Tea Time Magazine

Sydney Lum

CSI 3450

April 25, 2025

Welcome to the Tea Time Magazine

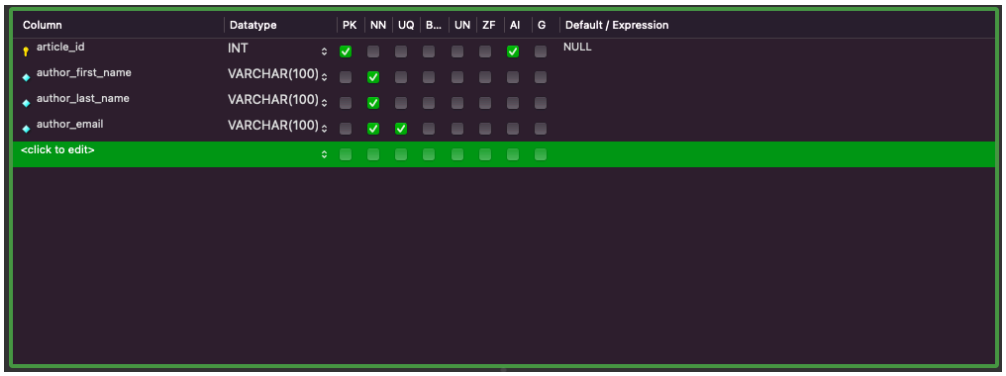
This site is a creative hub for recipes, articles, fashion, sewing, and home decor. Users can explore detailed content, view photos, and find inspiration across different lifestyle topics. Organized and easy to navigate, the site connects ideas, styles, and projects in one place.

The target audience for this site includes individuals interested in cooking, fashion, sewing, home decor, and lifestyle inspiration. It appeals to creative hobbyists, DIY enthusiasts, home cooks, and readers looking for fresh ideas and projects. The site is designed for people who enjoy learning new skills, exploring personal style, and finding creative ways to enhance their daily lives.

Homepage:

Recipe Page:

Articles:



Each article is written by exactly one author. I am storing the author's information directly in the articles table. Each entity has a unique ID for each article, and the author's information is stored in separate tables (first name, last name, email address). The article_id column is auto-incremented and only holds int data types, whereas the first_name, last_name, and email columns hold varchar data types. If it was not for one author per article, I would create multiple tables. This design is normalized because there is no repetition of authors across multiple articles and there is no need to update author information in multiple places.

	article_id	author_first_name	author_last_name	author_email
▶	1	April	Showers	bringmayflowers@gmail.com
	2	Ray	Geraldi	rgeraldi@gmail.com
	3	Eve	Tokens	evetokens@gmail.com
	4	Kaylee	Pope	kpope@gmail.com
	5	David	Mela	Dmelameals@gmail.com
	6	Steve	Peters	speters@gmail.com
	7	AnnMarie	Mattila	amm@gmail.com
	8	Hannah	Klinger	hklinger@gmail.com
	9	Simone	Howem	Showem@gmail.com
	10	Lauren	Cochrane	cochrane@gmail.com
	11	Emily	May	eemay@gmail.com
	12	Isabel	Steele	steele.isa@gmail.com
	13	Erica	Ball	ericaballstyle@gmail.com
	14	Jennifer	Ebert	jennyebert@gmail.com
	15	Michelle	Brunner	Michbrun@gmail.com
	16	Christina	Dennis	cdennis@gmail.com
	17	Lauren	Hamilton	laurenh@gmail.com
	18	Heather	Luckhurst	luckhurst@gmail.com
	19	Jessi	Lockheart	thecoffeemom@gmail.com
	20	chefworks		chefworks@gmail.com
	21	Karen	Howell	karenhowell@yahoo.com
	22	Katrin	Bennhold	katrin@yahoo.com
	23	Lisa Selin	Davis	lisaselindavis@yahoo.com
	24	Elizabeth	Nohammer	nohammer@ymail.com
	25	Nick	Pope	npope@hotmail.com
	26	Daniel	Johnson	djohnson@hotmail.com
	27	Daphne	Ewing-Chow	daphchow@hotmail.com
	28	Allison	Aubrey	alliaubrey@gmail.com
	29	Leonard	Sax	lsax@gmail.com
	30	Anna	Marie	annamarie@gmail.com
	31	Andrea	Hsu	hsuandrea@gmail.com
	32	Eva	Cifre	Evacifre@ymail.com
	NULL	NULL	NULL	NULL

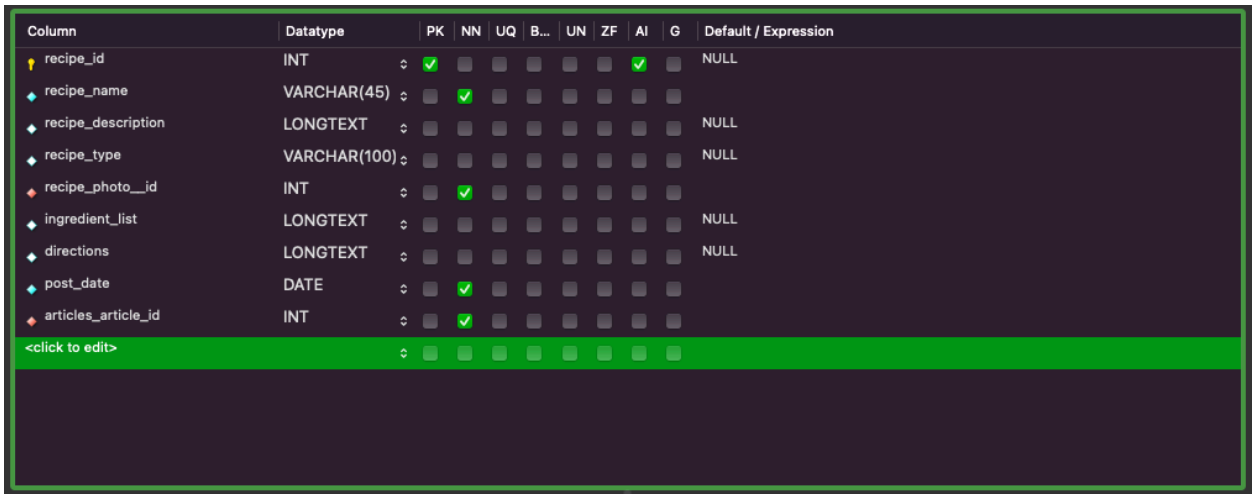
- article_id (INT, PRIMARY KEY): Uniquely identifies each article.
- author_first_name (VARCHAR, 100): Stores the first name of the author.
- author_last_name (VARCHAR, 100): Stores the last name of the author.
- author_email (VARCHAR, 255): Stores the email address of the author.
- article_name (VARCHAR, 255): The title of the article.
- article_description (TEXT): A short description or summary of the article.
- publish_date (DATE): The date the article was published.

