RecipeDelight Database Project

ITCS 3160 - 001

Fall 2019

Korey Heiser

Chris Myrick

December 04, 2019

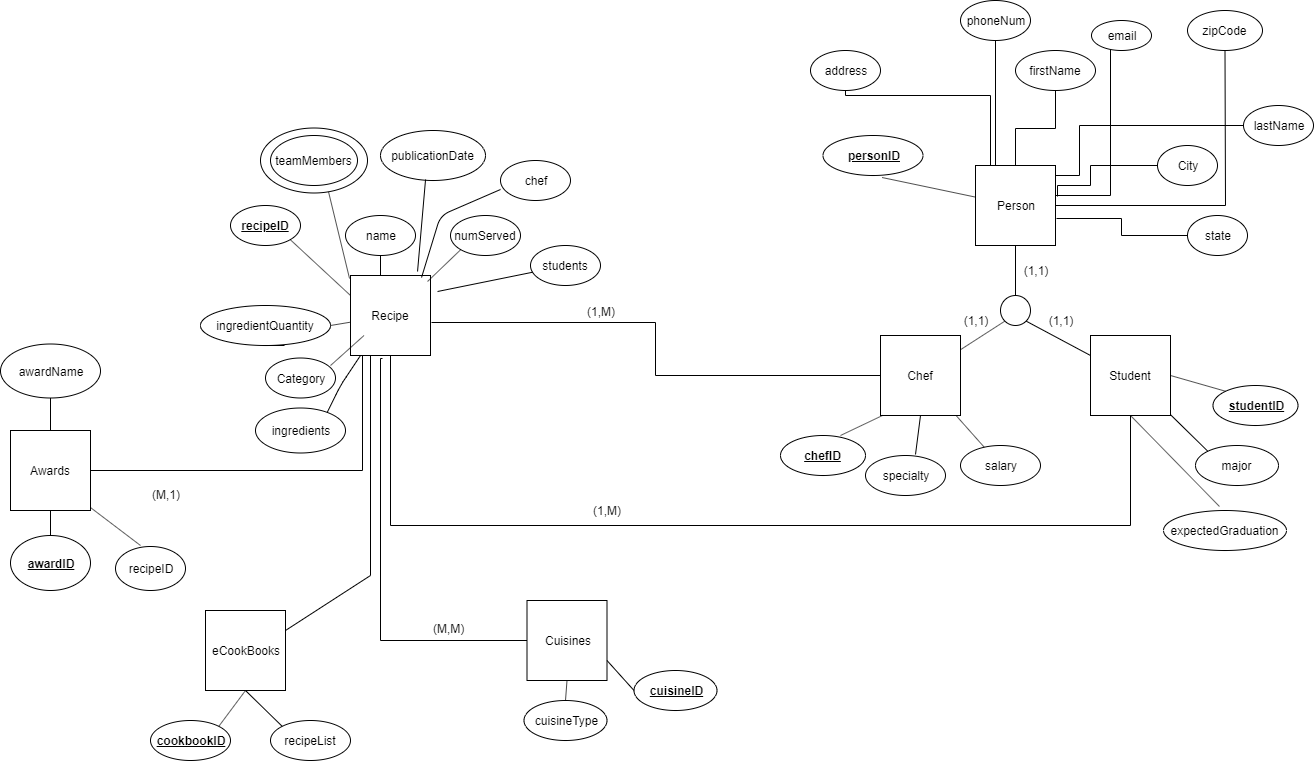
**Executive Summary**

RecipeDelight is a database designed for the creation of recipes. The database tracks recipes, along with a selection of students and chefs that take part in them. The following data is collected and stored within this database:

* Recipes
* Students
* Chefs
* Awards
* Cookbooks
* Cuisine Types

Designs of our database created during the planning stages were created using draw.io. This includes the creation of our EERD model and relational database model. MySQL is the DMS used to implement this database.

