TEAM 035 STAGE 3

Provide a screenshot of the connection (i.e. showing your terminal/command-line information):

```
Welcome to Cloud Shell! Type "help" to get started.
anagha_tiwari@cloudshell:~ (cs411-415303) $ gcloud sql connect team035 --user=root --quiet
Allowlisting your IP for incoming connection for 5 minutes...done.
Connecting to database with SQL user [root].Enter password:
Welcome to the MySQL monitor. Commands end with; or \g.
Your MySQL connection id is 89582
Server version: 8.0.31-google (Google)

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

DDL Commands:

In total, we have 5 tables. Their DDL command code is shown below:

```
CREATE TABLE User (
UserID VARCHAR(255),
UserName VARCHAR(100),
Password VARCHAR(20),
PRIMARY KEY (UserID)
);
CREATE TABLE Recipe (
  RecipeTitle VARCHAR(225),
  Ingredients TEXT,
  Directions TEXT,
  PRIMARY KEY (RecipeTitle)
);
CREATE TABLE Food (
FoodName VARCHAR(225),
Fat INT,
Protein INT,
Carbs INT.
Measures INT.
Category VARCHAR(255),
PRIMARY KEY (FoodName)
);
```

```
CREATE TABLE Favorites (
FavoriteID VARCHAR(255),
DateAdded int.
UserID VARCHAR(255),
PRIMARY KEY (FavoriteID, UserID)
);
CREATE TABLE MyRecipes (
RecipeTitle VARCHAR(255),
Ingredients TEXT,
Directions TEXT,
UserID VARCHAR(255),
PRIMARY KEY (RecipeTitle, UserID)
);
(We know we should use this DDL for the Favorites and MyRecipes table but it did not work on
GCP when we put the foreign key in the DDL command so we used DDL command written
above.
CREATE TABLE Favorites (
FavoriteID VARCHAR(255),
DateAdded int,
UserID VARCHAR(255),
PRIMARY KEY (FavoriteID, UserID),
FOREIGN KEY (UserID) REFERENCES User(UserID)
);
CREATE TABLE MyRecipes (
RecipeTitle VARCHAR(255),
Ingredients TEXT,
Directions TEXT,
UserID VARCHAR(255),
PRIMARY KEY (RecipeTitle),
FOREIGN KEY (UserID) REFERENCES User(UserID)
);
)
```

(In our diagram in Stage 2, we had ThumbsUp as an attribute in MyRecipes. But we decided to remove this since we will rank users with other factor to rank the users such as how many recipes they uploaded as shown in query 4)

