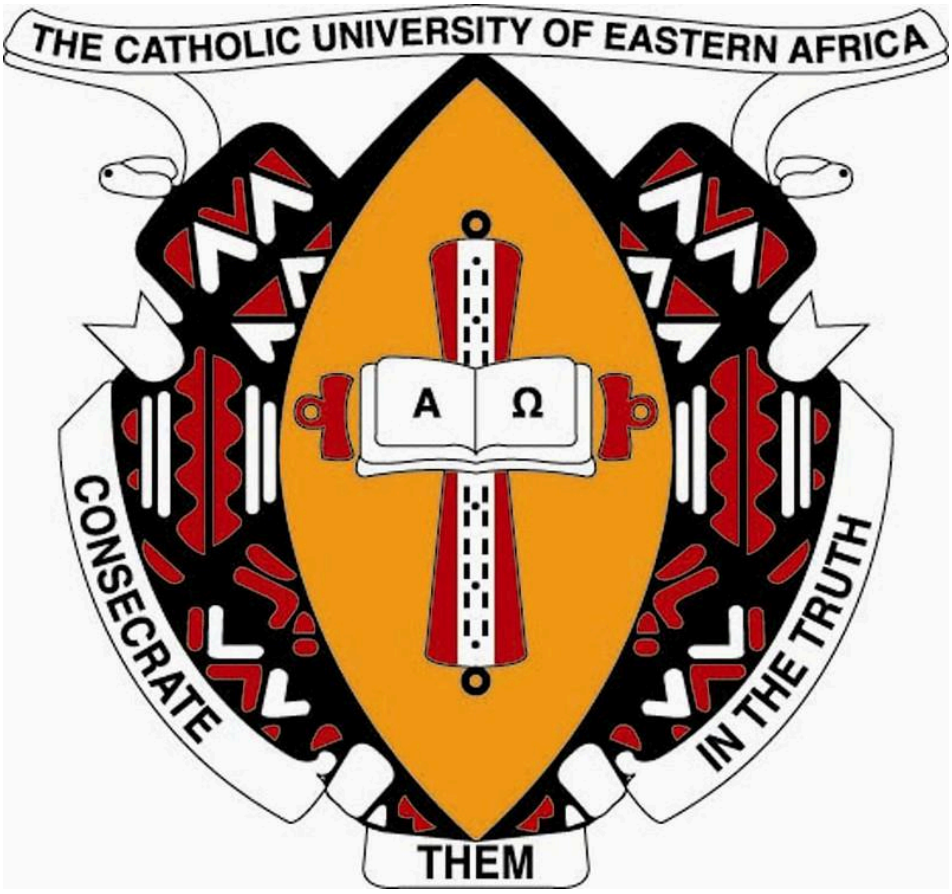


Members:

1. Eric Nzomo - 1052079
2. Andanje Lewis - 1049466
3. Okongo Eugine - 1049480
4. Eve Njore- 1049493
5. Barasa Laureen- 1048781

ADVANCED DATABASE (CMT 302)

Food Delivery Management System

Submission Date: 19/11/2024

Table Of Contents

Project Overview

Introduction

Food Delivery Management System (FDMS) is a database solution designed to optimize and manage the operations of food delivery services. It addresses the challenges of order placement, restaurant inventories, menu availability and delivery rider records. Over the years the food & restaurant industries have grown at rapid rates. The demand for food delivery systems has been on a rampant rise as the current system relies on manual and inefficient record keeping methods. The FDMS enhances order accuracy, ensures seamless communication between restaurants, customers and riders. Through this system we aim to reduce errors by a significant percentage and maximize operational efficiency.

Objectives

General Objectives

To develop a database system that helps to manage Food delivery operations such as restaurant details order placement, payment details, menu options and rider details.

Specific Objectives

1. Allow customers to efficiently track orders from placement to delivery.
2. Allow restaurants to manage orders, menus and their delivery riders
3. Allow rider management to easily allocate and deallocate deliveries based on workload.