Design Considerations:

author_first_name, author_last_name, and author_email are stored as separate columns, allowing for efficient querying of author information. The article details like article_name and article_description are easily queried as text fields.

Purpose: Stores information about various articles, including the author's name and email, article name, and description.

Recipes:



Column	Datatype	PK	NN	UQ	B...	UN	ZF	AI	G	Default / Expression
recipe_id	INT	✓						✓		NULL
recipe_name	VARCHAR(45)		✓							
recipe_description	LONGTEXT									NULL
recipe_type	VARCHAR(100)									NULL
recipe_photo__id	INT		✓							
ingredient_list	LONGTEXT									NULL
directions	LONGTEXT									NULL
post_date	DATE		✓							
articles_article_id	INT		✓							
<click to edit>										

The recipes table stores individual recipes along with key information about each dish. Included in the table is the name of the dish, description, type of meal, a reference photo, list of ingredients, directions, and the date the recipe was posted. The recipe_id is an int data type, and it is the unique identifier. The recipe_photo_id references a photo of the dish through a separate photos table that includes the photo id and url. I use a foreign key to link recipe_photo to recipes.

recipe_id	recipe_name	recipe_description	recipe_type	recipe_photo_id	ingredient_list	directions	post_date	article_id
1	mac and cheese	This homemade m…	noodle	1	8 ounces unco…	Gather all ingredients. Preheat the oven to 350…	2024-11-21	1
2	PBJ	A college students…	sandwich	2	2 slices of bre…	Gather all ingredients. Spread peanut butter on…	2023-01-13	2
3	1 Minute Breakfast Burrito	A fast DIY breakfas…	tortilla	3	2 large eggs,…	Gather all ingredients. Spray a bowl with nonstic…	2022-03-26	3
4	Cheesy Lasagna Sheet Pasta	This 3 ingredient c…	noodles	4	8 ounces lasa…	Bring a large pot of lightly salted water to a boil.…	2024-05-28	4
5	Taco Casserole	This taco casserole…	casserole	5	1 pound lean…	Gather all ingredients. Preheat the oven to 350…	2024-06-14	5
6	Salmon with Brown Sugar Glaze	This brown sugar s…	fish	6	4 (6 ounce) bo…	Preheat the oven broiler and set an oven rack a…	2024-11-16	6
7	Pesto Cheesy Chicken Rolls	This is a very simpl…	chicken	7	4 skinless, bo…	Preheat the oven to 350 degrees F (175 degree…	2024-11-25	7
8	Spicy Lime Grilled Shrimp	Grilled shrimp with…	shrimp	8	3 tablespoons…	Mix together Cajun seasoning, lime juice, and oi…	2023-03-29	8
9	Fried Cabbage and Egg Noodles	This is a German r…	noodle	9	1 (16 ounce) p…	Bring a large pot of lightly salted water to a boil.…	2023-03-29	9
10	Turkey Burgers	I love to cook and l…	sandwich	10	1 pound groun…	Preheat a grill for high heat. In a large bowl, co…	2024-08-08	10
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

- recipe_id (INT, PRIMARY KEY): Uniquely identifies each recipe.
- recipe_name (VARCHAR, 255): Stores the name of the recipe.
- recipe_description (TEXT): Stores a detailed description of the recipe.

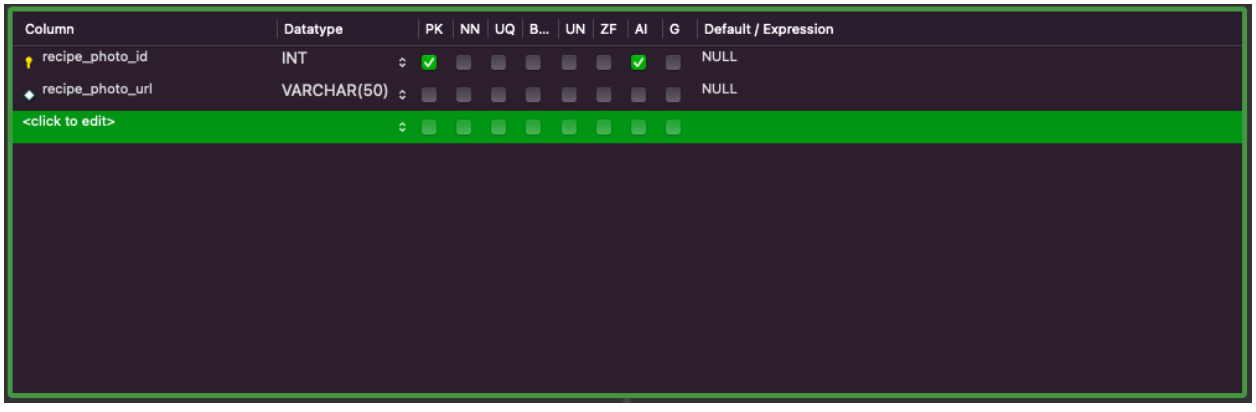
- recipe_type (VARCHAR, 100): Stores the type/category of the recipe (e.g., "Dessert", "Main Course").
- recipe_photo_id (INT, FOREIGN KEY): Links to the recipe_photo table for the recipe's image.
- ingredient_list (TEXT): Stores a list of ingredients, typically as a comma-separated string (may need further normalization).
- directions (TEXT): Provides the instructions for preparing the recipe.
- post_date (DATE): The date the recipe was posted.

Design Considerations:

This table identifies each recipe by a unique recipe_id, and recipe_photo_id is used to link to a photo stored in the recipe_photo table. The ingredient_list and directions columns allow for text storage without limitations on length, as they may contain long data.

Purpose: Stores information about different recipes, including details like the recipe name, description, and photo.