```
CREATE TABLE have (
RecipeTitle varchar(255),
FoodName varchar(255),
PRIMARY KEY (RecipeTitle, FoodName));
CREATE TABLE Contains (
FavoriteID varchar(255),
RecipeTitle varchar(255),
PRIMARY KEY (FavoriteID, RecipeTitle));
```

Inserting at least 1000 rows in at least 3 tables. (You should do a count query to show this): As shown below, our 3 tables: User, Food, and Recipe have at least 1000 rows each.

SQL QUERIES:

1. Filter out the recipes using ingredients that contain proteins more than 15g.

Get the average value of protein per category by using Group by. Select the categories that have an average value of proteins that is greater than 15 in the subquery. Then use the selected categories to filter out the recipes that use the ingredients in those categories. Recipe table is joined with Food table by seeing the Food.FoodName is found in Recipe.Ingredients.

This query will be used for the filtering functionality that helps the user to find the recipes that include a specific amount of some nutrition; such as recipes that include carbs less than 10, and recipes that include fat less than 5.

SELECT r.RecipeTitle, r.Ingredients, r.Directions
FROM Recipe r JOIN Food f ON r.Ingredients LIKE CONCAT('%', f.FoodName, '%')
WHERE f.Category IN (
 SELECT Category
 FROM Food
 GROUP BY Category
 HAVING AVG(Protein) > 15

) LIMIT 15;

RecipeTitle	Ingredients	
	["1 1/3 c. oi1", "2 c. sugar", "2 eggs", "3 c. flour", "1/4 tsp. salt", "1 tsp. baking soda", "2	["Beat together first three ingredients.", "Mix together in separate bowl all dry ingredients.", "Add oi
1 mixture to dry mixture.", "Pur Cream Cheese Fruit Squares,"["1	ee apple pie filling and add to mixture along with chopped pecans.", "Bake at 350\u00b0 in greased a 1/2 c. sugar, divided in half", "1/3 c. butter", "1 1/2 c. graham cracker crumbs", "8 oz. cream chee	nd floured tube pan for 1 hour and 15 minutes.", "Freezes well."]" se", "4 eggs", "1 tsp. vanilla", "1 can fruit topping mix (cherry, blueberry, apple or boysenberry)"]
Creole Green Beans	["1 green pepper, chopped", "1 onion, chopped", "2 thap. bacon drippings", "1 thap. flour", "1 tag.	p. salt", "l tsp. sugar", "dash of paprika", "l (16 oz.) can tomatoes", "l (16 oz.) can green peas") ["Saute green pepper and onion in hot bacon drippings until tender. Add flour, salt, sugar and paprika."
, "Stir until blended. Add tomate Banana Bread,"["2 eggs", "3 banas	oes and simmer for about 15 minutes.", "Stir in green peas; heat through and serve." " nas, cut in small pieces", "1 stick butter, cut in small pieces", "1 c. sugar", "1 3/4 c. flour", "3	/4 tsp. soda", "1 1/4 tsp. cream of tartar", "1 c. chopped nuts (optional)"]
I Cornbread Salad	["2 pkg. jiffy cornbread mix, cooked, cooled and crumbled", "1 1/2 c. chopped onion", "1 medium g	reen pepper, chopped", "3 medium firm tomatoes", " to 10 slices bacon, cooked, drained and crumbled", "1 ["Mix well and let set overnight.", "Will make 2 quarts and 1 pint.", "You can cut this in half."]"
a (much better)", "8 to 10 oz. g n and layer of whole kernel corn Apple Crisp Pie,"("1 graham crac p. ground nutmeg", "3/4 c. all-p! Clan Chowder and pepper to taste" ickened, if necessary." "	rated cheddar chesse"] ", "Top with grated chesse.", "Cover and bake for 1 hour at 375\w00b0. Serves 6 to 8."]" ker crust", "1 large egg yolk", "5 1/2 c. sliced apples", "1 tbsp. lemon juice", "1/2 c. sugar", "1/ urpose flour", "1/4 c. sugar", "1/4 c. light brown suger", "1/3 c. butter"]	ernel corn, drained", "I can creamed corn", "I can okra and tomatoes or frozen okra or fresh smothered okr "Put in buttered baking dish a layer of okra, layer of creamed corn, layer of onion, green papper, bacc 4 c. light brown sugar", "I thap. all-purpose flour", "1/4 tap. salt", "1/2 tap. ground cinnamon", "1/4 tap. ced class, drained (reserve I c. clas juice)", "2 c. half and half", "1 c. water", "2 thop, butter, "salt" "Dice bacon and onion. Saute in soup pot. Add remaining ingredients.", "Meaty do not boil.", "May be th
	 ["4 tbsp. bacon drippings", "1 tsp. salt", "3 tbsp. self-rising flour", "2 c. whole milk"]	["Add bacon grease, salt and flour.", "Stir until brown.", "Add milk slowly, stirring until thickens.",
"Serves 8."]" Italian Pasta And Bean Soup(Past		esh tomatoes", "2 (13 3/4 oz.) cans beef broth", "1 than, minced paraley", "1/2 tan, salt", "1/4 tan, free
h ground pepper", "1/8 tsp. basi		a, cooxed \"ai dente\"", "grated cneese" c. sunflower seeds", "1/2 c. raisins", "6 slices bacon, fried and cooled", "1/2 c. mayonnaise", "2 tbsp. v
inegar", "1/8 c. sugar"] "Low-Fat "French Fries"		["Mix broccoli, onion, sunflower seeds, raisins and bacon."]"
Broccoli Cauliflower Salad	["1 head broccoli, chopped", "1/2 head cauliflower, chopped", "red onion, chopped to taste", "shr	edded low-fat cheese", "bacon bits", "1 c. fat-free miracle whip", "3 pkg. aweet 'n low", "2 thap, vinegar ["Mix together the Miracle Whip, Sweet 'N Low and Vinegar.", "Four over chopped vegetables.", "Add bacor
onion and bacon.", "Add re Cranberry Relish,"["1 lb.	emaining ingredients. Bake in bean pot at 350\u000b0 for 35 to 40 minutes."]" cranberries", "1 c. raisins", "1 2/3 c. sugar", "1 c. water", "1 1/2 tsp. ginge	", "I tsp. cinnamon", "1/2 tsp. cloves", "1/2 c. celery", "I c. chopped apple"]
Bacon-Flavored Chicken E	Breasts ["8 boneless chicken breasts", "1 jar chipped beef", "8 slices bacon",	"1 can cream of mushroom soup", "1 c. sour cream"]
Decadent Fudge Cake,"["1 of te mini morsels, divided", tsp. shortening, divided", Bacon Nuts	om of baking dish.", "Wrap chicken breasts with bacon.", "Mix sour cream and soup. butter or margarine, softened", "1 1/2 c. sugger", "4 eggs", "1/2 tap, baking; "2 (4 oz.) bars sweet baking chocolate, melted and cooled", "1/3 c. chocolate "chocolate and white chocolate leaves (optional)"] ["1 can water chestnuts, drained (8 oz.)", "1/4 c. soy sauce", "1/2 lb. auce for 4 hours.", "Wrap each chestnut in a 1/2 strip of bacon and secure with a	<pre>ioda*, "1 c. buttermilk*, "2 1/2 c. all-purpose flour*, "1 1/2 c. semi-sweet chocola uyrup", "2 tsp. vanilla extract", "4 oz. white chocolate, chopped", "2 tbsp. plus 2 .bacon"]</pre> <pre> ["Soak</pre>
Pina Colada Cake, "["1 box oconut"]	duncan hines butter cake mix", "4 eggs", "1/2 c. oil", "1 c. water", "1 can crea	um of coconut", "l can crushed pineapple (optional)", "l large cool whip", "flaked c
Bacon Corn Chowder el corn", "2 c. milk", "2 potatoes and drain.", "Fr Quick Freezer Fudge,"["1 h	["1/2 lb. bacon, cut in 1/2 inch pieces", "4 medium potatoes, cut in 1/2 salt", "pepper to taste"] by bacon, then brown onion in fat. Add rest of ingredients.", "Simmer 10 minutes. confectioners sugar", "1/2 c. cocoa", "1/4 tsp. salt", "1/4 c. milk", "1 tbsp. confectioners sugar", "1/2 c. cocoa", "1/4 tsp. salt", "1/4 c. milk", "1 tbsp.	/2 inch cubes", "1 medium onion, chopped", "2 c. cream-style corn", "2 c. whole kern ["Cook
skim milk", "1/2 tsp. pepp eat oven to 450\u00b0.", ' 350\u00b0. Bake for 10 to per serving."]"	per"] 'Spread bacon in bottom of 9-inch pie plate.", "Sprinkle with cheese.", "Beat ego	c. finely chopped muenster, cheddar or monterey jack cheese", "4 eggs", "1 1/2 c. ["Preh gs with milk and pepper. Pour over cheese.", "Bake for 15 minutes.", "Reduce heat to a rack to cool for 10 minutes.", "Cut in wedges.", "Yields 10 servings, 249 calories", "dash of salt"]
15 rows in set (0.01 sec)		

2. Filter out the recipes that use ingredients from some specific categories (e.g. Dairy products and Seafood is selected).

Filter out ingredients that use ingredients in the Dairy products category.

Join the Food table (where it was filtered out by seeing if the category of the food is Dairy products) with Recipe tables by seeing if the Food.FoodName is included in the Recipe.Ingredients. Then do the same thing for the category Seafood.

Finally, use set operator Intersect to output all of the recipes that use Dairy products and Seafood.

This query will be used to filter out the recipes according to the user's category selection when we give the recipe recommendation on our app.

