DATABASE PROJECT

ONLINE FOOD ORDERING SYSTEM

SQL CODE:

```
CREATE TABLE Users (
  user_id INT PRIMARY KEY NOT NULL,
  username VARCHAR(100) UNIQUE,
  email VARCHAR(100) UNIQUE,
  password VARCHAR(100),
  first_name VARCHAR(100),
  last_name VARCHAR(100),
  phone_number VARCHAR(100),
  user_address VARCHAR(100)
);

CREATE TABLE Restaurants (
  restaurant_id INT PRIMARY KEY NOT NULL,
  name VARCHAR(100),
  description TEXT,
  location VARCHAR(100),
  rating DECIMAL(2,1),
  delivery_fee DECIMAL(10,2),
  minimum_order DECIMAL(10,2),
  user_id INT REFERENCES Users(user_id)
);
```

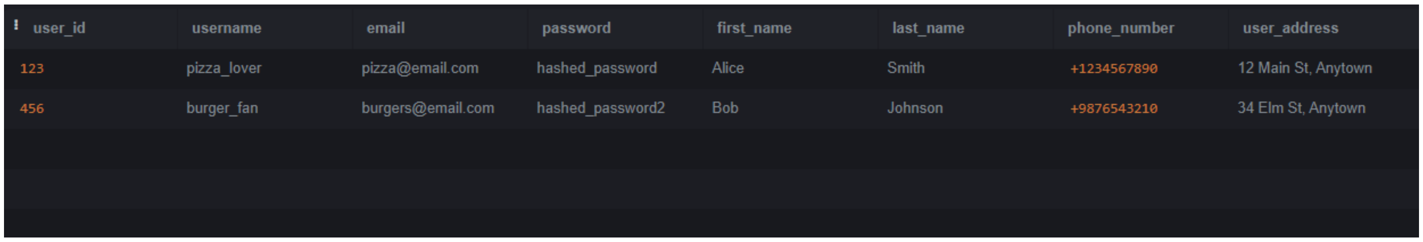
```
CREATE TABLE Categories (
  category_id INT PRIMARY KEY NOT NULL,
  name VARCHAR(100) UNIQUE
);

CREATE TABLE Foods (
  food_id INT PRIMARY KEY NOT NULL,
  name VARCHAR(100),
  description TEXT,
  price DECIMAL(10,2),
  category_id INT REFERENCES Categories(category_id),
  restaurant_id INT REFERENCES Restaurants(restaurant_id),
  is_available VARCHAR(100) DEFAULT 'yes' CHECK(is_available IN ('yes', 'no'))
);

CREATE TABLE Orders (
  order_id INT PRIMARY KEY NOT NULL,
  user_id INT REFERENCES Users(user_id),
  restaurant_id INT REFERENCES Restaurants(restaurant_id),
  order_date DATETIME,
  status VARCHAR(100) CHECK(status IN ('pending', 'confirmed', 'cancelled', 'delivered')),
  delivery_address VARCHAR(100),
  payment_method VARCHAR(100) CHECK(payment_method IN ('cash_on_delivery', 'online_transfer')),
  payment_status VARCHAR(100) CHECK(payment_status IN ('pending', 'paid'))
);

CREATE TABLE Order_Items (
  order_item_id INT PRIMARY KEY NOT NULL,
  order_id INT REFERENCES Orders(order_id),
  food_id INT REFERENCES Foods(food_id),
  quantity INT,
  price DECIMAL(10,2),
);

CREATE TABLE Reviews (
  review_id INT PRIMARY KEY NOT NULL,
  user_id INT REFERENCES Users(user_id),
  restaurant_id INT REFERENCES Restaurants(restaurant_id),
  order_id INT REFERENCES Orders(order_id),
  rating INT,
  comments TEXT,
);

CREATE TABLE Coupons (
  coupon_id INT PRIMARY KEY NOT NULL,
  code VARCHAR(100) UNIQUE,
  discount_type VARCHAR(100) CHECK(discount_type IN ('percentage', 'fixed_amount')),
  discount_value DECIMAL(10,2),
  minimum_order DECIMAL(10,2),
  expiry_date DATETIME
);

CREATE TABLE User_Coupons (
  user_coupon_id INT PRIMARY KEY NOT NULL,
  user_id INT REFERENCES Users(user_id),
  coupon_id INT REFERENCES Coupons(coupon_id),
  is_used VARCHAR(100) DEFAULT 'no' CHECK(is_used IN ('yes', 'no'))
);
```

```sql
CREATE TABLE Order_Payments (
  order_payment_id INT PRIMARY KEY NOT NULL,
  order_id INT REFERENCES Orders(order_id),
  payment_gateway VARCHAR(100),
  transaction_id VARCHAR(100)
);

INSERT INTO Users (user_id, username, email, password, first_name, last_name, phone_number, user_address)
