**JAVA SWING BASED – FOOD AND NUTRITION SUGGESYION SYSTEM- SQL CONNECTIVITY USING JDBC**

*A*

*Report*

*Submitted in partial fulfilment of the*

*Requirements for the award of the Degree of*

**BACHELOR OF ENGINEERING**

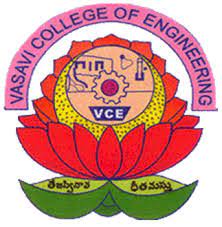
**IN**

**INFORMATION TECHNOLOGY**

**By**

**B.Kirankumar<1602-21-737-025>**

**Under the guidance of Ms B. Leelavathy**



**Department of Information Technology**

**Vasavi College of Engineering (Autonomous)**

**(Affiliated to Osmania University)**

**Ibrahimbagh, Hyderabad-31**

**2022-2023**

**BONAFIDE CERTIFICATE**

This is to certify that this project report titled

‘***FOOD AND NUTRITION SUGGESTION SYSTEM***’

is a project work of ***B.KiranKumar***

bearing roll no. 1602-21-737-025

who carried out

this project under my supervision

in the IV semester

of the academic year 2022- 2023

Signature Signature

External Examiner Internal Examiner

**FOOD AND NUTRITION SUGGESTION SYSTEM**

**SUBJECT :** DATABASE MANAGEMENT SYSTEM

**ABSTARCT**:

In recent years, the increasing prevalence of lifestyle diseases has led to a growing interest in personalized nutrition recommendations. The Food and Nutrition Suggestion System (FNSS) is a computer-based system that leverages machine learning algorithms to provide personalized food and nutrition recommendations to users. The system takes into account the user's dietary preferences, health conditions, and lifestyle factors to generate personalized food and nutrition plans. The FNSS also provides users with access to a database of recipes that are tailored to their specific nutritional needs. The system aims to improve the user's overall health and well-being by providing them with a comprehensive and personalized approach to nutrition. The effectiveness of the FNSS has been demonstrated through various studies, highlighting its potential to reduce the risk of chronic diseases and improve overall health outcomes.

**REQUIREMENTS:**

Over all 5 tables are required for the ER model representation, they are:

1.Users

2.Food

3.Meal

4.Healthcondition

5.Nutritionsuggestion

**Table 1: USERS**

**Attributes**

* userid (primary key)
* name(varchar)
* age(number)
* gender(varchar)
* height(float)
* weight(float)
* activity\_level (varchar)

**Domain Types:**

* user id,age:integer
* height,weight:float
* name,gender,activity\_level:varchar

**constraints:**

primary key:user\_id

**Table 2: FOOD**

**Attributes:**

**-**foodid(number)

-name(varchar)

-calorie(float)

-carbohydrate(float)

-Protein(float)

-fat(float)

**Domain Types**:

-foodid:integer

-calorie,carbohydrate,protein,fat:float

-name:varchar

**Constraints:**

-prinmary key:food\_id

**Table 3:Meal**

**Attributes:**

**-**mealid(number)

-userid(number)

-Foodid(number)

-Quantity(float)

-date(date)

**Domain Types:**

-mealid,userid,foodid:integer

-quantity-float

- date:date

**Constraints:**

Primary key: meal\_id

Foreign key: user\_id,food\_id

**Table 4:Healthcondition**

**Attributes:**

-healthcondition\_id(number)

-user\_id(number)

-condition\_name(varchar)

-condition\_description(varchar)

**Domain Types:**

-Healthcondition\_id,user\_id:integer

-condition\_name,condition\_description:varchar

**Constarints:**

Primary key:healthcondition\_id

Foreign key:user\_id

**Table 5:NutritionSuggestion**

**Attributes:**

-suggestion\_id(number)

-user\_id(number)

-suggestion(varchar)

-suggestion\_date(date)

**Domain Types:**

- suggestion\_id , userid :integer

--suggestion: varchar

- suggestion\_date: date

**Constraints:**

Primary key:suggestion\_id

Foreign key:user\_id;

**Mapping Cardinalities:**

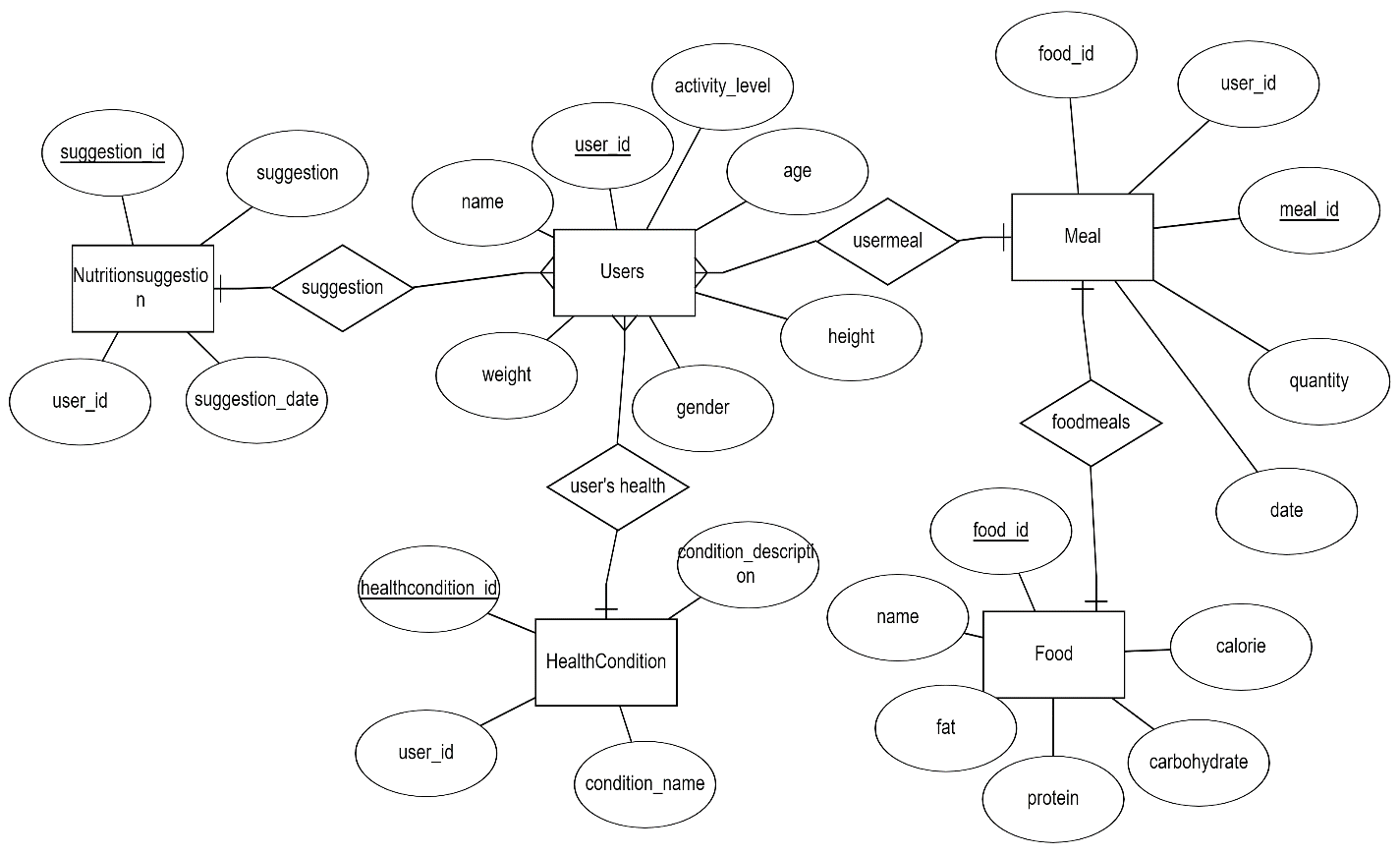
- One User can have Many Meals (One-to-Many)

- One Meal can have One Food (One-to-One)

- One User can have Many Health Conditions (One-to-Many)

- One User can have Many Nutrition Suggestions (One-to-Many)

**ER DAIGRAM:**



**DDL COMMANDS:**

**USER TABLE:**

**a)create():**

SQL> create table Users(

2 user\_id number,

3 name varchar(50),

4 age number,

5 gender varchar(50),

6 height float,

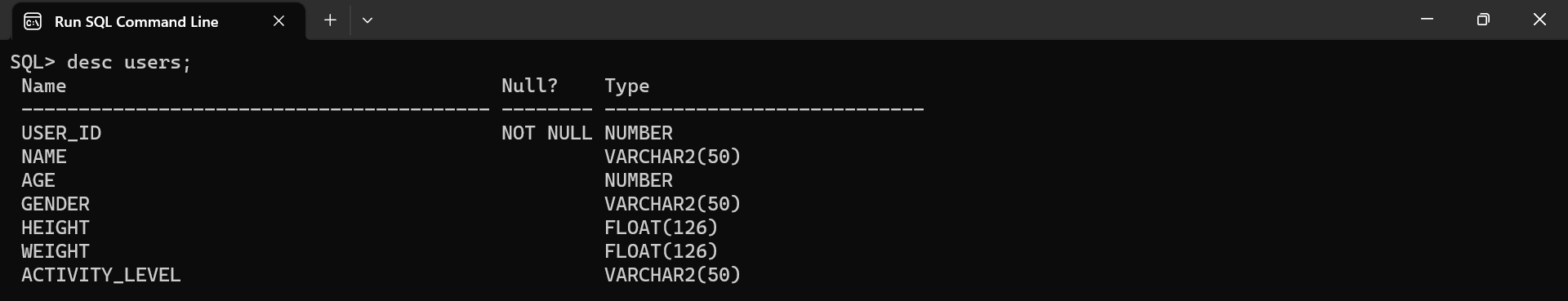
7 weight float,

8 activity\_level varchar(50),

9 primary key(user\_id));

Table created.

****

****

**FOOD TABLE:**

SQL> create table Food(

2 food\_id number,

3 name varchar(50),

4 calorie float,

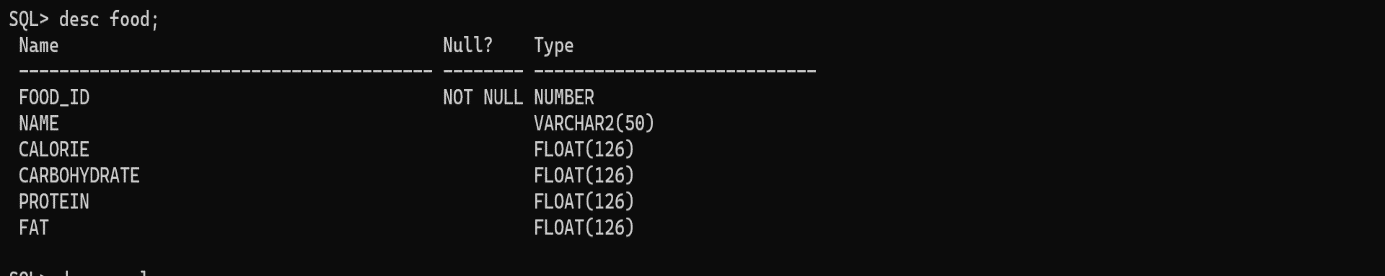
5 carbohydrate float,

6 protein float,

7 fat float,

8 primary key(food\_id));

Table created.

****

****

**MEAL TABLE:**

SQL> SQL> create table Meal(

2 meal\_id number,

3 user\_id number,

4 food\_id number,

5 quantity float,

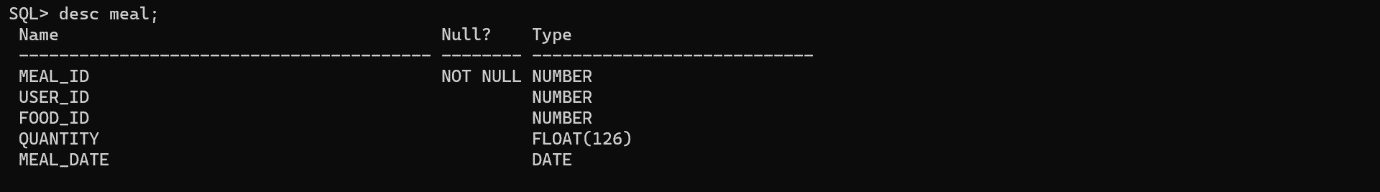
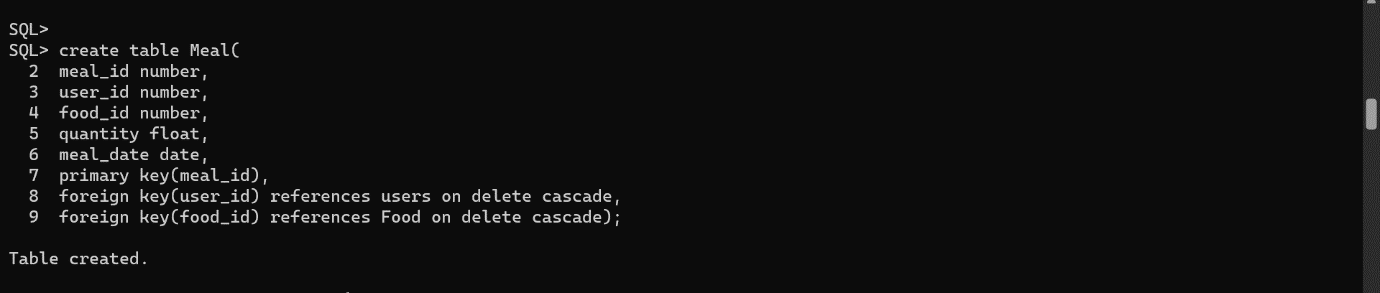
6 meal\_date date,

7 primary key(meal\_id),

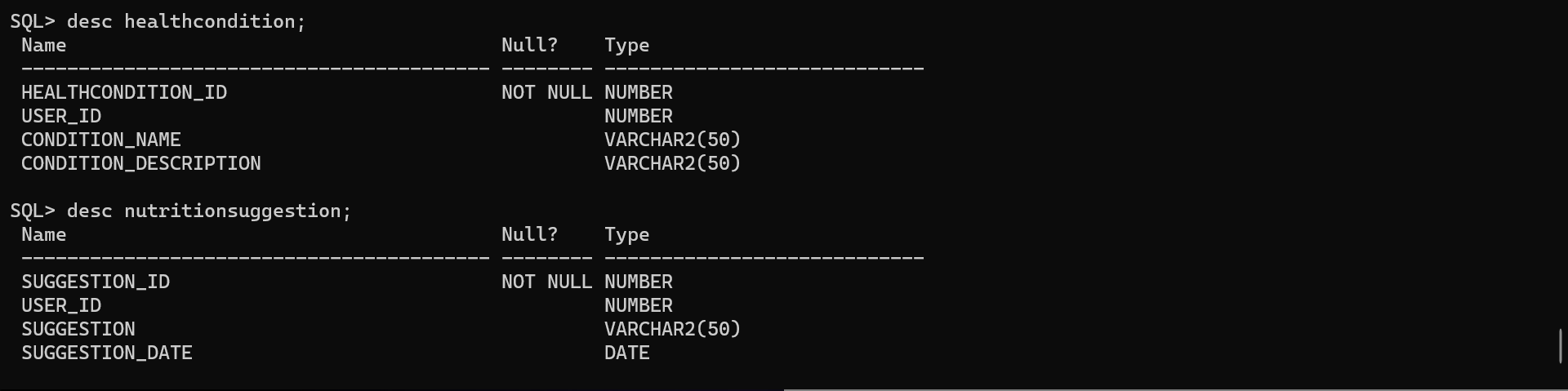
8 foreign key(user\_id) references users on delete cascade,

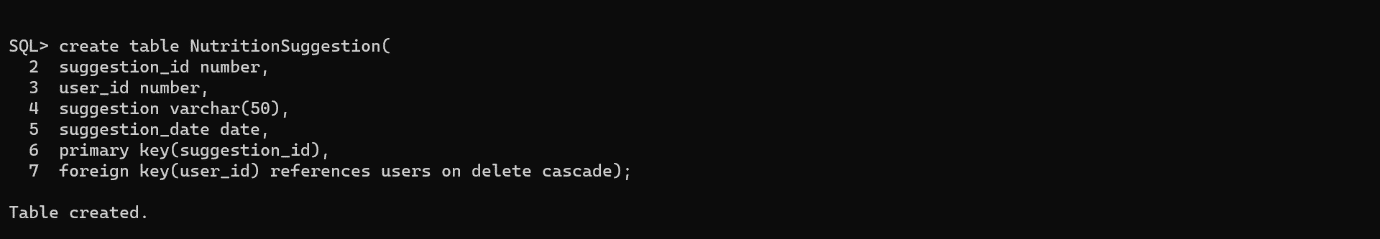
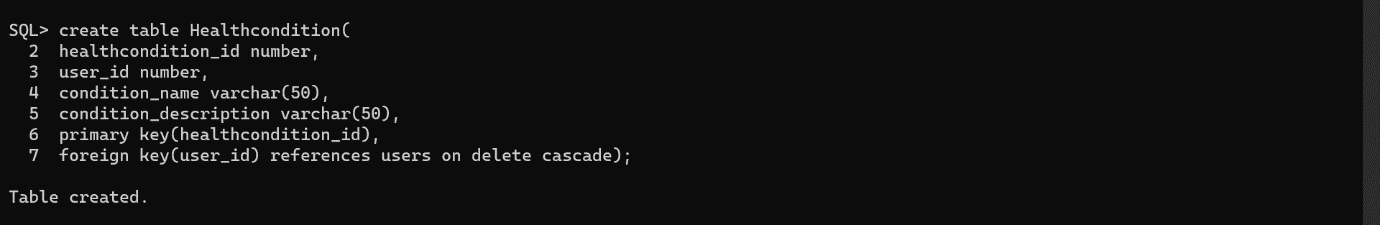
9 foreign key(food\_id) references Food on delete cascade);

Table created.

