What’s Cooking?

Sean Arackal, Alexander Bell, Spencer Greene, Axel Luca, Jamie Ortiz

About The Team

- Alexander Bell (CS MS ‘25)
- Axel Luca (CS BS/MS ‘25)
- Sean Arackal (CS BS/MS ‘26)
- Jamie Ortiz (CS BS ‘26)
- Spencer Greene (CS BS/MS ‘26)

What did we do?

- Worked to create a community-driven forum for recipe creation
    - Scraped recipes from the dataset found at https://www.kaggle.com/datasets/dimitryzub/allrecipes-all-us-recipes-by-state
    - Built out database structure
    - Created a frontend for our program, hosted at https://whats-cooking.fyi/
    - Users can build recipes and publish them for the community to see
    - Users are able to come and comment/rate on those recipes
    - Users are able to filter recipes among various metrics, such as:
        - Ingredients
        - Rating
        - Calories
        - Recipe Name
    - Created and tested triggers & views

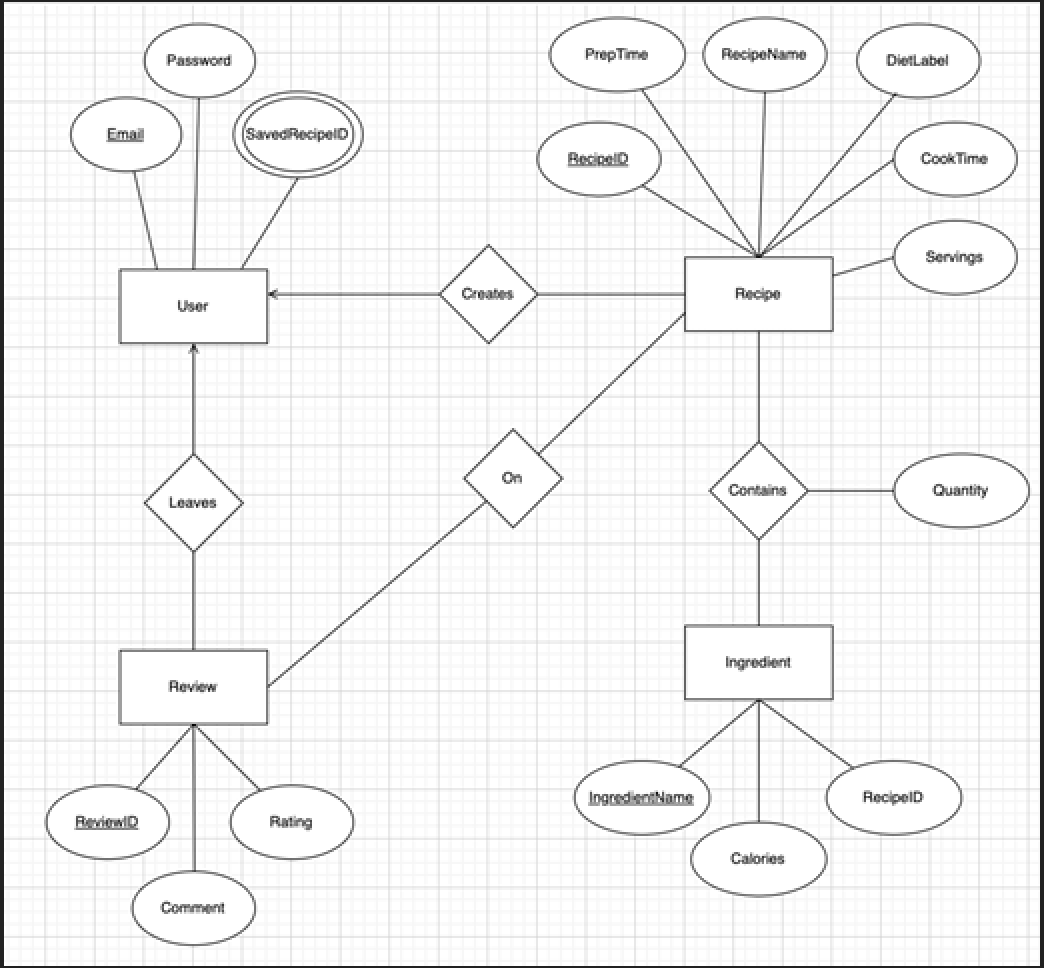
How did we do this?

- Databases, of course.
    - We will use a MySQL database to store our user data, comment information, recipe data, and ingredient data.
- Website Technologies
    - React, Node.JS, and Express.JS

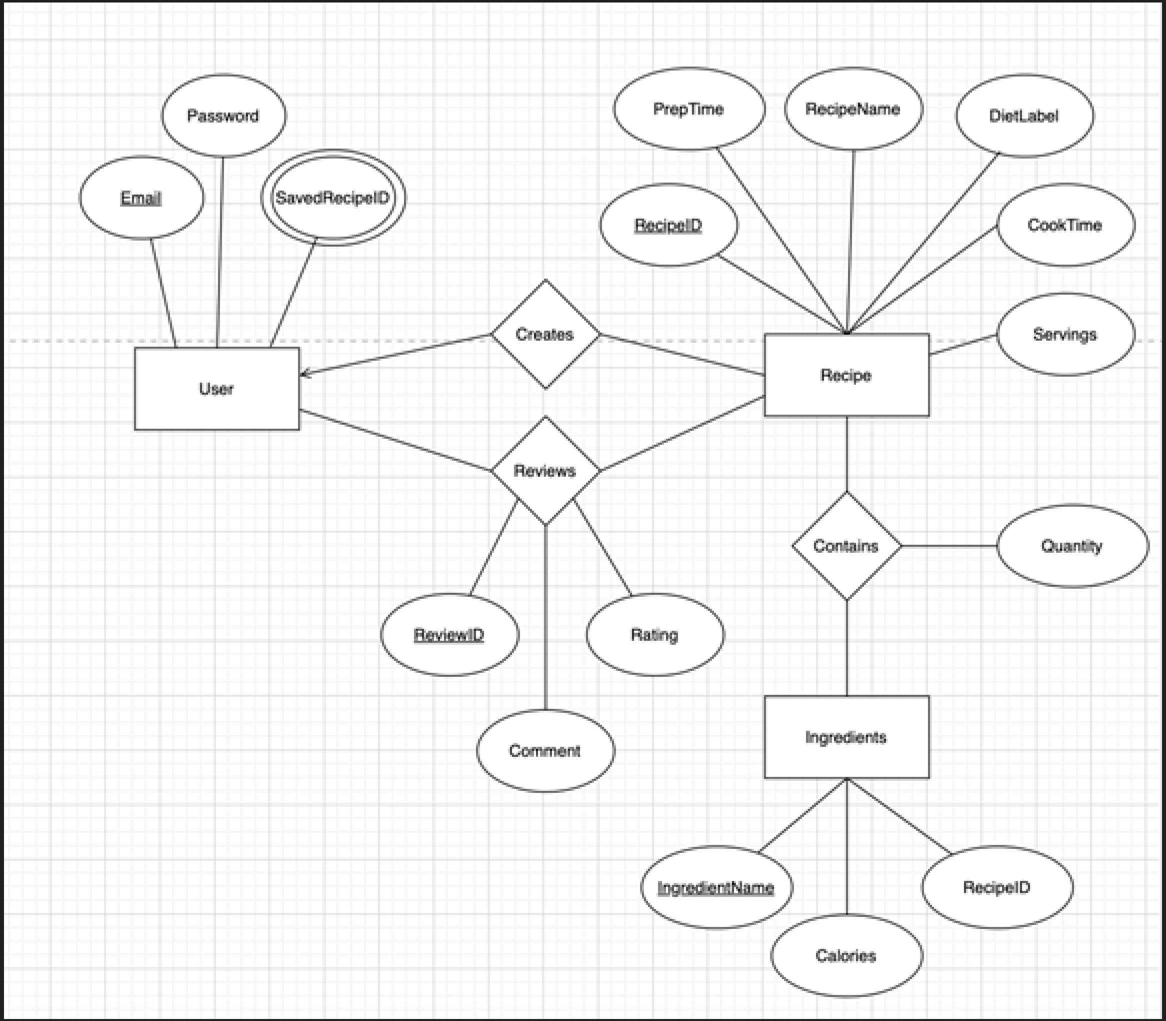
Why did we choose MySQL?