```
(
SELECT r.RecipeTitle, r.Ingredients, r.Directions
FROM Recipe r
JOIN Food f ON r.Ingredients LIKE CONCAT('%', f.FoodName, '%')
WHERE f.Category = 'Dairy products'
)
INTERSECT
(
SELECT r.RecipeTitle, r.Ingredients, r.Directions
FROM Recipe r
JOIN Food f ON r.Ingredients LIKE CONCAT('%', f.FoodName, '%')
WHERE f.Category LIKE '%Seafood'
) LIMIT 15;
```

RecipeTitle	Ingredients
	Directions
+	
Baked Tuna Chow Mein Casserole milk and 1/4 c. water", "3 oz. (1 1/2 c	[*1 c. chopped celery", *1/4 c. chopped onion*, *1/4 c. chopped green pepper*, *1 tbsp. butter*, *1 (7 oz.) can tuna*, *1 [10 1/2 oz.) can cream of mushroom soup, thinned with 1/4 c) chow mein modiles (save 1/3 c. for topping)*, *1 can water chestnute*, *3/4 c. (4 oz.) salted cashew nuta*, *1/8 tp, pepper*] [*Beat oven to 350\U005000*, "Saute celery, onion and green pepper in butter.*, *Mix1 nest of ingredutes pour into there of 1/2-quart baking dish.*, *Sprinkle with 1/3 cup chow
mein noodles. Bake 30 minutes.", "Make	[["Mast oven to solvulubul", "Saute celery", onlow and green pepper in butter.", "Mix in rest of ingredients; pour into buttered 1 1/2-quart baking dish.", "Sprinkle with 1/3 cup chows is 4 to 5 servings."]" turkey", "1 c. diced potatoes", "1 c. cut celery", "1 c. mushroom soup", "cracker crumbs"]
Battered Fried Fish	["1 (1 lb.) haddock", "salt", "1 egg", "1 1/4 c. evaporated milk", "1 c. flour", "2 tsp. baking powder", "1/4 c. cornmeal"]
at at 375\u00b0 for 5 minutes or until	["Cut haddock", "into serving-size pieces, sprinkle lightly with salt.", "Mix\tl/4 teaspoon salt with remaining ingredients, dip haddock pieces into", "egg mixture.", "Fry in deep f brown."] milk, scalded", "5 round tbsp. flour", "2 eggs, slightly beaten", "dash of salt", "1 tbsp. butter or margarine", "1 tsp. vanilla"]
	milk, Scaledor, "5 round thep. flour, "2 eggs, slightly beater", "dash of Sait", "1 thep. butter or margarine", "1 tag. vanila"] ["1 can salmon, drained and deboned", "1/3 c. celery, chopped", "1/3 c. onion, chopped", "3 eggs", "1 c. cheddar cheese, shredded", "1 thep. lemon juice", "1/2 tap. sait", "1/8 tap. page", "1 c. cheddar cheese, shredded", "1 thep. lemon juice", "1/2 tap. sait", "1/8 tap. page", "1 c. cheddar cheese, shredded", "1 thep. lemon juice", "1/2 tap. sait", "1/8 tap. page", "1 c. cheddar cheese, shredded", "1 thep. lemon juice", "1/2 tap. sait", "1/8 tap. page", "1 c. cheddar cheese, shredded", "1 thep. lemon juice", "1/2 tap. sait", "1/8 tap. page", "1 c. cheddar cheese, shredded", "1 thep. lemon juice", "1/2 tap. sait", "1/8 tap. page", "1 c. cheddar cheese, shredded", "1 thep. lemon juice", "1/2 tap. sait", "1/8 tap. page", "1 c. cheddar cheese, shredded", "1 thep. lemon juice", "1/2 tap. sait", "1/8 tap. page", "1 c. cheddar cheese, shredded", "1 thep. page", "1 tap. page"
epper", "1/2 c. bread crumbs", "1/2 c.	can season, statue and seconder, "17 S. Celery, chopped," -17 S. Onlon, chopped, "-2 eggs, "-1 C. chestar cheese, Shresberg, "1 Cesp. lemon juice", "17 Esp. saic", "17 Esp. s
Tuna marbers, "["I (8 02.) can tuna, dra	tind, I tog, minord ditum, "I tap, sembli jutor", "IZ togo, mustato", "I egg , "IZ C. detenda", "Crackez of bread crumbs";
Crab Meat Bisque	["3 cans cream of celery soup", "3 cans milk", "1/2 tsp. old bay seasoning", "1 box cooked, chopped broccoli", "1 stick butter", "1 lb. crab meat"]
Virginia'S Christmas Salad,"["1 (3 oz.)	["Mix and heat ingredients and add crab meat at the end."]" cherry jell-o", "I c. boiling water", "1/4 c. nuts", "1/2 c. chopped celery", "I can whole cranberry sauce", "I c. sour cream", "1/2 c. diced apple"]
Crab Mold(Dip)	[["1/4 c. mayonnaise", "1 can cream of mushroom soup", "6 oz. cream cheese", "1 envelope unflavored gelatin", "3 thsp. cold water", "3 green onions, chopped", "1/4 c. chopped celery",
"I small jar chipped pimentos", "dash o se mixture.", "Pour soup into cream che	f salt and pepper or seasoning of choice (mrs. dash is good!", "! (T or.) can crab meat") "Cream mayonnaise and cream cheese together.", "Marm soup in small pan.", "Dissolve gelatin in water and add to soup.", "Add onlons, celery, pimentos and seasonings into cream chee eee mixture; stir well.", "Add crab; mix well. Bour into a mold.", "Befrigerate hour or more.", "Can serve from bowl like a dip or loosen by setting mold in hot water until side of mi
xture loosens from mold pan, then turn Fruit Dip, "["3 oz. pkg. instant vanilla	onto dish and serve with crackers or vegetables "]" pudding", "8 oz. cool whip", "1 tep. cinnamon", "1/2 tsp. cloves", "1/2 tsp. nutmeg", "2 to 3 tbsp. amaretto"]
Crab Spread(Appetizer)	["8 oz. cream cheese", "1 thsp. mayonnaise", "1 thsp. worcestershire sauce", "1 tsp. lenon juice", "garlic to taste", "parsley flakes", "1 lb. cooked shrimp or crab flakes", "cocktail
sauce"] e with chips or crackers."]"	["Mix all ingredients, except parsley, shrimp (or crab) and cocktail sauce.", "Spread on glass plate.", "Top with cocktail sauce and shrimp (or crab); sprinkle with parsley.", "Serv
Crispy Baked Onion Rings,"["2 large swe	et onions", "1 (7 oz.) pkg. corn flakes", "1 tsp. seasoned salt", "2 tsp. sugar", "1 tsp. paprika", "1 c. egg substitute")
Crabmeat Au Gratin opped", "1/3 c. slivered almonds", "4 h	["1 can cream of shrims soup", "1/2 can milk (use soup can)", "1 lb. faux crahmest", "4 c. celery, chopped", "1/2 c. green pepper, chopped", "2 pimentos, chopped", "2 thep. onions, chard-cocked eggs, chopped", "1 c. shredded sharp cheddar choses," butter of bread crumbs." ["Mix everything together, except cheese and crumbs, in casserole.", "Top with cheese, then with bread crumbs.", "Bake at 380,0000 for 35 minutes."]" c. actury," "1/4 c. nustrad", "1/4 c. barbeous sauce", "2 (if cs.) cans siloned potatoses, drained", "2 (foo.) cans prok and beams," 1 (if col.) can which kernel corn, drained", "2 thep.
minced onion flakes"]	
	("1/2 head cabbage", "2 large carrots", "1 onion", "2 c. meat (hamburger, shredded pork, chicken, shrimp)", "2 thap, soy sauce", "egg roll skins") ("Shred cabbage", "Dice carrots and onion", "Cook meat and drain grease.", "Add yenetables to skillet and stir-fry until crisp-tender.", "Mix in soy sauce.", "Place a heaning table
espoonful of filling in center of each ination of vegetables may be used."]" Round Steaks Smothered In Onions."["2 1	["Shred cabbage.", "Dice carrots and onion.", "Gook meat and drain grease.", "Add vegetables to skillet and stir-fry until crisp-tender.", "Mix in sey sauce.", "Flace a heaping table egg roll skin.", "Fold in sides, then fold over long sides, sealing shut with cornstarch and water paste. Deep fry until golden brown.", "Serve with sweet and sour sauce. Any other comb
Layered Tuna Salad dded lettuce", "1 1/2 c. (4 cz.) medium "1 c. (4 cz.) shredded cheddar cheese" read dressing mixture over bacon, seali Chicken Divan Pot Pie, "["2 pet-rizz reg (10 cz.) pkg. chopped broccoli, thawed	["1 1/2 c. miracle whip salad dressing", "1/4 c. green onion, sliced", "2 tap. dijon mustard", "2 (6 1/2 oz.) cans tuns, drained and flaked", "1 tap. dill weed (optional)", "4 c. shree abell meazeroni, cooked and drained", "2 c. chopped tocamber", "2 c. chopped tocamber", "1 (10 oz.) pyg. frozen peas, thaved and drained", "6 ozen chicse, crisply cooked and crumbled", [(Combine dressing, green onions and marter) set aside. "Toss tuns with dill.", "1 a 3-mart serving bowl, layer lettine, shells counsely, counsely, tuns, peas and become, "50 ozen and 50
	["3 thps, butter or oleo", "3 thps, flour", "1 tep, salt", "1/4 tap, pepper", "1 c. milk", "1/2 c. cream or half and half", "1 to 1 1/2 c. cooked lobater meat"] ["Malt butter", "Mad flour and seasonings.", "Stir until smooth. Add milk and cream slowly; cook over low heat until thickened. Heat slowly.", "Do not let boil."]" crunchy peanut butter", "1/2 c. nonfat dry milk"]
	["1 pt. clams (bay quahaugs)", "1/4 c. salt pork or bacon, chopped", "1/4 c. chopped onion", "1/2 tsp. salt", "dash of pepper", "1 c. clam liquor (add water to make i c.)", "1 c. dice garmiah"] ["Drain clams; save liquor and chop.", "Fry salt pork or bacon until lightly brown.", "Add onion and cook until tender.", "Add liquor, potatoes and seasonings.", "Cook 15 minutes un
til potatoes are tender.", "Add milk an Cheesecake,"["1 large pkg. lemon jello"	d best thoroughly.", "Stir in chopped clams and warm gently 2 to 5 minutes.", "Garnish with parsley."]" , "2 1/4 c. graham cracker crumbs", "1 cream cheese, softened", "1 can pet milk, chilled", "1 c. sugar"]
Quick One Dish Meal	["1" box kraft macaroni and cheese", "1 green pepper, chopped", "1 (6 oz.) can tuna", "1 small onion, chopped"]
Instant Spiced Russian Tea,"["1 c. tang	["Prepare macaroni and cheese as directed; add green pepper, onion and tuna.", "Mix well and let set for 5 or 10 minutes with the lid on."]" ", "1/4 c. sugar", "1/2 c. preswestened lemon flavored instant tea", "1/2 tsp. cinnamon", "1/2 tsp. ground cloves"]
1	
Dandy Candies,"["1/3 c. honey", "1/2 c.	crunchy peanut butter", "1/2 c. nonfat dry milk"]
New England Clam Chowder(Serves 6) d potatoes", "2 c. milk", "parsley for	["1 pt. clams (bay quahaugs)", "1/4 c. salt pork or bacon, chopped", "1/4 c. chopped onion", "1/2 tsp. salt", "dash of pepper", "1 c. clam liquor (add water to make 1 c.)", "1 c. dice garmish"]
til potatoes are tender.", "Add milk an Cheesecake,"["1 large pkg. lemon jello"	[Train class; save liquor and chop.", "Ery salt pork or bacon until lightly brown.", "Add onton and cook until tender.", "Add liquor, potatoes and seasonings.", "Cook 15 minutes und best thoroughly.", "Stir in chopped class and warm gently 2 to 5 minutes.", "Garnish with paraley."]" , '2.1/4 c. graham cracker crumbo", "l crass cheese, softened", "l can pet milk, childer, "l c. sugar")
Quick One Dish Meal	 ["1 box kraft macaroni and cheese", "1 green pepper, chopped", "1 (6 oz.) can tuna", "1 small onion, chopped"]
Instant Spiced Russian Tea,"["1 c. tang	["Prepare macaroni and cheese as directed; add green pepper, onion and tuna.", "Mix well and let set for 5 or 10 minutes with the lid on."]" , "1/4 c. sugar", "1/2 c. presweetened lemon flavored instant tea", "1/2 tsp. cinnamon", "1/2 tsp. ground cloves"]
 Salmon Ball	["2 c. (1 lb. can) drained, canned salmon", "1 (8 oz.) pkg. soft cream cheese", "1 tbsp. minced onion", "1/4 tsp. salt", "1 tbsp. lenon juice", "1 tsp. prepared horseradish", "chopped
narslev")	['Th box], combine all ingeredients, except paralse,", "Mix well. Refrigerate about thours or until firm enough to shape into bail.", "Shape in bail; roll in paraley (and chopped sast i hour before serving.", "Serve with crackers or small slices of rye bread.", "Makes 5-inch ball."]" attices, "mozza-cracial cheese", "regular cheese," "cottage cheese," "gottage cheese," gottage cheese, "got
Salmon Casserole Baked In Sour Cream iced", "1 tbsp. chopped parsley"	("1 (1 lb.) can salmon, drained and juice reserved", "1 thap. lemon juice", "1/2 tap. salt", "pepper to taste", "1 c. sour cream", "1/2 tap. dry dill weed", "1 medium onion, thinly sl
erole.", "Pour sour cream sauce over sa Carrie'S Cabbage Soup,"["6 c. shredded	["Place salmon liquid in mixing bowl, add lemon juice, asit, peoper, sour cream and dily weed," "Mix to smooth sauce," Break salmon in large chunks place in greased l-quart cass almon.", "Spread with onion; sprinkle with paraley,", "Cover and bake at 350/u0000 for 30 minutes.", "Hields 4 servings."]" green cobbage," 1c. chopped onion," 2c. diced celery," 1/2c. diced green peoper, "1/2 tap, thyme," 1/4 tap, majoram, "I thop, anisette", "6e peled ripe red tomatoes", 2c pkg. low, "water", "1/4 tap, the people of the p
	lon", "water", "1/4 tsp. tarragon", "2 tbsp. granulated sugar", "8 oz. tomato sauce", "1 lb. ground beef, browned (optional)", "1 lb. sausage (link, polish, italian, kielbasi or ground, ("1 (14 3/4 oz.) can pink salmon, drained", "1 1/2 c. mashed potatoes", "1 small onion, grated", "1 large egg", "1/2 tsp. pepper", "1/4 c. packaged bread crumbs", "3 tbsp. vegetable o
<pre>il"] at.", "Heat oil in a large nonstick ski</pre>	["Mix salmon, potatoes, onion, egg and pepper until blended. Form into 8 patties, each about 3/4-inch thick.", "Spread crumbs on waxed paper.", "Gently press patties in crumbs to co llet.", "Add patties and cook over medium heat about 4 minutes per side until heated through, golden and crisp. Serves 4."]"
"Croquettes (Chicken, Fish Or Meat)	

3. Show the ranking of recipes according to how many users marked those recipes as favorites.

Group by Contains.RecipeTitle and store the the count of favoriteID for each RecipeTitle into FavoriteCount. Join Favorite and Recipe tables using the Contains table. (many to many relationship, Contains is the relationship table) Then list up the RecipeTitle in descending order by the FavoriteCount.

This will be used to recommend the recipes according to which recipes are popular to the users.