VALUES (123, 'pizza_lover', 'pizza@email.com', 'hashed_password', 'Alice', 'Smith', '+1234567890', '12 Main St, Anytown');
INSERT INTO Users (user_id, username, email, password, first_name, last_name, phone_number, user_address)
VALUES (456, 'burger_fan', 'burgers@email.com', 'hashed_password2', 'Bob', 'Johnson', '+9876543210', '34 Elm St, Anytown');

INSERT INTO Restaurants (restaurant_id, name, description, location, rating, delivery_fee, minimum_order)
VALUES (1000, 'Pizza Palace', 'Delicious pizzas made with fresh ingredients', '5th Avenue', 4.2, 2.50, 15.00);
INSERT INTO Restaurants (restaurant_id, name, description, location, rating, delivery_fee, minimum_order)
VALUES (2000, 'Burger Barn', 'Home of the juiciest burgers in town', '1st Street', 4.8, 1.00, 10.00);

INSERT INTO Categories (category_id, name)
VALUES (7, 'Pizzas');
INSERT INTO Categories (category_id, name)
VALUES (3, 'Burgers');

INSERT INTO Foods (food_id, name, description, price, category_id, restaurant_id, is_available)
```

```sql
VALUES (147, 'Margherita Pizza', 'Classic pizza with tomato sauce and mozzarella cheese', 2000, 7, 1000,
'yes');

INSERT INTO Foods (food_id, name, description, price, category_id, restaurant_id, is_available)

VALUES (579, 'Pepperoni Pizza', 'Pizza topped with pepperoni slices', 2100, 7, 1000, 'yes');

INSERT INTO Foods (food_id, name, description, price, category_id, restaurant_id, is_available)

VALUES (358, 'Cheeseburger', 'Classic burger with cheese, lettuce, tomato, and onion', 500, 3, 2000, 'yes');

INSERT INTO Foods (food_id, name, description, price, category_id, restaurant_id, is_available)

VALUES (169, 'Bacon Cheeseburger', 'Cheeseburger with added bacon', 700, 3, 2000, 'yes');


INSERT INTO Orders (order_id, user_id, restaurant_id, order_date, status, delivery_address,
payment_method, payment_status)

VALUES (753698, 123, 1000, 7-5-24, 'pending', '12 Main St, Anytown', 'cash_on_delivery', 'pending');

INSERT INTO Orders (order_id, user_id, restaurant_id, order_date, status, delivery_address,
payment_method, payment_status)

VALUES (148527, 456, 2000, 7-5-24, 'pending', '34 Elm St, Anytown', 'online_transfer', 'pending');


INSERT INTO Order_Items (order_item_id, order_id, food_id, quantity, price)

VALUES (4956, 753698, 147, 1, 2000);

INSERT INTO Order_Items (order_item_id, order_id, food_id, quantity, price)

VALUES (6578, 148527, 358, 2, 500 * 2);


INSERT INTO Reviews (review_id, user_id, restaurant_id, order_id, rating, comments)

VALUES (975147, 123, 1000, 753698, 9, 'it was really delicious');


INSERT INTO Coupons (coupon_id, code, discount_type, discount_value, minimum_order, expiry_date)

VALUES (357951, 'rs500off', 'fixed_amount', 500, 2000, 7-6-24);


INSERT INTO User_Coupons (user_coupon_id, user_id, coupon_id, is_used)

VALUES (157078, 123, 357951, 'yes');
```