Recipe_photo:



The recipe_photo table holds images of the recipes featured in the database. It includes a recipe_photo_id as the primary key and a recipe_photo_url where the image is stored. Keeping recipe photos in a separate table avoids clutter in the recipes table and provides flexibility for managing images independently, such as updating or replacing recipe images without affecting recipe details.

recipe_photo_id	recipe_photo_url
1	macandcheese.png
2	pbj.png
3	breakfastburrito.png
4	cheesylasagna.png
5	tacocasserole.png
6	salmonwithbrowns…
7	pestochickenrolls.…
8	spicylimeshrimp.png
9	friedcabbageandn…
15	turkeyburgers.png
NULL	NULL

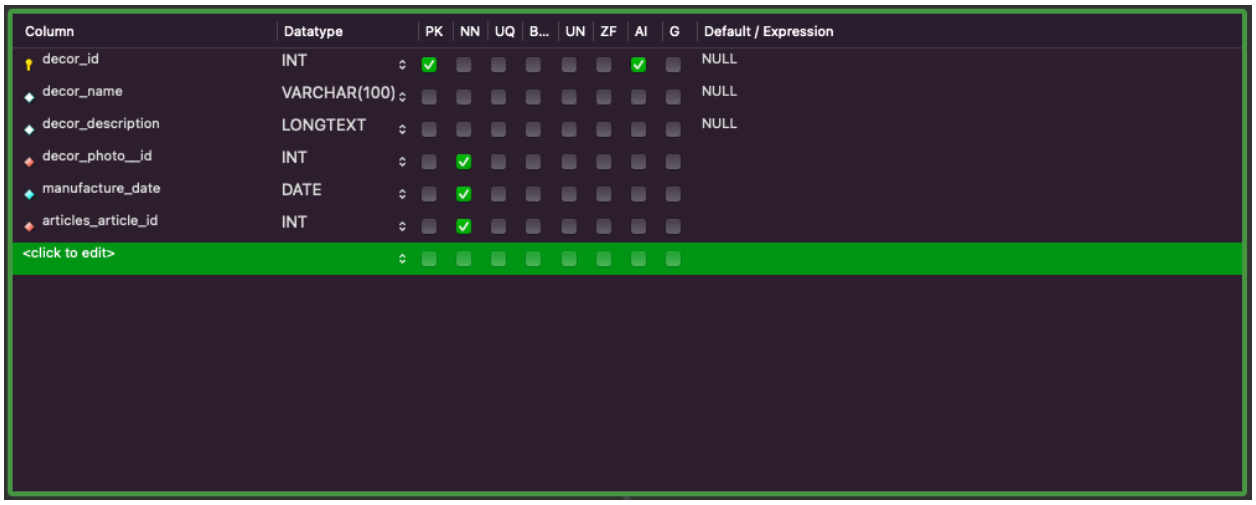
- recipe_photo_id (INT, PRIMARY KEY): Uniquely identifies each recipe photo.
- recipe_photo_url (VARCHAR, 255): Stores the URL of the recipe's photo.

Design Considerations:

The recipe_photo table is designed to store all photos associated with recipes in a separate table. This allows for better organization and scalability, especially when multiple recipes share the same photo or when photos need to be updated independently.

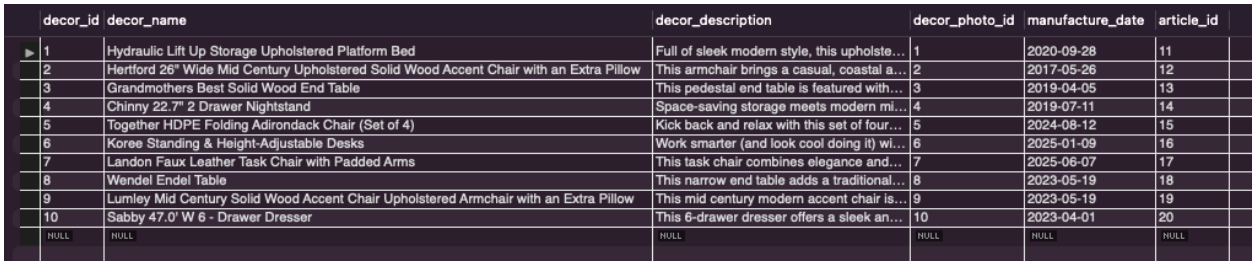
Purpose: Stores photo URLs for recipe images.

Decor:



Column	Datatype	PK	NN	UQ	B...	UN	ZF	AI	G	Default / Expression
decor_id	INT	✓						✓		NULL
decor_name	VARCHAR(100)									NULL
decor_description	LONGTEXT									NULL
decor_photo__id	INT		✓							
manufacture_date	DATE		✓							
articles_article_id	INT		✓							
<click to edit>										

The decor table contains the id of the item, the name, description, a photo reference, and manufacture date. There is a separate table that is linked through a foreign key called decor_photo_id, and it references the decor_photo table where the photo id is stored along with a url of the image.



decor_id	decor_name	decor_description	decor_photo_id	manufacture_date	article_id
1	Hydraulic Lift Up Storage Upholstered Platform Bed	Full of sleek modern style, this upholste...	1	2020-09-28	11
2	Hertford 26" Wide Mid Century Upholstered Solid Wood Accent Chair with an Extra Pillow	This armchair brings a casual, coastal a...	2	2017-05-26	12
3	Grandmothers Best Solid Wood End Table	This pedestal end table is featured with...	3	2019-04-05	13
4	Chinny 22.7" 2 Drawer Nightstand	Space-saving storage meets modern mi...	4	2019-07-11	14
5	Together HDPE Folding Adirondack Chair (Set of 4)	Kick back and relax with this set of four...	5	2024-08-12	15
6	Koree Standing & Height-Adjustable Desks	Work smarter (and look cool doing it) wi...	6	2025-01-09	16
7	Landon Faux Leather Task Chair with Padded Arms	This task chair combines elegance and...	7	2025-06-07	17
8	Wendel Endel Table	This narrow end table adds a traditional...	8	2023-05-19	18
9	Lumley Mid Century Solid Wood Accent Chair Upholstered Armchair with an Extra Pillow	This mid century modern accent chair is...	9	2023-05-19	19
10	Sabby 47.0' W 6 - Drawer Dresser	This 6-drawer dresser offers a sleek an...	10	2023-04-01	20
NULL	NULL	NULL	NULL	NULL	NULL

