import os

import pandas as pd

import db\_base as db

import csv

class Recipe:

def \_\_init\_\_(self, id, name, ingredients, category):

self.id = id

self.name = name

self.ingredients = ingredients

self.category = category

class Category:

def \_\_init\_\_(self, category\_id, category\_name, predefined):

self.category\_id = category\_id

self.category\_name = category\_name

self.predefined = predefined

class RecipeManager(db.DBbase):

def \_\_init\_\_(self, db\_name):

super().\_\_init\_\_(db\_name)

self.recipes = []

self.category\_options = []

def check\_DB(self):

table\_name = 'Recipes'

table\_exists = super().get\_cursor.execute(

"""SELECT count(\*) FROM sqlite\_master WHERE type='table' AND name = ?;""", (table\_name,)).fetchone()

if table\_exists[0] == 0:

self.reset\_or\_create\_db()

self.load\_data\_from\_csv\_to\_DB()

self.create\_initial\_categories()

else:

print("Table already exists! No data loading from CSV to DB.")

def reset\_or\_create\_db(self):

try:

super().execute\_script("""

DROP TABLE IF EXISTS Recipes;

CREATE TABLE IF NOT EXISTS Recipes (

id INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE,

name varchar(50) NOT NULL,

ingredients TEXT NOT NULL,

category TEXT NOT NULL

);

DROP TABLE IF EXISTS Categories;

CREATE TABLE IF NOT EXISTS Categories (

category\_id INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE,

category\_name varchar(50) NOT NULL,

predefined TEXT

);""")

except Exception as e:

print("An error occurred while resetting the database: ", e)

def create\_initial\_categories(self):

initial\_categories = [

Category(1, 'Dessert', 'Y'),

Category(2, 'Main Course', 'Y'),

Category(3, 'Breakfast', 'Y'),

Category(4, 'Snack', 'Y')

]

# Check if the categories.csv file is empty

categories\_csv\_empty = os.stat('categories.csv').st\_size == 0

for category in initial\_categories:

if categories\_csv\_empty:

save\_categories\_to\_csv(category)

self.add\_category(category)

def load\_data\_from\_csv\_to\_DB(self):

try:

filename = 'recipes.csv'

with open(filename, "r") as csvfile:

csv\_reader = csv.DictReader(csvfile)

if filename == 'recipes.csv':

for row in csv\_reader:

id = row['id']

name = row['Recipe Name']

ingredients = [ingredient.strip() for ingredient in row['Ingredients'].split(', ')]

category = row['Category']

recipe = Recipe(id, name, ingredients, category)

added\_recipe = self.add\_recipe(recipe)

print("Data copied from CSV to the database successfully.")

except FileNotFoundError as e:

print(f"Error: The specified CSV file '{filename}' was not found: {e}")

except Exception as e:

print("An error occurred: ", e)

def create\_recipe(self):

name = input("Enter recipe name: ").strip()

ingredients = input("Enter ingredients (comma-separated): ").split(',')

try:

category = self.select\_category\_options()

return Recipe(None, name, ingredients, category)

except Exception as e:

print("An error has occurred.", e)

def select\_category\_options(self):

try:

rows = super().get\_cursor.execute("""SELECT \* FROM Categories;""").fetchall()

if rows:

print("Available Category Options: ")

last\_id = 1

for row in rows:

id = row[0]

category\_name = row[1]

last\_id = id

print(f"{id}. {category\_name}")

custom\_id = last\_id + 1

print(f"{custom\_id}. Create a Custom Category")

category\_option = int(input("Enter choice number of category: "))

if category\_option == custom\_id:

category = input("Enter custom category: ").strip()

new\_category = Category(None, category, 'N')

self.add\_category(new\_category)

print(f"Added {new\_category.category\_name} category successfully.")

save\_categories\_to\_csv(new\_category)

else:

category = rows[category\_option - 1][1].strip()

else:

category = input("Enter choice number of category: ").strip()

return category

except Exception as e:

print("An error has occurred.", e)

def add\_category(self, category):

try:

super().get\_cursor.execute("INSERT INTO Categories (category\_name, predefined) VALUES (?, ?)",

