Project Basis Data - Group 5
“Small Business Inventory and Sales System”
Basic CRUD Operations

Disusun oleh:
Aloysius Pijar Hutama Indrianto (24/534591/PA/22675)
Pison Golda Mountera (24/543770/PA/23107)
Indratanaya Budiman (24/534784/PA/22683)
KOMA

DEPARTEMEN ILMU KOMPUTER DAN ELEKTRONIKA
FAKULTAS MATEMATIKA DAN PENGETAHUAN ALAM
SEMESTER GASAL 2025/2026

YOGYAKARTA

1. Authentication

a. Login (POST /auth/login)

Langkah pertama yang perlu dilakukan adalah mencari pengguna, yaitu dengan perintah sebagai berikut: (Find by Username)

```sql
SELECT user_id, hashed_password FROM Users WHERE username = $username;
```

Contoh hasil query tersebut adalah seperti berikut:

```sql
### POST /auth/login
SELECT
    BIN_TO_UUID(user_id) AS readable_user_id,
    hashed_password
FROM
    Users
WHERE
    username = 'admin_toko';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

readable_user_id	hashed_password
6f7fdcda-c3c8-11f0-b1df-0a002700000f	$2a$12$LOKp2s4M03wVrML.0dHxO.xQ30.1.xS...

Dalam pengaplikasian dalam sistem backend-nya, request yang akan diproses akan berbentuk dalam format JSON sebagai berikut:

POST http://localhost:3000/auth/login

Docs Params Authorization Headers (8) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```json
{
    "username": "admin_toko",
    "password": "rahasiabanget"
}
```

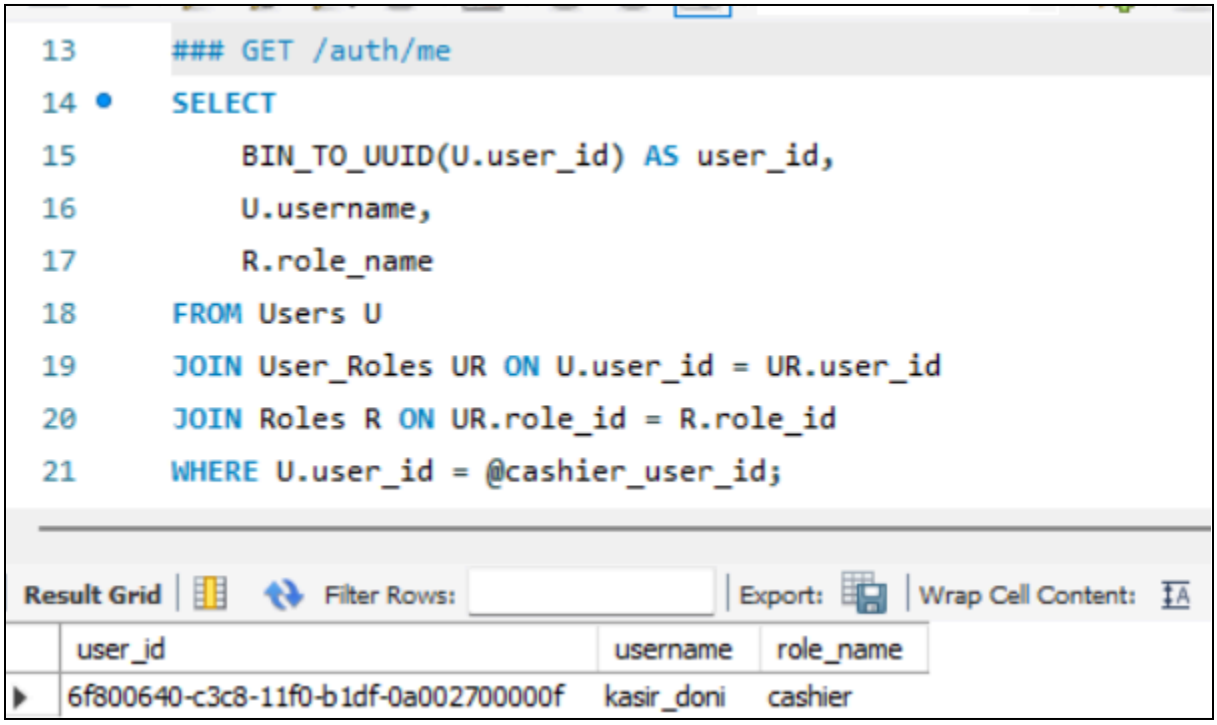
Jika server backend berhasil menemukan username-nya dan password dari baris user tersebut cocok, maka dihasilkan token yang akan digunakan oleh user (client) untuk melakukan autentikasi pada Header HTTP Request, sebagai berikut:

b. Info User (GET /auth/me)

Berikut adalah Query SQL untuk melakukan fitur ini:

```sql
SELECT U.user_id, U.username, R.role_name
FROM Users U
JOIN User_Roles UR ON U.user_id = UR.user_id
JOIN Roles R ON UR.role_id = R.role_id
WHERE U.user_id = $token_user_id;
```

Simulasi dari eksekusi Query SQL tersebut, nantinya akan menghasilkan output seperti pada gambar di bawah ini:



Hasil output tersebut akan ditampilkan dalam format JSON pada praktik implementasi backend.

