--> Get Dishname, Rating and RecipeURL

CREATE VIEW ShowAvailableRecipes AS

SELECT r.DishName, r.Rating, r.TotalCookingTime AS TotalCookingTime, i.ImageURL

FROM Recipes r

LEFT JOIN Images i ON r.RecipeID = i.RecipeID;

GO

A screenshot of a computer

Description automatically generated

This view combines data from the ‘Recipes’ and ‘Images’ tables. It prints each recipe’s name, rating, total cooking time and image URL. A LEFT JOIN is used to include all the recipes (even those without images).

--> View to show all recipe instructions

--NOTE: this view prints out multiple lines for the results, I’m still yet to figure out how to make it work:

CREATE VIEW RecipeInstructions AS

SELECT R.DishName AS RecipeName,

R.Description,

R.Rating,

I.IngredientName,

I.Quantity AS IngredientQuantity,

I.MeasurementUnit AS IngredientMeasurementUnit,

IM.ImageURL,

I2.StepNumber,

I2.InstructionDescription

FROM Recipes R

INNER JOIN Ingredients I ON R.RecipeID = I.RecipeID

LEFT JOIN (SELECT RecipeID, StepNumber, InstructionDescription

FROM Instructions) I2 ON R.RecipeID = I2.RecipeID

LEFT JOIN (SELECT RecipeID, ImageURL

FROM Images) IM ON R.RecipeID = IM.RecipeID;

GO

This view combines data from the ‘Recipes’, ‘Ingredients’, ‘Instructions’, and ‘Images’ tables to provide a general overview of a recipe. It prints the recipe name, description, rating, ingredient details (name, quantity, unit), image URL, and cooking instructions (step number and description). The view uses INNER JOIN to link recipes with their ingredients and LEFT JOIN to include optional images and instructions.

--View Recipe Instructions

SELECT \*

FROM RecipeInstructions

WHERE RecipeName = 'Fried Rice';

GO

The query above utilizes the ‘RecipeInstructions’ view to print the general details of the recipe ‘Fried Rice’.

--> Insert new User

CREATE PROCEDURE InsertUser(

@Username NVARCHAR(50),

@Password NVARCHAR(50),

@DateOfBirth DATE,

@Email NVARCHAR(100),

@UserRole NVARCHAR(50)

)

AS

BEGIN

INSERT INTO Users (Username, Password, DateOfBirth, Email, UserRole)

VALUES (@Username, @Password, @DateOfBirth, @Email, @UserRole);

END;

GO

EXEC InsertUser 'user4', 'passsostrong', '1994-01-04', 'user4@example.com', 'contributor';

GO

A screenshot of a computer

Description automatically generated

**NOTE:** The insert procedures take the same format as the one used for the ‘InsertUsers’ above so we’ll will just show the queries to save space in the document, and avoid it being too clunky.

The ‘InsertUser’ procedure streamlines the process of inserting new users into the ‘Users’ table; it takes all the necessary data and passes it into the table.

--> Insert new category

CREATE PROCEDURE InsertCategory(

@CategoryName VARCHAR(255)

) AS

BEGIN

INSERT INTO Categories (CategoryName) VALUES (@CategoryName)

END;

GO

EXEC InsertCategory 'Seafood';

GO

Similar to the ‘InsertUser’ procedure, the ‘InsertCategory’ simplifies the process of inserting new data into the Categories table.

--> Insert new Recipes

CREATE PROCEDURE InsertRecipe(

@CategoryID INT,

@UserID INT,

@DishName VARCHAR(100),

@TotalCookingTime VARCHAR(50),

@Description VARCHAR(255),

@Rating VARCHAR(255)

) AS

BEGIN

INSERT INTO Recipes (CategoryID, UserID, DishName, TotalCookingTime, Description, Rating)

VALUES (@CategoryID, @UserID, @DishName, @TotalCookingTime, @Description, @Rating)

END;

GO

EXEC InsertRecipe '4', '7', 'Shrimp Fried Rice', '55 mins',

'a symphony of flavors and textures that tantalizes the taste buds with each bite', '4.9';

GO

The ‘InsertRecipe’ procedure optimises the process of inserting new data into the Recipes table.