- In order to successfully complete, we decided to use MySQL as our backend. Some of the primary reasons for this choice were that MySQL is open-source, that it is used in multiple kinds of applications, its ease of setup, and the features it provides. To be more precise, the fact that MySQL is open-source is what allowed us to easily download and set up some of its products such as MySQLWorkbench, and make a MySQL server that could be shared remotely across different computers. In addition to this, examples of applications that rely on MySQL for functionality include Netflix, Uber, and Airbnb (Oracle, 2021). Moreover, MySQL provides several functionality features that we think are essential for our project such as views and triggers (Thomsen, 2002). Along with those, some additional features we thought would be useful for that project that MySQL includes are a LIMIT clause that allows users to limit the amount of rows shown when running queries, and an IFNULL function that allows users to specify a default value to return if a desired result returns NULL (Siahaan & Sianipar, 2019; Stephens & Russell, 2004).

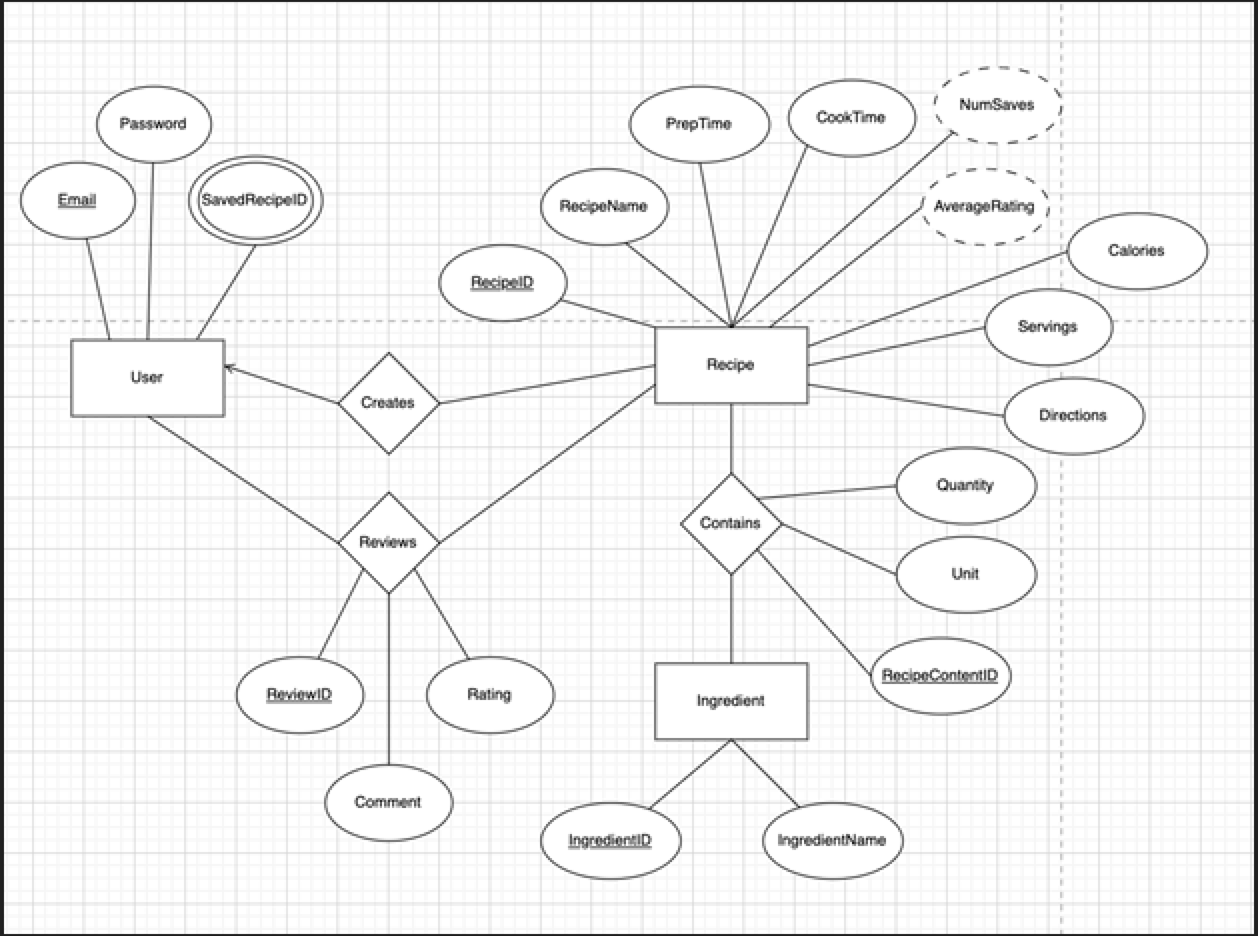
Initial ERD


Password
Email
SavedRecipeID
User
Creates
PrepTime
RecipeName
DietLabel
RecipeID
CookTime
Servings
Recipe
Leaves
On
Contains
Quantity
Review
Ingredient
ReviewID
Rating
Comment
IngredientName
RecipeID
Calories

ERD v2


Password
Email
SavedRecipeID
User
Creates
Reviews
ReviewID
Comment
Rating
PrepTime
RecipeName
DietLabel
RecipeID
CookTime
Servings
Recipe
Contains
Quantity
Ingredients
IngredientName
Calories
RecipeID

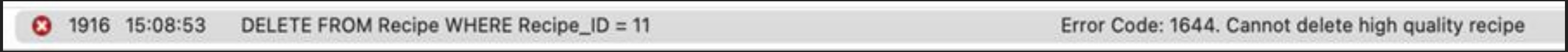
ERD v3



Password
Email
SavedRecipeID
User
Creates
Reviews
ReviewID
Rating
Comment
PrepTime
CookTime
NumSaves
RecipeName
AverageRating
RecipeID
Calories
Servings
Recipe
Directions
Quantity
Contains
Unit
RecipeContentID
Ingredient
IngredientID
IngredientName

