

LAPORAN *FINAL PROJECT* PEMROGRAMAN BASIS DATA

“Database UpTown Cafe and Space”

Dosen Pengampu :
Hendri Kurniawan Prakosa, S.Kom.,M.Cs



Kelompok
Disusun Oleh :

Eunique Lydia Stephany	22.11.4545
Hamzah Miftah Farid	22.11.4561
Niko Yasin Febrian	22.11.4563
Thomas Wendra Atmadja	22.11.4586

**PROGRAM STUDI INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS AMIKOM YOGYAKARTA**

2024

Daftar Isi

Daftar Isi	2
Latar Belakang	4
ERD	5
Lampiran Hasil	6
Desain Basis Data:	9
Implementasi Fungsi	9
Implementasi <i>Trigger</i>	9
Implementasi <i>Procedure</i>	10
<i>FUNCTION</i>	11
<i>Function</i> untuk menghitung harga menu Food ditambah pajak 10%	11
Contoh <i>Query</i> memanggil <i>Function</i> di atas	12
<i>Function</i> untuk menghitung harga menu minuman ditambah pajak 10%	12
Contoh <i>Query</i> memanggil <i>Function</i> diatas	13
<i>TRIGGER</i>	15
<i>Trigger</i> untuk membuat <i>Id</i> pada <i>Table drink_categories</i>	15
<i>Trigger</i> untuk membuat <i>Id</i> pada <i>Table food_categories</i>	16
<i>Trigger</i> untuk membuat <i>Id</i> pada <i>Table drink_menu</i>	18
<i>Trigger</i> untuk membuat <i>Id</i> pada <i>Table food_menu</i>	20
<i>Trigger</i> untuk membuat <i>Id</i> pada <i>Table special_bundling</i>	22
Contoh <i>query</i> untuk menggunakan <i>trigger</i>	23
<i>PROCEDURE</i>	25
<i>Procedure</i> untuk mencari berdasarkan Nama	25
Contoh <i>Query</i> untuk memanggil <i>Procedure</i>	27
<i>Procedure</i> untuk mengubah status <i>food_menu</i>	27
<i>Procedure</i> untuk mengubah status <i>drink_menu</i>	29
Contoh <i>Query</i> untuk melakukan <i>procedure</i> ubah status	30
CALL <code>food_status('Nasi Goreng Uptown')</code>	30
<i>Procedure</i> untuk melihat detail <i>special_bundling</i>	31
Contoh <i>Query</i> memanggil <i>Procedure</i> di atas	33

<i>VIEW</i>	34
Membuat <i>View</i> Status <i>Not Available</i> untuk <i>drink_menu</i>	34
Membuat <i>View</i> Status <i>Not Available</i> untuk <i>food_menu</i>	35
Contoh <i>Query</i> Untuk Melihat isi <i>View</i>	36

Latar Belakang

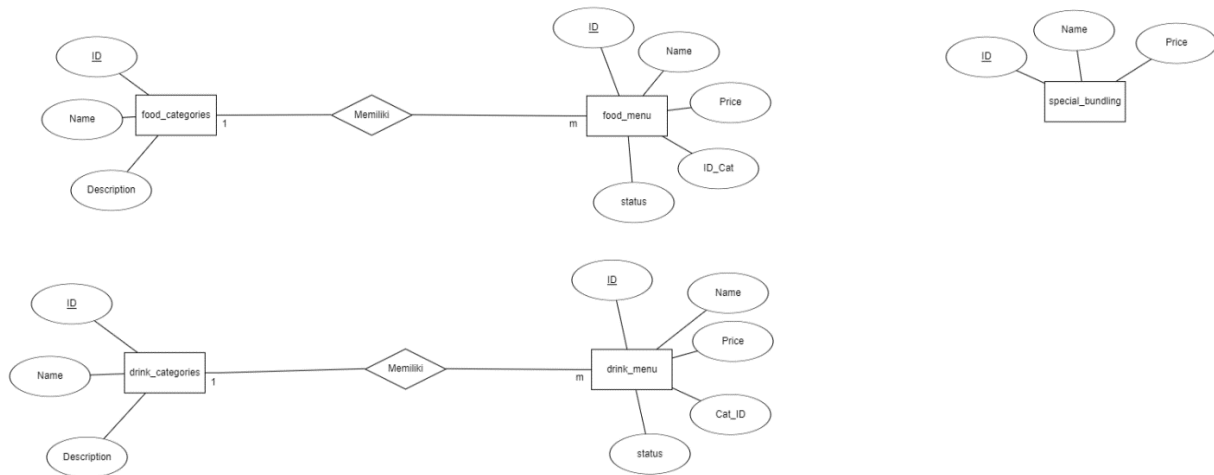
Sebuah usaha apabila semakin besar, maka akan semakin banyak data yang perlu diolah. Dalam hal ini, sebuah kafe memiliki banyak menu dan kategori menu. Seiring berjalannya waktu, kafe akan membuat terobosan beberapa menu baru sebagai salah satu strategi bisnisnya. Dalam konteks ini, pengelolaan data menu makanan dan minuman serta kategori-kategorinya merupakan komponen vital. Mengingat variasi menu yang ditawarkan restoran, termasuk makanan pembuka, utama, pencuci mulut, dan minuman, diperlukan pengelolaan data yang baik agar mengurangi terjadinya *human error*.

Penerapan sistem yang efektif dalam mengelola data ini tidak hanya memudahkan pengelolaan internal restoran tetapi juga meningkatkan pengalaman pelanggan. Pelanggan dapat dengan mudah melihat menu yang up-to-date, mengetahui kategori makanan, dan mendapatkan informasi yang jelas mengenai harga dan deskripsi produk. Selain itu, pemilik restoran dapat mengelola persediaan bahan baku lebih efisien, mengurangi pemborosan, dan memaksimalkan profitabilitas.

Untuk menerapkan solusi dari masalah yang ada, database berikut adalah realisasi yang dapat diterapkan, dengan fokus pada implementasi fungsi, prosedur, dan *trigger* dalam sistem manajemen basis data dengan harapan akan memudahkan kafe tersebut dalam melakukan pengolahan data lebih lanjut.