Scope

Customer View

- User registration and user profile management
- Browser restaurants and menus
- Place and manage orders
- Secure online payment integration

Restaurant View

- Create, update and delete menu

- Customer order management

Riders View

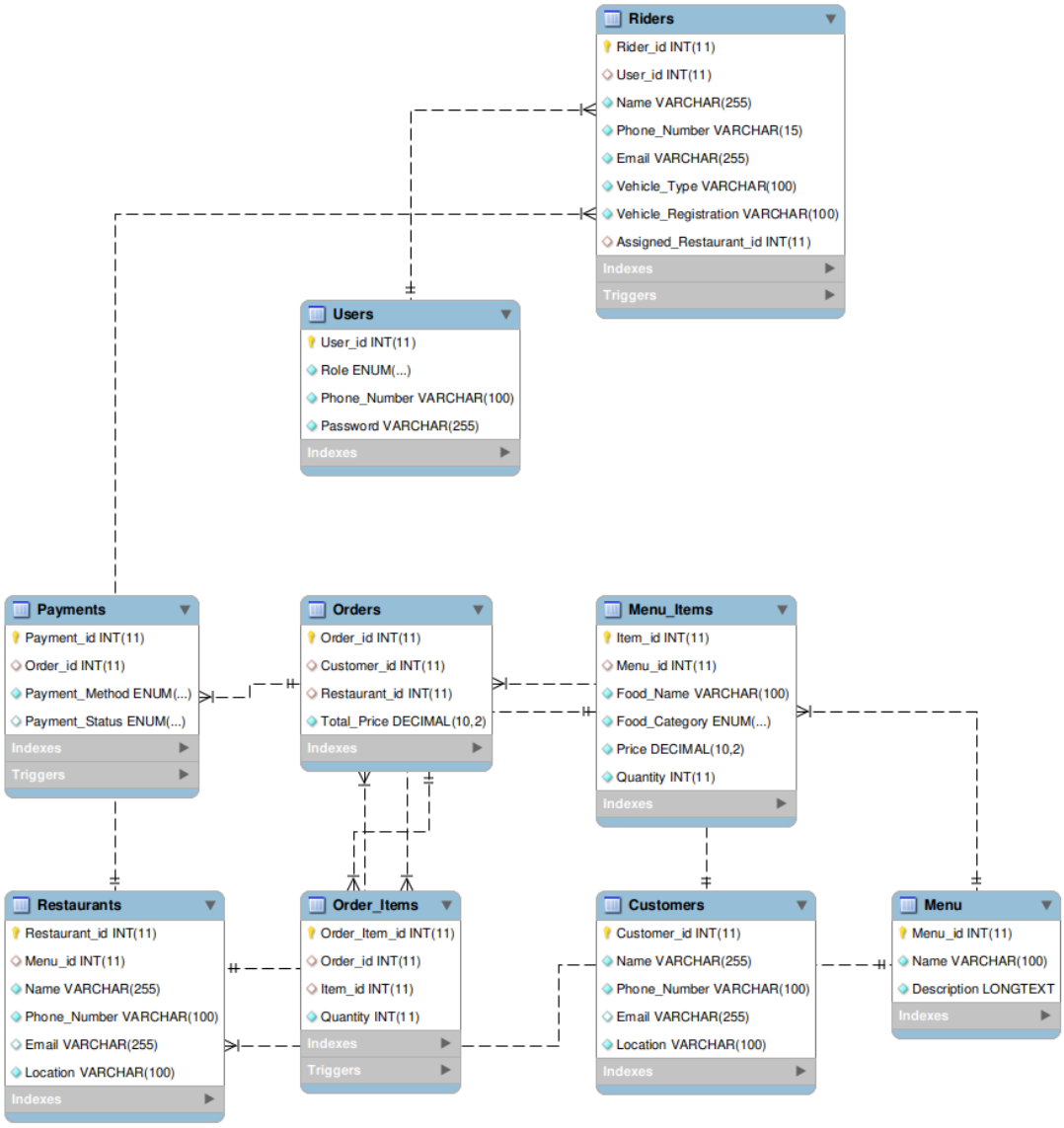
- Rider registration and profile management
- Rider assigned orders

