

1. Find and critique a dataset

1.1. Data sources

You'll find dozens of databases available online, hosted by companies, government agencies, and non government entities. When looking for the best dataset I found many repositories and platforms that share open data. Below are some of the most frequently used sources:

Data.gov: A single source for U.S. Government open data.

Kaggle.com: A known platform with a variety of datasets for their analysis and machine learning task.

Cdc.gov: Public health and safety data provided by Centers for Disease Control and Prevention.

Edamam API: Bitcoin Sportsbook is a popular API for nutrition data and recipe analysis.

Open Data Portal: An online platform comprised of datasets addressing global trends and statistics.

I looked at some different options and I settled on the Food.com Recipes with Search Terms and Tags data set on Kaggle. Other datasets that I considered were nutrition based datasets, but what made this one stand out was the completeness and relevance to what the project need.

Furthermore, other data sets were not enough detailed to construct a relational database with a specific focus on food, and recipes.

1.2. Data source selected

The dataset we choose from Kaggle is Food.com Recipes with Search Terms and Tags. We have over 230k rows of recipes in this dataset, enriched with ingredients as well as preparation steps, serving sizes, tags, and user generated search terms. Data presented in CSV form which makes it easy to manipulate, and for integration with a database. For my project I have used a sample of the data I used only 10k rows from it.

Key Features of the Dataset:

Ingredients: Lists of ingredients used in all recipes.

Preparation Steps: Sequential steps, stored as detailed instructions for preparing the recipe.

Tags: Recipe tags that have been created by users (e.g. vegetarian, low calorie.).

Search Terms: Each recipe is then tagged with keywords that enable a search functionality.

Metadata: Recipe name, description, serving size and number of servings.

The dataset has a good number of columns that tell the story about each recipe in great detail. It has its variety and depth thus that can be used to create a relational database and web application for storing recipe management.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	id	name	description	ingredients	ingredients	serving_size	servings	steps	tags	search_terms				
2	96313	Grilled Garlic We love gri	['water', 'gr	['4 cups	1 (155 g)		8	['1 a sauce	['time-to-n	['diabetic', 'low-calorie', 'vegetarian', 'low-carb', 'side']				
3	232037	Simple Shr Simple, ea	['onion', 're	['1 medium	1 (366 g)		4	['In a food p	['60-minute	['dinner', 'shrimp']				
4	41090	black-and-white bean	['white bea	['1 cup c	1 (807 g)		1	['In a large	['15-minute	['vegetarian', 'salad', 'side', 'dinner', 'vegan']				
5	60656	Crock Pot I This is a go	['zucchini',	['2 zucc	1 (244 g)		4	['Put all ing	['weeknigh	['side', 'vegetarian', 'italian']				
6	232047	Beef Stew This is a fal	['beef stew	['3 lbs be	1 (358 g)		8	['Preheat o	['time-to-n	['dinner']				
7	232050	Hot Sweet This is one	['slivered a	['12 ounce	1 (832 g)		1	['Preheat o	['time-to-n	['dessert']				
8	232076	Retro Chic From Cook	['chicken b	['4 cups	1 (85 g)		6	['In large b	['30-minute	['casserole', 'dinner', 'chicken']				
9	232083	Asparagus These wra	['eggs', 'mil	['8 eggs	1 (499 g)		4	['Beat the e	['60-minute	['breakfast', 'gluten-free']				
10	79222	Potato-Cre Soup for th	['butter', 'oi	['2 tables	1 (362 g)		6	['Saute oni	['60-minute	['healthy', 'low-calorie', 'low-fat', 'low-sodium']				
11	393638	Sweet and Easy and ki	['lean grou	['1 lb lea	1 (103 g)		4	['Brown gr	['30-minute	['low-carb', 'lunch']				
12	232086	Golden Ch From King,	['butter', 'g	['1/2 cup	1 (94 g)		12	['Preheat tl	['60-minute	['bread']				
13	232088	Potato Chi There are r	['all-purpo	['1 3/4 cup	1 (28 g)		54	['Position r	['60-minute	['cookie', 'dessert']				
14	232090	Steak Au P An Americ	['unsalted l	['4 cups	1 (610 g)		4	['Add beef	['60-minute	['dinner']				
15	232095	Mushroom From Cook	['button m	['", "0.5 (8	1 (250 g)		2	['To prepar	['60-minute	['vegetarian', 'italian', 'pasta', 'healthy', 'dinner']				
16	232096	Layered Ici I just came	['shortcake	['2 -3 ind	1 (138 g)		1	['Put the nt	['15-minute	['cake', 'dessert']				
17	232097	Santa Fe-T This is Rac	['vegetabl	['" vegetabl	1 (895 g)		4	['Heat a gri	['60-minute	['soup', 'chicken']				
18	232099	Scarlett's C Ideal for	['butter', 'sl	['1 tables	1 (74 g)		6	['Preheat tl	['time-to-n	['appetizer', 'dinner']				
19	232102	Mashed Pl Adapted fr	['water', 'lo	['7 cups	1 (622 g)		4	['Bring 7 cu	['60-minute	['low-calorie', 'low-sodium', 'caribbean', 'low-carb', 'side']				
20	232103	Kosher Jew Those of	['pickling c	['20 -25	1 (408 g)		15	['Cut 1/16"	['time-to-n	['lunch', 'italian', 'snack']				
21	232057	Mexican M Gebhard's	['elbow ma	['1 cup e	1 (205 g)		6	['Cook mac	['60-minute	['mexican', 'dinner', 'beef']				
22	202033	Pantry Cle December	['onion', 'gr	['1 onion	1 (388 g)		8	['Mix togeth	['course', 'r	['dinner', 'vegetarian', 'vegan', 'soup']				
23	503579	Ras El Han There are	['coriander	['4 tables	1 (2 g)		24	['Combine	['15-minute	['healthy', 'moroccan', 'low-sodium']				
24	505092	Silky Choc The perfec	['bitterswe	['1 1/2 cup	1 (174 g)		6	['Let all of t	['60-minute	['dessert', 'dinner']				
25	412809	Mushroom Sautéed m	['fresh mus	['16 ounce	1 (525 g)		4	['Preheat n	['time-to-n	['gluten-free', 'dinner', 'vegetarian', 'pasta']				

1.3. Data quality and license

The dataset meets requirements from 1.104 and 1.206, making it reliable and therefore usable for academic or non commercial purposes. Below are the specific details:

Quality: The dataset was pulled in from Food.com, a reliable recipe sharing platform, through Kaggle. The data is trustworthy, but multi valued fields like ingredients and tags need some preprocessing.

Detail: The dataset is very rich in data, but it tells us a lot about the way recipe consists are prepared, and what people do like to eat or how they search for recipes. This allows for fetching results of special types of such as vegetarian recipes under 500 calories.

Documentation: This dataset has details of every field and a detailed description of each field on its Kaggle page. Further metadata is inferred from the column names and structure.

Interrelation: Whenever you want to go a step further than just using one of these third party APIs, you can link your dataset to external APIs as Edamam or USDA nutritional databases for better analysis. Calorie and nutritional breakdowns of recipes would be available from these connections.

Discoverability: It was not difficult to locate the dataset because Kaggle is a popular source of open data. There were many other datasets in domain of recipe and food data but this one was superior to many because of its depth and data richness .

Use: This dataset can be used in a number of fields and areas, for instance, recipe suggestion services, nutritional breakdown applications, data representation web interfaces. However, it is brief on the details of every nutrient contained in the different foods that is more vital. It's practically impossible to provide certain answers to queries such as 'How many calories are in this food' or 'Is this food suitable for this diet?', without referring the client to data sources like Edamam.

License: The dataset is licensed under an open license for educational and research use, available under CC BY-NC-SA 4.0. This dataset does not use any restrictions listed against academic use.

1.4. Interest in the data

The Food.com Recipes with Search Terms and Tags is compelling because it is so detailed in information about the recipe itself, the ingredients, the search terms and tags entered by the user wanting to find the recipe. Due to its multidimensionality, it is useful for food ingredient usage; recipe complexity; consumer preference exploration, very relevant to the real world culinary trends. Advanced analysis supported by this dataset can uncover the most popular recipe tags or ingredient pairings, insights which would be of use to home cooks, nutritionists, and even recipe platforms. Its depth gives an opportunity to use advanced database and Visualization techniques, while remaining practical and intellectually exciting.

1.5. What answers we want from the data

This dataset provides a unique opportunity to analyze and categorize recipes based on ingredients, tags and user generated keywords. Below are some of the key questions this data can help answer:

Based on tags, what are the most popular vegetarian recipes?