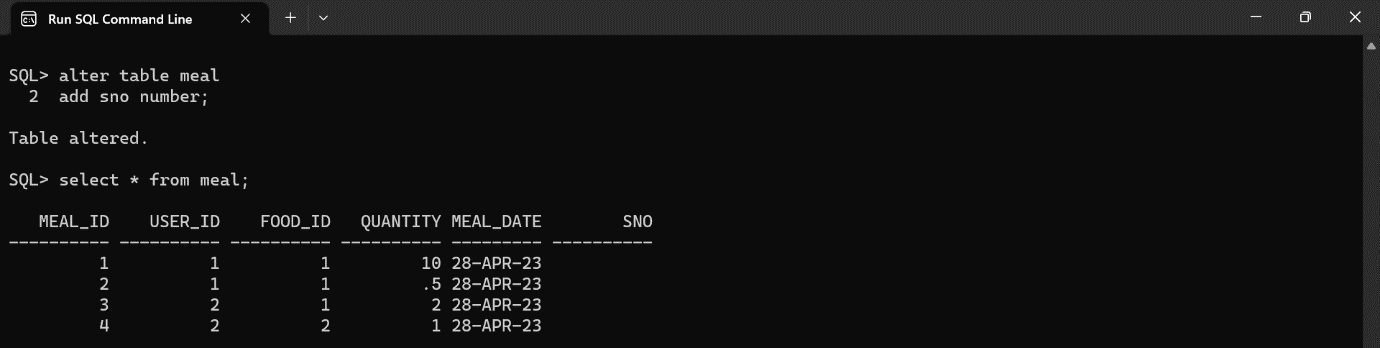
****

**HealthCondition &NutritionSuggestion Table:**

****

****

**b)alter():**

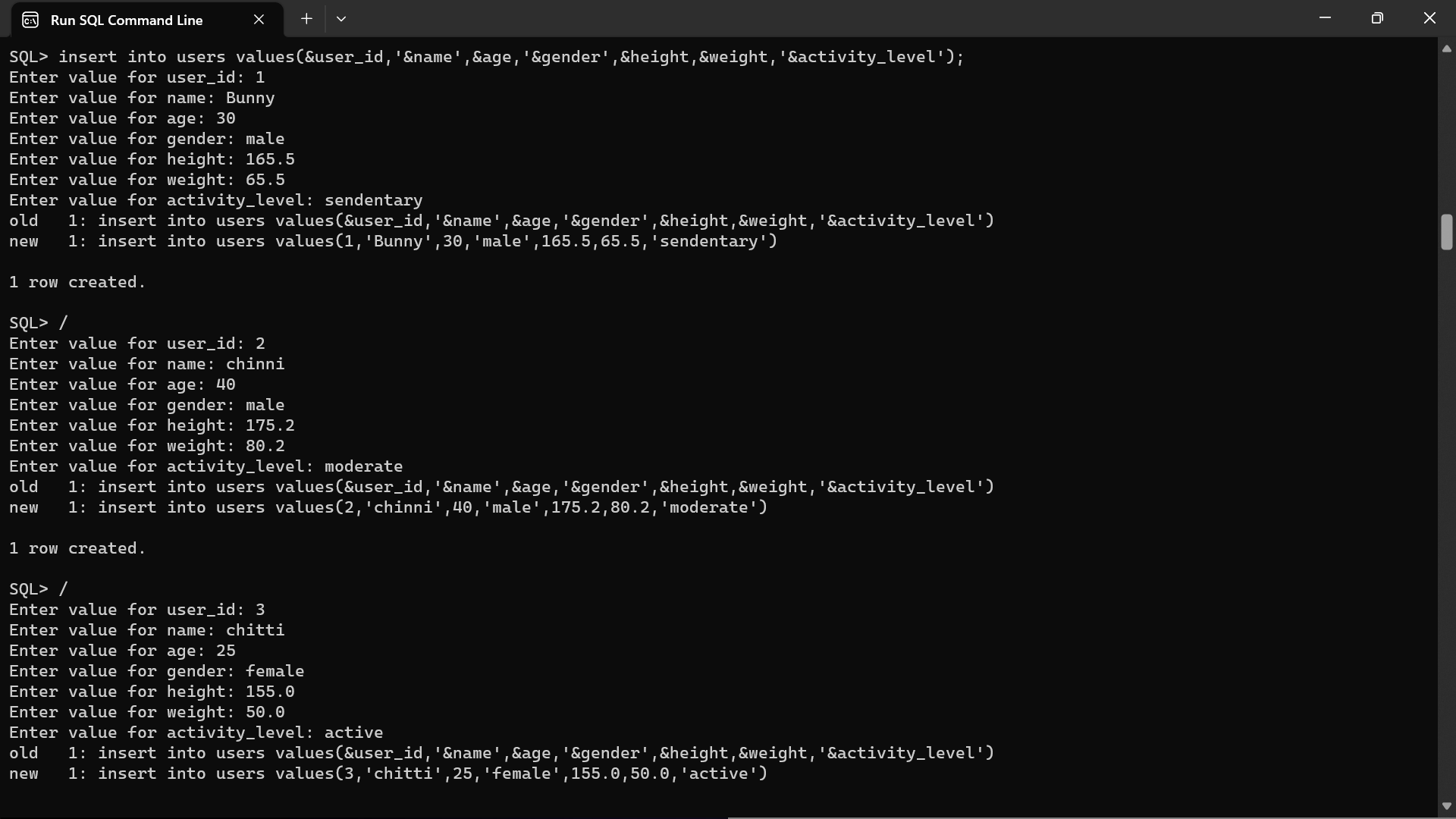
****

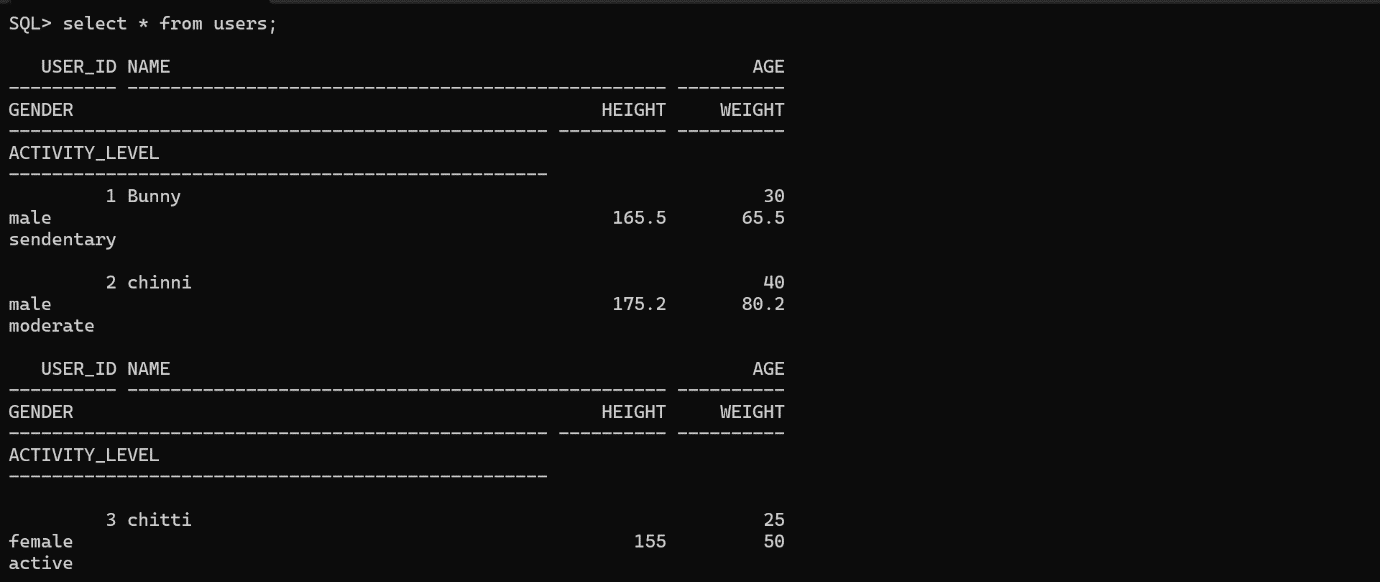
**C)drop:**

****

**DML COMMANDS:**

**USERS TABLE:**

****

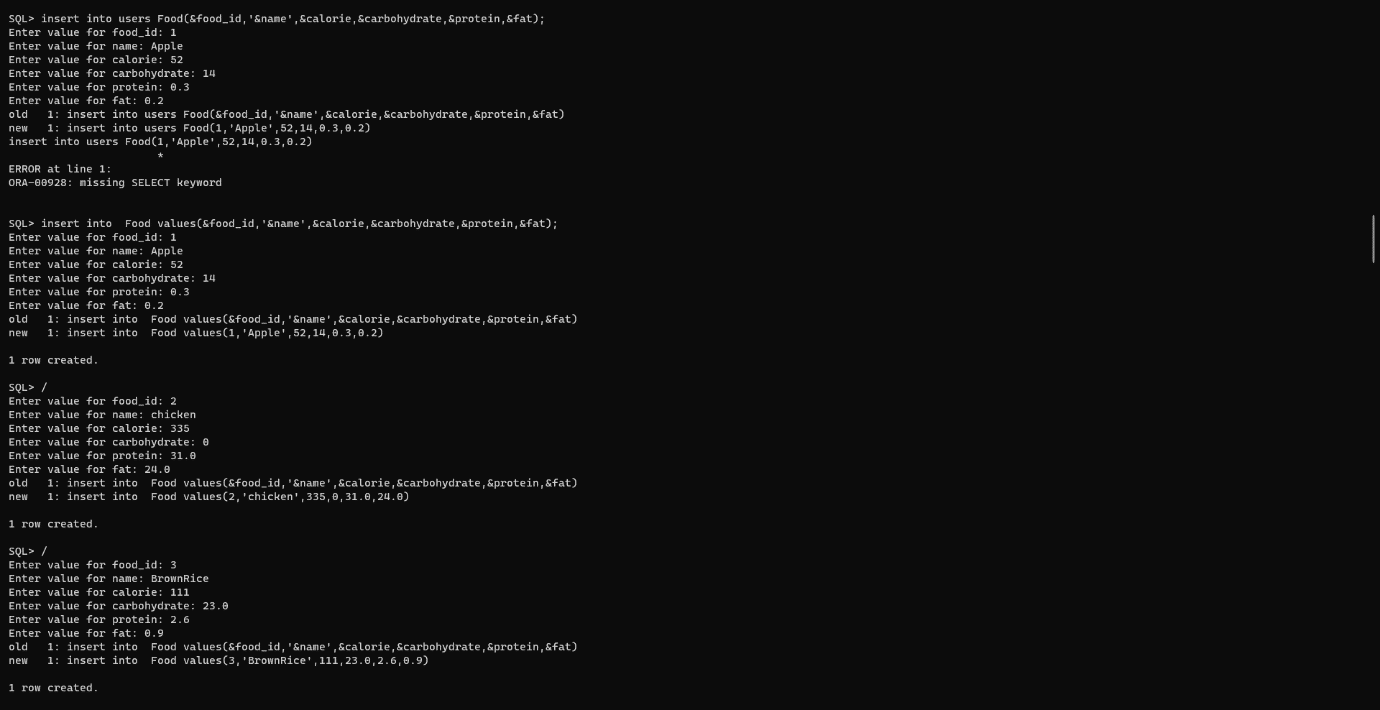


**FOOD TABLE:**

****

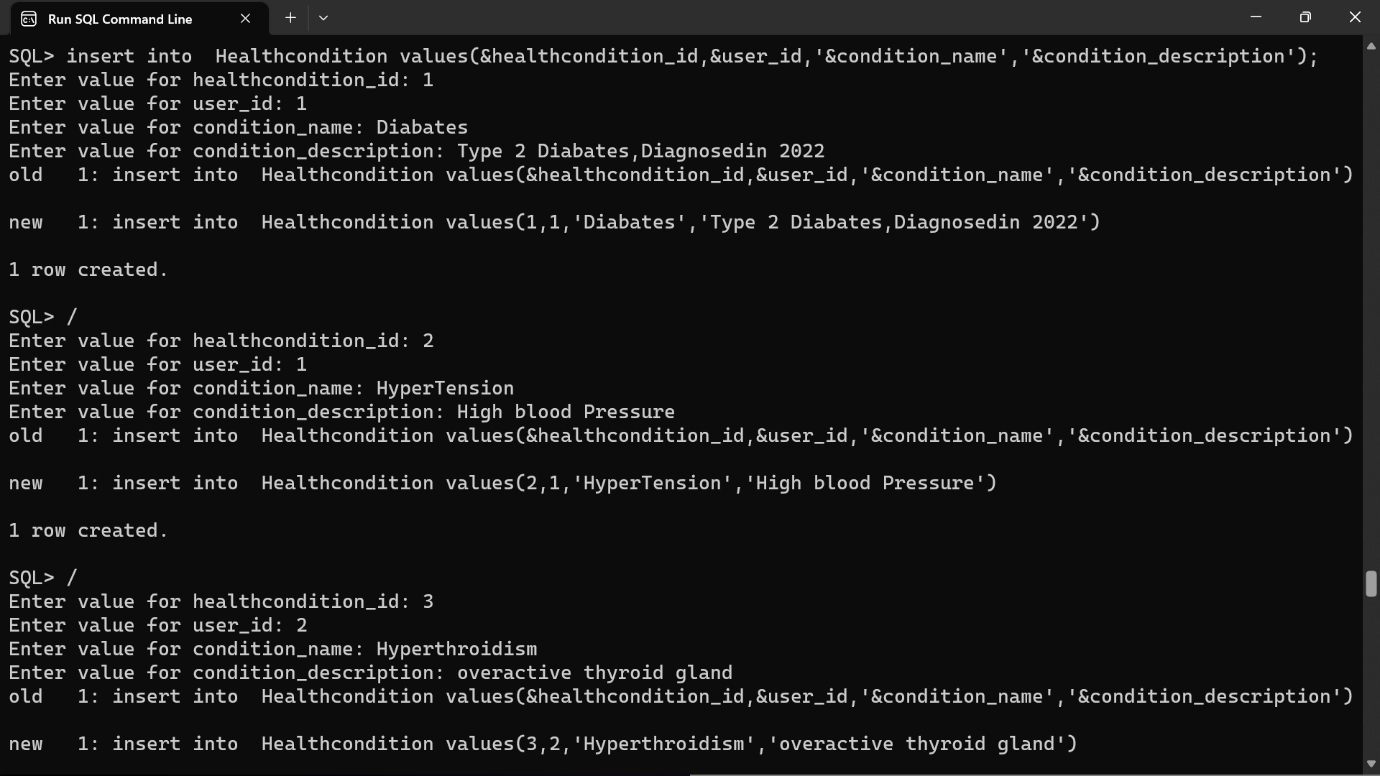


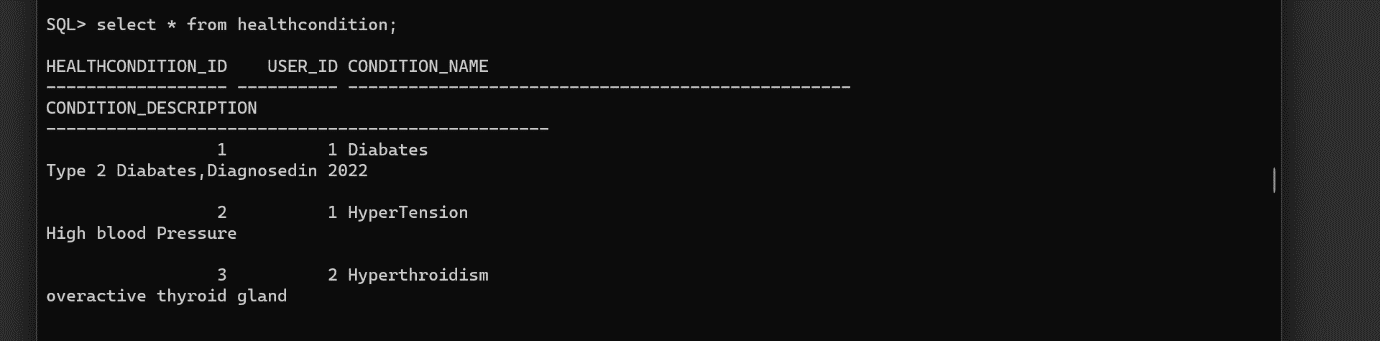
**MEAL TABLE:**

****

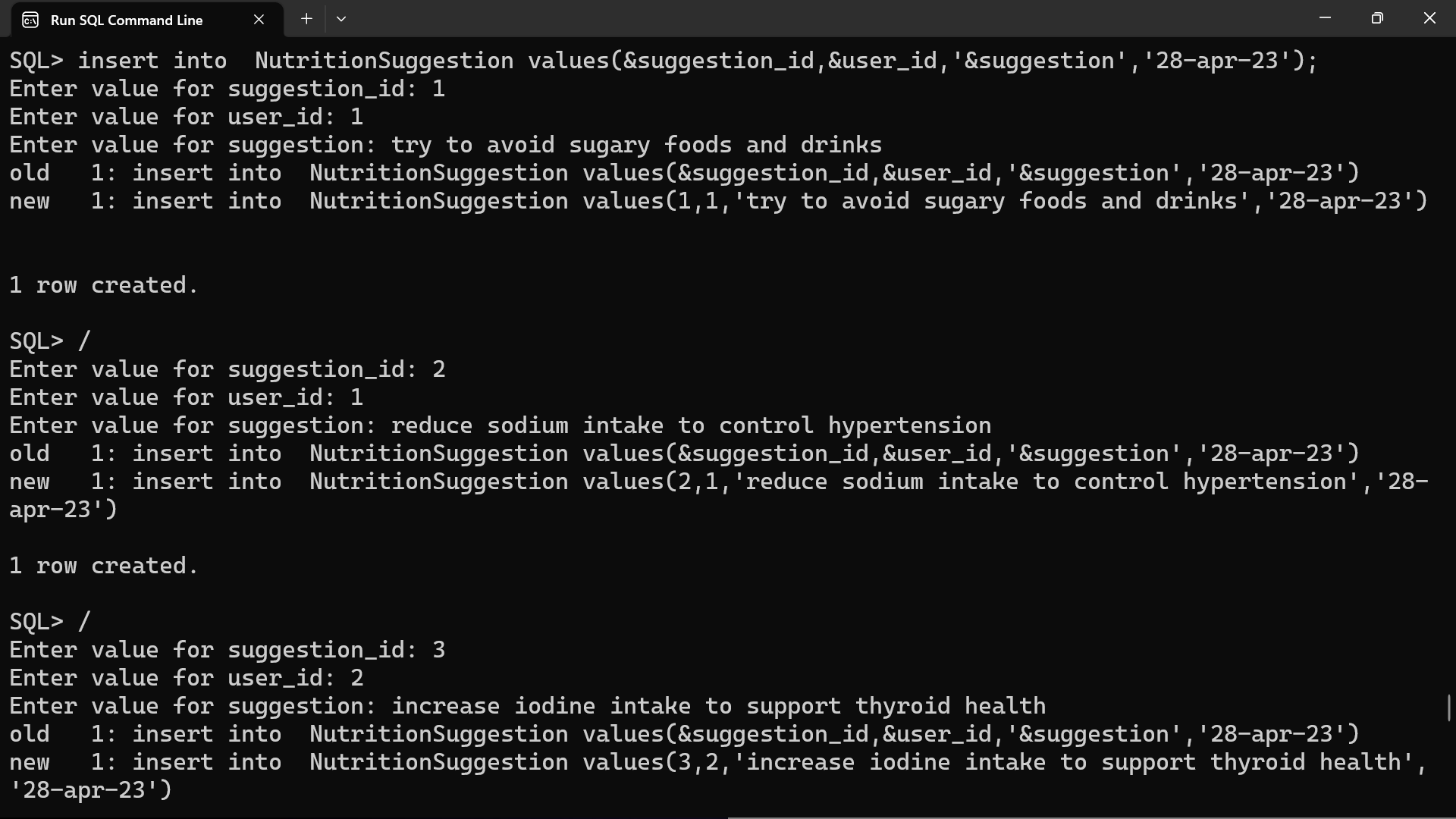


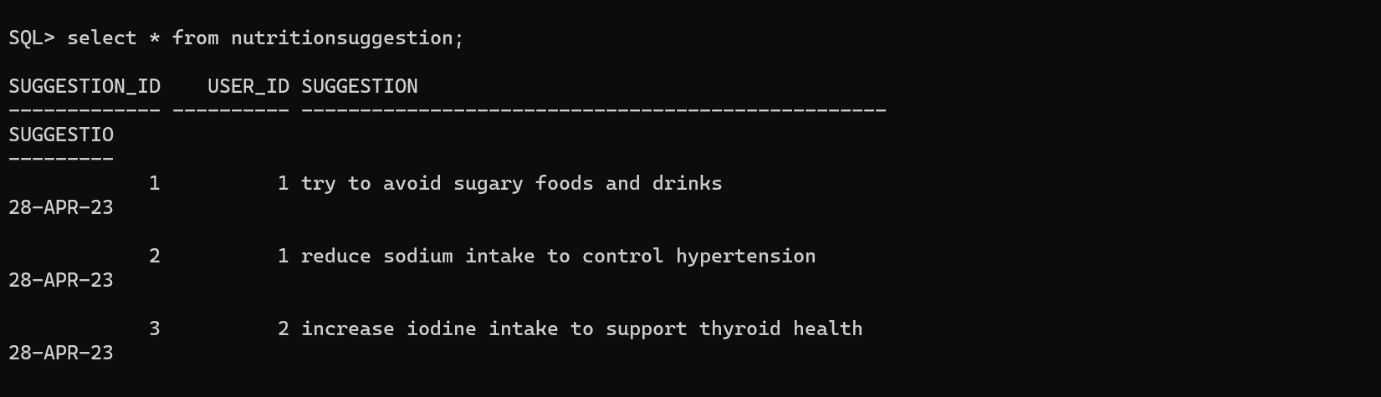
**HealthCondition Table:**

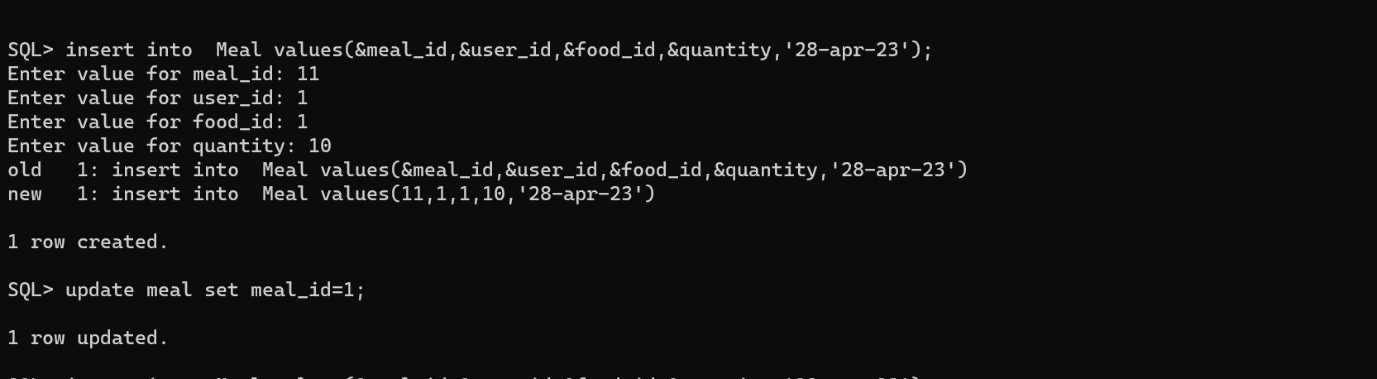
****



**NUTRITIONSUGGESTION TABLE:**







**IMPLEMENTATION**

**JAVA- SQL Connectivity using JDBC:**

**Java Database Connectivity (JDBC)** is an application programming interface (API) for the programming language Java, which defines how a client may access a database. It is a Java-based data access technology used for Java database connectivity. It is part of the Java Standard Edition platform, from Oracle Corporation. It provides methods to query and update data in a database and is oriented towards relational databases.

The connection to the database can be performed using Java programming (JDBC API) as:

**package main;**

import java.sql.\*;

public class ConnectionManager (

private static String url = "jdbc:oracle: thin:@localhost:1521:xe"; private static String username = "kiran";

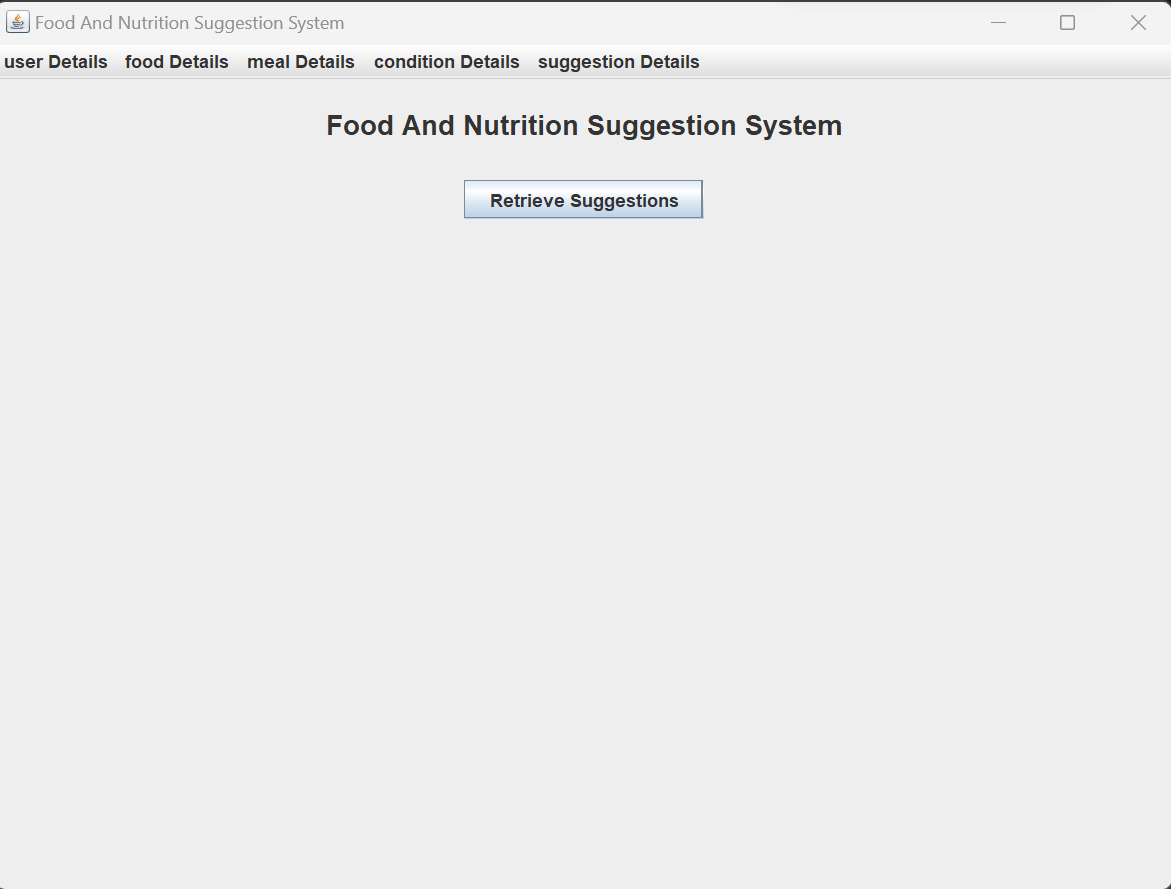
private static String password = "bunny";

private static Connection con;

public static Connection getConnection () throws Exception [ con = DriverManager.getConnection (url, username, password);

return con;

**Main Page:**



Code:

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

public class MainPage extends JFrame {

/\*\*

\*

\*/

private static final long serialVersionUID = 1L;

private JButton retrievedetailsButton;

public MainPage() {

setTitle("Food And Nutrition Suggestion System");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

JLabel welcomeLabel = new JLabel("Food And Nutrition Suggestion System");

welcomeLabel.setFont(new Font("Arial", Font.BOLD, 18));

welcomeLabel.setHorizontalAlignment(SwingConstants.CENTER);

welcomeLabel.setBorder(BorderFactory.createEmptyBorder(20, 0, 20, 0));

add(welcomeLabel, BorderLayout.NORTH);

JPanel buttonPanel = new JPanel();

