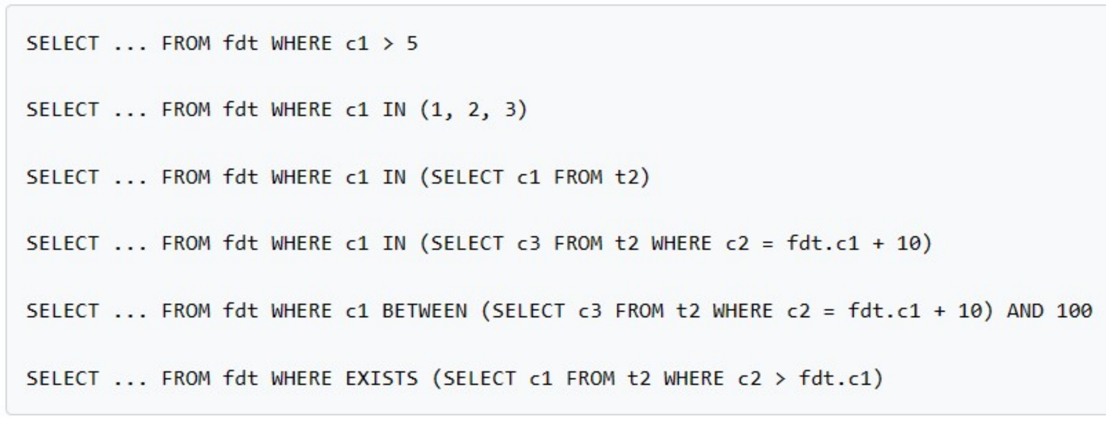


```
FROM (SELECT * FROM table1) AS alias_name
```



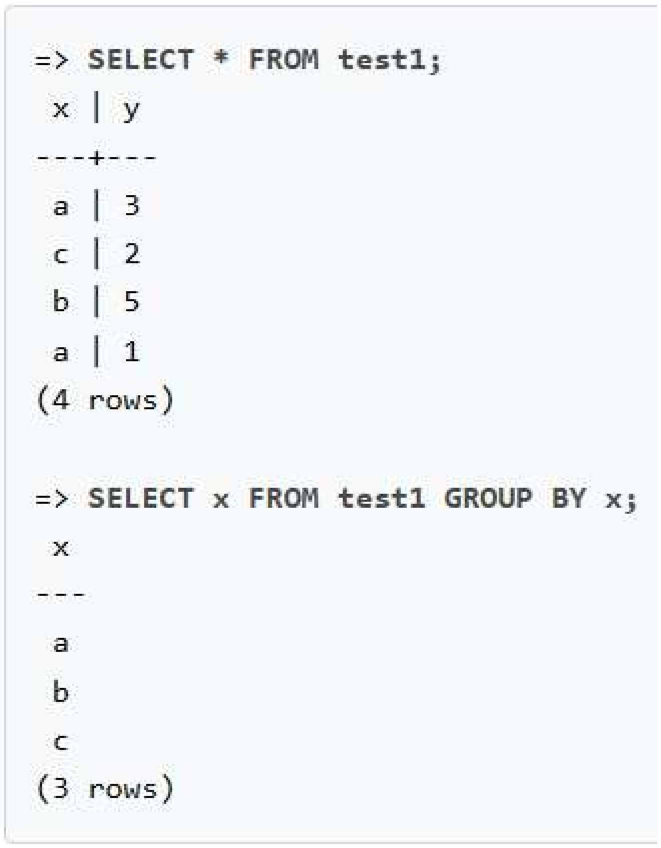
```
FROM (VALUES ('anne', 'smith'), ('bob', 'jones'), ('joe', 'blow'))
     AS names(first, last)
```

e. Where clause

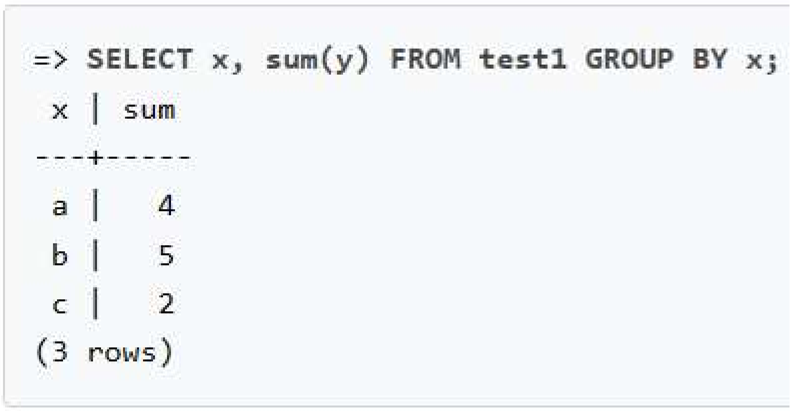


```
SELECT ... FROM fdt WHERE c1 > 5

SELECT ... FROM fdt WHERE c1 IN (1, 2, 3)

SELECT ... FROM fdt WHERE c1 IN (SELECT c1 FROM t2)

SELECT ... FROM fdt WHERE c1 IN (SELECT c3 FROM t2 WHERE c2 = fdt.c1 + 10)

SELECT ... FROM fdt WHERE c1 BETWEEN (SELECT c3 FROM t2 WHERE c2 = fdt.c1 + 10) AND 100

SELECT ... FROM fdt WHERE EXISTS (SELECT c1 FROM t2 WHERE c2 > fdt.c1)
```