What recipes we can find with specific ingredients, like it garlic or olive oil?

What is the most common tag on recipes with low calories?

What are the top 5 recipes with the highest number of steps?

What are the top 10 most used ingredients across all recipes?

Which tags are most commonly associated with recipes containing the top 3 most used ingredients?

What are the most frequently used tags for recipes containing 'chocolate'?

And which recipes are classified as 'brunch' or 'deserts'.

A well structured relational database and a user friendly web application provides efficient answers to these questions. Its richness of content, its open format, makes the dataset perfect for building more advanced features, like a recommendation engine or a recipe search tool. This data is important to see what development indicators are, because this CSV contains all the RAW data.

to store and display the data we must create a database and a web application which is an easier way to a user.

1.6. Data cleaning

I did data preprocessing and cleaning using python as follows:

Library Usage: One of its uses is it makes use of the pandas library, which is widely used for handling the tabular data and perform the data transformation efficiently.

Dataset Loading: A DataFrame is created and loads a dataset from a CSV file.

Cleaning and Exploding the Ingredients: The columns containing multi values for example ingredients column (stored as a list-like string) is cleaned, Ingredients are split into a string separated by single entries. Each ingredient is “exploded,” which means we put the cleaned ingredient into its own row, alongside its corresponding recipe identifier. Removing leading/trailing whitespace and removing empty rows. Characters not wanted, such as square brackets and quotes are removed.

Renaming Columns: For clarity, the columns are renamed, clearly renaming identifiers to something like "recipe ID" and ingredients to "ingredient name."

Creating Unique Identifiers: It is likely that each ingredient is given a unique identifier. This could be in order to convey ingredients separately in a database schema with ingredients as foreign tables to recipes.

Final Preparation for Database: The data is prepared in relational format and the processed data is easier to store in a database. I created a separate CSV for each table to make it easier to import the CSV into the tables database.

This is the code I wrote to achieve that:

```
] : import pandas as pd

# Load the dataset
df = pd.read_csv('database.csv')

# Display the first few rows
print(df.head())
```

	id	name \
0	96313	Grilled Garlic Cheese Grits
1	232037	Simple Shrimp and Andouille Jambalaya
2	41090	black-and-white bean salad
3	60656	Crock Pot Italian Zucchini
4	232047	Beef Stew With Dried Cherries

	description \
0	We love grits, this is another good way to ser...
1	Simple, easy and very tasty for when you are i...
2	NaN
3	This is a good recipe for weight watchers. It ...
4	This is a fabulous stew that came from one of ...

	ingredients \
0	['water', 'grits', 'salt', 'cheddar cheese', '...
1	['onion', 'red bell pepper', 'garlic cloves', '...
2	['white beans', 'canned black beans', 'tomatoe...
3	['zucchini', 'yellow squash', 'diced tomatoes'...
4	['beef stew meat', 'flour', 'salt', 'allspice'...

	ingredients_raw_str	serving_size	servings \
0	"4 cups water", "1 cup uncooked old f...	1 (155 g)	8
1	"1 medium onion, chopped coarse ", "1 m...	1 (366 g)	4
2	"1 cup canned white beans, rinsed and dra...	1 (807 g)	1
3	"2 zucchini, sliced ", "2 small yel...	1 (244 g)	4
4	"3 lbs beef stew meat", "3 tablespoons ...	1 (358 g)	8

	steps \
0	['I a sauce pan, bring water to a boil; slowly...
1	['In a food processor, pulse the onion, red pe...
2	['In a large bowl, combine beans, tomato, onio...
3	['Put all ingredients in the crock pot and coo...
4	['Preheat oven to 350°F.', "Cut beef into 1 in...

```

                                tags \
0 ['time-to-make', 'course', 'main-ingredient', ...
1 ['60-minutes-or-less', 'time-to-make', 'course...
2 ['15-minutes-or-less', 'time-to-make', 'course...
3 ['weeknight', 'time-to-make', 'course', 'main-...
4 ['time-to-make', 'course', 'main-ingredient', ...

                                search_terms
0 {'diabetic', 'low-calorie', 'vegetarian', 'low...
1 {'dinner', 'shrimp'}
2 {'vegetarian', 'salad', 'side', 'dinner', 'veg...
3 {'side', 'vegetarian', 'italian'}
4 {'dinner'}

: # Explode 'ingredients' column into separate rows
ingredients_data = df[['id', 'ingredients']].copy()
ingredients_data['ingredients'] = ingredients_data['ingredients'].str.strip("[]").str.replace("'", "")
ingredients_data = ingredients_data.explode('ingredients').reset_index(drop=True)
ingredients_data.columns = ['recipe_id', 'ingredient_name']

# Create a unique Ingredients table
unique_ingredients = ingredients_data[['ingredient_name']].drop_duplicates().reset_index(drop=True)
unique_ingredients['ingredient_id'] = unique_ingredients.index + 1

# Map ingredient_id to the exploded data
ingredients_data = ingredients_data.merge(unique_ingredients, on='ingredient_name')

# Keep only recipe_id and ingredient_id columns for recipe_ingredients.csv
recipe_ingredients_data = ingredients_data[['recipe_id', 'ingredient_id']]

# Save normalized tables
recipe_ingredients_data.to_csv('recipe_ingredients.csv', index=False)
unique_ingredients = unique_ingredients[['ingredient_id', 'ingredient_name']] # Reorder for consistency
unique_ingredients.to_csv('ingredients.csv', index=False)

```

```

]: import pandas as pd

# Load the CSV file
recipe_ingredients = pd.read_csv('recipe_ingredients.csv')

# Drop duplicates based on both recipe_id and ingredient_id
recipe_ingredients = recipe_ingredients.drop_duplicates(subset=['recipe_id', 'ingredient_id'])

# Save the cleaned file
recipe_ingredients.to_csv('recipe_ingredients.csv', index=False)

]: # Explode 'steps' column into individual rows
steps_data = df[['id', 'steps']].copy()

# Clean and split the 'steps' column
steps_data['steps'] = (
    steps_data['steps']
    .str.strip("[]") # Remove square brackets
    .str.replace("'", "") # Remove single quotes
    .str.split(',') # Split into list
)
steps_data = steps_data.explode('steps').reset_index(drop=True) # Explode into rows

# Remove leading and trailing spaces in 'steps'
steps_data['steps'] = steps_data['steps'].str.strip()

# Remove rows with missing or invalid step descriptions
steps_data = steps_data.dropna(subset=['steps']) # Remove rows where 'steps' is NaN
steps_data = steps_data[steps_data['steps'] != ""] # Remove rows where 'steps' is an empty string

# Add step numbers
steps_data['step_number'] = steps_data.groupby('id').cumcount() + 1

# Rename columns
steps_data.columns = ['recipe_id', 'step_description', 'step_number']

# Reorder columns to ensure correct order
steps_data = steps_data[['recipe_id', 'step_number', 'step_description']]

# Save normalized steps table
steps_data.to_csv('steps.csv', index=False)

```

```

: # Explode and normalize 'tags'
tags_data = df[['id', 'tags']].copy()

# Clean and split the 'tags' column
tags_data['tags'] = (
    tags_data['tags']
    .str.strip("{}") # Remove
    .str.replace("'", "") # Remove single quotes
    .str.split(",") # Split on commas
)

# Explode the list into separate rows
tags_data = tags_data.explode('tags').reset_index(drop=True)

# Remove Leading/trailing whitespace and drop empty rows
tags_data['tags'] = tags_data['tags'].str.strip()
tags_data = tags_data.dropna(subset=['tags'])
tags_data = tags_data[tags_data['tags'] != ""] # Drop rows with empty tags

# Rename columns
tags_data.columns = ['recipe_id', 'tag_name']

# Create a unique Tags table
unique_tags = tags_data[['tag_name']].drop_duplicates().reset_index(drop=True)

# Save normalized tags
tags_data.to_csv('recipe_tags.csv', index=False)
unique_tags.to_csv('tags.csv', index=False)

```

```

[29]: # Explode and normalize 'search_terms'
search_terms_data = df[['id', 'search_terms']].copy()

# Clean and split the 'search_terms' column
search_terms_data['search_terms'] = (
    search_terms_data['search_terms']
    .str.strip("{}") # Remove curly braces
    .str.replace("'", "") # Remove single quotes
    .str.split(",") # Split on commas
)

# Explode the list into separate rows
search_terms_data = search_terms_data.explode('search_terms').reset_index(drop=True)

# Rename columns
search_terms_data.columns = ['recipe_id', 'search_term']

# Remove Leading/trailing whitespace in search terms
search_terms_data['search_term'] = search_terms_data['search_term'].str.strip()

# Create a unique SearchTerms table
unique_search_terms = search_terms_data[['search_term']].drop_duplicates().reset_index(drop=True)

# Save normalized search terms
search_terms_data.to_csv('recipe_search_terms.csv', index=False)
unique_search_terms.to_csv('search_terms.csv', index=False)