```
SELECT
c.RecipeTitle,
COUNT(c.FavoriteID) AS FavoriteCount
FROM
Contains c
JOIN
Favorites f ON c.FavoriteID = f.FavoriteID
GROUP BY
c.RecipeTitle
ORDER BY
FavoriteCount DESC
LIMIT 15;
```

RecipeTitle	FavoriteCount			
No-Bake Nut Cookies	12			
Creamy Corn	3			
Reeses Cups(Candy)	3			
Jewell Ball'S Chicken	2			
Chicken Funny	1			
Cheeseburger Potato Soup	1			
Rhubarb Coffee Cake	1			
Scalloped Corn	1			
Nolan's Pepper Steak				
Millionaire Pie				
Double Cherry Delight Buckeye Candy				
Quick Barbecue Wings	l 1 l			
Pink Stuff(Frozen Dessert)	1 1			
Fresh Strawberry Pie	1 1			
++				
15 rows in set (0.00 sec)				

4. Show the ranking of users according to how many MyRecipes they uploaded.

Join User and MyRecipes using the right outer join.

Group by UserID to get count of how many RecipeTitle was uploaded by one user and store that count to RecipesUploaded. Then make a list of UserIDs in descending order of that RecipesUploaded.

Return the list of UserName according to that UserID list and show the RecipesUploaded beside the names.

This will be used to show the ranking of users according to the contribution to encourage participation.

SELECT
User.UserName,
COUNT(MyRecipes.RecipeTitle) AS RecipesUploaded
FROM
User
RIGHT JOIN
MyRecipes ON User.UserID = MyRecipes.UserID
GROUP BY
MyRecipes.UserID
ORDER BY

RecipesUploaded DESC;

+ UserName	RecipesUploaded	
Matthew Allen	8 6 5 4 2 1 1 1 1 1 1 1 1 1 1	
Zoey Carter Scarlett Walker	1 1	
John Sanchez	1 1 1	
Nathan Collins Gabriel Perez	1 1 1	
++ 27 rows in set (0.00 sec)		

INDEXING ANALYSIS

Advanced Query #1