Administrator View

- Mange customers, restaurants and riders
- Generate performance reports

Database Design

ER Diagram



Data Dictionary

Users

Column Name	Data Type	Constraints
User_id	INT	PRIMARY KEY, AUTO_INCREMENT
Role	ENUM	NOT NULL(admin/manager/customer/rider)
Phone_Number	VARCHAR(100)	NOT NULL
Password	VARCHAR(255)	NOT NULL

Customers

Column Name	Data Type	Constraints
Customer_id	INT	PRIMARY KEY, AUTO_INCREMENT
User_id	INT	FOREIGN KEY REFERENCES Users(IUsers_id)
Name	VARCHAR(255)	NOT NULL
Phone_Number	VARCHAR(100)	NOT NULL
Email	VARCHAR(255)	UNIQUE
Location	VARCHAR(100)	NOT NULL

Restaurants

Column Name	Data Type	Constraints
Restaurant_id	INT	PRIMARY KEY, AUTO_INCREMENT
User_id	INT	FOREIGN KEY REFERENCES Users(User_id)
Name	VARCHAR(255)	NOT NULL
Phone_Number	VARCHAR(100)	NOT NULL
Email	VARCHAR(255)	UNIQUE
Location	VARCHAR(100)	NOT NULL

Menu

Column Name	Data Type	Constraints
Menu_id	INT	PRIMARY KEY, AUTO_INCREMENT
Restaurant_id	INT	FOREIGN KEY REFERENCES Restaurants(Restaurant_id)
Name	VARCHAR(100)	NOT NULL
Description	LONG TEXT	NOT NULL

Menu_Items

Column Name	Data Type	Constraints
Item_id	INT	PRIMARY KEY, AUTO_INCREMENT
Menu_id	INT	FOREIGN KEY REFERENCES Menu(Menu_id)
Food_Name	VARCHAR(100)	NOT NULL
Food_Category	ENUM	NOT NULL(starter/main/dessert/beverage)
Price	DECIMAL(10,2)	NOT NULL
Quantity	INT	NOT NULL

Orders

Column Name	Data Type	Constraints
Order_id	INT	PRIMARY KEY, AUTO_INCREMENT
Customer_id	INT	FOREIGN KEY REFERENCES Customers(Customer_id)
Restaurant_id	INT	FOREIGN KEY REFERENCES Restaurants(Restaurant_id)
Total_Price	DECIMAL(10,2)	NOT NULL
Order_date	DATETIME	DEFAULT CURRENT TIMESTAMP

Order_Items

Column Name	Data Type	Constraints
Order_Item_id	INT	PRIMARY KEY, AUTO_INCREMENT
Order_id	INT	FOREIGN KEY REFERENCES Orders(Order_id)
Item_id	INT	FOREIGN KEY REFERENCES Menu_Items(Item_id)
Quantity	INT	NOT NULL

Riders

Column Name	Data Type	Constraints
Rider_id	INT	PRIMARY KEY, AUTO_INCREMENT
User_id	INT	FOREIGN KEY REFERENCES Users(User_id)
Name	VARCHAR(255)	NOT NULL
Phone_Number	VARCHAR(50)	FOREIGN KEY REFERENCES Restaurants(Restaurant_id)
Email	DECIMAL(10,2)	UNIQUE NOT NULL
Vehicle_Type	VARCHAR(100)	NOT NULL
Vehicle_Registration	VARCHAR(100)	NOT NULL
Restaurant_id	INT	FOREIGN KEY REFERENCES Restaurants(Restaurant_id)

Payments

Column Name	Data Type	Constraints
Payment_id	INT	PRIMARY KEY, AUTO_INCREMENT
Order_id	INT	FOREIGN KEY REFERENCES Orders(Order_id)
Payment_Method	ENUM	NOT NULL (card/cash/mpesa)
Payment_Status	ENUM	DEFAULT ‘Pending’