--> Insert new Ingredients

CREATE PROCEDURE InsertIngredient(

@RecipeID INT,

@IngredientName VARCHAR(255),

@Quantity VARCHAR(255),

@MeasurementUnit VARCHAR(255)

) AS

BEGIN

INSERT INTO Ingredients (RecipeID, IngredientName, Quantity, MeasurementUnit)

VALUES (@RecipeID, @IngredientName, @Quantity, @MeasurementUnit)

END;

GO

EXEC InsertIngredient '13', 'Rice', '2 cups', 'cups';

EXEC InsertIngredient '13', 'Vegetable Oil', '2 tablespoons', 'tablespoons';

EXEC InsertIngredient '13', 'Onions', '1 large', 'large';

EXEC InsertIngredient '13', 'Carrots', '2 medium', 'medium';

EXEC InsertIngredient '13', 'Green Peas', '1 cup', 'cup';

EXEC InsertIngredient '13', 'Bell Peppers', '2', '';

EXEC InsertIngredient '13', 'Salt', 'to taste', '';

EXEC InsertIngredient '13', 'Seasoning Cubes', '2', '';

GO

--> Insert new OptionalIngredients

CREATE PROCEDURE InsertOptionalIngredient(

@RecipeID INT,

@IngredientName VARCHAR(255),

@Quantity VARCHAR(255)

) AS

BEGIN

INSERT INTO OptionalIngredient (RecipeID, IngredientName, Quantity)

VALUES (@RecipeID, @IngredientName, @Quantity)

END;

GO

EXEC InsertOptionalIngredient @RecipeID = 13, @IngredientName = 'Soy Sauce', @Quantity = '2 tablespoons';

EXEC InsertOptionalIngredient @RecipeID = 13, @IngredientName = 'Sesame Oil', @Quantity = '1 teaspoon';

EXEC InsertOptionalIngredient @RecipeID = 13, @IngredientName = 'Scallions', @Quantity = '2 stalks';

GO

--> Insert new CookingHardware

CREATE PROCEDURE InsertCookingHardware(

@RecipeID INT,

@HardwareName VARCHAR(255)

) AS

BEGIN

INSERT INTO CookingHardware (RecipeID, HardwareName)

VALUES (@RecipeID, @HardwareName)

END;

EXEC InsertCookingHardware @RecipeID = 13, @HardwareName= 'Large pan or wok';

EXEC InsertCookingHardware @RecipeID = 13, @HardwareName= 'Stove or cooker';

EXEC InsertCookingHardware @RecipeID = 13, @HardwareName= 'Measuring cups and spoons';

EXEC InsertCookingHardware @RecipeID = 13, @HardwareName= 'Knife';

EXEC InsertCookingHardware @RecipeID = 13, @HardwareName= 'Cutting board';

GO

The above procedure inserts the provided data into the ‘CookingHardware’ table.

--> Insert new Image

CREATE PROCEDURE InsertImage(

@RecipeID INT,

@ImageURL VARCHAR(255)

) AS

BEGIN

INSERT INTO Images (RecipeID, ImageURL)

VALUES (@RecipeID, @ImageURL)

END;

GO

EXEC InsertImage '13', 'https://www.lecremedelacrumb.com/wp-content/uploads/2019/05/one-pan-spanish-shrimp-rice-1.jpg';

GO

This procedure functions like other insert procedures, it inserts image URLs into the Images table and associates them with the corresponding RecipeID.

--> Delete Recipe data from the recipe database

CREATE PROCEDURE DeleteRecipe @RecipeID INT

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRY

BEGIN TRANSACTION;

DELETE FROM RecipeIngredients WHERE RecipeID = @RecipeID;

DELETE FROM CookingHardware WHERE RecipeID = @RecipeID;

DELETE FROM Images WHERE RecipeID = @RecipeID;

DELETE FROM Instructions WHERE RecipeID = @RecipeID;

DELETE FROM OptionalIngredient WHERE RecipeID = @RecipeID;

DELETE FROM Ingredients WHERE RecipeID = @RecipeID;

DELETE FROM Recipes WHERE RecipeID = @RecipeID;

COMMIT TRANSACTION;

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION;

-- errors

PRINT ERROR\_MESSAGE();

END CATCH;

END;

GO

EXEC DeleteRecipe '13';

GO

The DeleteRecipe procedure streamlines the process of deleting an entire recipe's data from the database.

--> Insert new entire Recipe

CREATE PROCEDURE InsertEntireRecipe

@RecipeName VARCHAR(255),

@TotalCookingTime VARCHAR(50),

@IngredientsList XML = NULL,

@InstructionsList XML,

@CookingHardwareList XML = NULL,

@ImageURL VARCHAR(255),

@CategoryName VARCHAR(255),

@Description VARCHAR(255),

@Rating VARCHAR(255),

@Username VARCHAR(50)

AS

BEGIN

SET NOCOUNT ON;

DECLARE @RecipeID INT;

-- Insert into Recipes table

INSERT INTO Recipes (DishName, TotalCookingTime, CategoryID, Description, Rating, UserID)

VALUES (@RecipeName, @TotalCookingTime, (SELECT CategoryID FROM Categories WHERE CategoryName = @CategoryName), @Description, @Rating, (SELECT UserID FROM Users WHERE Username = @Username));

SET @RecipeID = SCOPE\_IDENTITY();

-- Insert into Ingredients table

IF @IngredientsList IS NOT NULL

BEGIN

INSERT INTO Ingredients (RecipeID, IngredientName, Quantity, MeasurementUnit)