```
SELECT r.RecipeTitle, r.Ingredients, r.Directions
FROM Recipe r JOIN Food f ON r.Ingredients LIKE CONCAT('%', f.FoodName, '%')
WHERE f.Category IN (
  SELECT Category
  FROM Food
  GROUP BY Category
  HAVING AVG(Protein) > 15
) LIMIT 15;
```

1. Index Design #1: No Index

Cost: 121292.00 (using Explain Analyze command)

```
| -> Limit: 15 row(s) (cost=121292.00 rows=15) (actual time=8.447..11.549 rows=15 loops=1)
-> Filter: (<in_optimizer>(f.Category, f.Category in (select #2)) and (r.Ingredients like concat('$', f.FoodName, '$'))) (cost=12129
2.00 rows=1211420) (actual time=8.446..11.546 rows=15 loops=1)
-> Inner hash join (no condition) (cost=121292.00 rows=1211420) (actual time=1.809..6.354 rows=4570 loops=1)
-> Table scan on f (cost=0.14 rows=1190) (actual time=0.677..0.684 rows=13 loops=1)
-> Table scan on f (cost=0.14 rows=1190) (actual time=0.677..0.684 rows=13 loops=1)
-> Table scan on f (cost=0.14 rows=1190) (actual time=0.677..0.684 rows=13 loops=1)
                                           -> Table scan on r (cost=126.05 rows=1018) (actual time=0.043..0.377 rows=356 loops=1)
                     -> Select #2 (subquery in condition; run only once)
-> Filter: ((f.Category = `<materialized_subquery>`.Category)) (cost=0.00..0.00 rows=0) (actual time=0.108..0.108 rows=0
                                   -> Limit: 1 row(s) (cost=0.00..0.00 rows=0) (actual time=0.107..0.107 rows=0 loops=13)
-> Index lookup on <materialized_subquery> using <auto_distinct_key> (Category=f.Category) (actual time=0.107..0.
                                                         -> Materialize with deduplication (cost=0.00.0.00 rows=0) (actual time=1.381.1.381 rows=5 loops=1)
-> Filter: (avg(Food.Protein) > 15) (actual time=1.355..1.363 rows=5 loops=1)
-> Table scan on <temporary> (actual time=1.349..1.352 rows=22 loops=1)
-> Aggregate using temporary table (actual time=1.348..1.348 rows=22 loops=1)
-> Table scan on Food (cost=121.25 rows=1190) (actual time=0.101..0.513 rows=1190 loops=1)
```

Explanation: We first ran our EXPLAIN ANALYZE command when we did not have any index, so that we would have a baseline to compare our possible indexes.

Index Design #2: CREATE INDEX food cat idx on Food(Category); Cost: 121292

Explanation: This did not improve the overall cost of the query. We believe that creating an index on the food category key does not affect the overall cost because the category attribute stores non-numerical data that cannot be "naturally ordered," and would still require a search through every single value.

3. Index Design #3: CREATE INDEX food protein idx on Food(Protein); Cost: 121292.00

