-- Table: SignupUser

CREATE TABLE SignupUser (

user\_id INT AUTO\_INCREMENT PRIMARY KEY,

email VARCHAR(100) UNIQUE NOT NULL,

first\_name VARCHAR(50) NOT NULL,

last\_name VARCHAR(50) NOT NULL,

phone VARCHAR(15),

password VARCHAR(64) NOT NULL,

street VARCHAR(100),

city VARCHAR(100) NOT NULL,

zip\_code VARCHAR(20) NOT NULL,

reward\_points INT DEFAULT 0, -- Points earned by the user

otp INT,

otp\_expiration DATETIME DEFAULT NULL, -- Expiration time for OTP

is\_verified BOOLEAN DEFAULT FALSE,

INDEX (email)

);

-- Table: GuestUsers

CREATE TABLE GuestUsers (

guest\_id INT AUTO\_INCREMENT PRIMARY KEY,

first\_name VARCHAR(50) NOT NULL,

last\_name VARCHAR(50) NOT NULL,

email VARCHAR(100) UNIQUE NOT NULL,

phone VARCHAR(15),

street VARCHAR(100),

city VARCHAR(100) NOT NULL,

zip\_code VARCHAR(20) NOT NULL

);

-- Table: FoodItems(Menu)

CREATE TABLE FoodItems (

food\_item\_id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100) NOT NULL,

description TEXT,

price DECIMAL(10, 2) NOT NULL CHECK (price > 0),

image\_url VARCHAR(255) NOT NULL,

category VARCHAR(50) NOT NULL

);

-- Table: Orders

CREATE TABLE Orders (

order\_id INT AUTO\_INCREMENT PRIMARY KEY,

customer\_type ENUM('User', 'Guest') NOT NULL,

user\_id INT DEFAULT NULL, -- User ID set to NULL if customer\_type is 'Guest'

guest\_id INT DEFAULT NULL, -- Guest ID set to NULL if customer\_type is 'User'

food\_items JSON NOT NULL,

-- these columns are in the json file.

-- "food\_item\_name"

-- "quantity"

-- "add\_extra"

-- “price\_of\_extra”

-- "ingredients\_to\_remove"

total\_amount DECIMAL(10, 2) DEFAULT 0.00,

order\_timestamp DATETIME NOT NULL,

delivery\_method ENUM('Pick Up', 'Curbside', 'Delivery') NOT NULL,

FOREIGN KEY (user\_id) REFERENCES SignupUser(user\_id) ON DELETE CASCADE,

FOREIGN KEY (guest\_id) REFERENCES GuestUsers(guest\_id) ON DELETE CASCADE

);

-- Table: Feedback

CREATE TABLE Feedback (

feedback\_id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT DEFAULT NULL, -- User ID can be set to NULL if feedback is from a guest

food\_item\_id INT NOT NULL,

order\_id INT NOT NULL,

rating ENUM('Poor', 'Average', 'Neutral', 'Good', 'Excellent') NOT NULL,

feedback\_date DATE NOT NULL,

FOREIGN KEY (user\_id) REFERENCES SignupUser(user\_id) ON DELETE SET NULL,

FOREIGN KEY (food\_item\_id) REFERENCES FoodItems(food\_item\_id) ON DELETE CASCADE,

FOREIGN KEY (order\_id) REFERENCES Orders(order\_id) ON DELETE CASCADE

);

-- SQL Script to Create Payments Table

CREATE TABLE Payments (

payment\_id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT DEFAULT NULL, -- For User payments

guest\_id INT DEFAULT NULL, -- For Guest payments

customer\_type ENUM('User', 'Guest') NOT NULL,

total\_amount DECIMAL(10, 2) NOT NULL CHECK (total\_amount > 0), -- Total payment amount must be greater than zero

clover\_id VARCHAR(255) NOT NULL, -- Unique identifier for Clover payment

payment\_timestamp DATETIME NOT NULL DEFAULT CURRENT\_TIMESTAMP,

order\_id INT NOT NULL, -- Foreign key reference to Orders table

FOREIGN KEY (user\_id) REFERENCES SignupUser(user\_id) ON DELETE CASCADE,

FOREIGN KEY (guest\_id) REFERENCES GuestUsers(guest\_id) ON DELETE CASCADE,

FOREIGN KEY (order\_id) REFERENCES Orders(order\_id) ON DELETE CASCADE

);

-- Trigger to automatically set OTP expiration time to current time + 5 minutes when OTP is generated

DELIMITER //

CREATE TRIGGER set\_otp\_expiration

BEFORE INSERT ON SignupUser

FOR EACH ROW

BEGIN

SET NEW.otp\_expiration = NOW() + INTERVAL 5 MINUTE; -- Set OTP expiration to be valid for only five minutes

END;

//

DELIMITER ;

-- Procedure to calculate reward points based on total amount spent by the user.

DELIMITER //

CREATE PROCEDURE CalculateRewardPoints(IN userId INT)

BEGIN

DECLARE totalSpent DECIMAL(10,2);

DECLARE rewardPoints INT;

SELECT SUM(total\_amount)

INTO totalSpent

FROM Orders

WHERE user\_id = userId;

SET rewardPoints = FLOOR(totalSpent); -- Earns one point for each dollar spent

UPDATE SignupUser

SET reward\_points = reward\_points + rewardPoints

WHERE user\_id = userId;

END;

//

DELIMITER ;

DELIMITER //

CREATE TRIGGER validate\_payment\_amount

BEFORE INSERT ON Payments

FOR EACH ROW

BEGIN

DECLARE orderTotal DECIMAL(10, 2);

-- Fetch the total amount from the Orders table based on customer type and ID

IF NEW.customer\_type = 'User' THEN

SELECT SUM(total\_amount)

INTO orderTotal

FROM Orders

WHERE user\_id = NEW.user\_id;

ELSEIF NEW.customer\_type = 'Guest' THEN

SELECT SUM(total\_amount)

INTO orderTotal

FROM Orders

WHERE guest\_id = NEW.guest\_id;

END IF;

-- Compare the payment total amount with the order total amount

IF NEW.total\_amount != orderTotal THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Payment total amount does not match the order total amount';

END IF;

END;

//

DELIMITER ;