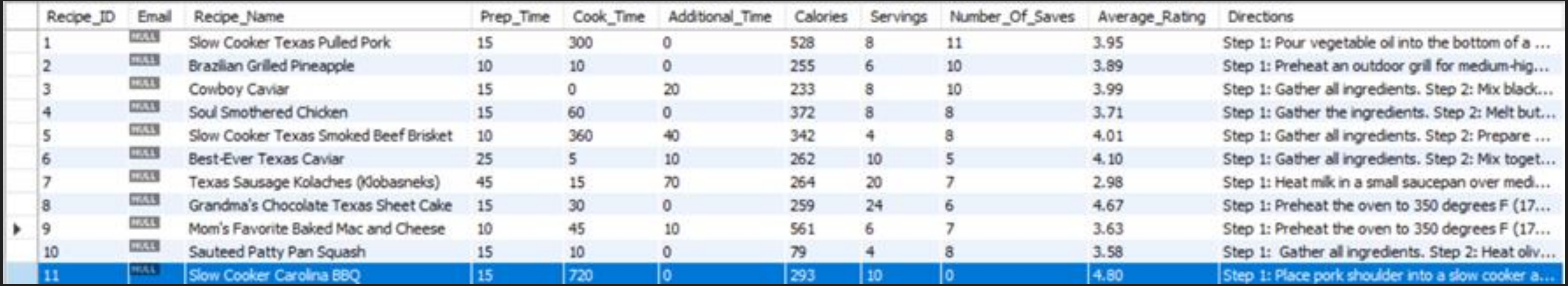
Testing Triggers

Trigger: Prevent high-quality recipes from being deleted



```
1916  15:08:53    DELETE FROM Recipe WHERE Recipe_ID = 11                    Error Code: 1644. Cannot delete high quality recipe
```

This trigger correctly prevents recipe ID = 11 (Whose average rating is 4.80) from being removed from the recipes table

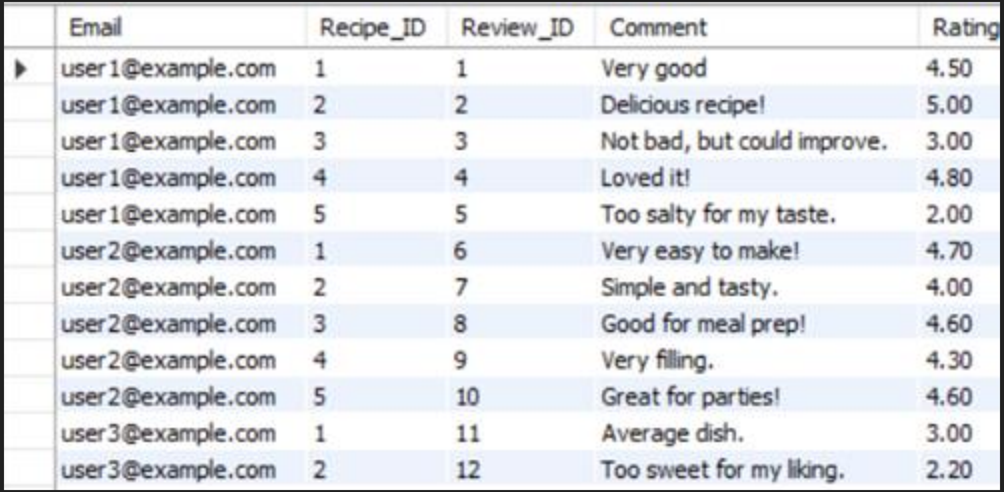


Recipe_ID	Email	Recipe_Name	Prep_Time	Cook_Time	Additional_Time	Calories	Servings	Number_Of_Saves	Average_Rating	Directions
1	NULL	Slow Cooker Texas Pulled Pork	15	300	0	528	8	11	3.95	Step 1: Pour vegetable oil into the bottom of a ...
2	NULL	Brazilian Grilled Pineapple	10	10	0	255	6	10	3.89	Step 1: Preheat an outdoor grill for medium-hig...
3	NULL	Cowboy Caviar	15	0	20	233	8	10	3.99	Step 1: Gather all ingredients. Step 2: Mix black...
4	NULL	Soul Smothered Chicken	15	60	0	372	8	8	3.71	Step 1: Gather the ingredients. Step 2: Melt but...
5	NULL	Slow Cooker Texas Smoked Beef Brisket	10	360	40	342	4	8	4.01	Step 1: Gather all ingredients. Step 2: Prepare ...
6	NULL	Best-Ever Texas Caviar	25	5	10	262	10	5	4.10	Step 1: Gather all ingredients. Step 2: Mix toget...
7	NULL	Texas Sausage Kolaches (Klobasneks)	45	15	70	264	20	7	2.98	Step 1: Heat milk in a small saucepan over medi...
8	NULL	Grandma's Chocolate Texas Sheet Cake	15	30	0	259	24	6	4.67	Step 1: Preheat the oven to 350 degrees F (17...
9	NULL	Mom's Favorite Baked Mac and Cheese	10	45	10	561	6	7	3.63	Step 1: Preheat the oven to 350 degrees F (17...
10	NULL	Sauteed Patty Pan Squash	15	10	0	79	4	8	3.58	Step 1: Gather all ingredients. Step 2: Heat oliv...
11	NULL	Slow Cooker Carolina BBQ	15	720	0	293	10	0	4.80	Step 1: Place pork shoulder into a slow cooker a...

Testing Triggers

Trigger: Increment review ID by one every time a new review is created

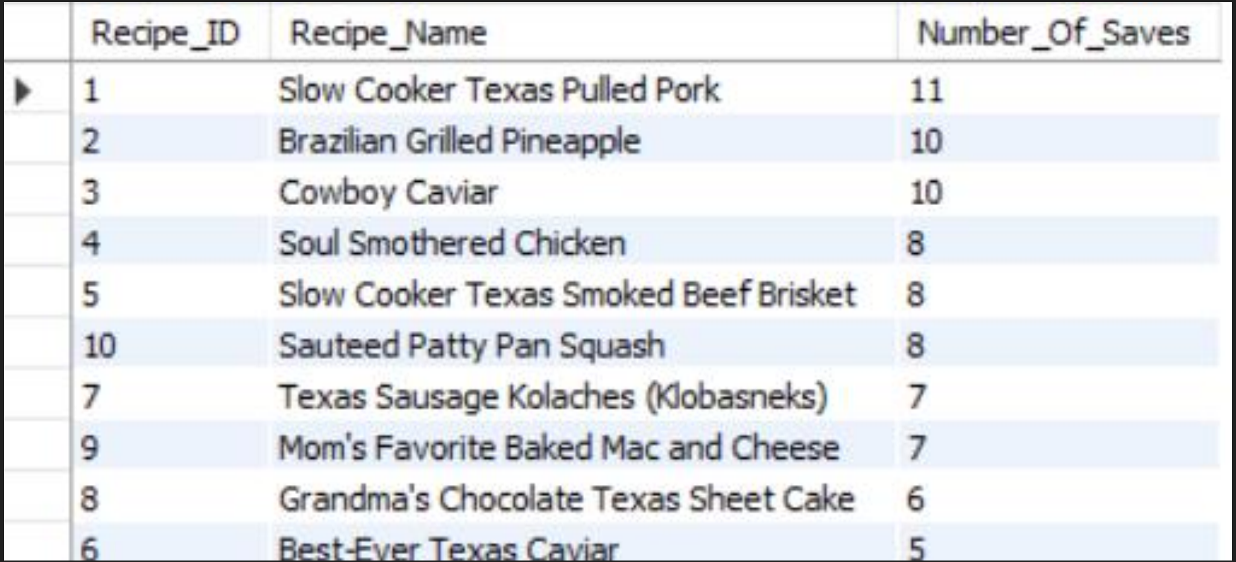
We couldn’t use the Auto Increment feature of MySQL Due to the structure of our Database, so we wanted to test to ensure that Review ID is being correctly incremented



