XD HOBBY SHOP

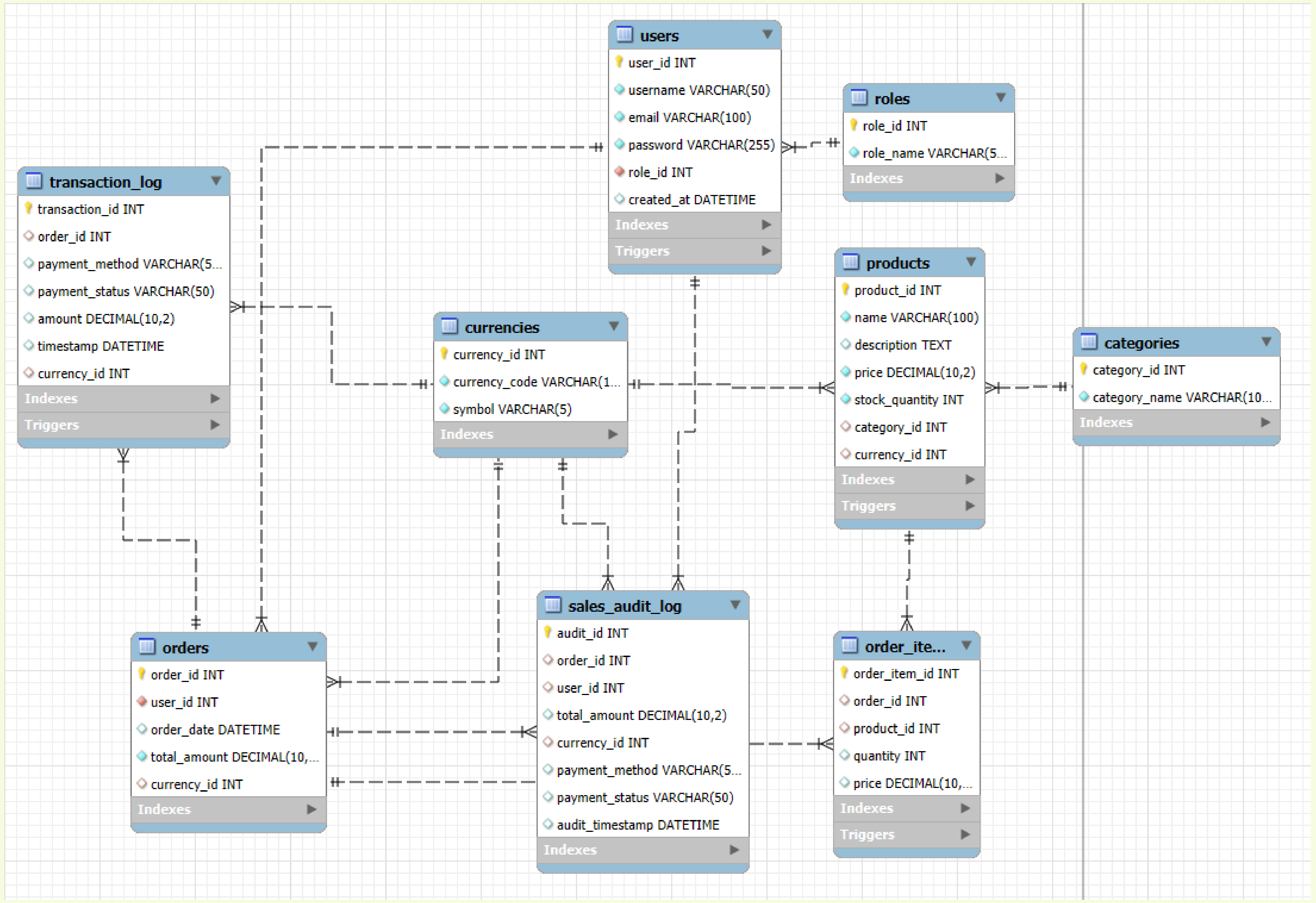
WHAT IS XD HOBBY SHOP?

XD Hobby Shop is an online retail platform dedicated to bringing Filipino Pokémon Trading Card Game (TCG) enthusiasts and collectors a wide variety of products. From singles, packs, and sealed boxes to accessories, we offer an extensive selection of Pokémon TCG items. Beyond Pokémon, we also cater to fans of other popular trading cards, including One Piece, NBA, MLB, and various sports cards.

Our mission is to be the go-to destination for collectors, competitive players, and hobbyists by providing both local and international card options at competitive prices.

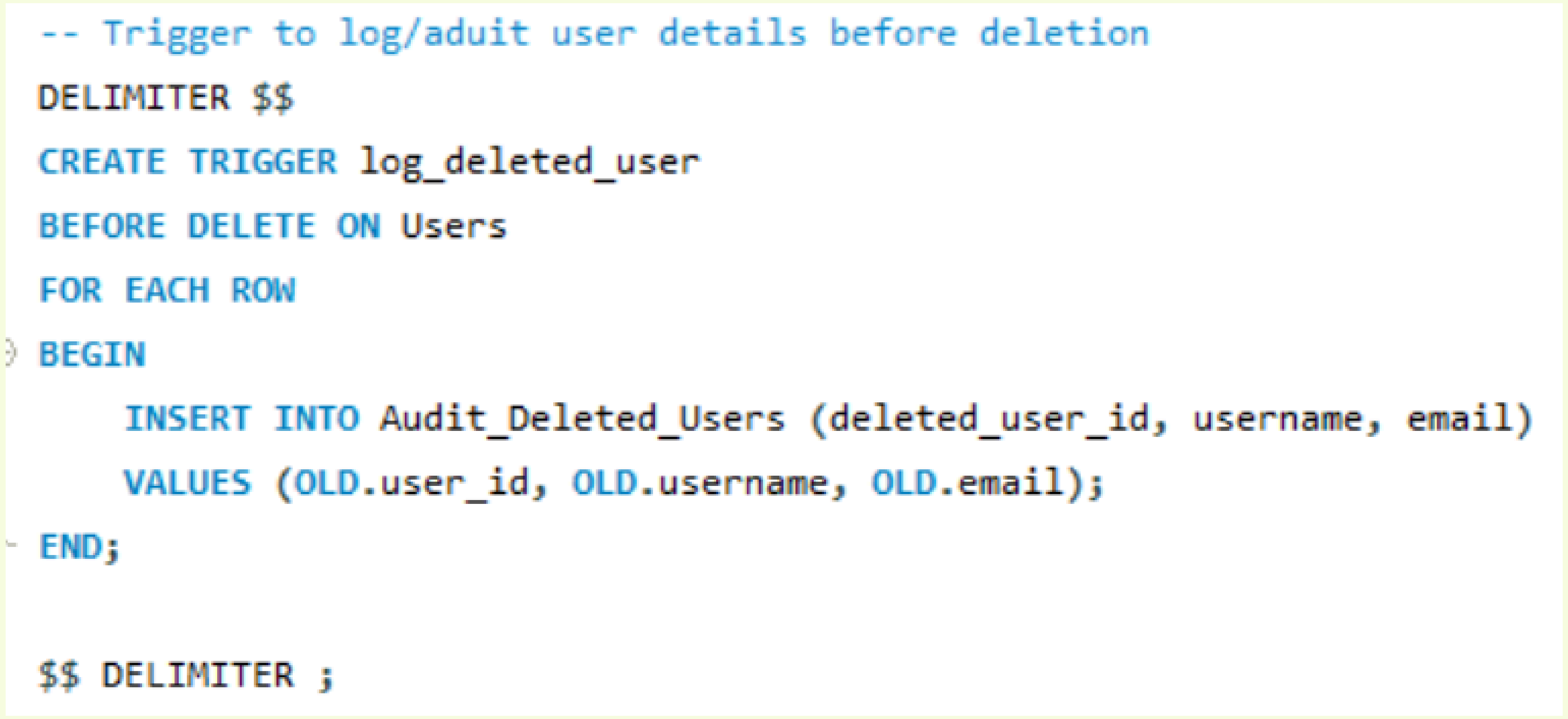
ERD DIAGRAM

ERD DIAGRAM


users
user_id INT
username VARCHAR(50)
email VARCHAR(100)
password VARCHAR(255)
role_id INT
created_at DATETIME
Indexes
Triggers
roles
role_id INT
role_name VARCHAR(5...
Indexes
transaction_log
transaction_id INT
order_id INT
payment_method VARCHAR(5...
payment_status VARCHAR(50)
amount DECIMAL(10,2)
timestamp DATETIME
currency_id INT
Indexes
Triggers
currencies
currency_id INT
currency_code VARCHAR(1...
symbol VARCHAR(5)
Indexes
products
product_id INT
name VARCHAR(100)
description TEXT
price DECIMAL(10,2)
stock_quantity INT
category_id INT
currency_id INT
Indexes
Triggers
categories
category_id INT
category_name VARCHAR(10...
Indexes
orders
order_id INT
user_id INT
order_date DATETIME
total_amount DECIMAL(10,...
currency_id INT
Indexes
sales_audit_log
audit_id INT
order_id INT
user_id INT
total_amount DECIMAL(10,2)
currency_id INT
payment_method VARCHAR(5...
payment_status VARCHAR(50)
audit_timestamp DATETIME
Indexes
order_ite...
order_item_id INT
order_id INT
product_id INT
quantity INT
price DECIMAL(10,...
Indexes
Triggers