ERD

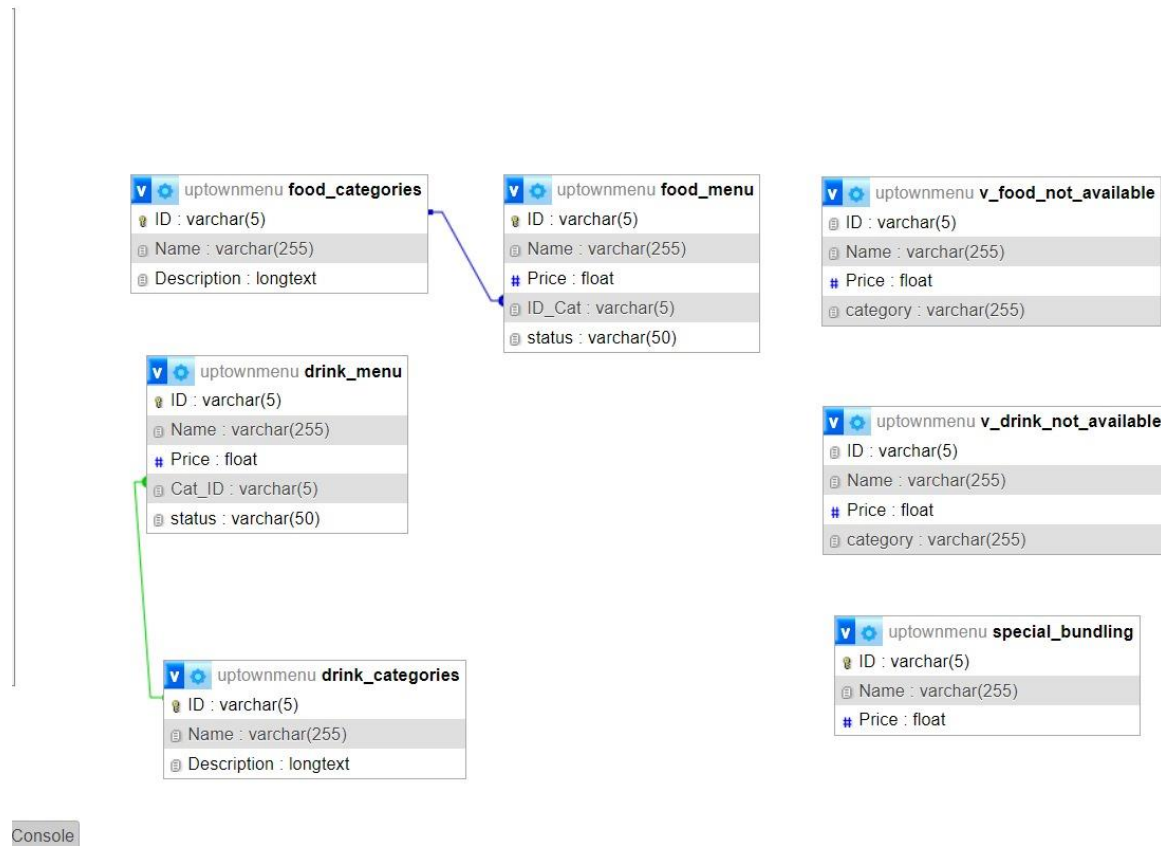
Tampilan ERD



Kardinalitas dan penjelasannya

1. Relasi antara '*food_categories*' dan '*food_menu*'
 - a. Kardinalitasnya adalah 1 to many
 - b. Artinya, setiap kategori makanan ('*food_categories*') dapat memiliki banyak item menu makanan (*food_menu*), tetapi setiap item menu makanan hanya dapat dimiliki oleh satu kategori makanan.
2. Relasi antara '*drink_categories*' dan '*drink_menu*'
 - a. Kardinalitasnya adalah 1 to many
 - b. Artinya, setiap kategori minuman ('*drink_categories*') dapat memiliki banyak item menu minuman (*drink_menu*), tetapi setiap item menu minuman hanya dapat dimiliki oleh satu kategori minuman.
3. Entitas '*special_bundling*'
 - a. *Special_bundling* adalah entitas yang berdiri sendiri dan tidak terhubung secara langsung dengan entitas lainnya.

Lampiran Hasil



- Drink_Categories**

	ID	Name	Description
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	DC001	Healthy Drink	Minuman sehat untuk kalian yang sedang menjaga asu...
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	DC002	Basic Coffee	Minuman dengan bahan dasar kopi yang ditambahkan v...
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	DC003	Traditional Drink	Minuman tradisional khas Indonesia yang terbuat da...
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	DC004	Coffee Addicted	Minuman original kopi dengan berbagai teknik penya...
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	DC005	Non Coffee	Minuman yang terbuat dari bahan selain kopi
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	DC006	Milkshake	Minuman yang terbuat dari campuran susu dengan bua...
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	DC007	Milkshake Float	Minuman yang terbuat dari campuran susu dengan bua...
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	DC008	Fruit + Ice Cream	Campuran es krim dan buah
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	DC009	Bread + Ice Cream	Variasi es krim yang dibalut dengan roti tawar dan...
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	DC010	Soda	Minuman bersoda dicampur dengan berbagai macam var...
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	DC011	Yakult	Minuman dengan bahan dasar yakult yang dicampur de...

☐ Check all ☐ With selected ☐ Edit ☐ Copy ☐ Delete ☐ Export

- Food_Categories**

				ID	Name	Description
<input type="checkbox"/>				FC001	Snack	Cemilan ringan dengan berbagai macam rasa
<input type="checkbox"/>				FC002	Rice Bowl	Sajian Makanan yang berisi Nasi dan varian Lauk di...
<input type="checkbox"/>				FC003	Nasi Goreng	Sajian Makanan Nasi Goreng yang berisikan topping ...
<input type="checkbox"/>				FC004	Nasi + Lauk	Sajian Nasi dengan berbagai lauk Pilihan
<input type="checkbox"/>				FC005	Spaghetti	Sajian olahan Pasta Spagehthti dengan topping saus ...
<input type="checkbox"/>				FC006	Mie	Sajian Makanan olahan berbahan dasar Mie
<input type="checkbox"/>				FC007	Sayur	Sajian Makanan olahan sayur dengan campuran varian...
<input type="checkbox"/>		Check all	With selected:			
						Export
















- **Drink_Menu**

				ID	Name	Price	Cat_ID	status
<input type="checkbox"/>				DR001	Jus Jambu	10000	DC001	Not Available
<input type="checkbox"/>				DR002	Jus Jeruk	10000	DC001	Available
<input type="checkbox"/>				DR003	Jus Tomat	10000	DC001	Available
<input type="checkbox"/>				DR004	Jus Wortel	10000	DC001	Available
<input type="checkbox"/>				DR005	Jus Strawberry	12000	DC001	Available
<input type="checkbox"/>				DR006	Jus Alpukat	14000	DC001	Available
<input type="checkbox"/>				DR007	Jus Buah Naga	14000	DC001	Available
<input type="checkbox"/>				DR008	Jus Mangga	14000	DC001	Not Available
<input type="checkbox"/>				DR009	Kopi Lemon Hot	9000	DC002	Available
<input type="checkbox"/>				DR010	Americano Hot	12000	DC002	Available
<input type="checkbox"/>				DR011	Cappucinno Hot	12000	DC002	Available
<input type="checkbox"/>				DR012	Kopi Latte Hot	12000	DC002	Available
<input type="checkbox"/>				DR013	Kopi Gula Aren Hot	12000	DC002	Available
<input type="checkbox"/>				DR014	Moccacino Latte Hot	12000	DC002	Available
<input type="checkbox"/>				DR015	Mojito Kopi Hot	13000	DC002	Available
<input type="checkbox"/>				DR016	Hazelnut Latte Hot	14000	DC002	Available
<input type="checkbox"/>				DR017	Vanilla Latte Hot	14000	DC002	Available
<input type="checkbox"/>				DR018	Caramel Latte Hot	14000	DC002	Available
<input type="checkbox"/>				DR019	Pandan Latte Hot	16000	DC002	Available
<input type="checkbox"/>				DR020	Almond Latte Hot	16000	DC002	Available
<input type="checkbox"/>				DR021	Kopi Lemon Ice	10000	DC002	Available
<input type="checkbox"/>				DR022	Americano Ice	13000	DC002	Available
<input checked="" type="checkbox"/>				DR023	Cappucinno Ice	13000	DC002	Available