Email	Recipe_ID	Review_ID	Comment	Rating
user1@example.com	1	1	Very good	4.50
user1@example.com	2	2	Delicious recipe!	5.00
user1@example.com	3	3	Not bad, but could improve.	3.00
user1@example.com	4	4	Loved it!	4.80
user1@example.com	5	5	Too salty for my taste.	2.00
user2@example.com	1	6	Very easy to make!	4.70
user2@example.com	2	7	Simple and tasty.	4.00
user2@example.com	3	8	Good for meal prep!	4.60
user2@example.com	4	9	Very filling.	4.30
user2@example.com	5	10	Great for parties!	4.60
user3@example.com	1	11	Average dish.	3.00
user3@example.com	2	12	Too sweet for my liking.	2.20

Testing Views

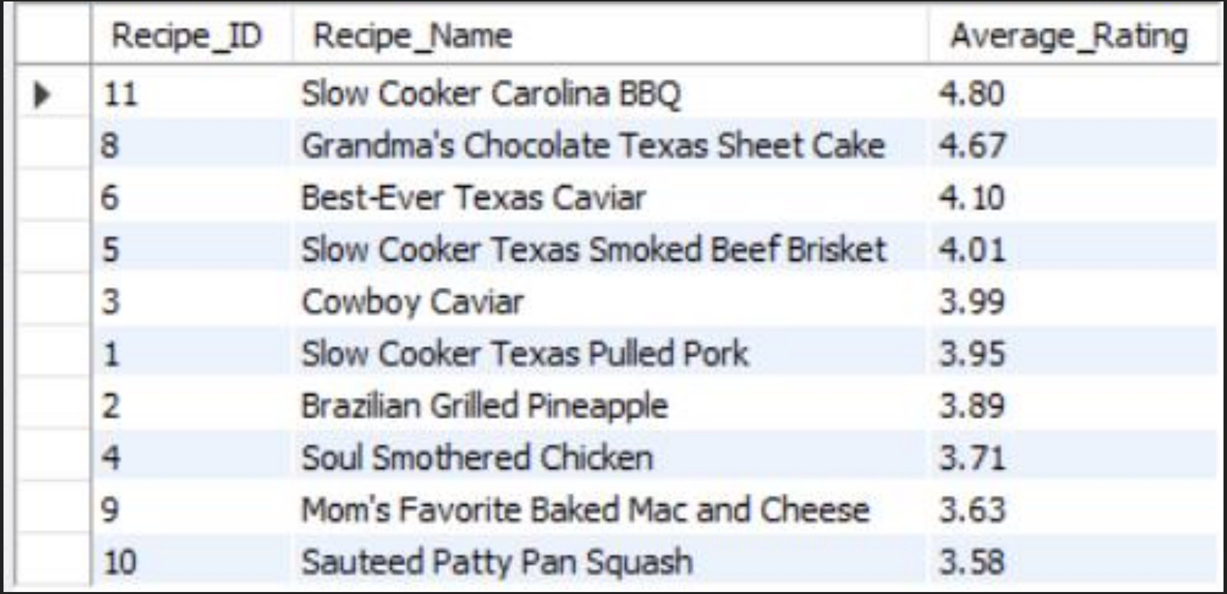
This view's purpose is to get 10 most saved recipes. This also tested the trigger that calculates the new number of saves when a recipe is saved.



Recipe_ID	Recipe_Name	Number_Of_Saves
1	Slow Cooker Texas Pulled Pork	11
2	Brazilian Grilled Pineapple	10
3	Cowboy Caviar	10
4	Soul Smothered Chicken	8
5	Slow Cooker Texas Smoked Beef Brisket	8
10	Sauteed Patty Pan Squash	8
7	Texas Sausage Kolaches (Klobasneks)	7
9	Mom's Favorite Baked Mac and Cheese	7
8	Grandma's Chocolate Texas Sheet Cake	6
6	Best-Ever Texas Caviar	5

Testing Views

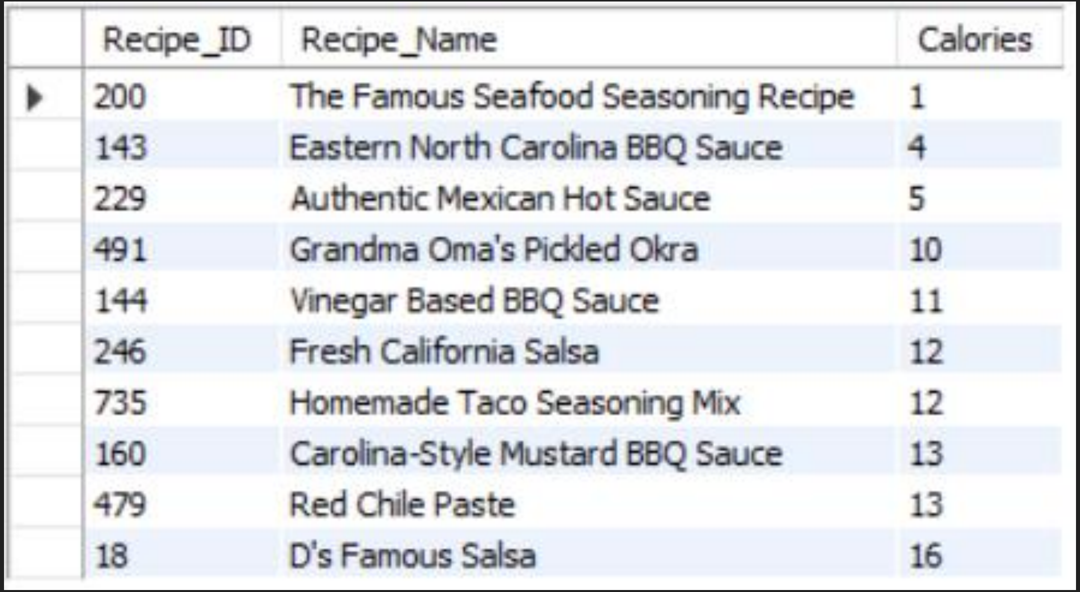
This view’s purpose is to get 10 highest average rating recipes. This also tested the trigger that calculates the new average rating when a review is created.



Recipe_ID	Recipe_Name	Average_Rating
11	Slow Cooker Carolina BBQ	4.80
8	Grandma's Chocolate Texas Sheet Cake	4.67
6	Best-Ever Texas Caviar	4.10
5	Slow Cooker Texas Smoked Beef Brisket	4.01
3	Cowboy Caviar	3.99
1	Slow Cooker Texas Pulled Pork	3.95
2	Brazilian Grilled Pineapple	3.89
4	Soul Smothered Chicken	3.71
9	Mom's Favorite Baked Mac and Cheese	3.63
10	Sauteed Patty Pan Squash	3.58

Testing Views

This view’s purpose is to get 10 lowest calorie recipes.