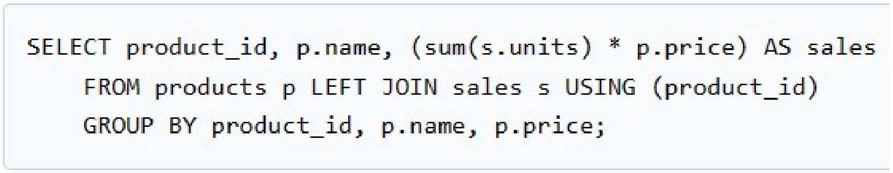
f. Penggunaan Group By



```
=> SELECT * FROM test1;
 x | y
---+---
 a | 3
 c | 2
 b | 5
 a | 1
(4 rows)

=> SELECT x FROM test1 GROUP BY x;
 x
---
 a
 b
 c
(3 rows)
```



```
=> SELECT x, sum(y) FROM test1 GROUP BY x;
 x | sum
---+-----
 a |   4
 b |   5
 c |   2
(3 rows)
```



```
SELECT product_id, p.name, (sum(s.units) * p.price) AS sales
    FROM products p LEFT JOIN sales s USING (product_id)
    GROUP BY product_id, p.name, p.price;
```

g. Penggunaan Having
   Having adalah kondisi yang digunakan setelah di group, sedangkan where kondisi sebelum di group

```
=> SELECT x, sum(y) FROM test1 GROUP BY x HAVING sum(y) > 3;
 x | sum
---+-----
 a |   4
 b |   5
(2 rows)

=> SELECT x, sum(y) FROM test1 GROUP BY x HAVING x < 'c';
 x | sum
---+-----
 a |   4
 b |   5
(2 rows)
```

```
SELECT product_id, p.name, (sum(s.units) * (p.price - p.cost)) AS profit
    FROM products p LEFT JOIN sales s USING (product_id)
    WHERE s.date > CURRENT_DATE - INTERVAL '4 weeks'
    GROUP BY product_id, p.name, p.price, p.cost
    HAVING sum(p.price * s.units) > 5000;
```

h. Select With

```
WITH regional_sales AS (
    SELECT region, SUM(amount) AS total_sales
    FROM orders
    GROUP BY region
), top_regions AS (
    SELECT region
    FROM regional_sales
    WHERE total_sales > (SELECT SUM(total_sales)/10 FROM regional_sales)
)
SELECT region,
       product,
       SUM(quantity) AS product_units,
       SUM(amount) AS product_sales
FROM orders
WHERE region IN (SELECT region FROM top_regions)
GROUP BY region, product;
```

```
WITH moved_rows AS (
    DELETE FROM products
    WHERE
        "date" >= '2010-10-01' AND
        "date" < '2010-11-01'
    RETURNING *
)
INSERT INTO products_log
SELECT * FROM moved_rows;
```

```
WITH t AS (
    UPDATE products SET price = price * 1.05
    RETURNING *
)
SELECT * FROM t;
```

i. Operator

The BETWEEN predicate simplifies range tests:

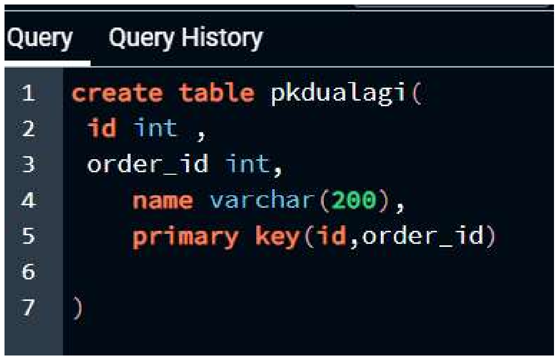
```
a BETWEEN x AND y
```

is equivalent to

```
a >= x AND a <= y
```

j.

I. Solusi multiple primary key not allowed:



```
create table pkdualagi(
 id int ,
 order_id int,
    name varchar(200),
    primary key(id,order_id)

)
```

J. Object Database lainnya

a. View
b. Function
c. Procedure
d. Rollback