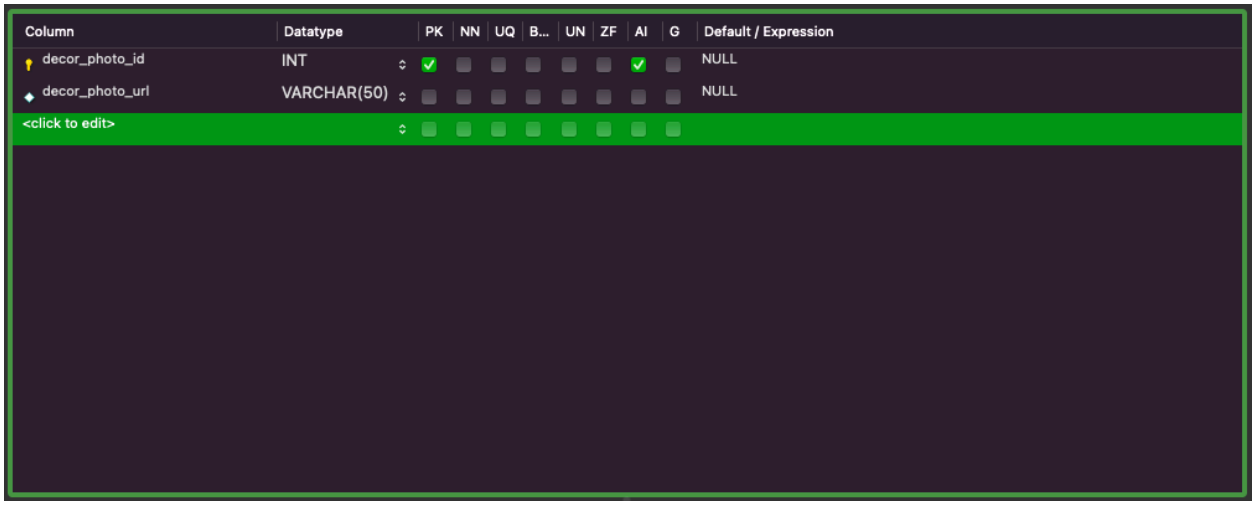
- decor_id (INT, PRIMARY KEY): Uniquely identifies each decor item.
- name (VARCHAR, 255): Stores the name of the decor item.
- description (TEXT): A detailed description of the decor.
- photo_id (INT, FOREIGN KEY): Links to the decor_photo table for the decor's image.
- manufacture_date (DATE): The date when the decor item was manufactured.

Design Considerations:

By separating the photo details into a separate table (decor_photo), the database maintains flexibility in storing and associating images with decor items. This also prevents duplication if multiple decor items share the same image.

Purpose: Stores information about decor items, such as their name, description, and associated photo.

decor_photo:



The decor_photo table contains photos related to decor items. Each photo entry is uniquely identified with a decor_photo_id and includes a decor_photo_url to reference the image. This design helps separate decor images from decor item descriptions, making updates to photos simple and keeping the decor data streamlined and easy to maintain.

decor_photo_id	decor_photo_url
1	platformbed.png
2	solidwoodaccentchair.png
3	endtable.png
4	chinnydrawernightstand.png
5	togetherchair.png
6	koreestandingdesk.png
7	landonarmchair.png
8	wendelendeltable.png
9	lumleyarmchair.png
10	sabbydrawerdresser.png
NULL	NULL

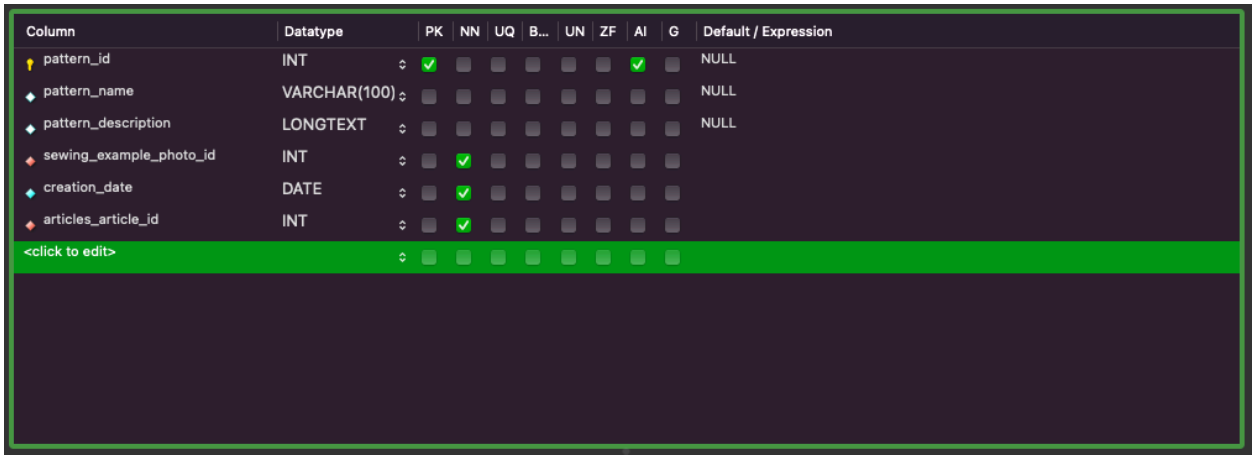
- decor_photo_id (INT, PRIMARY KEY): Uniquely identifies each decor photo.
- decor_photo_url (VARCHAR, 255): Stores the URL of the photo.

Design Considerations:

The decor_photo table allows for separate storage of photos, which keeps the decor item data independent from the actual image, supporting easy image updates and management.

Purpose: Stores photo URLs for decor item images.

Sewing:



The sewing table contains the id of a sewing pattern, the pattern name, description, a photo reference of the design, and pattern creation date. There is a separate table that is linked

through a foreign key called sewing_example_photo_id, and it references sewing_example_photo, where the photo id is stored along with a url of the image outcome. If I will later track materials or the amount of steps used in a pattern, I may add more tables and relationships.

pattern_id	pattern_name	pattern_description	sewing_example_photo_id	creation_date	article_id
1	The Asheville Dress Free Sewing Pattern	Perfect for capturing that relaxed yet refined sea…	1	2025-04-24	31
2	The Ibiza Trousers Free Sewing Pattern	Designed with structure and versatility, these tro…	2	2025-04-03	32
3	The Marbella Foldover Top Free Sewing Pattern	This off-the-shoulder top features long sleeves…	3	2025-04-02	33
4	The Sorrento Set Free Sewing Pattern	The adaptable two-piece ensemble features a c…	4	2023-03-29	34
5	The Ischia Cardigan Free Sewing Pattern	Perfect for both formal and informal events, the l…	5	2024-10-08	35
6	The Taormina Joggers Free Sewing Pattern	As a fabric company, we know the importance o…	6	2022-06-17	36
7	The Ophelia Dress Free Sewing Pattern	This free sewing pattern is perfect for those who…	7	2025-02-03	37
8	The Ravenna Suit Free Sewing Pattern	This free sewing pattern features a blazer with a…	8	2025-01-28	38
9	The Palermo Pajamas Free Sewing Pattern	This free sewing pattern includes a loose-fitting…	9	2023-09-07	39
10	The Ela Peplum Top and Ava Mini Skirt	This sleek two-piece set is made with suiting fab…	10	2024-12-24	40
NULL	NULL	NULL	NULL	NULL	NULL