PROCEDURES & TRIGGERS

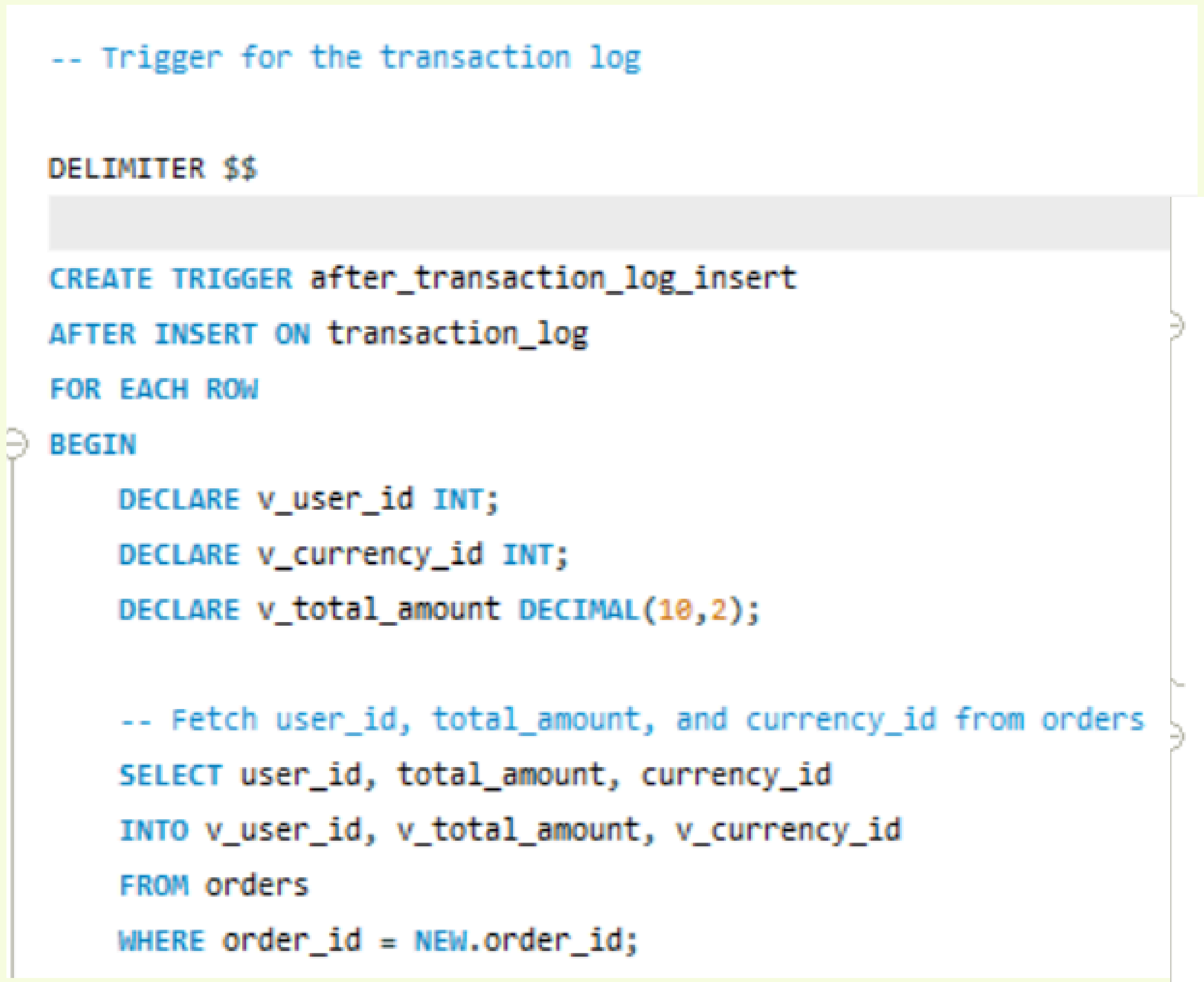
PROCEDURES & TRIGGERS



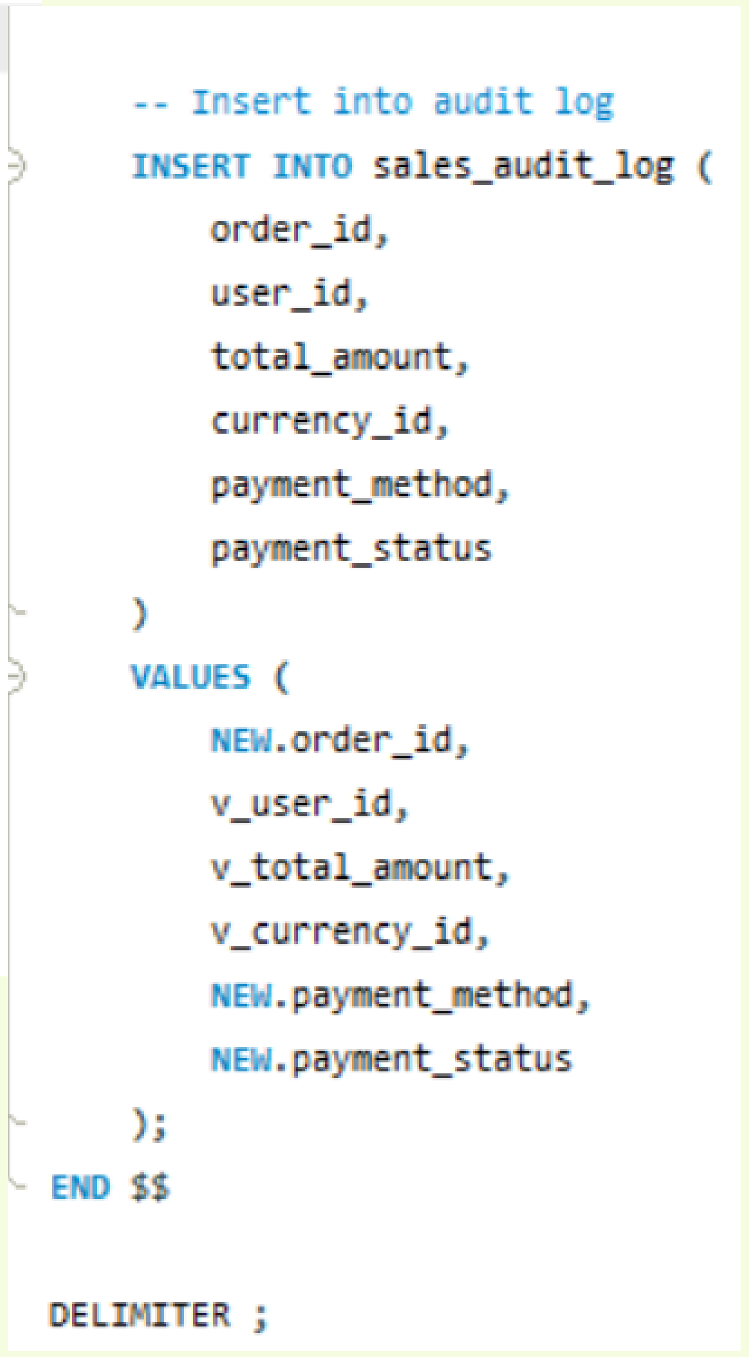
```
-- Trigger to log/aduit user details before deletion
DELIMITER $$
CREATE TRIGGER log_deleted_user
BEFORE DELETE ON Users
FOR EACH ROW
BEGIN
    INSERT INTO Audit_Deleted_Users (deleted_user_id, username, email)
    VALUES (OLD.user_id, OLD.username, OLD.email);
END;

$$ DELIMITER ;
```

This trigger logs user details (user ID, username, and email) into the *Audit_Deleted_Users* table before a user is deleted from the Users table. It ensures that user information is captured for auditing purposes prior to deletion.

PROCEDURES & TRIGGERS



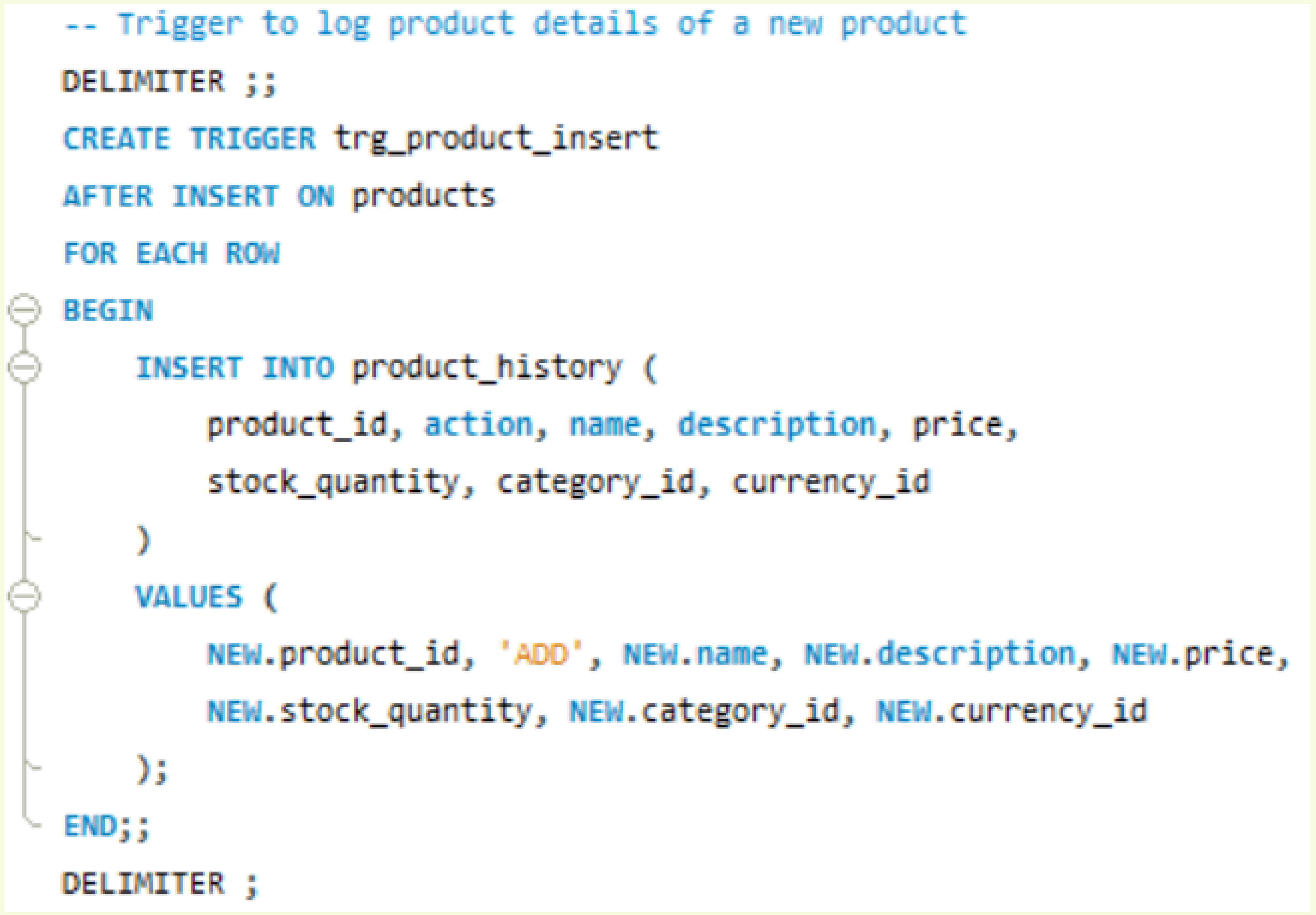
```sql
-- Trigger for the transaction log

DELIMITER $$

CREATE TRIGGER after_transaction_log_insert
AFTER INSERT ON transaction_log
FOR EACH ROW
BEGIN
    DECLARE v_user_id INT;
    DECLARE v_currency_id INT;
    DECLARE v_total_amount DECIMAL(10,2);

    -- Fetch user_id, total_amount, and currency_id from orders
    SELECT user_id, total_amount, currency_id
    INTO v_user_id, v_total_amount, v_currency_id
    FROM orders
    WHERE order_id = NEW.order_id;
```



```sql
    -- Insert into audit log
    INSERT INTO sales_audit_log (
        order_id,
        user_id,
        total_amount,
        currency_id,
        payment_method,
        payment_status
    )
    VALUES (
        NEW.order_id,
        v_user_id,
        v_total_amount,
        v_currency_id,
        NEW.payment_method,
        NEW.payment_status
    );
END $$

DELIMITER ;
```

This trigger logs transaction details into the *sales_audit_log* table after a new entry is inserted into the *transaction_log* table. It retrieves the *user_id, total_amount,* and *currency_id* from the related *orders* table and stores these, along with the payment method and status, for auditing purposes.

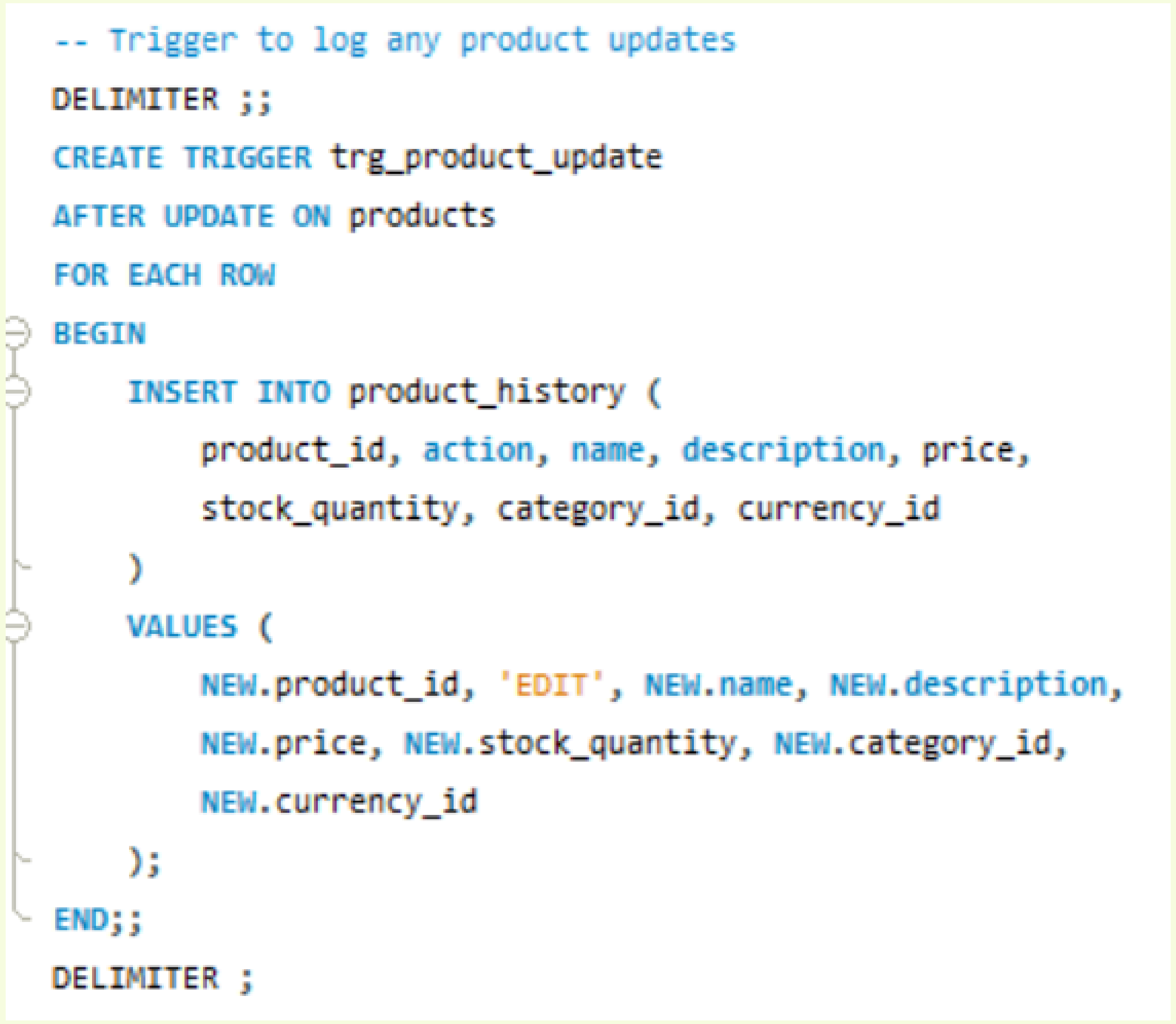
PROCEDURES & TRIGGERS



```
-- Trigger to log product details of a new product
DELIMITER ;;
CREATE TRIGGER trg_product_insert
AFTER INSERT ON products
FOR EACH ROW
BEGIN
    INSERT INTO product_history (
        product_id, action, name, description, price,
        stock_quantity, category_id, currency_id
    )
    VALUES (
        NEW.product_id, 'ADD', NEW.name, NEW.description, NEW.price,
        NEW.stock_quantity, NEW.category_id, NEW.currency_id
    );
END;;
DELIMITER ;
```