2. User Management

a. (Admin) Create New User (POST /users)

Harus dijalankan dalam transaksi karena melibatkan dua tabel.

```sql
-- (Backend menghasilkan $new_user_id dan $hashed_password)
INSERT INTO Users (user_id, username, hashed_password)
VALUES ($new_user_id, $username, $hashed_password);

INSERT INTO User_Roles (user_id, role_id)
VALUES ($new_user_id, $role_id);
```

```sql
## 2. User Management
### POST /users
-- 1. Siapkan variabel untuk data pengguna baru
SET @new_user_id = UUID_TO_BIN(UUID()); -- Membuat ID biner baru yang unik
SET @new_username = 'kasir_siti';
SET @new_password_hash = '$2a$12$dummyhashforSiti.xQ30.1.xS/fM8/21G11aG14.aJj19Xa'; -- (Contoh hash)
SET @new_user_role_id = 2; -- (ID 2 = cashier dari tabel Roles)

-- 2. Mulai transaksi
START TRANSACTION;

-- 3. Masukkan data ke tabel Users
INSERT INTO Users (user_id, username, hashed_password)
VALUES (@new_user_id, @new_username, @new_password_hash);

-- 4. Masukkan data ke tabel User_Roles (menggunakan ID yang sama)
INSERT INTO User_Roles (user_id, role_id)
VALUES (@new_user_id, @new_user_role_id);

-- 5. Selesaikan transaksi
COMMIT;
```

b. (Admin) Find All Users (GET /users)

```sql
SELECT U.user_id, U.username, R.role_name
FROM Users U
JOIN User_Roles UR ON U.user_id = UR.user_id
JOIN Roles R ON UR.role_id = R.role_id;
```

Berikut adalah simulasi output dari fitur ini:

```
55      ### GET /users
56      SELECT
57          BIN_TO_UUID(U.user_id) AS user_id,
58          U.username,
59          R.role_name
60      FROM Users U
61      JOIN User_Roles UR ON U.user_id = UR.user_id
62      JOIN Roles R ON UR.role_id = R.role_id;
```

Result Grid | Filter Rows: | Export: | Wrap C

user_id	username	role_name
6f7fdcda-c3c8-11f0-b1df-0a002700000f	admin_toko	admin
0a7ef318-c3ca-11f0-b1df-0a002700000f	kasir_siti	cashier
6f800640-c3c8-11f0-b1df-0a002700000f	kasir_doni	cashier

c. <u>(Admin) Find User by ID (GET /users/{userId})</u>

SQL

```sql
SELECT U.user_id, U.username, R.role_name
FROM Users U
JOIN User_Roles UR ON U.user_id = UR.user_id
JOIN Roles R ON UR.role_id = R.role_id
WHERE U.user_id = $user_id;
```

```
64      ### GET /users/{userId}
65      SELECT
66          BIN_TO_UUID(U.user_id) AS user_id,
67          U.username,
68          R.role_name
69      FROM Users U
70      JOIN User_Roles UR ON U.user_id = UR.user_id
71      JOIN Roles R ON UR.role_id = R.role_id
72      WHERE U.user_id = @admin_user_id;
```

Result Grid | Filter Rows: | Export: | Wrap C

user_id	username	role_name
6f7fdcda-c3c8-11f0-b1df-0a002700000f	admin_toko	admin

d. (Admin) Update User (PUT /users/{userId})

SQL

```
-- (Backend akan secara kondisional menyertakan 'hashed_password' jika di-input)
UPDATE Users
SET username = $username, hashed_password = $new_hashed_password
WHERE user_id = $user_id;

UPDATE User_Roles
SET role_id = $role_id
WHERE user_id = $user_id;
```

```
### PUT /users/{userId}
-- 1. Siapkan data baru dan ID target
-- (Simulasi backend mengambil ID dari URL, misal untuk 'kasir_doni')
SET @user_to_update_id = (SELECT user_id FROM Users WHERE username = 'kasir_doni');
SET @new_username = 'doni_setiawan';
SET @new_password_hash = '$2a$12$newHashForDoni.xQ30.1.xS/fM8/21G11aG14.aJj19Xa'; -- (Hash baru)
SET @new_role_id = 1; -- (ID 1 = 'admin')

-- 2. Mulai transaksi
START TRANSACTION;

-- 3. Update tabel Users
-- (Sesuai kueri Anda, kita update username DAN password)
UPDATE Users
SET
    username = @new_username,
    hashed_password = @new_password_hash
WHERE user_id = @user_to_update_id;

-- 4. Update tabel User_Roles
UPDATE User_Roles
SET role_id = @new_role_id
WHERE user_id = @user_to_update_id;

-- 5. Selesaikan transaksi
COMMIT;
```

```
-- 6. (Opsional) Tampilkan bukti bahwa data telah berubah
SELECT
    BIN_TO_UUID(U.user_id) AS user_id,
    U.username,
    R.role_name
FROM Users U
JOIN User_Roles UR ON U.user_id = UR.user_id
JOIN Roles R ON UR.role_id = R.role_id
WHERE U.user_id = @user_to_update_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

user_id	username	role_name
6f800640-c3c8-11f0-b1df-0a002700000f	doni_setiawan	admin

e. (Admin) Delete User (DELETE /users/{userId})

Berkat ON DELETE CASCADE di tabel User_Roles, kueri ini otomatis menghapus data di kedua tabel.