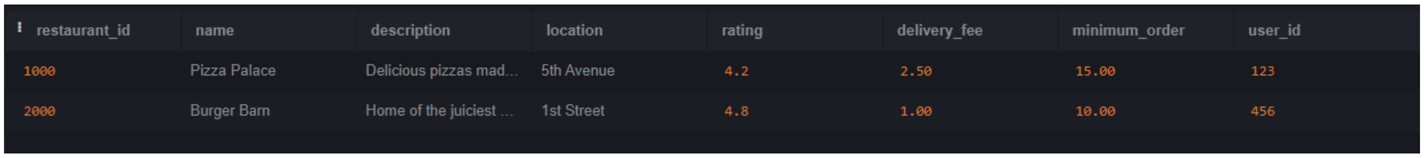
```
INSERT INTO Order_Payments (order_item_id, order_id, payment_gateway, transaction_id)

VALUES (487321, 753698, 'hand_to_hand', 't012');

INSERT INTO Order_Payments (order_item_id, order_id, payment_gateway, transaction_id)

VALUES (987820, 148527, 'jazzcash', 't047');
```

USER



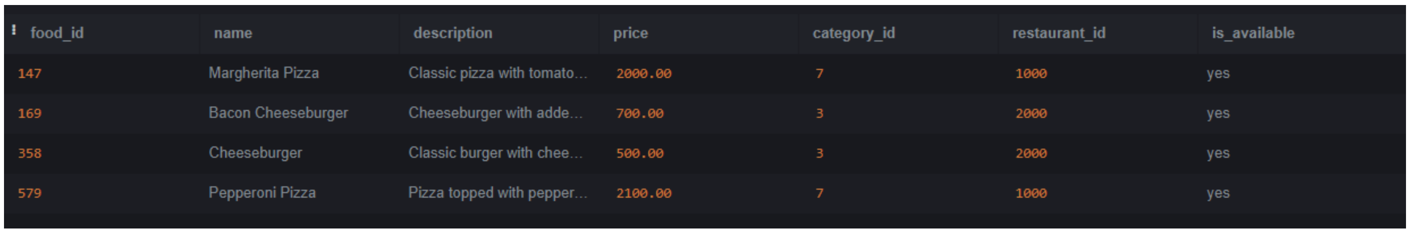
user_id	username	email	password	first_name	last_name	phone_number	user_address
123	pizza_lover	pizza@email.com	hashed_password	Alice	Smith	+1234567890	12 Main St, Anytown
456	burger_fan	burgers@email.com	hashed_password2	Bob	Johnson	+9876543210	34 Elm St, Anytown

RESTAURANT



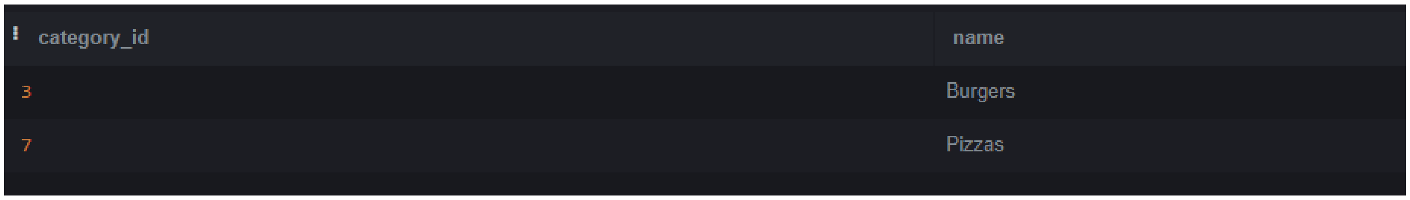
restaurant_id	name	description	location	rating	delivery_fee	minimum_order	user_id
1000	Pizza Palace	Delicious pizzas mad...	5th Avenue	4.2	2.50	15.00	123
2000	Burger Barn	Home of the juiciest ...	1st Street	4.8	1.00	10.00	456

FOODS



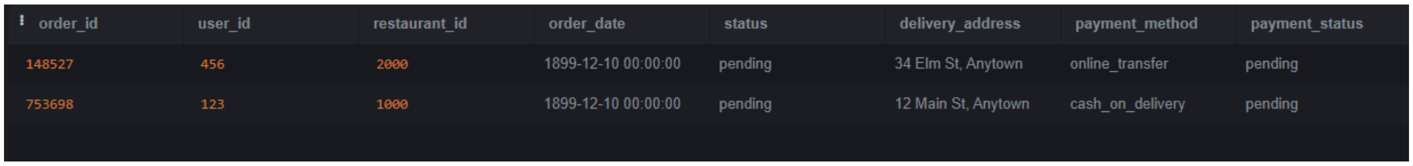
food_id	name	description	price	category_id	restaurant_id	is_available
147	Margherita Pizza	Classic pizza with tomato...	2000.00	7	1000	yes
169	Bacon Cheeseburger	Cheeseburger with adde...	700.00	3	2000	yes
358	Cheeseburger	Classic burger with chee...	500.00	3	2000	yes
579	Pepperoni Pizza	Pizza topped with pepper...	2100.00	7	1000	yes