```
er: (<in_optimizer>(f.Category,f.Category in (select f2)) and (f.Ingredients like contact ( ,...ocmum, , ws=15 loops=1)
Inner hash join (no condition) (cost=121292.00 rows=1211420) (actual time=0.518..5.101 rows=4570 loops=1)
-> Table scan on f (cost=0.14 rows=1190) (actual time=0.039..0.047 rows=13 loops=1)
                            sh she scan on r (cost=126.05 rows=1018) (actual time=0.039..0.364 rows=356 loops=1)

*2 (subquery in condition; run only once)

lter: ((f.Category = 'cmaterialized subquery'.Category)) (cost=0.00..00 rows=0) (actual time=0.148..0.148 rows=0 loops=13)

-> Index lookup on <materialized subquery susing <auto distinct keys (Category=f.Category) (actual time=0.148..0.148 rows=0 loops=13)

-> Materialize with deduplication (cost=0.00..00 rows=0) (actual time=1.096..1906 rows=5 loops=1)

-> Filter: (avg (Food.Protein) > 15) (actual time=1.889..1.897 rows=5 loops=1)

-> Table scan on <a href="temporary">temporary</a> (actual time=1.881..1.884 rows=22 loops=1)

-> Aggregate using temporary table (actual time=1.879..1.879 rows=22 loops=1)

-> Table scan on Food (cost=121.25 rows=1190) (actual time=0.510..1.019 rows=1190 loops=1)
```

Explanation: We thought to create an index on the Food Protein key since our WHERE clause includes filtering the average amount of protein > 15. This did not change the overall cost because we believe that the protein attribute wasn't used for inner hashing join operations and neither was it used for table scanning. Further, in our original non-index cost-time analysis, there was no cost for finding the average(food.protein) > 15, which is why we think indexing on protein does not affect overall cost.

Index Design #4: CREATE INDEX recipe_ing_idx ON Recipe(Ingredients(255));
 Cost: 121292.00

```
| -> Limit: 15 row(s) (cost=121292.00 rows=15) (actual time=7.642..11.157 rows=15 loops=1)
-> Filter: (<in_optimizer>(f.Category, f.Category in (select #2)) and (r.Ingredients like concat('%', f.FoodName, '%'))) (cost=121292.00 rows=1211420) (actual time=7.640..11.154 rows=15 loops=1)
-> Inner hash join (no condition) (cost=121292.00 rows=1211420) (actual time=0.539..5.507 rows=4570 loops=1)
-> Table scan on f (cost=0.14 rows=1190) (actual time=0.040..0.051 rows=13 loops=1)
-> Hash
-> Table scan on r (cost=0.14 rows=1190) (actual time=0.031..0.381 rows=356 loops=1)
-> Select #2 (subquery in condition; run only once)
-> Filter: ((f.Category = 'cmaterialized subquery'. Category)) (cost=0.00..00 rows=0) (actual time=0.128..0.128 rows=0 loops=13)
-> Limit: 1 row(s) (cost=0.00..0.00 rows=0) (actual time=1.618..1.634 rows=5 loops=1)
-> Materialize with deduplication (cost=0.00..0.00 rows=0) (actual time=1.618..1.624 rows=5 loops=1)
-> Materialize with deduplication (cost=0.00..0.00 rows=0) (actual time=1.561..1.624 rows=5 loops=1)
-> Filter: (avg(Food.Protein) > 15) (actual time=1.616..1.624 rows=22 loops=1)
-> Aggregate using temporary table (actual time=1.660..1.609 rows=22 loops=1)
-> Aggregate using temporary table (actual time=1.609..1.609 rows=22 loops=1)
-> Table scan on food (cost=121.25 rows=1190) (actual time=0.231..0.688 rows=1190 loops=1)
```

Explanation: We thought to create an index on the Recipe Ingredients because that is one of the non-primary keys where we do a JOIN operation. However, this does not change the overall cost because Ingredients is of type "TEXT," and doing indexing on such large key types can be inefficient and not change the cost at all. Further, when looking at the no-index original cost analysis, there was actually no cost associated with recipe ingredients lookups, which is why it makes sense that indexing on it would not change anything.

Final Index Design Used: No-Index Configuration (Index Design #1)
We believe that this is the best indexing design since none of the other index designs improved our overall cost, we believe that choosing no index is the best since it avoids the extra memory and time needed to create an unnecessary index that does not change the overall cost of the SQL query. We believe that there are a lot of factors that affect the Cost of the SQL queries, and simply indexing on variables that don't impact the cost outcome would not be very valuable; instead, it would be better to not have any indexing involved as to save computing and memory resources.

Advanced Query #2

```
(
SELECT r.RecipeTitle, r.Ingredients, r.Directions
FROM Recipe r
JOIN Food f ON r.Ingredients LIKE CONCAT('%', f.FoodName, '%')
WHERE f.Category = 'Dairy products'
```

```
)
INTERSECT
(
SELECT r.RecipeTitle, r.Ingredients, r.Directions
FROM Recipe r
JOIN Food f ON r.Ingredients LIKE CONCAT('%', f.FoodName, '%')
WHERE f.Category LIKE '%Seafood'
) LIMIT 15;
```

 Index Design #1: No Index Cost: 27254.71 (using Explain-Analyze command)

```
| -> Limit: 15 row(s) (cost=27254.71..27254.89 rows=15) (actual time=362.698..362.797 rows=15 loops=1)
-> Table scan on <intersect temporary> (cost=27254.71..274254.70 ..27254.70 rows=13459) (actual time=362.696..362.795 rows=15 loops=1)
-> Filter: (r.Ingredients like concat('%',f.FoodName,'%')) (cost=12280.89 rows=13459) (actual time=362.695..362.685 rows=776 loops=1)
-> Table scan on r (cost=0.48 rows=1018) (actual time=0.048..8.731 rows=69874 loops=1)
-> Table scan on r (cost=0.48 rows=1018) (actual time=0.012..1.640 rows=1127 loops=1)
-> Hash
-> Filter: (f.Category = 'Dairy products') (cost=121.25 rows=119) (actual time=0.470..1.013 rows=62 loops=1)
-> Table scan on f (cost=121.25 rows=1190) (actual time=0.463..0.881 rows=1190 loops=1)
-> Table scan on f (cost=131.25 rows=1190) (actual time=0.875..8.108 rows=72128 loops=1)
-> Table scan on f (cost=0.45 rows=1018) (actual time=0.012..1.300 rows=1127 loops=1)
-> Hash
-> Filter: (f.Category like '%Seafood') (cost=121.25 rows=132) (actual time=0.063..0.809 rows=64 loops=1)
-> Table scan on f (cost=121.25 rows=1190) (actual time=0.048..0.436 rows=1190 loops=1)
-> Table scan on f (cost=121.25 rows=1190) (actual time=0.048..0.436 rows=1190 loops=1)
```

Explanation: We first ran our EXPLAIN ANALYZE command when we did not have any index, so that we would have a baseline to compare our possible indexes.

Index Design #2: CREATE INDEX recipe_ing_idx ON Recipe(Ingredients(255)); Cost: 27254.71