```sql
DELETE FROM Users WHERE user_id = $user_id;
```

```sql
### DELETE /users/{userId}
-- 1. Tentukan ID pengguna yang akan dihapus
-- (Kita cari 'kasir_siti' yang baru kita buat)
SET @user_to_delete_id = (SELECT user_id FROM Users WHERE username = 'kasir_siti');

-- 2. Tampilkan data SEBELUM dihapus (Opsional)
SELECT 'SEBELUM HAPUS' AS 'Status', BIN_TO_UUID(user_id) AS user_id, username FROM Users WHERE user_id = @user_to_delete_id;
SELECT 'SEBELUM HAPUS' AS 'Status', BIN_TO_UUID(user_id) AS user_id, role_id FROM User_Roles WHERE user_id = @user_to_delete_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Status	user_id	role_id
SEBELUM HAPUS	0a7ef318-c3ca-11f0-b1df-0a002700000f	2

```sql
-- 3. Jalankan kueri DELETE (Mengganti $user_id dengan variabel)
DELETE FROM Users
WHERE user_id = @user_to_delete_id;

-- 4. Tampilkan data SETELAH dihapus (Bukti)
-- (Kedua kueri ini seharusnya mengembalikan 'Empty set')
SELECT 'SETELAH HAPUS' AS 'Status', BIN_TO_UUID(user_id) AS user_id, username FROM Users WHERE user_id = @user_to_delete_id;
SELECT 'SETELAH HAPUS' AS 'Status', BIN_TO_UUID(user_id) AS user_id, role_id FROM User_Roles WHERE user_id = @user_to_delete_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Status	user_id	role_id

f. (Admin) Find All Role (GET /roles)

```sql
SELECT role_id, role_name FROM Roles;
```

```sql
### GET /roles
SELECT role_id, role_name FROM Roles;
```

Result Grid | Filter Rows: | Edit:

role_id	role_name
1	admin
2	cashier
NULL	NULL

3. Categories

a. (Admin) Create New Category (POST /categories)

```sql
-- (Backend menghasilkan $new_category_id)
INSERT INTO Categories (category_id, category_name)
VALUES ($new_category_id, $category_name);
```

```sql
## 3. Categories
### POST /categories
-- 1. Siapkan variabel untuk data kategori baru
SET @new_category_id = UUID_TO_BIN(UUID()); -- Membuat ID biner baru yang unik
SET @new_category_name = 'Peralatan Rumah Tangga';

-- 2. Jalankan kueri INSERT (Mengganti placeholder)
INSERT INTO Categories (category_id, category_name)
VALUES (@new_category_id, @new_category_name);

-- 3. (Opsional) Tampilkan bukti bahwa kategori baru telah dibuat
SELECT
    BIN_TO_UUID(category_id) AS category_id,
    category_name
FROM Categories
WHERE category_name = 'Peralatan Rumah Tangga';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

category_id	category_name
fc4e165f-c3cb-11f0-b1df-0a002700000f	Peralatan Rumah Tangga

b. Find All Categories (GET /categories)

```sql
SELECT category_id, category_name FROM Categories;
```

```sql
### GET /categories
SELECT
    BIN_TO_UUID(category_id) AS category_id,
    category_name
FROM Categories;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

category_id	category_name
6f81ca03-c3c8-11f0-b1df-0a002700000f	Makanan Ringan
6f81a9d0-c3c8-11f0-b1df-0a002700000f	Minuman
fc4e165f-c3cb-11f0-b1df-0a002700000f	Peralatan Rumah Tangga

c. (Admin) Update a Category (PUT /categories/{categoryId})

SQL

```sql
UPDATE Categories
SET category_name = $category_name
WHERE category_id = $category_id;
```

```sql
### PUT /categories/{categoryId}
-- 1. Siapkan data baru dan ID target
-- (Simulasi backend mengambil ID dari URL, misal untuk 'Minuman')
SET @category_to_update_id = (SELECT category_id FROM Categories WHERE category_name = 'Minuman');
SET @new_category_name = 'Minuman Dingin';

-- 2. Tampilkan data SEBELUM diubah (Opsional)
SELECT 'SEBELUM UPDATE' AS 'Status', BIN_TO_UUID(category_id), category_name
FROM Categories
WHERE category_id = @category_to_update_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Status	BIN_TO_UUID(category_id)	category_name
SEBELUM UPDATE	6f81a9d0-c3c8-11f0-b1df-0a002700000f	Minuman

```sql
-- 3. Jalankan kueri UPDATE (Mengganti placeholder)
UPDATE Categories
SET category_name = @new_category_name
WHERE category_id = @category_to_update_id;

-- 4. Tampilkan data SETELAH diubah (Bukti)
SELECT 'SETELAH UPDATE' AS 'Status', BIN_TO_UUID(category_id), category_name
FROM Categories
WHERE category_id = @category_to_update_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Status	BIN_TO_UUID(category_id)	category_name
SEBELUM UPDATE	6f81a9d0-c3c8-11f0-b1df-0a002700000f	Minuman

d. (Admin) Delete Category (DELETE /categories/{categoryId})

SQL

```sql
DELETE FROM Categories WHERE category_id = $category_id;
```

```sql
### DELETE /categories/{categoryId}
-- 1. Tentukan ID kategori yang akan dihapus
-- (Kita cari 'Peralatan Rumah Tangga' yang baru kita buat)
SET @category_to_delete_id = (SELECT category_id FROM Categories WHERE category_name = 'Peralatan Rumah Tangga');