retrievedetailsButton = new JButton("Retrieve Marks");

buttonPanel.add(retrievedetailsButton);

JMenuBar menuBar = new JMenuBar();

JMenu UserTableMenu = new JMenu("user Details");

JMenu FoodMenu = new JMenu("food Details");

JMenu MealMenu = new JMenu("meal Details");

JMenu HealthConditionMenu = new JMenu("condition Details");

JMenu NutritionSuggestionMenu = new JMenu("suggestion Details");

// Create menu item for student menu

JMenuItem viewUserTableDetails = new JMenuItem("View user Details");

viewUserTableDetails.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

new UserTable();

}

});

JMenuItem viewFoodDetails = new JMenuItem("View Food Details");

viewFoodDetails.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

new Food();

}

});

JMenuItem viewMealDetails = new JMenuItem("View Meal Details");

viewMealDetails.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

new Meal();

}

});

JMenuItem viewHealthConditionDetails = new JMenuItem("View HealthCondition Details");

viewHealthConditionDetails.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

new HealthCondition();

}

});

JMenuItem viewNutritionSuggestionDetails = new JMenuItem("View Suggestion Details");

viewNutritionSuggestionDetails.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

new NutritionSuggestion();

}

});

UserTableMenu.add(viewUserTableDetails);

FoodMenu.add(viewFoodDetails);

MealMenu.add(viewMealDetails);

HealthConditionMenu.add(viewHealthConditionDetails);

NutritionSuggestionMenu.add(viewNutritionSuggestionDetails);

menuBar.add(UserTableMenu);

menuBar.add(FoodMenu);

menuBar.add(MealMenu);

menuBar.add(HealthConditionMenu);

menuBar.add(NutritionSuggestionMenu);

setJMenuBar(menuBar);

add(buttonPanel, BorderLayout.CENTER);

retrievedetailsButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

RetrieveProgram retrieveProgram=new RetrieveProgram();

retrieveProgram.setVisible(true);

}

});

addWindowStateListener(new WindowStateListener() {

public void windowStateChanged(WindowEvent e) {

if ((e.getNewState() & Frame.MAXIMIZED\_BOTH) == Frame.MAXIMIZED\_BOTH) {

System.out.println("Window maximized");

} else {

System.out.println("Window not maximized");

}

}

});

setSize(800, 600);

setVisible(true);

}

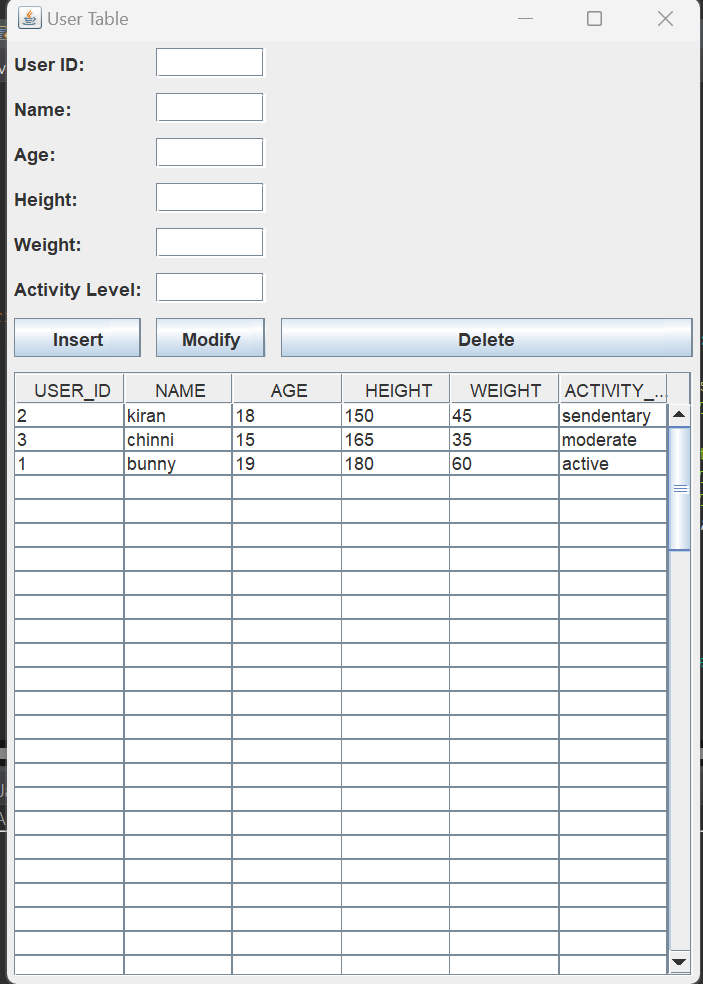
public static void main(String[] args) {

new MainPage();