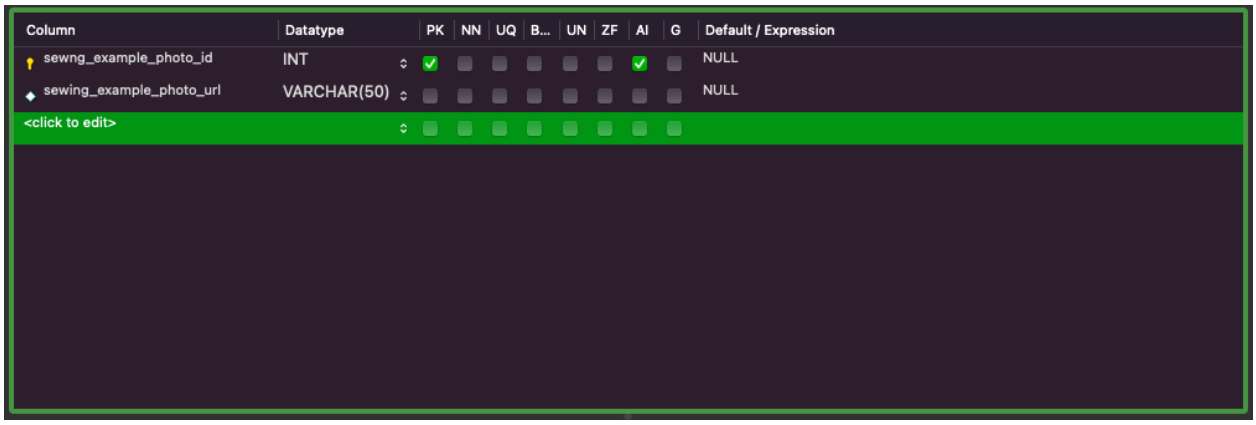
- pattern_id (INT, PRIMARY KEY): Uniquely identifies each pattern.
- name (VARCHAR, 255): The name of the sewing pattern.
- description (TEXT): A detailed description of the sewing pattern.
- sewing_example_photo_id (INT, FOREIGN KEY): Links to a photo in the sewing_example_photo table.
- creation_date (DATE): The date the pattern was created.

Design Considerations:

The sewing_example_photo_id links to a photo table, which stores images separately, keeping the sewing table clean and ensuring that each pattern's image can be reused if necessary.

Purpose: Stores information about sewing patterns, their descriptions, and related images.

sewing_example_photo:



The sewing_example_photo table stores photos that showcase examples of completed sewing patterns. It uses a sewing_example_photo_id as the primary key and a sewing_example_photo_url to hold the URL of the image. By linking these photos to sewing patterns separately, the database maintains clean relationships and makes it easier to manage visual examples without duplicating pattern information.

sewing_example_photo_id	sewing_example_photo_url
1	asheville.png
2	ibiza.png
3	marbella.png
4	sorrento.png
5	ischia.png
6	taormina.png
7	ophelia.png
8	ravenna.png
9	palermo.png
10	elaandava.png
NULL	NULL

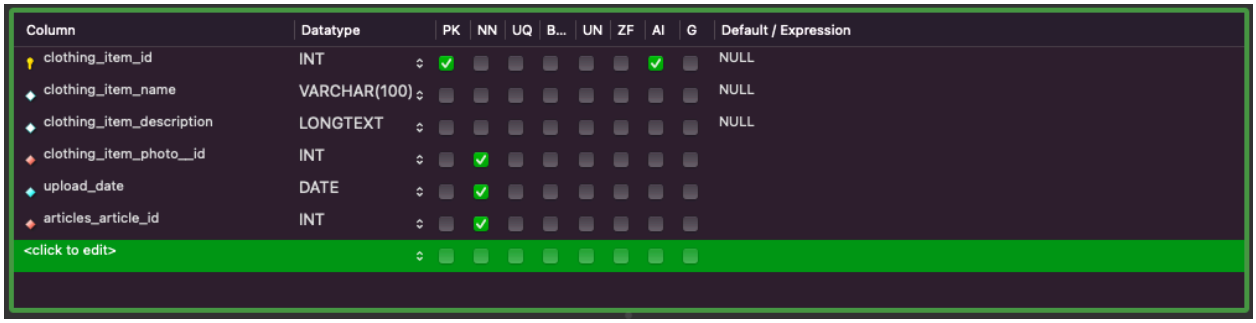
- sewing_example_photo_id (INT, PRIMARY KEY): Uniquely identifies each sewing example photo.
- sewing_example_photo_url (VARCHAR, 255): Stores the URL of the photo.

Design Considerations:

Similar to the clothing_item_photo table, this table separates photo data from the main sewing pattern data, making it easier to manage images and ensure efficient updates.

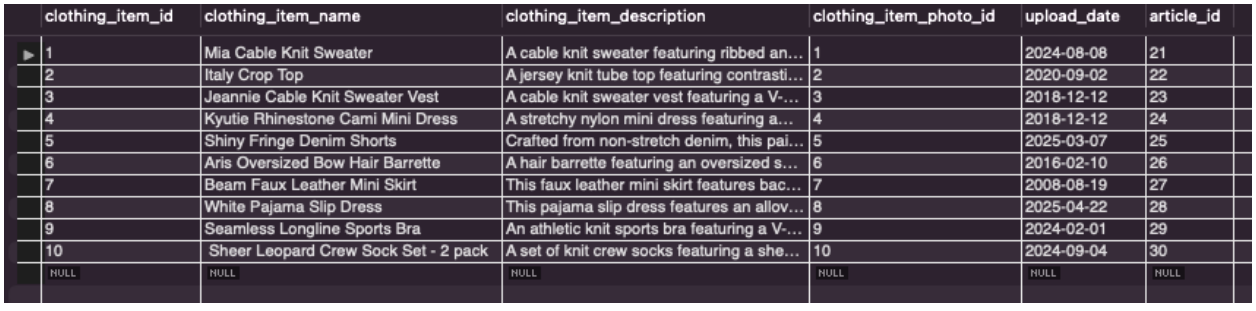
Purpose: Stores photo URLs for sewing pattern examples.