This trigger logs product details into the *product_history* table whenever a new product is added to the *products* table. It captures key information, including product name, description, price, stock quantity, category, and currency, along with the action 'ADD'.

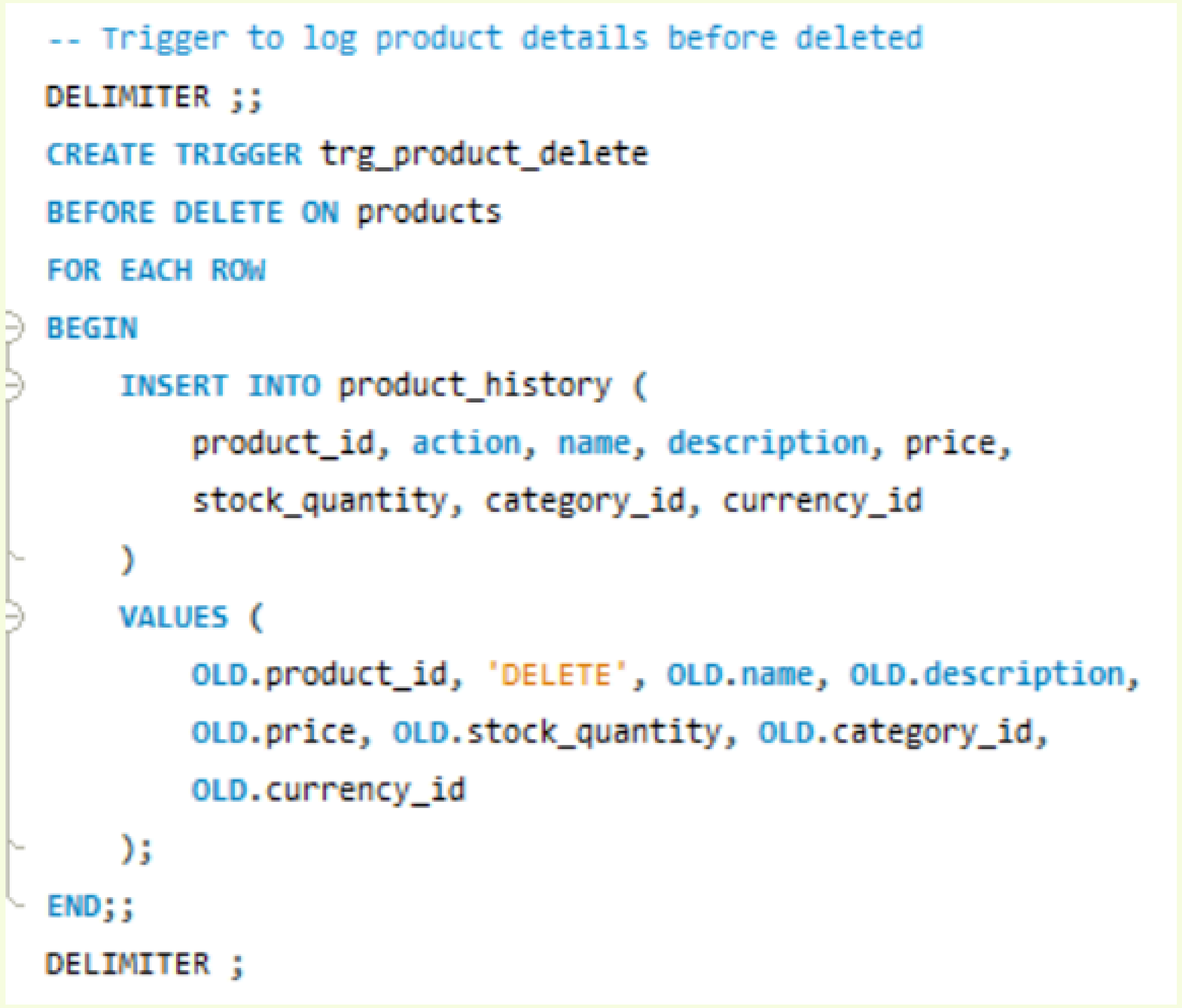
PROCEDURES & TRIGGERS



```
-- Trigger to log any product updates
DELIMITER ;;
CREATE TRIGGER trg_product_update
AFTER UPDATE ON products
FOR EACH ROW
BEGIN
    INSERT INTO product_history (
        product_id, action, name, description, price,
        stock_quantity, category_id, currency_id
    )
    VALUES (
        NEW.product_id, 'EDIT', NEW.name, NEW.description,
        NEW.price, NEW.stock_quantity, NEW.category_id,
        NEW.currency_id
    );
END;;
DELIMITER ;
```

This trigger logs the updated product details into the *product_history* table after a product in the *products* table is modified. It captures the updated product information with the action 'EDIT'.

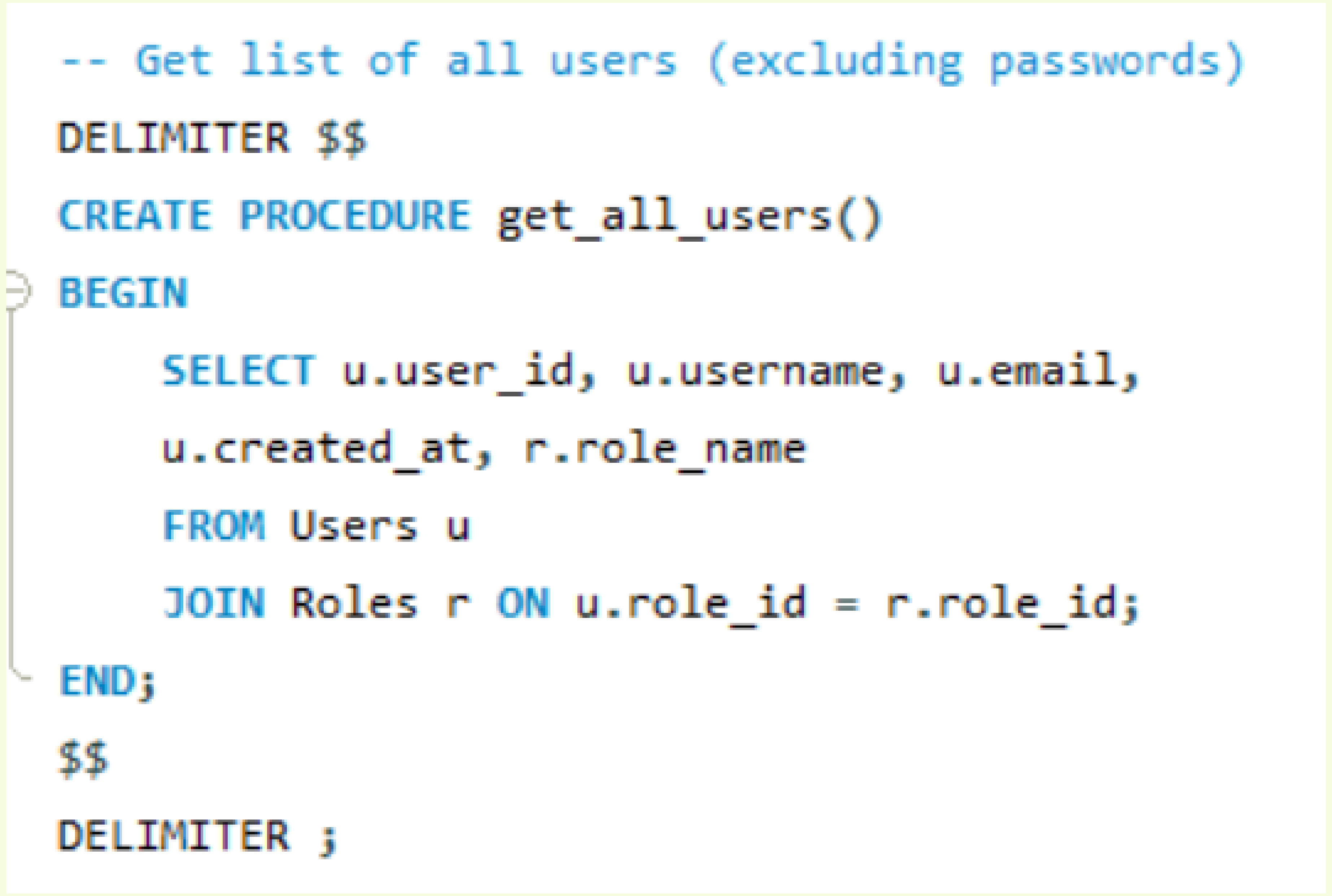
PROCEDURES & TRIGGERS



```
-- Trigger to log product details before deleted
DELIMITER ;;
CREATE TRIGGER trg_product_delete
BEFORE DELETE ON products
FOR EACH ROW
BEGIN
    INSERT INTO product_history (
        product_id, action, name, description, price,
        stock_quantity, category_id, currency_id
    )
    VALUES (
        OLD.product_id, 'DELETE', OLD.name, OLD.description,
        OLD.price, OLD.stock_quantity, OLD.category_id,
        OLD.currency_id
    );
END;;
DELIMITER ;
```

This trigger logs the product details into the *product_history* table before a product is deleted from the *products* table. It captures the product's information with the action 'DELETE' for auditing purposes.