}

}

**User Page:**

****

**Code:**

**import javax.swing.\*;**

**import java.awt.event.\*;**

**import java.awt.\*;**

**import java.sql.\*;**

**import javax.swing.table.DefaultTableModel;**

**public class UserTable extends JFrame {**

**private JTextField userIdField;**

**private JTextField nameField;**

**private JTextField ageField;**

**private JTextField heightField;**

**private JTextField weightField;**

**private JTextField activityLevelField;**

**private JButton insertButton;**

**private JButton modifyButton;**

**private JButton deleteButton;**

**private JTable displayTable;**

**public UserTable() {**

**setTitle("User Table");**

**setLayout(new GridBagLayout());**

**GridBagConstraints constraints = new GridBagConstraints();**

**setVisible(true);**

**constraints.fill = GridBagConstraints.HORIZONTAL;**

**constraints.insets = new Insets(5, 5, 5, 5);**

**JLabel userIdLabel = new JLabel("User ID:");**

**constraints.gridx = 0;**

**constraints.gridy = 0;**

**add(userIdLabel, constraints);**

**userIdField = new JTextField(10);**

**constraints.gridx = 1;**

**constraints.gridy = 0;**

**add(userIdField, constraints);**

**JLabel nameLabel = new JLabel("Name:");**

**constraints.gridx = 0;**

**constraints.gridy = 1;**

**add(nameLabel, constraints);**

**nameField = new JTextField(20);**

**constraints.gridx = 1;**

**constraints.gridy = 1;**

**add(nameField, constraints);**

**JLabel ageLabel = new JLabel("Age:");**

**constraints.gridx = 0;**

**constraints.gridy = 2;**

**add(ageLabel, constraints);**

**ageField = new JTextField(10);**

**constraints.gridx = 1;**

**constraints.gridy = 2;**

**add(ageField, constraints);**

**JLabel heightLabel = new JLabel("Height:");**

**constraints.gridx = 0;**

**constraints.gridy = 3;**

**add(heightLabel, constraints);**

**heightField = new JTextField(10);**

**constraints.gridx = 1;**

**constraints.gridy = 3;**

**add(heightField, constraints);**

**JLabel weightLabel = new JLabel("Weight:");**

**constraints.gridx = 0;**

**constraints.gridy = 4;**

**add(weightLabel, constraints);**

**weightField = new JTextField(10);**

**constraints.gridx = 1;**

**constraints.gridy = 4;**

**add(weightField, constraints);**

**JLabel activityLevelLabel = new JLabel("Activity Level:");**

**constraints.gridx = 0;**

**constraints.gridy = 5;**

**add(activityLevelLabel, constraints);**

**activityLevelField = new JTextField(10);**

**constraints.gridx = 1;**

**constraints.gridy = 5;**

**add(activityLevelField, constraints);**

**insertButton = new JButton("Insert");**

**constraints.gridx = 0;**

**constraints.gridy = 6;**

**add(insertButton, constraints);**

**modifyButton = new JButton("Modify");**

**constraints.gridx = 1;**

**constraints.gridy = 6;**

**add(modifyButton, constraints);**

**deleteButton = new JButton("Delete");**

**constraints.gridx = 2;**

**constraints.gridy = 6;**

**add(deleteButton, constraints);**

**displayTable = new JTable();**

**JScrollPane scrollPane = new JScrollPane(displayTable);**

**constraints.gridx = 0;**

**constraints.gridy = 7;**

**constraints.gridwidth = 3;**

**constraints.fill = GridBagConstraints.BOTH;**

**constraints.weightx = 1.0;**

**constraints.weighty = 1.0;**

**add(scrollPane, constraints);**

**addWindowListener(new WindowAdapter() {**

**@Override**

**public void windowClosing(WindowEvent e) {**

**dispose();**

**}**

**});**

**insertButton.addActionListener(new ActionListener() {**

**@Override**

**public void actionPerformed(ActionEvent e) {**

**insertUser();**

**}**

**});**

**modifyButton.addActionListener(new ActionListener() {**

**@Override**

**public void actionPerformed(ActionEvent e) {**

**modifyUser();**

**}**

**});**

**deleteButton.addActionListener(new ActionListener() {**

**@Override**

**public void actionPerformed(ActionEvent e) {**

**deleteUser();**

**}**

**});**

**displayTable.addMouseListener(new MouseAdapter() {**

**@Override**

**public void mouseClicked(MouseEvent e) {**

**int selectedRow = displayTable.getSelectedRow();**

**if (selectedRow >= 0) {**

**int userId = Integer.parseInt(displayTable.getValueAt(selectedRow, 0).toString());**

**String name = displayTable.getValueAt(selectedRow, 1).toString();**

**int age = Integer.parseInt(displayTable.getValueAt(selectedRow, 2).toString());**

**int height = Integer.parseInt(displayTable.getValueAt(selectedRow, 3).toString());**

**int weight = Integer.parseInt(displayTable.getValueAt(selectedRow, 4).toString());**

**String activityLevel = displayTable.getValueAt(selectedRow, 5).toString();**

**userIdField.setText(String.valueOf(userId));**

**nameField.setText(name);**

**ageField.setText(String.valueOf(age));**

**heightField.setText(String.valueOf(height));**

**weightField.setText(String.valueOf(weight));**

**activityLevelField.setText(activityLevel);**

**}**

**}**

**});**

**pack();**

**setLocationRelativeTo(null);**

**displayUsers();**

**}**

**private void insertUser() {**

**try (Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "kiran", "bunny")) {**

**String sql = "INSERT INTO userstable (user\_id, name, age, height, weight, activity\_level) VALUES (?, ?, ?, ?, ?, ?)";**

**PreparedStatement statement = conn.prepareStatement(sql);**

**statement.setInt(1, Integer.parseInt(userIdField.getText()));**

**statement.setString(2, nameField.getText());**

**statement.setInt(3, Integer.parseInt(ageField.getText()));**

**statement.setInt(4, Integer.parseInt(heightField.getText()));**

**statement.setInt(5, Integer.parseInt(weightField.getText()));**

**statement.setString(6, activityLevelField.getText());**

**int rowsInserted = statement.executeUpdate();**

**if (rowsInserted > 0) {**

**displayUsers();**

**clearFields();**

**JOptionPane.showMessageDialog(this, "User inserted successfully.");**

**}**

**} catch (SQLException ex) {**

**ex.printStackTrace();**

**JOptionPane.showMessageDialog(this, "Failed to insert user: " + ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);**

**}**

**}**

**private void modifyUser() {**

**try (Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "kiran", "bunny")) {**

**String sql = "UPDATE userstable SET name = ?, age = ?, height = ?, weight = ?, activity\_level = ? WHERE user\_id = ?";**

**PreparedStatement statement = conn.prepareStatement(sql);**

**statement.setString(1, nameField.getText());**

**statement.setInt(2, Integer.parseInt(ageField.getText()));**

**statement.setInt(3, Integer.parseInt(heightField.getText()));**

**statement.setInt(4, Integer.parseInt(weightField.getText()));**

**statement.setString(5, activityLevelField.getText());**

**statement.setInt(6, Integer.parseInt(userIdField.getText()));**

**int rowsUpdated = statement.executeUpdate();**

**if (rowsUpdated > 0) {**

**displayUsers();**

**clearFields();**

**JOptionPane.showMessageDialog(this, "User modified successfully.");**

**}**

**} catch (SQLException ex) {**

**ex.printStackTrace();**

**JOptionPane.showMessageDialog(this, "Failed to modify user: " + ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);**

**}**

**}**

**private void deleteUser() {**

**try (Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "kiran", "bunny")) {**

**String sql = "DELETE FROM userstable WHERE user\_id = ?";**

**PreparedStatement statement = conn.prepareStatement(sql);**

**statement.setInt(1, Integer.parseInt(userIdField.getText()));**

**int rowsDeleted = statement.executeUpdate();**

**if (rowsDeleted > 0) {**

**displayUsers();**

**clearFields();**

**JOptionPane.showMessageDialog(this, "User deleted successfully.");**

**}**

**} catch (SQLException ex) {**

**ex.printStackTrace();**

**JOptionPane.showMessageDialog(this, "Failed to delete user: " + ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);**

**}**

**}**

**private void displayUsers() {**

**try (Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "kiran", "bunny")) {**

**String sql = "SELECT user\_id, name, age, height, weight, activity\_level FROM userstable";**

**Statement statement = conn.createStatement();**

**ResultSet resultSet = statement.executeQuery(sql);**

**displayTable.setModel(buildTableModel(resultSet));**

**} catch (SQLException ex) {**

**ex.printStackTrace();**

**JOptionPane.showMessageDialog(this, "Failed to fetch user data: " + ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);**

**}**

**}**

**private void clearFields() {**

**userIdField.setText("");**

**nameField.setText("");**

**ageField.setText("");**

**heightField.setText("");**

**weightField.setText("");**

**activityLevelField.setText("");**

**}**

**private DefaultTableModel buildTableModel(ResultSet resultSet) throws SQLException {**

**ResultSetMetaData metaData = resultSet.getMetaData();**

**int columnCount = metaData.getColumnCount();**

**String[] columnNames = new String[columnCount];**

**for (int i = 0; i < columnCount; i++) {**

**columnNames[i] = metaData.getColumnLabel(i + 1);**

**}**

**Object[][] data = new Object[100][columnCount];**

**int rowCount = 0;**

**while (resultSet.next()) {**

**for (int i = 0; i < columnCount; i++) {**

**data[rowCount][i] = resultSet.getObject(i + 1);**

**}**

**rowCount++;**

**}**

**return new DefaultTableModel(data, columnNames);**

**}**

**public static void main(String[] args) {**

**SwingUtilities.invokeLater(new Runnable() {**

**public void run() {**

**new UserTable();**

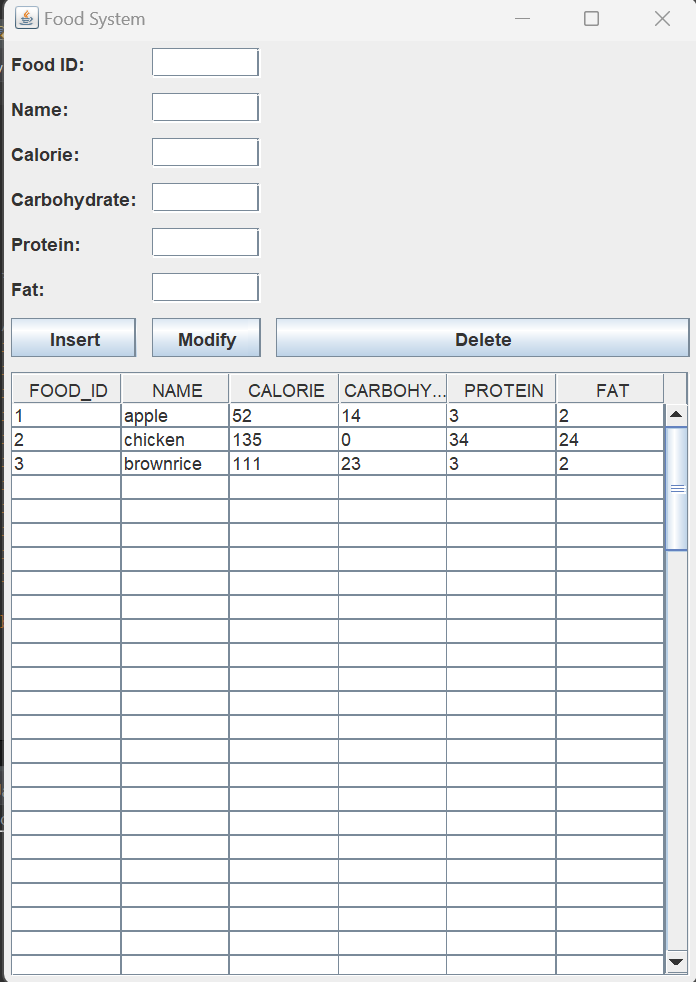
**}**

**});**

**}**

**}**

**Food Page:**

****

**Code:**

**import javax.swing.\*;**

**import java.awt.event.\*;**

**import java.awt.\*;**

**import java.sql.\*;**

**import javax.swing.table.DefaultTableModel;**

**public class Food extends JFrame {**

**private static final long serialVersionUID = 1L;**

**private JTextField foodIdField;**

**private JTextField nameField;**

**private JTextField calorieField;**

**private JTextField carbohydrateField;**

**private JTextField proteinField;**

**private JTextField fatField;**

**private JButton insertButton;**

**private JButton modifyButton;**

**private JButton deleteButton;**

**private JTable displayTable;**

**public Food() {**

**setTitle("Food System");**

**setLayout(new GridBagLayout());**

**GridBagConstraints constraints = new GridBagConstraints();**

**setVisible(true);**

**constraints.fill = GridBagConstraints.HORIZONTAL;**

**constraints.insets = new Insets(5, 5, 5, 5);**

**JLabel foodIdLabel = new JLabel("Food ID:");**

**constraints.gridx = 0;**

**constraints.gridy = 0;**

**add(foodIdLabel, constraints);**

**foodIdField = new JTextField(10);**

**constraints.gridx = 1;**

**constraints.gridy = 0;**

**add(foodIdField, constraints);**

**JLabel nameLabel = new JLabel("Name:");**

**constraints.gridx = 0;**

**constraints.gridy = 1;**

**add(nameLabel, constraints);**

**nameField = new JTextField(20);**

**constraints.gridx = 1;**

**constraints.gridy = 1;**

**add(nameField, constraints);**

**JLabel calorieLabel = new JLabel("Calorie:");**

**constraints.gridx = 0;**

**constraints.gridy = 2;**

**add(calorieLabel, constraints);**

**calorieField = new JTextField(10);**

**constraints.gridx = 1;**

**constraints.gridy = 2;**

**add(calorieField, constraints);**

**JLabel carbohydrateLabel = new JLabel("Carbohydrate:");**

**constraints.gridx = 0;**

**constraints.gridy = 3;**

**add(carbohydrateLabel, constraints);**

**carbohydrateField = new JTextField(10);**

**constraints.gridx = 1;**

**constraints.gridy = 3;**

**add(carbohydrateField, constraints);**

**JLabel proteinLabel = new JLabel("Protein:");**

**constraints.gridx = 0;**

**constraints.gridy = 4;**

**add(proteinLabel, constraints);**

**proteinField = new JTextField(10);**

**constraints.gridx = 1;**

**constraints.gridy = 4;**

**add(proteinField, constraints);**

**JLabel fatLabel = new JLabel("Fat:");**

**constraints.gridx = 0;**

**constraints.gridy = 5;**

**add(fatLabel, constraints);**

**fatField = new JTextField(10);**

**constraints.gridx = 1;**

**constraints.gridy = 5;**

**add(fatField, constraints);**

**insertButton = new JButton("Insert");**

**constraints.gridx = 0;**

**constraints.gridy = 6;**

**add(insertButton, constraints);**

**modifyButton = new JButton("Modify");**

**constraints.gridx = 1;**

**constraints.gridy = 6;**

**add(modifyButton, constraints);**

**deleteButton = new JButton("Delete");**

**constraints.gridx = 2;**

**constraints.gridy = 6;**

**add(deleteButton, constraints);**

**displayTable = new JTable();**

**JScrollPane scrollPane = new JScrollPane(displayTable);**

**constraints.gridx = 0;**

**constraints.gridy = 7;**

**constraints.gridwidth = 3;**

**constraints.fill = GridBagConstraints.BOTH;**

**constraints.weightx = 1.0;**

**constraints.weighty = 1.0;**

**add(scrollPane, constraints);**

**addWindowListener(new WindowAdapter() {**

**@Override**

**public void windowClosing(WindowEvent e) {**

**dispose();**

**}**

**});**

**insertButton.addActionListener(new ActionListener() {**

**@Override**

**public void actionPerformed(ActionEvent e) {**

**insertFood();**

**}**

**});**

**modifyButton.addActionListener(new ActionListener() {**

**@Override**

**public void actionPerformed(ActionEvent e) {**

**modifyFood();**

**}**

**});**

**deleteButton.addActionListener(new ActionListener() {**

**@Override**

**public void actionPerformed(ActionEvent e) {**

**deleteFood();**

**}**

**});**

**displayTable.addMouseListener(new MouseAdapter() {**

**@Override**

**public void mouseClicked(MouseEvent e) {**

**int selectedRow = displayTable.getSelectedRow();**

**if (selectedRow >= 0) {**

**int foodId = Integer.parseInt(displayTable.getValueAt(selectedRow, 0).toString());**

**String name = displayTable.getValueAt(selectedRow, 1).toString();**

**int calorie = Integer.parseInt(displayTable.getValueAt(selectedRow, 2).toString());**

**int carbohydrate = Integer.parseInt(displayTable.getValueAt(selectedRow, 3).toString());**

**int protein = Integer.parseInt(displayTable.getValueAt(selectedRow, 4).toString());**

**int fat = Integer.parseInt(displayTable.getValueAt(selectedRow, 5).toString());**

**foodIdField.setText(String.valueOf(foodId));**

**nameField.setText(name);**

**calorieField.setText(String.valueOf(calorie));**

**carbohydrateField.setText(String.valueOf(carbohydrate));**

**proteinField.setText(String.valueOf(protein));**

**fatField.setText(String.valueOf(fat));**

**}**

**}**

**});**

**pack();**

**setLocationRelativeTo(null);**

**displayFoods();**

**}**

**private void insertFood() {**

**try (Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "kiran", "bunny")) {**

**String sql = "INSERT INTO food (food\_id, name, calorie, carbohydrate, protein, fat) VALUES (?, ?, ?, ?, ?, ?)";**

**PreparedStatement statement = conn.prepareStatement(sql);**

**statement.setInt(1, Integer.parseInt(foodIdField.getText()));**

**statement.setString(2, nameField.getText());**

**statement.setInt(3, Integer.parseInt(calorieField.getText()));**

**statement.setInt(4, Integer.parseInt(carbohydrateField.getText()));**

**statement.setInt(5, Integer.parseInt(proteinField.getText()));**

**statement.setInt(6, Integer.parseInt(fatField.getText()));**

**int rowsInserted = statement.executeUpdate();**

**if (rowsInserted > 0) {**

**displayFoods();**

**clearFields();**

**JOptionPane.showMessageDialog(this, "Food inserted successfully.");**

**}**

**} catch (SQLException ex) {**

**ex.printStackTrace();**

**JOptionPane.showMessageDialog(this, "Failed to insert food: " + ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);**