**EERD**



**Business Rules and Assumptions**

1. Person – Chef/Student is a generalization-specialization relationship (superclass/subclass)

2. A chef may be simultaneously assigned to multiple recipe projects. A project has a chef designed is designated the Lead Chef. Egos being what they are, there may be multiple Lead Chefs

3. A student may be simultaneously assigned to multiple recipe projects.

4. A recipe is designated as available for download or for inclusion in an eCookBook when it is published as indicated by its publication date.

5. A chef or a student may be associated with multiple recipe projects.

6. A chef may be the only person assigned to a recipe project.

7. A student may not be the only person assigned to a recipe project.

8. Many recipes contain some of the same ingredient(s), and it is advisable for standardization that ensures references to an ingredient name are the same across all recipes.

9. An ingredient may be listed only once in a given recipe.

10. You may choose to express ingredient quantities in metric or English units.

11. Quantities should be expressed precisely. For instance, ½ stored as a float field might be rendered as 0.4999999 instead of 0.5. Therefore, quantity fields should probably not be the float data type.

12. Quantities should be usable in arithmetic computations. For instance, if the Number Served field of a recipe needs to be doubled, you may assume that all ingredient Quantities may be multiplied by two.

13. Awards are associated with recipes.

14. A recipe can win an award only once. Awards may differentiated by time. A recipe could win both the Best Entre 2019 and Best Entre 2020 awards but could not win the Best Entre 2019 more than once.

15. A recipe can be included in a given eCookBook only once.

16. A recipe can be included in multiple eCookBooks.

17. Chef reports:

a. Listing unpublished recipe projects by Lead Chief.

b. Listing published recipe projects by Lead Chief.

c. Listing all recipe projects and include all assigned chefs.

18. Student reports

a. Listing all students and their published recipe projects with assigned chefs.

19. Management report

a. Listing all chefs including names and salaries.

b. Listing of all students with names, a count of assigned, unpublished recipe projects and expected graduation date.

20. Recipes and eCookBooks

a. List each recipe with Recipe Name, Lead Chef, and number of times it has been downloaded.

b. List each Recipe Name with its Ingredients. Include the Quantity and Units.

c. List each Recipe Name with its Instructions.

d. List each eCookBook with eCookBook Name and number of recipes in it.

21. Awards

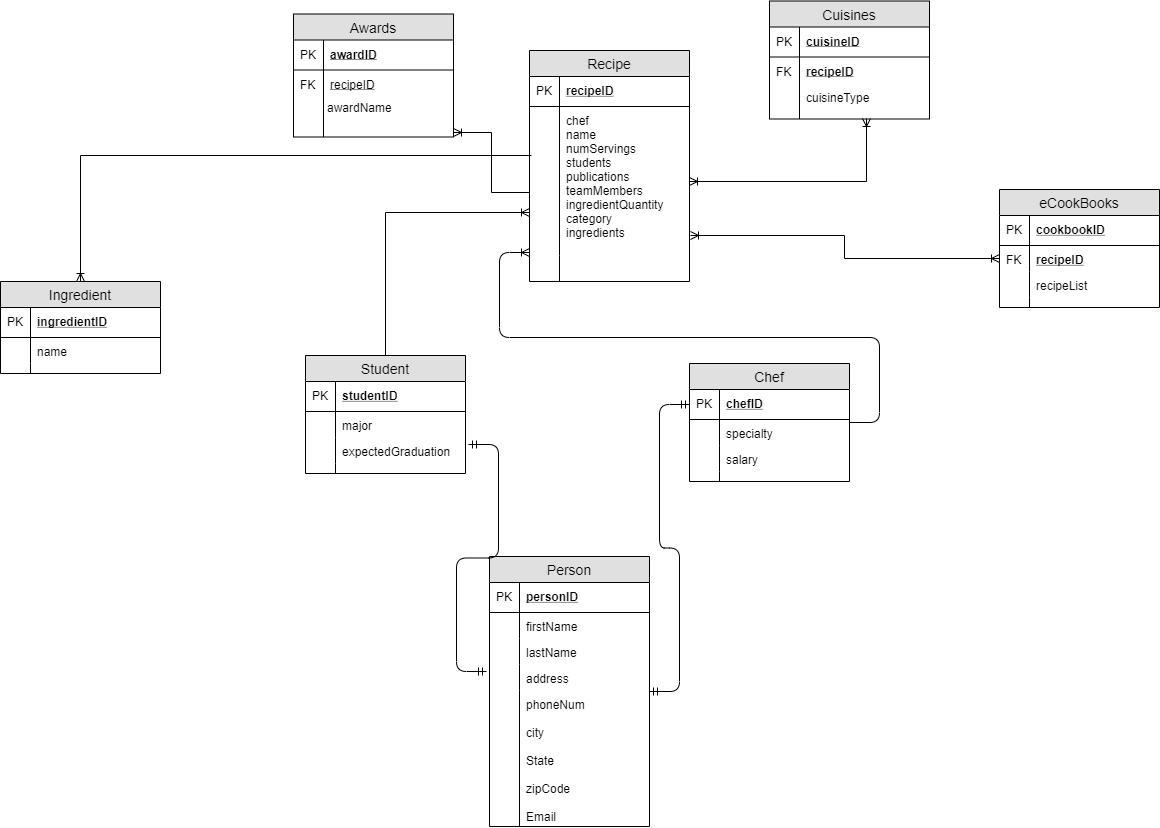
a. List all recipes that have received awards including the name of the award.