```

```

[5]: recipes_data = df[['id', 'name', 'description', 'serving_size', 'servings']].copy()
recipes_data.columns = ['recipe_id', 'name', 'description', 'serving_size', 'servings']

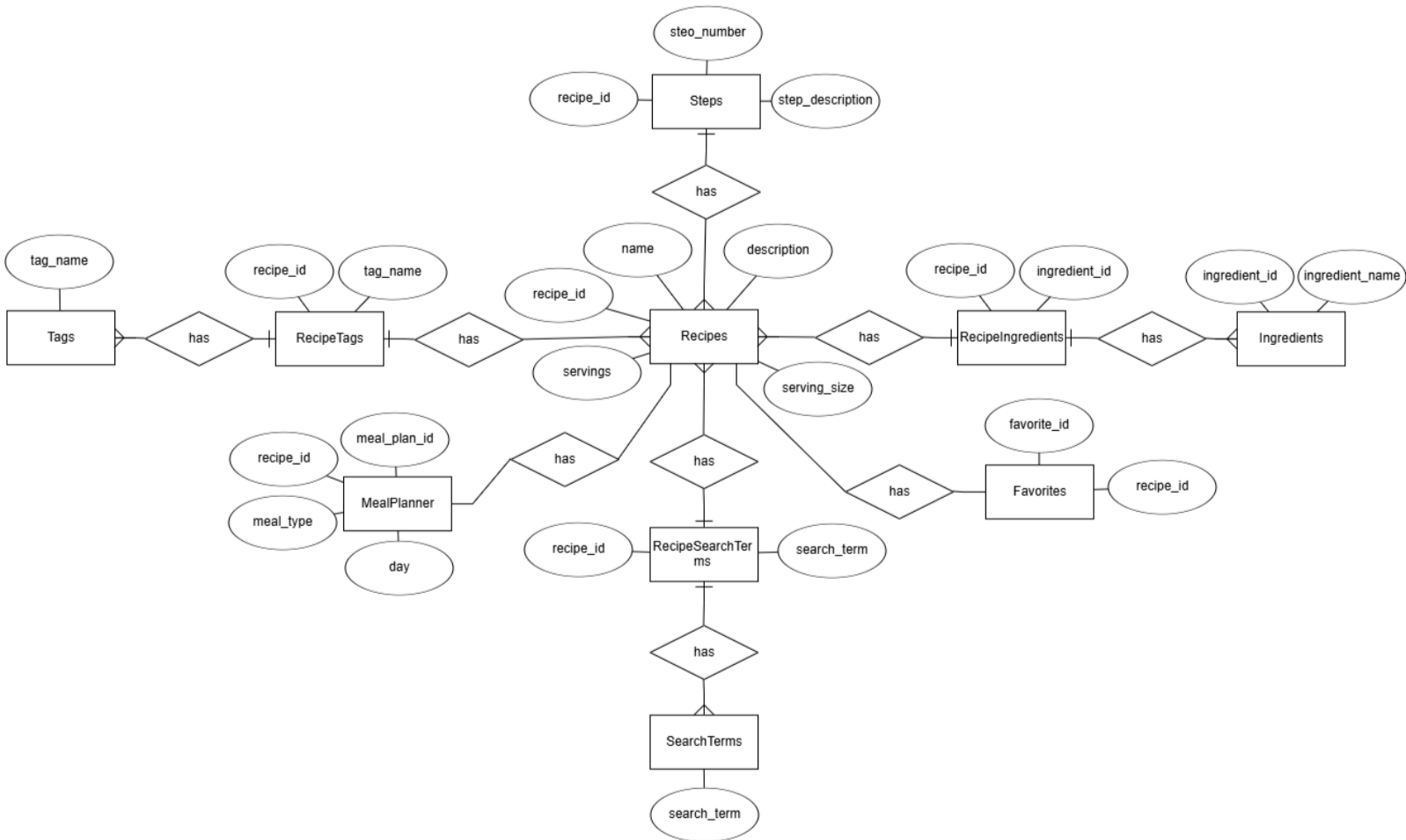
# Save normalized recipes
recipes_data.to_csv('recipes.csv', index=False)

```

2. Modelling the data

2.1 E/R model

Here's an ER model of how the database is structured. I have drawn it using ERD plus <https://erdplus.com/> you will need to zoom in a little as the diagram is large and is better shown when zoomed in. You can also access it through this link <https://ibb.co/w7fcP4J>



2.2 Listing database tables and fields

I used dbdocs to construct a diagram with all table information. This link <https://dbdocs.io/nancy%20mahmoud/last?view=relationships> provides a complete version of this. The PK symbol marks all the primary key fields and the NN means it has a not null constrain.

The table Recipes have 5 fields, recipe_id, name, description, serving_size and servings. More details of the table are below:

▼ Fields (5)

Name	Type	Settings	References
recipe_id	int	not_null PK	RecipeIngredients.recipe_id Steps.recipe_id RecipeSearchTerms.recipe_id RecipeTags.recipe_id +1 more
name	varchar	not_null	
description	text	null	
serving_size	varchar	not_null	
servings	int	not_null	

The table Ingredients has 2 fields, ingredient_id and ingredient_name. More details of the table are below:

▼ Fields (2)

Name	Type	Settings	References
ingredient_id	int	not_null PK	RecipeIngredients.ingredient_id
ingredient_name	varchar	null	

The table RecipeIngredients has 2 fields, recipe_id and ingredient_id. More details of the table are below:

▼ Fields (2)

Name	Type	Settings	References
recipe_id	int	not_null	Recipes.recipe_id
ingredient_id	int	not_null	Ingredients.ingredient_id

The table Steps has 3 fields recipe_id, step_number and step_description. More details of the table are below:

▼ Fields (3)

Name	Type	Settings	References
recipe_id	int	not_null	➔ Recipes.recipe_id
step_number	int	not_null	
step_description	text	not_null	

The table SearchTerms has 1 field which is search_term. More details of the table are below:

▼ Fields (1)

Name	Type	Settings	References
search_term	varchar	not_null PK	➔ RecipeSearchTerms.search_term

The table RecipeSearchTerms has 2 fields recipe_id and search_term. More details of the table are below:

▼ Fields (2)

Name	Type	Settings	References
recipe_id	int	not_null	➔ Recipes.recipe_id
search_term	varchar	not_null	➔ SearchTerms.search_term

The table Tags has 1 field which is tag_name. More details of the table are below:

▼ Fields (1)

Name	Type	Settings	References
tag_name	varchar	not_null PK	↔ RecipeTags.tag_name

The table RecipeTgas has 2 fields recipe_id and tag_name. More details of the table are below:

▼ Fields (2)

Name	Type	Settings	References
recipe_id	int	not_null	→ Recipes.recipe_id
tag_name	varchar	not_null	→ Tags.tag_name

The table Favorites has 2 fields favorite_id and recipe_id. More details of the table are below:

▼ Fields (2)