- **Food_Menu**

<div><div>← T →</div><div>▼</div></div>				ID	Name	Price	ID_Cat	status			
<input type="checkbox"/>		Edit		Copy		Delete	FD001	Pisang Goreng	12000	FC001	Available
<input type="checkbox"/>		Edit		Copy		Delete	FD002	Roti Bakar	15000	FC001	Available
<input type="checkbox"/>		Edit		Copy		Delete	FD003	Singkong Goreng	11000	FC001	Available
<input type="checkbox"/>		Edit		Copy		Delete	FD004	Mendoan	10000	FC001	Available
<input type="checkbox"/>		Edit		Copy		Delete	FD005	Cireng	12000	FC001	Available
<input type="checkbox"/>		Edit		Copy		Delete	FD006	Tempura	12000	FC001	Available
<input type="checkbox"/>		Edit		Copy		Delete	FD007	French Fries	12000	FC001	Available
<input type="checkbox"/>		Edit		Copy		Delete	FD008	Nugget	12000	FC001	Available
<input type="checkbox"/>		Edit		Copy		Delete	FD009	Otak-Otak	12000	FC001	Available
<input type="checkbox"/>		Edit		Copy		Delete	FD010	Mix Platter	17000	FC001	Available
<input type="checkbox"/>		Edit		Copy		Delete	FD011	Onion Rings	19000	FC001	Available
<input type="checkbox"/>		Edit		Copy		Delete	FD012	Egg Ricebowl	12000	FC002	Available
<input type="checkbox"/>		Edit		Copy		Delete	FD013	Chicken Ricebowl	18000	FC002	Available
<input type="checkbox"/>		Edit		Copy		Delete	FD014	Seafood Ricebowl	22000	FC002	Available
<input type="checkbox"/>		Edit		Copy		Delete	FD015	Nasi Goreng Uptown	20000	FC003	Available
<input type="checkbox"/>		Edit		Copy		Delete	FD016	Nasi Goreng Bakso	14000	FC003	Available
<input type="checkbox"/>		Edit		Copy		Delete	FD017	Nasi Goreng Sosis	14000	FC003	Available
<input type="checkbox"/>		Edit		Copy		Delete	FD018	Nasi Goreng telur	12000	FC003	Available
<input type="checkbox"/>		Edit		Copy		Delete	FD019	Nasi Katsu Original	18000	FC004	Available
<input type="checkbox"/>		Edit		Copy		Delete	FD020	Nasi Ayam Geprek	15000	FC004	Available
<input type="checkbox"/>		Edit		Copy		Delete	FD021	Nasi Telur Pontianak	12000	FC004	Available
<input type="checkbox"/>		Edit		Copy		Delete	FD022	Nasi Telur Sambal	10000	FC004	Available

- **Special_Bundling**

				ID	Name	Price
<input type="checkbox"/>	 Edit	 Copy	 Delete	SB001	Paket Uptown 1	30000
<input type="checkbox"/>	 Edit	 Copy	 Delete	SB002	Paket Uptown 2	35000
<input type="checkbox"/>	 Edit	 Copy	 Delete	SB003	Paket Uptown 3	40000
<input type="checkbox"/>	 Edit	 Copy	 Delete	SB004	Paket Uptown 4	45000
<input type="checkbox"/>	 Edit	 Copy	 Delete	SB005	Paket Uptown 5	50000
<input type="checkbox"/>	 Edit	 Copy	 Delete	SB006	Paket Uptown 6	55000
<input type="checkbox"/>	 Edit	 Copy	 Delete	SB007	Paket Uptown 7	60000

Desain Basis Data:

- **Tabel Menu (*drink and food*):** Berisi informasi tentang setiap item makanan dan minuman, termasuk nama, deskripsi, harga, kategori, dan status ketersediaan.
- **Tabel Kategori:** Menyimpan berbagai kategori seperti makanan pembuka, makanan utama, pencuci mulut, dan minuman.
- **Tabel *Special Bundling*:** Berisi paket spesial yang dimiliki oleh kafe

Implementasi Fungsi

- **Hitung Total Pesanan:** Fungsi untuk menghitung total harga pesanan termasuk pajak, membantu dalam memberikan informasi yang akurat kepada pelanggan dan sistem kasir.

Implementasi *Trigger*

- ***Trigger* untuk Penginputan Stok:** Memastikan setiap produk yang dimasukkan menggunakan kode unik yang merupakan *primary key* dari tabel tersebut.

Implementasi *Procedure*

- ***Procedure* untuk melakukan perubahan status menu:** Mengubah status menu baik makanan dan minuman menjadi sebaliknya (misal semula *available* menjadi *not available*)
- ***Procedure* untuk melakukan pencarian *by name*:** memasukan nama sebagai kata kunci pencarian, jika ditemukan akan ditampilkan data yang terdapat di tabel
- ***Procedure* untuk menampilkan detail dari *special_bundling*:** *special bundling* menawarkan produk yang tidak tersedia di menu reguler, hal ini yang memberikan pengalaman berbeda bagi mereka yang mengambil *special bundling*.

FUNCTION

Function untuk menghitung harga menu Food ditambah pajak 10%

```
DELIMITER //
CREATE FUNCTION food_total_price (in_name VARCHAR(255))
RETURNS float
BEGIN
    DECLARE menu_price float;
    DECLARE total float;

    -- Mendapatkan harga item dari tabel menu berdasarkan Name
    SELECT Price INTO menu_price
    FROM food_menu
    WHERE Name = in_name;

    -- Menghitung total harga ditambah pajak 10%
    SET total = menu_price * 1.10;

    RETURN total;
END//
DELIMITER ;
```

Penjelasan :

DELIMITER //

Mengganti delimiter default dari ; menjadi // agar bisa menggunakan ; di dalam tubuh trigger tanpa menandakan akhir dari definisi trigger.

CREATE FUNCTION food_total_price (in_name VARCHAR(255)) RETURNS float
Mendefinisikan sebuah fungsi dengan nama food_total_price yang menerima satu parameter in_name dengan tipe data VARCHAR(255) dan mengembalikan nilai bertipe float.

BEGIN ... END

Blok BEGIN ... END mendefinisikan tubuh dari fungsi, yang berisi pernyataan-pernyataan yang akan dijalankan.

DECLARE menu_price float;

Mendeklarasikan variabel menu_price dengan tipe data float untuk menyimpan harga item makanan.

DECLARE total float;

Mendeklarasikan variabel total dengan tipe data float untuk menyimpan total harga setelah ditambah pajak.

SELECT Price INTO menu_price FROM food_menu WHERE Name = in_name; Mengambil harga item makanan dari tabel food_menu berdasarkan kolom Name yang sesuai dengan nilai in_name, dan menyimpannya ke dalam variabel menu_price.

SET total = menu_price * 1.10;

Menghitung total harga dengan menambahkan pajak 10% (harga asli dikali 1.10) dan menyimpannya ke dalam variabel total.

RETURN total;

Mengembalikan nilai total sebagai hasil dari fungsi.

Contoh *Query* memanggil *Function* di atas

```
SELECT food_total_price('Nasi Ayam Geprek')
```

The screenshot shows a web-based SQL interface. At the top, a green status bar indicates "Showing rows 0 - 0 (1 total, Query took 0.0003 seconds.)". Below this, the executed query is displayed: `SELECT food_total_price('Nasi Ayam Geprek');`. A toolbar offers options like "Profiling", "Edit inline", "Edit", "Explain SQL", "Create PHP code", and "Refresh". Below the toolbar, there are controls for "Show all", "Number of rows" (set to 25), and a "Filter rows" search box. An "Extra options" button is also present. The query result is shown in a table with one row: the function name `food_total_price('Nasi Ayam Geprek')` and its value `16500`. At the bottom, there is another set of controls for displaying the result table.