**Member Assignments:**

Korey: Relational Model Diagram, Table Creation

Chris: Sample Test Data, SQL listings

**Relational Model**



**Sample Test Data**

* **Recipe**
  + recipeID - int
  + Chef - string
  + Name - sting
  + numServings - int
  + Students - string
  + Publications - string
  + teamMembers - string
  + ingredientQuantity - int
  + Category - string
  + Ingredients - string
* **Ingredients**
  + recipeID
  + ingredientID
* **Cuisines**
  + cuisineID - int
  + cuisineType - string
* **Awards**
  + awardID - string
* **eCookBooks**
  + cookbookID - int
  + recipeList - int
* **Person**
  + personID - int
  + firstName - string
  + lastName - string
  + Address - string
  + phoneNum - int
  + City - string
  + state - string
  + zipCode - int
  + email - string
* **Chef**
  + chefID - int
  + specialty - string
  + Salary - int
* **Student**
  + studentID - int
  + major - string
  + expectedGraduation - int

**Recipe**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **recipeID** | **chef** | **name** | **numServings** | **students** | **publications** | **teamMembers** | **category** | **ingredients** |
| 1 | Mike | Grilled Chicken | 6 | SillySauls | crapfood.com | FireChix | BBQ | 1 |

**Recipe Ingredients**

|  |  |
| --- | --- |
| IngredientID | name |
| 1 | onion |
| 2 | potatoe |
| 3 | kale |

**Recipe Awards**

|  |  |
| --- | --- |
| **awardID** | **recipeID** |
| bcr18 | 1 |
| bqr19 | 1 |

**Cuisines**

|  |  |  |
| --- | --- | --- |
| **cuisineID** | cuisineType | recipeID |
| 1 | american | 1 |
| 2 | chinese | 3 |

**eCookBooks**

|  |  |  |
| --- | --- | --- |
| **cookbookID** | recipeListID | recipeList |
| 14569 | 11 | Chicken List |
| 34521 | 12 | Steak List |

**Person**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **personID** | firstName | lastName | address | phoneNum | city | state | zipcode | email |
| 800917 | Jerry | Crow | 123 apple rd | 704-765-9801 | Charlotte | NC | 28223 | jerrycrow@gmail.com |
| 800918 | Mary | Crow | 123 apple rd | 704-756-3761 | Charlotte | NC | 28223 | marycrow@gmail.com |
| 800919 | Jerry Jr. | Crow | 123 apple rd | 704-794-8164 | Charlotte | NC | 28223 | jerrycrowjr@gmail.com |