```
| -> Limit: 15 row(s) (cost=27254.71..2725.89 rows=15) (actual time=366.820..366.931 rows=15 loops=1)
-> Table scan on <intersect temporary> (cost=27254.71..27425.42 rows=13459) (actual time=366.820..366.929 rows=15 loops=1)
-> Intersect materialize with deduplication (cost=27254.70..27254.70 rows=13459) (actual time=366.807..366.807 rows=776 loops=1)
-> Filter: (r.Ingredients like concat('$', f.FoodName, '$')) (cost=12280.89 rows=13459) (actual time=0.7651..179.110 rows=1537 loops=1)
-> Inner hash join (no condition) (cost=122280.89 rows=13459) (actual time=0.6501..8282 rows=69874 loops=1)
-> Table scan on r (cost=0.48 rows=1018) (actual time=0.010..1.834 rows=1127 loops=1)
-> Filter: (f.Category = 'Dairy products') (cost=121.25 rows=119) (actual time=0.111..0.621 rows=62 loops=1)
-> Table scan on f (cost=121.25 rows=1190) (actual time=0.104..0.495 rows=1190 loops=1)
-> Filter: (r.Ingredients like concat('$', f.FoodName, '$')) (cost=13627.92 rows=14953) (actual time=12.493..175.582 rows=39 loops=1)
-> Table scan on r (cost=0.45 rows=1018) (actual time=0.845..8.117 rows=72128 loops=1)
-> Table scan on f (cost=121.25 rows=1018) (actual time=0.845..8.117 rows=72128 loops=1)
-> Hash
-> Filter: (f.Category like '$Seafood') (cost=121.25 rows=132) (actual time=0.063..0.807 rows=64 loops=1)
-> Table scan on f (cost=121.25 rows=1190) (actual time=0.047..0.439 rows=1190 loops=1)
```

Explanation: We thought that creating an index on the Recipe ingredients would decrease the overall runtime since the ingredients attribute was used in the "JOIN" clause in this SQL query. However, this does not improve the overall cost because we believe that recipe ingredients don't have a natural order that would improve the search time and cost.

Index Design #3: CREATE INDEX food_cat_idx ON Food(Category); Cost: 20692.97

Explanation: We thought of creating an index on the food "Category" key as it is used in the "WHERE" clause of the SQL query. This in fact decreased our overall cost from 27254.71 to 20692.97 because the overall cost to scan the table and hash the food category decreased after adding the index. We expected this to happen, as it would make searching for certain categories in the WHERE clause easier.

Index Design #4: CREATE INDEX recipe_dir_idx ON Recipe(Directions(255));
 Cost: 27254.71

```
| -> Limit: 15 row(s) (cost=27254.71..27254.89 rows=15) (actual time=367.984..368.085 rows=15 loops=1)
| -> Table scan on <intersect temporary> (cost=27254.71..27425.42 rows=13459) (actual time=367.983..368.083 rows=15 loops=1)
| -> Intersect materialize with deduplication (cost=27254.70..27254.70 rows=13459) (actual time=367.970..367.970 rows=776 loops=1)
| -> Filter: (r.Ingredients like concat('%t,f.FoodName,'%t)) (cost=12280.89 rows=13459) (actual time=0.636..8.380 rows=69874 loops=1)
| -> Inner hash join (no condition) (cost=12280.89 rows=13459) (actual time=0.626..8.380 rows=69874 loops=1)
| -> Table scan on r (cost=0.48 rows=1018) (actual time=0.090..1.589 rows=1127 loops=1)
| -> Filter: (f.Category = 'Dairy products') (cost=121.25 rows=119) (actual time=0.066..0.598 rows=62 loops=1)
| -> Filter: (r.Ingredients like concat('%t,f.FoodName,'%t)) (cost=13627.92 rows=14953) (actual time=12.484..179.034 rows=39 loops=1)
| -> Table scan on r (cost=0.45 rows=1018) (actual time=0.815..8.343 rows=72128 loops=1)
| -> Table scan on r (cost=0.45 rows=1018) (actual time=0.011..1.350 rows=1127 loops=1)
| -> Table scan on r (cost=0.45 rows=1018) (actual time=0.011..1.350 rows=1127 loops=1)
| -> Filter: (f.Category like '%Seafood') (cost=121.25 rows=132) (actual time=0.059..0.778 rows=64 loops=1)
| -> Table scan on f (cost=121.25 rows=1190) (actual time=0.044..0.416 rows=1190 loops=1)
```

Explanation: We thought of creating an index on the Recipe "Direction" key as it is used to filter and select the rows in our SQL queries including with the "INTERSECT" command. This did not decrease the cost of our SQL query, and after further inspection, we saw that the recipe directions were not used in an inner hashing join operation or table scanning functions or loops. It made sense then that the overall cost stayed the same as the indexing did not impact any of these operations.

Final Index Design Used: Index Design #3

Out of all 4 configurations, we believe that index design #3 involving an index for the Food Category key is the best configuration, as it decreased our cost from 2725.41 to 20692.97. This makes sense as the cost was originally increasing due to the hash function when searching for a specific category that met a certain condition (as stated in the SQL query). By indexing our category, we could sort the category values and ensure that they were searched in less time and cost than before.

Advanced Query #3

```
SELECT
c.RecipeTitle,
COUNT(c.FavoriteID) AS FavoriteCount
FROM
Contains c
JOIN
Favorites f ON c.FavoriteID = f.FavoriteID
GROUP BY
c.RecipeTitle
ORDER BY
FavoriteCount DESC
LIMIT 15;
```

 Index Design #1: No Index Cost: 45.25

Explanation: We first ran our EXPLAIN ANALYZE command when we did not have any index, so that we would have a baseline to compare our possible indexes.

Index Design #2: CREATE INDEX dateadded_idx on Favorites(DateAdded); Cost: 45.25