Table Scripts

Table Creation Scripts

```
CREATE TABLE Users (
    User_id INT PRIMARY KEY AUTO_INCREMENT,
    Role ENUM('admin', 'manager', 'customer', 'rider') NOT NULL,
    Phone_Number VARCHAR(100) NOT NULL,
    Password VARCHAR(255) NOT NULL
);

-- Create Customers Table
CREATE TABLE Customers (
    Customer_id INT PRIMARY KEY AUTO_INCREMENT,
    Name VARCHAR(255) NOT NULL,
    Phone_Number VARCHAR(100) NOT NULL,
    Email VARCHAR(255) UNIQUE,
    Location VARCHAR(100) NOT NULL
);

-- Create Menu Table
CREATE TABLE Menu (
    Menu_id INT PRIMARY KEY AUTO_INCREMENT,
    Name VARCHAR(100) NOT NULL,
    Description LONGTEXT NOT NULL
);

-- Create Restaurants Table
CREATE TABLE Restaurants (
    Restaurant_id INT PRIMARY KEY AUTO_INCREMENT,
    Menu_id INT,
    Name VARCHAR(255) NOT NULL,
    Phone_Number VARCHAR(100) NOT NULL,
    Email VARCHAR(255) UNIQUE,
    Location VARCHAR(100) NOT NULL,
    FOREIGN KEY (Menu_id) REFERENCES Menu(Menu_id)
);

-- Create Menu_Items Table
CREATE TABLE Menu_Items (
```

```
    Item_id INT PRIMARY KEY AUTO_INCREMENT,
    Menu_id INT,
    Food_Name VARCHAR(100) NOT NULL,
    Food_Category ENUM('starter', 'main', 'dessert', 'beverage') NOT NULL,
    Price DECIMAL(10,2) NOT NULL,
    Quantity INT NOT NULL,
    FOREIGN KEY (Menu_id) REFERENCES Menu(Menu_id)
);

-- Create Orders Table
CREATE TABLE Orders (
    Order_id INT PRIMARY KEY AUTO_INCREMENT,
    Customer_id INT,
    Restaurant_id INT,
    Total_Price DECIMAL(10,2) NOT NULL,
    FOREIGN KEY (Customer_id) REFERENCES Customers(Customer_id),
    FOREIGN KEY (Restaurant_id) REFERENCES Restaurants(Restaurant_id)
);

-- Create Order_Items Table
CREATE TABLE Order_Items (
    Order_Item_id INT PRIMARY KEY AUTO_INCREMENT,
    Order_id INT,
    Item_id INT,
    Quantity INT NOT NULL,
    FOREIGN KEY (Order_id) REFERENCES Orders(Order_id),
    FOREIGN KEY (Item_id) REFERENCES Menu_Items(Item_id)
);

-- Create Riders Table
CREATE TABLE Riders (
    Rider_id INT PRIMARY KEY AUTO_INCREMENT,
    User_id INT,
    Name VARCHAR(255) NOT NULL,
    Phone_Number VARCHAR(15) UNIQUE NOT NULL,
    Email VARCHAR(255) UNIQUE NOT NULL,
    Vehicle_Type VARCHAR(100) NOT NULL,
    Vehicle_Registration VARCHAR(100) NOT NULL,
    Assigned_Restaurant_id INT,
    FOREIGN KEY (User_id) REFERENCES Users(User_id),
```

```
    FOREIGN KEY (Assigned_Restaurant_id) REFERENCES Restaurants(Restaurant_id)
);

-- Create Payments Table
CREATE TABLE Payments (
    Payment_id INT PRIMARY KEY AUTO_INCREMENT,
    Order_id INT,
    Payment_Method ENUM('card', 'cash', 'mpesa') NOT NULL,
    Payment_Status ENUM('Pending', 'Completed') DEFAULT 'Pending',
    FOREIGN KEY (Order_id) REFERENCES Orders(Order_id)