Function untuk menghitung harga menu minuman ditambah pajak 10%

```
DELIMITER //
CREATE FUNCTION drink_total_price (in_name VARCHAR(255))
RETURNS float
BEGIN
    DECLARE menu_price float;
    DECLARE total float;

    -- Mendapatkan harga item dari tabel menu berdasarkan Name
    SELECT Price INTO menu_price
    FROM drink_menu
    WHERE Name = in_name;

    -- Menghitung total harga ditambah pajak 10%
```

```
        SET total = menu_price * 1.10;

        RETURN total;
END//
DELIMITER ;
```

Penjelasan :

DELIMITER //

Mengganti delimiter default dari ; menjadi // agar kita bisa menggunakan ; di dalam tubuh fungsi tanpa menandakan akhir dari definisi fungsi.

CREATE FUNCTION drink_total_price (in_name VARCHAR(255)) RETURNS float
Mendefinisikan sebuah fungsi dengan nama drink_total_price yang menerima satu parameter in_name dengan tipe data VARCHAR(255) dan mengembalikan nilai bertipe float.

BEGIN ... END

Blok BEGIN ... END mendefinisikan tubuh dari fungsi, yang berisi pernyataan-pernyataan yang akan dijalankan.

DECLARE menu_price float;

Mendeklarasikan variabel menu_price dengan tipe data float untuk menyimpan harga item minuman.

DECLARE total float;

Mendeklarasikan variabel total dengan tipe data float untuk menyimpan total harga setelah ditambah pajak.

SELECT Price INTO menu_price FROM drink_menu WHERE Name = in_name; Mengambil harga item minuman dari tabel drink_menu berdasarkan kolom Name yang sesuai dengan nilai in_name, dan menyimpannya ke dalam variabel menu_price.

SET total = menu_price * 1.10;

Menghitung total harga dengan menambahkan pajak 10% (harga asli dikali 1.10) dan menyimpannya ke dalam variabel total.

RETURN total;

Mengembalikan nilai total sebagai hasil dari fungsi.

Contoh Query memanggil Function diatas

```
SELECT drink_total_price('Americano Hot')
```

✓ Showing rows 0 - 0 (1 total, Query took 0.0003 seconds.)

```
SELECT drink_total_price('Americano Hot');
```

☐ Profiling [[Edit inline](#)] [[Edit](#)] [[Explain SQL](#)] [[Create PHP code](#)] [[Refresh](#)]

☐ Show all | Number of rows: 25 ▼ Filter rows:

Extra options

drink_total_price('Americano Hot')

13200

☐ Show all | Number of rows: 25 ▼ Filter rows:

TRIGGER

Trigger untuk membuat Id pada Table *drink_categories*

```
DELIMITER //
```

```
CREATE TRIGGER id_for_cat_drink
BEFORE INSERT ON drink_categories
FOR EACH ROW
BEGIN
    DECLARE new_id VARCHAR(5);
    DECLARE prefix VARCHAR(2) DEFAULT 'DC';
    DECLARE number_part INT;

    -- Mengambil nilai ID terakhir dan memisahkan bagian angka
    SELECT IFNULL(MAX(CAST(SUBSTRING(ID, 3, 3) AS UNSIGNED)), 0) + 1
    INTO number_part
    FROM drink_categories
    WHERE id LIKE CONCAT(prefix, '%');

    -- Membuat ID baru
    SET new_id = CONCAT(prefix, LPAD(number_part, 3, '0'));

    -- Mengatur nilai ID baru
    SET NEW.ID = new_id;
END//
```

```
DELIMITER ;
```

Penjelasan :

DELIMITER //

Mengganti delimiter default dari ; menjadi // agar bisa menggunakan ; di dalam tubuh trigger tanpa menandakan akhir dari definisi trigger.

CREATE TRIGGER id_for_cat_drink

Membuat sebuah trigger dengan nama id_for_cat_drink.

BEFORE INSERT ON drink_categories

Menentukan bahwa trigger ini akan dijalankan sebelum operasi INSERT pada tabel drink_categories.

FOR EACH ROW

Menentukan bahwa trigger ini akan dijalankan untuk setiap baris yang dimasukkan ke dalam tabel.

BEGIN ... END Blok BEGIN ... END mendefinisikan tubuh dari trigger, yang berisi pernyataan-pernyataan yang akan dijalankan.

```
DECLARE new_id VARCHAR(5);
```

Mendeklarasikan variabel new_id dengan tipe data VARCHAR(5) untuk menyimpan ID baru yang akan dibuat.

```
DECLARE prefix VARCHAR(2) DEFAULT 'DC';
```

Mendeklarasikan variabel prefix dengan tipe data VARCHAR(2) dan nilai default 'DC'. Ini akan digunakan sebagai prefix untuk ID baru.

```
DECLARE number_part INT;
```

Mendeklarasikan variabel number_part dengan tipe data INT untuk menyimpan bagian angka dari ID.

```
SELECT IFNULL(MAX(CAST(SUBSTRING(ID, 3, 3) AS UNSIGNED)), 0) + 1 INTO  
number_part
```

Mengambil nilai ID terakhir dari tabel drink_categories, memisahkan bagian angka dari ID, dan mengubahnya menjadi tipe UNSIGNED integer. Kemudian menambahkan 1 untuk menghasilkan nilai angka berikutnya. Jika tidak ada ID yang cocok, maka nilai default adalah 0.

```
FROM drink_categories WHERE id LIKE CONCAT(prefix, '%');
```

Memfilter ID yang sesuai dengan prefix 'DC'.

```
SET new_id = CONCAT(prefix, LPAD(number_part, 3, '0'));
```

Menggabungkan prefix 'DC' dengan bagian angka yang di-padded dengan nol di sebelah kiri hingga mencapai panjang 3 karakter, untuk menghasilkan ID baru.

```
SET NEW.ID = new_id;
```

Mengatur nilai ID baru yang dihasilkan ke kolom ID dari baris baru yang akan dimasukkan ke dalam tabel.

Trigger untuk membuat Id pada Table food_categories

```
DELIMITER //  
  
CREATE TRIGGER id_for_cat_food  
BEFORE INSERT ON food_categories  
FOR EACH ROW  
BEGIN  
    DECLARE new_id VARCHAR(5);  
    DECLARE prefix VARCHAR(2) DEFAULT 'FC';  
    DECLARE number_part INT;
```

```

-- Mengambil nilai ID terakhir dan memisahkan bagian angka
SELECT IFNULL(MAX(CAST(SUBSTRING(ID, 3, 3) AS UNSIGNED)), 0) + 1
INTO number_part
FROM food_categories
WHERE id LIKE CONCAT(prefix, '%');

-- Membuat ID baru
SET new_id = CONCAT(prefix, LPAD(number_part, 3, '0'));

-- Mengatur nilai ID baru
SET NEW.ID = new_id;
END//

DELIMITER ;

```

Penjelasan :

DELIMITER //

Mengganti delimiter default dari ; menjadi // agar kita bisa menggunakan ; di dalam tubuh trigger tanpa menandakan akhir dari definisi trigger.

CREATE TRIGGER id_for_cat_food

Membuat sebuah trigger dengan nama id_for_cat_food.

BEFORE INSERT ON food_categories

Menentukan bahwa trigger ini akan dijalankan sebelum operasi INSERT pada tabel food_categories.

FOR EACH ROW

Menentukan bahwa trigger ini akan dijalankan untuk setiap baris yang dimasukkan ke dalam tabel.

BEGIN ... END

Blok BEGIN ... END mendefinisikan tubuh dari trigger, yang berisi pernyataan-pernyataan yang akan dijalankan.

DECLARE new_id VARCHAR(5);

Mendeklarasikan variabel new_id dengan tipe data VARCHAR(5) untuk menyimpan ID baru yang akan dibuat.