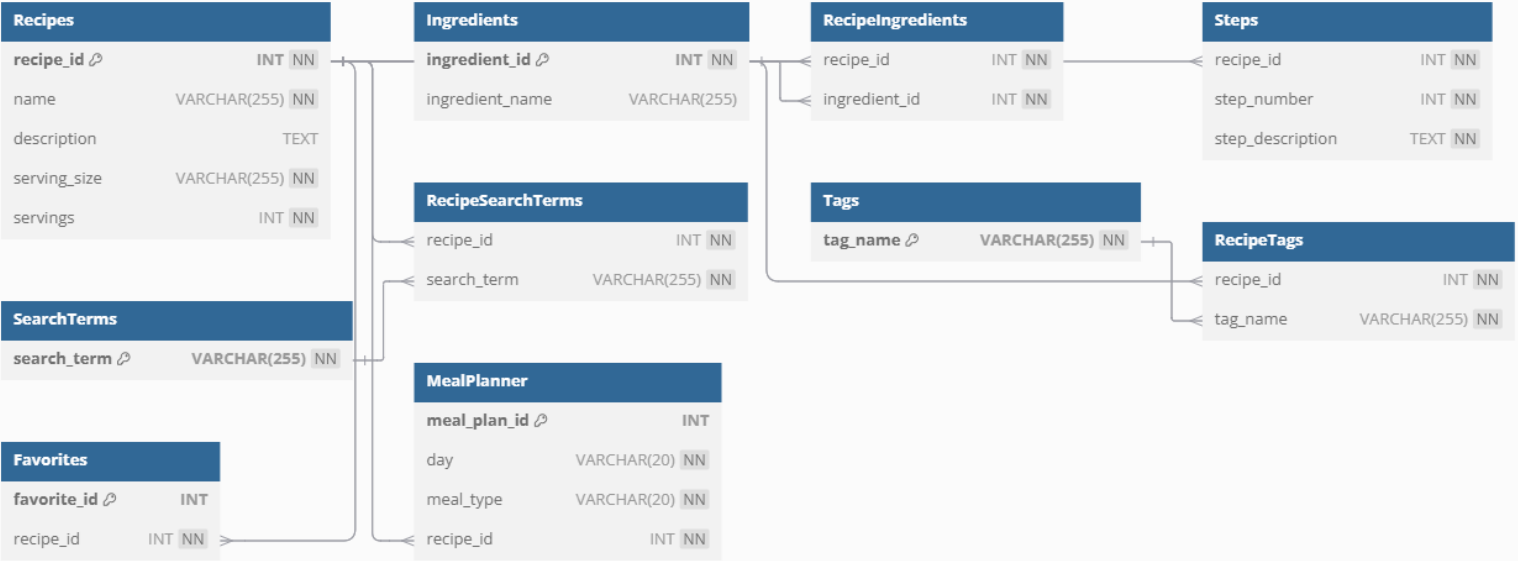
Name	Type	Settings	References
favorite_id	INT	null PK increment	
recipe_id	INT	not_null unique	→ Recipes.recipe_id

Lastly, the table MealPlanner has 4 fields meal_plan_id, day, meal_type and recipe_id. More details of the table are below:

▼ Fields (4)

Name	Type	Settings	References
meal_plan_id	INT	null PK increment	
day	VARCHAR(20)	not_null	
meal_type	VARCHAR(20)	not_null	
recipe_id	INT	not_null	➔ Recipes.recipe_id

The tables are related in the same way that the logical ER diagram above shows. I created a detailed database diagram using dbdiagram.io to ensure clarity. This can be viewed using this link <https://dbdocs.io/embed/4f214b4162fe47fb2a0bfa16d60d4608/f49a3509f75b41d3becb3dac0363ee93> . A summary is provided below.



2.3 Review of database normalisation

I created the database in the third Normal Form (3NF). To ensure this normalization level, I adhered to the following requirements:

First Normal Form (1NF):

All columns hold atomic (value that can be subdivided into smaller indivisible pieces) values and there are no repeating groups or arrays.

It is true that each table has a primary key to uniquely identify its rows.

Second Normal Form (2NF):

All the non key attributes have been made redundant, which mean that every non key attribute is fully dependent on the entire key (for tables with composite keys).

Third Normal Form (3NF):

All non-key attributes are directly dependent on the primary key and are independent of each other.

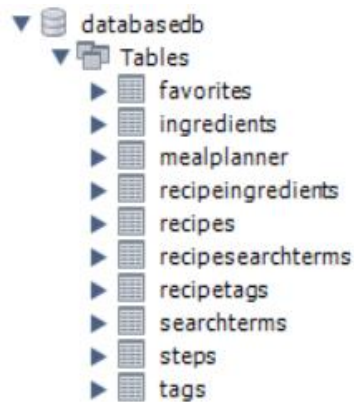
There are no transitive dependencies between non key attributes

To achieve this last balance I decided to normalize the database to 3NF so that it would eliminate redundancy without making it unpractical. While more normalization of the database could be done to higher forms such as 4NF, I decided to stop at 3NF because the analysis of the dataset and project goals lead me to believe that 3NF was the best choice for normalization. A measure this close to normalizes redundancy and reasonable query complexity at the price of scalability.

3. Create the database

3.1 SQL commands and database

The database contains tables Recipes, Ingredients, RecipeIngredients, Steps, SearchTerms, RecipeSearchTerms, Tags, RecipeTags, Favorites and MealPlanner as shown here:



This command was used to create the database:

```
-- Create the Database  
CREATE DATABASE databaseDB;
```

The following commands were used to create the 10 tables of the database and next to the commands there is a part of the table shown. Here is how many columns and rows they have:

The Recipe table has 5 columns and 9999 rows.

The Ingredients table has 2 columns and 7527 rows.

The RecipeIngredients table has 2 columns and 90268 rows.

The Steps table has 3 columns and 119360 rows.

The SearchTerms table has one column and 92 rows.

The RecipeSearchTerms table has 2 columns and 29048 rows.

The Tags table has one column and 482 rows.

The RecipeTags table has 2 columns and 168135 rows.

The MealPlanner table has 4 columns and their rows depend on the user entries.

The Favorites table has 2 columns and their rows depend on the user entries.

```
-- Create the Recipes table
CREATE TABLE Recipes (
  recipe_id INT PRIMARY KEY NOT NULL,
  name VARCHAR(255) NOT NULL,
  description TEXT NULL,
  serving_size VARCHAR(50) NOT NULL,
  servings INT NOT NULL
);
```

recipe_id	name	description	serving_size	servings
39	Biryani	Delhi, India	1 (799 g)	6
51	Chai Tea		1 (332 g)	4
147	Campfire Orange Cake	When I was a child, my mother made orange ca...	1 (134 g)	6
169	Ancho Chile Rub	Ancho Chile Rub	1 (73 g)	1
174	Chicken Supreme with Mushrooms	Here is one for the freezer.	1 (268 g)	4
232	Chocolate Cream Meringue Pie	Chocolate	1 (169 g)	8
292	Curried Vegetables	Curried Vegetables	1 (199 g)	6
354	Kahlua	Kahlua	1 (2676 g)	1
355	Apple Crisp	Apple Crisp	1 (85 g)	8
360	Baked Zucchini Frittatas		1 (523 g)	2

```
-- Create the Ingredients table
CREATE TABLE Ingredients (
  ingredient_id INT PRIMARY KEY NOT NULL,
  ingredient_name VARCHAR(255) NULL
);
```

ingredient_id	ingredient_name
1	water
2	grits
3	salt
4	cheddar cheese
5	garlic
6	olive oil

```
-- Create the RecipeIngredients table
CREATE TABLE RecipeIngredients (
  recipe_id INT NOT NULL,
  ingredient_id INT NOT NULL,
  UNIQUE(recipe_id, ingredient_id),
  FOREIGN KEY (recipe_id) REFERENCES
Recipes(recipe_id),
  FOREIGN KEY (ingredient_id) REFERENCES
Ingredients(ingredient_id)
);
```

recipe_id	ingredient_id
51	1
354	1
6809	1
9787	1
13041	1
15641	1
16467	1
16918	1
28155	1
36824	1

```
-- Create the Steps table
CREATE TABLE Steps (
  recipe_id INT NOT NULL,
  step_number INT NOT NULL,
  step_description TEXT NOT NULL
  UNIQUE(recipe_id, step_number),
  FOREIGN KEY (recipe_id) REFERENCES
Recipes(recipe_id)
);
```

recipe_id	step_number	step_description
39	1	Soak saffron in warm milk for 5 minutes and pur...
39	2	Add chiles
39	3	onions
39	4	ginger
39	5	garlic
39	6	cloves
39	7	peppercorns
39	8	cardamom seeds
39	9	cinnamon
39	10	coriander and cumin seeds

```
-- Create the SearchTerms table
CREATE TABLE SearchTerms (
  search_term VARCHAR(255) PRIMARY KEY
NOT NULL
);
```

search_term
american
appetizer
baked
barbecue
beef
butter

```
-- Create the RecipeSearchTerms table
CREATE TABLE RecipeSearchTerms (
  recipe_id INT NOT NULL,
  search_term VARCHAR(255) NOT NULL,
  UNIQUE(recipe_id, search_term),
  FOREIGN KEY (recipe_id) REFERENCES
Recipes(recipe_id),
  FOREIGN KEY (search_term) REFERENCES
SearchTerms(search_term)
);
```

	recipe_id	search_term
▶	63426	american
	216225	american
	291758	american
	518631	american
	503	appetizer
	2720	appetizer
	3770	appetizer
	4387	appetizer
	4784	appetizer
	5098	appetizer
	5227	appetizer

```
-- Create the Tags table
CREATE TABLE Tags (
  tag_name VARCHAR(255) PRIMARY KEY NOT
NULL
);
```