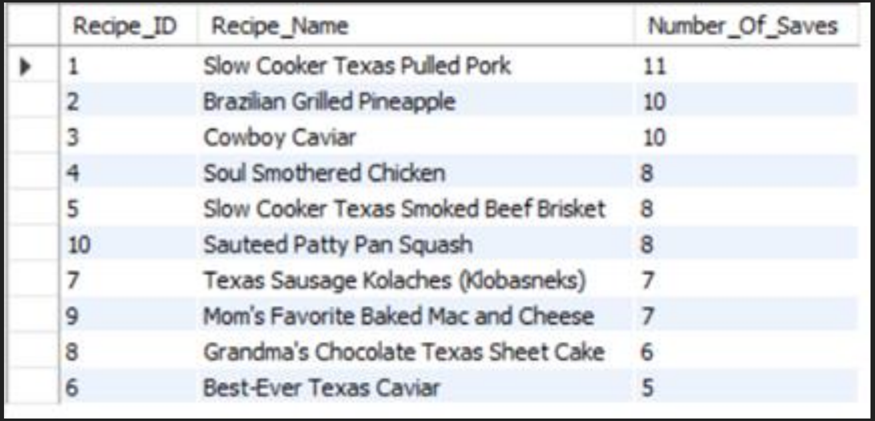


Recipe_ID	Recipe_Name	Calories
200	The Famous Seafood Seasoning Recipe	1
143	Eastern North Carolina BBQ Sauce	4
229	Authentic Mexican Hot Sauce	5
491	Grandma Oma's Pickled Okra	10
144	Vinegar Based BBQ Sauce	11
246	Fresh California Salsa	12
735	Homemade Taco Seasoning Mix	12
160	Carolina-Style Mustard BBQ Sauce	13
479	Red Chile Paste	13
18	D's Famous Salsa	16

Testing Triggers

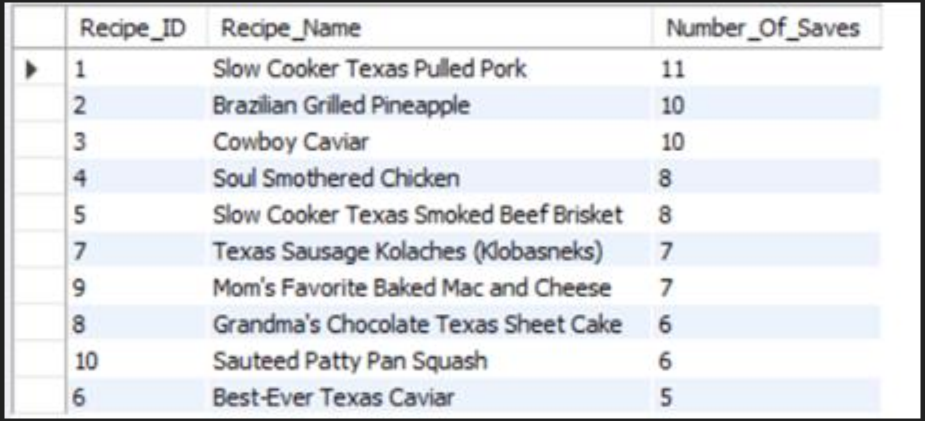
Trigger: Update the number of saves a recipe has when a user unsaves a recipe

In this example, we “unsave” recipe 10 twice, and the number of saves drops accordingly



Recipe_ID	Recipe_Name	Number_Of_Saves
1	Slow Cooker Texas Pulled Pork	11
2	Brazilian Grilled Pineapple	10
3	Cowboy Caviar	10
4	Soul Smothered Chicken	8
5	Slow Cooker Texas Smoked Beef Brisket	8
10	Sauteed Patty Pan Squash	8
7	Texas Sausage Kolaches (Klobasneks)	7
9	Mom's Favorite Baked Mac and Cheese	7
8	Grandma's Chocolate Texas Sheet Cake	6
6	Best-Ever Texas Caviar	5

Before



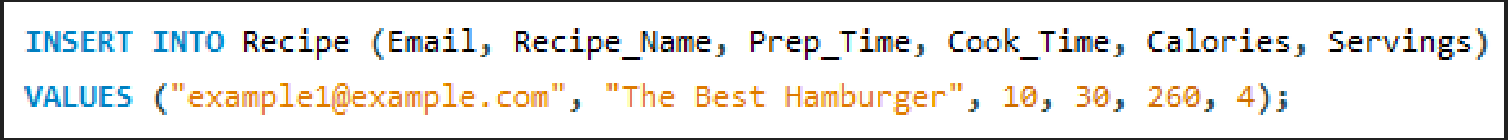
Recipe_ID	Recipe_Name	Number_Of_Saves
1	Slow Cooker Texas Pulled Pork	11
2	Brazilian Grilled Pineapple	10
3	Cowboy Caviar	10
4	Soul Smothered Chicken	8
5	Slow Cooker Texas Smoked Beef Brisket	8
7	Texas Sausage Kolaches (Klobasneks)	7
9	Mom's Favorite Baked Mac and Cheese	7
8	Grandma's Chocolate Texas Sheet Cake	6
10	Sauteed Patty Pan Squash	6
6	Best-Ever Texas Caviar	5

After

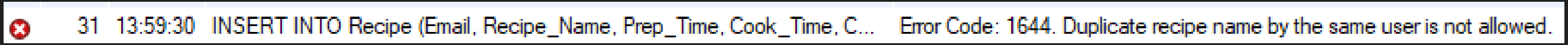
Testing Triggers

Trigger: Don’t allow multiple recipes of the same name from the same user.

In this example, user “Example1@example.com” tried to create a recipe called “The Best Hamburger” for a second time, which causes the trigger to raise an error.



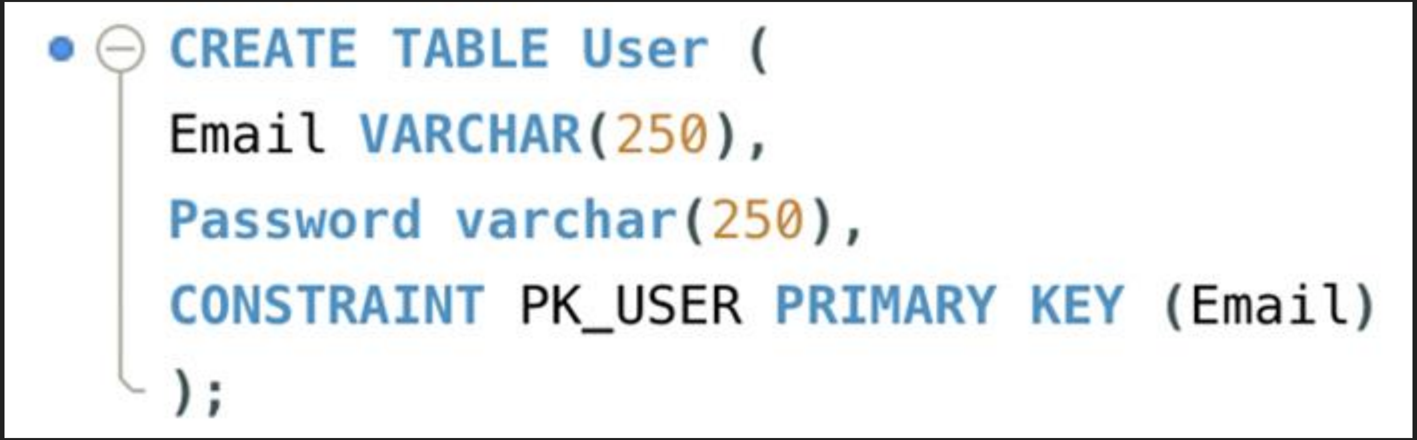
```
INSERT INTO Recipe (Email, Recipe_Name, Prep_Time, Cook_Time, Calories, Servings)
VALUES ("example1@example.com", "The Best Hamburger", 10, 30, 260, 4);
```



```
31  13:59:30  INSERT INTO Recipe (Email, Recipe_Name, Prep_Time, Cook_Time, C...  Error Code: 1644. Duplicate recipe name by the same user is not allowed.
```