Fashion:



Column	Datatype	PK	NN	UQ	B...	UN	ZF	AI	G	Default / Expression
clothing_item_id	INT	✓						✓		NULL
clothing_item_name	VARCHAR(100)									NULL
clothing_item_description	LONGTEXT									NULL
clothing_item_photo__id	INT		✓							
upload_date	DATE		✓							
articles_article_id	INT		✓							
<click to edit>										

The fashion table contains the id of a clothing item, the clothing name, a description, a photo reference of the item, and the clothing item upload date. There is a separate table that is linked through a foreign key called clothing_item_photo_id, and it references clothing_item_photo, where the photo id is stored along with a url of the item.



clothing_item_id	clothing_item_name	clothing_item_description	clothing_item_photo_id	upload_date	article_id
1	Mia Cable Knit Sweater	A cable knit sweater featuring ribbed an…	1	2024-08-08	21
2	Italy Crop Top	A jersey knit tube top featuring contrasti…	2	2020-09-02	22
3	Jeannie Cable Knit Sweater Vest	A cable knit sweater vest featuring a V-…	3	2018-12-12	23
4	Kyutie Rhinestone Cami Mini Dress	A stretchy nylon mini dress featuring a…	4	2018-12-12	24
5	Shiny Fringe Denim Shorts	Crafted from non-stretch denim, this pai…	5	2025-03-07	25
6	Aris Oversized Bow Hair Barrette	A hair barrette featuring an oversized s…	6	2016-02-10	26
7	Beam Faux Leather Mini Skirt	This faux leather mini skirt features bac…	7	2008-08-19	27
8	White Pajama Slip Dress	This pajama slip dress features an allov…	8	2025-04-22	28
9	Seamless Longline Sports Bra	An athletic knit sports bra featuring a V-…	9	2024-02-01	29
10	Sheer Leopard Crew Sock Set - 2 pack	A set of knit crew socks featuring a she…	10	2024-09-04	30
NULL	NULL	NULL	NULL	NULL	NULL

- clothing_item_id (INT, PRIMARY KEY): Uniquely identifies each clothing item.
- name (VARCHAR, 255): The name of the clothing item.

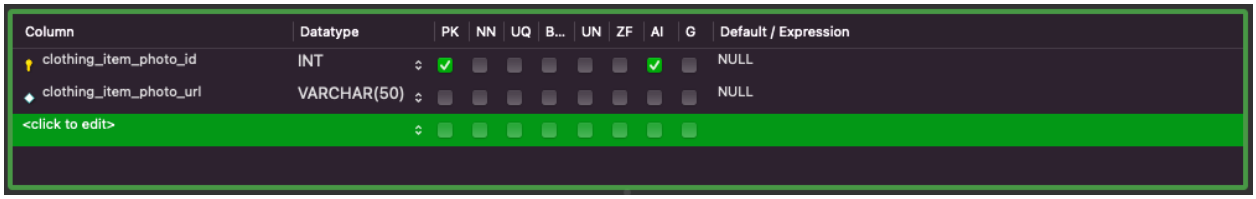
- description (TEXT): A detailed description of the clothing item.
- photo_id (INT, FOREIGN KEY): Links to the clothing_item_photo table for the clothing item's image.
- upload_date (DATE): The date the clothing item was uploaded.

Design Considerations:

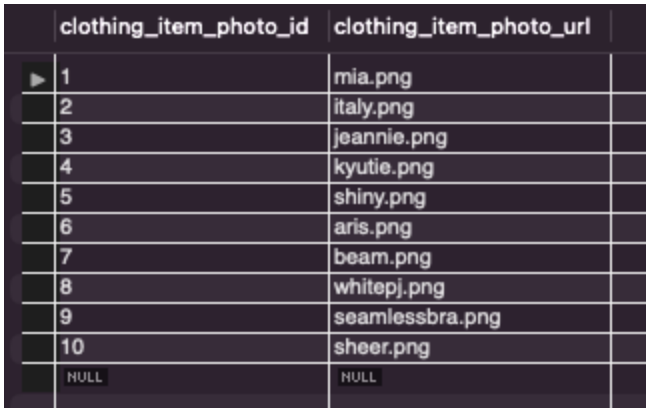
The photo_id links the clothing item to a separate photo table, allowing for easy management of multiple clothing items with different images. The upload_date provides useful metadata about when the item was added to the system.

Purpose: Stores details about fashion items, such as clothing descriptions, photos, and upload dates.

clothing_item_photo:



The clothing_item_photo table is designed to store images associated with clothing items listed in the fashion section of the database. It includes a clothing_item_photo_id as the primary key to uniquely identify each photo and a clothing_item_photo_url to store the location of the image file. Separating clothing photos into their own table keeps the database organized and allows updates or changes to images without affecting the main clothing item records.



clothing_item_photo_id	clothing_item_photo_url
1	mia.png
2	italy.png
3	jeannie.png
4	kyutie.png
5	shiny.png
6	aris.png
7	beam.png
8	whitepj.png
9	seamlessbra.png
10	sheer.png
NULL	NULL

- clothing_item_photo_id (INT, PRIMARY KEY): Uniquely identifies each clothing item photo.
- clothing_item_photo_url (VARCHAR, 255): Stores the URL of the photo.

Design Considerations:

The clothing_item_photo table ensures that photos are stored separately from the clothing items, allowing for more flexibility in managing images and enabling multiple clothing items to share the same photo.

Purpose: Stores photo URLs for clothing items.

Normalization:

In this database, which stores information about recipes, articles, decor, fashion, and sewing, normalization organizes the data logically, making it easier to query, update, and maintain. By applying the principles of 1st, 2nd, and 3rd Normal Forms (1NF, 2NF, and 3NF), I have created a system that reduces redundancy and prevents issues like data duplication and update inconsistencies, keeping the database reliable and scalable as it expands.

1st Normal Form (1NF) guarantees that each column holds atomic values and that each row is uniquely identifiable. For example, in the fashion table, the clothing_item_description column contains detailed descriptions of each clothing item, with no multiple values stored in a single field. This structure allows for each piece of information, such as the description or photo URL, to be kept separate and identifiable, which helps maintain clarity and improves querying efficiency.