CATEGORIES



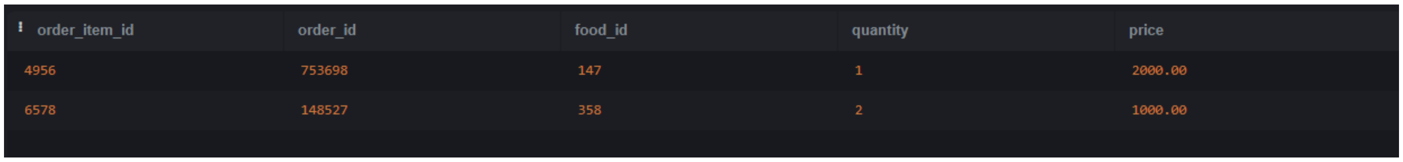
category_id	name
3	Burgers
7	Pizzas

ORDERS



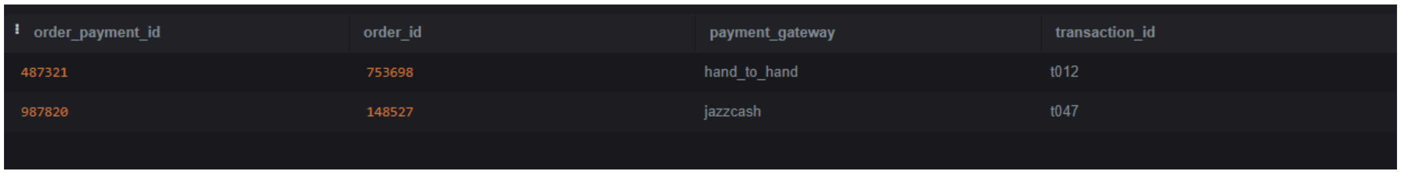
order_id	user_id	restaurant_id	order_date	status	delivery_address	payment_method	payment_status
148527	456	2000	1899-12-10 00:00:00	pending	34 Elm St, Anytown	online_transfer	pending
753698	123	1000	1899-12-10 00:00:00	pending	12 Main St, Anytown	cash_on_delivery	pending

ORDER_ITEMS



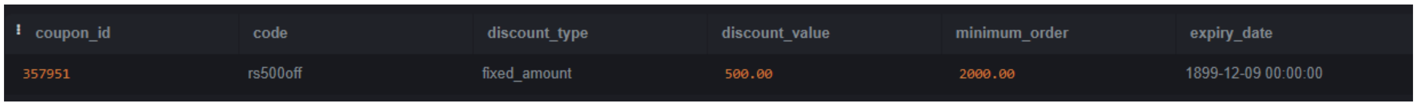
order_item_id	order_id	food_id	quantity	price
4956	753698	147	1	2000.00
6578	148527	358	2	1000.00

ORDER_PAYMENTS



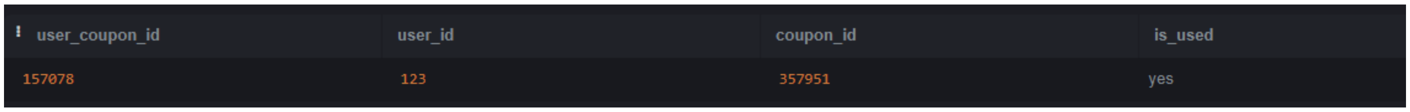
order_payment_id	order_id	payment_gateway	transaction_id
487321	753698	hand_to_hand	t012
987820	148527	jazzcash	t047

COUPONS



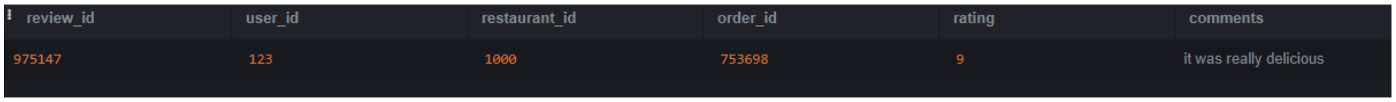
coupon_id	code	discount_type	discount_value	minimum_order	expiry_date
357951	rs500off	fixed_amount	500.00	2000.00	1899-12-09 00:00:00

USER_COUPONS



user_coupon_id	user_id	coupon_id	is_used
157078	123	357951	yes

REVIEWS



review_id	user_id	restaurant_id	order_id	rating	comments
975147	123	1000	753698	9	it was really delicious

OVERVIEW:

Users Table

- user_id (INT, Primary Key, Not Null): Unique identifier for each user.
- username (VARCHAR, Unique): User's chosen username for login.
- email (VARCHAR, Unique): User's email address for communication and login.
- password (VARCHAR): User's hashed password for authentication.
- first_name (VARCHAR): User's first name.
- last_name (VARCHAR): User's last name.
- phone_number (VARCHAR): User's contact phone number.
- user_address (VARCHAR): User's residential or mailing address.