Normalization of Database Tables: User Table

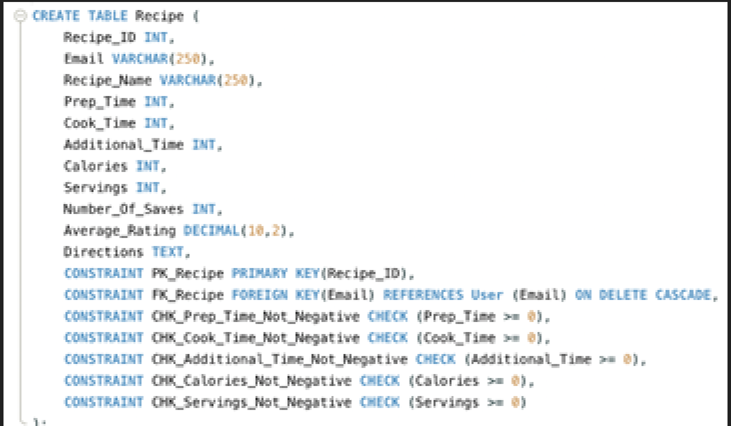
- **Functional Dependency:** Each unique email determines a password (Email → Password).
- **Normalization:** Since all functional dependencies have a superkey on the left-hand side, the table satisfies BCNF.



```
CREATE TABLE User (
Email VARCHAR(250),
Password varchar(250),
CONSTRAINT PK_USER PRIMARY KEY (Email)
);
```

Normalization: Recipe Table

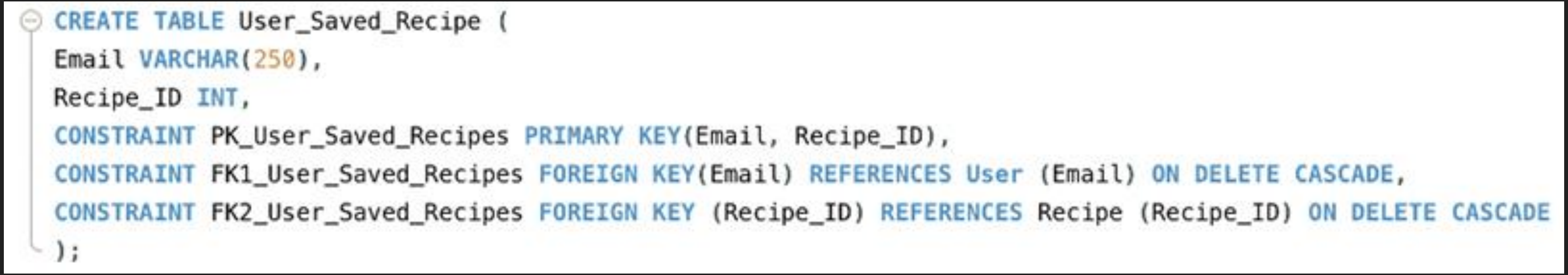
- **Functional Dependency:** The primary key, Recipe_ID, uniquely determines all other attributes (Recipe_ID → any subset of the attributes Email, Recipe_Name, Prep_Time, Cook_Time, Additional_Time, Calories, Servings, Number_Of_Saves, Average_Rating, Directions).
- **Normalization:** The table is in BCNF as all functional dependencies have a superkey on their left side.



```
CREATE TABLE Recipe (
    Recipe_ID INT,
    Email VARCHAR(250),
    Recipe_Name VARCHAR(250),
    Prep_Time INT,
    Cook_Time INT,
    Additional_Time INT,
    Calories INT,
    Servings INT,
    Number_Of_Saves INT,
    Average_Rating DECIMAL(10,2),
    Directions TEXT,
    CONSTRAINT PK_Recipe PRIMARY KEY(Recipe_ID),
    CONSTRAINT FK_Recipe FOREIGN KEY(Email) REFERENCES User (Email) ON DELETE CASCADE,
    CONSTRAINT CHK_Prep_Time_Not_Negative CHECK (Prep_Time >= 0),
    CONSTRAINT CHK_Cook_Time_Not_Negative CHECK (Cook_Time >= 0),
    CONSTRAINT CHK_Additional_Time_Not_Negative CHECK (Additional_Time >= 0),
    CONSTRAINT CHK_Calories_Not_Negative CHECK (Calories >= 0),
    CONSTRAINT CHK_Servings_Not_Negative CHECK (Servings >= 0)
);
```

Normalization: User Saved Recipe Table

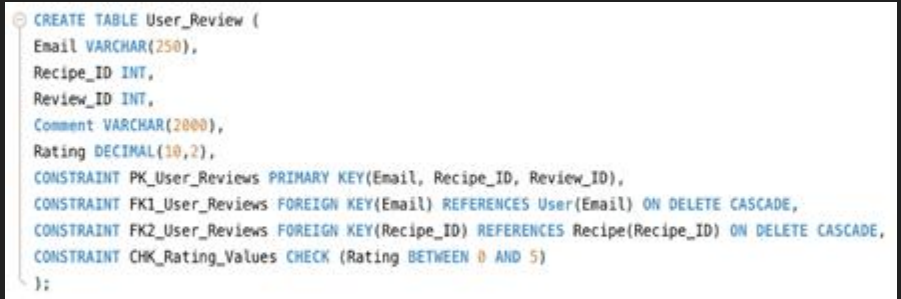
- **Modeling Approach**: Each user can save multiple recipes, represented as a composite attribute in the ER diagram.
- **Primary Key**: The composite primary key is the combination of Email and Recipe_ID.
- **Functional Dependencies**: There are no functional dependencies as the full set of attributes forms the candidate key.
- **Normalization**: The table is in BCNF



```sql
CREATE TABLE User_Saved_Recipe (
Email VARCHAR(250),
Recipe_ID INT,
CONSTRAINT PK_User_Saved_Recipes PRIMARY KEY(Email, Recipe_ID),
CONSTRAINT FK1_User_Saved_Recipes FOREIGN KEY(Email) REFERENCES User (Email) ON DELETE CASCADE,
CONSTRAINT FK2_User_Saved_Recipes FOREIGN KEY (Recipe_ID) REFERENCES Recipe (Recipe_ID) ON DELETE CASCADE
);
```