**Chef**

|  |  |  |
| --- | --- | --- |
| **chefID** | specialty | salary |
| 167 | fish | 61,789 |
| 191 | steak | 64,264 |

**Students**

|  |  |  |
| --- | --- | --- |
| **studentID** | major | expectedGraduation |
| 955 | Culinary management | Spring 2020 |
| 919 | Food Science | Spring 2020 |

**SQL CODE**

DROP DATABASE IF EXISTS RecipeDelight;

CREATE DATABASE RecipeDelight;

USE RecipeDelight;

DROP TABLE IF EXISTS person;

CREATE TABLE `recipedelight`.`person` (

`personID` INT NOT NULL,

`firstName` VARCHAR(20) NOT NULL,

`lastName` VARCHAR(20) NOT NULL,

`address` VARCHAR(45) NOT NULL,

`phoneNumber` VARCHAR(45) NULL,

`city` VARCHAR(45) NOT NULL,

`state` VARCHAR(5) NOT NULL,

`zipCode` VARCHAR(45) NOT NULL,

`email` VARCHAR(45) NULL,

PRIMARY KEY (`personID`));

DROP TABLE IF EXISTS student;

CREATE TABLE `recipedelight`.`student` (

`studentID` INT NOT NULL,

`major` VARCHAR(45) NOT NULL,

`expectedGraduation` DATE NULL,

PRIMARY KEY (`studentID`));

DROP TABLE IF EXISTS chef;

CREATE TABLE `recipedelight`.`chef` (

`chefID` INT NOT NULL,

`specialty` VARCHAR(20) NULL,

`salary` INT NULL,

PRIMARY KEY (`chefID`));

DROP TABLE IF EXISTS recipe;

CREATE TABLE `recipedelight`.`recipe` (

`recipeID` INT NOT NULL,

`chef` VARCHAR(20) NULL,

`name` VARCHAR(20) NULL,

`numServings` INT NULL,

`students` VARCHAR(20) NULL,

`publications` VARCHAR(45) NULL,

`teamMembers` VARCHAR(45) NULL,

`category` VARCHAR(45) NULL,

`ingredients` VARCHAR(45) NULL,

`numIngredients` INT NULL,

PRIMARY KEY (`recipeID`));

DROP TABLE IF EXISTS ingredient;

CREATE TABLE `recipedelight`.`ingredient` (

`ingredientID` INT NOT NULL,

`name` VARCHAR(10) NOT NULL,

PRIMARY KEY (`ingredientID`));

DROP TABLE IF EXISTS cuisines;

CREATE TABLE `recipedelight`.`cuisines` (

`cuisineID` INT NOT NULL,

`recipeID` INT NOT NULL,

`cuisineType` VARCHAR(10) NOT NULL,

PRIMARY KEY (`cuisineID`));

DROP TABLE IF EXISTS eCookBooks;

CREATE TABLE `recipedelight`.`eCookBooks` (

`cookbookID` INT NOT NULL,

`recipeID` INT NOT NULL,

`recipeList` VARCHAR(45) NOT NULL,

PRIMARY KEY (`cookbookID`));

**Insert Statements**

INSERT INTO person (personID, firstName, lastName, address, phoneNumber, city, state, zipCode, email)

VALUES(800917, 'Jerry', 'Crow', '123 apple rd', '704-765-9801', 'Charlotte', 'NC', '28223', 'jerrycrow@gmail.com');

INSERT INTO person (personID, firstName, lastName, address, phoneNumber, city, state, zipCode, email)

VALUES(800918, 'Mary', 'Crow', '123 apple rd', '704-756-3761', 'Charlotte', 'NC', '28223', 'marycrow@gmail.com');