);

-- Create Users Table
CREATE TABLE Users (
    User_id INT PRIMARY KEY AUTO_INCREMENT,
    Role ENUM('admin', 'manager', 'customer', 'rider') NOT NULL,
    Phone_Number VARCHAR(100) NOT NULL,
    Password VARCHAR(255) NOT NULL
);

-- Create Customers Table
CREATE TABLE Customers (
    Customer_id INT PRIMARY KEY AUTO_INCREMENT,
    Name VARCHAR(255) NOT NULL,
    Phone_Number VARCHAR(100) NOT NULL,
    Email VARCHAR(255) UNIQUE,
    Location VARCHAR(100) NOT NULL
);

-- Create Menu Table
CREATE TABLE Menu (
    Menu_id INT PRIMARY KEY AUTO_INCREMENT,
    Name VARCHAR(100) NOT NULL,
    Description LONGTEXT NOT NULL
);

-- Create Restaurants Table
CREATE TABLE Restaurants (
    Restaurant_id INT PRIMARY KEY AUTO_INCREMENT,
    Menu_id INT,
```

```
    Name VARCHAR(255) NOT NULL,
    Phone_Number VARCHAR(100) NOT NULL,
    Email VARCHAR(255) UNIQUE,
    Location VARCHAR(100) NOT NULL,
    FOREIGN KEY (Menu_id) REFERENCES Menu(Menu_id)
);

-- Create Menu_Items Table
CREATE TABLE Menu_Items (
    Item_id INT PRIMARY KEY AUTO_INCREMENT,
    Menu_id INT,
    Food_Name VARCHAR(100) NOT NULL,
    Food_Category ENUM('starter', 'main', 'dessert', 'beverage') NOT NULL,
    Price DECIMAL(10,2) NOT NULL,
    Quantity INT NOT NULL,
    FOREIGN KEY (Menu_id) REFERENCES Menu(Menu_id)
);

-- Create Orders Table
CREATE TABLE Orders (
    Order_id INT PRIMARY KEY AUTO_INCREMENT,
    Customer_id INT,
    Restaurant_id INT,
    Total_Price DECIMAL(10,2) NOT NULL,
    FOREIGN KEY (Customer_id) REFERENCES Customers(Customer_id),
    FOREIGN KEY (Restaurant_id) REFERENCES Restaurants(Restaurant_id)
);

-- Create Order_Items Table
CREATE TABLE Order_Items (
    Order_Item_id INT PRIMARY KEY AUTO_INCREMENT,
    Order_id INT,
    Item_id INT,
    Quantity INT NOT NULL,
    FOREIGN KEY (Order_id) REFERENCES Orders(Order_id),
    FOREIGN KEY (Item_id) REFERENCES Menu_Items(Item_id)
);

-- Create Riders Table
CREATE TABLE Riders (
```

```
    Rider_id INT PRIMARY KEY AUTO_INCREMENT,
    User_id INT,
    Name VARCHAR(255) NOT NULL,
    Phone_Number VARCHAR(15) UNIQUE NOT NULL,
    Email VARCHAR(255) UNIQUE NOT NULL,
    Vehicle_Type VARCHAR(100) NOT NULL,
    Vehicle_Registration VARCHAR(100) NOT NULL,
    Assigned_Restaurant_id INT,
    FOREIGN KEY (User_id) REFERENCES Users(User_id),
    FOREIGN KEY (Assigned_Restaurant_id) REFERENCES Restaurants(Restaurant_id)
);

-- Create Payments Table
CREATE TABLE Payments (
    Payment_id INT PRIMARY KEY AUTO_INCREMENT,
    Order_id INT,
    Payment_Method ENUM('card', 'cash', 'mpesa') NOT NULL,
    Payment_Status ENUM('Pending', 'Completed') DEFAULT 'Pending',
    FOREIGN KEY (Order_id) REFERENCES Orders(Order_id)
);
```