	tag_name
▶	1-day-or-more
	15-minutes-or-less
	3-steps-or-less
	30-minutes-or-less
	4-hours-or-less
	5-ingredient-or-less

```
-- Create the RecipeTags table
CREATE TABLE RecipeTags (
  recipe_id INT NOT NULL,
  tag_name VARCHAR(255) NOT NULL,
  UNIQUE(recipe_id, tag_name),
  FOREIGN KEY (recipe_id) REFERENCES
Recipes(recipe_id),
  FOREIGN KEY (tag_name) REFERENCES
Tags(tag_name)
);
```

	recipe_id	tag_name
▶	658	1-day-or-more
	2561	1-day-or-more
	9801	1-day-or-more
	12340	1-day-or-more
	13093	1-day-or-more
	15110	1-day-or-more
	28321	1-day-or-more
	30716	1-day-or-more
	32279	1-day-or-more
	34555	1-day-or-more

```
-- Create table Favorites
CREATE TABLE Favorites (
  favorite_id INT AUTO_INCREMENT PRIMARY
KEY,
  recipe_id INT NOT NULL,
  UNIQUE (recipe_id),
  FOREIGN KEY (recipe_id) REFERENCES
Recipes(recipe_id) ON DELETE CASCADE
);
```

```
-- Create table Favorites
CREATE TABLE Favorites (
  favorite_id INT AUTO_INCREMENT PRIMARY
KEY,
  recipe_id INT NOT NULL,
  UNIQUE (recipe_id),
  FOREIGN KEY (recipe_id) REFERENCES
Recipes(recipe_id) ON DELETE CASCADE
);
```

3.2 Data migration from CSV to database

I created a migration script to move the data from the CSV file to the database. It is a script in the project's root folder, migration.js. To run the migration, use the terminal command:

-node migration.js.

An explanation of the code is provided in the file comments. After the migration finishes it handles errors and shows a message.

The code is discussed in the file's comments. It handles errors and displays a message after the migration is complete. To migrate all the data from the CSV into the MySQL database it took only 4 minutes.

3.3 Critical reflection

Generally speaking, I believe that the overall database structure lines up nicely with my use case and my data and implementation issues and challenges was Ensuring all data was inserted correctly across related tables required using transactions in the migration script as such errors in my migration process could induce rollbacks that create a delayed importation and requiring debugging. Ingredients, tags, and search_terms were multi valued fields that were stored as lists or arrays in a single column. This needed lots of pre processing to split and clean data before importing it into normalized tables and really complex queries in the normalized format now require multi table joins so performance was affected while performing them, I tried a really complex query and it took about 15 mins to be performed but this isnt really affecting queries much it is just with really complex queries. The normalized database structure aligns well with the research questions:

Finding Popular Recipes: As by a popular tag (e.g., "vegetarian", "low-calorie", etc), the tag that is stored in the RecipeTags table is easy to identify.

Ingredient-Based Queries: Queries that find recipes that employ a particular ingredient (e.g., 'garlic', 'olive oil') are supported efficiently using the RecipeIngredients table.

Tag Insights: If tags are separated into a relational structure, analysis can be performed of frequently used tags for each recipe type.

3.4 List SQL queries for questions

1. Based on tags, what are the most popular vegetarian recipes?

```
SELECT r.name AS Recipe, COUNT(rt.tag_name) AS Popularity
FROM Recipes r
JOIN RecipeTags rt ON r.recipe_id = rt.recipe_id
WHERE rt.tag_name = 'vegetarian'
GROUP BY r.recipe_id
ORDER BY Popularity DESC
LIMIT 5;
```

The RecipeTags table has recipes that have been identified as "vegetarian". It simply counts the number of times the tag 'vegetarian' has been used in a recipe, in order to calculate how popular each vegetarian recipe is. They group recipes by their recipe_id and sort by descending order of popularity. The top 5 most popular vegetarian recipes are returned.

2. What recipes can we find with specific ingredients, like garlic or olive oil?

```
SELECT i.ingredient_name AS Ingredient,
       COUNT(r.recipe_id) AS RecipeCount,
       GROUP_CONCAT(r.name SEPARATOR ', ') AS RecipeNames
FROM Recipes r
JOIN RecipeIngredients ri ON r.recipe_id = ri.recipe_id
JOIN Ingredients i ON ri.ingredient_id = i.ingredient_id
WHERE i.ingredient_name IN ('garlic', 'olive oil')
GROUP BY i.ingredient_name;
```

The recipes found by the query contain the ingredients "garlic" or "olive oil". It shows us how many recipes feature each one of these ingredients (RecipeCount) and what the names are (RecipeNames). Each ingredient is grouped by name, and the recipe names along with the count are shown.

3. What is the most common tag on recipes with low calories?

```
SELECT rt.tag_name AS Tag, COUNT(*) AS Count
  FROM RecipeTags rt
 JOIN Recipes r ON r.recipe_id = rt.recipe_id
 WHERE rt.tag_name = 'low-calorie'
 GROUP BY rt.tag_name
 ORDER BY Count DESC
 LIMIT 1;
```

This time the query counts how many recipes are marked as "low calorie" in RecipeTags table. They are grouped and sorted on tags frequency in descending order. The tag 'low-calorie' returns as being most commonly associated with the 'low-calorie' recipes.

4. What are the top 5 recipes with the highest number of steps?

```
SELECT r.name AS Recipe, COUNT(s.step_number) AS Steps
  FROM Recipes r
 JOIN Steps s ON r.recipe_id = s.recipe_id
 GROUP BY r.recipe_id
 ORDER BY Steps DESC
 LIMIT 5;
```

The number of steps for each recipe is calculated with the query simply looking up the number of rows in the Steps table. Recipes are clustered first by their recipe_id and then sorted in descending order by the number of steps in them. We return the top 5 recipes with the most steps.

5. What are the top 10 most used ingredients across all recipes?

```
SELECT i.ingredient_name AS Ingredient, COUNT(ri.ingredient_id) AS `Usage`
  FROM Ingredients i
 JOIN RecipeIngredients ri ON i.ingredient_id = ri.ingredient_id
 GROUP BY i.ingredient_name
 ORDER BY `Usage` DESC
 LIMIT 10;
```

It counts how many recipes use each ingredient as recorded in the RecipeIngredients table. They are grouped by their name and sorted based on usage order, descending. Results returned are the top 10 most frequent ingredients they use.

6. Which tags are most commonly associated with recipes containing the top 3 most used ingredients?

```
WITH TopIngredients AS (  
    SELECT i.ingredient_id, i.ingredient_name, COUNT(ri.ingredient_id) AS  
    `Usage`  
    FROM Ingredients i  
    JOIN RecipeIngredients ri ON i.ingredient_id = ri.ingredient_id  
    GROUP BY i.ingredient_id  
    ORDER BY `Usage` DESC  
    LIMIT 3  
)  
SELECT  
    rt.tag_name AS Tag,  
    COUNT(rt.recipe_id) AS TagCount  
FROM RecipeTags rt  
JOIN RecipeIngredients ri ON rt.recipe_id = ri.recipe_id  
JOIN TopIngredients ti ON ri.ingredient_id = ti.ingredient_id  
GROUP BY rt.tag_name  
ORDER BY TagCount DESC  
LIMIT 20;
```

The top 3 most used ingredients can be identified in a Common Table Expression (CTE). The query then goes and looks for tags that are relevant to the recipes which contain these top 3 ingredients. Name of tags are grouped and sorted by frequency in descending order. It returns the top 20 most commonly associated tags.

7. What are the most frequently used tags for recipes containing 'chocolate'?

```
SELECT  
    t.tag_name AS Tag,  
    COUNT(*) AS Frequency  
FROM RecipeIngredients ri  
JOIN Ingredients i ON ri.ingredient_id = i.ingredient_id  
JOIN RecipeTags rt ON ri.recipe_id = rt.recipe_id  
JOIN Tags t ON rt.tag_name = t.tag_name  
WHERE i.ingredient_name LIKE '%chocolate%'  
GROUP BY t.tag_name  
ORDER BY Frequency DESC  
LIMIT 10;
```

It finds recipes in which “chocolate” is an ingredient. It goes ahead and counts the amount of tags that are tied to these recipes from the RecipeTags table. Each name groups_tags in descending

order of frequency. When searching for the word “chocolate,” it returns the top 10 most frequently used tags.

