Recipe Organizer

Mercedes F. Alejandro, Damien A. Beltran, Kevin D. Henderson, Chris M. Villalobos, and Jayden Hendrix

CS-3743-001

Team Project

Business Information

Business Summary

- Developed a digital platform for a local, family-owned restaurant group.
- Connects home cooks, food lovers, and restaurants through:
  - Recipe discovery and reviews
  - Shopping list creation
  - Access to restaurant menus
- Features a robust database to manage recipes, ingredients user feedback, and menu updates.
- Supports the client's focus on local ingredients and regional culinary community.

Business Rules

User and Profile Management

- Users must provide full contact info (Email, Name, Address, etc.) to register.
- Each user is assigned a unique **ID Number.**
- User data must be **encrypted** and may be shared with 3rd parties (with notice).
- Users can request **data deletion** at any time.
- Users can export saved recipes and meal plans as PDF's.

Recipes & Reviews

- Each recipe must be linked to a valid **restaurant**.
- Recipes require: **title**, **ingredients**, **description**, **rating**, **instructions**.
- Recipes must include at least **one ingredient**.
- Users can **rate and review** recipes; reviews must link to a **user profile**.

Business Rules (Cont.)

Restaurant Requirements

- Restaurants must provide: **Name**, **Address**, **City**, **State**, **Zip Code**, **Phone Number**, **Average Rating**.

Search and Discovery

- The platform must support search filters for dietary preferences (e.g., vegan, keto, gluten-free, lactose-free)

ERD (Did we achieve a 3rd Normal Form?)

ERD

Our Diagram:

[Lucidchart ERD]

Does it Achieve 3rd Normal Form?

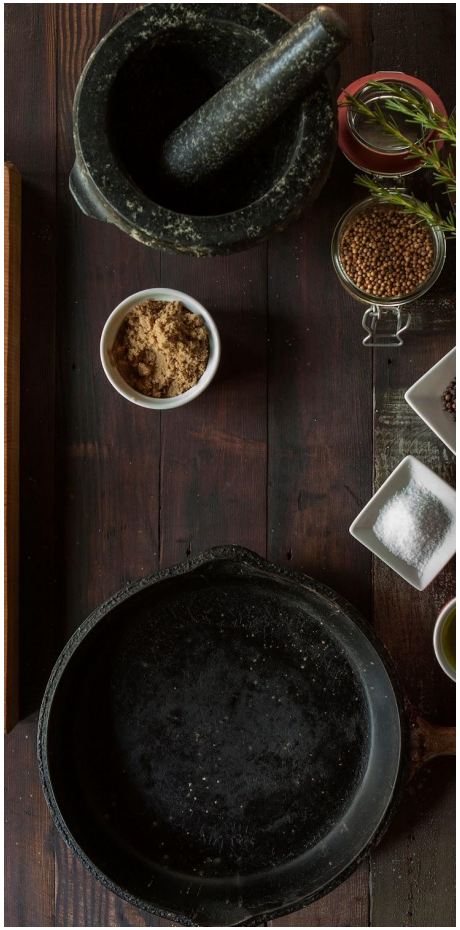
- Yes it does!

Why?

- No transitive dependencies, meaning all non-key attributes are dependent only on the primary key, not on other non key attributes.

Example

- In users, contact info is tied directly to the user, and the location details (zip, city, state) are offloaded to the **Zipcode** table.



SAMPLE DATA

Our DDL and Handling Repetition

Link to our DDL

Recipe Organizer DDL

Handling Repetition

Normalizing many-to-many relationships using junctions tables.

Example: The relationship between **Recipes** and **Ingredients**.

```
CREATE TABLE RecipeIngredients (
        Recipe_id INT NOT NULL,
        Ingredient_id INT NOT NULL,
        Quantity DECIMAL(5,2) NOT NULL,
        PRIMARY KEY (recipe_id,
ingredient_id);
```

This allows:

- Ingredients to be reused across multiple recipes without duplicating ingredient details.
- Quantities to be tracked individually per recipe.
- Data to stay normalized and consistent.

Query Demonstration

Query Demonstration - Kevin Henderson

Team DDL and CREATE Restaurants in DB

Link to our DDL CREATE Restaurants;

Recipe Organizer DDL

SQL Query:

```
CREATE TABLE IF NOT EXISTS Restaurants (
 `restaurant_id` int NOT NULL
AUTO_INCREMENT PRIMARY KEY,
 `name` varchar(500) NOT NULL,
 `address` varchar(500) NOT NULL,
 `city` varchar(500) NOT NULL,
 `state` varchar(500) NOT NULL,
 `zip_code` varchar(500) NOT NULL,
 `phone` varchar(500) NOT NULL,
 `rating` decimal(3,2) NOT NULL
) COMMENT 'Contains data about
restaurants that may offer recipes or
menu items.';
```

Explanation:

the `CREATE TABLE Restaurants` query is a direct implementation of the business rule requiring comprehensive details for each restaurant. It uses column definitions and `NOT NULL` constraints to ensure that the database structure itself mandates the provision of this essential information, which is critical for the platform's functionality and its service to the restaurant group and its users.