DECLARE prefix VARCHAR(2) DEFAULT 'FC';

Mendeklarasikan variabel prefix dengan tipe data VARCHAR(2) dan nilai default 'FC'. Ini akan digunakan sebagai prefix untuk ID baru.

```
DECLARE number_part INT;
```

Mendeklarasikan variabel number_part dengan tipe data INT untuk menyimpan bagian angka dari ID.

```
SELECT IFNULL(MAX(CAST(SUBSTRING(ID, 3, 3) AS UNSIGNED)), 0) + 1 INTO  
number_part
```

Mengambil nilai ID terakhir dari tabel food_categories, memisahkan bagian angka dari ID, dan mengubahnya menjadi tipe UNSIGNED integer. Kemudian menambahkan 1 untuk menghasilkan nilai angka berikutnya. Jika tidak ada ID yang cocok, maka nilai default adalah 0.

```
FROM food_categories WHERE id LIKE CONCAT(prefix, '%');
```

Memfilter ID yang sesuai dengan prefix 'FC'.

```
SET new_id = CONCAT(prefix, LPAD(number_part, 3, '0'));
```

Menggabungkan prefix 'FC' dengan bagian angka yang di-padded dengan nol di sebelah kiri hingga mencapai panjang 3 karakter, untuk menghasilkan ID baru.

```
SET NEW.ID = new_id;
```

Mengatur nilai ID baru yang dihasilkan ke kolom ID dari baris baru yang akan dimasukkan ke dalam tabel.

Trigger untuk membuat Id pada Table drink_menu

```
DELIMITER //
```



```
CREATE TRIGGER id_for_drink  
BEFORE INSERT ON drink_menu  
FOR EACH ROW  
BEGIN  
    DECLARE new_id VARCHAR(5);  
    DECLARE prefix VARCHAR(2) DEFAULT 'DR';  
    DECLARE number_part INT;  
  
    -- Mengambil nilai ID terakhir dan memisahkan bagian angka  
    SELECT IFNULL(MAX(CAST(SUBSTRING(ID, 3, 3) AS UNSIGNED)), 0) + 1  
    INTO number_part  
    FROM drink_menu  
    WHERE id LIKE CONCAT(prefix, '%');
```



```
    -- Membuat ID baru  
    SET new_id = CONCAT(prefix, LPAD(number_part, 3, '0'));
```

```
-- Mengatur nilai ID baru
SET NEW.ID = new_id;
END//

DELIMITER ;
```

Penjelasan :

DELIMITER //

Mengganti delimiter default dari ; menjadi // agar kita bisa menggunakan ; di dalam tubuh trigger tanpa menandakan akhir dari definisi trigger.

CREATE TRIGGER id_for_drink

Membuat sebuah trigger dengan nama id_for_drink.

BEFORE INSERT ON drink_menu

Menentukan bahwa trigger ini akan dijalankan sebelum operasi INSERT pada tabel drink_menu.

FOR EACH ROW

Menentukan bahwa trigger ini akan dijalankan untuk setiap baris yang dimasukkan ke dalam tabel.

BEGIN ... END

Blok BEGIN ... END mendefinisikan tubuh dari trigger, yang berisi pernyataan-pernyataan yang akan dijalankan.

DECLARE new_id VARCHAR(5);

Mendeklarasikan variabel new_id dengan tipe data VARCHAR(5) untuk menyimpan ID baru yang akan dibuat.

DECLARE prefix VARCHAR(2) DEFAULT 'DR';

Mendeklarasikan variabel prefix dengan tipe data VARCHAR(2) dan nilai default 'DR'. Ini akan digunakan sebagai prefix untuk ID baru.

DECLARE number_part INT;

Mendeklarasikan variabel number_part dengan tipe data INT untuk menyimpan bagian angka dari ID.

SELECT IFNULL(MAX(CAST(SUBSTRING(ID, 3, 3) AS UNSIGNED)), 0) + 1 INTO
number_part

Mengambil nilai ID terakhir dari tabel `drink_menu`, memisahkan bagian angka dari ID, dan mengubahnya menjadi tipe UNSIGNED integer. Kemudian menambahkan 1 untuk menghasilkan nilai angka berikutnya. Jika tidak ada ID yang cocok, maka nilai default adalah 0.

```
FROM drink_menu WHERE id LIKE CONCAT(prefix, '%');
```

Memfilter ID yang sesuai dengan prefix 'DR'.

```
SET new_id = CONCAT(prefix, LPAD(number_part, 3, '0'));
```

Menggabungkan prefix 'DR' dengan bagian angka yang di-padded dengan nol di sebelah kiri hingga mencapai panjang 3 karakter, untuk menghasilkan ID baru.

```
SET NEW.ID = new_id;
```

Mengatur nilai ID baru yang dihasilkan ke kolom ID dari baris baru yang akan dimasukkan ke dalam tabel.

Trigger untuk membuat Id pada Table food_menu

```
DELIMITER //
```

```
CREATE TRIGGER id_for_food
BEFORE INSERT ON food_menu
FOR EACH ROW
BEGIN
    DECLARE new_id VARCHAR(5);
    DECLARE prefix VARCHAR(2) DEFAULT 'FD';
    DECLARE number_part INT;

    -- Mengambil nilai ID terakhir dan memisahkan bagian angka
    SELECT IFNULL(MAX(CAST(SUBSTRING(ID, 3, 3) AS UNSIGNED)), 0) + 1
    INTO number_part
    FROM food_menu
    WHERE id LIKE CONCAT(prefix, '%');

    -- Membuat ID baru
    SET new_id = CONCAT(prefix, LPAD(number_part, 3, '0'));

    -- Mengatur nilai ID baru
    SET NEW.ID = new_id;
END//
```

```
DELIMITER ;
```

Penjelasan :

DELIMITER //

Mengganti delimiter default dari ; menjadi // untuk mendefinisikan trigger dengan benar.

CREATE TRIGGER id_for_food

Mendefinisikan trigger dengan nama id_for_food.

BEFORE INSERT ON food_menu

Menentukan bahwa trigger ini akan dijalankan sebelum baris baru dimasukkan ke dalam tabel food_menu.

FOR EACH ROW

Menentukan bahwa trigger ini akan diterapkan untuk setiap baris yang dimasukkan.

BEGIN ... END

Blok yang mendefinisikan tubuh dari trigger.

DECLARE new_id VARCHAR(5);

Mendeklarasikan variabel new_id untuk menyimpan ID baru yang akan dibuat.

DECLARE prefix VARCHAR(2) DEFAULT 'FD';

Mendeklarasikan variabel prefix dengan nilai default 'FD' yang akan digunakan sebagai awalan ID.

DECLARE number_part INT;

Mendeklarasikan variabel number_part untuk menyimpan bagian angka dari ID.

SELECT IFNULL(MAX(CAST(SUBSTRING(ID, 3, 3) AS UNSIGNED)), 0) + 1 INTO number_part

Mengambil nilai ID tertinggi dari tabel yang dimulai dengan prefix 'FD', mengonversi bagian angka ke integer, dan menambahkan 1. Jika tidak ada ID yang cocok, nilai default adalah 0.

FROM food_menu WHERE id LIKE CONCAT(prefix, '%');

Mengambil ID dari tabel yang memiliki awalan yang sama dengan prefix.

SET new_id = CONCAT(prefix, LPAD(number_part, 3, '0'));

Membuat ID baru dengan awalan 'FD' dan menambahkan angka yang di-padded dengan nol di sebelah kiri agar panjangnya 3 digit.