2nd Normal Form (2NF) requires that non-key attributes depend on the entire primary key. For instance, in the recipes table, attributes such as recipe_name and recipe_description are fully dependent on the recipe_id. With a single primary key per table, no partial dependencies exist, helping to maintain consistency and prevent unnecessary duplication.

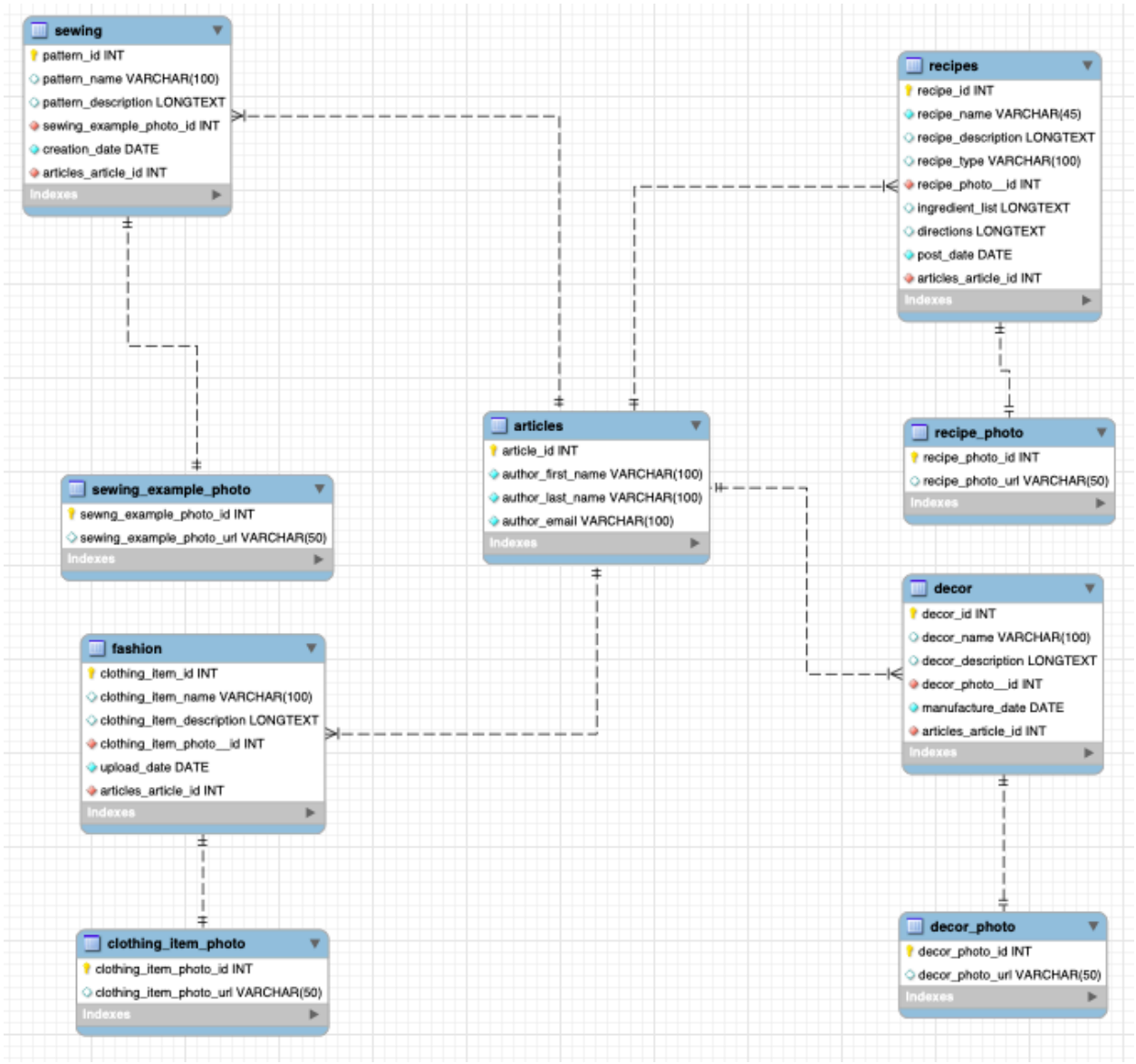
3rd Normal Form (3NF) is followed by making sure that all non-key attributes depend solely on the primary key, with no indirect dependencies. In the recipes table, for example, recipe_name and recipe_description are linked only to recipe_id, and photo-related data, such as recipe_photo_id and recipe_photo_url, is stored in a separate recipe_photo table.

This eliminates transitive dependencies, confirming that the database is efficient and free from redundant relationships.

Normalization is essential for maintaining data integrity, enhancing performance, and avoiding anomalies. By organizing data into separate, related tables, the database reduces redundancy and verifies efficient handling of updates and deletions. This structure enhances scalability and establishes consistent and accurate data management over time.

The database is designed effectively using normalization, with tables for recipes, articles, decor, fashion, and sewing. Following 1NF, 2NF, and 3NF affirms the system remains organized, scalable, and efficient while minimizing redundancy and supporting reliable querying and maintenance as the database grows.

EER Diagram:


sewing
pattern_id INT
pattern_name VARCHAR(100)
pattern_description LONGTEXT
sewing_example_photo_id INT
creation_date DATE
articles_article_id INT
Indexes
recipes
recipe_id INT
recipe_name VARCHAR(45)
recipe_description LONGTEXT
recipe_type VARCHAR(100)
recipe_photo__id INT
ingredient_list LONGTEXT
directions LONGTEXT
post_date DATE
articles_article_id INT
Indexes
articles
article_id INT
author_first_name VARCHAR(100)
author_last_name VARCHAR(100)
author_email VARCHAR(100)
Indexes
recipe_photo
recipe_photo_id INT
recipe_photo_url VARCHAR(50)
Indexes
sewing_example_photo
sewng_example_photo_id INT
sewing_example_photo_url VARCHAR(50)
Indexes
decor
decor_id INT
decor_name VARCHAR(100)
decor_description LONGTEXT
decor_photo__id INT
manufacture_date DATE
articles_article_id INT
Indexes
fashion
clothing_item_id INT
clothing_item_name VARCHAR(100)
clothing_item_description LONGTEXT
clothing_item_photo__id INT
upload_date DATE
articles_article_id INT
Indexes
clothing_item_photo
clothing_item_photo_id INT
clothing_item_photo_url VARCHAR(50)
Indexes
decor_photo
decor_photo_id INT
decor_photo_url VARCHAR(50)
Indexes

A general SELECT query with an ORDER BY clause

```
SELECT * FROM articles
ORDER BY author_last_name asc;
```

This query selects all of the columns from the articles table using SELECT *. It sorts the results based on the author_last_name column in ascending alphabetical order. This is useful for organizing articles by the authors' last names to make the articles easier to find.