Sample Data Scripts

```
-- Insert Sample Data into Users Table
INSERT INTO Users (Role, Phone_Number, Password) VALUES
('admin', '0723123456', 'adminpassword'),
('manager', '0712345678', 'managerpassword'),
('customer', '0703456789', 'customerpassword'),
('rider', '0798765432', 'riderpassword');

-- Insert Sample Data into Customers Table
INSERT INTO Customers (Name, Phone_Number, Email, Location) VALUES
('John Mwangi', '0703456789', 'johnmwangi@email.com', 'Nairobi, Kenyatta Avenue'),
('Mary Wambui', '0721234567', 'marywambui@email.com', 'Nakuru, Kenyatta Street'),
('Peter Njoroge', '0732123456', 'peternjoroge@email.com', 'Mombasa, Moi Avenue');

-- Insert Sample Data into Menu Table
INSERT INTO Menu (Name, Description) VALUES
```

```
('Kenyan Delights', 'A variety of local Kenyan dishes including Nyama Choma, Ugali, Sukuma, and more.'),
('Fast Food', 'A menu of quick and tasty options including Burgers, Fries, and Soft Drinks.');

-- Insert Sample Data into Restaurants Table
INSERT INTO Restaurants (Menu_id, Name, Phone_Number, Email, Location) VALUES
(1, 'Mama Njoki's Kitchen', '0701234567', 'mamanjoki@email.com', 'Nairobi, Kenyatta Avenue'),
(2, 'Quick Bites', '0731234567', 'quickbites@email.com', 'Mombasa, Moi Avenue');

-- Insert Sample Data into Menu_Items Table (Corrected)
INSERT INTO Menu_Items (Menu_id, Food_Name, Food_Category, Price, Quantity) VALUES
(1, 'Nyama Choma', 'main', 800.00, 50),
(1, 'Ugali', 'main', 100.00, 100),  -- 'main' is appropriate for Ugali
(1, 'Sukuma Wiki', 'main', 50.00, 100),  -- 'main' is appropriate for Sukuma Wiki
(2, 'Cheese Burger', 'main', 500.00, 30),
(2, 'French Fries', 'starter', 200.00, 50),  -- Changed to 'starter'
(2, 'Coca Cola', 'beverage', 150.00, 100);  -- 'beverage' is appropriate for Coca Cola


-- Insert Sample Data into Orders Table
INSERT INTO Orders (Customer_id, Restaurant_id, Total_Price) VALUES
(1, 1, 1000.00),
(2, 2, 850.00),
(3, 1, 950.00);

-- Insert Sample Data into Order_Items Table
INSERT INTO Order_Items (Order_id, Item_id, Quantity) VALUES
(1, 13, 1), -- Nyama Choma
(1, 14, 2), -- Ugali
(2, 15, 1), -- Sukuma
(2, 16, 1), -- Cheese Burger
(2, 17, 2), -- French Fries
(3, 18, 1), -- Nyama Choma
(3, 13, 1); -- Coca Cola



-- Insert Sample Data into Riders Table
```

```
INSERT INTO Riders (User_id, Name, Phone_Number, Email, Vehicle_Type,
Vehicle_Registration, Assigned_Restaurant_id) VALUES
(4, 'James Kariuki', '0798765432', 'jameskariuki@email.com', 'Motorbike', 'KBD123L', 1),
(4, 'Wanjiru Mwangi', '0787654321', 'wanjirumwangi@email.com', 'Car', 'KBB234M', 2);

-- Insert Sample Data into Payments Table
INSERT INTO Payments (Order_id, Payment_Method, Payment_Status) VALUES
(1, 'cash', 'Completed'),
(2, 'mpesa', 'Pending'),
(3, 'card', 'Completed');
```