Normalization: User Review Table

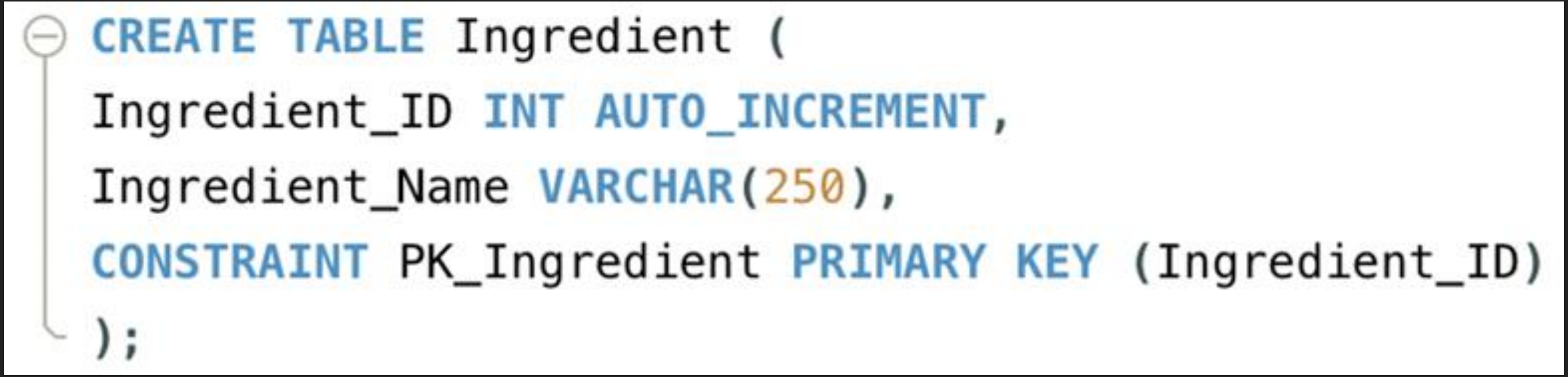
- **Modeling Approach:** Designed as a many-to-many relationship table linking User and Recipe tables, with Email and Recipe_ID initially as primary keys.
- **Primary Key Adjustment:** Added Review_ID to the primary key to allow multiple reviews for the same recipe by a single user.
- **Functional Dependencies:**
  - Composite dependency: (Email, Recipe_ID, Review_ID) → Comment, Rating.
  - Single attribute dependency: Review_ID → any subset of the attributes Email, Recipe_ID, Comment, Rating.
- **Candidate Key:** Review_ID is the candidate key as it uniquely identifies each row.
- **Normalization:** The table satisfies BCNF as all functional dependencies have a superkey on their left side.



```
CREATE TABLE User_Review (
Email VARCHAR(250),
Recipe_ID INT,
Review_ID INT,
Comment VARCHAR(2000),
Rating DECIMAL(10,2),
CONSTRAINT PK_User_Reviews PRIMARY KEY(Email, Recipe_ID, Review_ID),
CONSTRAINT FK1_User_Reviews FOREIGN KEY(Email) REFERENCES User(Email) ON DELETE CASCADE,
CONSTRAINT FK2_User_Reviews FOREIGN KEY(Recipe_ID) REFERENCES Recipe(Recipe_ID) ON DELETE CASCADE,
CONSTRAINT CHK_Rating_Values CHECK (Rating BETWEEN 0 AND 5)
);
```

Normalization: Ingredient Table

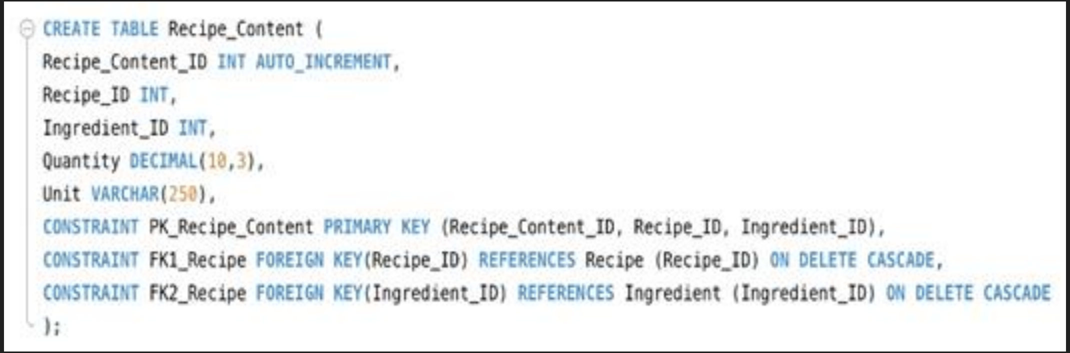
- **Functional Dependency:** The primary key, Ingredient_ID, uniquely determines Ingredient_Name (Ingredient_ID → Ingredient_Name).
- **Uniqueness:** Ingredient_ID ensures each row is unique.
- **Normalization:** The table is in BCNF as all functional dependencies have a superkey on their left side.



```
CREATE TABLE Ingredient (
Ingredient_ID INT AUTO_INCREMENT,
Ingredient_Name VARCHAR(250),
CONSTRAINT PK_Ingredient PRIMARY KEY (Ingredient_ID)
);
```

Normalization: Recipe Content Table

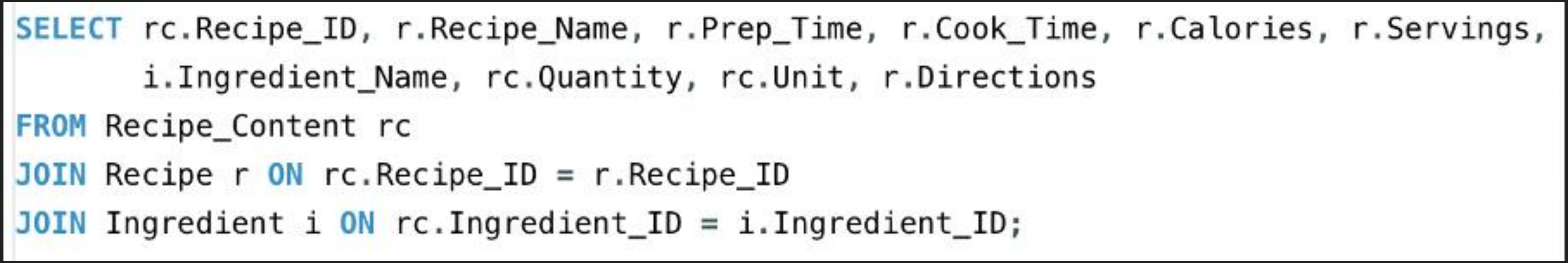
- **Modeling Approach:** Designed as a many-to-many relationship table for recipes and ingredients with Recipe_ID and Ingredient_ID as initial keys.
- **Primary Key Adjustment:** Added Recipe_Content_ID to the primary key to handle variations in ingredient quantity and units for recipes.
- **Functional Dependencies:**
  - Composite dependency: (Recipe_Content_ID, Recipe_ID, Ingredient_ID) → Quantity, Unit.
  - Single attribute dependency: Recipe_Content_ID → any subset of the attributes Recipe_ID, Ingredient_ID, Quantity, Unit.
- **Candidate Key:** Recipe_Content_ID uniquely identifies each row.
- **Normalization:** The table satisfies BCNF as all functional dependencies have a superkey on their left side.



```sql
CREATE TABLE Recipe_Content (
Recipe_Content_ID INT AUTO_INCREMENT,
Recipe_ID INT,
Ingredient_ID INT,
Quantity DECIMAL(10,3),
Unit VARCHAR(250),
CONSTRAINT PK_Recipe_Content PRIMARY KEY (Recipe_Content_ID, Recipe_ID, Ingredient_ID),
CONSTRAINT FK1_Recipe FOREIGN KEY(Recipe_ID) REFERENCES Recipe (Recipe_ID) ON DELETE CASCADE,
CONSTRAINT FK2_Recipe FOREIGN KEY(Ingredient_ID) REFERENCES Ingredient (Ingredient_ID) ON DELETE CASCADE
);
```

Query Optimization

- Our database table with the most rows is Recipe_Content, with over 15000 rows and will continue to grow as more recipes are created.
- The below query is commonly called to get information from recipes

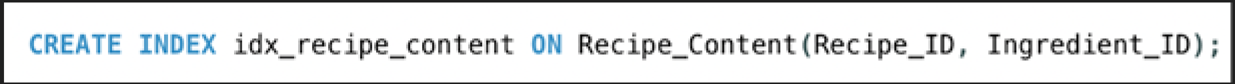


```
SELECT rc.Recipe_ID, r.Recipe_Name, r.Prep_Time, r.Cook_Time, r.Calories, r.Servings,
       i.Ingredient_Name, rc.Quantity, rc.Unit, r.Directions
FROM Recipe_Content rc
JOIN Recipe r ON rc.Recipe_ID = r.Recipe_ID
JOIN Ingredient i ON rc.Ingredient_ID = i.Ingredient_ID;
```

- This query causes a lot of I/Os in its current format
- We added an index to alleviate this

Query Optimization (Cont.)

