# Exercises: SQLAlchemy

This document defines the problems for the in-class lab for the [Python ORM course @ Software University](https://softuni.bg/trainings/4253/python-orm-october-2023).

## Model Recipe

Your task is to define a Python class, **Recipe**, which inherits from **SQLAlchemy's** **Base** class and represents the **"recipes"** table. The table should have the following columns:

* **id** (Primary Key):
  + Data Type: **Integer**
  + A **unique** identifier for each recipe.
* **name**:
  + Data Type: **String**
  + The **name** of the recipe. This field is **required** and must **not** be left **empty**.
* **ingredients**:
  + Data Type: **Text**
  + A field to store the list of **ingredients** for the recipe. This field is **required** and must not be left empty.
* **instructions**:
  + Data Type: **Text**
  + A field to store the cooking **instructions** for the recipe. This field is **required** and must **not** be left **empty**.

## Create Recipe

Function: "**create\_recipe(name: str, ingredients: str, instructions: str)**" **creates** a new **recipe** instance with the given **arguments**, **adds** it to the **session**, **commits** it, and **returns** it as a result.

### Examples

|  |  |  |
| --- | --- | --- |
| **name** | **ingredients** | **instructions** |
| Spaghetti Carbonara | Pasta, Eggs, Pancetta, Cheese | Cook the pasta, mix it with eggs, pancetta, and cheese |
| Chicken Stir-Fry | Chicken, Bell Peppers, Soy Sauce, Vegetables | Stir-fry chicken and vegetables with soy sauce |
| Caesar Salad | Romaine Lettuce, Croutons, Caesar Dressing | Toss lettuce with dressing and top with croutons |

|  |
| --- |
| **Test Code** |
| # Query all recipes recipes = session.query(Recipe).all()  # Loop through each recipe and print its details for recipe in recipes:  print(f"Recipe name: {recipe.name}") |
| **Output** |
| Recipe name: Spaghetti Carbonara  Recipe name: Chicken Stir-Fry  Recipe name: Caesar Salad |

## Update Recipe

Function: "**update\_recipe\_by\_name(name: str, new\_name: str, new\_ingredients: str, new\_instructions: str)**" **updates** the **recipe** by the given **name** with the new **arguments**, and **commits** the **session**.

### Examples

|  |
| --- |
| **Test Code** |
| # Update a recipe by name update\_recipe\_by\_name(  name="Spaghetti Carbonara",  new\_name="Carbonara Pasta",  new\_ingredients="Pasta, Eggs, Guanciale, Cheese",  new\_instructions="Cook the pasta, mix with eggs, guanciale, and cheese" )  # Query the updated recipe  updated\_recipe = session.query(Recipe).filter\_by(name="Carbonara Pasta").first()  # Print the updated recipe details print("Updated Recipe Details:") print(f"Name: {updated\_recipe.name}") print(f"Ingredients: {updated\_recipe.ingredients}") print(f"Instructions: {updated\_recipe.instructions}") |
| **Output** |
| Name: Carbonara Pasta  Ingredients: Pasta, Eggs, Guanciale, Cheese  Instructions: Cook the pasta, mix with eggs, guanciale, and cheese |

## Delete Recipe

Function: "**delete\_recipe\_by\_name(name: str)**" **deletes** the recipe object with the given **name** and **commits** the **session**.

### Examples

|  |
| --- |
| **Test Code** |
| # Delete a recipe by name delete\_recipe\_by\_name("Carbonara Pasta")  # Query all recipes recipes = session.query(Recipe).all()  # Loop through each recipe and print its details for recipe in recipes:  print(f"Recipe name: {recipe.name}") |
| **Output** |
| Recipe name: Chicken Stir-Fry  Recipe name: Caesar Salad |

## Filter Recipes

Function: "**get\_recipes\_by\_ingredient(ingredient\_name: str)**" **returns** all **recipes** that contain the specified **ingredient** **name**.

We'll be working with fresh **instances**, and it would be beneficial to start with a **clean** **database**.