-- 2. Tampilkan data SEBELUM dihapus (Opsional)
SELECT 'SEBELUM HAPUS' AS 'Status', BIN_TO_UUID(category_id), category_name
FROM Categories
WHERE category_id = @category_to_delete_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Status	BIN_TO_UUID(category_id)	category_name
SEBELUM HAPUS	fc4e165f-c3cb-11f0-b1df-0a002700000f	Peralatan Rumah Tangga

```sql
-- 2. Tampilkan data SEBELUM dihapus (Opsional)
SELECT 'SEBELUM HAPUS' AS 'Status', BIN_TO_UUID(category_id), category_name
FROM Categories
WHERE category_id = @category_to_delete_id;

-- 3. Jalankan kueri DELETE (Mengganti $category_id dengan variabel)
DELETE FROM Categories
WHERE category_id = @category_to_delete_id;

-- 4. Tampilkan data SETELAH dihapus (Bukti)
-- (Kueri ini seharusnya mengembalikan 'Empty set')
SELECT 'SETELAH HAPUS' AS 'Status', BIN_TO_UUID(category_id), category_name
FROM Categories
WHERE category_id = @category_to_delete_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Status	BIN_TO_UUID(category_id)	category_name

4. Suppliers

a. POST /suppliers
(Admin) Buat Supplier Baru

```sql
-- (Backend menghasilkan $new_supplier_id)
INSERT INTO Suppliers (supplier_id, supplier_name, phone_number, email)
VALUES ($new_supplier_id, $supplier_name, $phone_number, $email);
```

```sql
### POST /suppliers
-- 1. Siapkan variabel untuk data supplier baru
SET @new_supplier_id = UUID_TO_BIN(UUID()); -- Membuat ID biner baru yang unik
SET @new_supplier_name = 'PT Indofood Sukses Makmur';
SET @new_phone = '021-5795-8822';
SET @new_email = 'contact@indofood.com';

-- 2. Jalankan kueri INSERT (Mengganti placeholder)
INSERT INTO Suppliers (supplier_id, supplier_name, phone_number, email)
VALUES (@new_supplier_id, @new_supplier_name, @new_phone, @new_email);

-- 3. (Opsional) Tampilkan bukti bahwa supplier baru telah dibuat
SELECT
    BIN_TO_UUID(supplier_id) AS supplier_id,
    supplier_name,
    phone_number,
    email
FROM Suppliers
WHERE supplier_name = 'PT Indofood Sukses Makmur';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

supplier_id	supplier_name	phone_number	email
f7242c95-c3cc-11f0-b1df-0a002700000f	PT Indofood Sukses Makmur	021-5795-8822	contact@indofood.com

b. GET /suppliers
Dapatkan Semua Supplier

```sql
SELECT supplier_id, supplier_name, phone_number, email FROM Suppliers;
```

```sql
SELECT
    BIN_TO_UUID(supplier_id) AS supplier_id,
    supplier_name,
    phone_number,
    email
FROM Suppliers;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

supplier_id	supplier_name	phone_number	email
6f828cea-c3c8-11f0-b1df-0a002700000f	PT Unilever Indonesia	021-555-1234	sales@unilever.com
6f82b19f-c3c8-11f0-b1df-0a002700000f	PT Mayora Indah Tbk	021-888-9012	contact@mayora.co.id
f7242c95-c3cc-11f0-b1df-0a002700000f	PT Indofood Sukses Makmur	021-5795-8822	contact@indofood.com

c. PUT /suppliers/{supplierId}
(Admin) Update Supplier

SQL

```sql
UPDATE Suppliers
SET
  supplier_name = $supplier_name,
  phone_number = $phone_number,
  email = $email
WHERE supplier_id = $supplier_id;
```

```sql
### PUT /suppliers/{supplierId}
-- 1. Siapkan data baru dan ID target
-- (Simulasi backend mengambil ID dari URL, misal untuk 'PT Mayora Indah Tbk')
SET @supplier_to_update_id = (SELECT supplier_id FROM Suppliers WHERE supplier_name = 'PT Mayora Indah Tbk');
SET @new_supplier_name = 'PT Mayora Global';
SET @new_phone = '021-999-1111';
SET @new_email = 'info@mayora-global.com';

-- 2. Tampilkan data SEBELUM diubah (Opsional)
SELECT 'SEBELUM UPDATE' AS 'Status', BIN_TO_UUID(supplier_id), supplier_name, phone_number, email
FROM Suppliers
WHERE supplier_id = @supplier_to_update_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Status	BIN_TO_UUID(supplier_id)	supplier_name	phone_number	email
SEBELUM UPDATE	6f82b19f-c3c8-11f0-b1df-0a002700000f	PT Mayora Indah Tbk	021-888-9012	contact@mayora.co.id

```sql
-- 3. Jalankan kueri UPDATE (Mengganti placeholder)
UPDATE Suppliers
SET
    supplier_name = @new_supplier_name,
    phone_number = @new_phone,
    email = @new_email
WHERE supplier_id = @supplier_to_update_id;

-- 4. Tampilkan data SETELAH diubah (Bukti)
SELECT 'SETELAH UPDATE' AS 'Status', BIN_TO_UUID(supplier_id), supplier_name, phone_number, email
FROM Suppliers
WHERE supplier_id = @supplier_to_update_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Status	BIN_TO_UUID(supplier_id)	supplier_name	phone_number	email
SETELAH UPDATE	6f82b19f-c3c8-11f0-b1df-0a002700000f	PT Mayora Global	021-999-1111	info@mayora-global.com

d. DELETE /suppliers/{supplierId}
(Admin) Hapus Supplier

SQL