**}**

**}**

**private void modifyFood() {**

**try (Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "kiran", "bunny")) {**

**String sql = "UPDATE food SET name = ?, calorie = ?, carbohydrate = ?, protein = ?, fat = ? WHERE food\_id = ?";**

**PreparedStatement statement = conn.prepareStatement(sql);**

**statement.setString(1, nameField.getText());**

**statement.setInt(2, Integer.parseInt(calorieField.getText()));**

**statement.setInt(3, Integer.parseInt(carbohydrateField.getText()));**

**statement.setInt(4, Integer.parseInt(proteinField.getText()));**

**statement.setInt(5, Integer.parseInt(fatField.getText()));**

**statement.setInt(6, Integer.parseInt(foodIdField.getText()));**

**int rowsUpdated = statement.executeUpdate();**

**if (rowsUpdated > 0) {**

**displayFoods();**

**clearFields();**

**JOptionPane.showMessageDialog(this, "Food modified successfully.");**

**}**

**} catch (SQLException ex) {**

**ex.printStackTrace();**

**JOptionPane.showMessageDialog(this, "Failed to modify food: " + ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);**

**}**

**}**

**private void deleteFood() {**

**try (Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "kiran", "bunny")) {**

**String sql = "DELETE FROM food WHERE food\_id = ?";**

**PreparedStatement statement = conn.prepareStatement(sql);**

**statement.setInt(1, Integer.parseInt(foodIdField.getText()));**

**int rowsDeleted = statement.executeUpdate();**

**if (rowsDeleted > 0) {**

**displayFoods();**

**clearFields();**

**JOptionPane.showMessageDialog(this, "Food deleted successfully.");**

**}**

**} catch (SQLException ex) {**

**ex.printStackTrace();**

**JOptionPane.showMessageDialog(this, "Failed to delete food: " + ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);**

**}**

**}**

**private void displayFoods() {**

**try (Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "kiran", "bunny")) {**

**String sql = "SELECT food\_id, name, calorie, carbohydrate, protein, fat FROM food";**

**Statement statement = conn.createStatement();**

**ResultSet resultSet = statement.executeQuery(sql);**

**displayTable.setModel(buildTableModel(resultSet));**

**} catch (SQLException ex) {**

**ex.printStackTrace();**

**JOptionPane.showMessageDialog(this, "Failed to fetch food data: " + ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);**

**}**

**} private void clearFields() {**

**foodIdField.setText("");**

**nameField.setText("");**

**calorieField.setText("");**

**carbohydrateField.setText("");**

**proteinField.setText("");**

**fatField.setText("");**

**}**

**private DefaultTableModel buildTableModel(ResultSet resultSet) throws SQLException {**

**ResultSetMetaData metaData = resultSet.getMetaData();**

**int columnCount = metaData.getColumnCount();**

**String[] columnNames = new String[columnCount];**

**for (int i = 0; i < columnCount; i++) {**

**columnNames[i] = metaData.getColumnLabel(i + 1);**

**}**

**Object[][] data = new Object[100][columnCount];**

**int rowCount = 0;**

**while (resultSet.next()) {**

**for (int i = 0; i < columnCount; i++) {**

**data[rowCount][i] = resultSet.getObject(i + 1);**

**}**

**rowCount++;**

**}**

**return new DefaultTableModel(data, columnNames);**

**}**

**public static void main(String[] args) {**

**SwingUtilities.invokeLater(new Runnable() {**

**public void run() {**

**new Food();**

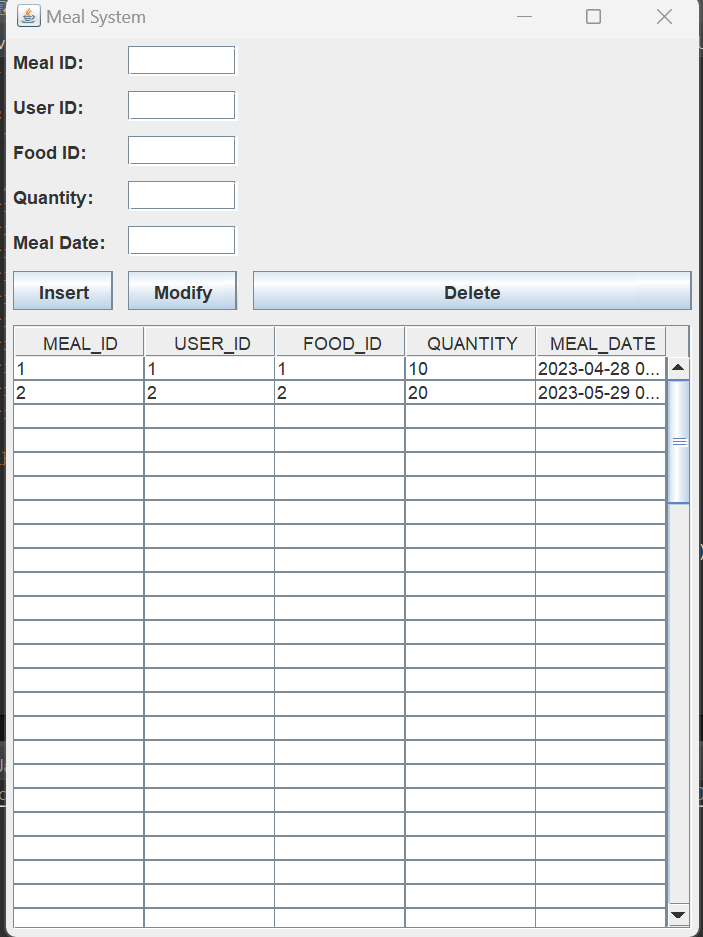
**}**

**});**

**}**

**}**

**Meal Page:**

****

**Code:**

import javax.swing.\*;

import java.awt.event.\*;

import java.awt.\*;

import java.sql.\*;

import javax.swing.table.DefaultTableModel;

public class Meal extends JFrame {

private static final long serialVersionUID = 1L;

private JTextField mealIdField;

private JTextField userIdField;

private JTextField foodIdField;

private JTextField quantityField;

private JTextField mealDateField;

private JButton insertButton;

private JButton modifyButton;

private JButton deleteButton;

private JTable displayTable;

public Meal() {

setTitle("Meal System");

setLayout(new GridBagLayout());

GridBagConstraints constraints = new GridBagConstraints();

setVisible(true);

constraints.fill = GridBagConstraints.HORIZONTAL;

constraints.insets = new Insets(5, 5, 5, 5);

JLabel mealIdLabel = new JLabel("Meal ID:");

constraints.gridx = 0;

constraints.gridy = 0;

add(mealIdLabel, constraints);

mealIdField = new JTextField(10);

constraints.gridx = 1;

constraints.gridy = 0;

add(mealIdField, constraints);

JLabel userIdLabel = new JLabel("User ID:");

constraints.gridx = 0;

constraints.gridy = 1;

add(userIdLabel, constraints);

userIdField = new JTextField(10);

constraints.gridx = 1;

constraints.gridy = 1;

add(userIdField, constraints);

JLabel foodIdLabel = new JLabel("Food ID:");

constraints.gridx = 0;

constraints.gridy = 2;

add(foodIdLabel, constraints);

foodIdField = new JTextField(10);

constraints.gridx = 1;

constraints.gridy = 2;

add(foodIdField, constraints);

JLabel quantityLabel = new JLabel("Quantity:");

constraints.gridx = 0;

constraints.gridy = 3;

add(quantityLabel, constraints);

quantityField = new JTextField(10);

constraints.gridx = 1;

constraints.gridy = 3;

add(quantityField, constraints);

JLabel mealDateLabel = new JLabel("Meal Date:");

constraints.gridx = 0;

constraints.gridy = 4;

add(mealDateLabel, constraints);

mealDateField = new JTextField(10);

constraints.gridx = 1;

constraints.gridy = 4;

add(mealDateField, constraints);

insertButton = new JButton("Insert");

constraints.gridx = 0;

constraints.gridy = 5;

add(insertButton, constraints);

modifyButton = new JButton("Modify");

constraints.gridx = 1;

constraints.gridy = 5;

add(modifyButton, constraints);

deleteButton = new JButton("Delete");

constraints.gridx = 2;

constraints.gridy = 5;

add(deleteButton, constraints);

displayTable = new JTable();

JScrollPane scrollPane = new JScrollPane(displayTable);

constraints.gridx = 0;

constraints.gridy = 6;

constraints.gridwidth = 3;

constraints.fill = GridBagConstraints.BOTH;

constraints.weightx = 1.0;

constraints.weighty = 1.0;

add(scrollPane, constraints);

addWindowListener(new WindowAdapter() {

@Override

public void windowClosing(WindowEvent e) {

dispose();

}

});

insertButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

insertMeal();

}

});

modifyButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

modifyMeal();

}

});

deleteButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

deleteMeal();

}

});

displayTable.addMouseListener(new MouseAdapter() {

@Override

public void mouseClicked(MouseEvent e) {

int selectedRow = displayTable.getSelectedRow();

if (selectedRow >= 0) {

int mealId = Integer.parseInt(displayTable.getValueAt(selectedRow, 0).toString());

int userId = Integer.parseInt(displayTable.getValueAt(selectedRow, 1).toString());

int foodId = Integer.parseInt(displayTable.getValueAt(selectedRow, 2).toString());

int quantity = Integer.parseInt(displayTable.getValueAt(selectedRow, 3).toString());

String mealDate = displayTable.getValueAt(selectedRow, 4).toString();

mealIdField.setText(String.valueOf(mealId));

userIdField.setText(String.valueOf(userId));

foodIdField.setText(String.valueOf(foodId));

quantityField.setText(String.valueOf(quantity));

mealDateField.setText(mealDate);

}

}

});

pack();

setLocationRelativeTo(null);

displayMeals();

}

private void insertMeal() {

try (Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "kiran", "bunny")) {

String sql = "INSERT INTO meal (meal\_id, user\_id, food\_id, quantity, meal\_date) VALUES (?, ?, ?, ?, ?)";

PreparedStatement statement = conn.prepareStatement(sql);

statement.setInt(1, Integer.parseInt(mealIdField.getText()));

statement.setInt(2, Integer.parseInt(userIdField.getText()));

statement.setInt(3, Integer.parseInt(foodIdField.getText()));

statement.setInt(4, Integer.parseInt(quantityField.getText()));

statement.setString(5, mealDateField.getText());

int rowsInserted = statement.executeUpdate();

if (rowsInserted > 0) {

displayMeals();

clearFields();

JOptionPane.showMessageDialog(this, "Meal inserted successfully.");

}

} catch (SQLException ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(this, "Failed to insert meal: " + ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

}

}

private void modifyMeal() {

try (Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "kiran", "bunny")) {

String sql = "UPDATE meal SET user\_id = ?, food\_id = ?, quantity = ?, meal\_date = ? WHERE meal\_id = ?";

PreparedStatement statement = conn.prepareStatement(sql);

statement.setInt(1, Integer.parseInt(userIdField.getText()));

statement.setInt(2, Integer.parseInt(foodIdField.getText()));

statement.setInt(3, Integer.parseInt(quantityField.getText()));

statement.setString(4, mealDateField.getText());

statement.setInt(5, Integer.parseInt(mealIdField.getText()));

int rowsUpdated = statement.executeUpdate();

if (rowsUpdated > 0) {

displayMeals();

clearFields();

JOptionPane.showMessageDialog(this, "Meal modified successfully.");

}

} catch (SQLException ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(this, "Failed to modify meal: " + ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

}

}

private void deleteMeal() {

try (Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "kiran", "bunny")) {

String sql = "DELETE FROM meal WHERE meal\_id = ?";

PreparedStatement statement = conn.prepareStatement(sql);

statement.setInt(1, Integer.parseInt(mealIdField.getText()));

int rowsDeleted = statement.executeUpdate();

if (rowsDeleted > 0) {

displayMeals();

clearFields();

JOptionPane.showMessageDialog(this, "Meal deleted successfully.");

}

} catch (SQLException ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(this, "Failed to delete meal: " + ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

}

}

private void displayMeals() {

try (Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "kiran", "bunny")) {

String sql = "SELECT meal\_id, user\_id, food\_id, quantity, meal\_date FROM meal";

Statement statement = conn.createStatement();

ResultSet resultSet = statement.executeQuery(sql);

displayTable.setModel(buildTableModel(resultSet));

} catch (SQLException ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(this, "Failed to fetch meal data: " + ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

}

}

private void clearFields() {

mealIdField.setText("");

userIdField.setText("");

foodIdField.setText("");

quantityField.setText("");

mealDateField.setText("");

}

private DefaultTableModel buildTableModel(ResultSet resultSet) throws SQLException {

ResultSetMetaData metaData = resultSet.getMetaData();

int columnCount = metaData.getColumnCount();

String[] columnNames = new String[columnCount];

for (int i = 0; i < columnCount; i++) {

columnNames[i] = metaData.getColumnLabel(i + 1);

}

Object[][] data = new Object[100][columnCount];

int rowCount = 0;

while (resultSet.next()) {

for (int i = 0; i < columnCount; i++) {

data[rowCount][i] = resultSet.getObject(i + 1);

}

rowCount++;

}

return new DefaultTableModel(data, columnNames);

}

public static void main(String[] args) {

SwingUtilities.invokeLater(new Runnable() {

public void run() {

new Meal();

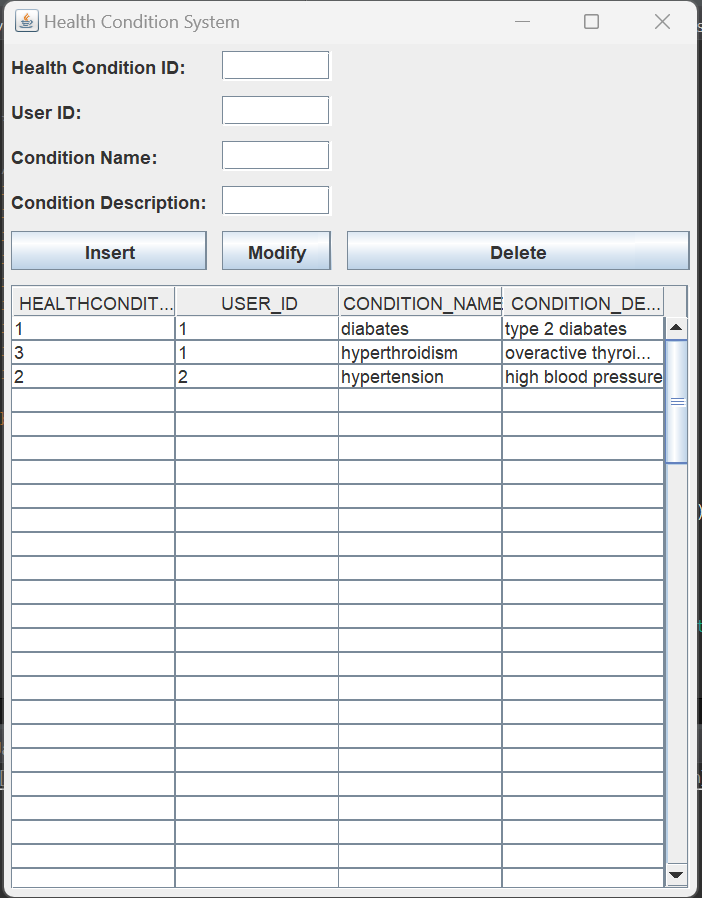
}

});

}

}

**HealthCondition Page:**

****

**Code:**

**import javax.swing.\*;**

**import java.awt.event.\*;**

**import java.awt.\*;**

**import java.sql.\*;**

**import javax.swing.table.DefaultTableModel;**

**public class HealthCondition extends JFrame {**

**/\*\***

**\***

**\*/**

**private static final long serialVersionUID = 1L;**

**private JTextField healthConditionIdField;**

**private JTextField userIdField;**

**private JTextField conditionNameField;**

**private JTextField conditionDescriptionField;**

**private JButton insertButton;**

**private JButton modifyButton;**

**private JButton deleteButton;**

**private JTable displayTable;**

**public HealthCondition() {**

**setTitle("Health Condition System");**

**setLayout(new GridBagLayout());**

**GridBagConstraints constraints = new GridBagConstraints();**

**setVisible(true);**

**constraints.fill = GridBagConstraints.HORIZONTAL;**

**constraints.insets = new Insets(5, 5, 5, 5);**

**JLabel healthConditionIdLabel = new JLabel("Health Condition ID:");**

**constraints.gridx = 0;**

**constraints.gridy = 0;**

**add(healthConditionIdLabel, constraints);**

**healthConditionIdField = new JTextField(10);**

**constraints.gridx = 1;**

**constraints.gridy = 0;**

**add(healthConditionIdField, constraints);**

**JLabel userIdLabel = new JLabel("User ID:");**

**constraints.gridx = 0;**

**constraints.gridy = 1;**

**add(userIdLabel, constraints);**

**userIdField = new JTextField(10);**

**constraints.gridx = 1;**

**constraints.gridy = 1;**

**add(userIdField, constraints);**

**JLabel conditionNameLabel = new JLabel("Condition Name:");**

**constraints.gridx = 0;**

**constraints.gridy = 2;**

**add(conditionNameLabel, constraints);**

**conditionNameField = new JTextField(20);**

**constraints.gridx = 1;**

**constraints.gridy = 2;**

**add(conditionNameField, constraints);**

**JLabel conditionDescriptionLabel = new JLabel("Condition Description:");**

**constraints.gridx = 0;**

**constraints.gridy = 3;**

**add(conditionDescriptionLabel, constraints); conditionDescriptionField = new JTextField(20);**

**constraints.gridx = 1;**

**constraints.gridy = 3;**

**add(conditionDescriptionField, constraints);**

**insertButton = new JButton("Insert");**

**constraints.gridx = 0;**

**constraints.gridy = 4;**

**add(insertButton, constraints);**

**modifyButton = new JButton("Modify");**

**constraints.gridx = 1;**

**constraints.gridy = 4;**

**add(modifyButton, constraints);**

**deleteButton = new JButton("Delete");**

**constraints.gridx = 2;**

**constraints.gridy = 4;**

**add(deleteButton, constraints);**

**displayTable = new JTable();**

**JScrollPane scrollPane = new JScrollPane(displayTable);**

**constraints.gridx = 0;**

**constraints.gridy = 5;**

**constraints.gridwidth = 3;**

**constraints.fill = GridBagConstraints.BOTH;**

**constraints.weightx = 1.0;**

**constraints.weighty = 1.0;**

**add(scrollPane, constraints);**

**addWindowListener(new WindowAdapter() {**

**@Override**

**public void windowClosing(WindowEvent e) {**

**dispose();**

**}**

**});**

**insertButton.addActionListener(new ActionListener() {**

**@Override**

**public void actionPerformed(ActionEvent e) {**

**insertHealthCondition();**

**}**

**});**

**modifyButton.addActionListener(new ActionListener() {**

**@Override**

**public void actionPerformed(ActionEvent e) {**

**modifyHealthCondition();**

**}**

**});**

**deleteButton.addActionListener(new ActionListener() {**

**@Override**

**public void actionPerformed(ActionEvent e) {**

**deleteHealthCondition();**

**}**

**});**

**displayTable.addMouseListener(new MouseAdapter() {**

**@Override**

**public void mouseClicked(MouseEvent e) {**

**int selectedRow = displayTable.getSelectedRow();**

**if (selectedRow >= 0) {**

**int healthConditionId = Integer.parseInt(displayTable.getValueAt(selectedRow, 0).toString());**

**int userId = Integer.parseInt(displayTable.getValueAt(selectedRow, 1).toString());**

**String conditionName = displayTable.getValueAt(selectedRow, 2).toString();**

**String conditionDescription = displayTable.getValueAt(selectedRow, 3).toString();**

**healthConditionIdField.setText(String.valueOf(healthConditionId));**

**userIdField.setText(String.valueOf(userId));**

**conditionNameField.setText(conditionName);**

**conditionDescriptionField.setText(conditionDescription);**

**}**

**}**

**});**

**pack();**

**setLocationRelativeTo(null);**

**displayHealthConditions();**

**}**

**private void insertHealthCondition() {**

**try (Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "kiran", "bunny")) {**

**String sql = "INSERT INTO healthcondition (healthcondition\_id, user\_id, condition\_name, condition\_description) VALUES (?, ?, ?, ?)";**