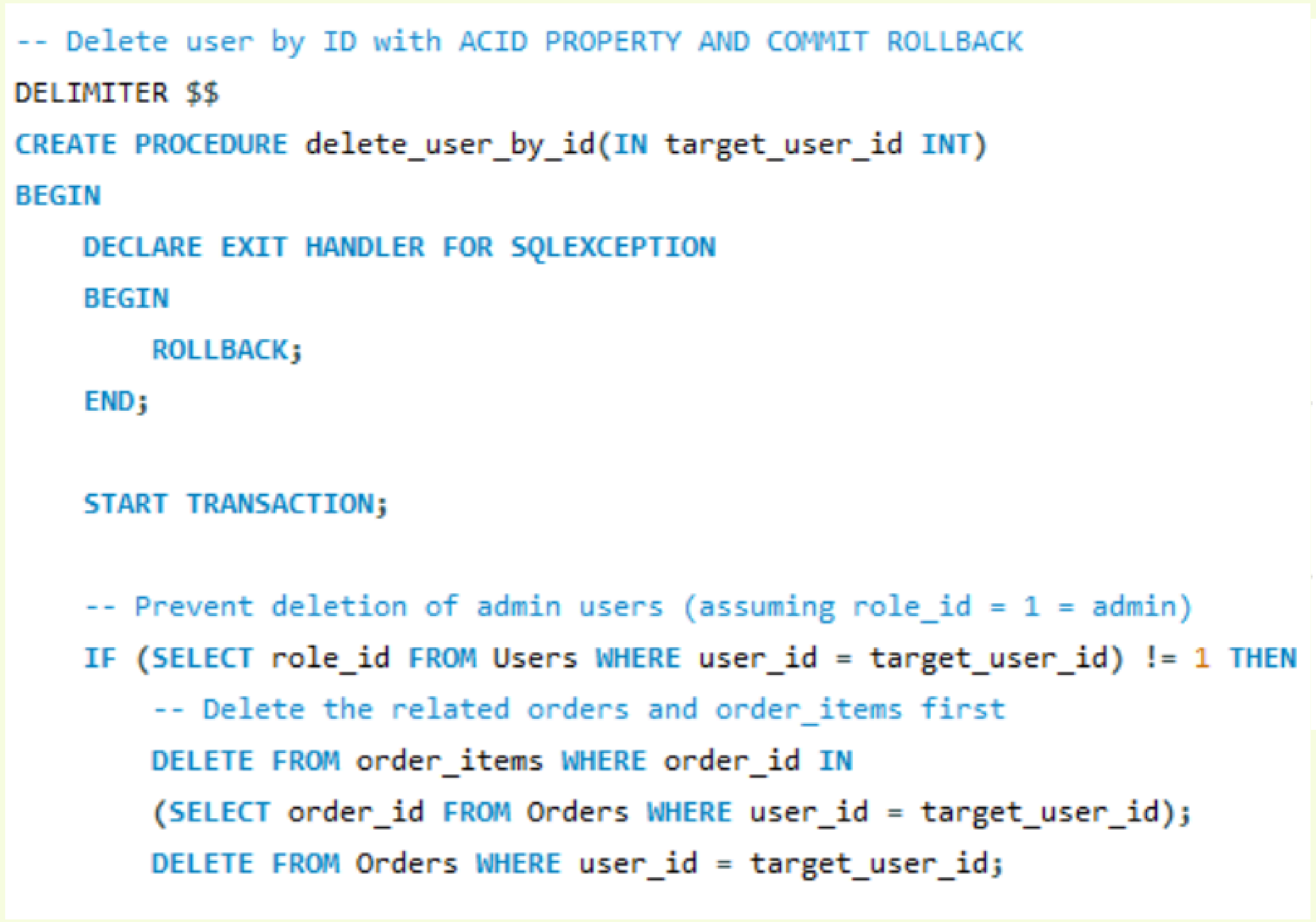
PROCEDURES & TRIGGERS



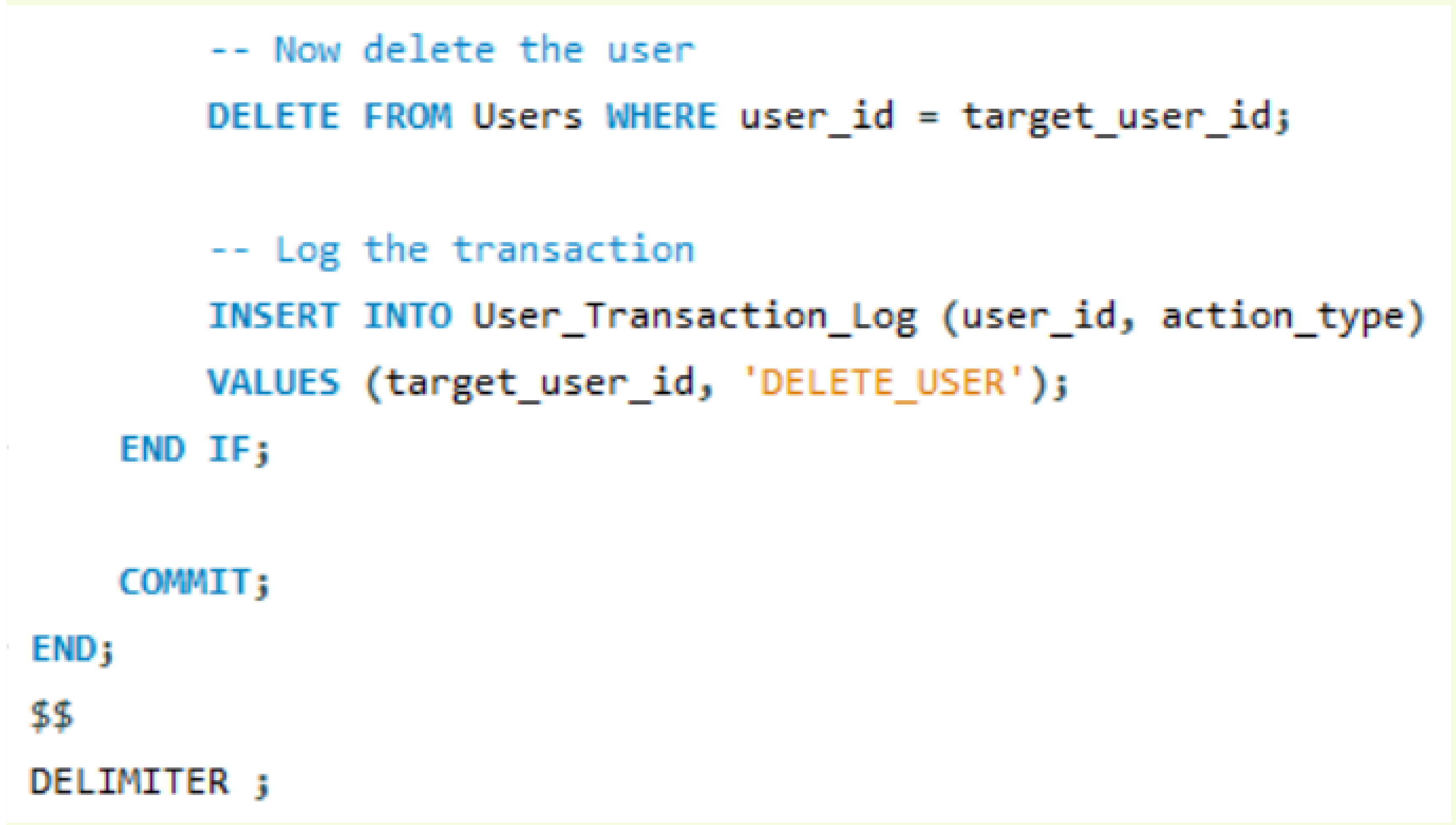
```
-- Get list of all users (excluding passwords)
DELIMITER $$
CREATE PROCEDURE get_all_users()
BEGIN
    SELECT u.user_id, u.username, u.email,
    u.created_at, r.role_name
    FROM Users u
    JOIN Roles r ON u.role_id = r.role_id;
END;
$$
DELIMITER ;
```

This stored procedure retrieves a list of all users from the *Users* table, excluding passwords. It selects the user ID, username, email, account creation date, and associated role name by joining the *Users* table with the *Roles* table based on the *role_id*.

PROCEDURES & TRIGGERS



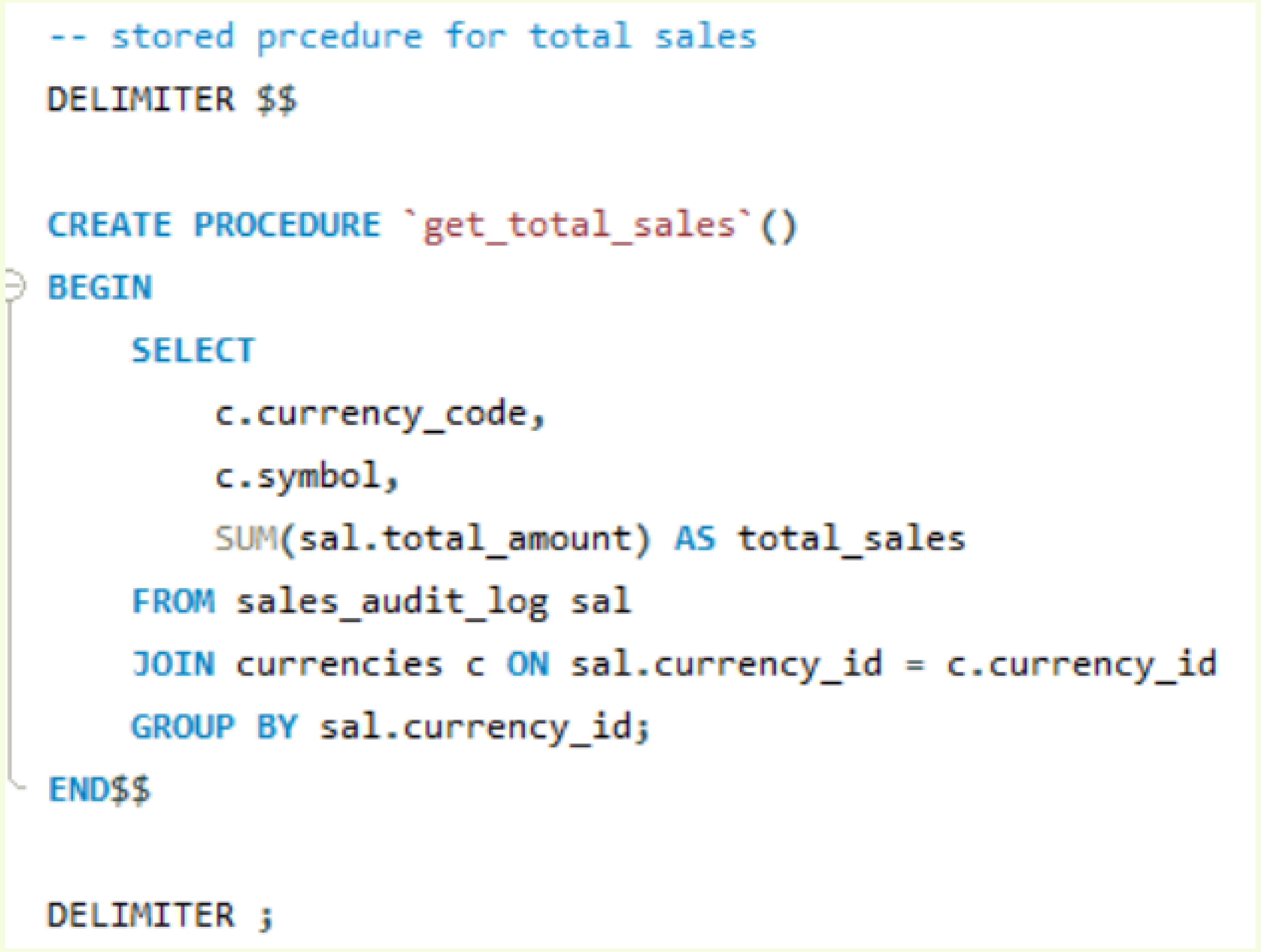
```
-- Delete user by ID with ACID PROPERTY AND COMMIT ROLLBACK
DELIMITER $$
CREATE PROCEDURE delete_user_by_id(IN target_user_id INT)
BEGIN
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        ROLLBACK;
    END;

    START TRANSACTION;

    -- Prevent deletion of admin users (assuming role_id = 1 = admin)
    IF (SELECT role_id FROM Users WHERE user_id = target_user_id) != 1 THEN
        -- Delete the related orders and order_items first
        DELETE FROM order_items WHERE order_id IN
        (SELECT order_id FROM Orders WHERE user_id = target_user_id);
        DELETE FROM Orders WHERE user_id = target_user_id;
```



```
        -- Now delete the user
        DELETE FROM Users WHERE user_id = target_user_id;

        -- Log the transaction
        INSERT INTO User_Transaction_Log (user_id, action_type)
        VALUES (target_user_id, 'DELETE_USER');
    END IF;

    COMMIT;
END;
$$
DELIMITER ;
```

This stored procedure deletes a user by their *user_id* with ACID properties. It prevents the deletion of admin users (*role_id = 1*) and first deletes related orders and order items before removing the user from the *Users* table. It logs the deletion action in the *User_Transaction_Log* table. In case of errors, the transaction is rolled back; otherwise, it is committed.