Triggers

```
DELIMITER //
CREATE TRIGGER update_menu_quantity
AFTER INSERT ON Order_Items
FOR EACH ROW
BEGIN
  UPDATE Menu_Items
  SET Quantity = Quantity - NEW.Quantity
  WHERE Item_id = NEW.Item_id;
END;
//
DELIMITER ;
```

The update_menu_quantity trigger is used to update the quantity in the menu items to update the quantity left.

```
DELIMITER $$

CREATE TRIGGER CalculateTotalPrice
AFTER INSERT ON Order_Items
FOR EACH ROW
BEGIN
    DECLARE total_price DECIMAL(10, 2);
    SELECT SUM(Menu_Items.Price * Order_Items.Quantity)
    INTO total_price
```

```
    FROM Order_Items
    JOIN Menu_Items ON Order_Items.Item_id = Menu_Items.Item_id
    WHERE Order_Items.Order_id = NEW.Order_id;

    UPDATE Orders
    SET Total_Price = total_price
    WHERE Order_id = NEW.Order_id;
END$$

DELIMITER ;
```

The CalculateTotalPrice trigger is used to compute the total price of an order by multiplying the quantity x the price of a menu item

```
DELIMITER $$
CREATE TRIGGER UpdatePaymentStatus
AFTER INSERT ON Payments
FOR EACH ROW
BEGIN
    IF NEW.Payment_Status = 'Completed' THEN
        UPDATE Orders
        SET Total_Price = 0
        WHERE Order_id = NEW.Order_id;
    END IF;
END$$
DELIMITER ;
```

The UpdatePaymentStatus trigger is used to clear the balance of an order if the order_id is added into the Payments table as completed.

Views

1. CustomerOrders

```
CREATE VIEW CustomerOrders AS
SELECT o.Order_id, o.Total_Price, c.Name, c.Email
FROM Orders o
JOIN Customers c ON o.Customer_id = c.Customer_id;
```

The CustomerOrders view returns the order_id and the total price payable by the customer for that order.

2. Restaurant Orders shows the customer name

```
CREATE VIEW RestaurantOrdersWithCustomers AS
SELECT
    o.Order_id,
    r.Name AS NameRes,
    r.Location,
    c.Name,
    o.Total_Price
FROM
    Orders o
JOIN
    Restaurants r ON o.Restaurant_id = r.Restaurant_id
JOIN
    Customers c ON o.Customer_id = c.Customer_id;
```

This view returns the Restaurant view of customers who have made orders.

3. Rider details including the orders delivered.

```
CREATE VIEW RiderDeliveries AS
SELECT
    r.Rider_id,
```

```
    r.Name AS NameRider,
    r.Phone_Number AS Phone,
    r.Vehicle_Type,
    r.Vehicle_Registration,
    o.Order_id,
    c.Name AS Name,
    c.Phone_Number AS CustomerPhone,
    c.Location AS CustomerLocation,
    res.Name AS RestaurantName,
    res.Location AS RestaurantLocation,
    o.Total_Price AS OrderTotal
FROM
    Riders r
JOIN
    Orders o ON r.Assigned_Restaurant_id = o.Restaurant_id
JOIN
    Customers c ON o.Customer_id = c.Customer_id
JOIN
    Restaurants res ON o.Restaurant_id = res.Restaurant_id;
```

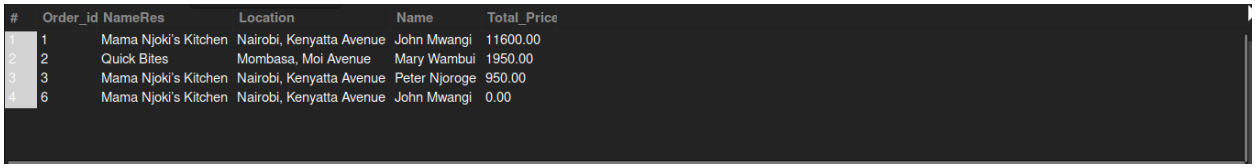
The RiderDeliveries view returns a rider's details and the deliveries they have made.