**PreparedStatement statement = conn.prepareStatement(sql);**

**statement.setInt(1, Integer.parseInt(healthConditionIdField.getText()));**

**statement.setInt(2, Integer.parseInt(userIdField.getText()));**

**statement.setString(3, conditionNameField.getText());**

**statement.setString(4, conditionDescriptionField.getText());**

**int rowsInserted = statement.executeUpdate();**

**if (rowsInserted > 0) {**

**displayHealthConditions();**

**clearFields();**

**JOptionPane.showMessageDialog(this, "Health condition inserted successfully.");**

**}**

**} catch (SQLException ex) {**

**ex.printStackTrace();**

**JOptionPane.showMessageDialog(this, "Failed to insert health condition: " + ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);**

**}**

**}**

**private void modifyHealthCondition() {**

**try (Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "kiran", "bunny")) {**

**String sql = "UPDATE healthcondition SET user\_id = ?, condition\_name = ?, condition\_description = ? WHERE healthcondition\_id = ?";**

**PreparedStatement statement = conn.prepareStatement(sql);**

**statement.setInt(1, Integer.parseInt(userIdField.getText()));**

**statement.setString(2, conditionNameField.getText());**

**statement.setString(3, conditionDescriptionField.getText());**

**statement.setInt(4, Integer.parseInt(healthConditionIdField.getText()));**

**int rowsUpdated = statement.executeUpdate();**

**if (rowsUpdated > 0) {**

**displayHealthConditions();**

**clearFields();**

**JOptionPane.showMessageDialog(this, "Health condition modified successfully.");**

**}**

**} catch (SQLException ex) {**

**ex.printStackTrace();**

**JOptionPane.showMessageDialog(this, "Failed to modify health condition: " + ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);**

**}**

**}**

**private void deleteHealthCondition() {**

**try (Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "kiran", "bunny")) {**

**String sql = "DELETE FROM healthcondition WHERE healthcondition\_id = ?";**

**PreparedStatement statement = conn.prepareStatement(sql);**

**statement.setInt(1, Integer.parseInt(healthConditionIdField.getText()));**

**int rowsDeleted = statement.executeUpdate();**

**if (rowsDeleted > 0) {**

**displayHealthConditions();**

**clearFields();**

**JOptionPane.showMessageDialog(this, "Health condition deleted successfully.");**

**}**

**} catch (SQLException ex) {**

**ex.printStackTrace();**

**JOptionPane.showMessageDialog(this, "Failed to delete health condition: " + ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);**

**}**

**}**

**private void displayHealthConditions() {**

**try (Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "kiran", "bunny")) {**

**String sql = "SELECT healthcondition\_id, user\_id, condition\_name, condition\_description FROM healthcondition";**

**Statement statement = conn.createStatement();**

**ResultSet resultSet = statement.executeQuery(sql);**

**displayTable.setModel(buildTableModel(resultSet));**

**} catch (SQLException ex) {**

**ex.printStackTrace();**

**JOptionPane.showMessageDialog(this, "Failed to fetch health condition data: " + ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);**

**}**

**}**

**private void clearFields() {**

**healthConditionIdField.setText("");**

**userIdField.setText("");**

**conditionNameField.setText("");**

**conditionDescriptionField.setText("");**

**}**

**private DefaultTableModel buildTableModel(ResultSet resultSet) throws SQLException {**

**ResultSetMetaData metaData = resultSet.getMetaData();**

**int columnCount = metaData.getColumnCount();**

**String[] columnNames = new String[columnCount];**

**for (int i = 0; i < columnCount; i++) {**

**columnNames[i] = metaData.getColumnLabel(i + 1);**

**}**

**Object[][] data = new Object[100][columnCount];**

**int rowCount = 0;**

**while (resultSet.next()) {**

**for (int i = 0; i < columnCount; i++) {**

**data[rowCount][i] = resultSet.getObject(i + 1);**

**}**

**rowCount++;**

**}**

**return new DefaultTableModel(data, columnNames);**

**}**

**public static void main(String[] args) {**

**SwingUtilities.invokeLater(new Runnable() {**

**public void run() {**

**new HealthCondition();**

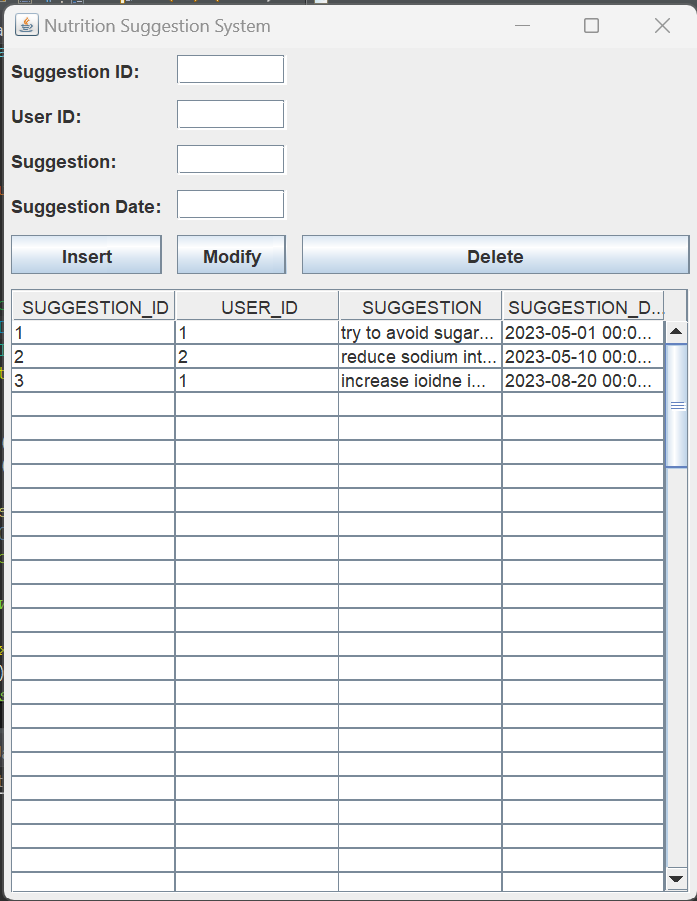
**}**

**});**

**}**

**}**

**NutritionSuggestion Page:**

****

**Code:**

**import javax.swing.\*;**

**import java.awt.event.\*;**

**import java.awt.\*;**

**import java.sql.\*;**

**import javax.swing.table.DefaultTableModel;**

**public class NutritionSuggestion extends JFrame {**

**private JTextField suggestionIdField;**

**private JTextField userIdField;**

**private JTextField suggestionField;**

**private JTextField suggestionDateField;**

**private JButton insertButton;**

**private JButton modifyButton;**

**private JButton deleteButton;**

**private JTable displayTable;**

**public NutritionSuggestion() {**

**setTitle("Nutrition Suggestion System");**

**setLayout(new GridBagLayout());**

**GridBagConstraints constraints = new GridBagConstraints();**

**setVisible(true);**

**constraints.fill = GridBagConstraints.HORIZONTAL;**

**constraints.insets = new Insets(5, 5, 5, 5);**

**JLabel suggestionIdLabel = new JLabel("Suggestion ID:");**

**constraints.gridx = 0;**

**constraints.gridy = 0;**

**add(suggestionIdLabel, constraints);suggestionIdField = new JTextField(10);**

**constraints.gridx = 1;**

**constraints.gridy = 0;**

**add(suggestionIdField, constraints);**

**JLabel userIdLabel = new JLabel("User ID:");**

**constraints.gridx = 0;**

**constraints.gridy = 1;**

**add(userIdLabel, constraints);**

**userIdField = new JTextField(10);**

**constraints.gridx = 1;**

**constraints.gridy = 1;**

**add(userIdField, constraints);**

**JLabel suggestionLabel = new JLabel("Suggestion:");**

**constraints.gridx = 0;**

**constraints.gridy = 2;**

**add(suggestionLabel, constraints);**

**suggestionField = new JTextField(20);**

**constraints.gridx = 1;**

**constraints.gridy = 2;**

**add(suggestionField, constraints);**

**JLabel suggestionDateLabel = new JLabel("Suggestion Date:");**

**constraints.gridx = 0;**

**constraints.gridy = 3;**

**add(suggestionDateLabel, constraints);**

**suggestionDateField = new JTextField(10);**

**constraints.gridx = 1;**

**constraints.gridy = 3;**

**add(suggestionDateField, constraints);**

**insertButton = new JButton("Insert");**

**constraints.gridx = 0;**

**constraints.gridy = 4;**

**add(insertButton, constraints);**

**modifyButton = new JButton("Modify");**

**constraints.gridx = 1;**

**constraints.gridy = 4;**

**add(modifyButton, constraints);**

**deleteButton = new JButton("Delete");**

**constraints.gridx = 2;**

**constraints.gridy = 4;**

**add(deleteButton, constraints);**

**displayTable = new JTable();**

**JScrollPane scrollPane = new JScrollPane(displayTable);**

**constraints.gridx = 0;**

**constraints.gridy = 5;**

**constraints.gridwidth = 3;**

**constraints.fill = GridBagConstraints.BOTH;**

**constraints.weightx = 1.0;**

**constraints.weighty = 1.0;**

**add(scrollPane, constraints);**

**addWindowListener(new WindowAdapter() {**

**@Override**

**public void windowClosing(WindowEvent e) {**

**dispose();**

**}**

**});**

**insertButton.addActionListener(new ActionListener() {**

**@Override**

**public void actionPerformed(ActionEvent e) {**

**insertSuggestion();**

**}**

**});**

**modifyButton.addActionListener(new ActionListener() {**

**@Override**

**public void actionPerformed(ActionEvent e) {**

**modifySuggestion();**

**}**

**});**

**deleteButton.addActionListener(new ActionListener() {**

**@Override**

**public void actionPerformed(ActionEvent e) {**

**deleteSuggestion();**

**}**

**});**

**displayTable.addMouseListener(new MouseAdapter() {**

**@Override**

**public void mouseClicked(MouseEvent e) {**

**int selectedRow = displayTable.getSelectedRow();**

**if (selectedRow >= 0) {**

**int suggestionId = Integer.parseInt(displayTable.getValueAt(selectedRow, 0).toString());**

**int userId = Integer.parseInt(displayTable.getValueAt(selectedRow, 1).toString());**

**String suggestion = displayTable.getValueAt(selectedRow, 2).toString();**

**String suggestionDate = displayTable.getValueAt(selectedRow, 3).toString();**

**suggestionIdField.setText(String.valueOf(suggestionId));**

**userIdField.setText(String.valueOf(userId));**

**suggestionField.setText(suggestion);**

**suggestionDateField.setText(suggestionDate);**

**}**

**}**

**});**

**pack();**

**setLocationRelativeTo(null);**

**displaySuggestions();**

**}**

**private void insertSuggestion() {**

**try (Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "kiran", "bunny")) {**

**String sql = "INSERT INTO nutritionsuggestion (suggestion\_id, user\_id, suggestion, suggestion\_date) VALUES (?, ?, ?, ?)";**

**PreparedStatement statement = conn.prepareStatement(sql);**

**statement.setInt(1, Integer.parseInt(suggestionIdField.getText()));**

**statement.setInt(2, Integer.parseInt(userIdField.getText()));**

**statement.setString(3, suggestionField.getText());**

**statement.setString(4, suggestionDateField.getText());**

**int rowsInserted = statement.executeUpdate();**

**if (rowsInserted > 0) {**

**displaySuggestions();**

**clearFields();**

**JOptionPane.showMessageDialog(this, "Suggestion inserted successfully.");**

**}**

**} catch (SQLException ex) {**

**ex.printStackTrace();**

**JOptionPane.showMessageDialog(this, "Failed to insert suggestion: " + ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);**

**}**

**}**

**private void modifySuggestion() {**

**try (Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "kiran", "bunny")) {**

**String sql = "UPDATE nutritionsuggestion SET user\_id = ?, suggestion = ?, suggestion\_date = ? WHERE suggestion\_id = ?";**

**PreparedStatement statement = conn.prepareStatement(sql);**

**statement.setInt(1, Integer.parseInt(userIdField.getText()));**

**statement.setString(2, suggestionField.getText());**

**statement.setString(3, suggestionDateField.getText());**

**statement.setInt(4, Integer.parseInt(suggestionIdField.getText()));**

**int rowsUpdated = statement.executeUpdate();**

**if (rowsUpdated > 0) {**

**displaySuggestions();**

**clearFields();**

**JOptionPane.showMessageDialog(this, "Suggestion modified successfully.");**

**}**

**} catch (SQLException ex) {**

**ex.printStackTrace();**

**JOptionPane.showMessageDialog(this, "Failed to modify suggestion: " + ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);**

**}**

**}**

**private void deleteSuggestion() {**

**try (Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "kiran", "bunny")) {**

**String sql = "DELETE FROM nutritionsuggestion WHERE suggestion\_id = ?";**

**PreparedStatement statement = conn.prepareStatement(sql);**

**statement.setInt(1, Integer.parseInt(suggestionIdField.getText()));**

**int rowsDeleted = statement.executeUpdate();**

**if (rowsDeleted > 0) {**

**displaySuggestions();**

**clearFields();**

**JOptionPane.showMessageDialog(this, "Suggestion deleted successfully.");**

**}**

**} catch (SQLException ex) {**

**ex.printStackTrace();**

**JOptionPane.showMessageDialog(this, "Failed to delete suggestion: " + ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);**

**}**

**}**

**private void displaySuggestions() {**

**try (Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "kiran", "bunny")) {**

**String sql = "SELECT suggestion\_id, user\_id, suggestion, suggestion\_date FROM nutritionsuggestion";**

**Statement statement = conn.createStatement();**

**ResultSet resultSet = statement.executeQuery(sql);**

**displayTable.setModel(buildTableModel(resultSet));**

**} catch (SQLException ex) {**

**ex.printStackTrace();**

**JOptionPane.showMessageDialog(this, "Failed to fetch suggestion data: " + ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);**