SET NEW.ID = new_id;

Mengatur kolom ID pada baris baru yang akan dimasukkan dengan nilai new_id.

Trigger untuk membuat Id pada Table special_bundling

```
DELIMITER //  
  
CREATE TRIGGER id_for_special_bundling  
BEFORE INSERT ON special_bundling  
FOR EACH ROW  
BEGIN  
    DECLARE new_id VARCHAR(5);  
    DECLARE prefix VARCHAR(2) DEFAULT 'SB';  
    DECLARE number_part INT;  
  
    -- Mengambil nilai ID terakhir dan memisahkan bagian angka  
    SELECT IFNULL(MAX(CAST(SUBSTRING(ID, 3, 3) AS UNSIGNED)), 0) + 1  
    INTO number_part  
    FROM special_bundling  
    WHERE id LIKE CONCAT(prefix, '%');  
  
    -- Membuat ID baru  
    SET new_id = CONCAT(prefix, LPAD(number_part, 3, '0'));  
  
    -- Mengatur nilai ID baru  
    SET NEW.ID = new_id;  
END//  
  
DELIMITER ;
```

Penjelasan :

DELIMITER //

Mengganti delimiter default dari ; menjadi // untuk memungkinkan penggunaan ; di dalam tubuh trigger tanpa menandakan akhir dari definisi trigger.

CREATE TRIGGER id_for_special_bundling

Membuat sebuah trigger dengan nama id_for_special_bundling.

BEFORE INSERT ON special_bundling

Menentukan bahwa trigger ini akan dijalankan sebelum baris baru dimasukkan ke dalam tabel special_bundling.

FOR EACH ROW

Menentukan bahwa trigger ini akan dijalankan untuk setiap baris yang dimasukkan.

BEGIN ... END

Blok yang mendefinisikan tubuh dari trigger.

```
DECLARE new_id VARCHAR(5);
```

Mendeklarasikan variabel `new_id` untuk menyimpan ID baru yang akan dibuat.

```
DECLARE prefix VARCHAR(2) DEFAULT 'SB';
```

Mendeklarasikan variabel `prefix` dengan nilai default 'SB', yang akan digunakan sebagai awalan ID.

```
DECLARE number_part INT;
```

Mendeklarasikan variabel `number_part` untuk menyimpan bagian angka dari ID.

```
SELECT IFNULL(MAX(CAST(SUBSTRING(ID, 3, 3) AS UNSIGNED)), 0) + 1 INTO  
number_part
```

Mengambil nilai ID tertinggi dari tabel yang dimulai dengan prefix 'SB', mengonversi bagian angka dari ID ke tipe UNSIGNED integer, dan menambahkan 1. Jika tidak ada ID yang cocok, nilai default adalah 0.

```
FROM special_bundling WHERE id LIKE CONCAT(prefix, '%');
```

Mengambil ID yang sesuai dengan prefix 'SB' dari tabel `special_bundling`.

```
SET new_id = CONCAT(prefix, LPAD(number_part, 3, '0'));
```

Membuat ID baru dengan menggabungkan prefix 'SB' dengan angka yang dipadding dengan nol di sebelah kiri hingga panjangnya mencapai 3 digit.

```
SET NEW.ID = new_id;
```

Mengatur kolom ID pada baris baru yang akan dimasukkan dengan nilai `new_id`.

Contoh *query* untuk menggunakan *trigger*

```
INSERT INTO food_menu(Name, Price, category, status) VALUES  
('Ayam Bakar', '20000', 'FC004', 'Not Available')
```

```
INSERT INTO food_menu(Name, Price, category, status) VALUES  
('Es Teller', '22000', 'DC003', 'Not Available')
```

```
INSERT INTO food_categories(Name, Description) VALUES  
('lainnya', 'Menu lainnya')
```

```
INSERT INTO drink_categories(Name, Description) VALUES  
    ('lainnya', 'Menu lainnya')
```

```
INSERT INTO special_bundling(Name, Price) VALUES  
    ('PaHel', '39000')
```

PROCEDURE

Procedure untuk mencari berdasarkan Nama

```
DELIMITER //
CREATE PROCEDURE search_by_name(IN input_name VARCHAR(255))
BEGIN
    DECLARE item_found INT DEFAULT 0;
    -- Cari item di food_menu
    IF EXISTS (SELECT 1 FROM food_menu WHERE name = input_name) THEN
        SELECT f.ID, f.Name, f.Price, c.Name FROM food_menu f
        JOIN food_categories c ON ID_Cat = c.ID
        WHERE f.Name = input_name;
        SET item_found = 1;
    END IF;
    -- Jika tidak ditemukan di food_menu, cek di drink_menu
    IF item_found = 0 THEN
        IF EXISTS (SELECT 1 FROM drink_menu WHERE name = input_name) THEN
            SELECT d.ID, d.Name, d.Price, c.Name
            FROM drink_menu d
            JOIN drink_categories c ON Cat_ID = c.ID
            WHERE d.Name = input_name;
            SET item_found = 1;
        END IF;
    END IF;
    -- Jika tidak ditemukan di kedua tabel
    IF item_found = 0 THEN
        SELECT 'Item tidak ditemukan' AS result;
    END IF;
END //
DELIMITER ;
```

Penjelasan :

DELIMITER //

Mengganti delimiter default dari ; menjadi // untuk mendefinisikan stored procedure dengan benar tanpa mengganggu perintah ; di dalam tubuh stored procedure.

CREATE PROCEDURE search_by_name(IN input_name VARCHAR(255))

Mendefinisikan stored procedure bernama search_by_name dengan satu parameter input input_name bertipe VARCHAR(255).

BEGIN ... END

Blok yang mendefinisikan tubuh dari stored procedure.

```
DECLARE item_found INT DEFAULT 0;
```

Mendeklarasikan variabel item_found untuk melacak apakah item yang dicari ditemukan atau tidak. Nilai default adalah 0 (tidak ditemukan).

```
IF EXISTS (SELECT 1 FROM food_menu WHERE name = input_name) THEN
```

Mengecek apakah ada item dengan nama yang sama di tabel food_menu. Jika ada, lakukan query berikutnya:

Mengambil ID, nama, harga, dan kategori makanan dari tabel food_menu dan food_categories.

```
SET item_found = 1;
```

Mengatur item_found menjadi 1 untuk menandakan bahwa item telah ditemukan.

```
IF item_found = 0 THEN
```

Mengecek jika item_found masih 0, berarti item belum ditemukan di tabel food_menu. Kemudian, cek di tabel drink_menu:

```
IF EXISTS (SELECT 1 FROM drink_menu WHERE name = input_name) THEN
```

Mengecek apakah ada item dengan nama yang sama di tabel drink_menu. Jika ada, lakukan query berikutnya:

Mengambil ID, nama, harga, dan kategori minuman dari tabel drink_menu dan drink_categories.

```
SET item_found = 1;
```

Mengatur item_found menjadi 1 untuk menandakan bahwa item telah ditemukan.

```
IF item_found = 0 THEN
```

Mengecek jika item_found masih 0 setelah pencarian di kedua tabel. Jika tidak ditemukan, tampilkan pesan:

Contoh Query untuk memanggil *Procedure*

```
CALL search_by_name('Mendoan');
```

```
CALL search_by_name('Jus Jambu')
```

✓ Showing rows 0 - 0 (1 total, Query took 0.0005 seconds.)

```
CALL search_by_name('Mendoan');
```

[Edit inline] [Edit] [Create PHP code]