Restaurants Table

- restaurant_id (INT, Primary Key, Not Null): Unique identifier for each restaurant.
- name (VARCHAR): Name of the restaurant.
- description (TEXT): Description or details about the restaurant.
- location (VARCHAR): Location of the restaurant.
- rating (DECIMAL(2,1)): Average rating of the restaurant.
- delivery_fee (DECIMAL(10,2)): Delivery fee charged by the restaurant.
- minimum_order (DECIMAL(10,2)): Minimum order amount required for delivery.
- user_id (INT, Foreign Key - Users): ID of the user who owns or manages the restaurant.

Categories Table

- category_id (INT, Primary Key, Not Null): Unique identifier for each food category.

- name (VARCHAR, Unique): Name of the food category.

Foods Table

- food_id (INT, Primary Key, Not Null): Unique identifier for each food item.
- name (VARCHAR): Name of the food item.
- description (TEXT): Description or details about the food item.
- price (DECIMAL(10,2)): Price of the food item.
- category_id (INT, Foreign Key - Categories): ID of the category to which the food item belongs.
- restaurant_id (INT, Foreign Key - Restaurants): ID of the restaurant offering the food item.
- is_available (VARCHAR, Default 'yes', Check Constraint): Availability status of the food item.

Orders Table

- order_id (INT, Primary Key, Not Null): Unique identifier for each order.
- user_id (INT, Foreign Key - Users): ID of the user who placed the order.
- restaurant_id (INT, Foreign Key - Restaurants): ID of the restaurant from which the order was placed.
- order_date (DATETIME): Date and time when the order was placed.
- status (VARCHAR, Check Constraint): Status of the order (pending, confirmed, cancelled, delivered).
- total_price (DECIMAL(10,2)): Total price of the order.
- delivery_address (VARCHAR): Address where the order is to be delivered.
- payment_method (VARCHAR, Check Constraint): Payment method chosen for the order (cash_on_delivery, online_transfer).
- payment_status (VARCHAR, Check Constraint): Payment status of the order (pending, paid).

Order_Items Table

- order_item_id (INT, Primary Key, Not Null): Unique identifier for each order item.
- order_id (INT, Foreign Key - Orders): ID of the order to which the item belongs.
- food_id (INT, Foreign Key - Foods): ID of the food item included in the order.
- quantity (INT): Quantity of the food item ordered.
- price (DECIMAL(10,2)): Price of the order item.

Reviews Table

- review_id (INT, Primary Key, Not Null): Unique identifier for each review.
- user_id (INT, Foreign Key - Users): ID of the user who submitted the review.
- restaurant_id (INT, Foreign Key - Restaurants): ID of the restaurant being reviewed.
- order_id (INT, Foreign Key - Orders): ID of the order associated with the review.
- rating (INT): Rating given by the user.
- comments (TEXT): Comments or feedback provided by the user.

Coupons Table

- coupon_id (INT, Primary Key, Not Null): Unique identifier for each coupon.
- code (VARCHAR, Unique): Coupon code for redemption.
- discount_type (VARCHAR, Check Constraint): Type of discount offered (percentage, fixed_amount).
- discount_value (DECIMAL(10,2)): Value of the discount offered.
- minimum_order (DECIMAL(10,2)): Minimum order amount required for coupon redemption.
- expiry_date (DATE): Expiry date of the coupon.