```
| -> Limit: 15 row(s) (actual time=1.252.1.254 rows=15 loops=1)
-> Sort: FavoriteCount DESC, limit input to 15 row(s) per chunk (actual time=1.251.1.252 rows=15 loops=1)
-> Table scan on <temporary> (actual time=1.150.1.161 rows=84 loops=1)
-> Aggregate using temporary table (actual time=1.148.1.148 rows=84 loops=1)
-> Nested loop inner join (cost=45.25 rows=100) (actual time=0.707.1.049 rows=100 loops=1)
-> Covering index scan on c using PRIMARY (cost=10.25 rows=100) (actual time=0.673..0.696 rows=100 loops=1)
-> Covering index lookup on f using PRIMARY (FavoriteID=c.FavoriteID) (cost=0.25 rows=1) (actual time=0.003..0.003 rows=1 loops=100)
```

Explanation: It makes sense why we would not see an improvement in cost with this index. This is because when we perform our inner join, we still need to check each value in the Favorites table with the Contains table, so indexing by DateAdded doesn't change the amount of times we do this. Furthermore, we realize that DateAdded only gives us information about the date of Favorites table, so this does not really help us because even by indexing by information like this, our query still has to go through Favorites and count theme as well as order by our favorite count.

Index Design #3: CREATE INDEX food_category_idx on Food(Category);Cost: 45.25

```
| -> Limit: 15 row(s) (actual time=0.573..0.575 rows=15 loops=1)
-> Sort: FavoriteCount DESC, limit input to 15 row(s) per chunk (actual time=0.572..0.573 rows=15 loops=1)
-> Table scan on temporary> (actual time=0.523..0.539 rows=84 loops=1)
-> Aggregate using temporary table (actual time=0.521..0.521 rows=84 loops=1)
-> Nested loop inner join (cost=45.25 rows=100) (actual time=0.073..0.421 rows=100 loops=1)
-> Covering index scan on c using FRIMARY (FavoriteID=c.FavoriteID) (cost=0.25 rows=10) (actual time=0.003..0.003 rows=1 loops=10)
-> Covering index lookup on f using FRIMARY (FavoriteID=c.FavoriteID) (cost=0.25 rows=1) (actual time=0.003..0.003 rows=1 loops=100)
```

Explanation: I thought that adding this index could help us, since although we do not directly grab from the Food table, we do use the Contains table, which contains the key to the Food table. However, it does make sense why this key would not work, since we still have to go through all the values for our inner join, so indexing by a different attribute does not really change or improve our cost in any way. Since Food table is not even used in our query, it shows us that it most likely would not have any impact on our cost.

Index Design #4: CREATE INDEX food_measures_idx on Food(Measures);
 Cost: 45.25

```
|-> Limit: 15 row(s) (actual time=0.472..0.474 rows=15 loops=1)
-> Sort: FavoriteCount DESC, limit input to 15 row(s) per chunk (actual time=0.472..0.473 rows=15 loops=1)
-> Table scan on <temporary (actual time=0.439..0.450 rows=84 loops=1)
-> Aggregate using temporary table (actual time=0.438..0.436 rows=84 loops=1)
-> Nested loop inner join (cost=45.25 rows=100) (actual time=0.074..0.347 rows=100 loops=1)
-> Covering index scan on c using FRIMARY (cost=10.25 rows=100) (actual time=0.052..0.074 rows=100 loops=1)
-> Covering index lookup on f using FRIMARY (FavoriteID=c.FavoriteID) (cost=0.25 rows=1) (actual time=0.002..0.003 rows=1 loops=100)
```

Explanation: As seen above, we realize that when we use an attribute to index that does not appear in our query, it will not help our cost at all. This makes sense, because we would ideally want to index by some relevant attribute, so that we can improve our costs, searches, and queries. Unfortunately, we are not left with many index options from our query, so using an irrelevant attribute does not improve our cost. Since Food table is not even used in our query, it shows us that it most likely would not have any impact on our cost.

Final Index Design Used:

Overall, it is realized that we want to select a good index value that is not a primary key and that appears in our table. However, in our original query, we only use two tables, and the attributes from these tables are already primary keys, so we cannot use them in our index. This leads us to use other attributes from the JOIN tables, however we also see that these do not improve our cost since they are not even part of the original query. Overall, it is deemed that finding an index that improves this query is very difficult, since this query utilizes primary keys only. If we wanted to improve this cost from our query, we may want to try utilizing other attributes of the table that are not primary keys, so we can use them as an index value. Because of this, we ultimately find that using no index gives us the same cost as trying to create an index.

Advanced Query #4

```
SELECT
User.UserName,
COUNT(MyRecipes.RecipeTitle) AS RecipesUploaded
FROM
User
RIGHT JOIN
MyRecipes ON User.UserID = MyRecipes.UserID
GROUP BY
MyRecipes.UserID
ORDER BY
RecipesUploaded DESC;
```

 Index Design #1: No Index Cost: 23.25

Explanation: We first ran our EXPLAIN ANALYZE command when we did not have any index, so that we would have a baseline to compare our possible indexes.

Index Design #2: CREATE INDEX user_name_idx on User(UserName); Cost: 23.25

Explanation: When we tried indexing on UserName, we found that we did not get a cost improvement. Although this is an attribute in our query, it makes sense why we would not see any improvement, because this is just a select attribute, and indexing it does not really change our actual results, since we still have joins and groupbys on other attributes. We end up RIGHT JOINING our MyRecipes table on users, but we compare with userid, so we still have to make this comparison regardless of whether we index on an attribute in the Users table or not, therefore it makes sense that we do not see a cost improvement.

Index Design #3: CREATE INDEX password_idx on User(Password); Cost: 23.25

Explanation: We now try indexing on a different attribute of the User table, since we have limited options with this query. We assume we would get similar results to the UserName table because we are again wanting to go through our Users, however indexing by UserName or Password does not help optimize our query at all. We end up RIGHT JOINING our MyRecipes table on users, but we compare with userid, so we still have to make this comparison regardless of whether we index on an attribute in the Users table or not, therefore it makes sense that we do not see a cost improvement.

Index Design #4: CREATE INDEX ingredients_idx on Recipe(Ingredients(255));
 Cost: 23.25

Explanation: We again do not see a cost improvement here. On our RightJOIN, we still rely on the Userlds, and therefore still end up going through all of them, so even though indexing on Ingredients could seem helpful, it does not actually allow us any improvement because we still rely on Userlds to go through, and we cannot index on Userlds because that is a primary key.

Final Index Design Used:

Overall, we find that not much can be done to improve our costs for this query. One downside of the query is that we utilize a lot of primary keys, and we do not have many tables included in this query, so it is hard to find an index that is part of a table that we use in the query, but also is not a primary key. One way we could improve this, is by joining in more tables and expanding our query a little bit, so that we could introduce some different attributes to index on. Otherwise, as we can see above, all the indexes that we tried did not improve our cost, so this query performs the same with or without an index. This would make sense, because we would probably want to rewrite our query in such a way that we could maximize from an index by using joins. However, in our query, it is quite hard to have an index that is actually beneficial since we require a count and right join that are hard to optimize.