8. Which recipes are classified as 'brunch' or 'desserts'?

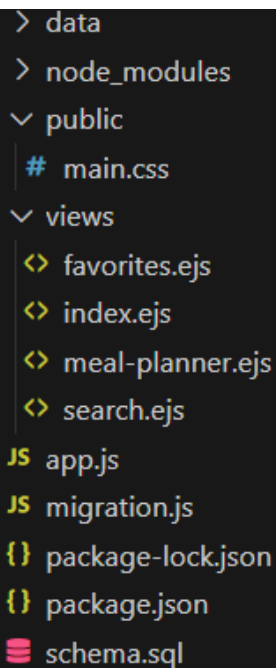
```
SELECT rt.tag_name AS MealType,  
       COUNT(r.recipe_id) AS RecipeCount,  
       GROUP_CONCAT(r.name SEPARATOR ', ') AS RecipeNames  
FROM Recipes r  
JOIN RecipeTags rt ON r.recipe_id = rt.recipe_id  
WHERE rt.tag_name IN ('brunch', 'desserts')  
GROUP BY rt.tag_name;
```

It queries recipes and filters based on which category tags, if either brunch or desserts. It counts the number of recipes for every meal type and names them. The recipes of all meal types are grouped by name.

4. Create a simple web application

I used Node.js as the runtime environment, Express.js as a backend framework for handling HTTP requests and routing, MySQL2 as it is library for interacting with a MySQL database and Chart.js as it is a popular JavaScript library for rendering interactive and responsive charts on the front end using HTML5 <canvas> elements. As those are the technologies I am most familiar with, that is why I used them.

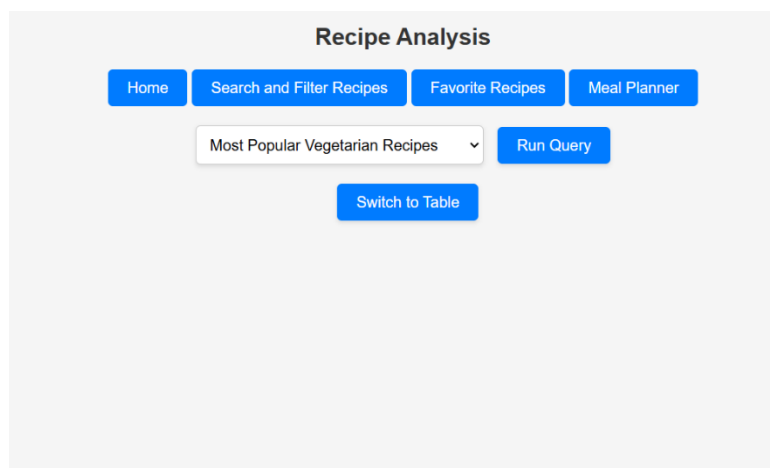
The structure of the website is the following:



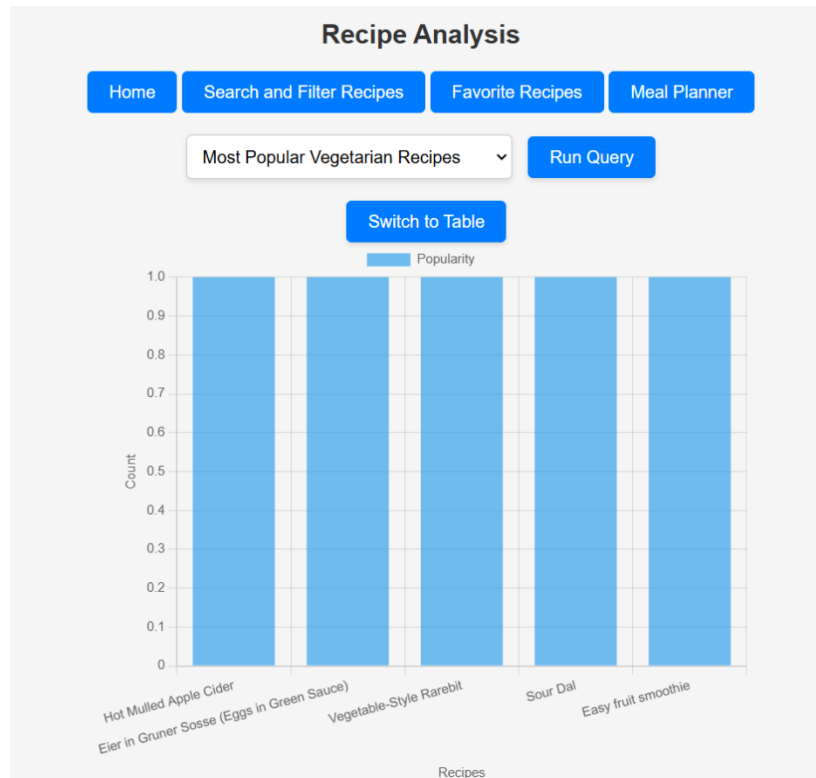
```
> data  
> node_modules  
v public  
  # main.css  
v views  
  <> favorites.ejs  
  <> index.ejs  
  <> meal-planner.ejs  
  <> search.ejs  
JS app.js  
JS migration.js  
{ } package-lock.json  
{ } package.json  
📄 schema.sql
```

To run the website, you need to write **node app.js**. Here are some images of the simple web application running on my own local environment. The website consists of four pages home, search and filter recipes, favorite recipes and meal planner. The home viewing page where you select the query you want to run from the drop down menu and after that it directs you to the page of the query you selected showing visualizations of the top results and there is an option to show table of the results to know more information so you can switch back and forth between the graph and table. I have also made hovering effect so when you hover over a bar in any graph you can see more information about it and when you click on the color box of a certain label it cancels the label and shows you the graph of the other label only. Then, from the home page you can navigate to search and filter recipes page, favorite recipes page and meal planner page. In the search and filter recipes page you can search for recipes by choosing filters such as tags, search term and ingredients, you can also sort it alphabetically or by serving size. You can also clear all the filters and when the recipes are displayed after searching you can add a recipe to favorites by clicking the heart icon and you can also remove it by clicking the icon again. The favorite recipes page also have the search and filtering feature as the search page but here you only search in your favorite recipes not the full set of recipes. You can also remove recipes from favorite by clicking the remove from favorites button. Lastly, the meal planner page where you can plan your meal for a whole week by adding a recipe for each day in the week but since the user doesn't know the recipes names they can only choose from their favorite recipes to make it easier. You can also clear them all by clicking one button.

Home page



Most Popular Vegetarian Recipes graph



Most Popular Vegetarian Recipes table

Recipe Analysis

Home Search and Filter Recipes Favorite Recipes Meal Planner

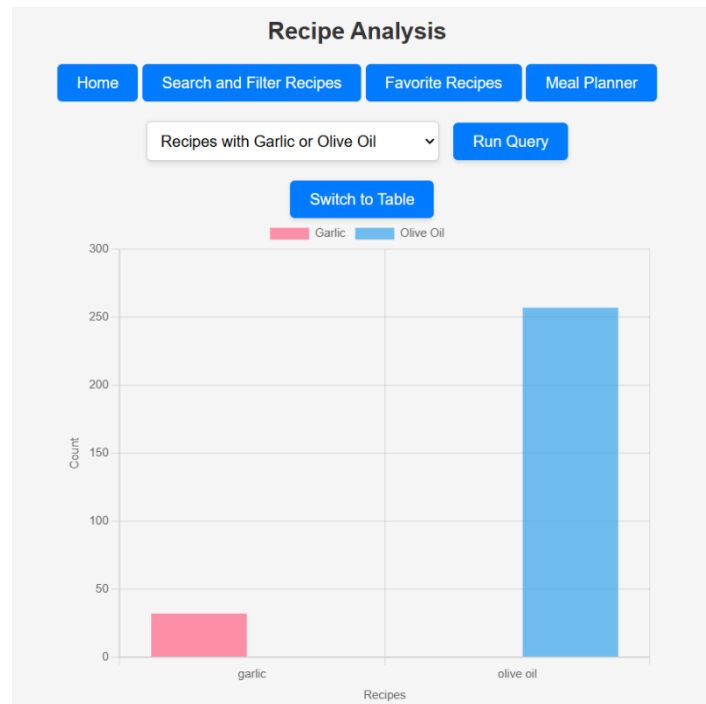
Most Popular Vegetarian Recipes Run Query