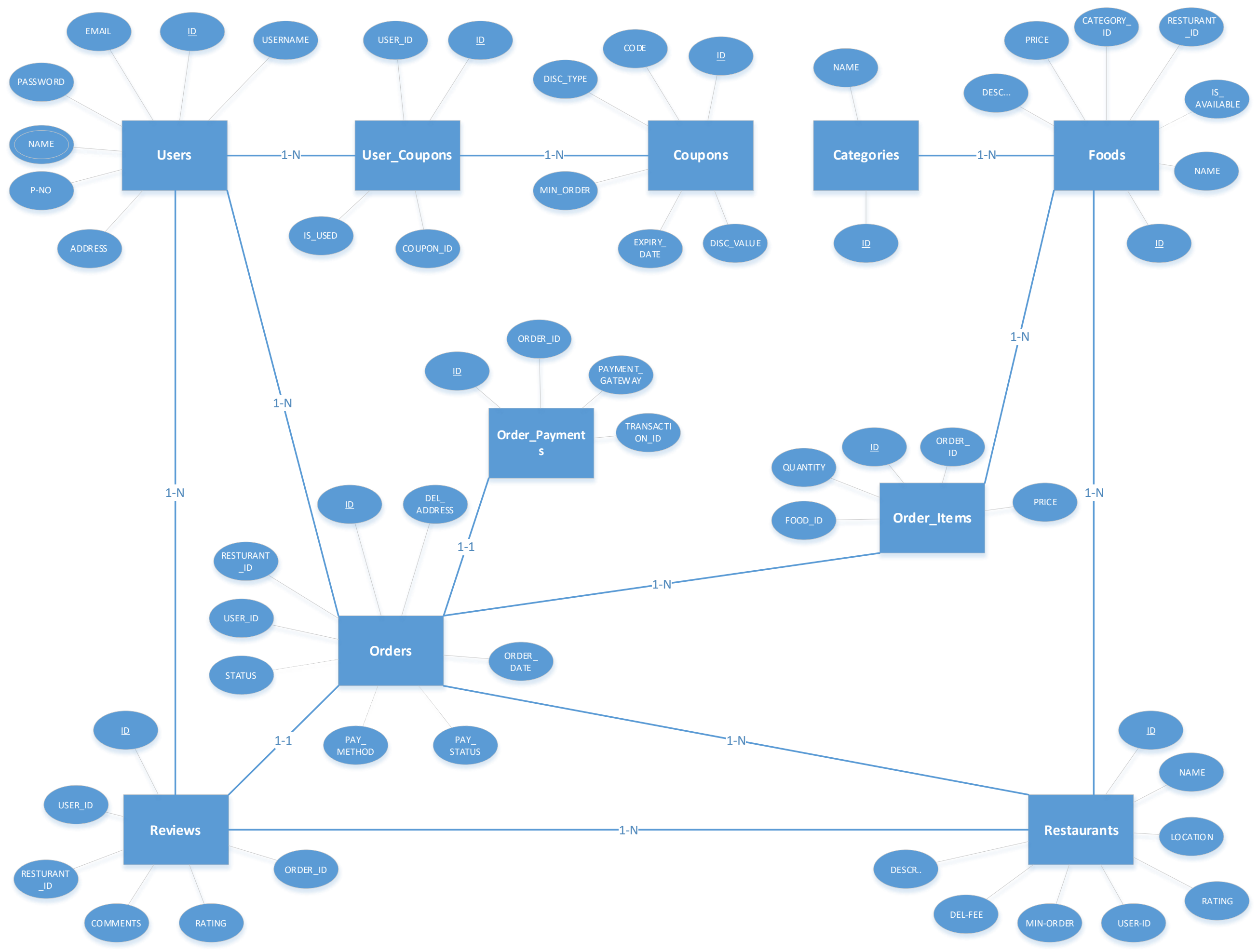
User_Coupons Table

- user_coupon_id (INT, Primary Key, Not Null): Unique identifier for each user coupon.
- user_id (INT, Foreign Key - Users): ID of the user who owns the coupon.
- coupon_id (INT, Foreign Key - Coupons): ID of the coupon.
- is_used (VARCHAR, Default 'no', Check Constraint): Usage status of the coupon.