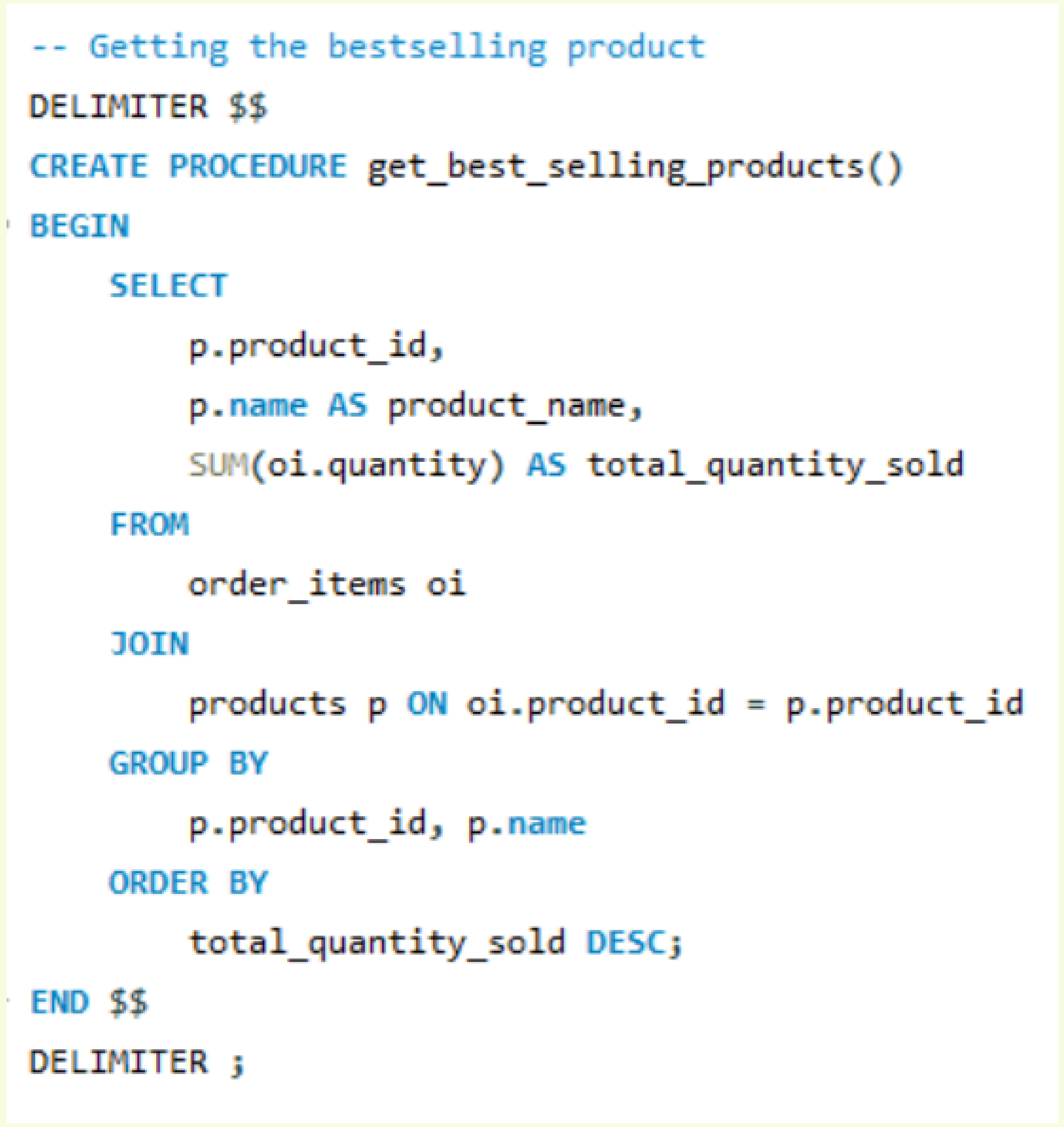
PROCEDURES & TRIGGERS



```
-- stored prcedure for total sales
DELIMITER $$

CREATE PROCEDURE `get_total_sales`()
BEGIN
    SELECT
        c.currency_code,
        c.symbol,
        SUM(sal.total_amount) AS total_sales
    FROM sales_audit_log sal
    JOIN currencies c ON sal.currency_id = c.currency_id
    GROUP BY sal.currency_id;
END$$

DELIMITER ;
```

This stored procedure calculates the total sales for each currency by summing the *total_amount* from the *sales_audit_log* table. It joins the *currencies* table to fetch the currency code and symbol and groups the results by *currency_id* to provide the total sales per currency.

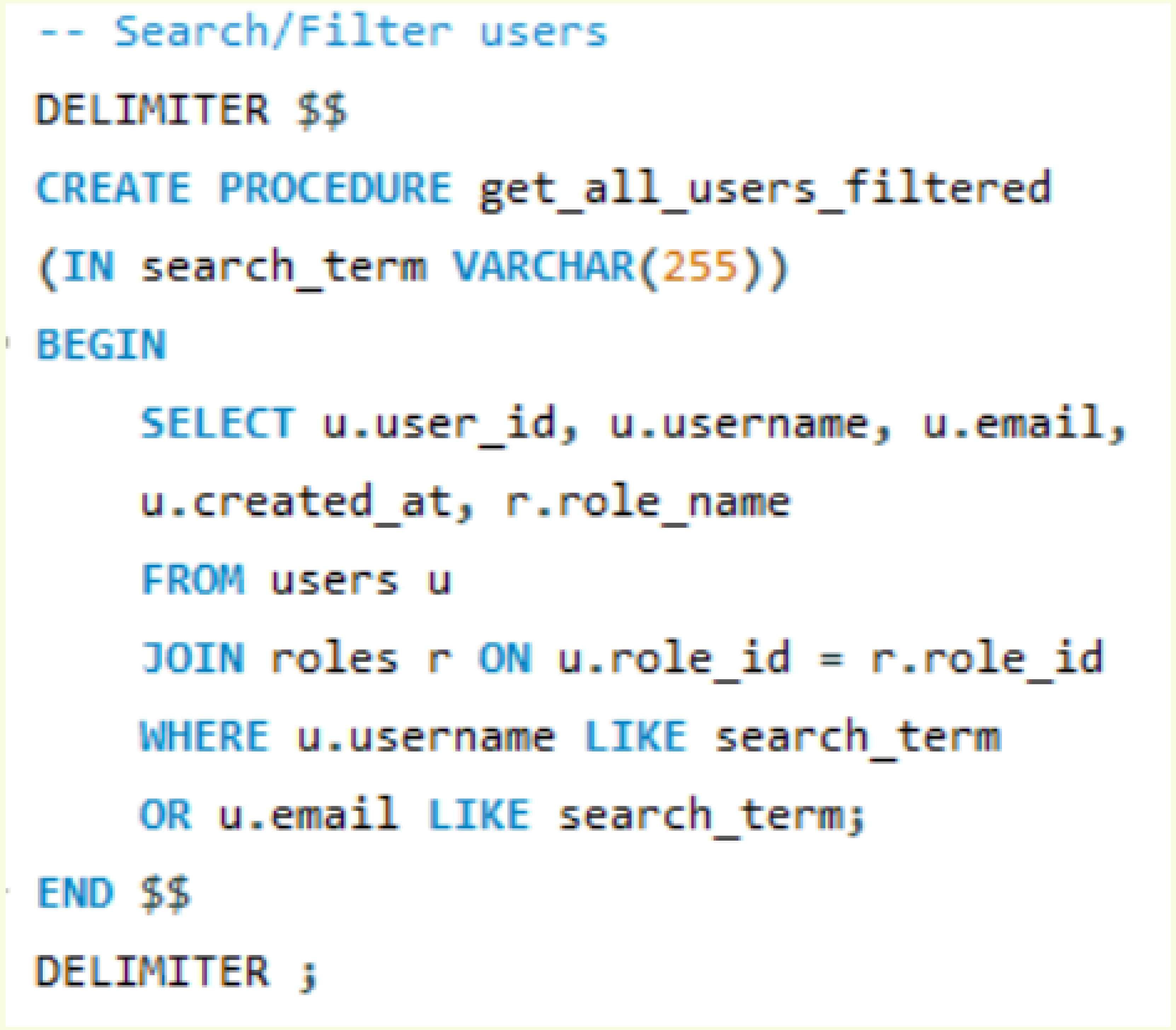
PROCEDURES & TRIGGERS



```
-- Getting the bestselling product
DELIMITER $$
CREATE PROCEDURE get_best_selling_products()
BEGIN
    SELECT
        p.product_id,
        p.name AS product_name,
        SUM(oi.quantity) AS total_quantity_sold
    FROM
        order_items oi
    JOIN
        products p ON oi.product_id = p.product_id
    GROUP BY
        p.product_id, p.name
    ORDER BY
        total_quantity_sold DESC;
END $$
DELIMITER ;
```

This stored procedure retrieves the best-selling products by calculating the total quantity sold for each product. It joins the *order_items* table with the *products* table to sum the quantities of each product sold. The results are grouped by product ID and name, then ordered in descending order by the total quantity sold, displaying the most popular products at the top.

PROCEDURES & TRIGGERS

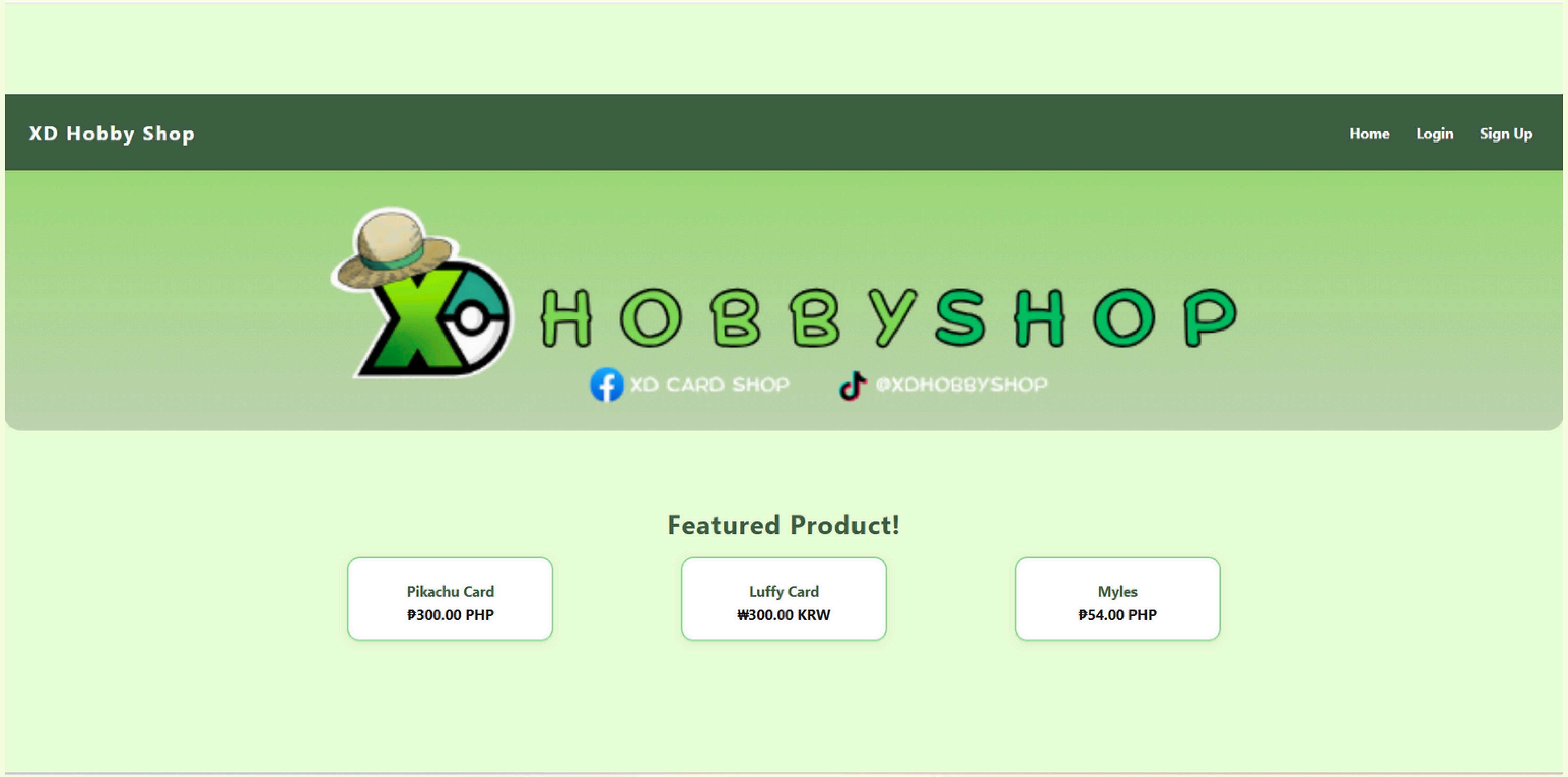


```
-- Search/Filter users
DELIMITER $$
CREATE PROCEDURE get_all_users_filtered
(IN search_term VARCHAR(255))
BEGIN
    SELECT u.user_id, u.username, u.email,
    u.created_at, r.role_name
    FROM users u
    JOIN roles r ON u.role_id = r.role_id
    WHERE u.username LIKE search_term
    OR u.email LIKE search_term;
END $$
DELIMITER ;
```

This stored procedure allows for searching and filtering users based on a provided search term. It takes a *search_term* as input and retrieves user details, including user ID, username, email, and role name. The procedure searches for matches in both the *username* and *email* fields, returning users whose details contain the search term. The results are joined with the *roles* table to display the associated role name for each user.