INSERT INTO person (personID, firstName, lastName, address, phoneNumber, city, state, zipCode, email)

VALUES(800919, 'Jerry Jr.', 'Crow', '123 apple rd', '704-794-8164', 'Charlotte', 'NC', '28223', 'jerrycrowjr@gmail.com');

INSERT INTO student (studentID, major, expectedGraduation)

VALUES(955, 'Culinary Management', '2020-04-20');

INSERT INTO student (studentID, major, expectedGraduation)

VALUES(919, 'Food Science', '2020-04-20');

INSERT INTO chef (chefID, specialty, salary)

VALUES(167, 'fish', 61789);

INSERT INTO chef (chefID, specialty, salary)

VALUES(191, 'steak', 64264);

INSERT INTO recipe (recipeID, chef, name, numServings, students, publications, teamMembers, category, ingredients, numIngredients)

VALUES(1, 'Mike', 'Grilled Chicken', 6, 'SillySauls', 'crapfood.com', 'FireChix', 'BBQ', 'Chicken', 1);

INSERT INTO ingredient (ingredientID, name)

VALUES(946, 'pork butt');

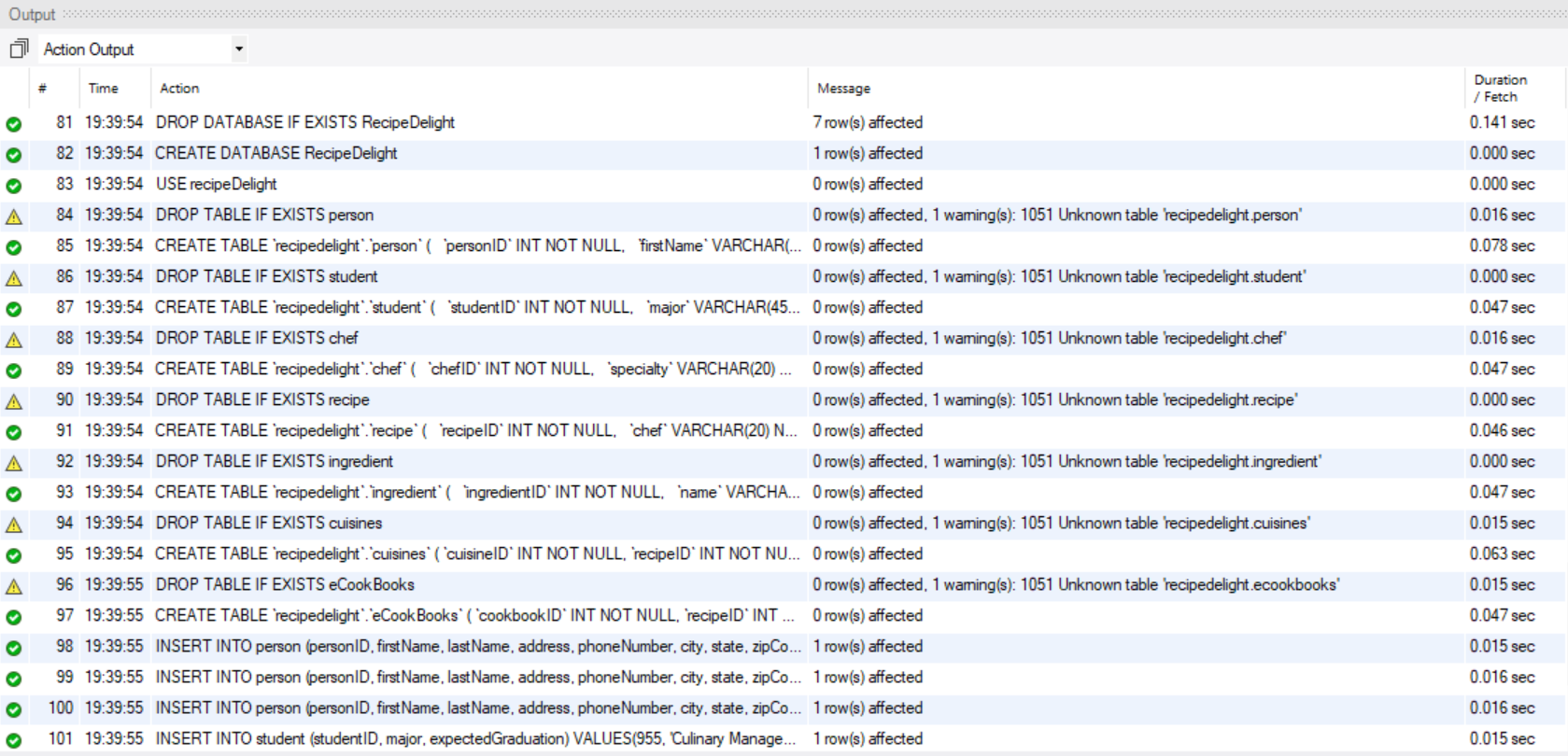
INSERT INTO cuisines (cuisineID, recipeID, cuisineType)