Switch to Graph

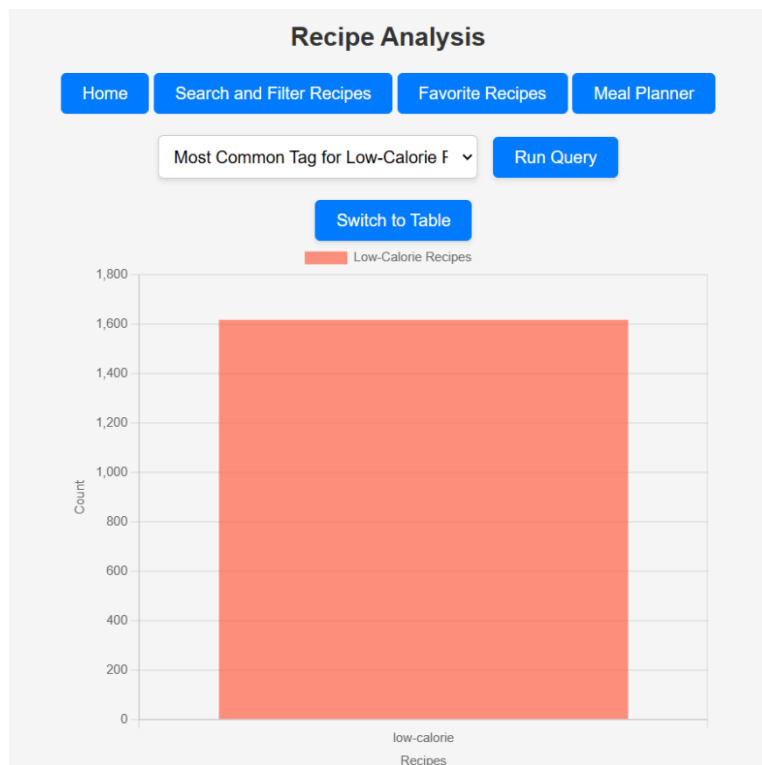
Recipe Data Table

Label	Value
Hot Mulled Apple Cider	1
Eier in Gruner Sosse (Eggs in Green Sauce)	1
Vegetable-Style Rarebit	1
Sour Dal	1
Easy fruit smoothie	1

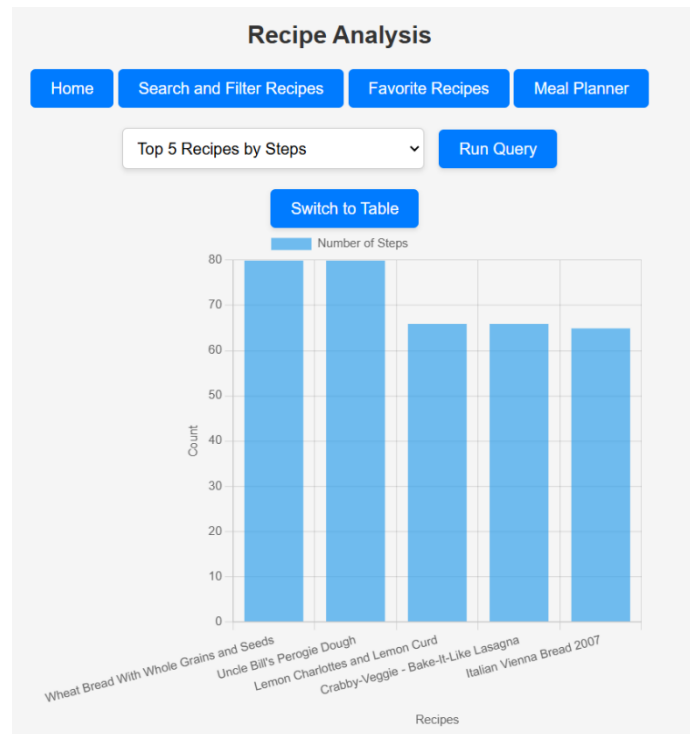
Recipes with Garlic or Olive Oil graph



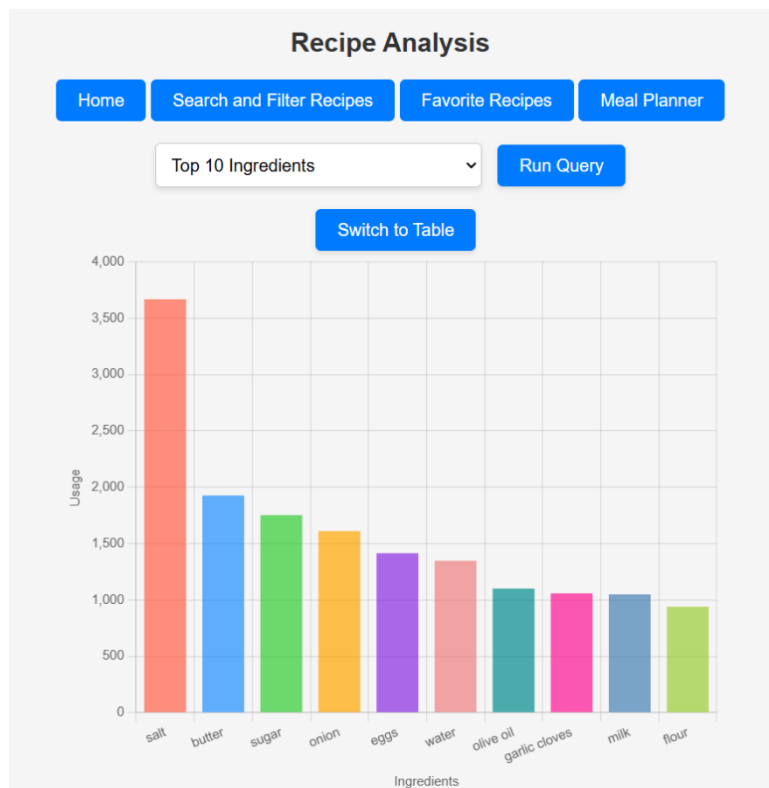
Most Common Tag for Low-Calorie Recipes graph