### Examples

|  |
| --- |
| **Test Code** |
| # Delete all objects (recipes) from the database session.query(Recipe).delete() session.commit()  # Create three Recipe instances with two of them sharing the same ingredient recipe1 = create\_recipe(  'Spaghetti Bolognese',  'Ground beef, tomatoes, pasta',  'Cook beef, add tomatoes, serve over pasta' )  recipe2 = create\_recipe(  'Chicken Alfredo',  'Chicken, fettuccine, Alfredo sauce',  'Cook chicken, boil pasta, mix with sauce' )  recipe3 = create\_recipe(  'Chicken Noodle Soup',  'Chicken, noodles, carrots',  'Boil chicken, add noodles, carrots' )  # Run the function and print the results ingredient\_to\_filter = 'Chicken' filtered\_recipes = get\_recipes\_by\_ingredient('Chicken')  print(f"Recipes containing {ingredient\_to\_filter}:") for recipe in filtered\_recipes:  print(f"Recipe name - {recipe.name}") |
| **Output** |
| Recipes containing Chicken:  Recipe name - Chicken Alfredo  Recipe name - Chicken Noodle Soup |

## Recipe Ingredients Swap Transaction

Function: "**swap\_recipe\_ingredients\_by\_name(first\_recipe\_name: str, second\_recipe\_name: str)**" **swaps** the **ingredients** between the given **recipes** by their **names**. You must do this through **transactions**.

* First, start the **transaction**.
* **Try** to get the **recipes** (**objects**) by their **name** from the **database.** Then **replace** the **ingredients** between both **recipes**. Do not forget to **commit** the **transaction**.
* **Except** foran **Exception** if any **error** occurs during the process. If that's the case, initiate a **transaction** **rollback** and **raise** an **Exception** with the **error** message (**not required**).
* **Finally,** close the **session**.

### Examples

|  |
| --- |
| **Test Code** |
| # Delete all objects (recipes) from the database session.query(Recipe).delete() session.commit()  # Create the first recipe create\_recipe("Pancakes", "Flour, Eggs, Milk", "Mix and cook on a griddle")  # Create the second recipe  create\_recipe("Waffles", "Flour, Eggs, Milk, Baking Powder", "Mix and cook in a waffle iron")  # Now, swap their ingredients swap\_recipe\_ingredients\_by\_name("Pancakes", "Waffles")  recipe1 = session.query(Recipe).filter\_by(name="Pancakes").first() recipe2 = session.query(Recipe).filter\_by(name="Waffles").first() print(f"Pancakes ingredients {recipe1.ingredients}") print(f"Waffles ingredients {recipe2.ingredients}") |
| **Output** |
| Pancakes ingredients: Flour, Eggs, Milk, Baking Powder  Waffles ingredients: Flour, Eggs, Milk |

### Hint

To begin with, let us explain what a **transaction** is in **SQLAlchemy**. In the context of **databases** and **database** management systems, a **transaction** is a fundamental concept that ensures the **integrity**, **consistency**, and **reliability** of data. A **transaction** is a sequence of one or more database **operations** that are treated as a single, indivisible unit of work. These operations typically include data **retrieval**, **insertion**, **updating**, and **deletion**.

Картина, която съдържа текст, екранна снимка, Шрифт, номер

Описанието е генерирано автоматично

## Model Chef

Your task is to define a Python class, **Chef**, which inherits from **SQLAlchemy's** **Base** class and represents the "**chefs**" table. The table should have the following columns:

* **id** (Primary Key):
  + Data Type: **Integer**
  + A **unique** identifier for each chef.
* **name**:
  + Data Type: **String**
  + The **name** of the chef. This field is **required** and must not be left empty.
* **Recipes** (Relationship to **Recipe** model):
  + Data Type: **Relationship**
  + Each **chef** can have multiple **recipes**. This is represented by a many-to-one relationship with the **Recipe** model. The **recipes** attribute allows you to access all the recipes created by a specific chef.
  + The **back\_populates** attribute ensures **bidirectional** navigation. Set its value to **"chef"**.

## Extend the Recipe Model

You are going to add the following fields to the **Recipe** model:

* **chef\_id** (Foreign Key):
  + Data Type: **Integer**
  + Description: A foreign key that establishes a relationship with the "**Chef**" model. It associates each recipe with a **chef**. This field is **required** and must not be left empty.
* **chef** (Relationship to **Chef** model):
  + Data Type: **Relationship**
  + Description: A **relationship** field **connecting** each recipe to a **chef**. This establishes the **association** that **one** chef can have **multiple** recipes and one **recipe** is owned by only **one** **chef**.
  + The **back\_populates** attribute ensures **bidirectional** navigation. Set its value to **"recipes"**.

## Recipe and Chef Relations

Function: "**relate\_recipe\_with\_chef\_by\_name(recipe\_name: str, chef\_name: str)**" **creates** a **relation** between the **recipe** and the **chef**, given by their **names**.