GUI SCREENSHOTS

GUI SCREENSHOTS



XD Hobby Shop

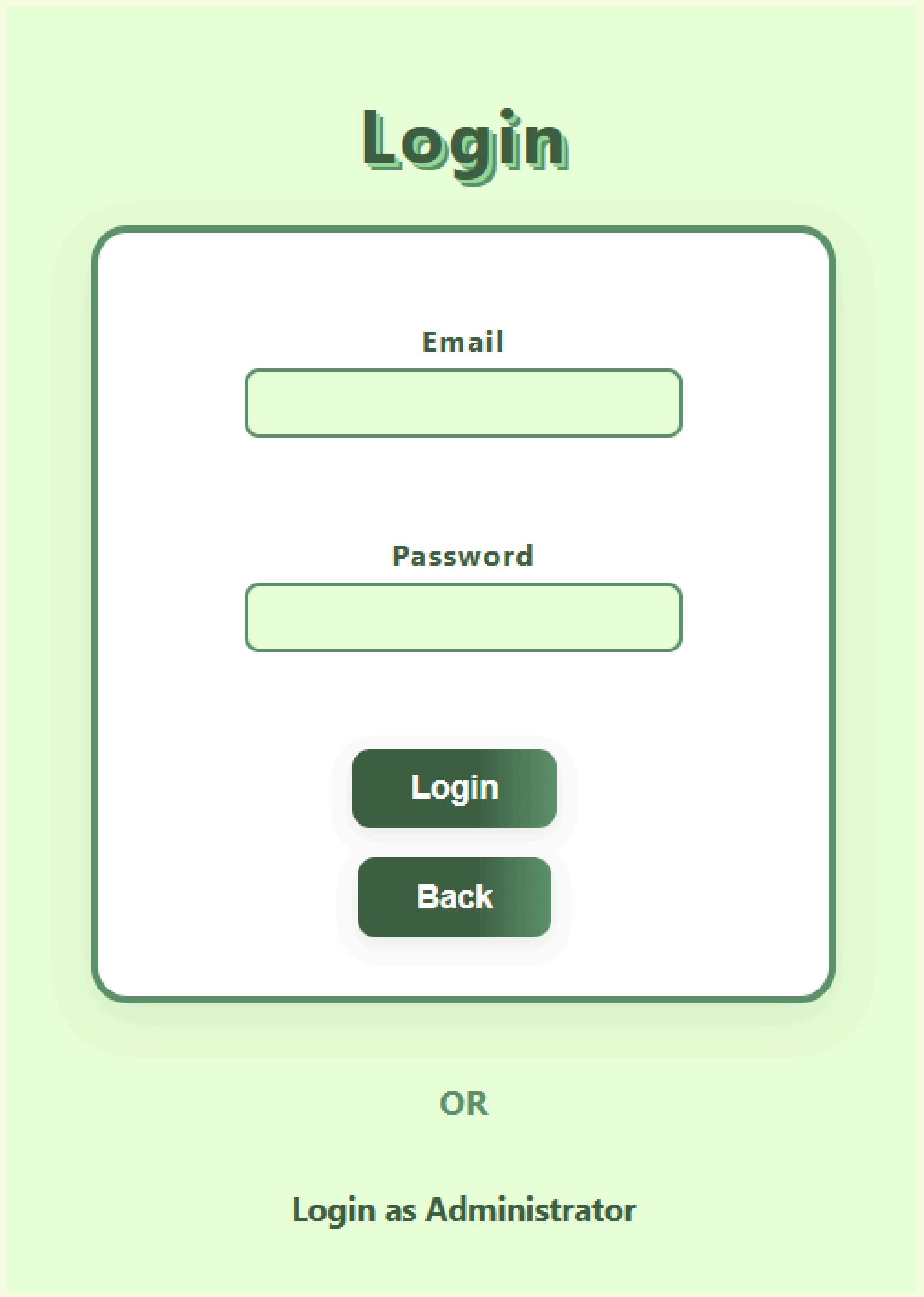
Featured Product!