Order_Payments Table

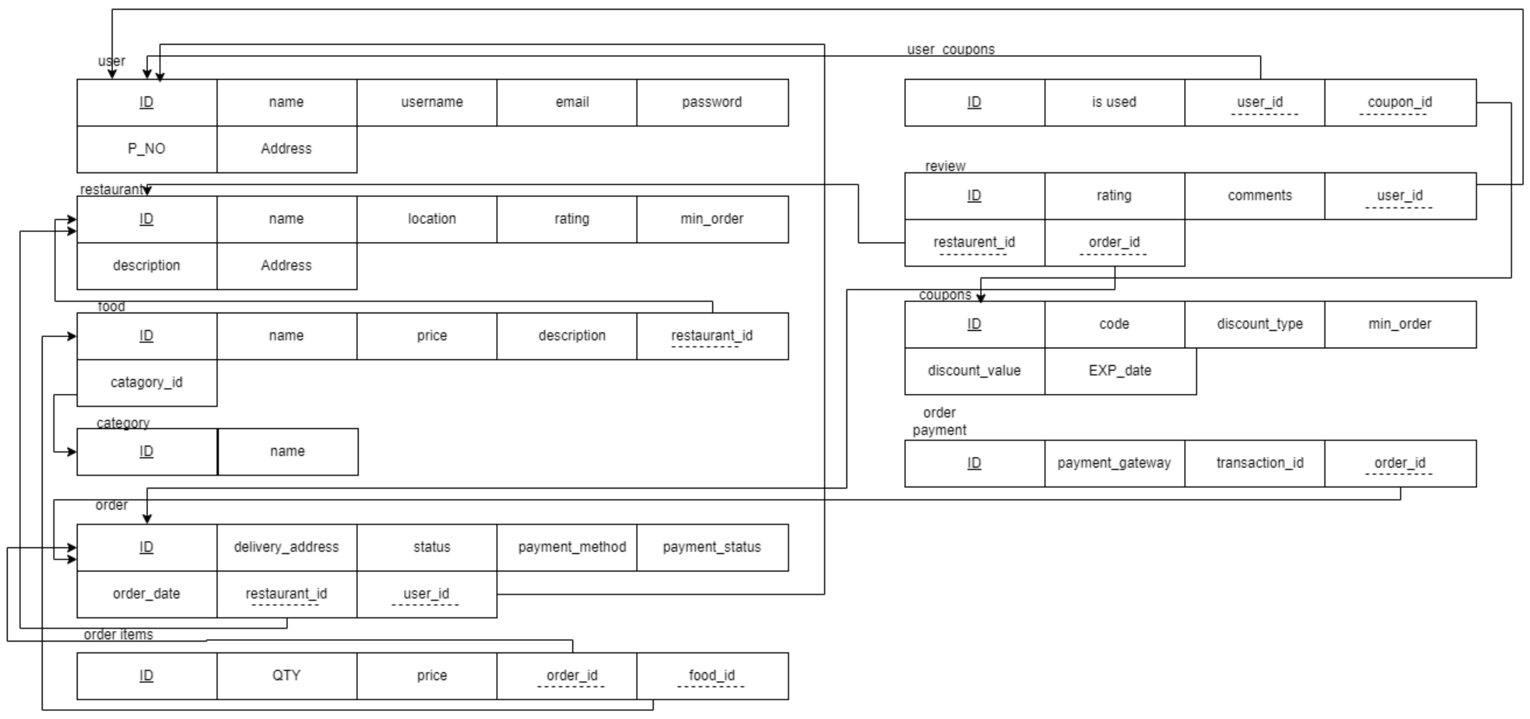
- order_payment_id (INT, Primary Key, Not Null): Unique identifier for each order payment.
- order_id (INT, Foreign Key - Orders): ID of the order for which payment is made.
- payment_gateway (VARCHAR): Payment gateway used for the transaction.
- transaction_id (VARCHAR): Unique identifier for the payment transaction.

ERD:



RELATIONAL SCHEMA:

Normalization:



1st normal form:

1st NF stipulates three main rules:

Atomic Values: Each table cell should contain a single atomic value. You don't see any columns with multiple values stored together (e.g., comma-separated lists).

Unique Identifiers: Each table should have a primary key that uniquely identifies each row. All the tables have an INT PRIMARY KEY which satisfies this requirement.

No Duplication: There shouldn't be any duplicate rows within a table. The combination of the primary key and the UNIQUE constraints on specific columns (like username, email, and code in your case) ensures no duplicate rows exist.

Therefore, based on these rules, all these tables are already in 1st NF.

2nd normal form:

Identify the Issue:

2NF builds upon 1NF by eliminating partial dependencies. A partial dependency occurs when a non-key attribute (attribute not part of the primary key) depends on only a part of the primary key.

In your tables, the user_id is part of the primary key in both the Restaurants and Orders tables. However, some columns in these tables (e.g., delivery_fee and payment_status in Orders) depend only on the user_id, not the entire primary key (which might also include restaurant_id or order_id). This creates partial dependencies.

Modified Tables in 2NF:

```
-- New User_Orders table
CREATE TABLE User_Orders (
  user_order_id INT PRIMARY KEY NOT NULL,
  user_id INT FOREIGN KEY REFERENCES Users(user_id),
  delivery_fee DECIMAL(10,2),
  payment_method VARCHAR(100) CHECK(payment_method IN ('cash_on_delivery', 'online_transfer')),
  payment_status VARCHAR(100) CHECK(payment_status IN ('pending', 'paid'))
);
-- Modified to reference User_Orders
CREATE TABLE Orders (
  order_id INT PRIMARY KEY NOT NULL,
  user_order_id INT REFERENCES User_Orders(user_order_id),
  user_id INT REFERENCES Users(user_id),
  restaurant_id INT REFERENCES Restaurants(restaurant_id),
  order_date DATETIME,
  status VARCHAR(100) CHECK(status IN ('pending', 'confirmed', 'cancelled', 'delivered')),
  delivery_address VARCHAR(100),
  payment_method VARCHAR(100) CHECK(payment_method IN ('cash_on_delivery', 'online_transfer')),
  payment_status VARCHAR(100) CHECK(payment_status IN ('pending', 'paid'))
);
```

3rd normal form:

Identify the Issue:

Here's what creates a violation of 3NF in Reviews:

The order_id attribute depends on both user_id and restaurant_id (both are part of the primary key). This creates a transitive dependency because order_id can be determined by the combination of user_id and restaurant_id.

To achieve 3NF in Reviews, we can remove this transitive dependency by separating the order information from user and restaurant details.