A SELECT query that includes a WHERE clause

```
SELECT author_first_name
FROM articles
WHERE author_email = "kpope@gmail.com";
```

This query retrieves the first_name of a user from the users table. It filters the results using a WHERE clause to match only the row where the email is "kpope@gmail.com". If a match is found, it returns the user's first name; if not, it returns nothing.

A Delete Query

```
DELETE
FROM recipes
WHERE recipe_id = 11;
```

This query deletes all rows from the recipes table where the recipe_id is 11. It uses a WHERE clause to make sure only recipes of that specific type are removed.

An Update Query

UPDATE recipes

SET recipe_type = 'sandwich'

WHERE recipe_id = 2

This query updates the recipe_type for a specific row in the recipes table. It sets the recipe_type to 'sandwich' where the recipe_id is 2, which is the PBJ. The WHERE clause ensures that only the recipe with recipe_id equal to 2 is updated, and no other rows are updated.

An Insert Query

INSERT INTO recipes (recipe_id, recipe_name, recipe_type, post_date)

VALUES (11, 'Tomato Soup', 'Soup', '2025-04-26');

This query inserts a new row into the recipes table with specific values for recipe_id, recipe_name, recipe_type, and post_date. The INSERT INTO statement specifies the table and columns where the data should go. The VALUES part lists the new data that will be added into the corresponding columns.

insert into articles (article_id, author_first_name, author_last_name, author_email)

values (33,

'Evan',

'Jennings',

'Evanjennings@ymail.com')

This query inserts a new row into the articles table with specific values for columns like article_id, author_first_name, author_last_name, and author_email. It adds an article written by Evan Jennings with all related information. Each value matches its corresponding column, and the email is recorded as Evanjennings@ymail.com.

A query for inner join

```
SELECT
    recipes.recipe_id,
    recipes.recipe_type,
    recipes.recipe_description,
    recipe_photo.recipe_photo_url
FROM recipes
INNER JOIN recipe_photo ON recipes.recipe_photo_id = recipe_photo.recipe_photo_id;
```

This query selects multiple columns (recipe_id, recipe_type, recipe_description) from the recipes table, along with recipe_photo_url from the recipe_photo table. It uses an INNER JOIN to combine the two tables, linking them on the matching recipe_photo_id field. Only rows with matching recipe_photo_id in both tables will be included in the result set.

A query for outer join (left or right)

```
SELECT
        p.recipe_photo_id,
        p.recipe_photo_url,
        r.recipe_id
FROM recipes r
RIGHT JOIN recipe_photo p ON r.recipe_photo_id = p.recipe_photo_id;
```

This query selects columns from two tables (recipe_photo_id, recipe_photo_url) in the recipe_photo table and recipe_id from the recipes table. It uses a RIGHT JOIN to combine the two tables, allowing all rows from the recipe_photo table to be included, even if there is no matching recipe_photo_id in the recipes table. If a photo doesn't have an associated recipe, the recipe_id will be NULL in the result set.

A query with an aggregate function(s)

```
SELECT COUNT(*) AS total_recipes
FROM recipes;
```

This query counts the total number of rows in the recipes table. It uses the COUNT(*) function, which counts every row, including ones with NULL values in columns. The result is returned with the label total_recipes as the column name.

```
SELECT AVG(LENGTH(ingredient_list) - LENGTH(REPLACE(ingredient_list, ',', '')) + 1) AS
avg_ingredients
```

FROM recipes

WHERE ingredient_list IS NOT NULL;

This query calculates the average number of ingredients listed in the ingredient_list column of the recipes table. It counts the number of commas in each ingredient_list (using LENGTH and REPLACE) and adds 1, assuming that ingredients are separated by commas. The WHERE clause ensures that only recipes with a non-NULL ingredient_list are included in the average calculation.

SELECT MAX(post_date) AS latest_post

FROM recipes;

Latest recipe posted

This query finds the most recent post_date from the recipes table by using the MAX() function. It returns the latest date when any recipe was posted, labeling the result as latest_post. This helps quickly identify the newest recipe added to the database.

A query that includes GROUP BY and HAVING clauses

SELECT recipe_type, COUNT(*) AS count

FROM recipes

GROUP BY recipe_type;

This query counts how many recipes exist for each recipe_type. It uses GROUP BY to group the results by recipe_type, then applies COUNT(*) to count the number of recipes in each group. The result shows each recipe type along with the number of recipes of that type.

A query with a subquery

```
SELECT *
FROM (
        SELECT recipe_type, COUNT(*) AS type_count
        FROM recipes
        GROUP BY recipe_type
) AS recipe_counts
WHERE type_count > 1;
```

This query creates a temporary result (a subquery) that counts how many recipes exist for each recipe_type. Then, from that result (called recipe_counts), it selects only the rows where the type_count is greater than 1. It shows only recipe types that appear more than once in the recipes table.

A query that includes a string function

```
insert into clothing_item_photo (clothing_item_photo_id, clothing_item_photo_url)
values (11, bluetop.png');
```

This query inserts a new row into the clothing_item_photo table. It sets the clothing_item_photo_id to 11 and the clothing_item_photo_url to bluetop.png'. This adds a new photo record to the table, linking the ID 11 with the image file ‘bluetop.png'.

A query that includes a numeric function

```
SELECT recipe_id, LENGTH(ingredient_list) AS ingredient_list_length
FROM recipes;
```

This query selects the recipe_id and uses the LENGTH() function to calculate the number of characters in the ingredient_list column. LENGTH() is a numeric function that returns a numeric value representing the text length. The result shows how long each recipe's ingredient list is.

A query that includes a date function

```
SELECT recipe_id, YEAR(post_date) AS post_year
FROM recipes;
```

This query selects the recipe_id and uses the YEAR() date function to extract the year from the post_date column. YEAR() returns just the four-digit year for each recipe. This is useful because it will group or filter recipes based on the year they were posted.