**Part 1: Database Diagram**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **recipes** | **recipe\_classes** | **recipe\_ingredients** | **ingredients** | **ingredient\_classes** | **measurements** |
| RecipeID | RecipeClassID | RecipeID | IngredientID | IngredientClassID | MeasureAmountID |
| RecipeTitle | RecipeClassDescription | RecipeSeqNo | IngredientName | IngredientClassDescription | MeasurementDescription |
| RecipeClassID |  | IngredientID | IngredientClassID |  |  |
| Preparation |  | MeasureAmountID | MeasureAmountID |  |  |
| Notes |  | Amount |  |  |  |

**Primary KEY/Foreign KEY**

recipes: RecipeID (PK), RecipeClassID (FK) [recipe\_classes]

recipe\_classes: RecipeClassID (PK)

recipe\_ingredients: RecipeID + RecipeSeqNo (PK), RecipeID (FK) [recipes], IngredientID (FK) [ingredients], MeasureAmountID (FK) [measurements]

ingredients: IngredientID (PK), IngredientClassID (FK) [ingredient\_classes], MeasureAmountID (FK) [measurements]

ingredient\_classes: IngredientClassID (PK)

measurements: MeasureAmountID (PK)

**Table Relations**

recipes → recipe\_ingredients: one-to-many

recipe\_classes → recipes: one-to-many

ingredients → recipe\_ingredients: one-to-many

measurements → recipe\_ingredients: one-to-many

ingredient\_classes → ingredients: one-to-many

measurements → ingredients: one-to-many

**Part 2: SQL Statements/Queries**

1. Question/Task: Select all ingredients that are either a dairy or wine using a union. (HINT: Wine needs to be extracted from the ingredient name)
   1. **Your SQL Statement:** SELECT IngredientID, IngredientName, IngredientClassID FROM `ingredients` WHERE IngredientClassID = 8 UNION SELECT IngredientID, IngredientName, IngredientClassID FROM ingredients WHERE IngredientName LIKE '%wine%'
      1. I didn’t include MeasureAmountID because it seemed like the one thing that didn’t truly define an Ingredient and was instead import for recipes.
2. Question/Task: Display all the recipes that are main dishes.
   1. **Your SQL Statement:** SELECT RecipeID, RecipeTitle, Preparation FROM `recipes` WHERE RecipeClassID = 1
      1. I chose not to return RecipeClassID because the only recipes being returned here have ClassID of 1. I also chose not to include Notes because some of the data is NULL and doesn’t seem to be truly important to what makes a Recipe.
3. Question/Task: Display all the types of lettuces.
   1. **Your SQL Statement:** SELECT IngredientName FROM `ingredients` WHERE IngredientName LIKE '%lettuce%' ORDER BY IngredientName
      1. I chose to only return IngredientName because this seemed to be the only thing the Question was asking for. It would be no use to someone who is asking for all the different types of lettuces you have in a store to be told each lettuce and their IngredientID, IngredientClassID and MeasureAmountID because they may not even be looking to use these products in a dish or recipe, like how the table is built, and instead only wanted to know what types of lettuces are on offer. I also chose to order the data by the IngredientName because it looked nicer to me.
4. Question/Task: Display all types of Meat or Seafood ingredients.
   1. **Your SQL Statement:** SELECT IngredientID, IngredientName, IngredientClassID FROM `ingredients` WHERE IngredientClassID = 2 OR IngredientClassID = 10 ORDER BY IngredientClassID, IngredientName
      1. I chose to not return MeasureAmountID because I felt it wasn’t what defined an ingredient. I ordered the data by IngredientClassID to have the meat and seafood ingredients grouped and from there, I further ordered the data by IngredientName for a clean alphabetic listing.
5. Question/Task: Display the recipes in increasing order by the number of ingredients they have/use.
   1. **Your SQL Statement:** SELECT recipes.RecipeID, recipes.RecipeTitle, recipes.RecipeClassID, recipes.Preparation, ing.TotalIngredients FROM `recipes` INNER JOIN ( SELECT RecipeID, SUM(amount) AS TotalIngredients FROM `recipe\_ingredients` GROUP BY RecipeID ) AS ing ON recipes.RecipeID = ing.RecipeID ORDER BY ing.TotalIngredients
      1. I chose not to return Notes because the NULL values in the table lead me to believe the column isn’t truly needed to define a recipe. Since this problem wants us to account for how much of an ingredient a recipe USES, I did a SUM over the Amount column.
6. Question/Task: What measurements are used in making Fettuccini Alfredo, in units of measure, i.e., “Cup”?
   1. **Your SQL Statement:** SELECT DISTINCT `measurements`.`MeasurementDescription` FROM `recipe\_ingredients` INNER JOIN `measurements` ON `recipe\_ingredients`.`MeasureAmountID` = `measurements`.`MeasureAmountID` WHERE RecipeID = 5 ORDER BY `measurements`.`MeasurementDescription` ASC
      1. I ordered by MeasurementDescription for a cleaner data representation.
7. Question/Task: Display all recipes that require less than 5 ingredients.
   1. **Your SQL Statement:** SELECT recipes.RecipeID, recipes.RecipeTitle, recipes.RecipeClassID, recipes.Preparation FROM `recipes` INNER JOIN ( SELECT RecipeID, COUNT(RecipeID) AS TotalIngredients FROM `recipe\_ingredients` GROUP BY RecipeID ) AS ing ON recipes.RecipeID = ing.RecipeID WHERE ing.TotalIngredients < 5 ORDER BY recipes.RecipeID
      1. I again chose not to return Notes because it isn’t required to define a Recipe. I also didn’t return the amount of TotalIngredients used in each recipe since the problem only wants us to return recipes with less than 5 ingredients. Since this problem only wants us to account for the number of ingredients the recipe has, instead of uses, I did a Count over RecipeID to essentially count the number of rows with a particular RecipeID, since each new row is a new ingredient.
8. Question/Task: Display all recipeID's that have 3 or more ingredients (without using aggregate functions)
   1. **Your SQL Statement:** SELECT DISTINCT RecipeID FROM recipe\_ingredients WHERE RecipeSeqNo >= 3 ORDER BY RecipeID
9. Question/Task: Display all recipeID's that have 3 or more ingredients (using aggregate functions)
   1. **Your SQL Statement:** SELECT recipes.RecipeID FROM `recipes` INNER JOIN ( SELECT RecipeID, COUNT(RecipeID) AS TotalIngredients FROM `recipe\_ingredients` GROUP BY RecipeID ) AS ing ON recipes.RecipeID = ing.RecipeID WHERE ing.TotalIngredients >= 3 ORDER BY recipes.RecipeID
      1. Since this problem only wants us to account for the number of ingredients the recipe has, instead of uses, I did a Count over RecipeID to essentially count the number of rows with a particular RecipeID, since each new row is a new ingredient.
10. Question/Task: List every ingredient and its corresponding recipeID. Also include ingredients that are **not** in any recipes. (**Hint:** Use left outer join).
    1. **Your SQL Statement:** SELECT ingredients.IngredientName, recipe\_ingredients.RecipeID FROM `ingredients` LEFT OUTER JOIN recipe\_ingredients ON recipe\_ingredients.IngredientID = ingredients.IngredientID ORDER BY ingredients.IngredientClassID, ingredients.IngredientName, ingredients.IngredientID, ingredients.MeasureAmountID, recipe\_ingredients.RecipeID
       1. I couldn’t figure out the ordering based on the partial output given to us for this problem. That’s why I have so many orderings.