(category.category\_name, category.predefined))

super().get\_connection.commit()

category\_id = super().get\_cursor.execute("""SELECT category\_id FROM Categories WHERE category\_name = ?;""",

(category.category\_name,)).fetchone()

category.category\_id = category\_id[0]

self.category\_options.append(category)

except Exception as e:

print("An error has occurred while adding a category: ", e)

def add\_recipe(self, recipe):

try:

super().get\_cursor.execute("INSERT INTO Recipes (name, ingredients, category) VALUES (?, ?, ?)",

(recipe.name, ', '.join(recipe.ingredients), recipe.category))

super().get\_connection.commit()

id = super().get\_cursor.execute("""SELECT id FROM Recipes WHERE name = ?;""", (recipe.name,)).fetchone()

recipe.id = id[0]

self.recipes.append(recipe)

return recipe

except Exception as e:

print("An error has occurred while adding a recipe: ", e)

def get\_recipes\_by\_category(self, category):

recipes = []

try:

if category is not None:

rows = super().get\_cursor.execute("""SELECT \* FROM Recipes WHERE category like ? ;""",

(category,)).fetchall()

for row in rows:

id = row[0]

name = row[1]

ingredients = [ingredient.strip() for ingredient in row[2].split(',')]

category = row[3]

recipes.append(Recipe(id, name, ingredients, category))

except Exception as e:

print("An error has occurred.", e)

return recipes

def get\_recipes\_by\_ingredient(self, ingredients):

recipes = []

try:

if ingredients is not None:

query = "SELECT \* FROM Recipes WHERE"

# Append '%' to each ingredient to perform a partial match.

ingredients\_list = ingredients.split(',')

query1 = " ingredients LIKE "

query2 = ""

for ingredient in ingredients\_list:

query2 = query2 + query1 + "'%" + ingredient.strip() + "%' " + 'OR'

n = 2

replaceStr = ";"

query2 = query2[:-n] + replaceStr

sql = query + " " + query2

rows = super().get\_cursor.execute(sql).fetchall()

for row in rows:

id = row[0]

name = row[1]

ingredients = [ingredient.strip() for ingredient in row[2].split(',')]

category = row[3]

recipes.append(Recipe(id, name, ingredients, category))

except Exception as e:

print("An error has occurred.", e)

return recipes

def get\_all\_recipes(self):

recipes = []

try:

rows = super().get\_cursor.execute("""SELECT \* FROM Recipes;""").fetchall()

for row in rows:

id = row[0]

name = row[1]

ingredients = [ingredient.strip() for ingredient in row[2].split(',')]

category = row[3]

recipes.append(Recipe(id, name, ingredients, category))

if recipes:

print(f"---------- All recipes ---------- ")

for recipe in recipes:

print("\n")

print(f"---------- {recipe.name} recipe: ---------- ")

print(f"ID of {recipe.name} recipe:")

print(recipe.id)

print(f"Ingredients of {recipe.name} recipe : ")

print(recipe.ingredients)

print(f"Category of {recipe.name} recipe : ")

print(recipe.category)

return recipes

else:

print("No recipes found!")

except Exception as e:

print("An error has occurred.", e)

def view\_recipe(self):

recipe\_name = input("Enter the recipe name you want to view: ").strip()

recipes = []

try:

if recipe\_name is not None:

rows = super().get\_cursor.execute("""SELECT \* FROM Recipes WHERE name like ? ;""",

(recipe\_name.strip(),)).fetchall()

for row in rows:

id = row[0]

name = row[1]

ingredients = [ingredient.strip() for ingredient in row[2].split(',')]

category = row[3]

recipes.append(Recipe(id, name, ingredients, category))

if recipes:

print(f"---------- {recipe\_name} recipe ---------- ")

for recipe in recipes:

print(f"ID of {recipe\_name} recipe : ")

print(recipe.id)

print(f"Ingredients of {recipe\_name} recipe : ")

print(recipe.ingredients)

print(f"Category of {recipe\_name} recipe : ")

print(recipe.category)

else:

print(f"{recipe\_name} recipe not found!")

else:

print(f"{recipe\_name} recipe not found!")