Team DDL and SELECT Queries in DB

Link to our DDL SELECT Queries;

[Recipe Organizer DDL]

SQL Query:

```
-- Kevin Henderson
-- Two Queries of Your Choice
-- 5) Total ingredient quantity per recipe
-- (shows how much "mass" each recipe calls for)
SELECT
    r.title,
    SUM(ri.quantity) AS total_qty
FROM RecipeIngredients ri
JOIN Recipes r ON ri.recipe_id = r.recipe_id
GROUP BY r.title
ORDER BY total_qty DESC;

-- Kevin Henderson
-- 6) Restaurants in TX offering items under $15
-- (find budget-friendly Texas menu items)
SELECT
    r.name AS restaurant,
    rec.title AS dish,
    rm.price
FROM RestaurantMenus rm
JOIN Restaurants r ON rm.restaurant_id = r.restaurant_id
JOIN Recipes rec ON rm.recipe_id = rec.recipe_id
WHERE r.state = 'TX' AND rm.price < 15.00;
```

Explanation of Query 1:

this query tells you for each dish on your menu, what's the total amount of ingredients needed to prepare it. This could be useful for inventory management, understanding the scale of different recipes, or even cost analysis.

Explanation of Query 2:

this query provides a list of restaurants in Texas along with the specific dishes they offer that cost less than $15. This is super helpful for customers looking for affordable dining options in Texas.

Query Demonstration - Mercedes Alejandro

Query with JOIN, GROUP BY, and HAVING - Active User Rating Summary

```sql
-- ============================================
-- Description: Lists users who have rated multiple recipes, showing their username
-- and the number of ratings they've submitted. This helps identify active users in
-- the recipe community for potential engagement or rewards.
-- ============================================
SELECT
    u.username AS User_Name,
    COUNT(ur.rating_id) AS Number_of_Ratings
FROM
    Users u
JOIN
    UserRatings ur ON u.user_id = ur.user_id
GROUP BY
    u.user_id, u.username
HAVING
    Number_of_Ratings > 1
ORDER BY
    Number_of_Ratings DESC;

```

User_Name	Number_of_Ratings
john_doe	3
jane_smith	2
alice_jones	2
bob_brown	2
charlie_black	2

Query Demonstration - Damien Beltran

Listing all recipes with their cooking time and difficulty

Query-

title	cooking_time	difficulty
Spaghetti Bolognese	30	medium
Chicken Curry	45	hard
Caesar Salad	15	easy
Beef Tacos	20	easy
Chocolate Cake	60	hard

```
SELECT
        title,
        cooking_time,
        difficulty
FROM
        Recipes;
```

Explanation-

This query,

- Retrieves key details like the title,, cooking time, and difficulty for each recipe from the Recipe table.
- Displays a concise overview of provided recipes, useful for users browsing or filtering based on time or complexity.
- Condoms that the Recipes table is populated and functioning correctly for basic data retrieval.

Query Demonstration - Jayden Hendrix

Query with JOIN, CONCAT, and DISTINCT – Restaurant + Location Summary

Query-

```sql
-- (JOIN + CONCAT + DISTINCT)
-- Description: Shows restaurant names, their combined location (city + state), and their menu prices.
SELECT DISTINCT
    res.restaurant_name AS Restaurant,
    CONCAT(loc.city, ', ', loc.state) AS Location,
    m.price AS Menu_Price
FROM
    Restaurants res
JOIN
    Locations loc ON res.location_id = loc.location_id
JOIN
    Menus m ON res.restaurant_id = m.restaurant_id
ORDER BY
    res.restaurant_name;
```

Explanation-

In this query,

- I used JOIN to combine the tables for “restaurant”, “locations”, and “menu”.
- I used CONCAT to combine the City and State together
- I used DISTINCT to eliminate duplicates

The output gives us the restaurant names, locations, and menu prices.

Query Demonstration - Chris Villalobos

Query for Total Number Ingredients in Inventory

Query-

```
-- Query 1:
Count the number of ingredients.

SELECT COUNT(*) AS total_ingredients

FROM Ingredients;
```

Explanation-

This query counts the total number of unique records in the Ingredients table. In the real world, this number represents the diversity of ingredients available in your database — for example, within a restaurant inventory, a meal planning app, or a grocery system.

Real-World Implications:

- Menu Design & Variety: A high count of ingredients (e.g., 157) suggests a wide range of dishes can be prepared, offering flexibility in menu creation.
- Inventory Management: Knowing the total number of ingredients helps businesses manage stock efficiently and reduce waste by identifying what's necessary versus excess.

Conclusion

Reflection - Mercedes Alejandro

- I was able to help out with the Business Rules for the database and assist with creating the data dependency diagram.
- One of the biggest challenges, was establishing clear business rules which really help set the tone for the database. However, after taking more time to really look into our entities and attributes I was able get a clear understanding of our end goal.
- Overall, I believe our team has met the expectations from the start of the project.

Reflection - Damien Beltran

- I was able to meet the original objectives and follow the business rules when designing the database.
- The most significant challenge was managing foreign key relationships across multiple tables.
    - I solved this by carefully planning the table creation order and using ‘ALTER TABLE’ to add constraints.

Reflection - Kevin Henderson

- I was able to help coordinate with the team to help build out the Business Rules for the Database and jump around in assisting with the creation of the Diagrams for the course project,

- this was challenging as there were lots of moving parts but also rewarding in that I was able to help give some solid structure to the project for the group as a whole collective.

Reflection - Chris Villalobos

- I was able to help develop the project to meet the original idea for the project. I helped in contributing to the details of the project such as developing the EDR and the data dependency diagram.

- A challenge that took us a bit to overcome was time management, especially when it came to deadlines. We managed by trying to finish parts of the project before the deadline came up.

- Overall, I believe that the project meet the expectations that we set from the start of the project

Reflection - Jayden Hendrix's

- I contributed to the development of our database structure and ensured it met the goals of the project.
- This project gave me better understanding of how different tables connect in a database.
- It also helped me learn to think from the user's perspective when designing queries.