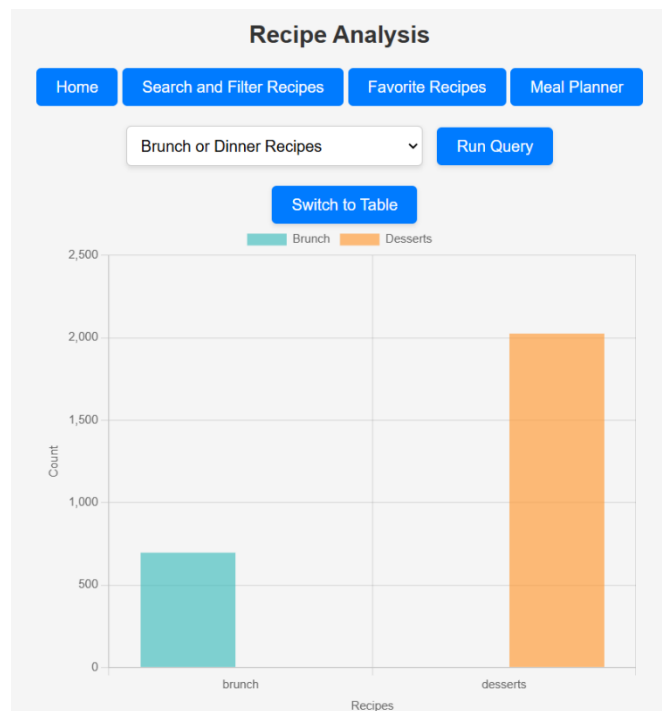
Top 5 Recipes by Steps graph



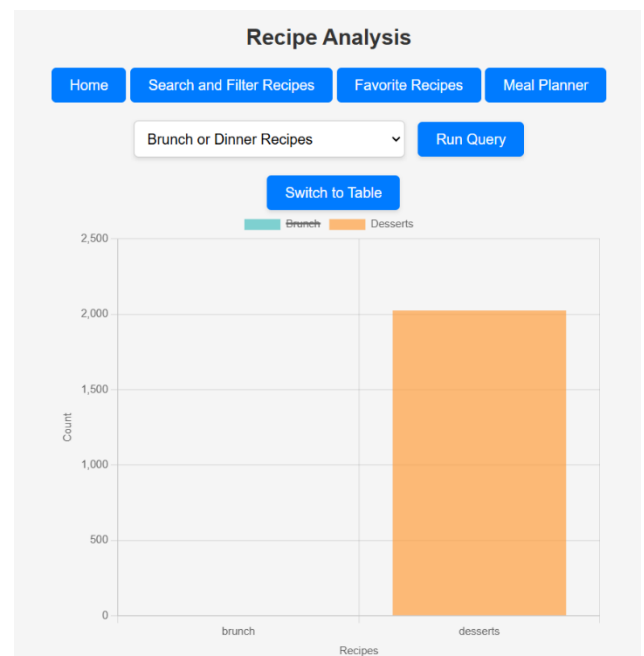
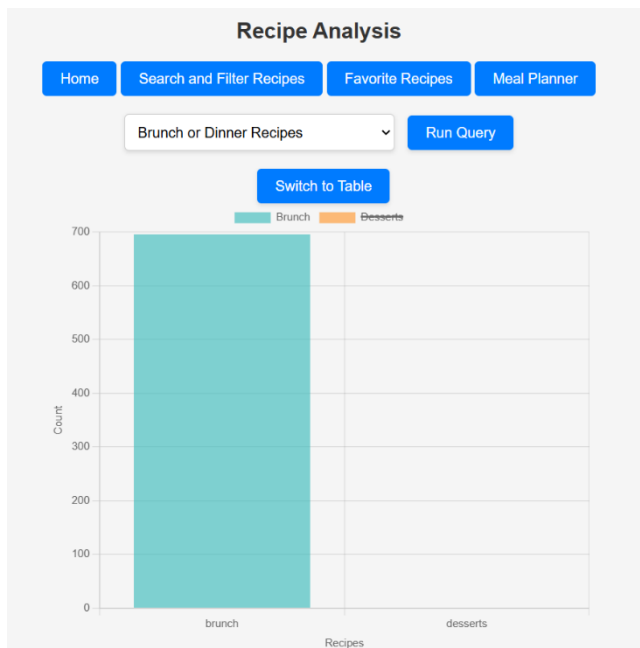
Top 10 Ingredients graph



Brunch or Dinner Recipes graph



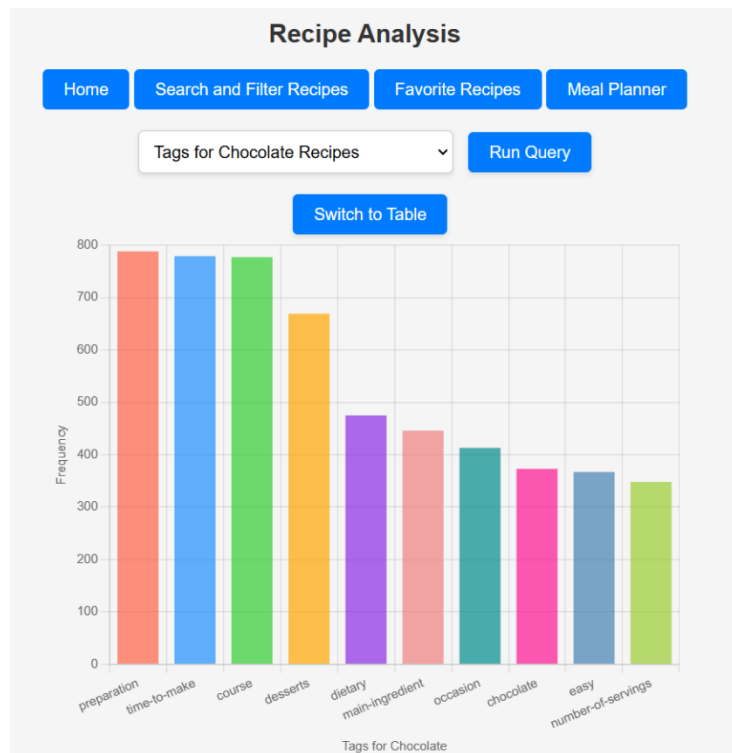
Brunch or Dinner Recipes graphs cancelling one in each by clicking the color box to display it alone



Top Tags for Top Ingredients graph



Tags for Chocolate Recipes graph



Search and Filter Recipes

[Home](#)[Meal Planner](#)[Favorites](#)

Filter by tag

Filter by search term

Filter by ingredient

Sort by

Search

Clear Filters

Search and Filter Recipes

Home

Meal Planner

Favorites

15-minutes-or-less

appetizer

salt

Alphabetical

Search

Clear Filters

Amish Pickled Eggs and Beets

Description: This is an easy recipe that my grandmother used to serve in the spring. My brother still refers to this dish as "Easter eggs". Slices of onion or hot pepper may also be added to the pickling liquid with the beets and eggs.

Servings: 6

Ingredients: salt

Artichoke Caviar

Description: This tart and tangy appetizer is made from finely chopped artichokes combined with other tasty Greek favorites.

Servings: 1

Ingredients: salt

Black Bean Dip

Description: One of my adopted recipes. Have not made yet:(Edited after review.)

Servings: 4

Ingredients: salt

Favorites Recipes Home page

Your Favorite Recipes

[Search and Filter Recipes](#)[Meal Planner](#)[Home](#)

Filter by tag

Filter by search term

Filter by ingredient

Sort by

Search

Clear Filters

Pineapple, Date and Nut Drops

Description: These are good for the kids' lunch boxes.

Servings: 1

Ingredients: baking powder, baking soda, brown sugar, butter, dates, egg yolk, nuts, pineapple, salt, white sugar, flour

Remove from Favorites

Yellow Squash Puffs

Description: I have made this recipe a thousand times and we just love it. It came up for adoption and I jumped on it. Goes great with just about any main dish.

Servings: 4

Ingredients: all-purpose flour, baking powder, cornmeal, egg, onion, salt, vegetable oil, yellow squash

Remove from Favorites

Curried Beef and Chicken Satay

Description: Every country or region has at least one dish that becomes symbolic. For SE Asia, symbolic foods would have to include satay. Popular in Malaysia, Singapore and Indonesia, it's a great treat when served hot from a charcoal grill. There are numerous ways of preparing this dish; the recipe below is only one of them.

Servings: 4

Ingredients: bamboo skewer, chicken, chile, chili sauce, coconut milk, curry powder, garlic cloves, honey, lemon juice, onions, peanut butter, salt, soy sauce, worcestershire sauce, beef

Meal Planner Home page

Personalized Meal Planner

[Home](#)[Favorite Recipes](#)[Search and Filter Recipes](#)

Clear All

Day	Breakfast	Lunch	Dinner
Monday	<div>Add Recipe</div>	<div>Add Recipe</div>	<div>Add Recipe</div>
Tuesday	<div>Add Recipe</div>	<div>Add Recipe</div>	<div>Add Recipe</div>
Wednesday	<div>Add Recipe</div>	<div>Add Recipe</div>	<div>Add Recipe</div>
Thursday	<div>Add Recipe</div>	<div>Add Recipe</div>	<div>Add Recipe</div>
Friday	<div>Add Recipe</div>	<div>Add Recipe</div>	<div>Add Recipe</div>
Saturday	<div>Add</div>	<div>Add</div>	<div>Add</div>

After clicking add recipe in the Meal Planner page

Personalized Meal Planner

Select a Recipe

Pineapple, Date and Nut Drops

These are good for the kids' lunch boxes.

Select

Yellow Squash Puffs

I have made this recipe a thousand times and we just love it. It came up for adoption and I jumped on it. Goes great with just about any main dish.

Select

Curried Beef and Chicken Satay

Every country or region has at least one dish that becomes symbolic. For SE Asia, symbolic foods would have to include satay. Popular in Malaysia, Singapore and Indonesia, it's a great treat when served hot from a charcoal grill. There are numerous ways of preparing this dish; the recipe below is only one of them.

Select

Campfire Orange Cake

When I was a child, my mother made orange cake every time we went camping. She would scoop out an orange, fill it with (i think) white jiffy cake mix, wrap it in foil and put it in the campfire to bake. They were wonderful!

Select

The Meal Planner Page After selecting a recipe from your favorite recipes

Personalized Meal Planner

HomeFavorite RecipesSearch and Filter Recipes

Clear All

Day	Breakfast	Lunch	Dinner
Monday	Pineapple, Date and Nut Drops	Curried Beef and Chicken Satay	Curried Beef and Chicken Satay
Tuesday	Add Recipe	Add Recipe	Add Recipe
Wednesday	Add Recipe	Add Recipe	Add Recipe
Thursday	Add Recipe	Add Recipe	Add Recipe
Friday	Yellow Squash Puffs	Add Recipe	Add Recipe
Saturday	Add Recipe	Add Recipe	Add Recipe
Sunday	Add Recipe	Add Recipe	Add Recipe

References

<https://www.kaggle.com/datasets/shuyangli94/foodcom-recipes-with-search-terms-and-tags>

<https://erdplus.com/>

<https://dbdocs.io/>

<https://www.chartjs.org/>

<https://www.geeksforgeeks.org/normal-forms-in-dbms/>

<https://w3schools.com/sql/default.asp>

<https://docs.anaconda.com/ae-notebooks/user-guide/basic-tasks/apps/jupyter/>