except Exception as e:

print("An error has occurred.", e)

def view\_recipe\_or\_continue(self):

val = input("Do you want to view the detailed recipe: (y/ n) ").strip()

if val == 'y':

self.view\_recipe()

else:

print("Continuing to the main menu!")

def update\_recipe(self, recipe\_id):

try:

recipe = self.get\_cursor.execute("SELECT \* FROM Recipes WHERE id=?", (recipe\_id,)).fetchone()

if recipe:

ingredients\_changed = False

recipe\_updated = False

updated\_recipe = Recipe(recipe\_id, recipe[1], recipe[2], recipe[3])

name\_change = input(f"Do you want to update Recipe Name of ID {recipe\_id}: (y/n) ").strip()

if name\_change.lower() == "y":

updated\_recipe\_name = input("Enter the new recipe name: ").strip()

updated\_recipe.name = updated\_recipe\_name

recipe\_updated = True

category\_change = input(f"Do you want to update the Category of ID {recipe\_id}: (y/n) ").strip()

if category\_change.lower() == "y":

new\_category = self.select\_category\_options()

updated\_recipe.category = new\_category

recipe\_updated = True

ingredients\_change = input(f"Do you want to update the Ingredients of ID {recipe\_id}: (y/n) ").strip()

if ingredients\_change.lower() == "y":

new\_ingredients\_list = input("Enter the new ingredients (comma-separated): ").split(',')

ingredients\_changed = True

updated\_recipe.ingredients = new\_ingredients\_list

if ingredients\_changed:

self.get\_cursor.execute("""

UPDATE Recipes

SET name=?, ingredients=?, category=?

WHERE id=?

""", (

updated\_recipe.name, ', '.join(updated\_recipe.ingredients), updated\_recipe.category, recipe\_id))

self.get\_connection.commit()

# update CSV record

self.update\_recipe\_in\_CSV(updated\_recipe, ingredients\_changed)

elif recipe\_updated:

self.get\_cursor.execute("""

UPDATE Recipes

SET name=?, ingredients=?, category=?

WHERE id=?

""", (

updated\_recipe.name, updated\_recipe.ingredients, updated\_recipe.category, recipe\_id))

self.get\_connection.commit()

# update CSV record

self.update\_recipe\_in\_CSV(updated\_recipe, ingredients\_changed)

else:

print(f"Recipe with {recipe\_id} is not updated!")

print(f"Recipe with ID {recipe\_id} updated successfully.")

else:

print(f"Recipe with ID {recipe\_id} not found.")

except Exception as e:

print("An error has occurred.", e)

def update\_recipe\_in\_CSV(self, updated\_recipe, ingredients\_changed):

try:

filename = 'recipes.csv'

df = pd.read\_csv(filename)

df.loc[int(updated\_recipe.id) - 1, 'Recipe Name'] = updated\_recipe.name

df.loc[int(updated\_recipe.id) - 1, 'Category'] = updated\_recipe.category

if ingredients\_changed:

df.loc[int(updated\_recipe.id) - 1, 'Ingredients'] = ', '.join(updated\_recipe.ingredients)

df.to\_csv(filename, index=False)

except FileNotFoundError as e:

print(f"Error: The specified CSV file was not found: {e}")

except Exception as e:

print("An error occurred: ", e)

def delete\_recipe(self, recipe\_id):

try:

recipe = self.get\_cursor.execute("SELECT \* FROM Recipes WHERE id=?", (recipe\_id,)).fetchone()

if recipe:

self.get\_cursor.execute("DELETE FROM Recipes WHERE id=?", (recipe\_id,))

self.get\_connection.commit()

self.delete\_recipe\_in\_CSV(recipe\_id)

print(f"Recipe with ID {recipe\_id} deleted successfully.")

else:

print(f"Recipe with ID {recipe\_id} not found.")

except Exception as e:

print("An error has occurred.", e)

def delete\_recipe\_in\_CSV(self, deleted\_recipe\_id):

try:

filename = 'recipes.csv'

df = pd.read\_csv(filename)

# Find the index of the row with the matching recipe\_id

index\_to\_delete = df[df['id'] == deleted\_recipe\_id].index

if not index\_to\_delete.empty:

# Drop the row with the specified index

df.drop(index\_to\_delete, inplace=True)

# Save the updated DataFrame back to the CSV file

df.to\_csv(filename, index=False)

except FileNotFoundError as e:

print(f"Error: The specified CSV file was not found: {e}")

except Exception as e:

print("An error occurred: ", e)

def save\_recipes\_to\_csv(recipe):

file\_name = 'recipes.csv'

with open(file\_name, 'a', newline='') as file:

writer = csv.writer(file)

# Check if the file is empty

is\_empty = os.stat(file\_name).st\_size == 0

if is\_empty:

writer.writerow(["id", "Recipe Name", "Category", "Ingredients"])

writer.writerow([recipe.id, recipe.name, recipe.category, ', '.join(recipe.ingredients)])

def save\_categories\_to\_csv(category):

file\_name = 'categories.csv'

with open(file\_name, 'a', newline='') as file:

writer = csv.writer(file)

# Check if the file is empty

is\_empty = os.stat(file\_name).st\_size == 0

if is\_empty:

writer.writerow(["Category id", "Category Name", "Predefined"])

writer.writerow([category.category\_id, category.category\_name, category.predefined])

def main():

recipe\_manager = RecipeManager("recipesDB.sqlite")

# recipe\_manager.reset\_or\_create\_db()

recipe\_manager.check\_DB()

while True:

print("\nRecipe Manager Menu:")

print("1. Create a Recipe")

print("2. View Recipes by Category")

print("3. View Recipes by Ingredient")

print("4. View Recipes by Recipe name")

print("5. View all Recipes")

print("6. Update Recipes")

print("7. Delete Recipes")

print("8. Quit")

choice = input("Enter the number of your choice: ").strip()

if choice == '1':

try:

recipe = recipe\_manager.create\_recipe()

added\_recipe = recipe\_manager.add\_recipe(recipe)

save\_recipes\_to\_csv(added\_recipe)

print(f"Recipe '{recipe.name}' added successfully!")

except Exception as e:

print("An error has occurred.", e)

elif choice == '2':

try:

category = input("Enter category to view recipes: ").strip()

recipes = recipe\_manager.get\_recipes\_by\_category(category.strip())

if recipes:

print(f"Recipes in the '{category}' category:")

for recipe in recipes:

print(recipe.name)

recipe\_manager.view\_recipe\_or\_continue()

else:

print(f"No recipes found in the '{category}' category.")

except Exception as e:

print("An error has occurred.", e)

elif choice == '3':

try:

ingredient = input("Enter ingredient to view recipes: ").strip()

recipes = recipe\_manager.get\_recipes\_by\_ingredient(ingredient.strip())

if recipes:

print(f"Recipes containing '{ingredient}':")

for recipe in recipes:

print(recipe.name)

recipe\_manager.view\_recipe\_or\_continue()

else:

print(f"No recipes found containing '{ingredient}'.")

except Exception as e:

print("An error has occurred.", e)

elif choice == '4':

try:

recipe\_manager.view\_recipe()

except Exception as e:

print("An error has occurred.", e)

elif choice == '5':

try:

recipes = recipe\_manager.get\_all\_recipes()

except Exception as e:

print("An error has occurred.", e)

elif choice == '6':

try:

recipes = recipe\_manager.get\_all\_recipes()

if recipes:

id = input("Enter the Recipe ID to update: ")

if id:

recipe\_manager.update\_recipe(id)

else:

print("Invalid input!")

except Exception as e:

print("An error has occurred.", e)

elif choice == '7':

try:

recipes = recipe\_manager.get\_all\_recipes()

id = int(input("Enter the Recipe ID you want to delete: "))

recipe\_manager.delete\_recipe(id)

except Exception as e:

print("An error has occurred.", e)

elif choice == '8':

try:

recipe\_manager.close\_db() # Close the database connection

print("Thank you for using the Recipe Creator and Manager Application!")

break

except Exception as e:

print("An error has occurred.", e)

else:

print("Invalid choice. Please try again.")

if \_\_name\_\_ == "\_\_main\_\_":

main()