```sql
DELETE FROM Suppliers WHERE supplier_id = $supplier_id;
```

```sql
### DELETE /suppliers/{supplierId}
-- 1. Tentukan ID supplier yang akan dihapus
-- (Kita cari 'PT Indofood Sukses Makmur' yang baru kita buat)
SET @supplier_to_delete_id = (SELECT supplier_id FROM Suppliers WHERE supplier_name = 'PT Indofood Sukses Makmur');

-- 2. Tampilkan data SEBELUM dihapus (Opsional)
SELECT 'SEBELUM HAPUS' AS 'Status', BIN_TO_UUID(supplier_id), supplier_name
FROM Suppliers
WHERE supplier_id = @supplier_to_delete_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Status	BIN_TO_UUID(supplier_id)	supplier_name
SEBELUM HAPUS	f7242c95-c3cc-11f0-b1df-0a002700000f	PT Indofood Sukses Makmur

```sql
-- 3. Jalankan kueri DELETE (Mengganti $supplier_id dengan variabel)
DELETE FROM Suppliers
WHERE supplier_id = @supplier_to_delete_id;

-- 4. Tampilkan data SETELAH dihapus (Bukti)
-- (Kueri ini seharusnya mengembalikan 'Empty set')
SELECT 'SETELAH HAPUS' AS 'Status', BIN_TO_UUID(supplier_id), supplier_name
FROM Suppliers
WHERE supplier_id = @supplier_to_delete_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Status	BIN_TO_UUID(supplier_id)	supplier_name

5. Products

a. POST /products
(Admin) Tambah Produk Baru

```sql
-- (Backend menghasilkan $new_product_id)
INSERT INTO Products (
  product_id, product_name, purchase_price, selling_price,
  stock_quantity, category_id, supplier_id
)
VALUES (
  $new_product_id, $product_name, $purchase_price, $selling_price,
  $stock_quantity, $category_id, $supplier_id
);
```

```sql
## 5. Products
### POST /products
-- 1. Siapkan variabel untuk data produk baru
SET @new_product_id = UUID_TO_BIN(UUID()); -- Membuat ID biner baru yang unik
SET @new_product_name = 'Indomie Goreng';
SET @new_purchase_price = 2800.00;
SET @new_selling_price = 3500.00;
SET @new_stock_quantity = 500;

-- 2. Dapatkan ID Foreign Key yang ada
-- (Menggunakan variabel dari mock data)
SET @category_id_fk = @cat_makanan_id; -- (ID 'Makanan Ringan')
SET @supplier_id_fk = (SELECT supplier_id FROM Suppliers WHERE supplier_name = 'PT Mayora Global'); -- (ID 'PT Mayora Global')

-- 3. Jalankan kueri INSERT (Mengganti placeholder)
INSERT INTO Products (
  product_id, product_name, purchase_price, selling_price,
  stock_quantity, category_id, supplier_id
)
VALUES (
  @new_product_id, @new_product_name, @new_purchase_price, @new_selling_price,
  @new_stock_quantity, @category_id_fk, @supplier_id_fk
);
```

```sql
-- 4. (Opsional) Tampilkan bukti bahwa produk baru telah dibuat
SELECT
    BIN_TO_UUID(product_id) AS product_id,
    product_name,
    stock_quantity,
    selling_price
FROM Products
WHERE product_name = 'Indomie Goreng';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

product_id	product_name	stock_quantity	selling_price
e1eadba5-c3cd-11f0-b1df-0a002700000f	Indomie Goreng	500	3500.00

b. GET /products
Dapatkan Semua Produk

```sql
SELECT
  product_id, product_name, purchase_price, selling_price,
  stock_quantity, category_id, supplier_id
FROM Products;
```

```
304     ### GET /products
305 •   SELECT
306         BIN_TO_UUID(product_id) AS product_id,
307         product_name,
308         purchase_price,
309         selling_price,
310         stock_quantity,
311         BIN_TO_UUID(category_id) AS category_id,
312         BIN_TO_UUID(supplier_id) AS supplier_id
313     FROM Products;
314
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

product_id	product_name	purchase_price	selling_price	stock_quantity	category_id	supplier_id
6f837dea-c3c8-11f0-b1df-0a002700000f	Teh Botol Kotak 250ml	2500.00	3500.00	98	6f81a9d0-c3c8-11f0-b1df-0a002700000f	6f828cea-c3c8-11f0-b1df-0a002700000f
6f839c45-c3c8-11f0-b1df-0a002700000f	Roma Malkist Abon	8000.00	10000.00	49	6f81ca03-c3c8-11f0-b1df-0a002700000f	6f82b19f-c3c8-11f0-b1df-0a002700000f
e1eadba5-c3cd-11f0-b1df-0a002700000f	Indomie Goreng	2800.00	3500.00	500	6f81ca03-c3c8-11f0-b1df-0a002700000f	6f82b19f-c3c8-11f0-b1df-0a002700000f

c. GET /products/{productId}
Dapatkan Produk by ID

```sql
SELECT
  product_id, product_name, purchase_price, selling_price,
  stock_quantity, category_id, supplier_id
FROM Products
WHERE product_id = $product_id;
```