* If there is a **relationship** between the **recipe** and some **chef**, **raise** an **Exception** with the message: **"Recipe: {recipe\_name} already has a related chef"**.
* Otherwise, **link** the **chef** to the **recipe**, **commit** the **session**, and **return** a result: **"Related recipe {recipe\_name} with chef {chef\_name}"**

### Examples

|  |
| --- |
| **Test Code** |
| # Create a recipe instance for Bulgarian Musaka musaka\_recipe = Recipe(  name="Musaka",  ingredients="Potatoes, Ground Meat, Onions, Eggs, Milk, Cheese, Spices",  instructions="Layer potatoes and meat mixture, pour egg and milk mixture on top, bake until golden brown." )  # Create a Bulgarian chef instances bulgarian\_chef1 = Chef(name="Ivan Zvezdev") bulgarian\_chef2 = Chef(name="Uti Buchvarov")  # Add the recipe instance to the session session.add(musaka\_recipe)  # Add the chef instances to the session session.add(bulgarian\_chef1) session.add(bulgarian\_chef2)  # Commit the changes to the database session.commit() |
| **Test Code** |
| print(relate\_recipe\_with\_chef\_by\_name("Musaka", "Ivan Zvezdev")) |
| **Output** |
| Related recipe Musaka with chef Ivan Zvezdev |
| **Test Code** |
| print(relate\_recipe\_with\_chef\_by\_name("Musaka", "Chef Uti")) |
| **Output** |
| Traceback (most recent call last):  File "C:\PyCharmProjects\Django ORM\10-SQLAlchemy\main.py", line 117, in <module>  print(relate\_recipe\_with\_chef\_by\_name("Musaka", "Chef Uti"))  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^  File "C:\PyCharmProjects\Django ORM\10-SQLAlchemy\main.py", line 81, in relate\_recipe\_with\_chef\_by\_name  raise Exception(f'Recipe: {recipe\_name} already has a related chef')  Exception: Recipe: Musaka already has a related chef |

## 10. Chef and Recipe Integration

Function: "**get\_recipes\_with\_chef()**" **returns** all **recipe names** with their **related** **chef name** in the following format:

* **"Recipe: {recipe\_name\_1} made by chef: {chef\_name\_1}**

**…**

**Recipe: {recipe\_name\_N} made by chef: {chef\_name\_N}"**

### Examples

|  |
| --- |
| **Test Code** |
| # Delete all objects (recipes and chefs) from the database session.query(Recipe).delete() session.query(Chef).delete() session.commit()  # Create chef instances chef1 = Chef(name="Gordon Ramsay") chef2 = Chef(name="Julia Child") chef3 = Chef(name="Jamie Oliver") chef4 = Chef(name="Nigella Lawson")  # Create recipe instances associated with chefs recipe1 = Recipe(name="Beef Wellington", ingredients="Beef fillet, Puff pastry, Mushrooms, Foie gras", instructions="Prepare the fillet and encase it in puff pastry.") recipe1.chef = chef1  recipe2 = Recipe(name="Boeuf Bourguignon", ingredients="Beef, Red wine, Onions, Carrots", instructions="Slow-cook the beef with red wine and vegetables.") recipe2.chef = chef2  recipe3 = Recipe(name="Spaghetti Carbonara", ingredients="Spaghetti, Eggs, Pancetta, Cheese", instructions="Cook pasta, mix ingredients.") recipe3.chef = chef3  recipe4 = Recipe(name="Chocolate Cake", ingredients="Chocolate, Flour, Sugar, Eggs", instructions="Bake a delicious chocolate cake.") recipe4.chef = chef4  recipe5 = Recipe(name="Chicken Tikka Masala", ingredients="Chicken, Yogurt, Tomatoes, Spices", instructions="Marinate chicken and cook in a creamy tomato sauce.") recipe5.chef = chef3  session.add\_all([chef1, chef2, chef3, chef4, recipe1, recipe2, recipe3, recipe4, recipe5]) session.commit() print(get\_recipes\_with\_chef()) |
| **Test Code** |
| Recipe: Beef Wellington made by chef: Gordon Ramsay  Recipe: Boeuf Bourguignon made by chef: Julia Child  Recipe: Spaghetti Carbonara made by chef: Jamie Oliver  Recipe: Chicken Tikka Masala made by chef: Jamie Oliver  Recipe: Chocolate Cake made by chef: Nigella Lawson |