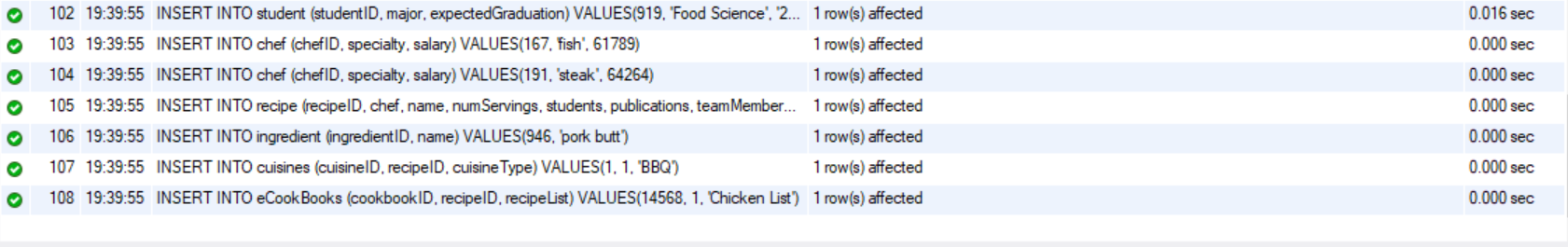
VALUES(1, 1, 'BBQ');

INSERT INTO eCookBooks (cookbookID, recipeID, recipeList)

VALUES(14568, 1, 'Chicken List');