- We created a query using the following command

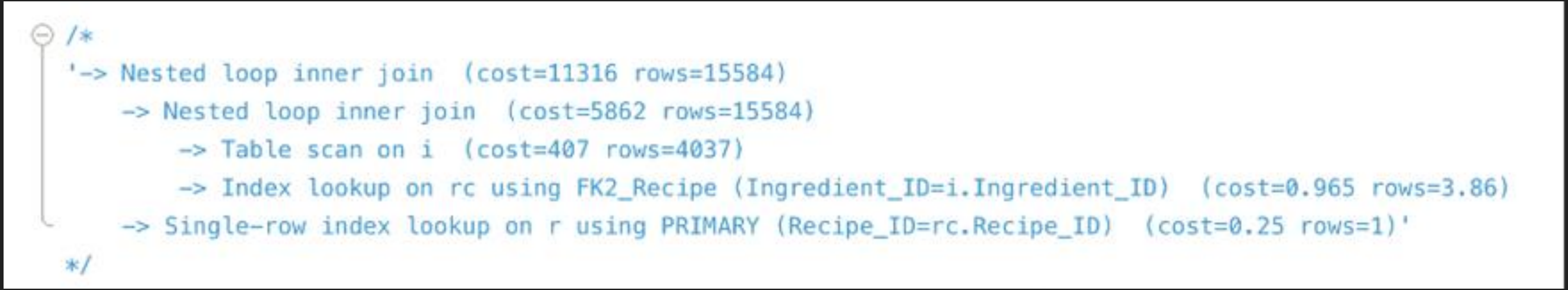


```
CREATE INDEX idx_recipe_content ON Recipe_Content(Recipe_ID, Ingredient_ID);
```

- This would allow us to more efficiently compute this query, since both the Ingredient table and Recipe table are join by Recipe_Content, and we assumed a composite index would help
- As shown in the next slide, adding the index does help the query run faster by performing less I/Os in both inner loops and the table scan

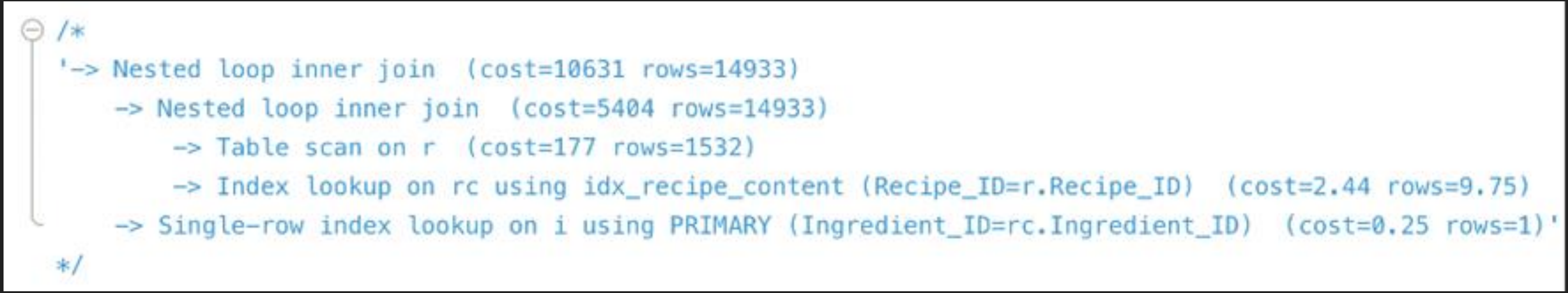
Query Optimization Results

Before:



```
/*
'-> Nested loop inner join  (cost=11316 rows=15584)
    -> Nested loop inner join  (cost=5862 rows=15584)
        -> Table scan on i  (cost=407 rows=4037)
        -> Index lookup on rc using FK2_Recipe (Ingredient_ID=i.Ingredient_ID)  (cost=0.965 rows=3.86)
    -> Single-row index lookup on r using PRIMARY (Recipe_ID=rc.Recipe_ID)  (cost=0.25 rows=1)'
*/
```

After:



```
/*
'-> Nested loop inner join  (cost=10631 rows=14933)
    -> Nested loop inner join  (cost=5404 rows=14933)
        -> Table scan on r  (cost=177 rows=1532)
        -> Index lookup on rc using idx_recipe_content (Recipe_ID=r.Recipe_ID)  (cost=2.44 rows=9.75)
    -> Single-row index lookup on i using PRIMARY (Ingredient_ID=rc.Ingredient_ID)  (cost=0.25 rows=1)'
*/
```

Future work:

- Allow users to import their own ingredients, instead of being forced to choose from the existing list
- Change the recipe webpage so that reviews are updated as soon as one is created by the user
- Allow users to view other user’s profiles, which could allow for smaller communities to form within the platform
    - Adding a social media element to our website with people sharing tutorials or attempts of recipes they cooked
- Prevent users from leaving multiple reviews on the same recipe
    - Instead it should update an existing review if one already exists, and if not create one

Access To Our Working Website

https://whats-cooking.fyi/

Video Demonstration Link

- https://wpi0-my.sharepoint.com/:v:/g/personal/jcordova_wpi_edu/EZeeI-oYLdxCpvUJPrXhsA4BwQa8_aSQ7iAFAZEUHWfOrg?nav=eyJyZWZlcnJhbEluZm8iOnsicmVmZXJyYWxBcHAiOiJTdHJlYW1XZWJBcHAiLCJyZWZlcnJhbFZpZXciOiJTaGFyZURpYWxvZy1MaW5rIiwicmVmZXJyYWxBcHBQbGF0Zm9ybSI6IldlYiIsInJlZmVycmFsTW9kZSI6InZpZXcifX0%3D&e=FZAUpg

References

- Oracle. (2021). *What is MySQL?* Oracle. https://www.oracle.com/mysql/what-is-mysql/
- Thomsen, C. (2002). Using Stored Procedures, Views, and Triggers: How to Use Stored Procedures, Views, and Triggers. In *Database Programming with C#* (pp. 367-412). Berkeley, CA: Apress.
- Siahaan, V., & Sianipar, R. H. (2019). *The Self-Taught Coder: The Definitive Guide to Database Programming with Python and MySQL* (Vol. 1). SPARTA Publishing.
- Stephens, J., & Russell, C. (2004). Optimizing Queries with Operators, Branching, and Functions. In *Beginning MySQL Database Design and Optimization: From Novice to Professional* (pp. 171-238). Berkeley, CA: Apress.
- Komperla, Varun & Pratiba, Deenadhayalan & Ghuli, Poonam & Pattar, Ramakanthkumar. (2022). React: A detailed survey. Indonesian Journal of Electrical Engineering and Computer Science. 26. 1710. 10.11591/ijeecs.v26.i3.pp1710-1717.