Pikachu Card
₱300.00 PHP

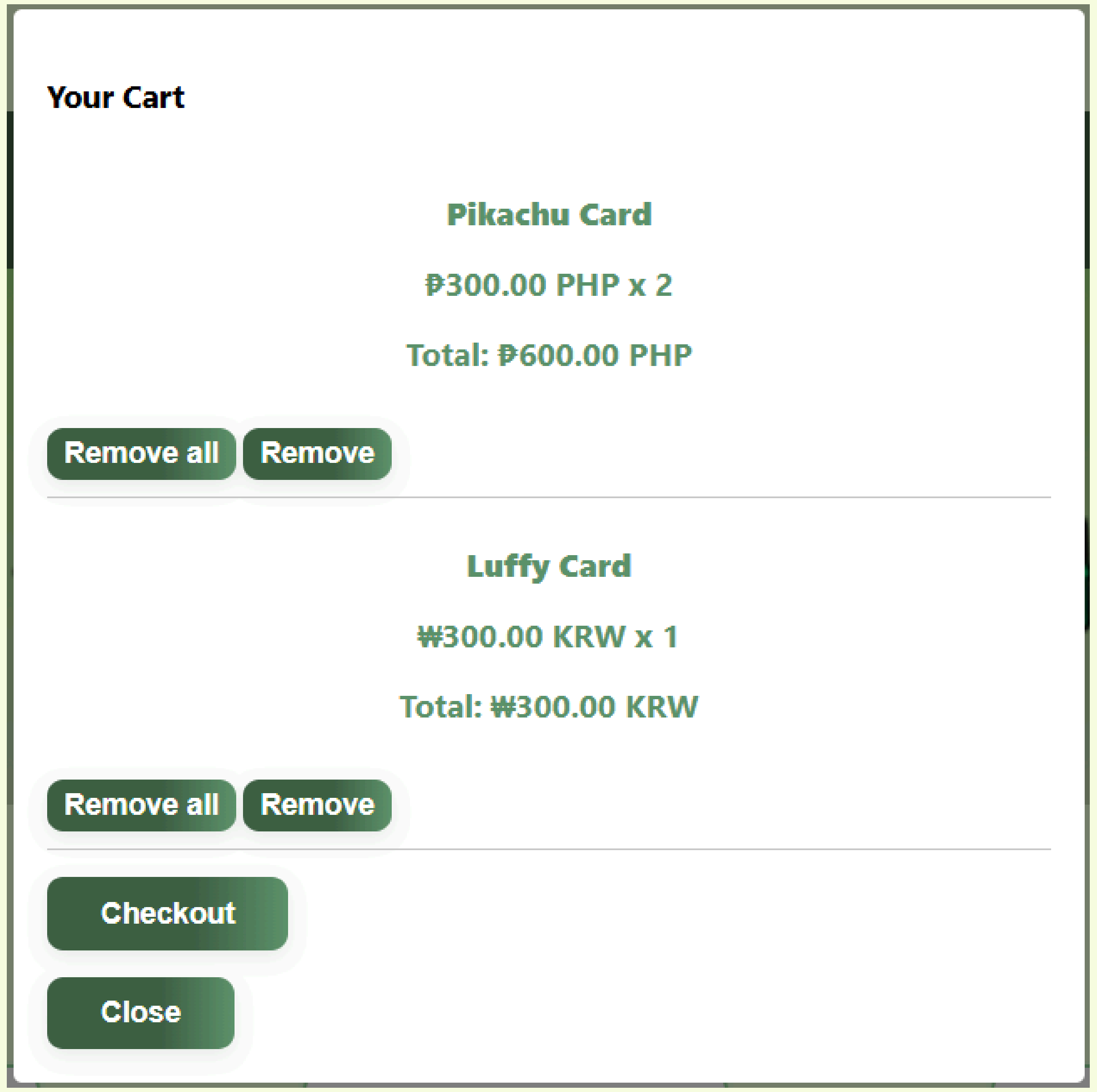
Luffy Card
₩300.00 KRW

Myles
₱54.00 PHP

GUI SCREENSHOTS



GUI SCREENSHOTS



Your Cart

Pikachu Card

₱300.00 PHP x 2

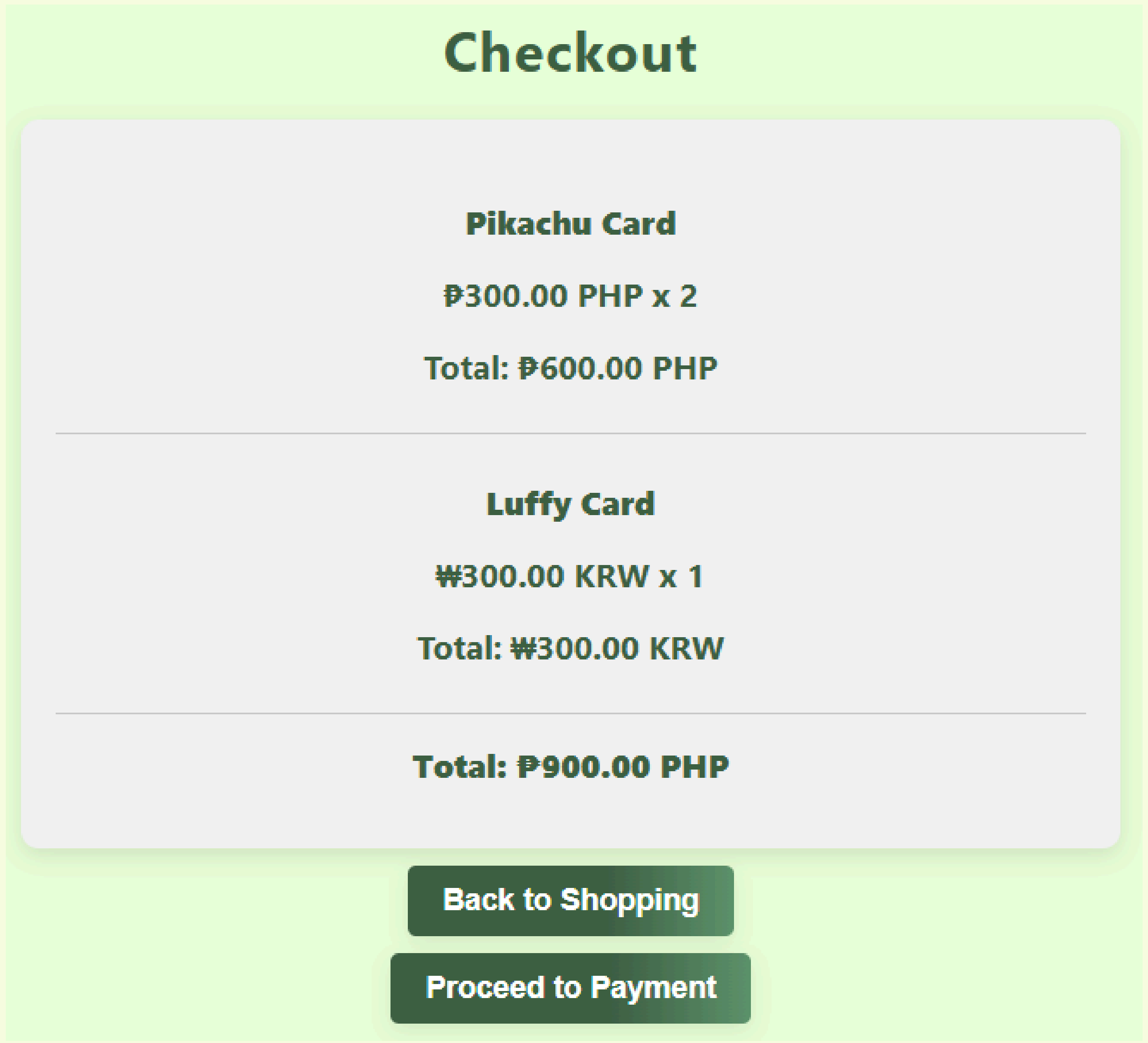
Total: ₱600.00 PHP

Luffy Card

₩300.00 KRW x 1

Total: ₩300.00 KRW

GUI SCREENSHOTS


Checkout

Pikachu Card

₱300.00 PHP x 2

Total: ₱600.00 PHP

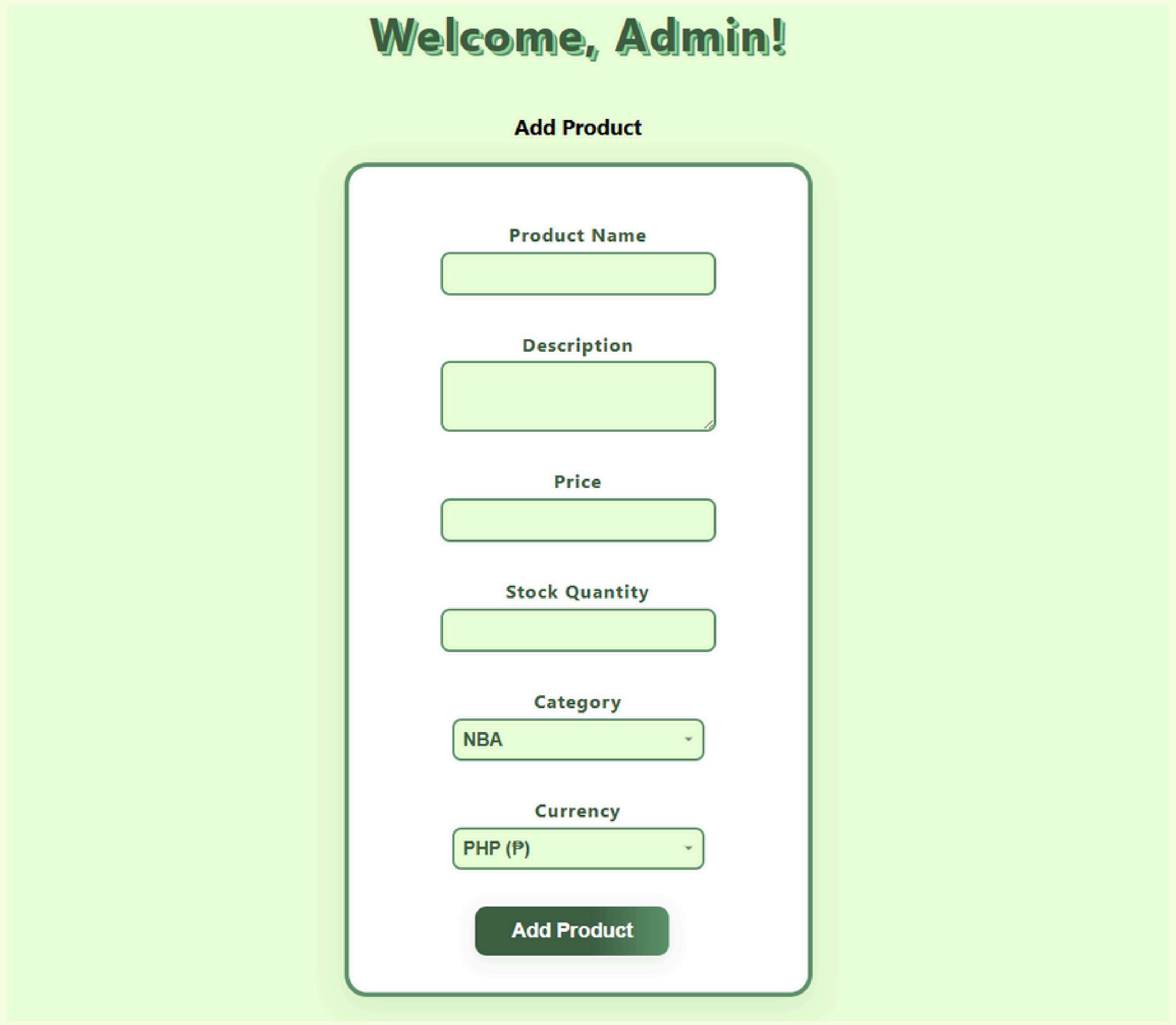
Luffy Card

₩300.00 KRW x 1

Total: ₩300.00 KRW

Total: ₱900.00 PHP

GUI SCREENSHOTS


Welcome, Admin!

Add Product

Product Name

Description

Price

Stock Quantity

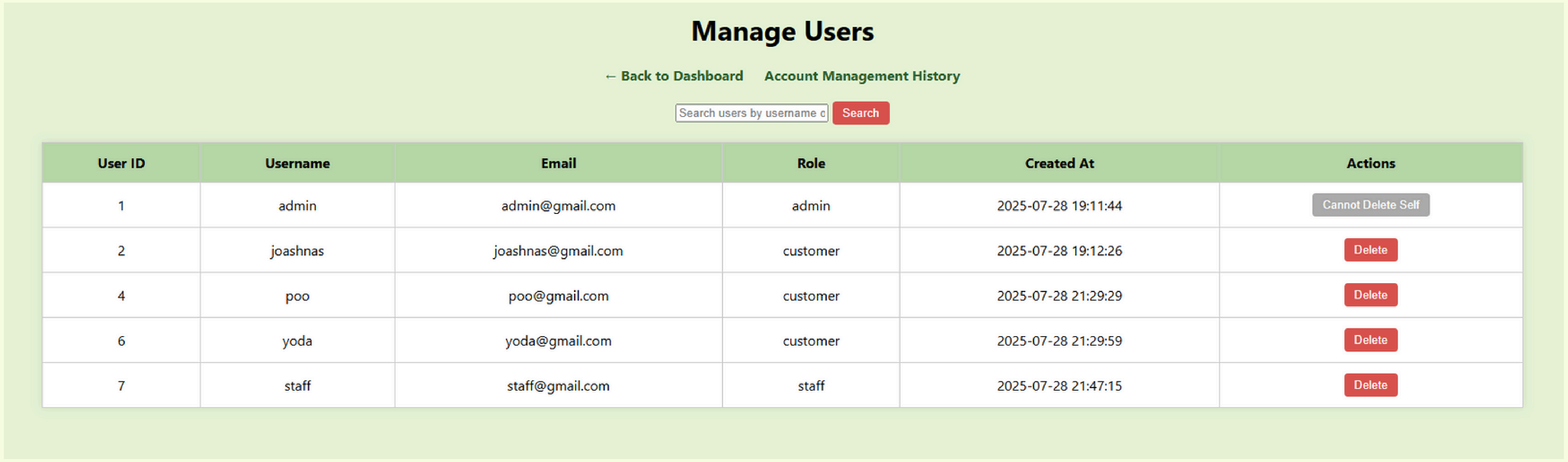
Category

NBA

Currency

PHP (₱)

GUI SCREENSHOTS



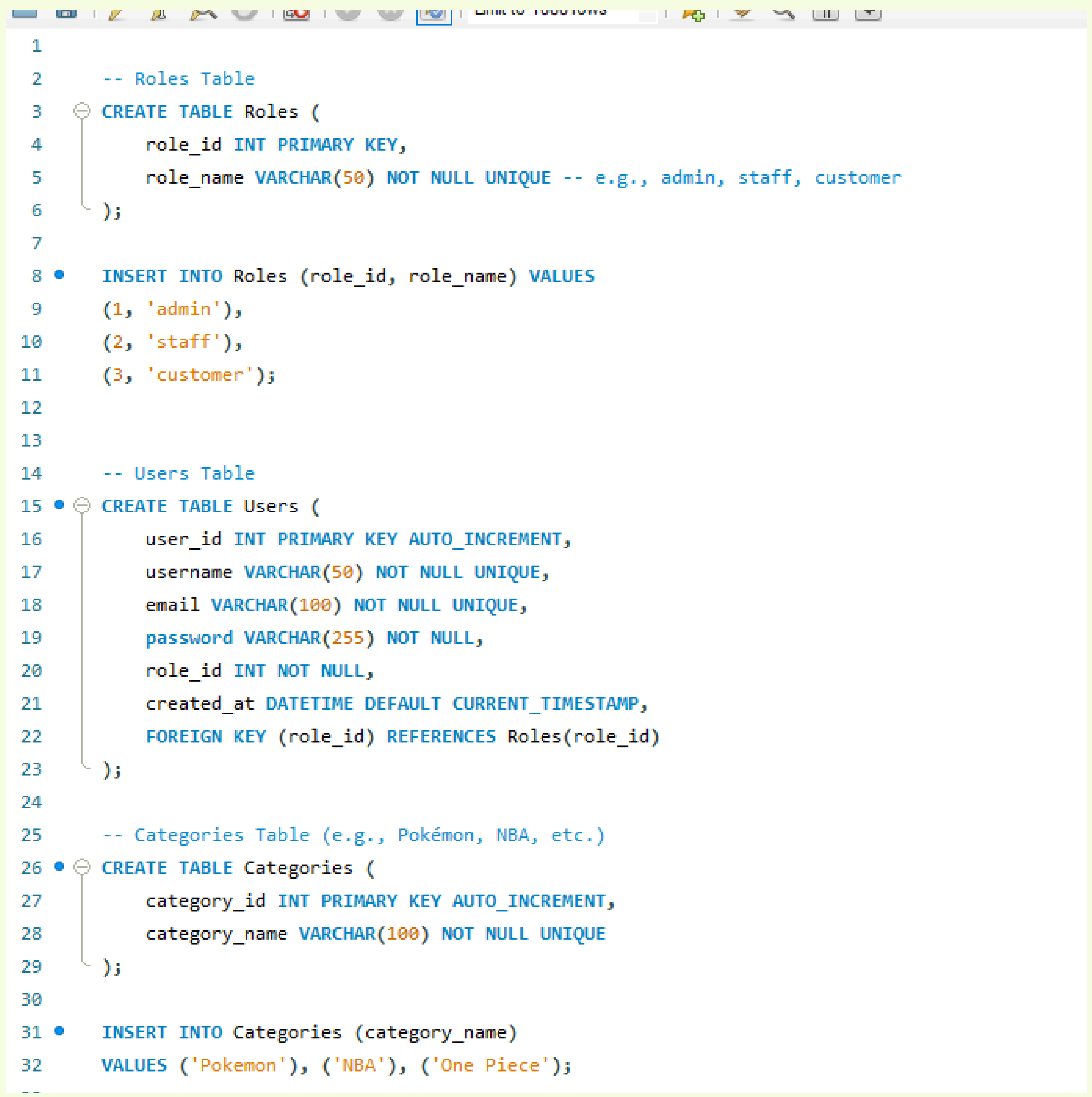
Manage Users

← Back to Dashboard   Account Management History

User ID	Username	Email	Role	Created At	Actions
1	admin	admin@gmail.com	admin	2025-07-28 19:11:44	Cannot Delete Self
2	joashnas	joashnas@gmail.com	customer	2025-07-28 19:12:26	Delete
4	poo	poo@gmail.com	customer	2025-07-28 21:29:29	Delete
6	yoda	yoda@gmail.com	customer	2025-07-28 21:29:59	Delete
7	staff	staff@gmail.com	staff	2025-07-28 21:47:15	Delete