```
315     ### GET /products/{productId}
316 •   SELECT
317         BIN_TO_UUID(product_id) AS product_id,
318         product_name,
319         purchase_price,
320         selling_price,
321         stock_quantity,
322         BIN_TO_UUID(category_id) AS category_id,
323         BIN_TO_UUID(supplier_id) AS supplier_id
324     FROM Products
325     WHERE product_id = @prod_teh_id;
326
327     ### PUT /products/{productId}
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

product_id	product_name	purchase_price	selling_price	stock_quantity	category_id	supplier_id
6f837dea-c3c8-11f0-b1df-0a002700000f	Teh Botol Kotak 250ml	2500.00	3500.00	98	6f81a9d0-c3c8-11f0-b1df-0a002700000f	6f828cea-c3c8-11f0-b1df-0a002700000f

d. PUT /products/{productId}
(Admin) Update Produk

```sql
UPDATE Products
SET
  product_name = $product_name,
  purchase_price = $purchase_price,
  selling_price = $selling_price,
  category_id = $category_id,
  supplier_id = $supplier_id
WHERE product_id = $product_id;
```

```sql
### PUT /products/{productId}
-- 1. Siapkan data baru dan ID target
-- (Simulasi backend mengambil ID dari URL, misal untuk 'Teh Botol Kotak 250ml')
SET @product_to_update_id = (SELECT product_id FROM Products WHERE product_name = 'Teh Botol Kotak 250ml');
SET @new_product_name = 'Teh Kotak Sosro 250ml';
SET @new_purchase_price = 2600.00;
SET @new_selling_price = 3800.00;
-- (Kita akan ubah supplier-nya ke 'PT Mayora Global' sebagai contoh)
SET @new_category_id_fk = @cat_minuman_id;
SET @new_supplier_id_fk = (SELECT supplier_id FROM Suppliers WHERE supplier_name = 'PT Mayora Global');


-- 2. Tampilkan data SEBELUM diubah (Opsional)
SELECT 'SEBELUM UPDATE' AS 'Status', BIN_TO_UUID(product_id), product_name, selling_price, BIN_TO_UUID(supplier_id)
FROM Products
WHERE product_id = @product_to_update_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Status	BIN_TO_UUID(product_id)	product_name	selling_price	BIN_TO_UUID(supplier_id)
SEBELUM UPDATE	6f837dea-c3c8-11f0-b1df-0a002700000f	Teh Botol Kotak 250ml	3500.00	6f828cea-c3c8-11f0-b1df-0a002700000f

```sql
-- 3. Jalankan kueri UPDATE (Mengganti placeholder)
UPDATE Products
SET
    product_name = @new_product_name,
    purchase_price = @new_purchase_price,
    selling_price = @new_selling_price,
    category_id = @new_category_id_fk,
    supplier_id = @new_supplier_id_fk
WHERE product_id = @product_to_update_id;

-- 4. Tampilkan data SETELAH diubah (Bukti)
SELECT 'SETELAH UPDATE' AS 'Status', BIN_TO_UUID(product_id), product_name, selling_price, BIN_TO_UUID(supplier_id)
FROM Products
WHERE product_id = @product_to_update_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Status	BIN_TO_UUID(product_id)	product_name	selling_price	BIN_TO_UUID(supplier_id)
SETELAH UPDATE	6f837dea-c3c8-11f0-b1df-0a002700000f	Teh Kotak Sosro 250ml	3800.00	6f82b19f-c3c8-11f0-b1df-0a002700000f

e. DELETE /products/{productId}
(Admin) Hapus Produk

```sql
DELETE FROM Products WHERE product_id = $product_id;
```

```sql
### DELETE /products/{productId}
-- 1. Tentukan ID produk yang akan dihapus
-- (Kita cari 'Indomie Goreng' yang baru kita buat)
SET @product_to_delete_id = (SELECT product_id FROM Products WHERE product_name = 'Indomie Goreng');

-- 2. Tampilkan data SEBELUM dihapus (Opsional)
SELECT 'SEBELUM HAPUS' AS 'Status', BIN_TO_UUID(product_id), product_name
FROM Products
WHERE product_id = @product_to_delete_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Status	BIN_TO_UUID(product_id)	product_name
SEBELUM HAPUS	e1eadba5-c3cd-11f0-b1df-0a002700000f	Indomie Goreng

```sql
-- 3. Jalankan kueri DELETE (Mengganti $product_id dengan variabel)
DELETE FROM Products
WHERE product_id = @product_to_delete_id;

-- 4. Tampilkan data SETELAH dihapus (Bukti)
-- (Kueri ini seharusnya mengembalikan 'Empty set')
SELECT 'SETELAH HAPUS' AS 'Status', BIN_TO_UUID(product_id), product_name
FROM Products
WHERE product_id = @product_to_delete_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Status	BIN_TO_UUID(product_id)	product_name

6. Inventory

a. POST /inventory/adjust

(Admin) Penyesuaian Stok Manual

Ini adalah operasi UPDATE untuk menyesuaikan stok, di mana $quantity bisa positif (menambah) atau negatif (mengurangi).

```sql
UPDATE Products
SET stock_quantity = stock_quantity + $quantity
WHERE product_id = $product_id;
```

```sql
## 6. Inventory
### POST /inventory/adjust
-- --- SKENARIO 1: MENAMBAH STOK (RESTOCK) ---

-- 1. Siapkan variabel
-- (Kita akan restock 'Roma Malkist Keju')
SET @product_to_adjust_id = (SELECT product_id FROM Products WHERE product_name = 'Roma Malkist Abon');
SET @quantity_to_add = 100;

-- 2. Tampilkan stok SEBELUM ditambah
SELECT 'SEBELUM RESTOCK' AS 'Status', product_name, stock_quantity
FROM Products
WHERE product_id = @product_to_adjust_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Status	product_name	stock_quantity
SEBELUM RESTOCK	Roma Malkist Abon	49