SELECT

@RecipeID,

ParamValues.x.value('(IngredientName)[1]', 'VARCHAR(255)'),

ParamValues.x.value('(Quantity)[1]', 'VARCHAR(255)'),

ParamValues.x.value('(MeasurementUnit)[1]', 'VARCHAR(255)')

FROM @IngredientsList.nodes('/Ingredients/Ingredient') AS ParamValues(x);

END;

-- Insert into Instructions table

INSERT INTO Instructions (RecipeID, StepNumber, InstructionDescription)

SELECT

@RecipeID,

ParamValues.x.value('(StepNumber)[1]', 'INT'),

ParamValues.x.value('(InstructionDescription)[1]', 'VARCHAR(255)')

FROM @InstructionsList.nodes('/Instructions/Instruction') AS ParamValues(x);

-- Insert into CookingHardware table

IF @CookingHardwareList IS NOT NULL

BEGIN

INSERT INTO CookingHardware (RecipeID, HardwareName)

SELECT

@RecipeID,

ParamValues.x.value('(HardwareName)[1]', 'VARCHAR(255)')

FROM @CookingHardwareList.nodes('/CookingHardware/Hardware') AS ParamValues(x);

END;

-- Insert into Images table

INSERT INTO Images (RecipeID, ImageURL)

VALUES (@RecipeID, @ImageURL);

END;

The InsertEntireRecipe stored procedure inserts a complete recipe into the database, including details about the recipe, ingredients, optional ingredients, instructions, cooking hardware, and images. It checks if the specified category exists, inserts it if not, and then proceeds to insert the recipe details, followed by the related ingredients, optional ingredients, instructions, cooking hardware, and images. The example executions demonstrate inserting different recipes with their associated details into the database.

-- Testing alternative recipe recommendations based on ingredients

-- Declare necessary variables

DECLARE @CategoryID INT;

DECLARE @UserID INT;

-- Check if the category exists and fetch its ID or insert it if it does not exist

DECLARE @CategoryName VARCHAR(255) = 'Asian Cuisine'; -- Example category name

-- Check for the existence of the category and insert if it does not exist

IF NOT EXISTS (SELECT 1 FROM Categories WHERE CategoryName = @CategoryName)

BEGIN

INSERT INTO Categories (CategoryName)

VALUES (@CategoryName);

SET @CategoryID = SCOPE\_IDENTITY(); -- Get the newly created Category ID

END

ELSE

BEGIN

SELECT @CategoryID = CategoryID FROM Categories WHERE CategoryName = @CategoryName; -- Get existing Category ID

END

-- Check if user exists and fetch UserID, or you must ensure a user with a specific username exists

SELECT @UserID = UserID FROM Users WHERE Username = 'chefuser'; -- Make sure this username exists in your database

IF @UserID IS NULL

BEGIN

-- Insert user logic here, or handle the error if the user must exist

RAISERROR('Specified user does not exist.', 16, 1);

RETURN;

END

-- Now, use the CategoryID and UserID to insert a new recipe

BEGIN TRY

BEGIN TRANSACTION;

-- Inserting recipe assuming CategoryID and UserID are now valid

INSERT INTO Recipes (DishName, TotalCookingTime, Description, Rating, CategoryID, UserID)

VALUES ('Fried Rice', 30, 'Traditional fried rice with vegetables and chicken.', 'good', @CategoryID, @UserID);

COMMIT TRANSACTION;

END TRY

BEGIN CATCH

IF @@TRANCOUNT > 0

ROLLBACK TRANSACTION;

-- Re-throw the error for further handling

DECLARE @ErrorMessage NVARCHAR(4000) = ERROR\_MESSAGE();

DECLARE @ErrorSeverity INT = ERROR\_SEVERITY();

DECLARE @ErrorState INT = ERROR\_STATE();

RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState);

END CATCH

<<<<<<>>>>>>

DECLARE @TargetRecipeID INT;

SELECT @TargetRecipeID = (SELECT TOP 1 RecipeID FROM Recipes WHERE DishName = 'Fried Rice');

IF @TargetRecipeID IS NULL

BEGIN

RAISERROR('No recipe found with the name Fried Rice.', 16, 1);

RETURN;

END

SELECT

R.DishName AS RecommendedDish,

COUNT(\*) AS SharedIngredientCount

FROM

RecipeIngredients RI

INNER JOIN

RecipeIngredients RI2 ON RI.IngredientID = RI2.IngredientID AND RI.RecipeID != RI2.RecipeID

INNER JOIN

Recipes R ON RI2.RecipeID = R.RecipeID

WHERE

RI.RecipeID = @TargetRecipeID

GROUP BY

R.DishName

ORDER BY

SharedIngredientCount DESC, R.DishName;