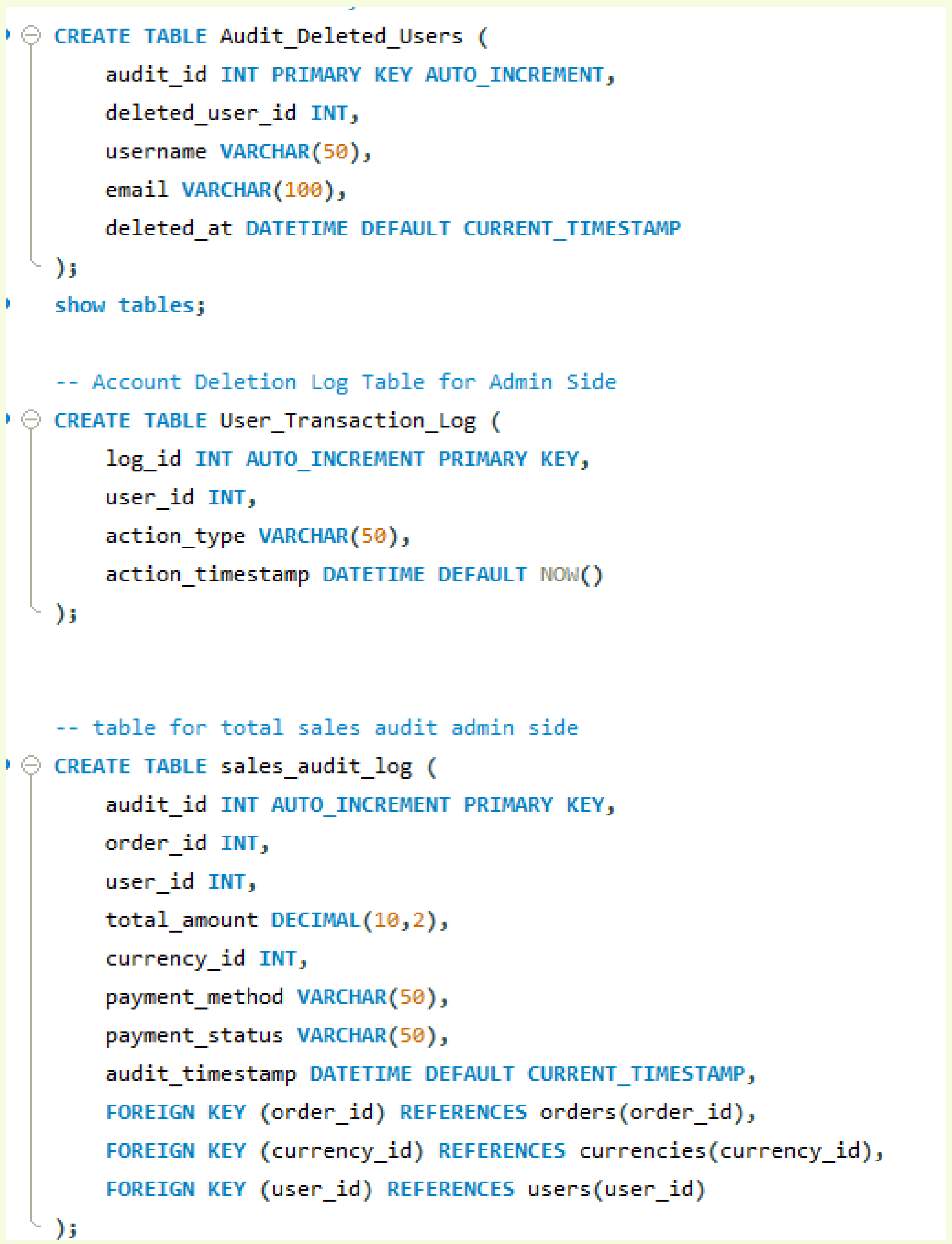
SQL CODE SNIPPETS

SQL CODE SNIPPETS



```sql

-- Roles Table
CREATE TABLE Roles (
    role_id INT PRIMARY KEY,
    role_name VARCHAR(50) NOT NULL UNIQUE -- e.g., admin, staff, customer
);

INSERT INTO Roles (role_id, role_name) VALUES
(1, 'admin'),
(2, 'staff'),
(3, 'customer');


-- Users Table
CREATE TABLE Users (
    user_id INT PRIMARY KEY AUTO_INCREMENT,
    username VARCHAR(50) NOT NULL UNIQUE,
    email VARCHAR(100) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL,
    role_id INT NOT NULL,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (role_id) REFERENCES Roles(role_id)
);

-- Categories Table (e.g., Pokémon, NBA, etc.)
CREATE TABLE Categories (
    category_id INT PRIMARY KEY AUTO_INCREMENT,
    category_name VARCHAR(100) NOT NULL UNIQUE
);

INSERT INTO Categories (category_name)
VALUES ('Pokemon'), ('NBA'), ('One Piece');
```

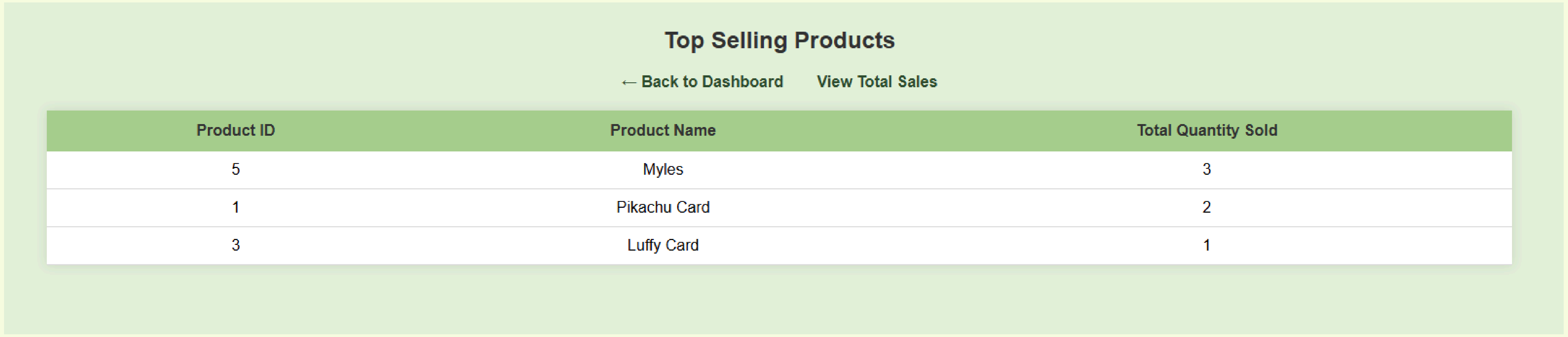
SQL CODE SNIPPETS



```
CREATE TABLE Audit_Deleted_Users (
    audit_id INT PRIMARY KEY AUTO_INCREMENT,
    deleted_user_id INT,
    username VARCHAR(50),
    email VARCHAR(100),
    deleted_at DATETIME DEFAULT CURRENT_TIMESTAMP
);
show tables;

-- Account Deletion Log Table for Admin Side
CREATE TABLE User_Transaction_Log (
    log_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT,
    action_type VARCHAR(50),
    action_timestamp DATETIME DEFAULT NOW()
);


-- table for total sales audit admin side
CREATE TABLE sales_audit_log (
    audit_id INT AUTO_INCREMENT PRIMARY KEY,
    order_id INT,
    user_id INT,
    total_amount DECIMAL(10,2),
    currency_id INT,
    payment_method VARCHAR(50),
    payment_status VARCHAR(50),
    audit_timestamp DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (order_id) REFERENCES orders(order_id),
    FOREIGN KEY (currency_id) REFERENCES currencies(currency_id),
    FOREIGN KEY (user_id) REFERENCES users(user_id)
);
```

EXAMPLE OUTPUTS

EXAMPLE OUTPUTS

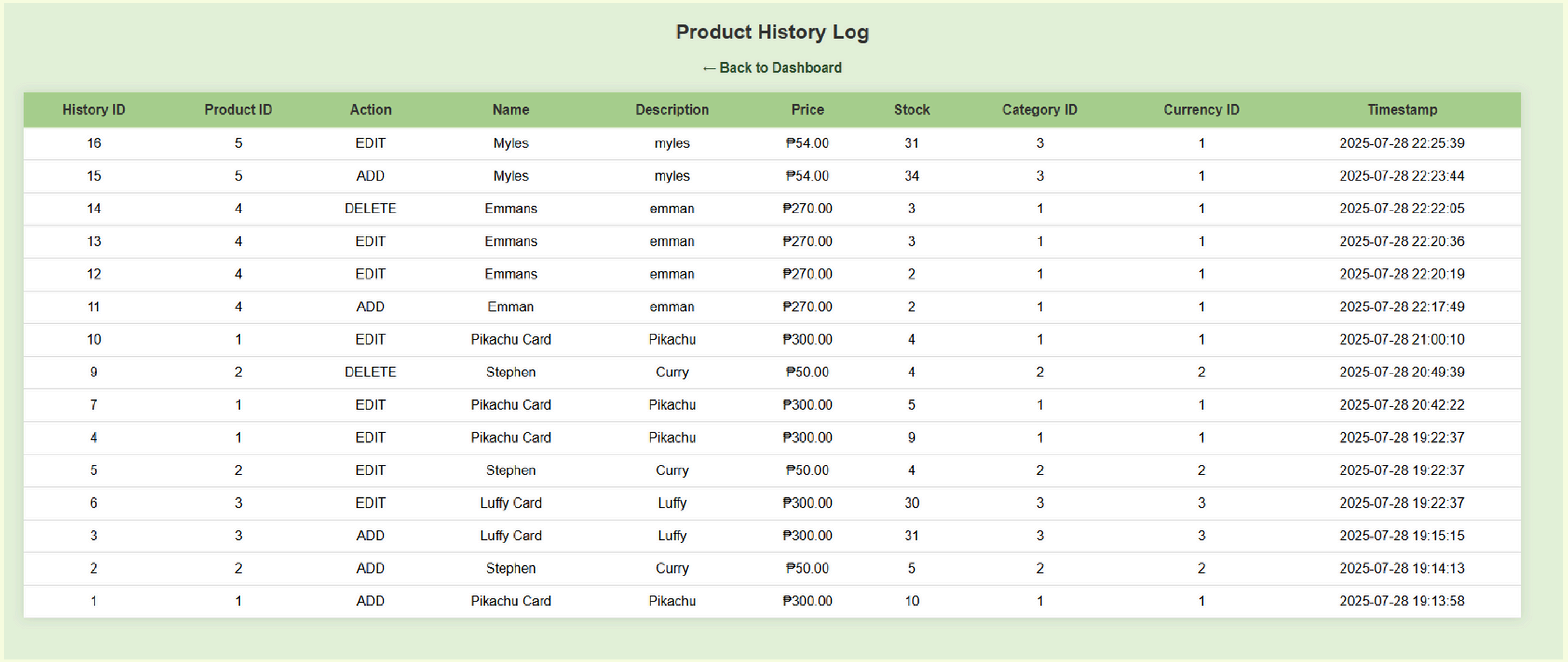


Top Selling Products

← Back to Dashboard   View Total Sales

Product ID	Product Name	Total Quantity Sold
5	Myles	3
1	Pikachu Card	2
3	Luffy Card	1

EXAMPLE OUTPUTS



Product History Log

← Back to Dashboard

History ID	Product ID	Action	Name	Description	Price	Stock	Category ID	Currency ID	Timestamp
16	5	EDIT	Myles	myles	₱54.00	31	3	1	2025-07-28 22:25:39
15	5	ADD	Myles	myles	₱54.00	34	3	1	2025-07-28 22:23:44
14	4	DELETE	Emmans	emman	₱270.00	3	1	1	2025-07-28 22:22:05
13	4	EDIT	Emmans	emman	₱270.00	3	1	1	2025-07-28 22:20:36
12	4	EDIT	Emmans	emman	₱270.00	2	1	1	2025-07-28 22:20:19
11	4	ADD	Emman	emman	₱270.00	2	1	1	2025-07-28 22:17:49
10	1	EDIT	Pikachu Card	Pikachu	₱300.00	4	1	1	2025-07-28 21:00:10
9	2	DELETE	Stephen	Curry	₱50.00	4	2	2	2025-07-28 20:49:39
7	1	EDIT	Pikachu Card	Pikachu	₱300.00	5	1	1	2025-07-28 20:42:22
4	1	EDIT	Pikachu Card	Pikachu	₱300.00	9	1	1	2025-07-28 19:22:37
5	2	EDIT	Stephen	Curry	₱50.00	4	2	2	2025-07-28 19:22:37
6	3	EDIT	Luffy Card	Luffy	₱300.00	30	3	3	2025-07-28 19:22:37
3	3	ADD	Luffy Card	Luffy	₱300.00	31	3	3	2025-07-28 19:15:15
2	2	ADD	Stephen	Curry	₱50.00	5	2	2	2025-07-28 19:14:13
1	1	ADD	Pikachu Card	Pikachu	₱300.00	10	1	1	2025-07-28 19:13:58

THANK YOU