**}**

**}**

**private void clearFields() {**

**suggestionIdField.setText("");**

**userIdField.setText("");**

**suggestionField.setText("");**

**suggestionDateField.setText("");**

**}**

**private DefaultTableModel buildTableModel(ResultSet resultSet) throws SQLException {**

**ResultSetMetaData metaData = resultSet.getMetaData();**

**// Get column names**

**int columnCount = metaData.getColumnCount();**

**String[] columnNames = new String[columnCount];**

**for (int i = 0; i < columnCount; i++) {**

**columnNames[i] = metaData.getColumnLabel(i + 1);**

**}**

**// Get data rows**

**Object[][] data = new Object[100][columnCount];**

**int rowCount = 0;**

**while (resultSet.next()) {**

**for (int i = 0; i < columnCount; i++) {**

**data[rowCount][i] = resultSet.getObject(i + 1);**

**}**

**rowCount++;**

**}**

**return new DefaultTableModel(data, columnNames);**

**}**

**public static void main(String[] args) {**

**SwingUtilities.invokeLater(new Runnable() {**

**public void run() {**

**new NutritionSuggestion();**

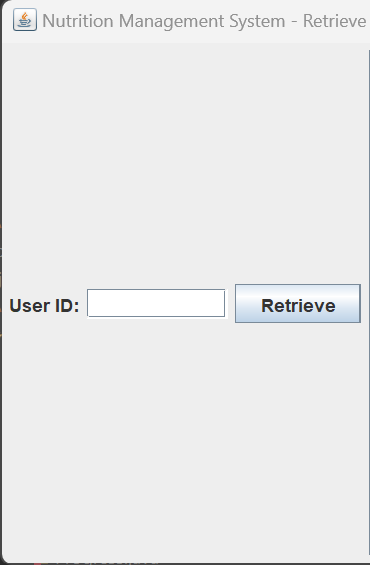
**}**

**});**

**}**

**}**

**Retrieve Page:**

****

**Code:**

**import javax.swing.\*;**

**import java.awt.\*;**

**import java.awt.event.ActionEvent;**

**import java.awt.event.ActionListener;**

**import java.sql.\*;**

**public class RetrieveProgram extends JFrame {**

**private static final long serialVersionUID = 1L;**

**private static final String JDBC\_URL = "jdbc:oracle:thin:@localhost:1521:xe";**

**private static final String USERNAME = "kiran";**

**private static final String PASSWORD = "bunny";**

**private JLabel userIdLabel;**

**private JTextField userIdField;**

**private JButton retrieveButton;**

**private JTextArea healthConditionArea;**

**private JTextArea suggestionArea;**

**public RetrieveProgram() {**

**setTitle("Nutrition Management System - Retrieve");**

**setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);**

**setLayout(new FlowLayout());**

**// Create form controls**

**userIdLabel = new JLabel("User ID:");**

**userIdField = new JTextField(10);**

**retrieveButton = new JButton("Retrieve");**

**healthConditionArea = new JTextArea(20, 50);**

**suggestionArea = new JTextArea(20, 50);**

**healthConditionArea.setEditable(false);**

**suggestionArea.setEditable(false);**

**retrieveButton.addActionListener(new ActionListener() {**

**@Override**

**public void actionPerformed(ActionEvent e) {**

**int userId = Integer.parseInt(userIdField.getText());**

**retrieveData(userId);**

**}**

**});**

**add(userIdLabel);**

**add(userIdField);**

**add(retrieveButton);**

**//add(new JScrollPane(healthConditionArea,));**

**//add(new JScrollPane(suggestionArea));**

**add(new JScrollPane(healthConditionArea, JScrollPane.VERTICAL\_SCROLLBAR\_ALWAYS, JScrollPane.HORIZONTAL\_SCROLLBAR\_ALWAYS));**

**add(new JScrollPane(suggestionArea, JScrollPane.VERTICAL\_SCROLLBAR\_ALWAYS, JScrollPane.HORIZONTAL\_SCROLLBAR\_ALWAYS));**

**pack();**

**setLocationRelativeTo(null); // Center the frame on the screen**

**}**

**private void retrieveData(int userId) {**

**try {**

**Connection connection = DriverManager.getConnection(JDBC\_URL, USERNAME, PASSWORD);**

**String userQuery = "SELECT \* FROM Users WHERE user\_id = ?";**

**PreparedStatement userStatement = connection.prepareStatement(userQuery);**

**userStatement.setInt(1, userId);**

**ResultSet userResultSet = userStatement.executeQuery();**

**if (userResultSet.next()) {**

**String userName = userResultSet.getString("name");**

**int age = userResultSet.getInt("age");**

**String gender = userResultSet.getString("gender");**

**int height = userResultSet.getInt("height");**

**int weight = userResultSet.getInt("weight");**

**String activityLevel = userResultSet.getString("activity\_level");**

**healthConditionArea.setText("User Information:\n");**

**healthConditionArea.append("Name: " + userName + "\n");**

**healthConditionArea.append("Age: " + age + "\n");**

**healthConditionArea.append("Gender: " + gender + "\n");**

**healthConditionArea.append("Height: " + height + "\n");**

**healthConditionArea.append("Weight: " + weight + "\n");**

**healthConditionArea.append("Activity Level: " + activityLevel + "\n");**

**String healthConditionQuery = "SELECT \* FROM Healthcondition WHERE user\_id = ?";**

**PreparedStatement healthConditionStatement = connection.prepareStatement(healthConditionQuery);**

**healthConditionStatement.setInt(1, userId);**

**ResultSet healthConditionResultSet = healthConditionStatement.executeQuery();**

**if (healthConditionResultSet.next()) {**

**int healthConditionId = healthConditionResultSet.getInt("healthcondition\_id");**

**String conditionName = healthConditionResultSet.getString("condition\_name");**

**String conditionDescription = healthConditionResultSet.getString("condition\_description");**

**healthConditionArea.append("\nHealth Condition:\n");**

**healthConditionArea.append("Condition ID: " + healthConditionId + "\n");**

**healthConditionArea.append("Condition Name: " + conditionName + "\n");**

**healthConditionArea.append("Condition Description: " + conditionDescription + "\n");**

**} else {**

**healthConditionArea.append("\nNo health condition found for the user.");**

**}**

**String nutritionSuggestionQuery = "SELECT \* FROM NutritionSuggestion WHERE user\_id = ?";**

**PreparedStatement nutritionSuggestionStatement = connection.prepareStatement(nutritionSuggestionQuery);**

**nutritionSuggestionStatement.setInt(1, userId);**

**ResultSet nutritionSuggestionResultSet = nutritionSuggestionStatement.executeQuery();**

**if (nutritionSuggestionResultSet.next()) {**

**int suggestionId = nutritionSuggestionResultSet.getInt("suggestion\_id");**

**String suggestion = nutritionSuggestionResultSet.getString("suggestion");**

**String suggestionDate = nutritionSuggestionResultSet.getString("suggestion\_date");**

**suggestionArea.setText("Nutrition Suggestion:\n");**

**suggestionArea.append("Suggestion ID: " + suggestionId + "\n");**

**suggestionArea.append("Suggestion: " + suggestion + "\n");**

**suggestionArea.append("Suggestion Date: " + suggestionDate + "\n");**

**} else {**

**suggestionArea.setText("No nutrition suggestion found for the user.");**

**}**

**else {**

**healthConditionArea.setText("No user found with the specified user ID.");**

**suggestionArea.setText("");**

**}**

**connection.close();**

**} catch (SQLException e) {**

**e.printStackTrace();**

**}**

**}**

**public static void main(String[] args) {**

**SwingUtilities.invokeLater(new Runnable() {**

**@Override**

**public void run() {**

**new RetrieveProgram().setVisible(true);**

**}**

**});**

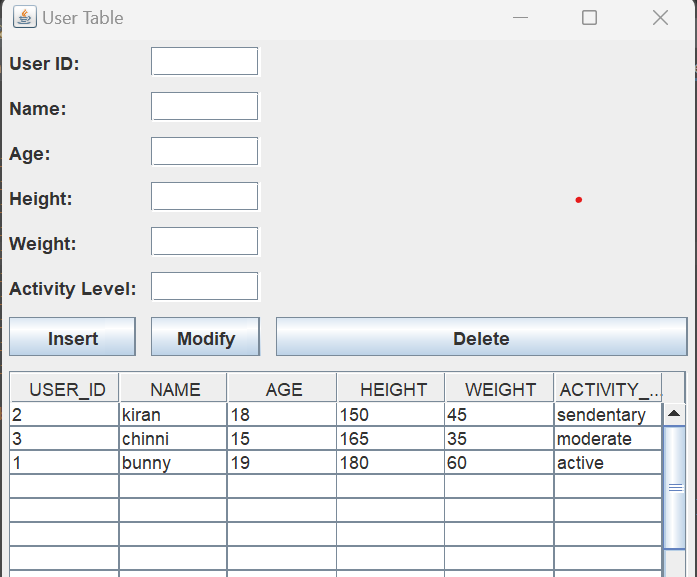
**}**

**}**

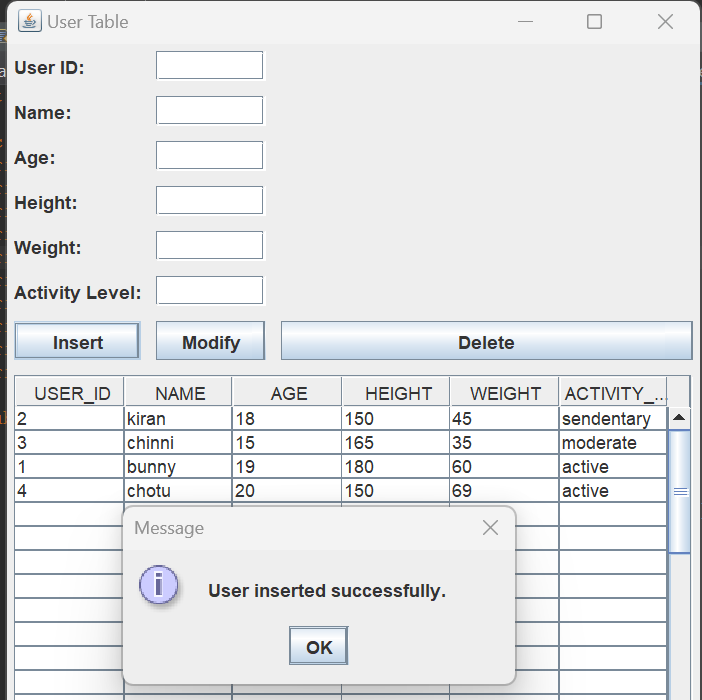
**TESTING**

**User Table:**

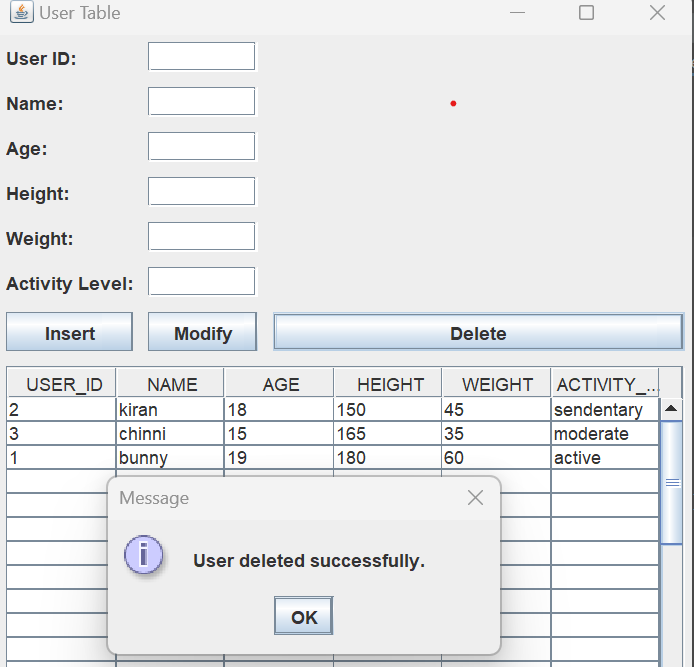
Before insertion into table :



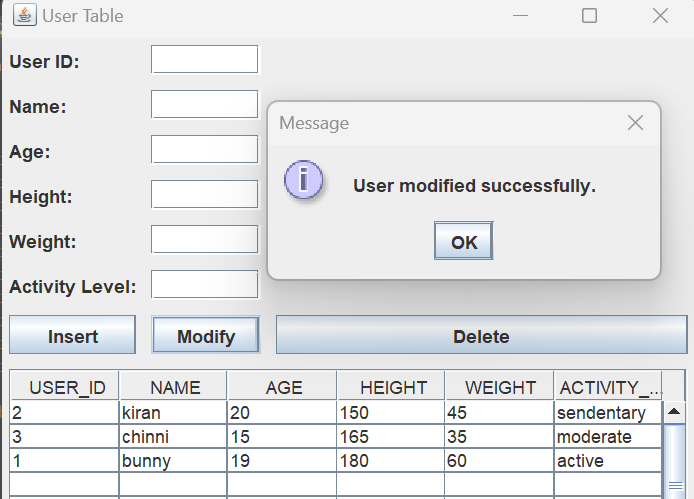
After Insertion:



Deletion:

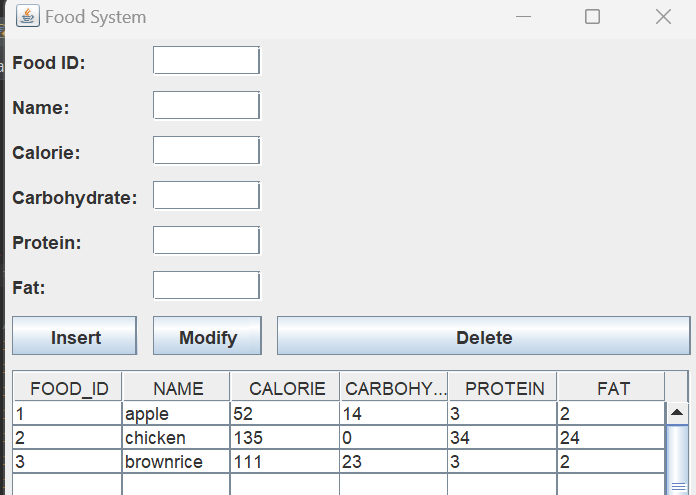


Modifying:

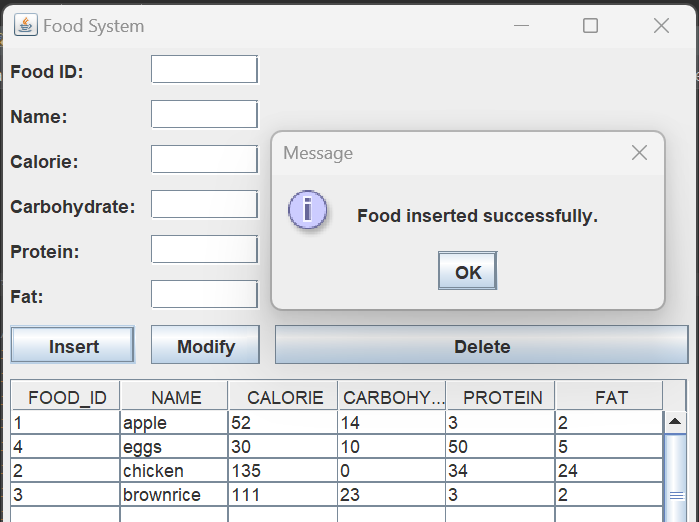


**Food Table:**

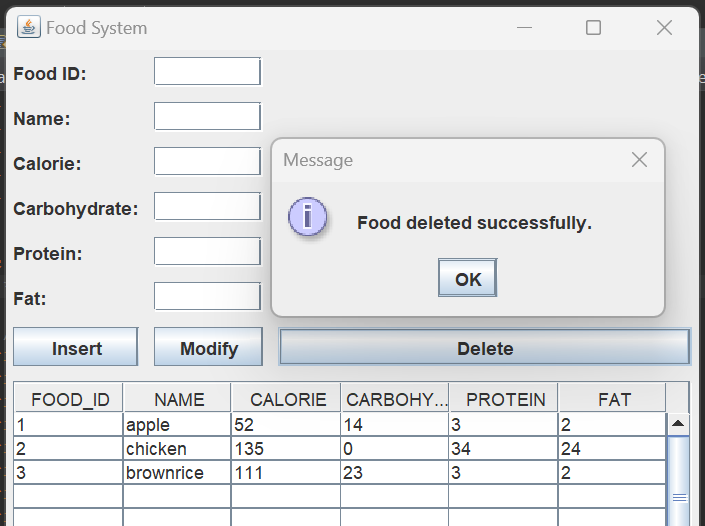
**Before Insertion:**

****

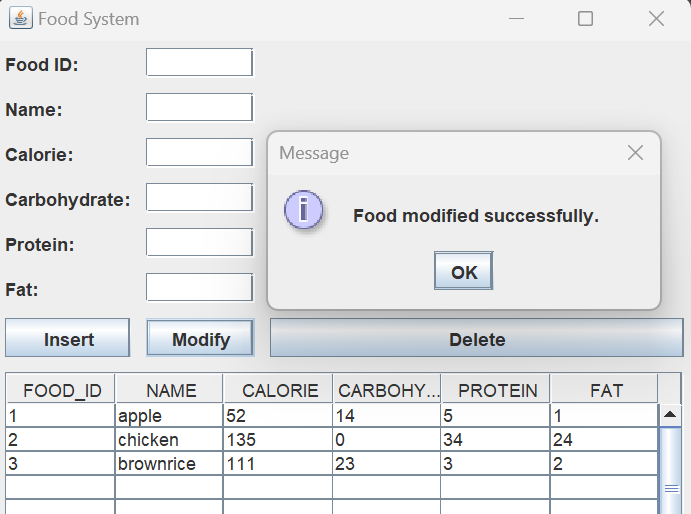
After Insertion:



Deletion:

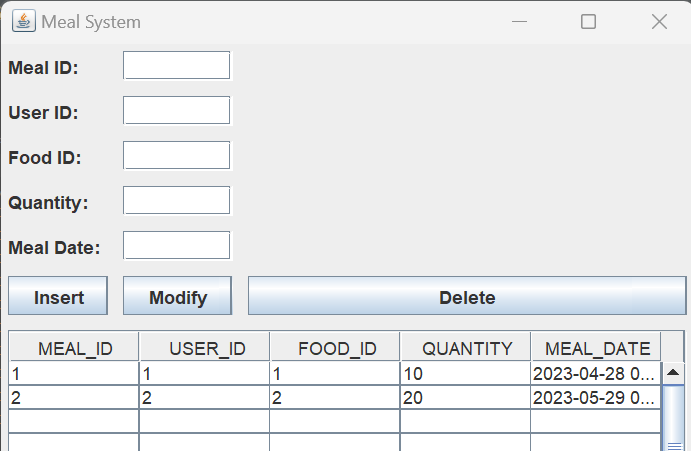


Modifying:

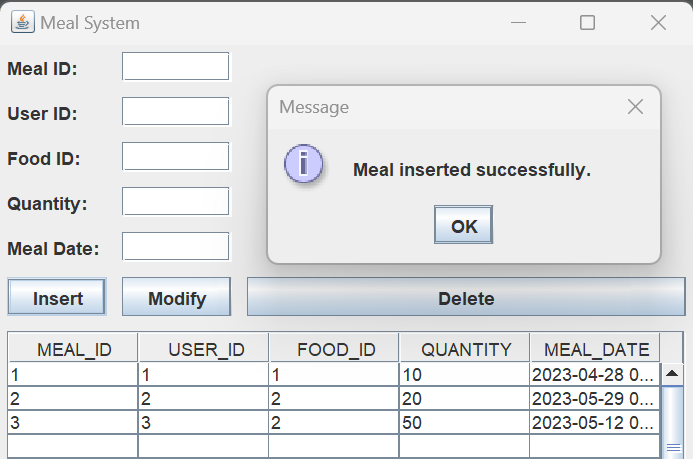


**Meal Table:**

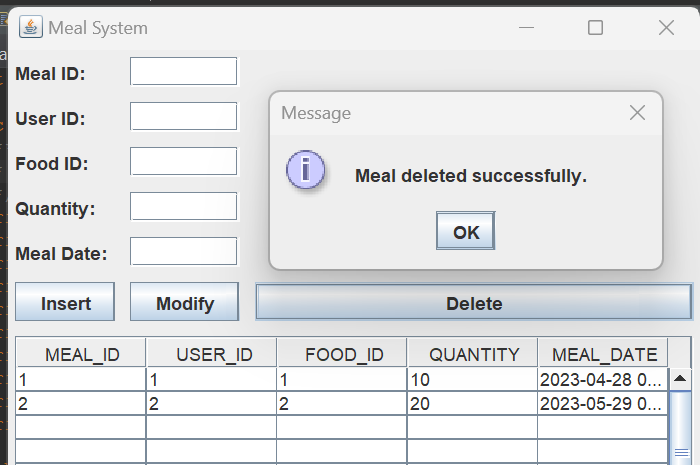
**Before Inserting:**

****

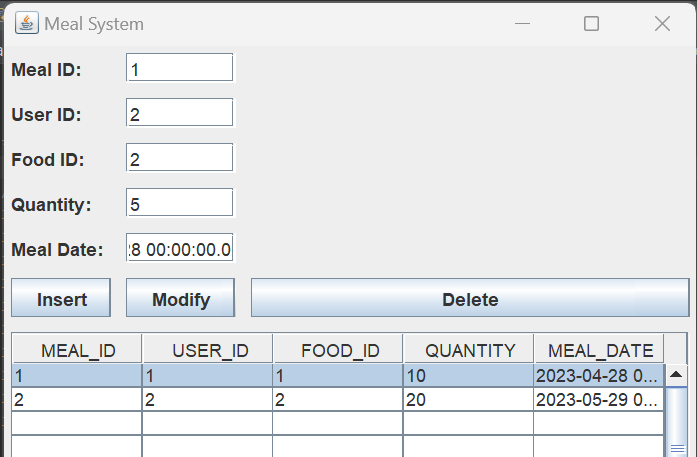
After Insertion:



Deletion:

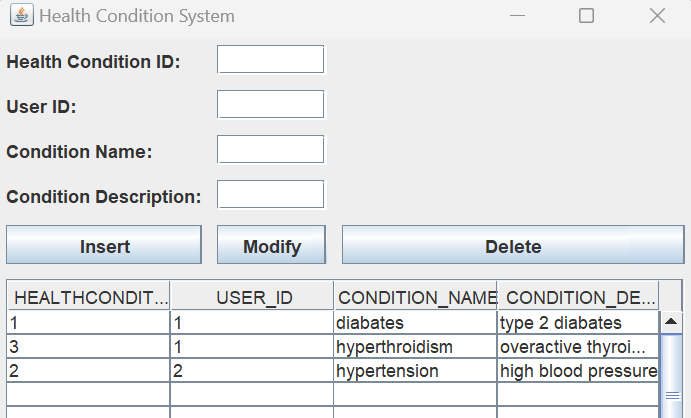


Modifying:

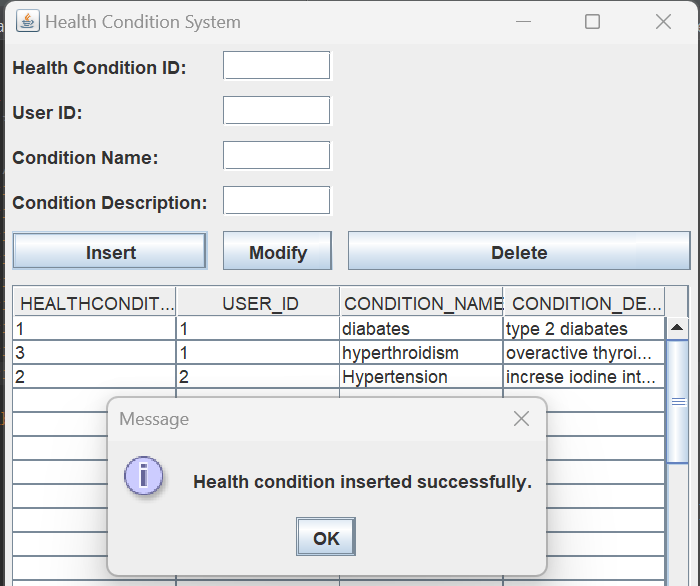


**HealthCondition Table:**

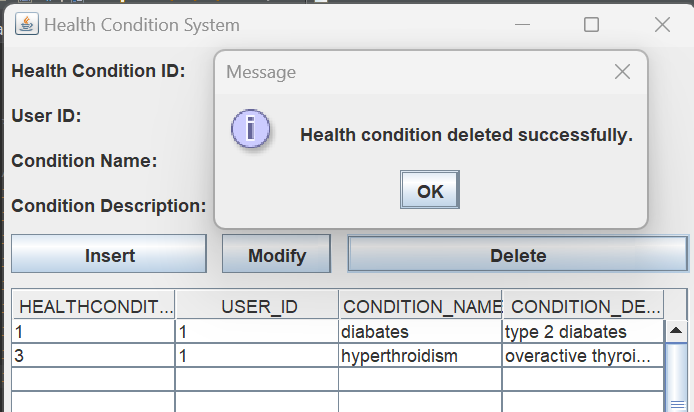
**Before Insertion:**

****

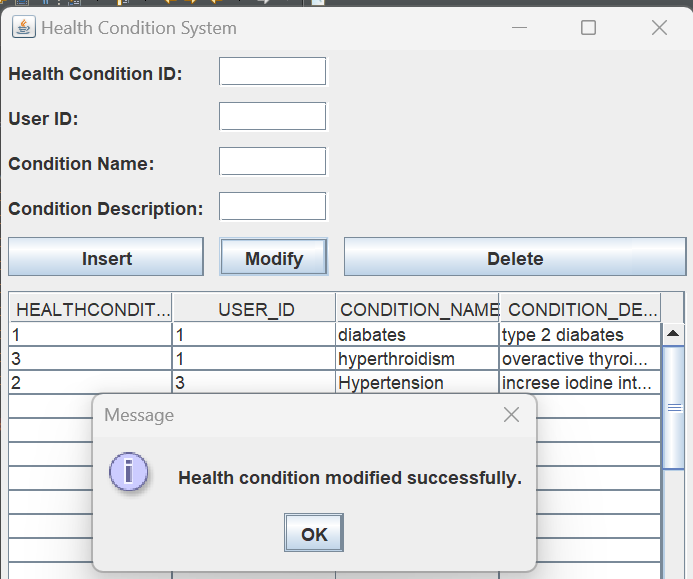
**After Insertion:**

****

**Deletion:**

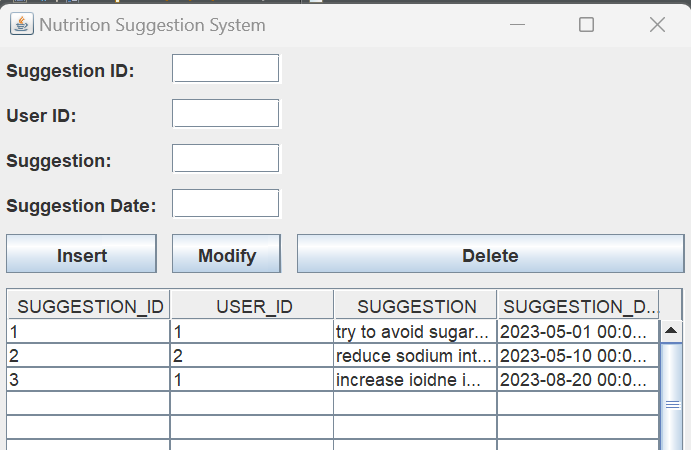
****

**Modifying:**

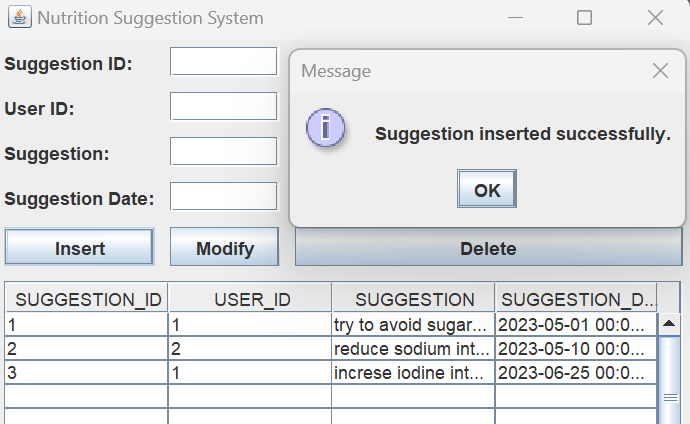
****

**NutritionSuggestion Table:**

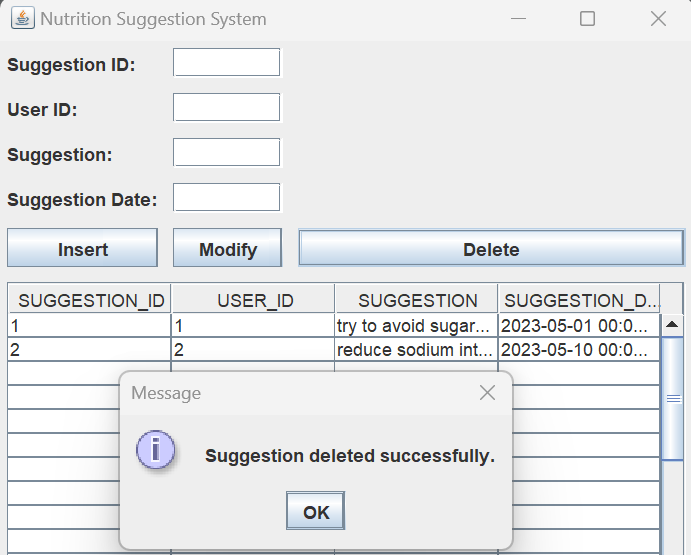
**Before Insertion:**



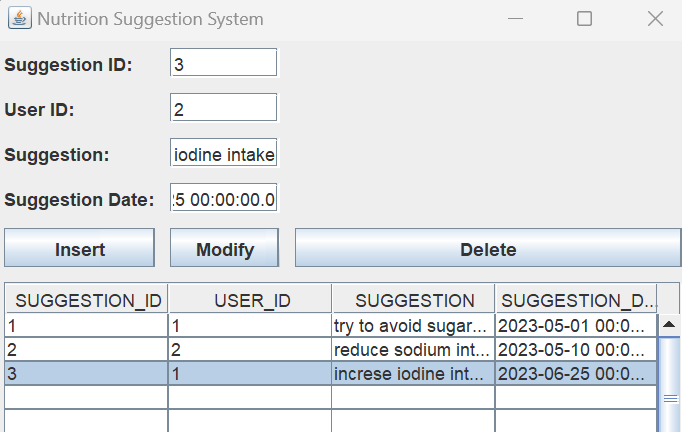
**After Insertion:**



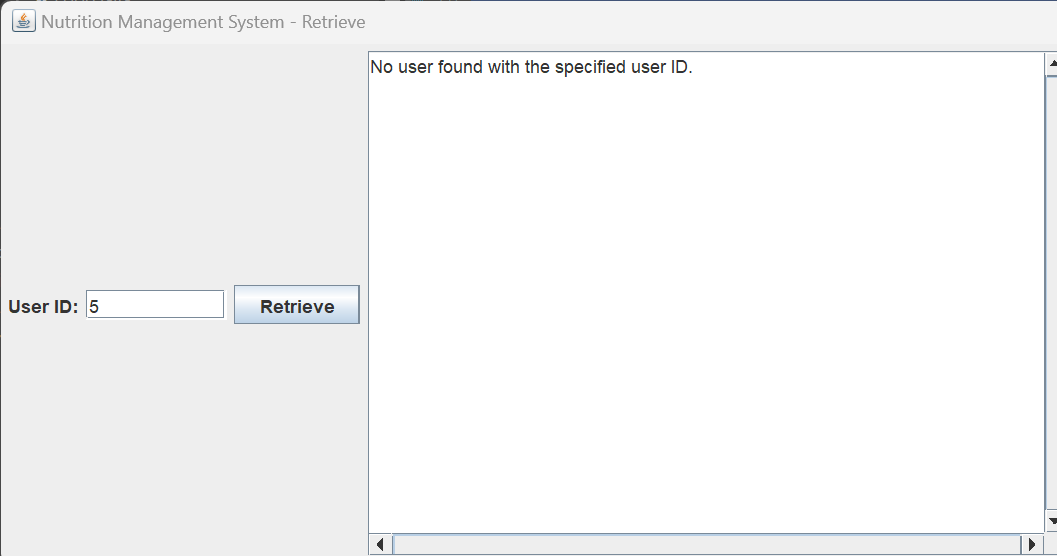
**Deletion:**

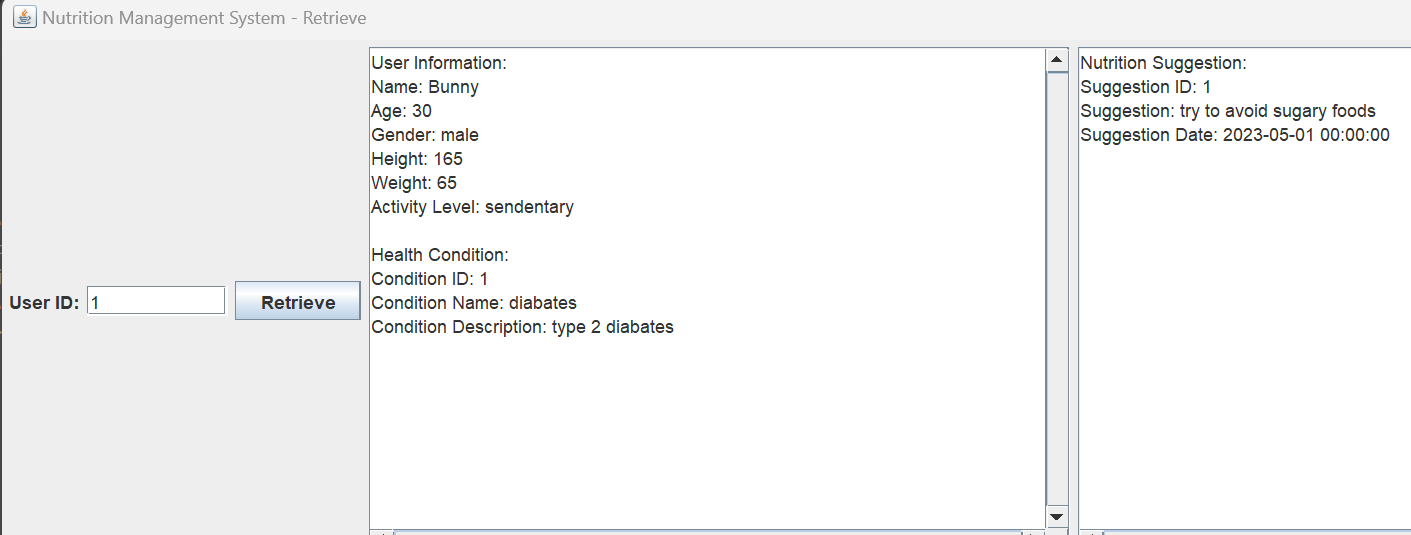
****

**Modifying:**

****

**Retrieve Details:**

****

****

**RESULT**

I have successfully completed my DBMS project ‘FOOD AND NUTRITION SUGGESTION SYSTEM’

**DISCUSSION AND FUTURE WORK**

**The effectiveness of a Food and Nutrition Suggestion System relies on its ability to provide personalized recommendations based on an individual's unique dietary needs, preferences, and health goals. Future work should focus on incorporating more comprehensive user profiles, including factors such as age, gender, body composition, activity level, and medical conditions. This information can be leveraged to offer tailored suggestions that optimize nutrition and overall well-being.**

**REFERENCES**

● https://docs.oracle.com/javase/7/docs/api/

● <https://www.javatpoint.com/java-swing>

● <https://stackoverflow.com/>

**Git-Hub LINK:**

https://github.com/kiranbunny/FOOD-AND-NUTRITION-SUGGESTION-SYSTEM