**SQL RUNNING**





**Data Dictionary**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Table Name** | **Field Name** | **Data Type** | **Field length** | **Constraint** | **Description** |
| **person** | personID | INT |  | Primary Key / NOT NULL | Unique person identifier |
|  | firstName | VARCHAR | 20 | NOT NULL | First name of person |
|  | lastName | VARCHAR | 20 | NOT NULL | Last name of person |
|  | address | VARCHAR | 45 | NOT NULL | Address of Person |
|  | phoneNumber | VARCHAR | 45 | NULL | Phone Number of Person |
|  | city | VARCHAR | 45 | NOT NULL | City of Person |
|  | state | VARCHAR | 5 | NOT NULL | State of Person |
|  | zipCode | VARCHAR | 45 | NOT NULL | Zip Code of Person |
|  | email | VARCHAR | 45 | NULL | Email of Person |
| **student** | studentID | INT |  | Primary Key / NOT NULL | Unique student identifier |
|  | major | VARCHAR | 45 | NOT NULL | Major of Student |
|  | expectedGraduation | DATE |  | NULL | Expected Graduation Date of Student |
| **chef** | chefID | INT |  | Primary Key / NOT NULL | Unique chef identifier |
|  | specialty | VARCHAR | 20 | NULL | Specialty of Chef |
|  | salary | INT |  | NULL | Salary of Chef |
| **recipe** | recipeID | INT |  | Primary Key / NOT NULL | Unique recipe identifier |
|  | chef | VARCHAR | 20 | NULL | Chef of the Recipe |
|  | name | VARCHAR | 20 | NULL | Name of Recipe |
|  | numServings | INT |  | NULL | Number of servings for Recipe |
|  | students | VARCHAR | 20 | NULL | Student Contributions to Recipe |
|  | publications | VARCHAR | 45 | NULL | The cookbook publications where this recipe can be found |
|  | teamMembers | VARCHAR | 45 | NULL | People who work on the recipe |
|  | category | VARCHAR | 45 | NULL | The type of recipe (BBQ, italian, etc) |
|  | ingredients | VARCHAR | 45 | NULL | What food items the recipe needs |
|  | numIngredients | INT |  | NULL | Total amount of ingredients needed |
| **ingredient** | ingredientID | INT |  | Primary Key / NOT NULL | Unique ingredient identifier |
|  | name | VARCHAR | 10 | NOT NULL | The name of the ingredient |
| **cuisines** | cuisineID | INT |  | NOT NULL | Unique cuisine identifier |
|  | recipeID | INT |  | NOT NULL | The recipeID the cuisine references |
|  | cuisineType | VARCHAR | 10 | NOT NULL | The type of cuisine this entry is |
| **eCookBooks** | cookbookID | INT |  | NOT NULL | Unique cookbook identifier |
|  | recipeID | INT |  | NOT NULL | The unique recipeIDs contained in the cookbook |
|  | recipeList | VARCHAR | 45 | NOT NULL | The information of the recipes contained in the cookbook |

**Source for Sample Output**

SELECT \* FROM person;

SELECT \* FROM student;

SELECT \* FROM chef;

SELECT \* FROM recipe;

SELECT \* FROM ingredient;

SELECT \* FROM cuisines;

SELECT \* FROM eCookBooks;

**Advanced SQL Functions**

* Generalization - Specialization Relationship
  + Person (generalized) can be either student or chef (specialized)
* Stored Procedure
  + Procedure person\_Count is a procedure that counts the number of entities within the person table
  + DROP PROCEDURE IF EXISTS person\_Count
  + DELIMITER //
  + CREATE
  + DEFINER = CURRENT\_USER
  + PROCEDURE person\_Count (OUT countParm INT)
  + BEGIN
  + SELECT COUNT(\*) INTO countParm FROM person;
  + END//
  + DELIMITER;
  + SHOW PROCEDURE STATUS WHERE db = ‘RecipeDelight’;
  + CALL person\_Count(@parm1);
  + SELECT @parm1;
* Trigger
  + This trigger is responsible for detecting when a user is trying to insert a salary with a value below 0 within the chef table. It catches the mistake and automatically sets the new value to 0.
  + DROP TRIGGER IF EXISTS chef\_INSERTSalaryRange;
  + DELIMITER;
  + CREATE
  + DEFINER = CURRENT\_USER
  + TRIGGER chef\_INSERTSalaryRange
  + BEFORE INSERT
  + ON chef FOR EACH ROW
  + BEGIN
  + IF NEW.salary < 0 THEN
  + SET NEW.salary = 0;
  + END IF;
  + END;
  + DELIMITER;
* Delete / Update
  + These functions delete/update entities within tables where entity fields match.
  + DELETE FROM person
  + WHERE personID = 800917;
  + DELETE FROM student
  + WHERE studentID = 955;
  + UPDATE chef
  + SET salary = 62485
  + WHERE chefID = 167;
  + UPDATE person
  + SET address = ‘321 elppa rd’
  + WHERE personID = 800918;
* Index
  + CREATE INDEX recipID ON recipe(name);
  + This creates an index on the recipe table under the name column. Considering this is a database to hold recipe information, this would naturally be the most commonly searched table. Thus, we can optimize the search time by using an index at this location.