```sql
-- 3. Jalankan kueri UPDATE (Mengganti placeholder)
UPDATE Products
SET stock_quantity = stock_quantity + @quantity_to_add
WHERE product_id = @product_to_adjust_id;

-- 4. Tampilkan stok SETELAH ditambah (Bukti)
SELECT 'SETELAH RESTOCK' AS 'Status', product_name, stock_quantity
FROM Products
WHERE product_id = @product_to_adjust_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Status	product_name	stock_quantity
SETELAH RESTOCK	Roma Malkist Abon	149

7. Transactions

a. POST /transactions

(Kasir) Buat Transaksi Baru

(Backend: Cek Stok) Untuk setiap item di keranjang, backend harus menjalankan ini terlebih dahulu untuk mengunci baris dan mendapatkan harga serta stok:

```sql
SELECT selling_price, stock_quantity
FROM Products
WHERE product_id = $item_product_id
FOR UPDATE;
```

(Jika stok cukup untuk semua item), jalankan kueri inti:

```sql
-- (Backend menghasilkan $new_transaction_id dan $token_user_id dari token)
INSERT INTO Transactions (transaction_id, user_id)
VALUES ($new_transaction_id, $token_user_id);

-- (Backend me-looping keranjang untuk membuat kueri ini)
INSERT INTO Transaction_Details (detail_id, transaction_id, product_id,
quantity, price)
VALUES
  ($detail_id_1, $new_transaction_id, $item_1_id, $item_1_qty, $item_1_price),
  ($detail_id_2, $new_transaction_id, $item_2_id, $item_2_qty, $item_2_price);
  -- (...dan seterusnya)

-- (Backend me-looping keranjang lagi untuk mengurangi stok)
UPDATE Products SET stock_quantity = stock_quantity - $item_1_qty WHERE
product_id = $item_1_id;
UPDATE Products SET stock_quantity = stock_quantity - $item_2_qty WHERE
product_id = $item_2_id;
-- (...dan seterusnya)
```

```
## 7. Transactions
### POST /transactions
-- ================================================================
-- SKENARIO: TRANSAKSI PENJUALAN BARU
-- ================================================================

-- 1. Siapkan variabel (simulasi backend)
--    - Kasir yang sedang login
--    - ID untuk transaksi baru
--    - Item-item di keranjang
-- ----------------------------------------------------------------
SET @kasir_id = (SELECT user_id FROM Users WHERE username = 'doni_setiawan');
SET @new_trans_id = UUID_TO_BIN(UUID()); -- ID unik untuk transaksi ini

-- Item 1: 3x Teh Kotak
SET @item1_id = (SELECT product_id FROM Products WHERE product_name = 'Teh Kotak Sosro 250ml');
SET @item1_qty = 3;

-- Item 2: 2x Roma Malkist Abon
SET @item2_id = (SELECT product_id FROM Products WHERE product_name = 'Roma Malkist Abon');
SET @item2_qty = 2;
```

```
-- 2. Mulai Transaksi
-- ----------------------------------------------------------------
START TRANSACTION;

-- 3. (Backend Cek Stok & Harga)
--    Kita 'SELECT' harga dan mengunci baris produk
--    (Simulasi dari 'SELECT ... FOR UPDATE')
-- ----------------------------------------------------------------
SELECT selling_price INTO @item1_price FROM Products WHERE product_id = @item1_id FOR UPDATE;
SELECT selling_price INTO @item2_price FROM Products WHERE product_id = @item2_id FOR UPDATE;

-- (Di backend, di sinilah Anda akan memeriksa apakah stoknya cukup.
--  Kita akan asumsikan stoknya cukup untuk demo ini.)

-- 4. (Kueri Inti 1) Masukkan header transaksi
--    (Mengganti $new_transaction_id dan $token_user_id)
-- ----------------------------------------------------------------
INSERT INTO Transactions (transaction_id, user_id)
VALUES (@new_trans_id, @kasir_id);

-- 5. (Kueri Inti 2) Masukkan detail keranjang
--    (Mengganti $detail_id_X, $item_X_id, dll.)
-- ----------------------------------------------------------------
INSERT INTO Transaction_Details (detail_id, transaction_id, product_id, quantity, price)
VALUES
    (UUID_TO_BIN(UUID()), @new_trans_id, @item1_id, @item1_qty, @item1_price),
    (UUID_TO_BIN(UUID()), @new_trans_id, @item2_id, @item2_qty, @item2_price);

-- 6. (Kueri Inti 3) Update (kurangi) stok produk
--    (Mengganti $item_X_qty dan $item_X_id)
-- ----------------------------------------------------------------
UPDATE Products SET stock_quantity = stock_quantity - @item1_qty WHERE product_id = @item1_id;
UPDATE Products SET stock_quantity = stock_quantity - @item2_qty WHERE product_id = @item2_id;

-- 7. Selesaikan Transaksi
-- ----------------------------------------------------------------
COMMIT;
```

```
478     -- ============================================================
479     -- 9. SELESAI - Tampilkan hasil untuk verifikasi
480     -- ============================================================
481
482     -- Tampilkan detail transaksi yang baru saja terjadi
483 ●   SELECT
484         BIN_TO_UUID(T.transaction_id) AS 'ID Transaksi',
485         U.username AS 'Kasir',
486         P.product_name AS 'Produk',
487         TD.quantity AS 'Jumlah',
488         TD.price AS 'Harga Satuan'
489     FROM Transactions T
490     JOIN Users U ON T.user_id = U.user_id
491     JOIN Transaction_Details TD ON T.transaction_id = TD.transaction_id
492     JOIN Products P ON TD.product_id = P.product_id
493     WHERE T.transaction_id = @new_trans_id;
494
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ID Transaksi	Kasir	Produk	Jumlah	Harga Satuan
ba779fcc-c3d0-11f0-b1df-0a002700000f	doni_setiawan	Teh Kotak Sosro 250ml	3	3800.00
ba779fcc-c3d0-11f0-b1df-0a002700000f	doni_setiawan	Roma Malkist Abon	2	10000.00

b. GET /transactions

Dapatkan Riwayat Transaksi

Kueri ini bergantung pada role pengguna (diambil dari $token_role dan $token_user_id).

Jika role = Admin:

```sql
SELECT transaction_id, transaction_time, user_id
FROM Transactions
ORDER BY transaction_time DESC;
```

```
499     ### GET /transasctions
500     -- (Simulasi Admin: Melihat SEMUA transaksi)
501 ●   SELECT
502         BIN_TO_UUID(T.transaction_id) AS transaction_id,
503         T.transaction_time,
504         BIN_TO_UUID(T.user_id) AS user_id,
505         U.username AS kasir_username -- (Bonus: Lebih mudah dibaca)
506     FROM Transactions T
507     JOIN Users U ON T.user_id = U.user_id -- (Bonus: Join ke Users)
508     ORDER BY T.transaction_time DESC;
509
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

transaction_id	transaction_time	user_id	kasir_username
ba779fcc-c3d0-11f0-b1df-0a002700000f	2025-11-17 23:16:38	6f800640-c3c8-11f0-b1df-0a002700000f	doni_setiawan
6f846e29-c3c8-11f0-b1df-0a002700000f	2025-11-17 10:00:00	6f800640-c3c8-11f0-b1df-0a002700000f	doni_setiawan

Jika role = Kasir:

```sql
SELECT transaction_id, transaction_time, user_id
FROM Transactions
WHERE user_id = $token_user_id
ORDER BY transaction_time DESC;
```

```sql
-- (Simulasi Kasir: Hanya melihat transaksi 'kasir_doni')
-- (Mengganti $token_user_id dengan @cashier_user_id)
SELECT
    BIN_TO_UUID(transaction_id) AS transaction_id,
    transaction_time,
    BIN_TO_UUID(user_id) AS user_id
FROM Transactions
WHERE user_id = @cashier_user_id -- <-- Menggunakan ID 'kasir_doni'
ORDER BY transaction_time DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

transaction_id	transaction_time	user_id
ba779fcc-c3d0-11f0-b1df-0a002700000f	2025-11-17 23:16:38	6f800640-c3c8-11f0-b1df-0a002700000f
6f846e29-c3c8-11f0-b1df-0a002700000f	2025-11-17 10:00:00	6f800640-c3c8-11f0-b1df-0a002700000f

c. GET /transactions/{transactionId}
Dapatkan Detail Transaksi by ID

```sql
SELECT
  T.transaction_id,
  T.transaction_time,
  T.user_id,
  TD.detail_id,
  TD.product_id,
  TD.quantity,
  TD.price
FROM Transactions T
JOIN Transaction_Details TD ON T.transaction_id = TD.transaction_id
WHERE T.transaction_id = $transaction_id;
```

```sql
### GET /transactions/{transactionId}
SELECT
    BIN_TO_UUID(T.transaction_id) AS transaction_id,
    T.transaction_time,
    BIN_TO_UUID(T.user_id) AS user_id,
    U.username AS kasir_username, -- (Bonus: Tampilkan nama kasir)
    BIN_TO_UUID(TD.detail_id) AS detail_id,
    BIN_TO_UUID(TD.product_id) AS product_id,
    P.product_name, -- (Bonus: Tampilkan nama produk)
    TD.quantity,
    TD.price,
    (TD.quantity * TD.price) AS subtotal -- (Bonus: Hitung subtotal)
FROM Transactions T
JOIN Transaction_Details TD ON T.transaction_id = TD.transaction_id
JOIN Users U ON T.user_id = U.user_id -- (Bonus Join)
JOIN Products P ON TD.product_id = P.product_id -- (Bonus Join)
WHERE T.transaction_id = @trans_1_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

transaction_id	transaction_time	user_id	kasir_username	detail_id	product_id	product_name	quantity	price	subtotal
6f846e29-c3c8-11f0-b1df-0a002700000f	2025-11-17 10:00:00	6f800640-c3c8-11f0-b1df-0a002700000f	doni_setiawan	ace9bb55-c3c8-11f0-b1df-0a002700000f	6f837dea-c3c8-11f0-b1df-0a002700000f	Teh Kotak Sosro 250ml	2	3500.00	7000.00
6f846e29-c3c8-11f0-b1df-0a002700000f	2025-11-17 10:00:00	6f800640-c3c8-11f0-b1df-0a002700000f	doni_setiawan	ace9bf2c-c3c8-11f0-b1df-0a002700000f	6f839c45-c3c8-11f0-b1df-0a002700000f	Roma Malkist Abon	1	10000.00	10000.00