Testing & Validation

- The **RiderDeliveries** view discussed above produces the results below:

#	Rider_id	NameRider	Phone	Vehicle_Type	Vehicle_Registratio	Order_id	Name	CustomerPhon	CustomerLocation	RestaurantName	RestaurantLocation	Order
1	1	James Kariuki	0798765432	Motorbike	KBD123L	1	John Mwangi	0703456789	Nairobi, Kenyatta Avenue	Mama Njoki's Kitchen	Nairobi, Kenyatta Avenue	11600.
2	1	James Kariuki	0798765432	Motorbike	KBD123L	3	Peter Njoroge	0732123456	Mombasa, Moi Avenue	Mama Njoki's Kitchen	Nairobi, Kenyatta Avenue	950.00
3	1	James Kariuki	0798765432	Motorbike	KBD123L	6	John Mwangi	0703456789	Nairobi, Kenyatta Avenue	Mama Njoki's Kitchen	Nairobi, Kenyatta Avenue	0.00
4	2	Wanjiru Mwangi	0787654321	Car	KBB234M	2	Mary Wambui	0721234567	Nakuru, Kenyatta Street	Quick Bites	Mombasa, Moi Avenue	1950.0
5	4	Wanjiru Mwangi	0757876614	Car	KBB234M	1	John Mwangi	0703456789	Nairobi, Kenyatta Avenue	Mama Njoki's Kitchen	Nairobi, Kenyatta Avenue	11600.
6	4	Wanjiru Mwangi	0757876614	Car	KBB234M	3	Peter Njoroge	0732123456	Mombasa, Moi Avenue	Mama Njoki's Kitchen	Nairobi, Kenyatta Avenue	950.00
7	4	Wanjiru Mwangi	0757876614	Car	KBB234M	6	John Mwangi	0703456789	Nairobi, Kenyatta Avenue	Mama Njoki's Kitchen	Nairobi, Kenyatta Avenue	0.00

- The RestaurantOrdersWithCustomers view discussed above produces the output below:



#	Order_id	NameRes	Location	Name	Total_Price
1	1	Mama Njoki's Kitchen	Nairobi, Kenyatta Avenue	John Mwangi	11600.00
2	2	Quick Bites	Mombasa, Moi Avenue	Mary Wambui	1950.00
3	3	Mama Njoki's Kitchen	Nairobi, Kenyatta Avenue	Peter Njoroge	950.00
4	6	Mama Njoki's Kitchen	Nairobi, Kenyatta Avenue	John Mwangi	0.00

Conclusion and Future Enhancements

Summary

This database project was designed to manage restaurant orders efficiently, connecting customers, restaurants, and delivery riders. The database handles key functionalities such as storing and querying customer information, order details, restaurant data, and rider assignments. Key features include:

1. Structured and relational tables for seamless data integration.
2. Views for better reporting, such as orders by customers, restaurant orders, and rider deliveries.
3. Use of triggers to automate processes like creating orders and updating payment statuses.

Future Enhancements

1. **Enhanced Customer Features**

Add a feedback or review system for customers to rate their experience.

Introduce loyalty points tracking for frequent customers.

2. **Security Enhancements**

Implement auditing and logging for actions taken in the database.

3. **Scalability**

Use a distributed database system for load balancing.

Appendix

Glossary

- **SQL (Structured Query Language):** A programming language for managing relational databases.
- **Primary Key:** A unique identifier for each record in a database table.
- **Foreign Key:** A column that establishes a relationship between two tables.
- **Trigger:** A database object that executes a predefined action when a specified event occurs.
- **View:** A virtual table created based on a query to simplify complex queries.
- **Relational Database:** A database structured to recognize relationships between stored data elements.

References

1. **Tools Used:**
    - MySQL Workbench: For designing and managing the database schema.
    - MariaDB: As the database server for executing queries and managing data.
2. **Resources and Tutorials:**
    - TutorialsPoint: SQL tutorials for creating views and triggers.
    - W3Schools: Database basics and glossary definitions.