Modified Tables in 3NF:

-- Updated Orders table:

```
CREATE TABLE Orders (
   order_id INT PRIMARY KEY NOT NULL,
   user_order_id INT REFERENCES User_Orders(user_order_id),
   restaurant_id INT REFERENCES Restaurants(restaurant_id),
   order_date DATETIME,
   status VARCHAR(100) CHECK(status IN ('pending', 'confirmed', 'cancelled', 'delivered')),
   delivery_address VARCHAR(100)
);
```

-- Updated Reviews table:

```
CREATE TABLE Reviews (
   review_id INT PRIMARY KEY NOT NULL,
   user_id INT REFERENCES Users(user_id),
   restaurant_id INT REFERENCES Restaurants(restaurant_id),
   user_order_id INT REFERENCES User_Orders(user_order_id),  -- New reference
   rating INT,
   comments TEXT,
);
```

SQL code after normalization:

```
CREATE TABLE Users (
  user_id INT PRIMARY KEY NOT NULL,
  username VARCHAR(100) UNIQUE,
  email VARCHAR(100) UNIQUE,
  password VARCHAR(100),
  first_name VARCHAR(100),
  last_name VARCHAR(100),
  phone_number VARCHAR(100),
  user_address VARCHAR(100)
);

CREATE TABLE Restaurants (
  restaurant_id INT PRIMARY KEY NOT NULL,
  name VARCHAR(100),
  description TEXT,
  location VARCHAR(100),
  rating DECIMAL(2,1),
  delivery_fee DECIMAL(10,2),
  minimum_order DECIMAL(10,2),
  user_id INT REFERENCES Users(user_id)
);

CREATE TABLE Categories (
  category_id INT PRIMARY KEY NOT NULL,
  name VARCHAR(100) UNIQUE
);

CREATE TABLE Foods (
  food_id INT PRIMARY KEY NOT NULL,
  name VARCHAR(100),
  description TEXT,
  price DECIMAL(10,2),
  category_id INT REFERENCES Categories(category_id),
  restaurant_id INT REFERENCES Restaurants(restaurant_id),
  is_available VARCHAR(100) DEFAULT 'yes' CHECK(is_available IN ('yes', 'no'))
);

CREATE TABLE User_Orders (
  user_order_id INT PRIMARY KEY NOT NULL,
  user_id INT FOREIGN KEY REFERENCES Users(user_id),
  delivery_fee DECIMAL(10,2),
  payment_method VARCHAR(100) CHECK(payment_method IN ('cash_on_delivery', 'online_transfer')),
  payment_status VARCHAR(100) CHECK(payment_status IN ('pending', 'paid'))
);

CREATE TABLE Orders (
  order_id INT PRIMARY KEY NOT NULL,
  user_order_id INT REFERENCES User_Orders(user_order_id),
  restaurant_id INT REFERENCES Restaurants(restaurant_id),
  order_date DATETIME,
  status VARCHAR(100) CHECK(status IN ('pending', 'confirmed', 'cancelled', 'delivered')),
  delivery_address VARCHAR(100),
);

CREATE TABLE Order_Items (
  order_item_id INT PRIMARY KEY NOT NULL,
  order_id INT REFERENCES Orders(order_id),
  food_id INT REFERENCES Foods(food_id),

  quantity INT,

  price DECIMAL(10,2),

);


CREATE TABLE Reviews (

  review_id INT PRIMARY KEY NOT NULL,

  user_id INT REFERENCES Users(user_id),

  restaurant_id INT REFERENCES Restaurants(restaurant_id),

  user_order_id INT REFERENCES User_Orders(user_order_id),

  rating INT,

  comments TEXT,

);


CREATE TABLE Coupons (

  coupon_id INT PRIMARY KEY NOT NULL,

  code VARCHAR(100) UNIQUE,

  discount_type VARCHAR(100) CHECK(discount_type IN ('percentage', 'fixed_amount')),

  discount_value DECIMAL(10,2),

  minimum_order DECIMAL(10,2),

  expiry_date DATETIME

);


CREATE TABLE User_Coupons (

  user_coupon_id INT PRIMARY KEY NOT NULL,

  user_id INT REFERENCES Users(user_id),

  coupon_id INT REFERENCES Coupons(coupon_id),

  is_used VARCHAR(100) DEFAULT 'no' CHECK(is_used IN ('yes', 'no'))

);
```

```
CREATE TABLE Order_Payments (
  order_payment_id INT PRIMARY KEY NOT NULL,
  order_id INT REFERENCES Orders(order_id),
  payment_gateway VARCHAR(100),
  transaction_id VARCHAR(100)
);
```