☐ Show all | Number of rows: 25 ▾ | Filter rows

Extra options

ID	Name	Price	Name
FD004	Mendoan	10000	Snack

✓ Showing rows 0 - 0 (1 total, Query took 0.0005 seconds.)

```
CALL search_by_name('Jus Jambu');
```

[Edit inline] [Edit] [Create PHP code]

☐ Show all | Number of rows: 25 ▾ | Filter rows

Extra options

ID	Name	Price	Name
DR001	Jus Jambu	10000	Healthy Drink

Procedure untuk mengubah status *food_menu*

```
DELIMITER //  
CREATE PROCEDURE food_status(IN input_name VARCHAR(255))  
BEGIN  
    DECLARE current_status varchar(50);  
    SELECT status INTO current_status FROM food_menu WHERE Name =  
    input_name;  
    IF(current_status = 'Available') THEN  
        UPDATE food_menu SET status = 'Not Available' WHERE Name = input_name;  
    ELSE  
        UPDATE food_menu SET status = 'Available' WHERE Name = input_name;  
    END IF;  
    SELECT * FROM food_menu WHERE Name = input_name;  
END//  
DELIMITER ;
```

Penjelasan:

DELIMITER //

Mengganti delimiter default dari ; menjadi // untuk mendefinisikan stored procedure dengan benar, tanpa mengganggu penggunaan ; di dalam tubuh prosedur.

```
CREATE PROCEDURE food_status(IN input_name VARCHAR(255))
```

Mendefinisikan stored procedure bernama food_status dengan satu parameter input input_name bertipe VARCHAR(255).

```
BEGIN ... END
```

Blok yang mendefinisikan tubuh dari stored procedure.

```
DECLARE current_status VARCHAR(50);
```

Mendeklarasikan variabel current_status untuk menyimpan status saat ini dari item makanan yang dicari.

```
SELECT status INTO current_status FROM food_menu WHERE Name = input_name;
```

Mengambil status dari tabel food_menu berdasarkan nama item yang diberikan dan menyimpannya dalam variabel current_status.

```
IF (current_status = 'Available') THEN
```

Memeriksa apakah status saat ini dari item adalah 'Available'. Jika benar, lakukan update berikut:

Mengubah status item menjadi 'Not Available'.

```
ELSE
```

Mengubah status item menjadi 'Available'.

```
END IF;
```

Menandakan akhir dari blok kondisi IF.

```
SELECT * FROM food_menu WHERE Name = input_name;
```

Mengambil dan menampilkan seluruh informasi item makanan dengan nama yang diberikan setelah status diubah.

```
END //
```

Menandakan akhir dari definisi stored procedure.

Procedure untuk mengubah status drink_menu

```
DELIMITER //
CREATE PROCEDURE drink_status(IN input_name VARCHAR(255))
BEGIN
DECLARE current_status varchar(50);
SELECT status INTO current_status FROM drink_menu WHERE Name =
input_name;
IF(current_status = 'Available') THEN
UPDATE drink_menu SET status = 'Not Available' WHERE Name = input_name;
ELSE
UPDATE drink_menu SET status = 'Available' WHERE Name = input_name;
END IF;
SELECT * FROM drink_menu WHERE Name = input_name;
END//
DELIMITER ;
```

Penjelasan :

DELIMITER //

Mengganti delimiter default dari ; menjadi // untuk mendefinisikan stored procedure dengan benar, sehingga tidak mengganggu penggunaan ; di dalam tubuh prosedur.

CREATE PROCEDURE drink_status(IN input_name VARCHAR(255))

Mendefinisikan stored procedure bernama drink_status dengan satu parameter input input_name bertipe VARCHAR(255).

BEGIN ... END

Blok yang mendefinisikan tubuh dari stored procedure.

DECLARE current_status VARCHAR(50);

Mendeklarasikan variabel current_status untuk menyimpan status saat ini dari item minuman yang dicari.

SELECT status INTO current_status FROM drink_menu WHERE Name = input_name;

Mengambil status dari tabel drink_menu berdasarkan nama item yang diberikan dan menyimpannya dalam variabel current_status.

IF (current_status = 'Available') THEN

Mengecek apakah status saat ini dari item adalah 'Available'. Jika benar, lakukan update berikut:

Mengubah status item menjadi 'Not Available'.

ELSE

Jika status saat ini bukan 'Available', maka:

Mengubah status item menjadi 'Available'.

END IF;

Menandakan akhir dari blok kondisi IF.

SELECT * FROM drink_menu WHERE Name = input_name;

Mengambil dan menampilkan seluruh informasi item minuman dengan nama yang diberikan setelah status diubah.

END //

Menandakan akhir dari definisi stored procedure.

Contoh *Query* untuk melakukan *procedure* ubah status

```
CALL drink_status('Jus Jambu')
```

✓ Showing rows 0 - 0 (1 total, Query took 0.0015 seconds.)

```
CALL drink_status('Jus Jambu');
```

[Edit inline] [Edit] [Create PHP code]

☐ Show all | Number of rows: 25 ▼ Filter rows

Extra options

ID	Name	Price	Cat_ID	status
DR001	Jus Jambu	10000	DC001	Available

```
CALL food_status('Nasi Goreng Uptown')
```

Show query box

✓ Showing rows 0 - 0 (1 total, Query took 0.0014 seconds.)

```
CALL food_status('Nasi Goreng Uptown');
```

[[Edit inline](#)] [[Edit](#)] [[Create PHP code](#)]

☐ Show all | Number of rows: 25 ▼ Filter rows:

Extra options

ID	Name	Price	ID_Cat	status
FD015	Nasi Goreng Uptown	20000	FC003	Not Available

Procedure untuk melihat detail special_bundling

```
DELIMITER //
CREATE PROCEDURE detail_bundling(IN input_name VARCHAR(255))
BEGIN
    DECLARE menu_details varchar(255);
    DECLARE price_details float;

    -- mendapat harga
    SELECT Price INTO price_details FROM special_bundling WHERE Name
= input_name;

    -- mendapat detail sesuai nama
CASE input_name
    WHEN 'Paket UpTown 1' THEN
        SET menu_details = 'nasi, soup, pilihan ayam/ikan, pilihan masakan
sayuran, sambal, kerupuk, buah, teh, air mineral';
    WHEN 'Paket UpTown 2' THEN
        SET menu_details = 'nasi, pilihan soup, pilihan ayam/ikan, tumisan
sayur, pilihan lauk pendamping, sambal, kerupuk, buah, teh, air
mineral';
    WHEN 'Paket UpTown 3' THEN
        SET menu_details = 'nasi, pilihan soup, pilihan ayam/ikan, tumisan
sayur, lauk pendamping, sambal, kerupuk, buah, teh, mineral, soft
drink';
    WHEN 'Paket UpTown 4' THEN
        SET menu_details = 'nasi, pilihan soup, pilihan ayam/ikan, tumisan
sayur, lauk pendamping, sambal, kerupuk, buah, teh, air mineral, es
krim soft drink' ;
    WHEN 'Paket UpTown 5' THEN
```

```

        SET menu_details = 'nasi, pilihan soup, pilihan ayam/ikan, pilihan
masakan daging, tumisan sayur, sambal, kerupuk, pudding, es krim, buah,
teh, air mineral, soft drink' ;
        WHEN 'Paket UpTown 6' THEN
            SET menu_details = 'nasi, pilihan soup, pilihan masakan ayam,
pilihan masakan daging, pilihan masakan ikan/seafood, tumisan sayur,
sambal, kerupuk, pudding, es krim, buah/salah buah, teh, air mineral,
soft drink';
        WHEN 'Paket UpTown 7' THEN
            SET menu_details = 'nasi, pilihan soup, pilihan masakan ayam,
pilihan masakan daging, pilihan masakan ikan/seafood, tumisan sayur,
pilihan masakan pelengkap, pilihan penyegar, sambal, kerupuk, pudding,
es krim, buah, teh, air mineral, soft drink' ;
        ELSE
            SET menu_details = 'Paket Tidak Ditemukan';
        END CASE;

        SELECT menu_details AS Bundling_Details, price_details;
    END //
DELIMITER ;

```

Penjelasan :

DELIMITER //

Mengganti delimiter default dari ; menjadi // untuk mendefinisikan stored procedure tanpa mengganggu penggunaan ; di dalam tubuh prosedur.

CREATE PROCEDURE detail_bundling(IN input_name VARCHAR(255))

Mendefinisikan stored procedure bernama detail_bundling dengan parameter input_name bertipe VARCHAR(255).

BEGIN ... END

Blok yang mendefinisikan tubuh dari stored procedure.

DECLARE menu_details VARCHAR(255);

Mendeklarasikan variabel menu_details untuk menyimpan rincian menu dari paket bundling.

DECLARE price_details FLOAT;

Deklarasi variabel price_details untuk menyimpan harga dari paket bundling.

Mengambil harga paket bundling dari tabel special_bundling berdasarkan nama paket yang diberikan dan menyimpannya dalam variabel price_details.

Menggunakan struktur CASE untuk menentukan rincian menu berdasarkan nama paket bundling yang diberikan.

WHEN: Menetapkan rincian menu sesuai dengan nama paket jika nama paket cocok dengan salah satu opsi yang telah ditentukan.

ELSE: Jika nama paket tidak cocok dengan salah satu opsi yang telah ditentukan, menetapkan rincian menu sebagai 'Paket Tidak Ditemukan'.

END CASE;

Akhir dari blok CASE.

SELECT menu_details AS Bundling_Details, price_details;

Mengambil dan menampilkan rincian menu dan harga paket bundling. menu_details diberi alias Bundling_Details untuk output.

END //

akhir dari definisi stored procedure.

Contoh *Query* memanggil *Procedure* di atas

```
CALL detail_bundling('Paket UpTown 1')
```

⚠ Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are disabled.

✓ Showing rows 0 - 0 (1 total, Query took 0.0003 seconds.)

```
CALL detail_bundling('Paket UpTown 1');
```

[Edit inline] [Edit] [Create PHP code]

☐ Show all

Number of rows: 25 ▼

Filter rows:

Extra options

Bundling_Details	price_details
nasi, soup, pilihan ayam/ikan, pilihan masakan say...	30000

☐ Show all

Number of rows: 25 ▼

Filter rows:

VIEW

Membuat *View Status Not Available* untuk *drink_menu*

```
CREATE VIEW v_drink_not_available AS
SELECT d.ID as ID, d.Name as Name, d.Price as Price, c.Name category
FROM drink_menu d
JOIN drink_categories c ON Cat_ID = c.ID
WHERE status='Not Available';
```

Penjelasan :

CREATE VIEW v_drink_not_available AS

Mendefinisikan sebuah view baru dengan nama v_drink_not_available. View adalah objek basis data yang menyimpan hasil dari sebuah query dan dapat digunakan seperti tabel.

SELECT d.ID as ID, d.Name as Name, d.Price as Price, c.Name category

Menentukan kolom-kolom yang akan ditampilkan dalam view:

d.ID akan ditampilkan sebagai ID.

d.Name akan ditampilkan sebagai Name.

d.Price akan ditampilkan sebagai Price.

c.Name dari tabel drink_categories akan ditampilkan sebagai category.

FROM drink_menu d

Menyebutkan tabel utama dari mana data akan diambil, yaitu drink_menu dengan alias d.

JOIN drink_categories c ON Cat_ID = c.ID

Melakukan operasi JOIN antara tabel drink_menu (alias d) dan tabel drink_categories (alias c) berdasarkan kesamaan Cat_ID dari drink_menu dengan ID dari drink_categories. Ini akan menggabungkan informasi kategori minuman dengan informasi minuman itu sendiri.

WHERE status='Not Available';

Menyaring hasil query untuk hanya menampilkan baris-baris di mana kolom status pada tabel drink_menu bernilai 'Not Available'.

Membuat *View Status Not Available* untuk *food_menu*

```
CREATE VIEW v_food_not_available AS
SELECT f.ID as ID, f.Name as Name, f.Price as Price, c.Name category
FROM food_menu f
JOIN food_categories c ON ID_Cat = c.ID
WHERE status='Not Available';
```

Penjelasan :

CREATE VIEW v_food_not_available AS

Mendefinisikan sebuah view baru dengan nama v_food_not_available. View ini menyimpan hasil dari query yang didefinisikan.

SELECT f.ID as ID, f.Name as Name, f.Price as Price, c.Name category

Menentukan kolom-kolom yang akan ditampilkan dalam view:

f.ID dari tabel food_menu ditampilkan sebagai ID.

f.Name dari tabel food_menu akan ditampilkan sebagai Name.

f.Price dari tabel food_menu akan ditampilkan sebagai Price.

c.Name dari tabel food_categories akan ditampilkan sebagai category.

FROM food_menu f

Menyebutkan tabel utama dari mana data akan diambil, yaitu food_menu dengan alias f.

JOIN food_categories c ON ID_Cat = c.ID

Melakukan operasi JOIN antara tabel food_menu (alias f) dan tabel food_categories (alias c) berdasarkan kesamaan ID_Cat dari food_menu dengan ID dari food_categories. Ini menggabungkan informasi kategori makanan dengan informasi makanan itu sendiri.

WHERE status='Not Available';

Menyaring hasil query untuk hanya menampilkan baris-baris di mana kolom status pada tabel food_menu bernilai 'Not Available'.

Contoh *Query* Untuk Melihat isi *View*

```
SELECT * FROM v_drink_not_available;
```

```
SELECT * FROM v_food_not_available;
```

⚠ Current selection does not contain a unique column. Grid edit, Edit, Copy and Delete features may result in undesired

















✓ Showing rows 0 - 5 (6 total, Query took 0.0003 seconds.)

```
SELECT * FROM v_drink_not_available;
```

☐ Profiling [[Edit inline](#)] [[Edit](#)] [[Explain SQL](#)] [[Create PHP code](#)] [[Refresh](#)]

☐ Show all | Number of rows: 25 | Filter rows:

Extra options

				ID	Name	Price	category
<input type="checkbox"/>	 Edit	 Copy	 Delete	DR001	Jus Jambu	10000	Healthy Drink
<input type="checkbox"/>	 Edit	 Copy	 Delete	DR008	Jus Mangga	14000	Healthy Drink
<input type="checkbox"/>	 Edit	 Copy	 Delete	DR104	Towar Vanilla Ice Cream	15000	Bread + Ice Cream
<input type="checkbox"/>	 Edit	 Copy	 Delete	DR105	Towar Coklat Ice Cream	15000	Bread + Ice Cream
<input type="checkbox"/>	 Edit	 Copy	 Delete	DR106	Towar Strawberry Ice Cream	15000	Bread + Ice Cream
<input type="checkbox"/>	 Edit	 Copy	 Delete	DR107	Towar Mix Ice Cream	15000	Bread + Ice Cream

⬆ ☐ Check all | With selected: Edit Copy Delete Export

☐ Show all | Number of rows: 25 | Filter rows: