الجامعة الافتراضية السورية
SYRIAN VIRTUAL UNIVERSITY

حل وظيفة مقرر قاعدة البيانات 1

قاعدة بيانات لمدرسة خاصة

الصف	الطلاب
C3	Abubakr_203648
C3	Aya_175481
C1	Omaran_158391

	الطلب	الرمز	العلامة
1	مخطط الكيانات العلائقي	ERD	25
2	تصميم القاعدة باستخدام Oracle + C#	DB+C#	20
3	ادخال البيانات	Insert + C#	10
4	اكتب جملة Select تعيد اسماء الطلاب الذين حصلوا على الدرجة الكاملة في مادة الرياضيات ولم ينجحوا في مقرر اللغة العربية في العام الدراسي 2023/2024.	Select1	15
5	اكتب جملة Select تعيد أسماء المقررات التي درست من قبل أكثر من مدرس، وأسماء هؤلاء المدرسين.	Select2	15
6	اكتب جملة Select تعيد أسماء الطلاب الأوائل في كل مقرر واسم المقرر.	Select3	15
	المجموع		100

Contents

1. مخطط ERD

1.1. الكيانات

1.1.1. الموظف:(Employee)

- **الوصف :**تخزن المعلومات عن جميع موظفي المدرسة، بما في ذلك المعلمين والموظفين الإداريين.
- **الأعمدة:**

Employee	
PK	employee_id
	first_name last_name date_of_birth phone_number address hire_date

  - employee_id: INT) مفتاح أساسي معرف فريد لكل موظف.
  - first_name: (NVARCHAR2(255)) اسم الموظف الأول.
  - last_name: (NVARCHAR2(255)) اسم الموظف الأخير.
  - date_of_birth: (DATE) تاريخ ميلاد الموظف.
  - phone_number: (VARCHAR2(20)) رقم هاتف الموظف.
  - address: (NVARCHAR2(255)) عنوان سكن الموظف.
  - hire_date: (DATE) تاريخ تعيين الموظف.

1.1.2. الدور:(Role)

- **الوصف :**يحدد الأدوار المختلفة التي يمكن أن يشغلها الأفراد داخل المدرسة (مثل: المعلم، المدير، أمين السر).
- **الأعمدة:**

Role	
PK	role_id
	role_name

  - role_id: INT، مفتاح أساسي معرف فريد لكل دور.
  - role_name: (NVARCHAR2(255)) اسم الدور.

1.1.3. دور الموظف:(EmployeeRole)

- **الوصف :**يربط الموظفين بأدوارهم داخل المدرسة. يمكن للموظف أن يشغل عدة أدوار.
- **الأعمدة:**

EmployeeRole	
PK	employee_role_id
	employee_id role_id

  - employee_role_id: INT، مفتاح أساسي معرف فريد لكل علاقة بين موظف ودور.
  - employee_id: INT، مفتاح خارجي، يشير إلى الموظف
  - role_id: INT، مفتاح خارجي، يشير إلى الدور

1.1.4. الصف:(Grade)

- **الوصف :**يمثل مستويات الصفوف الدراسية المختلفة في المدرسة (مثل: الصف الأول، الصف الثاني، ... الصف الثاني عشر).

Grade	
PK	grade_id
	grade_name

- **الأعمدة:**
  - grade_id: INT، مفتاح أساسي معرف فريد لكل مستوى صف.
  - grade_name: (NVARCHAR2(50)) اسم مستوى الصف.

1.1.5. المقرر:(Class)

- **الوصف :**يمثل مجموعة محددة من المواد الدراسية ومستوى معين مثل: الرياضيات 101، التاريخ 101

Class	
PK	class_id
	class_name grade_id

- **الأعمدة:**
  - class_id: INT، مفتاح أساسي معرف فريد لكل مقرر.
  - class_name: (NVARCHAR2(255)) اسم المقرر.
  - grade_id: INT، مفتاح خارجي، يشير إلى الصف مستوى الصف للمقرر.

1.1.6. الشعبة:(Section)

Section	
PK	section_id
	section_name class_id

- **الوصف :**يمثل قسم من الطلاب داخل مقرر محدد.
- **الأعمدة:**
  - section_id: INT، مفتاح أساسي معرف فريد لكل شعبة.
  - section_name: (NVARCHAR2(50)) اسم الشعبة (مثل: "الشعبة أ").
  - class_id: INT، مفتاح خارجي، يشير إلى المقرر المقرر الذي تنتمي إليه هذه الشعبة.

1.1.7. غرفة الصف:(Classroom)

- **الوصف :**تخزن معلومات عن غرف الصفوف المادية في المدرسة.

Classroom	
PK	classroom_id
	room_number capacity

- **الأعمدة:**
  - classroom_id: INT، مفتاح أساسي (معرف فريد لكل غرفة صف.

- room_number: (VARCHAR2(20)) رقم الغرفة أو معرفها.
- capacity: INT السعة القصوى للطلاب في غرفة الصف.

1.1.8. المادة :(Subject)

- **الوصف :**يمثل المواد الدراسية الأكاديمية المختلفة التي يتم تدريسها (مثل: الرياضيات، التاريخ، العلوم).

Subject	
PK	subject_id
	subject_name

- **الأعمدة:**
  - subject_id: INT، مفتاح أساسي معرف فريد لكل مادة.
  - subject_name: (NVARCHAR2(255)) اسم المادة.

1.1.9. جلسة المقرر:(ClassSession)

- **الوصف :**يمثل حالة محددة من المقرر الذي يتم تقديمه في فصل دراسي معين، يدرسه معلم معين.
- **الأعمدة:**
  - session_id: INT، مفتاح أساسي معرف فريد لكل جلسة مقرر.
  - subject_id: INT، مفتاح خارجي، يشير إلى المادة مادة جلسة المقرر.
  - teacher_id: INT، مفتاح خارجي، يشير إلى الموظف الموظف الذي يدرّس هذه الجلسة.
  - semester_id: INT، مفتاح خارجي، يشير إلى الفصل الدراسي الفصل الدراسي الذي تُعقد فيه هذه الجلسة.
  - class_id: INT، مفتاح خارجي، يشير إلى المقرر المقرر الذي يُقدم في هذه الجلسة.

ClassSession	
PK	session_id
	subject_id teacher_id semester_id class_id

1.1.10. الطالب:(Student)

- **الوصف :**تخزن المعلومات عن الطلاب المسجلين في المدرسة.
- **الأعمدة:**

Student	
PK	student_id
	first_name last_name date_of_birth gender address parent_name parent_contact enrollment_date

  - student_id: INT، مفتاح أساسي معرف فريد لكل طالب.
  - first_name: (NVARCHAR2(255)) اسم الطالب الأول.
  - last_name: (NVARCHAR2(255)) اسم الطالب الأخير.
  - date_of_birth: (DATE) تاريخ ميلاد الطالب.
  - gender: (NVARCHAR2(10)) جنس الطالب.
  - address: (NVARCHAR2(255)) عنوان سكن الطالب.
  - parent_name: (NVARCHAR2(255)) اسم والد/ولي أمر الطالب.
  - parent_contact: (VARCHAR2(20)) رقم اتصال والد/ولي أمر الطالب.
  - enrollment_date: (DATE) تاريخ تسجيل الطالب في المدرسة.

1.1.11. الفصل الدراسي:(Semester)

Semester	
PK	semester_id
	semester_name school_year

- **الوصف :**يمثل فصل دراسي أكاديمي (مثل: خريف 2024، ربيع 2024).
- **الأعمدة:**
  - semester_id: INT، مفتاح أساسي معرف فريد لكل فصل دراسي.
  - semester_name: (NVARCHAR2(50)) اسم الفصل الدراسي (مثل: "خريف", "ربيع").

- school_year: INT) العام الدراسي الذي ينتمي إليه الفصل الدراسي.

1.1.12. الجدول:(Schedule)

- **الوصف :**يحدد جدولًا زمنيًا للمقررات، يحدد متى وأين تحدث.
- **الأعمدة:**

Schedule	
PK	schedule_id
	class_id classroom_id session_id day_of_week start_time end_time

- schedule_id: INT، مفتاح أساسي معرف فريد لكل إدخال في الجدول.
- class_id: INT، مفتاح خارجي، يشير إلى المقرر المقرر الذي ينتمي إليه إدخال الجدول.
- classroom_id: INT، مفتاح خارجي، يشير إلى غرفة الصف مكان المقرر.
- session_id: INT، مفتاح خارجي، يشير إلى جلسة المقرر الجلسة المحددة التي يرتبط بها هذا الجدول.
- day_of_week: (NVARCHAR2(10)) يوم الأسبوع الذي تم جدولة المقرر فيه (مثل: "الاثنين", "الثلاثاء"...).
- start_time: (DATE)وقت بدء المقرر.
- end_time: (DATE)وقت انتهاء المقرر.

1.1.13. التسجيل:(Enrollment)

- **الوصف :**يسجل تسجيل الطلاب في جلسات مقررات معينة لفترة دراسية معينة.
- **الأعمدة:**
  - enrollment_id: INT، مفتاح أساسي معرف فريد لكل سجل تسجيل.
  - student_id: INT، مفتاح خارجي، يشير إلى الطالب الطالب المسجل.

- session_id: INT، مفتاح خارجي، يشير إلى جلسة المقرر جلسة المقرر التي مسجل فيها الطالب.
- semester_id: INT، مفتاح خارجي، يشير إلى الفصل الدراسي فصل دراسي التسجيل.

Enrollment	
PK	enrollment_id
	student_id session_id semester_id

1.1.14. عنصر الدرجة:(GradeComponent)

- **الوصف :**يحدد المكونات المختلفة التي تساهم في الدرجة النهائية للطالب (مثل: الواجب المنزلي، الاختبارات، الامتحانات).
- **الأعمدة:**
  - component_id: INT، مفتاح أساسي معرف فريد لكل عنصر درجة.
  - component_name: (NVARCHAR2(255))اسم العنصر (مثل: "الواجب المنزلي 1", "امتحان منتصف الفصل").

GradeComponent	
PK	component_id
	component_name

1.1.15. درجة الطالب:(StudentGrade)

- **الوصف :**تخزن الدرجات التي حققها الطلاب لكل عنصر درجة داخل التسجيل.
- **الأعمدة:**
  - grade_id: INT، مفتاح أساسي معرف فريد لكل إدخال درجة.
  - enrollment_id: INT، مفتاح خارجي، يشير إلى التسجيل التسجيل الذي تنتمي إليه هذه الدرجة.

- component_id: INT، مفتاح خارجي، يشير إلى عنصر الدرجة (عنصر الدرجة التي هي لهذه الدرجة.
- score: (DECIMAL(5,2)) درجة الطالب لهذا العنصر من الدرجة.

StudentGrade	
PK	grade_id
	enrollment_id component_id score

1.1.16. الملاحظة:(Note)

- **الوصف :**تخزن الملاحظات أو التعليقات أو ملاحظات حول أداء الطالب، من قبل أدوار مختلفة (مدير، معلم، ولي أمر).
- **الأعمدة:**

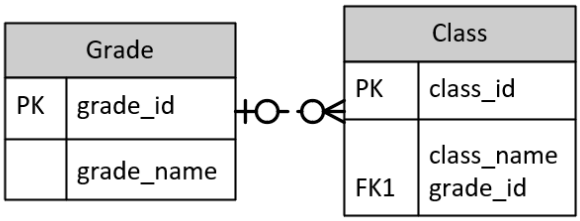
- note_id: INT، مفتاح أساسي معرف فريد لكل إدخال ملاحظة.
- enrollment_id: INT، مفتاح خارجي، يشير إلى التسجيل الذي ترتبط به هذه الملاحظة.
- semester_id: INT، مفتاح خارجي، يشير إلى الفصل الدراسي الفصل الدراسي الذي تم فيه إنشاء الملاحظة.
- author_role: (NVARCHAR2(50)) دور الشخص الذي يقوم بإنشاء الملاحظة (مثل: "مدير", "معلم", "ولي أمر").
- note_text: (NCLOB) محتوى الملاحظة.

Note	
PK	note_id
	enrollment_id semester_id author_role note_text

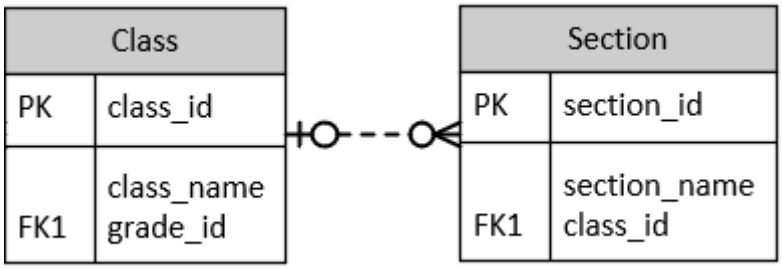
1.2. العلاقات:

1.2.1. واحد إلى متعدد:(1:M)

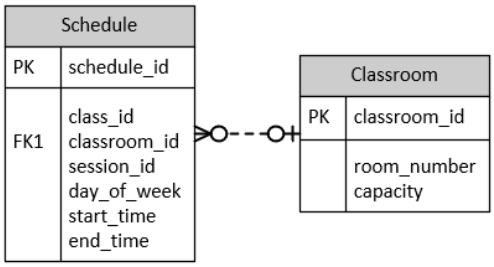
- **المقرر <---(1:M)--- الصف:**
  - **الوصف :**يمكن أن يكون لمستوى صف واحد (مثل: "الصف العاشر") العديد من المقررات الدراسية المرتبطة به (مثل: "الرياضيات 101", "البيولوجيا 101"), ولكن كل مقرر يمكن أن ينتمي فقط إلى مستوى صف واحد.



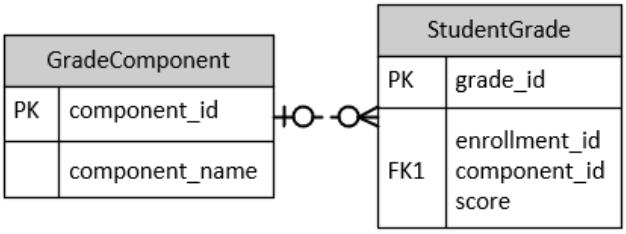
- **الشعبة <---(1:M)--- المقرر:**
  - **الوصف :**يمكن تقسيم المقرر الدراسي (مثل: "الرياضيات 101") إلى عدة شعب ("الشعبة أ", "الشعبة ب") لاستيعاب المزيد من الطلاب، لكن كل شعبة مرتبطة دائمًا بمقرر واحد محدد.



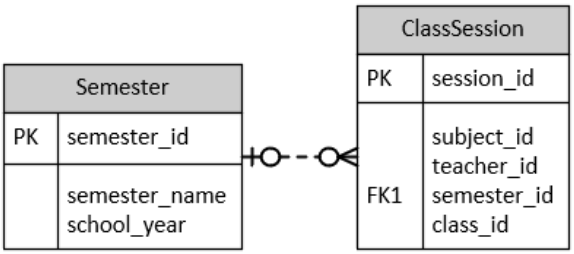
- **الجدول <---(1:M)--- غرفة الصف:**
  - **الوصف :**يمكن استخدام غرفة صف مادية في المدرسة لعدة جلسات مقررات مجدولة طوال الأسبوع (أو اليوم)، لكن كل إدخال في الجدول يشير إلى غرفة صف واحدة.



- **درجة الطالب <---(1:M)--- عنصر الدرجة:**
  - **الوصف :**يمكن استخدام عنصر درجة واحد (مثل: "امتحان منتصف الفصل") لتسجيل درجات لعدة طلاب مسجلين في جلسة مقرر، لكن كل إدخال فردي لدرجة الطالب يتوافق مع عنصر درجة واحد فقط.

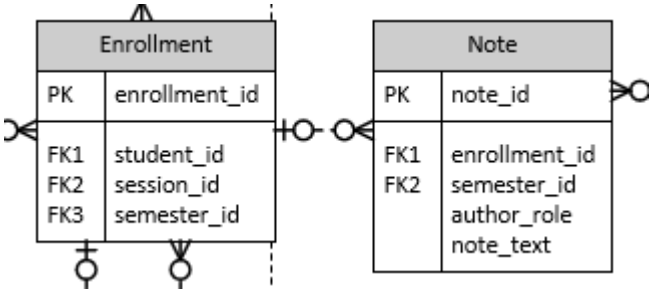


- **جلسة المقرر <---(1:M)--- الفصل الدراسي:**
  - **الوصف :**يمكن أن يكون لفترة دراسية أكاديمية (مثل: "خريف 2024") العديد من جلسات المقررات المعروضة خلال تلك الفترة، لكن جلسة المقرر ستكون دائمًا ضمن فصل دراسي واحد محدد.



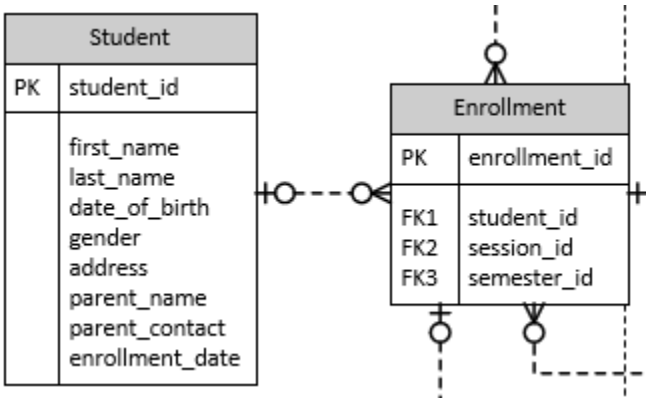
- **الملاحظة <---(1:M)--- الطالب:**

- الوصف: يمكن أن يكون لدى الطالب ملاحظات متعددة مكتوبة عنه طوال رحلته الأكاديمية. يمكن أن تكون هذه الملاحظات من المعلمين أو المرشدين أو المدير. ومع ذلك، ترتبط كل ملاحظة بطالب واحد فقط.



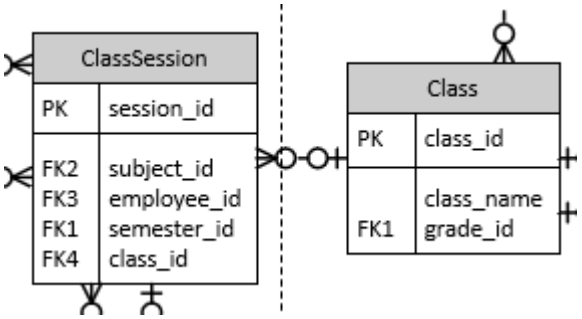
- **التسجيل <---(1:M)--- الطالب:**

- الوصف: يمكن للطالب التسجيل في فصول دراسية متعددة طوال فترة وجوده في المدرسة. يمثل كل سجل تسجيل تسجيل الطالب في جلسة فصل دراسي محددة. ينتمي كل سجل تسجيل إلى طالب واحد فقط.

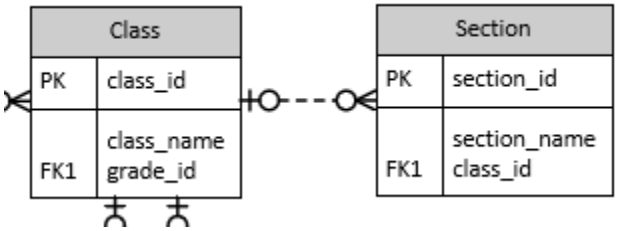


- **جلسة المقرر <---(1:M)--- المقرر:**

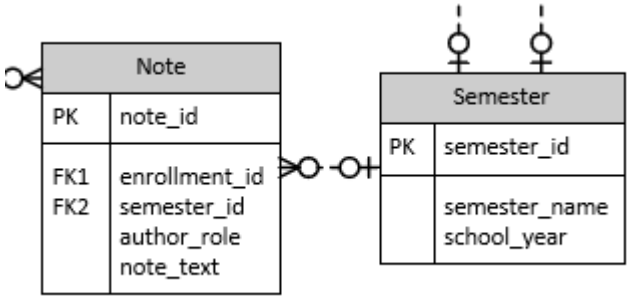
- الوصف: يمكن أن يكون لفصل دراسي محدد، مثل "الرياضيات 101"، جلسات دراسية متعددة معروضة عبر فصول دراسية أو سنوات دراسية مختلفة. ومع ذلك، تمثل كل جلسة فصل دراسي مثيلًا فريدًا لهذا الفصل الدراسي الذي يتم تقديمه وترتبط بفصل دراسي واحد.



- **الشعبة <---(1:M)--- المقرر:**
  - الوصف: يمكن تقسيم الفصل الدراسي، خاصةً الذي يحتوي على عدد كبير من الطلاب، إلى شعب متعددة (مثل "الشعبة أ" و "الشعبة ب") لاستيعاب جميع الطلاب. ترتبط كل شعبة دائمًا بفصل دراسي محدد واحد فقط.

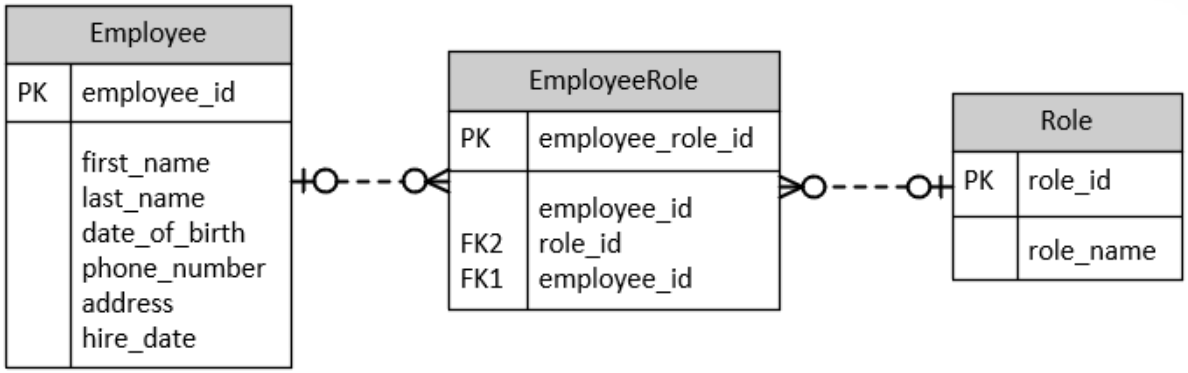


- **الملاحظة <---(1:M)--- الفصل الدراسي:**
  - الوصف: يمكن أن يكون للفصل الدراسي ملاحظات متعددة مرتبطة به، والتي تعكس عادةً الملاحظات والتقييمات التي تم إجراؤها خلال تلك الفترة المحددة. تتعلق كل ملاحظة بفصل دراسي واحد فقط.

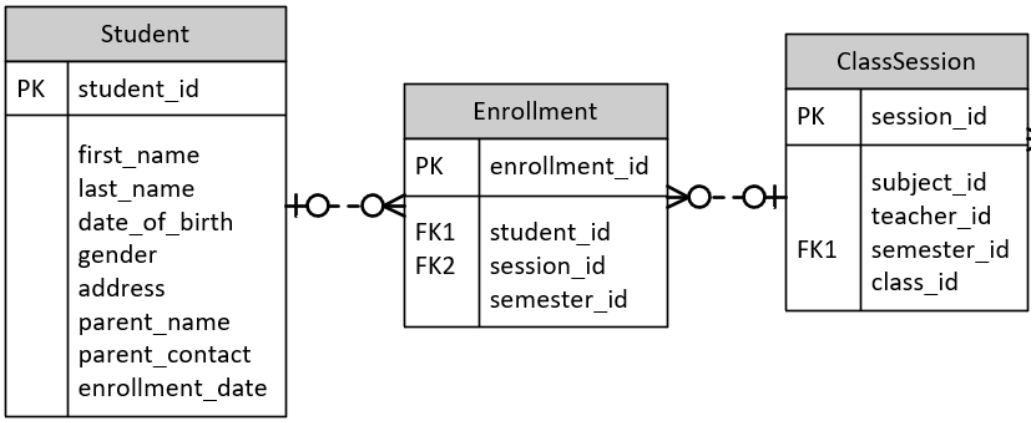


1.2.2. متعدد إلى متعدد:(M:N)

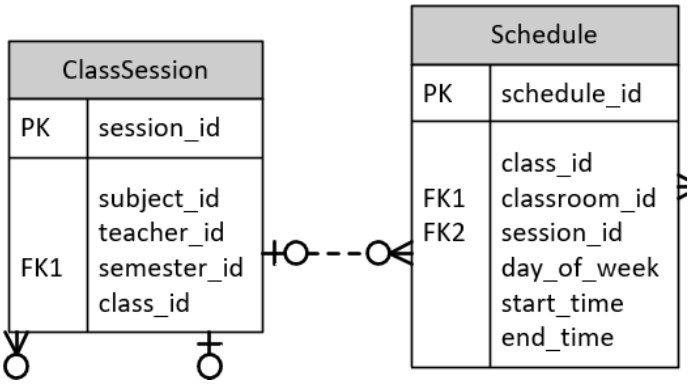
- **الموظف <---(M:N)---الدور>:**
  - **الوصف :**يمكن للموظف في المدرسة أن يشغل عدة أدوار (مثل: يمكن لشخص أن يكون "معلمًا" و"مستشارًا" في نفس الوقت)، ودور محدد ("معلم") يمكن تعيينه لعدة موظفين مختلفين.
  - **جدول الصلة :**يعمل جدول EmployeeRole كجسر بين هذين الجدولين، حيث يخزن العلاقات بين الموظفين والأدوار المعينة لهم.



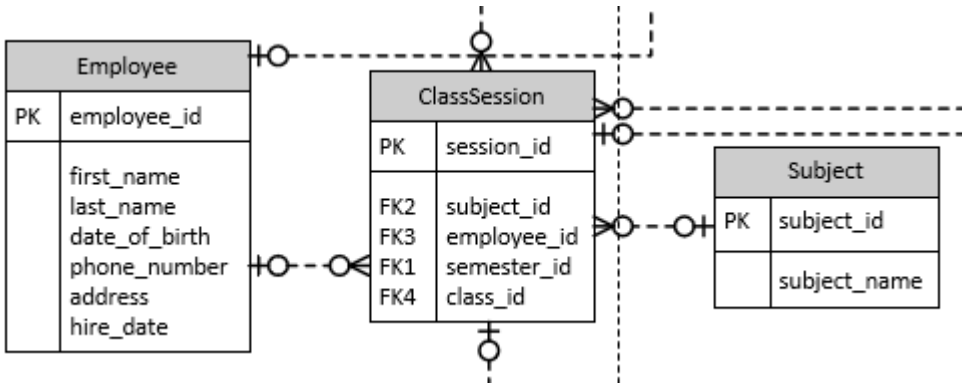
- **الطالب <---(M:N)---جلسة المقرر>:**
  - **الوصف :**يمكن للطالب أن يسجل في العديد من جلسات المقررات خلال فصل دراسي معين، ويمكن أن يكون لكل جلسة مقرر العديد من الطلاب الحاضرين.
  - **جدول الصلة :**يعمل جدول Enrollment كنقطة اتصال، حيث يتتبع الطلاب المسجلين في جلسات المقررات وأي فصل دراسي.



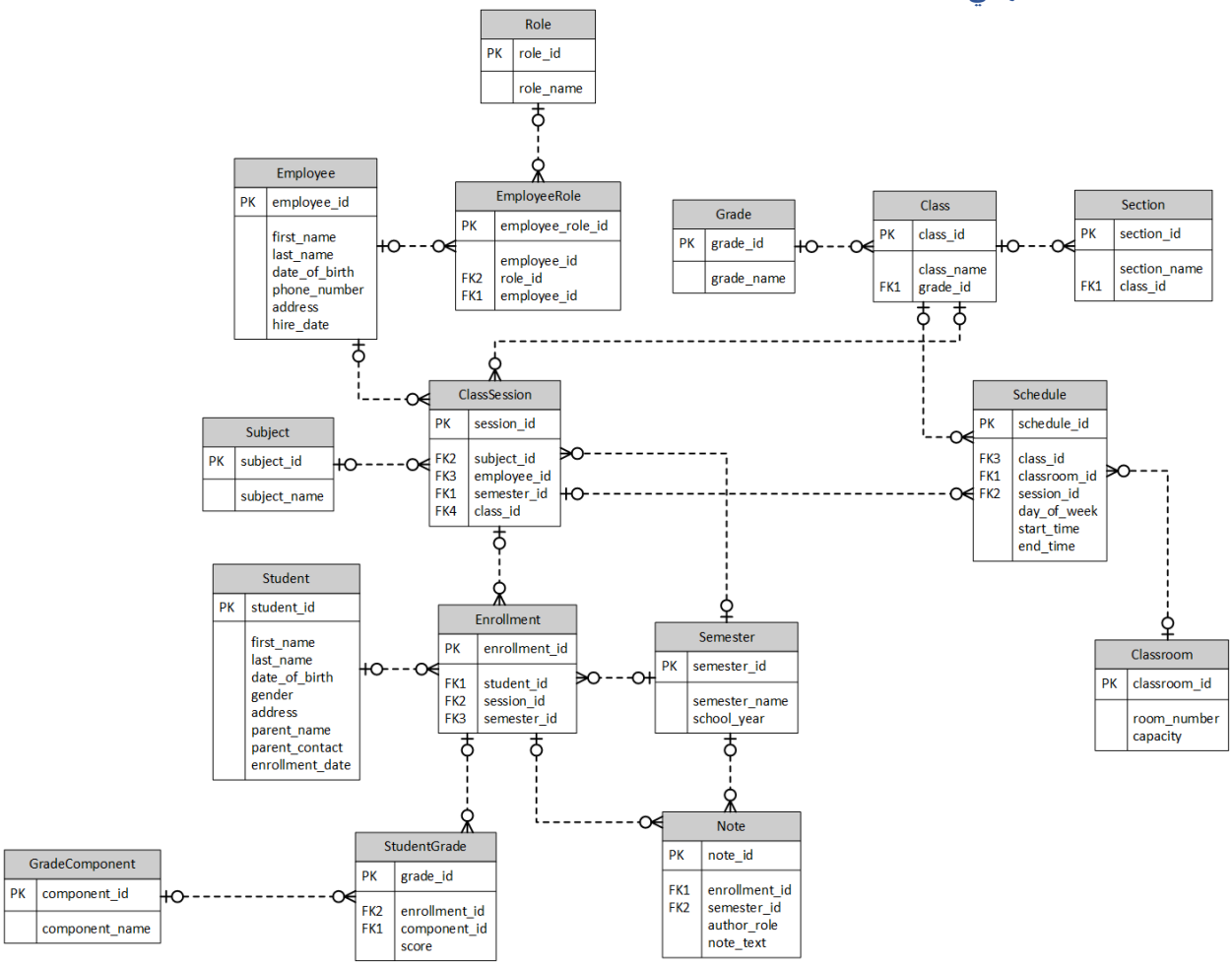
- **جلسة المقرر <---(M:N)---> الجدول:**
  - **الوصف :**قد يكون لجلسة مقرر واحدة عدة مرات مجدولة خلال الأسبوع (مثل: جلسة "الاثنين الأربعاء الجمعة" الساعة 10:00 صباحًا وجلسة "الثلاثاء الخميس" الساعة 2:00 مساءً)، ويمكن استخدام فتحة زمنية معينة في الجدول لجلسات مقررات مختلفة.
  - **جدول الصلة :**يعمل جدول Schedule كرابط، حيث يحدد غرفة الصف وأيام الأسبوع ووقت بدء/نهاية كل جلسة مقرر.



- **الموظف <---(M:N)---> المادة:**
  - الوصف: يمكن أن يكون الموظف (المعلم) مؤهلاً لتدريس مواد متعددة، وقد يتم تدريس مادة من قبل مدرسين متعددين خلال فصول دراسية أو سنوات دراسية مختلفة.



1.3. لمخطط النهائي



2. دمج oracle مع C#

2.1. تجهيز قاعدة البيانات

2.1.1. تركيب Oracle

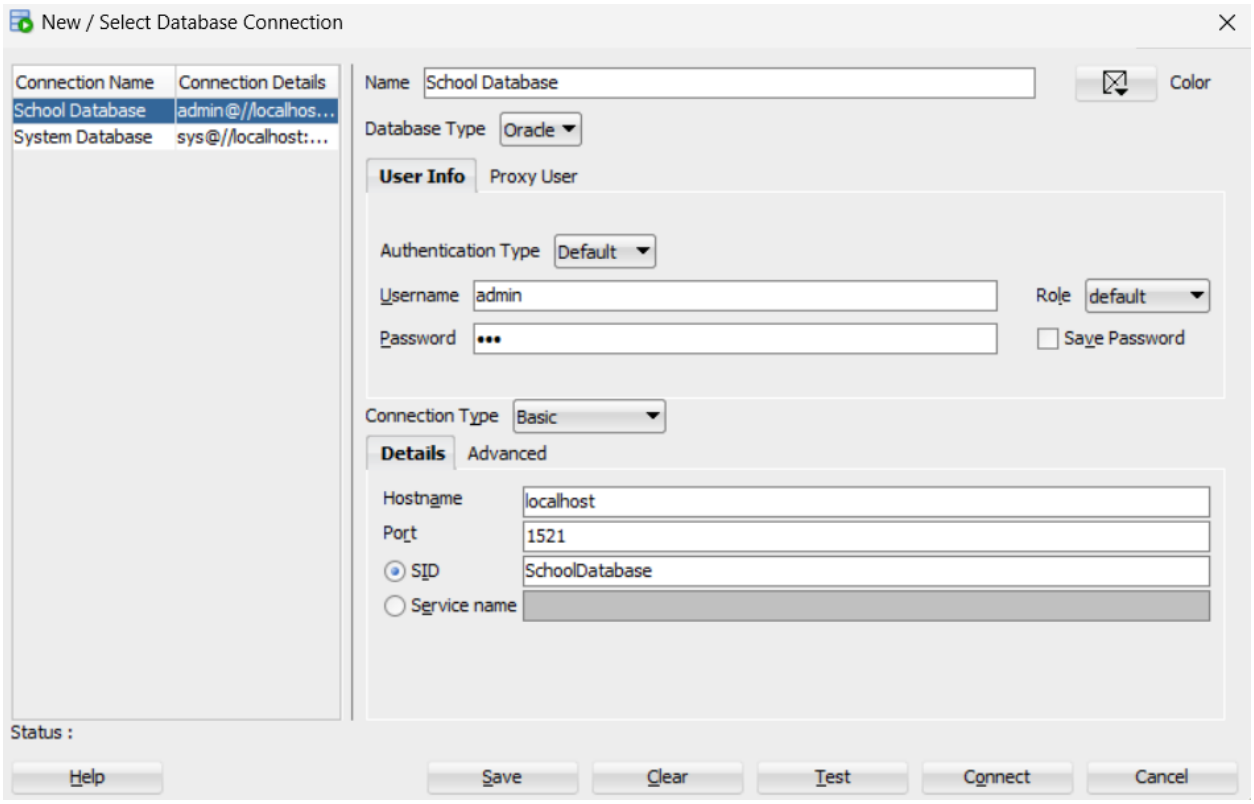
- أولاً، تم بتنزيل وتثبيت Oracle Database 11g تُشكل هذه الخطوة الأساس لخادم قاعدة البيانات.
- ثانياً، تم بتنزيل Oracle SQL Developer توفر هذه الأداة واجهة رسومية سهلة الاستخدام لإدارة قاعدة بيانات Oracle، بما في ذلك إنشاء الجداول، وإدخال البيانات، وتنفيذ استعلاماتSQL

2.1.2. إدارة المستخدمين

- لضمان الوصول الآمن وإدارة إنشاء الجداول والوصول إلى البيانات، تم إنشاء مستخدم جديد داخل قاعدة بيانات Oracleاخترت حساب مستخدم مخصص بدلاً من استخدام حساب النظام (الذي يحتوي على العديد من الجداول الموجودة مسبقًا) للحفاظ على بيانات المشروع منفصلة ومنظمة.

```
-- Create a user

CREATE USER admin IDENTIFIED BY 123;

-- Grant permissions

GRANT CONNECT, RESOURCE, DBA TO admin;
```

2.1.3. إنشاء قاعدة البيانات



2.1.4. إنشاء الجداول

تم استخدام عبارات SQL DDL (لغة تعريف البيانات) لإنشاء كل جدول في مخطط قاعدة بياناتي، مع التأكد من تحديد أنواع البيانات المناسبة والقيود:

```
CREATE TABLE Employee (
    employee_id INT PRIMARY KEY,
    first_name NVARCHAR2(255) NOT NULL,
    last_name NVARCHAR2(255) NOT NULL,
    date_of_birth DATE NOT NULL,
    phone_number VARCHAR2(20),
    address NVARCHAR2(255),
    hire_date DATE NOT NULL
);

CREATE TABLE Role (
    role_id INT PRIMARY KEY,
    role_name NVARCHAR2(255) NOT NULL
);

CREATE TABLE EmployeeRole (
    employee_role_id INT PRIMARY KEY,
    employee_id INT NOT NULL,
    role_id INT NOT NULL,
    FOREIGN KEY (employee_id) REFERENCES Employee(employee_id) ON DELETE CASCADE,
    FOREIGN KEY (role_id) REFERENCES Role(role_id) ON DELETE CASCADE
);

CREATE TABLE Grade (
    grade_id INT PRIMARY KEY,
    grade_name NVARCHAR2(50) NOT NULL
);

CREATE TABLE Class (
```

```
    class_id INT PRIMARY KEY,
    class_name NVARCHAR2(255) NOT NULL,
    grade_id INT NOT NULL,
    FOREIGN KEY (grade_id) REFERENCES Grade(grade_id) ON DELETE CASCADE
);

CREATE TABLE Section (
    section_id INT PRIMARY KEY,
    section_name NVARCHAR2(50) NOT NULL,
    class_id INT NOT NULL,
    FOREIGN KEY (class_id) REFERENCES Class(class_id) ON DELETE CASCADE
);

CREATE TABLE Classroom (
    classroom_id INT PRIMARY KEY,
    room_number VARCHAR2(20) NOT NULL,
    capacity INT NOT NULL
);

CREATE TABLE Subject (
    subject_id INT PRIMARY KEY,
    subject_name NVARCHAR2(255) NOT NULL
);

CREATE TABLE Semester (
    semester_id INT PRIMARY KEY,
    semester_name NVARCHAR2(50) NOT NULL,
    school_year INT NOT NULL
);

CREATE TABLE ClassSession (
    session_id INT PRIMARY KEY,
```

```
    subject_id INT NOT NULL,
    teacher_id INT NOT NULL,
    semester_id INT NOT NULL,
    class_id INT NOT NULL,
    FOREIGN KEY (subject_id) REFERENCES Subject(subject_id) ON DELETE CASCADE,
    FOREIGN KEY (teacher_id) REFERENCES Employee(employee_id) ON DELETE CASCADE,
    FOREIGN KEY (semester_id) REFERENCES Semester(semester_id) ON DELETE CASCADE,
    FOREIGN KEY (class_id) REFERENCES Class(class_id) ON DELETE CASCADE
);

CREATE TABLE Student (
    student_id INT PRIMARY KEY,
    first_name NVARCHAR2(255) NOT NULL,
    last_name NVARCHAR2(255) NOT NULL,
    date_of_birth DATE NOT NULL,
    gender VARCHAR2(10) NOT NULL,
    address NVARCHAR2(255),
    parent_name NVARCHAR2(255),
    parent_contact VARCHAR2(20),
    enrollment_date DATE NOT NULL
);

CREATE TABLE Schedule (
    schedule_id INT PRIMARY KEY,
    class_id INT NOT NULL,
    classroom_id INT NOT NULL,
    session_id INT NOT NULL,
    day_of_week VARCHAR2(10) NOT NULL,
    start_time DATE NOT NULL,
    end_time DATE NOT NULL,
    FOREIGN KEY (class_id) REFERENCES Class(class_id) ON DELETE CASCADE,
    FOREIGN KEY (classroom_id) REFERENCES Classroom(classroom_id) ON DELETE CASCADE,
```

```
    FOREIGN KEY (session_id) REFERENCES ClassSession(session_id) ON DELETE CASCADE
);

CREATE TABLE Enrollment (
    enrollment_id INT PRIMARY KEY,
    student_id INT NOT NULL,
    session_id INT NOT NULL,
    semester_id INT NOT NULL,
    FOREIGN KEY (student_id) REFERENCES Student(student_id) ON DELETE CASCADE,
    FOREIGN KEY (session_id) REFERENCES ClassSession(session_id) ON DELETE CASCADE,
    FOREIGN KEY (semester_id) REFERENCES Semester(semester_id) ON DELETE CASCADE
);

CREATE TABLE GradeComponent (
    component_id INT PRIMARY KEY,
    component_name NVARCHAR2(255) NOT NULL
);

CREATE TABLE StudentGrade (
    grade_id INT PRIMARY KEY,
    enrollment_id INT NOT NULL,
    component_id INT NOT NULL,
    score DECIMAL(5,2) NOT NULL,
    FOREIGN KEY (enrollment_id) REFERENCES Enrollment(enrollment_id) ON DELETE CASCADE,
    FOREIGN KEY (component_id) REFERENCES GradeComponent(component_id) ON DELETE CASCADE
);

CREATE TABLE Note (
    note_id INT PRIMARY KEY,
    enrollment_id INT NOT NULL,
    semester_id INT NOT NULL,
    author_role VARCHAR2(50) NOT NULL,
```

```
    note_text NCLOB NOT NULL,
    FOREIGN KEY (enrollment_id) REFERENCES Enrollment(enrollment_id) ON DELETE CASCADE,
    FOREIGN KEY (semester_id) REFERENCES Semester(semester_id) ON DELETE CASCADE
);
```

شرح إنشاء الجداول:

- **أنواع البيانات :**اخترت بعناية أنواع البيانات المناسبة لكل عمود، مثل INT للمعرفات العددية، و NVARCHAR2 للبيانات النصية، و DATE للتواريخ.
- **المفاتيح الأساسية :**يحتوي كل جدول على قيد PRIMARY KEY لتحديد كل صف بشكل فريد. يضمن ذلك سلامة البيانات واسترجاع البيانات بكفاءة.
- **غير فارغة :**تم تمييز الأعمدة الأساسية بـ NOT NULL لمنع القيم الفارغة.
- **المفاتيح الخارجية :**قمت بتنفيذ قيود FOREIGN KEY لفرض العلاقات بين الجداول. على سبيل المثال، يحتوي جدول EmployeeRoleعلى مفاتيح خارجية إلى Employeeو Role، مما يضمن اتساق البيانات.
- **ON DELETE CASCADE:**استخدمت ON DELETE CASCADE للمفاتيح الخارجية. يقوم هذا بحذف الصفوف ذات الصلة تلقائيًا في الجداول التابعة عند حذف صف أصل، مما يحافظ على سلامة المراجع.

2.1.5. المُحفّز(Trigger) لإنشاء المعرفات تلقائيًا

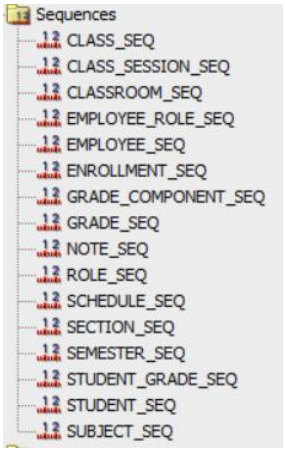
لتلقائية إنشاء معرفات فريدة لجداولِ مثل role_id، قمت بإنشاء مُحفّز :

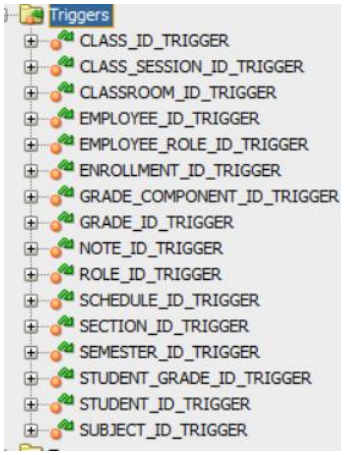
```
CREATE SEQUENCE role_seq START WITH 1 INCREMENT BY 1;


CREATE OR REPLACE TRIGGER role_id_trigger
BEFORE INSERT ON Role
FOR EACH ROW
BEGIN
  SELECT role_seq.NEXTVAL INTO :NEW.role_id FROM DUAL;
END;
/
```

شرح:

- **التسلسل** (SEQUENCE) **:**قمت بإنشاء تسلسل (role_seq) لتوليد أرقام متزايدة.
- **المُحفّز** (TRIGGER) **:**يُنشط role_id_trigger قبل إدخال أي صف في جدول .Role
- **اختيار** (NEXTVAL) **:**داخل المُحفّز، يتم جلب القيمة التالية من التسلسل ويتم إدخالها في عمود role_id للصف الجديد.

وقمت بنفس الامر لباقي الجداول:





2.1.6. ربط C# بـ:Oracle

- لتسهيل التواصل بين تطبيق C# وقاعدة بياناتOracle ، تم تنزيل حزمة .Oracle.ManagedDataAccess توفر هذه الحزمة الفئات والمكونات اللازمة في .NET للتفاعل مع.Oracle
- تم بدمج هذه الحزمة في المشروع باستخدام .NuGet تضمن ذلك أن كود C# يمكنه الوصول إلى وظائف قاعدة بيانات Oracle واستخدامها.

2.1.7. صف الاتصال:

- لتوضيح منطق الاتصال وتبسيط تفاعلات قاعدة البيانات في تطبيقC# ، تم إنشاء صف مخصصة تُسمى .SchoolDatabase تعمل هذه الفئة على مركزية إدارة الاتصال وتوفر طرقًا لتنفيذ استعلامات.SQL

كود C# صف SchoolDatabase:

```
public static class SchoolDatabase {
    private static string connectionString = "Data
Source=(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=localhost)(PORT=1521))(CONNECT_DATA=(SER
VER=DEDICATED)(SERVICE_NAME=SchoolDatabase)));User Id=admin;Password=123;";
    public static DataTable ExecuteQuery(string query, OracleParameter[] parameters =
null) {
        DataTable dataTable = new DataTable();

        using (OracleConnection connection = new OracleConnection(connectionString)) {
            using (OracleCommand command = new OracleCommand(query, connection)) {
                if (parameters != null) {
                    command.Parameters.AddRange(parameters);
                }

                try {
                    connection.Open();
                    using (OracleDataAdapter adapter = new OracleDataAdapter(command)) {
                        adapter.Fill(dataTable);
                    }
                }
                catch (OracleException ex) {
                    MessageBox.Show("Oracle Error: " + ex.Message);
                    throw;
```

```
                }
                catch (Exception ex) {
                    MessageBox.Show("Error: " + ex.Message);
                    throw;
                }
            }
        }

        return dataTable;
    }
    public static int ExecuteNonQuery(string query, OracleParameter[] parameters = null)
{
        int rowsAffected = 0;

        using (OracleConnection connection = new OracleConnection(connectionString)) {
            using (OracleCommand command = new OracleCommand(query, connection)) {
                if (parameters != null) {
                    command.Parameters.AddRange(parameters);
                }

                try {
                    connection.Open();
                    rowsAffected = command.ExecuteNonQuery();
                }
                catch (OracleException ex) {
                    MessageBox.Show("Oracle Error: " + ex.Message);
                    throw;
                }
                catch (Exception ex) {
                    MessageBox.Show("Error: " + ex.Message);
                    throw;
                }
            }
        }
        return rowsAffected;
    }
}
```

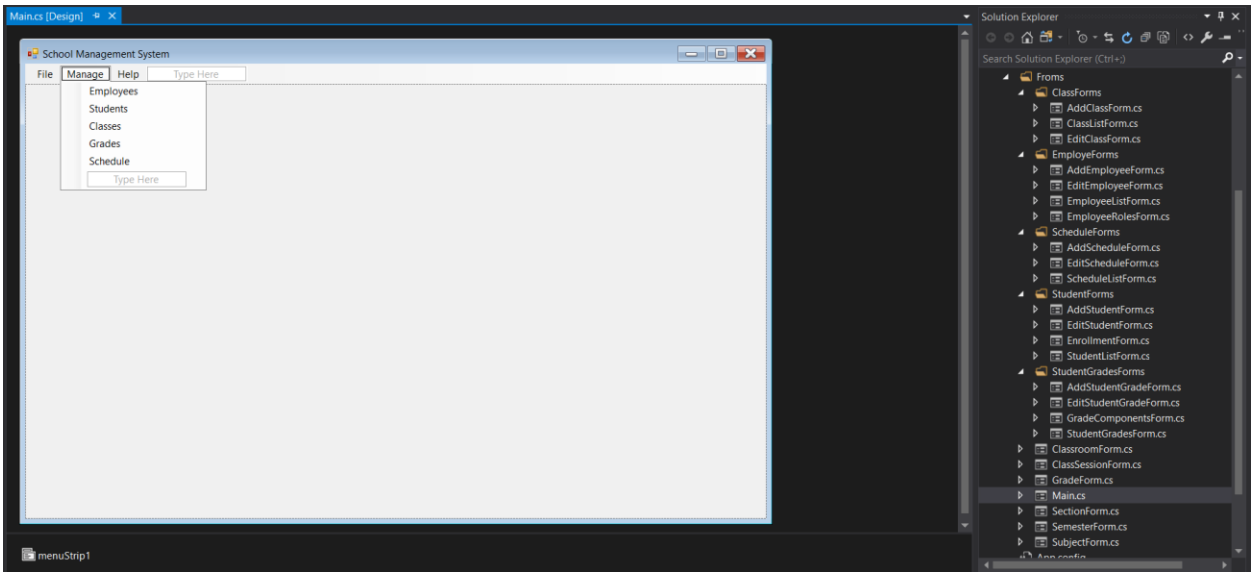
شرح:

- **سلسلة الاتصال :**تخزن متغير connectionString تفاصيل الاتصال للوصول إلى قاعدة بيانات Oracle تتضمن عنوان خادم قاعدة البيانات (المضيف، المنفذ)، واسم الخدمة، واسم المستخدم، وكلمة المرور. تأكد من استبدال هذه القيم بمعلومات الاتصال المحددة.
- **طريقة :ExecuteQuery** تقوم هذه الطريقة بمعالجة تنفيذ استعلامات .SELECT تقوم بفتح اتصال بقاعدة بياناتOracle ، وإنشاء كائن أمر، واستخدام OracleDataAdapter لملء DataTable بنتائج الاستعلام. تتضمن معالجة أخطاء قوية مع كتل catchمنفصلة لأخطاء Oracle المحددة والاستثناءات العامة.
- **طريقة :ExecuteNonQuery** تتعامل هذه الطريقة مع تنفيذ استعلامات INSERT و UPDATE و .DELETE تقوم بفتح اتصال، وإنشاء أمر، واستخدام ExecuteNonQuery لإرسال الاستعلام إلى قاعدة البيانات. مثل ExecuteQuery، تتضمن معالجة أخطاء قوية.

2.2. النماذج (Forms)

2.2.1. نموذج "Main"

نموذج "Main" هو النقطة المركزية للتحكم في نظام إدارة المدرسة بأكمله. يعمل كحاوية للنماذج الفرعية المختلفة (مثل "قائمة الموظفين", "قائمة الطلاب" وغيرها) ويوفر شريط القائمة للتنقل بين هذه الوظائف.



بناء النموذج:

تم بناء نموذج "Main" باستخدام Windows Forms في .C# هذه هي هيكله:

1. **شريط القائمة:**
   - في الأعلى، لدينا عنصر تحكم MenuStrip (menuStrip1). يحتوي شريط القائمة هذا على قوائم "ملف", "إدارة" و "مساعدة", وكلها تحتوي على عناصر فرعية.
2. **لوحة منطقة العمل:**
   - تحتل mainAreaPanel (mainAreaPanel) أكبر مساحة في النموذج. تعمل كحاوية للنماذج الفرعية التي سيتم عرضها. لها لوحتان فرعيتان:
     - welcomeScreenPanel: تُعَرض هذه اللوحة في البداية وقد تعرض رسالة ترحيب أو نظرة عامة على النظام.
     - subFormContainerPanel: سيتم استخدام هذه اللوحة لتحميل وعرض النماذج الفرعية بشكل ديناميكي عند تنقل المستخدم عبر القائمة.

وظائف النموذج:

يدير نموذج "Main" واجهة المستخدم ويوفر الميزات التالية:

1. **التنقل عبر القائمة:**

- تتيح عناصر القائمة للمستخدمين التنقل إلى أقسام مختلفة من التطبيق.
- **قائمة ملف :**تحتوي قائمة "ملف" على خيار "خروج" لإغلاق التطبيق بأكمله.
- **قائمة إدارة :**تحتوي قائمة "إدارة" على الخيارات للتفاعل مع كيانات مختلفة في النظام:
  - "الموظفين": يفتح هذا نموذجًا لعرض سجلات الموظفين وإضافتها وتعديلها وحذفها.
  - "الطلاب": يفتح هذا نموذجًا لإدارة بيانات الطلاب.
  - "الفصول": يسمح هذا بإدارة معلومات الفصول.
  - "الدرجات": يسمح هذا للمستخدمين بعرض وتعديل درجات الطلاب.
  - "الجدول الزمني": يُدير هذا القسم جدول الحصص الدراسية.
- **قائمة مساعدة :**تقدم قائمة "مساعدة" خيار "حول" يعرض معلومات حول النظام.

2. **تحميل النماذج الفرعية:**

- عندما يختار المستخدم عنصر قائمة (مثل "الموظفين" أو "الطلاب"), يستخدم نموذج "Main" subFormContainerPanel"للتحميل الديناميكي للنموذج الفرعي المناسب. يصبح هذا النموذج الفرعي نموذجًا تابعًا لنموذج ,"Main" لذلك يظهر داخل النافذة الرئيسية.
- هذه هي كيفية عمل عملية تحميل النماذج الفرعية بشكل عام:

1. مسح أي نماذج فرعية موجودة من .subFormContainerPanel
2. إنشاء مثيل للنموذج الفرعي المطلوب (على سبيل المثال.(EmployeeListForm ,
3. تعيين الخاصية TopLevel للنموذج الفرعي إلى falseبحيث يصبح نموذجًا تابعًا.
4. تعيين الخاصية FormBorderStyle للنموذج الفرعي إلى FormBorderStyle.Noneلإزالة حدوده (اختياري، لكن يجعل المظهر أكثر انسجامًا).
5. إضافة النموذج الفرعي إلى .subFormContainerPanel
6. عرض النموذج الفرعي.

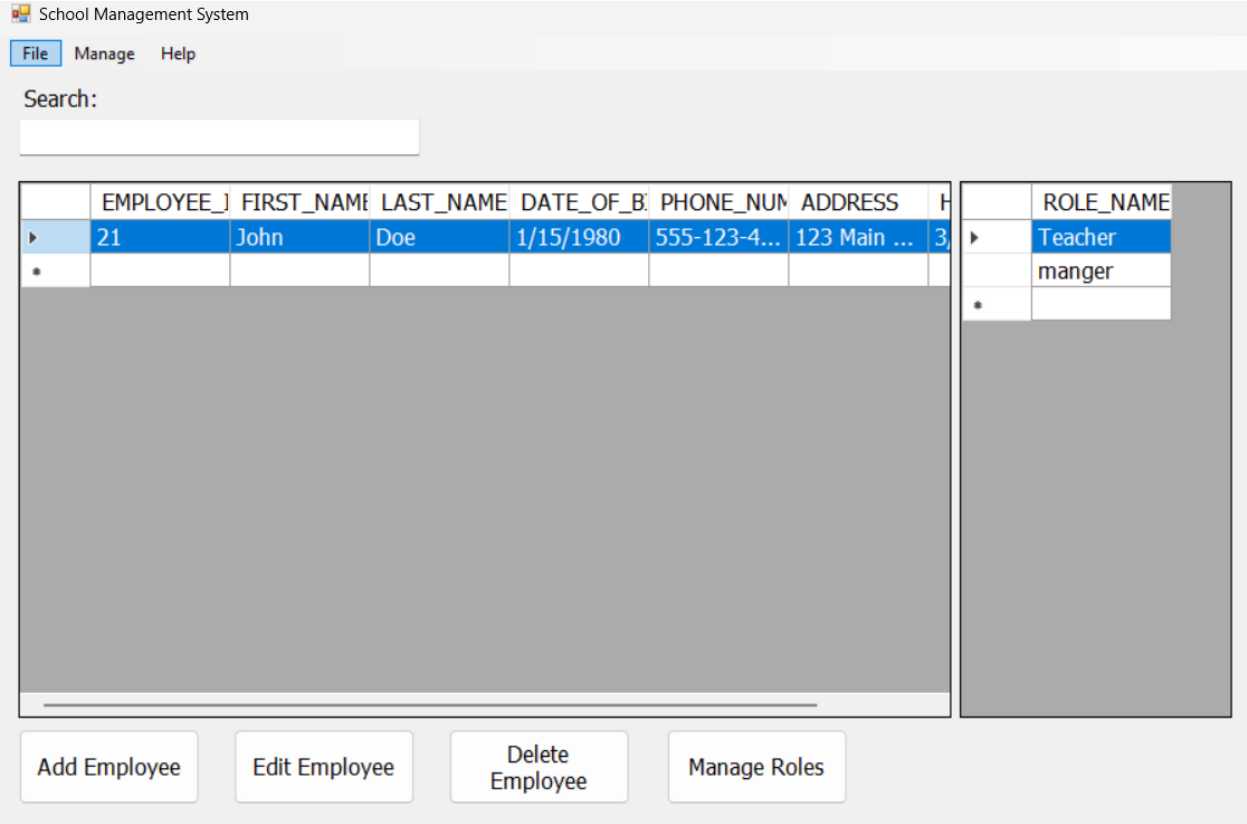
مثال كود:

```
private void employeesToolStripMenuItem_Click(object sender, EventArgs e) {
    subFormContainerPanel.Controls.Clear();

    welcomeScreenPanel.Visible = false;

    EmployeeListForm employeeListForm = new EmployeeListForm();

    employeeListForm.TopLevel = false;
    employeeListForm.FormBorderStyle = FormBorderStyle.None;
    subFormContainerPanel.Controls.Add(employeeListForm);
    employeeListForm.Show();
}
```

2.2.2. نماذج الموظفين: إدارة الموظفين

2.2.2.1. نموذج قائمة الموظفين

نموذج EmployeeListForm هذا هو مركز التحكم لسجلات الموظفين. إنه المكان الذي يمكننا فيه عرض معلومات الموظفين وإضافة سجلات جديدة وتعديلها وحذفها.



مكونات النموذج:

1. **شبكة بيانات الموظفين:**
   - العنصر الرئيسي هو employeeDataGridView. هذا هو عنصر DataGridView الذي يعرض قائمة الموظفين المسترجعة من قاعدة بيانات Oracle الخاصة بنا. يمثل كل صف موظفًا واحدًا، بينما يمثل كل عمود سمة من سمات الموظف (مثل employee_id, first_name, last_name, date_of_birth, وما إلى ذلك.)
2. **شريط البحث:**
   - تسمح لنا searchTextBox بالعثور على موظفين محددين بسرعة. عند الكتابة في مربع البحث، يقوم DataGridViewتلقائيًا بتصفية قائمة الموظفين لعرض فقط أولئك الذين يطابقون مصطلح البحث. البحث غير حساس لحالة الأحرف، ويبحث عن المطابقات في كل من الاسم الأول واسم العائلة.
3. **أزرار العمل:**
   - لدينا ثلاثة أزرار لأداء إجراءات:

- **"إضافة موظف :"**يفتح نموذجًا جديدًا AddEmployeeForm حيث يمكننا إدخال جميع تفاصيل موظف جديد.
- **"تعديل موظف :"**يفتح نموذج EditEmployeeForm عند تحديد صف موظف في employeeDataGridView.يقوم هذا النموذج مسبقًا بملء الحقول بالبيانات الموجودة للموظف لتسهيل التعديل.
- **"حذف موظف :"**يسمح لنا بإزالة موظف من النظام. يطلب التأكيد لتجنب الحذف العرضي.

4. **شبكة بيانات الأدوار:**

- لدينا DataGridViewثاني يُسمى rolesDataGridViewوالذي يُظهر الأدوار المخصصة للموظف المحدد. عند تحديد صف موظف، يتم تحميل الأدوار الخاصة بهذا الموظف وعرضها ديناميكيًا هنا.

5. **زر "إدارة الأدوار:"**

- يفتح هذا الزر نموذجًا منفصلاً يُسمى EmployeeRolesForm حيث يمكننا إضافة أو حذف أو إدارة الأدوار المتاحة في النظام. هذا النموذج مستقل عن الموظف المحدد.

وظائف النموذج:

1. **تحميل بيانات الموظفين:**

- عند تحميل النموذج، يسترجع جميع بيانات الموظفين من قاعدة بيانات Oracle باستخدام استعلام SELECTويربطه بـ employeeDataGridView.

2. **وظيفة البحث:**

- يقوم مُعالِج الأحداث searchTextBox_TextChanged بتشغيل بحث ديناميكي. يبني استعلام Oracle يستخدم عامل التشغيل LIKEمع أحرف البدل (%) لأداء مطابقة سلسلة جزئية. ثم يتم عرض نتائج البحث في employeeDataGridView.

3. **حذف موظف:**

- معالج الأحداث :deleteButton_Click
  1. يتأكد من تحديد صف موظف.
  2. يطلب التأكيد باستخدام MessageBox.
  3. إذا تم التأكيد، يقوم بتنفيذ استعلام DELETEلإزالة الموظف من قاعدة البيانات.
  4. يُعَدِّل employeeDataGridViewلنعكس التغييرات.

4. **تعديل موظف:**

- معالج الأحداث :editButton_Click
  1. يتأكد من تحديد صف موظف.
  2. يحصل على معرف الموظف.
  3. يفتح EditEmployeeForm، ويُمرر معرف الموظف.
  4. بعد إغلاق EditEmployeeForm (مع "موافق"), يتم تحديث employeeDataGridView.

5. **إضافة موظف:**
   - معالج الأحداث addButton_Click:
     1. يفتح AddEmployeeForm.
     2. بعد إغلاق AddEmployeeForm (مع "موافق"), يتم تحديث employeeDataGridView.
6. **عرض أدوار الموظف:**
   - يقوم مُعالِج الأحداث employeeDataGridView_SelectionChanged بتحديث rolesDataGridView.
     يسترد الأدوار الخاصة بالموظف المحدد باستخدام استعلام JOIN ويُعرضها في rolesDataGridView.

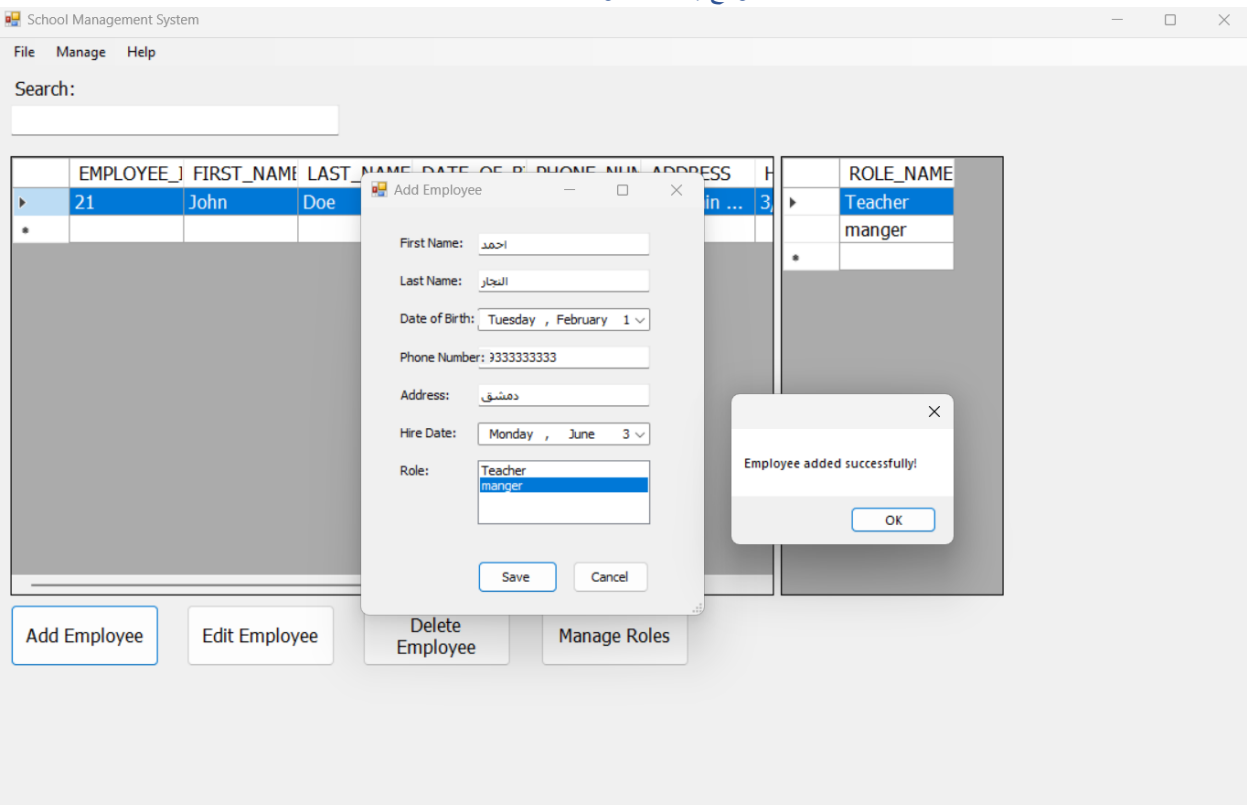
الكود:

```
public partial class EmployeeListForm : Form {
    public EmployeeListForm() {
        InitializeComponent();
    }

    private void EmployeeListForm_Load(object sender, EventArgs e) {
        DataTable employeesDataTable = SchoolDatabase.ExecuteQuery("select * from
employee");

        employeeDataGridView.DataSource = employeesDataTable;
    }

    private void searchTextBox_TextChanged(object sender, EventArgs e) {
        string searchTerm = searchTextBox.Text.Trim();

        string query = @"SELECT * FROM Employee WHERE First_Name LIKE :searchTerm OR
Last_Name LIKE :searchTerm";

        OracleParameter searchTermParameter = new OracleParameter(":searchTerm",
OracleDbType.Varchar2);
        searchTermParameter.Value = "%" + searchTerm + "%";
        DataTable searchResults = SchoolDatabase.ExecuteQuery(query, new
OracleParameter[] { searchTermParameter });
        employeeDataGridView.DataSource = searchResults;
    }

    private void deleteButton_Click(object sender, EventArgs e) {

        if (employeeDataGridView.SelectedRows.Count == 0) {
            MessageBox.Show("Please select an employee to delete.");
            return;
        }

        int employeeId =
Convert.ToInt32(employeeDataGridView.SelectedRows[0].Cells["employee_id"].Value);

        DialogResult result = MessageBox.Show(
            $"Are you sure you want to delete employee with ID {employeeId}?",
            "Confirm Deletion",
            MessageBoxButtons.YesNo,
            MessageBoxIcon.Warning);
```

```
        if (result == DialogResult.No) {
            return;
        }

        try {
            OracleParameter employeeIdParameter = new OracleParameter("employee_id",
employeeId);

            int rowsAffected = SchoolDatabase.ExecuteNonQuery(
                "DELETE FROM Employee WHERE employee_id = :employee_id",
                new OracleParameter[] { employeeIdParameter });

            if (rowsAffected > 0) {
                MessageBox.Show("Employee deleted successfully.");

                RefreshEmployeeList();
            }
            else {
                MessageBox.Show("An error occurred while deleting the employee.");
            }
        }
        catch (Exception ex) {
            MessageBox.Show("Error: " + ex.Message);
        }
    }


    private void editButton_Click(object sender, EventArgs e) {

        if (employeeDataGridView.SelectedRows.Count == 0) {
            MessageBox.Show("Please select an employee to edit.");
            return;
        }

        int selectedEmployeeId =
Convert.ToInt32(employeeDataGridView.SelectedRows[0].Cells["employee_id"].Value);
        EditEmployeeForm editEmployeeForm = new EditEmployeeForm(selectedEmployeeId);
        DialogResult result = editEmployeeForm.ShowDialog();

        if (result == DialogResult.OK) {
            RefreshEmployeeList();
        }
    }

    private void addButton_Click(object sender, EventArgs e) {
        AddEmployeeForm addEmployeeForm = new AddEmployeeForm();
        DialogResult result = addEmployeeForm.ShowDialog();
        if (result == DialogResult.OK) {
            DataTable employeesDataTable = SchoolDatabase.ExecuteQuery("select * from
employee");
            employeeDataGridView.DataSource = employeesDataTable;
        }
    }

    private void RefreshEmployeeList() {
        DataTable employeeDataTable = SchoolDatabase.ExecuteQuery("SELECT * FROM
Employee");
        employeeDataGridView.DataSource = employeeDataTable;
```

```
    }

    private void employeeDataGridView_SelectionChanged(object sender, EventArgs e) {
        if (employeeDataGridView.SelectedRows.Count > 0) {
            int selectedEmployeeId =
Convert.ToInt32(employeeDataGridView.SelectedRows[0].Cells["employee_id"].Value);

            string roleQuery = "SELECT r.role_name FROM EmployeeRole er JOIN Role r ON
er.role_id = r.role_id WHERE er.employee_id = :employeeId";
            OracleParameter employeeIdParam = new OracleParameter("employeeId",
OracleDbType.Int32, selectedEmployeeId, ParameterDirection.Input);
            DataTable roleData = SchoolDatabase.ExecuteQuery(roleQuery, new
OracleParameter[] { employeeIdParam });
            rolesDataGridView.DataSource = roleData;
        }
        else {
            rolesDataGridView.DataSource = null;
        }
    }

    private void viewRolesButton_Click(object sender, EventArgs e) {
        EmployeeRolesForm manageRolesForm = new EmployeeRolesForm();
        manageRolesForm.ShowDialog();
    }
}
```

2.2.2.2. نموذج إضافة الموظف



مكونات النموذج:

1. **اختيار الدور:**
   - roleListBox: يعرض مربع القائمة الأدوار المتاحة من جدول Role في قاعدة البيانات. يمكن للمستخدمين اختيار دور أو أكثر للموظف الجديد.
2. **أزرار الإجراء:**
   - saveButton: عند النقر عليه، يأخذ بيانات الإدخال ويحقق صحتها، ثم يقوم بإدراج سجل موظف جديد في جدول Employee وربط الموظف بالأدوار المختارة في جدول EmployeeRole.
   - cancelButton: هذا الزر يغلق النموذج فقط دون حفظ أي تغييرات.

وظائف النموذج:

1. **تحميل الأدوار:**
   - عند تحميل النموذج، يقوم LoadRolesIntoListBox باسترجاع جميع الأدوار المتاحة من جدول Role وتعبئة roleListBox.
2. **حفظ الموظف:**
   - معالج حدث saveButton_Click:

1. يجمع البيانات المدخلة من مربعات النصوص واختيار التواريخ ويقوم بإنشاء كائن Employee جديد.
2. ينفذ استعلام INSERT لإضافة الموظف الجديد إلى جدول Employee.
3. يسترد employee_id الذي تم إنشاؤه حديثًا للموظف المُدرج.
4. يُكرّر خلال الأدوار المختارة من roleListBox وينفذ استعلامات INSERT لإنشاء سجلات EmployeeRole المقابلة، مما يربط employee_id للموظف مع كل role_id مُحدد.
5. يعرض رسالة نجاح أو خطأ بناءً على نتيجة عمليات قاعدة البيانات.

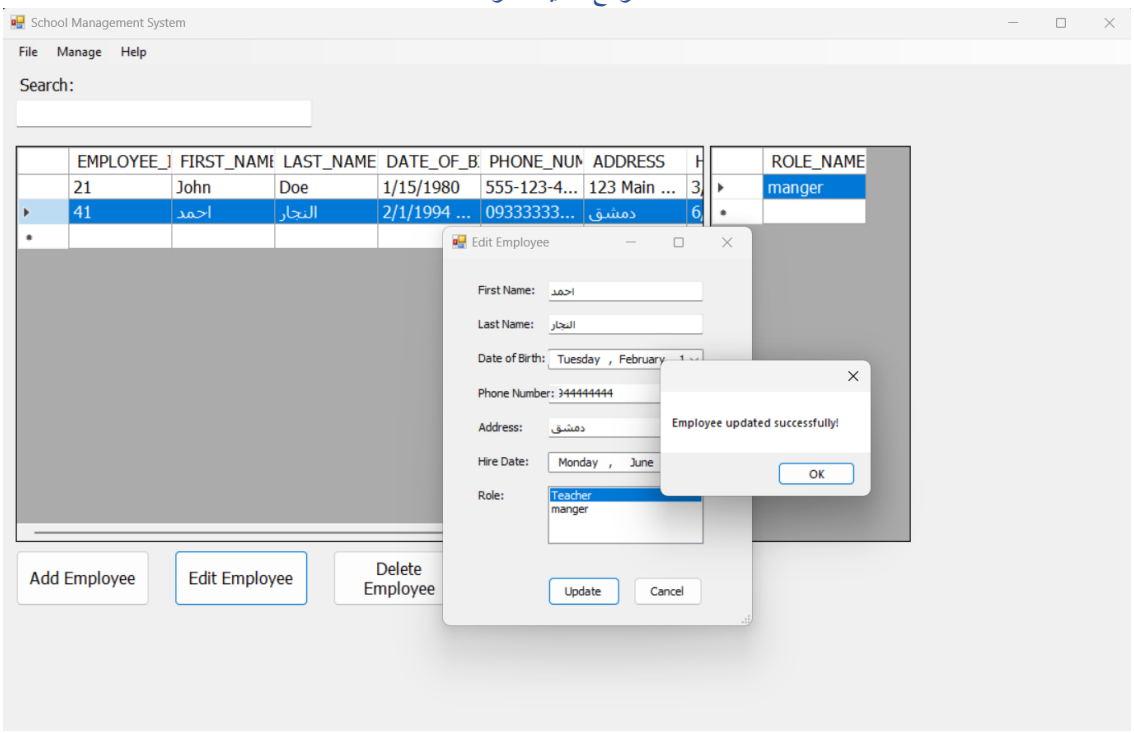
الكود:

```
public partial class AddEmployeeForm : Form {
    public AddEmployeeForm() {
        InitializeComponent();
        LoadRolesIntoListBox();
    }
    private void LoadRolesIntoListBox() {
        DataTable roles = SchoolDatabase.ExecuteQuery("SELECT role_id, role_name FROM
Role");
        roleListBox.DataSource = roles;
        roleListBox.DisplayMember = "role_name";
        roleListBox.ValueMember = "role_id";
    }

    private void AddEmployeeForm_Load(object sender, EventArgs e) {
    }

    private void cancelButton_Click(object sender, EventArgs e) {
        this.Close();
```

```
    }

    private void saveButton_Click(object sender, EventArgs e) {
        Employee newEmployee = new Employee {
            FirstName = firstNameTextBox.Text,
            LastName = lastNameTextBox.Text,
            DateOfBirth = dateOfBirthDateTimePicker.Value,
            PhoneNumber = phoneNumberTextBox.Text,
            Address = addressTextBox.Text,
            HireDate = hireDateDateTimePicker.Value
        };
        string insertQuery = "INSERT INTO Employee (first_name, last_name, date_of_birth,
phone_number, address, hire_date) VALUES (:firstName, :lastName, :dateOfBirth,
:phoneNumber, :address, :hireDate)";
        OracleParameter[] parameters = new OracleParameter[]
        {
            new OracleParameter("firstName", OracleDbType.Varchar2,
newEmployee.FirstName, ParameterDirection.Input),
            new OracleParameter("lastName", OracleDbType.Varchar2, newEmployee.LastName,
ParameterDirection.Input),
            new OracleParameter("dateOfBirth", OracleDbType.Date,
newEmployee.DateOfBirth, ParameterDirection.Input),
            new OracleParameter("phoneNumber", OracleDbType.Varchar2,
newEmployee.PhoneNumber, ParameterDirection.Input),
            new OracleParameter("address", OracleDbType.Varchar2, newEmployee.Address,
ParameterDirection.Input),
            new OracleParameter("hireDate", OracleDbType.Date, newEmployee.HireDate,
ParameterDirection.Input)
        };

        int rowsAffected = SchoolDatabase.ExecuteNonQuery(insertQuery, parameters);

        string getEmployeeIdQuery = "SELECT employee_id FROM Employee WHERE first_name =
:firstName AND last_name = :lastName AND date_of_birth = :dateOfBirth AND phone_number =
:phoneNumber AND address = :address AND hire_date = :hireDate";
        OracleParameter[] employeeIdParams = new OracleParameter[]
        {
            new OracleParameter("firstName", OracleDbType.Varchar2,
newEmployee.FirstName, ParameterDirection.Input),
            new OracleParameter("lastName", OracleDbType.Varchar2, newEmployee.LastName,
ParameterDirection.Input),
            new OracleParameter("dateOfBirth", OracleDbType.Date,
newEmployee.DateOfBirth, ParameterDirection.Input),
            new OracleParameter("phoneNumber", OracleDbType.Varchar2,
newEmployee.PhoneNumber, ParameterDirection.Input),
            new OracleParameter("address", OracleDbType.Varchar2, newEmployee.Address,
ParameterDirection.Input),
            new OracleParameter("hireDate", OracleDbType.Date, newEmployee.HireDate,
ParameterDirection.Input)
        };

        DataTable employeeIdData = SchoolDatabase.ExecuteQuery(getEmployeeIdQuery,
employeeIdParams);

        if (employeeIdData.Rows.Count > 0) {
            int newEmployeeId = Convert.ToInt32(employeeIdData.Rows[0]["employee_id"]);
```

```
            foreach (int roleId in
roleListBox.SelectedItems.Cast<DataRowView>().Select(row =>
Convert.ToInt32(row.Row["role_id"]))) {
                string insertRoleQuery = "INSERT INTO EmployeeRole (employee_id, role_id)
VALUES (:employeeId, :roleId)";
                OracleParameter[] insertRoleParams = new OracleParameter[]
                {
                    new OracleParameter("employeeId", OracleDbType.Int32, newEmployeeId,
ParameterDirection.Input),
                    new OracleParameter("roleId", OracleDbType.Int32, roleId,
ParameterDirection.Input)
                };
                SchoolDatabase.ExecuteNonQuery(insertRoleQuery, insertRoleParams);
            }
            if (rowsAffected > 0) {
                MessageBox.Show("Employee added successfully!");
                this.DialogResult = DialogResult.OK;
                this.Close();
            }
            else {
                MessageBox.Show("An error occurred while adding the employee.");
            }
        }
        else {
            MessageBox.Show("An error occurred while getting the new employee ID.");
        }
    }
}
```

2.2.2.3. نموذج تعديل الموظف



مكونات النموذج:

1. **حقول إدخال بيانات الموظف:**
   - يتم تعبئة هذه الحقول مسبقًا ببيانات موجودة للموظف المحدد عند فتح النموذج.
2. **قائمة تحديد الدور:**
   - يعرض roleListBox قائمة الأدوار المتاحة من قاعدة البيانات.
   - يتم تحديد الأدوار المخصصة حاليًا للموظف مسبقًا في القائمة عند تحميل النموذج. يمكن للمستخدمين إضافة أو إزالة الأدوار من خلال تحديدها أو إلغاء تحديدها في القائمة.
3. **أزرار الإجراء:**
   - يتم التحكم في إجراءات النموذج بواسطة زرين:
     - **"تحديث :"**يحفظ هذا الزر التغييرات التي تم إجراؤها على بيانات الموظف في قاعدة البيانات. يقوم بتحديث جدول Employeeويعدل أيضًا جدول EmployeeRole ليعكس أي تغييرات في الأدوار المحددة.
     - **"إلغاء :"**يغلق هذا الزر النموذج دون حفظ أي تغييرات.

وظائف النموذج:

1. **تحميل بيانات الموظف:**
   - يستدعي LoadEmployeeData بيانات الموظف المحدد من قاعدة البيانات باستخدام استعلام .SELECT
   - ثم يقوم بتعبئة حقول إدخال النموذج بالبيانات المسترجعة.
2. **تحميل الأدوار:**
   - يستدعي LoadRolesIntoListBoxجميع الأدوار من جدول .Role
   - ثم يقوم بتحديد الأدوار المخصصة بالفعل للموظف مسبقًا في .roleListBox
3. **تحديث الموظف:**
   - يعالج :updateButton_Click
     1. يخلق كائن Employeeباستخدام البيانات المُحدّثة من حقول النموذج.
     2. ينفذ استعلام UPDATEلتحديث تفاصيل الموظف في جدول .Employee
     3. يحذف الأدوار الموجودة في جدول EmployeeRoleللموظف.
     4. يُدخل أدوارًا جديدة (من roleListBoxفي جدول .EmployeeRole)
     5. يعرض رسالة نجاح إذا نجح التحديث، أو يعرض رسالة خطأ خلاف ذلك.
     6. يغلق النموذج.

الكود:

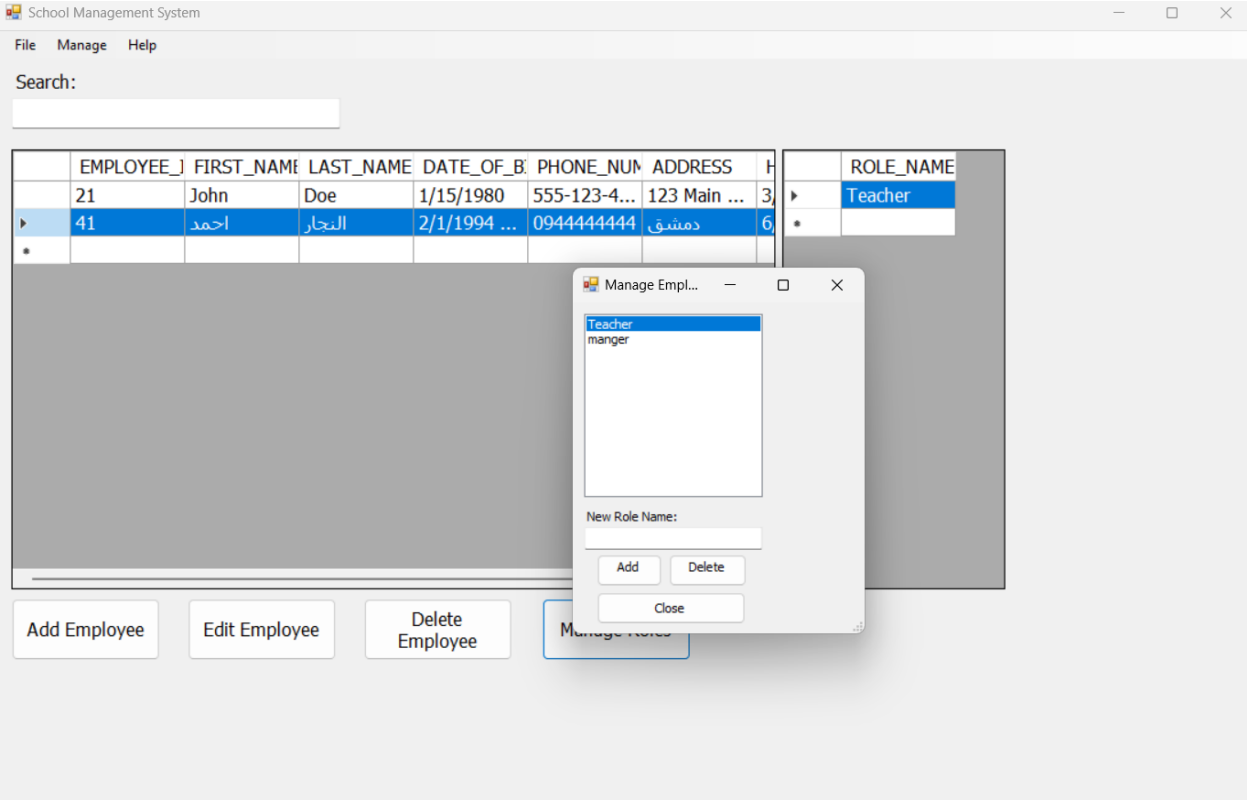
```
public partial class EditEmployeeForm : Form {
    private int employeeId;
```

```
    public EditEmployeeForm(int employeeId) {
        InitializeComponent();
        this.employeeId = employeeId;
        LoadRolesIntoListBox();
        LoadEmployeeData();
    }

    private void LoadEmployeeData() {
        string query = "SELECT * FROM Employee WHERE employee_id = :employeeId";
        OracleParameter parameter = new OracleParameter("employeeId", OracleDbType.Int32,
employeeId, ParameterDirection.Input);
        DataTable employeeData = SchoolDatabase.ExecuteQuery(query, new OracleParameter[]
{ parameter });

        if (employeeData.Rows.Count > 0) {
            DataRow row = employeeData.Rows[0];
            firstNameTextBox.Text = row["first_name"].ToString();
            lastNameTextBox.Text = row["last_name"].ToString();
            dateOfBirthDateTimePicker.Value = Convert.ToDateTime(row["date_of_birth"]);
            phoneNumberTextBox.Text = row["phone_number"].ToString();
            addressTextBox.Text = row["address"].ToString();
            hireDateDateTimePicker.Value = Convert.ToDateTime(row["hire_date"]);
        }
        else {
            MessageBox.Show("Employee not found.");
        }
    }

    private void LoadRolesIntoListBox() {
        DataTable roles = SchoolDatabase.ExecuteQuery("SELECT role_id, role_name FROM
Role");
        roleListBox.DataSource = roles;
        roleListBox.DisplayMember = "role_name";
        roleListBox.ValueMember = "role_id";

        string roleQuery = "SELECT r.role_id, r.role_name FROM EmployeeRole er JOIN Role
r ON er.role_id = r.role_id WHERE er.employee_id = :employeeId";
        OracleParameter employeeIdParam = new OracleParameter("employeeId",
OracleDbType.Int32, employeeId, ParameterDirection.Input);
        DataTable roleData = SchoolDatabase.ExecuteQuery(roleQuery, new OracleParameter[]
{ employeeIdParam });

        foreach (int roleId in roleData.AsEnumerable().Select(row =>
Convert.ToInt32(row["role_id"]))) {
            for (int i = 0; i < roleListBox.Items.Count; i++) {
                int listboxRoleId =
Convert.ToInt32(((DataRowView)roleListBox.Items[i])["role_id"].ToString());

                if (listboxRoleId == roleId) {
                    roleListBox.SetSelected(i, true);
                }
            }
        }
    }

    private void updateButton_Click(object sender, EventArgs e) {
        Employee updatedEmployee = new Employee {
```

```
            EmployeeId = employeeId,
            FirstName = firstNameTextBox.Text,
            LastName = lastNameTextBox.Text,
            DateOfBirth = dateOfBirthDateTimePicker.Value,
            PhoneNumber = phoneNumberTextBox.Text,
            Address = addressTextBox.Text,
            HireDate = hireDateDateTimePicker.Value
        };

        string updateQuery = "UPDATE Employee SET first_name = :firstName, last_name =
:lastName, date_of_birth = :dateOfBirth, phone_number = :phoneNumber, address = :address,
hire_date = :hireDate WHERE employee_id = :employeeId";
        OracleParameter[] parameters = new OracleParameter[]
        {
            new OracleParameter("firstName", OracleDbType.Varchar2,
updatedEmployee.FirstName, ParameterDirection.Input),
            new OracleParameter("lastName", OracleDbType.Varchar2,
updatedEmployee.LastName, ParameterDirection.Input),
            new OracleParameter("dateOfBirth", OracleDbType.Date,
updatedEmployee.DateOfBirth, ParameterDirection.Input),
            new OracleParameter("phoneNumber", OracleDbType.Varchar2,
updatedEmployee.PhoneNumber, ParameterDirection.Input),
            new OracleParameter("address", OracleDbType.Varchar2,
updatedEmployee.Address, ParameterDirection.Input),
            new OracleParameter("hireDate", OracleDbType.Date, updatedEmployee.HireDate,
ParameterDirection.Input),
            new OracleParameter("employeeId", OracleDbType.Int32,
updatedEmployee.EmployeeId, ParameterDirection.Input)
        };
        int rowsAffected = SchoolDatabase.ExecuteNonQuery(updateQuery, parameters);

        string deleteRoleQuery = "DELETE FROM EmployeeRole WHERE employee_id =
:employeeId";
        OracleParameter employeeIdParam = new OracleParameter("employeeId",
OracleDbType.Int32, updatedEmployee.EmployeeId, ParameterDirection.Input);
        SchoolDatabase.ExecuteNonQuery(deleteRoleQuery, new OracleParameter[] {
employeeIdParam });

        foreach (int roleId in roleListBox.SelectedItems.Cast<DataRowView>().Select(row
=> Convert.ToInt32(row.Row["role_id"]))) {
            string insertRoleQuery = "INSERT INTO EmployeeRole (employee_id, role_id)
VALUES (:employeeId, :roleId)";
            OracleParameter[] insertRoleParams = new OracleParameter[]
            {
                new OracleParameter("employeeId", OracleDbType.Int32,
updatedEmployee.EmployeeId, ParameterDirection.Input),
                new OracleParameter("roleId", OracleDbType.Int32, roleId,
ParameterDirection.Input)
            };
            SchoolDatabase.ExecuteNonQuery(insertRoleQuery, insertRoleParams);
        }
        if (rowsAffected > 0) {
            MessageBox.Show("Employee updated successfully!");
            this.DialogResult = DialogResult.OK;
            this.Close();
        }
        else {
            MessageBox.Show("An error occurred while updating the employee.");
```

```
        }
    }

    private void EditEmployeeForm_Load(object sender, EventArgs e) {
        LoadRolesIntoListBox();
    }

    private void cancelButton_Click(object sender, EventArgs e) {
        this.Close();
    }
}
```

2.2.2.4. نموذج أدوار الموظفين



مكونات النموذج:

1. **صندوق قائمة الأدوار:**
   - يُعرف باسم "availableRolesListBox" وهو عبارة عن عنصر تحكم "ListBox" يعرض جميع الأدوار المُعرّفة حاليًا في النظام. يُمثل كل عنصر في صندوق القائمة دورًا فريدًا، يُظهر اسم الدور.
2. **إدخال الدور الجديد:**
   - يوجد "newRoleNameTextBox" لإدخال اسم الدور الجديد.
3. **أزرار الإجراء:**

- توجد ثلاثة أزرار لسهولة إدارة الأدوار:

  - **"إضافة دور :"**عند النقر فوقه، يحاول إضافة الدور الجديد المُحدّد في "newRoleNameTextBox" إلى قاعدة البيانات. إذا كانت الإدخال صالحة، فإنه يُدخل الدور الجديد في جدول "Role" ويُعيد تحميل "availableRolesListBox".
  - **"حذف دور :"**يحذف الدور المحدد من "availableRolesListBox" ومن قاعدة البيانات. يطلب تأكيدًا لمنع الحذف العرضي.
  - **"إغلاق :"**يُغلق."EmployeeRolesForm"

وظائف النموذج:

1. **تحميل الأدوار:**
   - عند تحميل النموذج، يُطلق "LoadRolesIntoListBox()" لجلب جميع الأدوار من جدول "Role" في قاعدة بيانات Oracle وتعبئة."availableRolesListBox"
2. **إضافة دور:**
   - يُعالج "addRoleButton_Click" ما يلي:
     1. يُتحقق من صحة اسم الدور في."newRoleNameTextBox"
     2. إذا كان صالحة، فإنه يُنفذ استعلام "INSERT" لإضافة الدور الجديد إلى جدول."Role"
     3. يُعيد تحميل "availableRolesListBox" لعرض الدور المُضاف حديثًا.
3. **حذف دور:**
   - يُعالج "deleteRoleButton_Click" ما يلي:
     1. يُتحقق من تحديد دور في."availableRolesListBox"
     2. إذا تم تحديد دور، فإنه يُطلب تأكيدًا باستخدام."MessageBox"
     3. إذا تم التأكيد، فإنه يُنفذ استعلام "DELETE" لإزالة الدور من قاعدة البيانات.
     4. يُعيد تحميل "availableRolesListBox" لعكس الدور المحذوف.

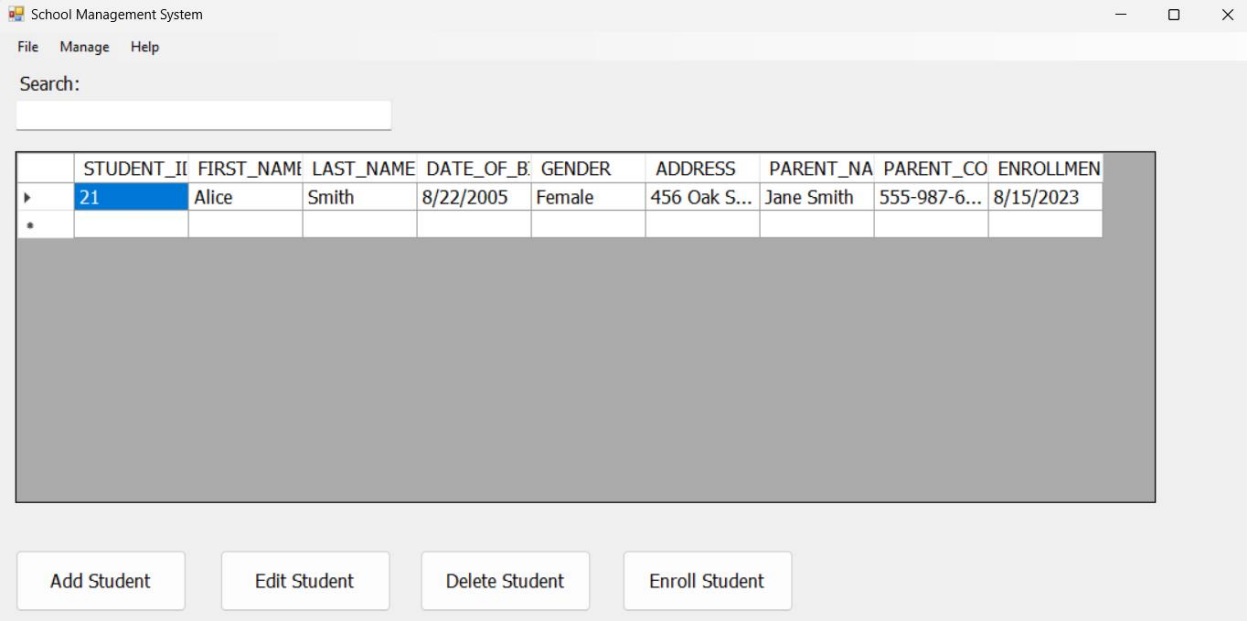
الكود:

```
public partial class EmployeeRolesForm : Form {
        public EmployeeRolesForm() {
            InitializeComponent();
            LoadRolesIntoListBox();
        }

        private void LoadRolesIntoListBox() {
            DataTable roles = SchoolDatabase.ExecuteQuery("SELECT role_id, role_name FROM
Role");
            availableRolesListBox.DataSource = roles;
            availableRolesListBox.DisplayMember = "role_name";
            availableRolesListBox.ValueMember = "role_id";
        }
```

```
        private void LoadRoles() {
            DataTable roles = SchoolDatabase.ExecuteQuery("SELECT role_id, role_name FROM
Role");
            availableRolesListBox.DataSource = roles;
            availableRolesListBox.DisplayMember = "role_name";
            availableRolesListBox.ValueMember = "role_id";
        }

        private void addRoleButton_Click(object sender, EventArgs e) {
            if (string.IsNullOrEmpty(newRoleNameTextBox.Text.Trim())) {
                MessageBox.Show("Please enter a valid role name.");
                return;
            }

            string insertRoleQuery = "INSERT INTO Role (role_name) VALUES (:roleName)";
            OracleParameter roleNameParam = new OracleParameter("roleName",
OracleDbType.Varchar2, newRoleNameTextBox.Text.Trim(), ParameterDirection.Input);
            SchoolDatabase.ExecuteNonQuery(insertRoleQuery, new OracleParameter[] {
roleNameParam });

            LoadRoles();
        }


        private void deleteRoleButton_Click(object sender, EventArgs e) {
            if (availableRolesListBox.SelectedItem == null) {
                MessageBox.Show("Please select a role to delete.");
                return;
            }

            int roleId = Convert.ToInt32(availableRolesListBox.SelectedValue);

            string deleteRoleQuery = "DELETE FROM Role WHERE role_id = :roleId";
            OracleParameter[] deleteRoleParams = new OracleParameter[]
            {
                new OracleParameter("roleId", OracleDbType.Int32, roleId,
ParameterDirection.Input)
            };
            SchoolDatabase.ExecuteNonQuery(deleteRoleQuery, deleteRoleParams);

            LoadRoles();
        }

        private void closeButton_Click(object sender, EventArgs e) {
            this.Close();
        }
        private void EmployeeRolesForm_Load(object sender, EventArgs e) {
        }
    }
```

2.2.3. نماذج الطلاب: إدارة الطلاب

2.2.3.1. نموذج قائمة الطلاب



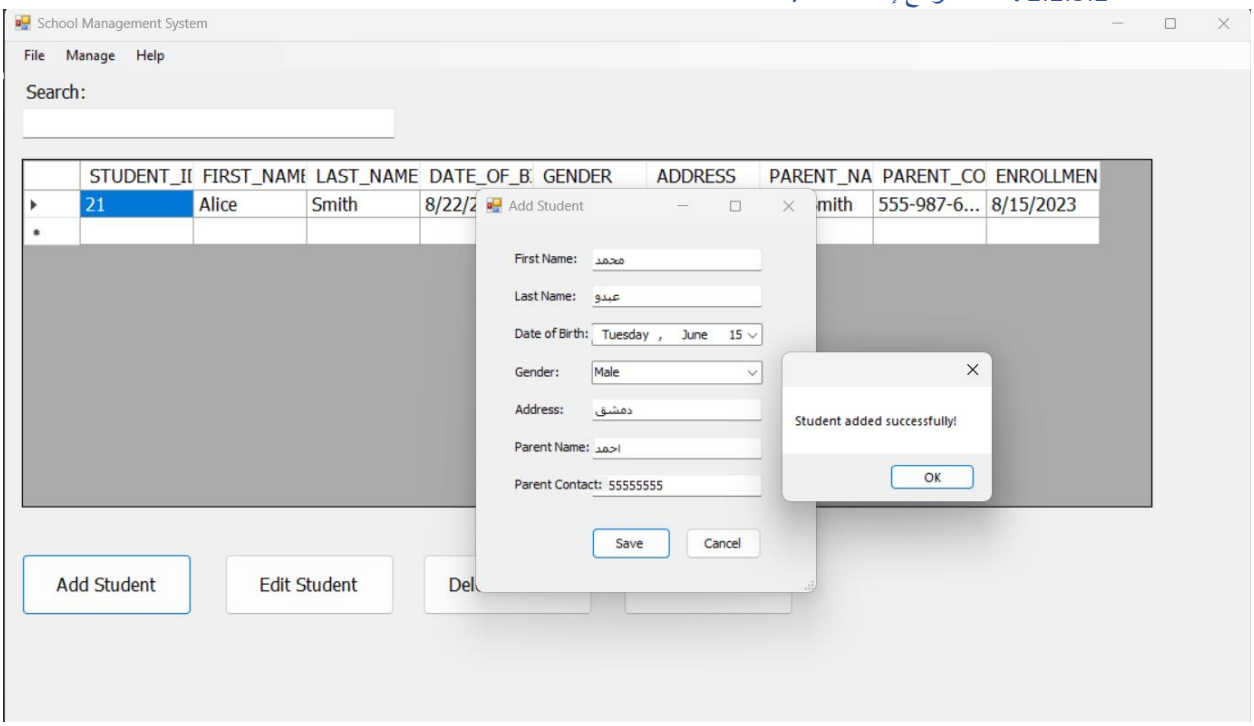
الكود:

```
public partial class StudentListForm : Form {
    public StudentListForm() {
        InitializeComponent();
    }

    private void StudentListForm_Load(object sender, EventArgs e) {
        DataTable studentsDataTable = SchoolDatabase.ExecuteQuery("select * from
student");
        studentDataGridView.DataSource = studentsDataTable;
    }

    private void searchTextBox_TextChanged(object sender, EventArgs e) {
        string searchTerm = searchTextBox.Text.Trim();
        string query = @"SELECT * FROM Student WHERE First_Name LIKE :searchTerm OR
Last_Name LIKE :searchTerm";
        OracleParameter searchTermParameter = new OracleParameter(":searchTerm",
OracleDbType.Varchar2);
        searchTermParameter.Value = "%" + searchTerm + "%";
        DataTable searchResults = SchoolDatabase.ExecuteQuery(query, new
OracleParameter[] { searchTermParameter });
        studentDataGridView.DataSource = searchResults;
    }

    private void deleteButton_Click(object sender, EventArgs e) {
        if (studentDataGridView.SelectedRows.Count == 0) {
            MessageBox.Show("Please select a student to delete.");
            return;
        }
        int studentId =
Convert.ToInt32(studentDataGridView.SelectedRows[0].Cells["student_id"].Value);
        DialogResult result = MessageBox.Show(
```

```
            $"Are you sure you want to delete student with ID {studentId}?",
            "Confirm Deletion",
            MessageBoxButtons.YesNo,
            MessageBoxIcon.Warning);
        if (result == DialogResult.No) {
            return;
        }
        try {
            OracleParameter studentIdParameter = new OracleParameter("student_id",
studentId);
            int rowsAffected = SchoolDatabase.ExecuteNonQuery(
                "DELETE FROM Student WHERE student_id = :student_id",
                new OracleParameter[] { studentIdParameter });
            if (rowsAffected > 0) {
                MessageBox.Show("Student deleted successfully.");
                RefreshStudentList();
            }
            else {
                MessageBox.Show("An error occurred while deleting the student.");
            }
        }
        catch (Exception ex) {
            MessageBox.Show("Error: " + ex.Message);
        }
    }

    private void editButton_Click(object sender, EventArgs e) {
        if (studentDataGridView.SelectedRows.Count == 0) {
            MessageBox.Show("Please select a student to edit.");
            return;
        }
        int selectedStudentId =
Convert.ToInt32(studentDataGridView.SelectedRows[0].Cells["student_id"].Value);
        EditStudentForm editStudentForm = new EditStudentForm(selectedStudentId);
        DialogResult result = editStudentForm.ShowDialog();
        if (result == DialogResult.OK) {
            RefreshStudentList();
        }
    }

    private void addButton_Click(object sender, EventArgs e) {
        AddStudentForm addStudentForm = new AddStudentForm();
        DialogResult result = addStudentForm.ShowDialog();
        if (result == DialogResult.OK) {
            DataTable studentsDataTable = SchoolDatabase.ExecuteQuery("select * from
student");
            studentDataGridView.DataSource = studentsDataTable;
        }
    }

    private void RefreshStudentList() {
        DataTable studentDataTable = SchoolDatabase.ExecuteQuery("SELECT * FROM
Student");
        studentDataGridView.DataSource = studentDataTable;
    }

    private void enrollButton_Click(object sender, EventArgs e) {
        if (studentDataGridView.SelectedRows.Count == 0) {
```

```
            MessageBox.Show("Please select a student to enroll.");
            return;
        }
        int selectedStudentId =
Convert.ToInt32(studentDataGridView.SelectedRows[0].Cells["student_id"].Value);
        EnrollmentForm enrollmentForm = new EnrollmentForm(selectedStudentId);
        enrollmentForm.ShowDialog();
    }

}
```

2.2.3.2. نموذج إضافة طالب



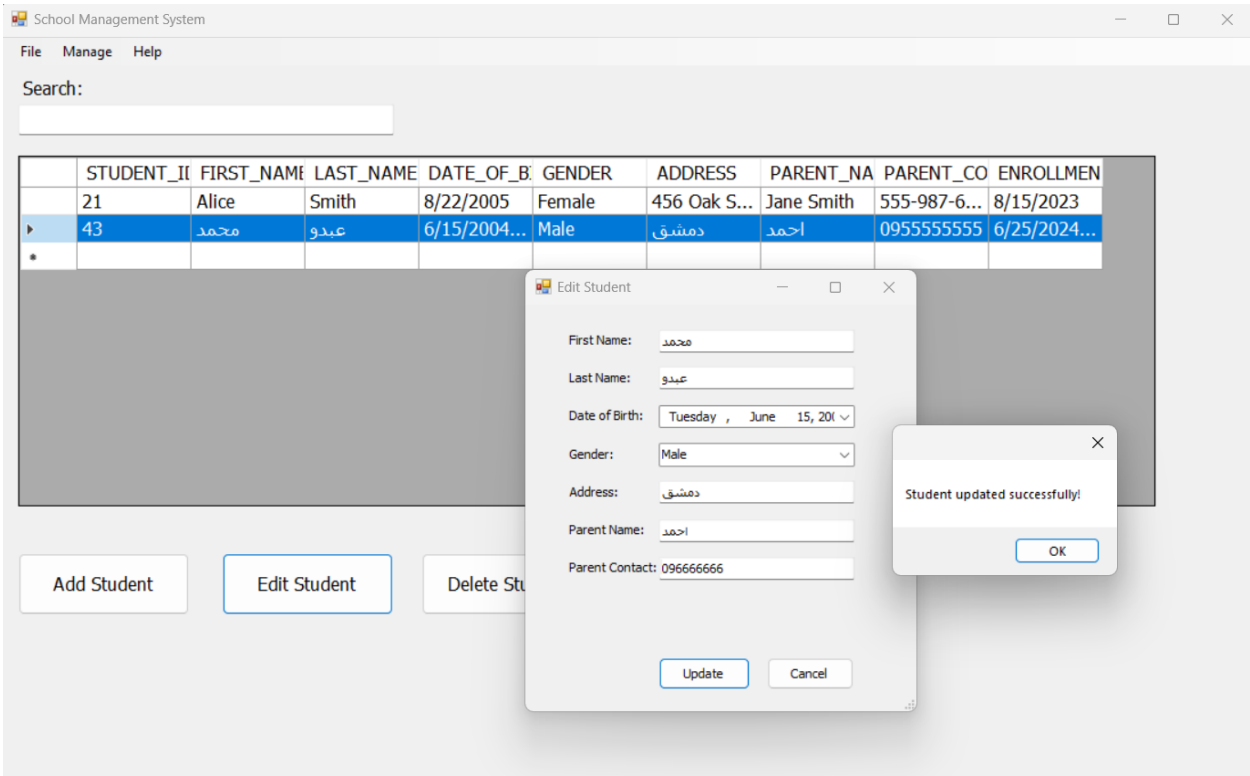
الكود:

```
private void saveButton_Click(object sender, EventArgs e) {
    Student newStudent = new Student {
        FirstName = firstNameTextBox.Text,
        LastName = lastNameTextBox.Text,
        DateOfBirth = dateOfBirthDateTimePicker.Value,
        Gender = genderComboBox.SelectedItem != null ?
genderComboBox.SelectedItem.ToString() : string.Empty,
        Address = addressTextBox.Text,
        ParentName = parentNameTextBox.Text,
        ParentContact = parentContactTextBox.Text,
        EnrollmentDate = DateTime.Now
    };

    string insertQuery = @"INSERT INTO Student (first_name, last_name, date_of_birth,
gender, address, parent_name, parent_contact, enrollment_date)
                    VALUES (:firstName, :lastName, :dateOfBirth, :gender, :address,
:parentName, :parentContact, :enrollmentDate)";
```

```
    OracleParameter[] parameters = new OracleParameter[]
    {
new OracleParameter("firstName", OracleDbType.Varchar2, newStudent.FirstName,
ParameterDirection.Input),
new OracleParameter("lastName", OracleDbType.Varchar2, newStudent.LastName,
ParameterDirection.Input),
new OracleParameter("dateOfBirth", OracleDbType.Date, newStudent.DateOfBirth,
ParameterDirection.Input),
new OracleParameter("gender", OracleDbType.Varchar2, newStudent.Gender,
ParameterDirection.Input),
new OracleParameter("address", OracleDbType.Varchar2, newStudent.Address,
ParameterDirection.Input),
new OracleParameter("parentName", OracleDbType.Varchar2, newStudent.ParentName,
ParameterDirection.Input),
new OracleParameter("parentContact", OracleDbType.Varchar2, newStudent.ParentContact,
ParameterDirection.Input),
new OracleParameter("enrollmentDate", OracleDbType.Date, newStudent.EnrollmentDate,
ParameterDirection.Input)
    };

    int rowsAffected = SchoolDatabase.ExecuteNonQuery(insertQuery, parameters);


    string getStudentIdQuery = @"SELECT student_id FROM Student
                            WHERE first_name = :firstName
                            AND last_name = :lastName
                            AND date_of_birth = :dateOfBirth
                            AND gender = :gender
                            AND address = :address
                            AND parent_name = :parentName
                            AND parent_contact = :parentContact
                            AND enrollment_date = :enrollmentDate";

    OracleParameter[] studentIdParams = new OracleParameter[]
    {
new OracleParameter("firstName", OracleDbType.Varchar2, newStudent.FirstName,
ParameterDirection.Input),
new OracleParameter("lastName", OracleDbType.Varchar2, newStudent.LastName,
ParameterDirection.Input),
new OracleParameter("dateOfBirth", OracleDbType.Date, newStudent.DateOfBirth,
ParameterDirection.Input),
new OracleParameter("gender", OracleDbType.Varchar2, newStudent.Gender,
ParameterDirection.Input),
new OracleParameter("address", OracleDbType.Varchar2, newStudent.Address,
ParameterDirection.Input),
new OracleParameter("parentName", OracleDbType.Varchar2, newStudent.ParentName,
ParameterDirection.Input),
new OracleParameter("parentContact", OracleDbType.Varchar2, newStudent.ParentContact,
ParameterDirection.Input),
new OracleParameter("enrollmentDate", OracleDbType.Date, newStudent.EnrollmentDate,
ParameterDirection.Input)
    };

    DataTable studentIdData = SchoolDatabase.ExecuteQuery(getStudentIdQuery,
studentIdParams);

    if (studentIdData.Rows.Count > 0) {
```

```
        if (rowsAffected > 0) {
            MessageBox.Show("Student added successfully!");
            this.DialogResult = DialogResult.OK;
            this.Close();
        }
        else {
            MessageBox.Show("An error occurred while adding the student.");
        }
    }
    else {
        MessageBox.Show("An error occurred while getting the new student ID.");
    }
}
```

3.2.2.3. نموذج تعديل طالب



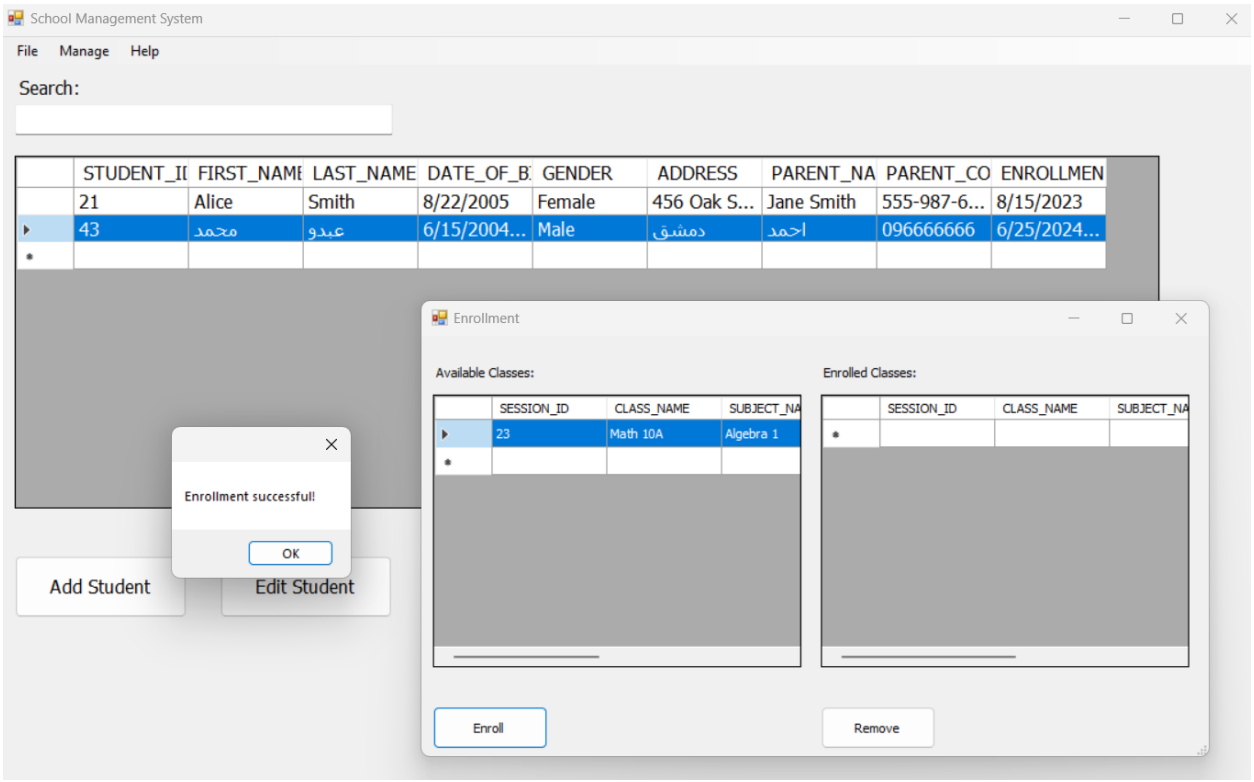
الكود:

```
public EditStudentForm(int studentId) {
    InitializeComponent();
    this.studentId = studentId;
    LoadStudentData();
}
```

```
private void LoadStudentData() {

    string query = "SELECT * FROM Student WHERE student_id = :studentId";
    OracleParameter parameter = new OracleParameter("studentId", OracleDbType.Int32,
studentId, ParameterDirection.Input);

    DataTable studentData = SchoolDatabase.ExecuteQuery(query, new OracleParameter[] {
parameter });

    if (studentData.Rows.Count > 0) {
        DataRow row = studentData.Rows[0];
        firstNameTextBox.Text = row["first_name"].ToString();
        lastNameTextBox.Text = row["last_name"].ToString();
        dateOfBirthDateTimePicker.Value = Convert.ToDateTime(row["date_of_birth"]);
        genderComboBox.SelectedItem = row["gender"].ToString();
        addressTextBox.Text = row["address"].ToString();
        parentNameTextBox.Text = row["parent_name"].ToString();
        parentContactTextBox.Text = row["parent_contact"].ToString();
    }
    else {
        MessageBox.Show("Student not found.");
    }
}

private void updateButton_Click(object sender, EventArgs e) {

    Student updatedStudent = new Student {
        StudentId = studentId,
        FirstName = firstNameTextBox.Text,
        LastName = lastNameTextBox.Text,
        DateOfBirth = dateOfBirthDateTimePicker.Value,
        Gender = genderComboBox.SelectedItem.ToString(),
        Address = addressTextBox.Text,
        ParentName = parentNameTextBox.Text,
        ParentContact = parentContactTextBox.Text
    };

    string updateQuery = @"UPDATE Student
                           SET first_name = :firstName,
                               last_name = :lastName,
                               date_of_birth = :dateOfBirth,
                               gender = :gender,
                               address = :address,
                               parent_name = :parentName,
                               parent_contact = :parentContact
                           WHERE student_id = :studentId";

    OracleParameter[] parameters = new OracleParameter[]
    {
        new OracleParameter("firstName", OracleDbType.Varchar2, updatedStudent.FirstName,
ParameterDirection.Input),
        new OracleParameter("lastName", OracleDbType.Varchar2, updatedStudent.LastName,
ParameterDirection.Input),
        new OracleParameter("dateOfBirth", OracleDbType.Date, updatedStudent.DateOfBirth,
ParameterDirection.Input),
        new OracleParameter("gender", OracleDbType.Varchar2, updatedStudent.Gender,
ParameterDirection.Input),
```

```
        new OracleParameter("address", OracleDbType.Varchar2, updatedStudent.Address,
ParameterDirection.Input),
        new OracleParameter("parentName", OracleDbType.Varchar2,
updatedStudent.ParentName, ParameterDirection.Input),
        new OracleParameter("parentContact", OracleDbType.Varchar2,
updatedStudent.ParentContact, ParameterDirection.Input),
        new OracleParameter("studentId", OracleDbType.Int32, updatedStudent.StudentId,
ParameterDirection.Input)
    };

    int rowsAffected = SchoolDatabase.ExecuteNonQuery(updateQuery, parameters);


    if (rowsAffected > 0) {
        MessageBox.Show("Student updated successfully!");
        this.DialogResult = DialogResult.OK;
        this.Close();
    }
    else {
        MessageBox.Show("An error occurred while updating the student.");
    }
}
```

2.2.3.4. نموذج تسجيل طالب



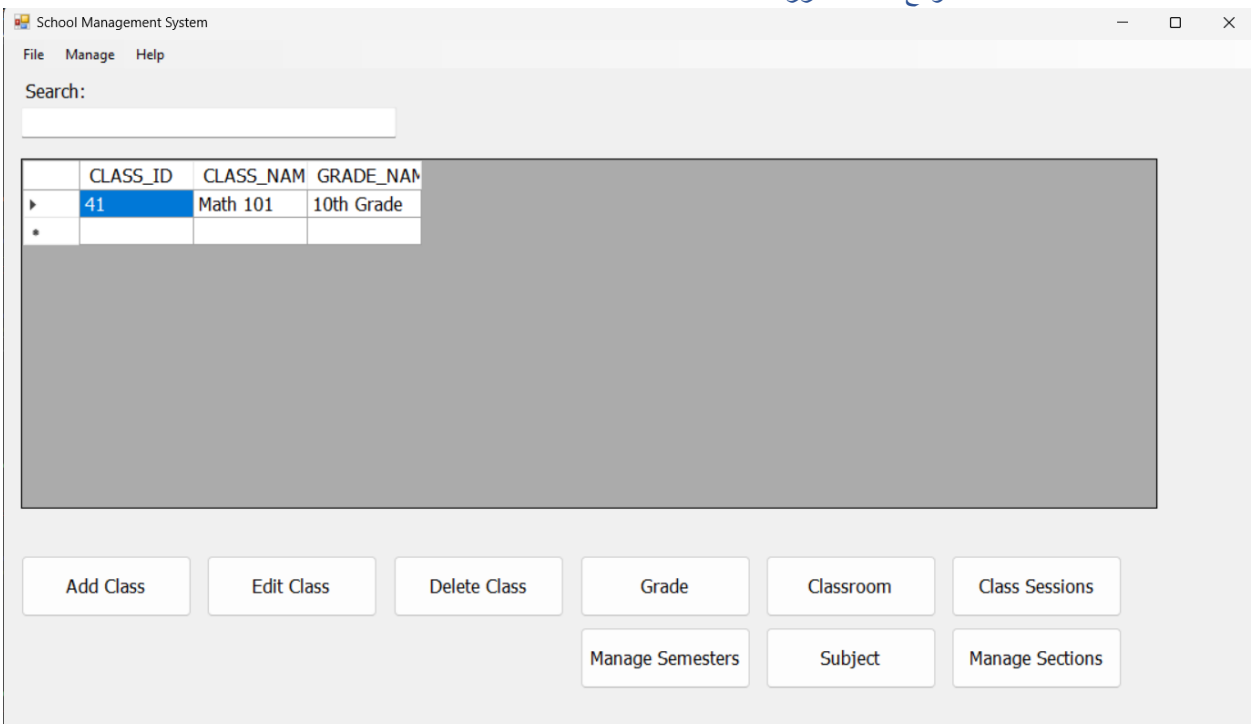
الكود:

```
public partial class EnrollmentForm : Form {
        private int studentId;
        private void EnrollmentForm_Load(object sender, EventArgs e) {

        }
        public EnrollmentForm(int studentId) {
            InitializeComponent();
            this.studentId = studentId;
            LoadAvailableClasses();
            LoadEnrolledClasses();
        }
        private void LoadAvailableClasses() {
            string query = @"SELECT
                                cs.session_id,
                                c.class_name,
                                s.subject_name,
                                g.grade_name,
                                se.semester_id,
                                se.semester_name
                            FROM ClassSession cs
                            JOIN Class c ON cs.class_id = c.class_id
                            JOIN Subject s ON cs.subject_id = s.subject_id
                            JOIN Grade g ON c.grade_id = g.grade_id
                            JOIN Semester se ON cs.semester_id = se.semester_id";

            DataTable availableClassesData = SchoolDatabase.ExecuteQuery(query);
            availableClassesDataGridView.DataSource = availableClassesData;
        }

        private void LoadEnrolledClasses() {
            string query = @"SELECT
                                cs.session_id,
                                c.class_name,
                                s.subject_name,
                                g.grade_name,
                                se.semester_name
                            FROM Enrollment e
                            JOIN ClassSession cs ON e.session_id = cs.session_id
                            JOIN Class c ON cs.class_id = c.class_id
                            JOIN Subject s ON cs.subject_id = s.subject_id
                            JOIN Grade g ON c.grade_id = g.grade_id
                            JOIN Semester se ON cs.semester_id = se.semester_id
                            WHERE e.student_id = :studentId";

            OracleParameter studentIdParam = new OracleParameter("studentId",
OracleDbType.Int32, studentId, ParameterDirection.Input);
            DataTable enrolledClassesData = SchoolDatabase.ExecuteQuery(query, new
OracleParameter[] { studentIdParam });
            enrolledClassesDataGridView.DataSource = enrolledClassesData;
        }

        private void enrollButton_Click(object sender, EventArgs e) {
            if (availableClassesDataGridView.SelectedRows.Count == 0) {
                MessageBox.Show("Please select a class to enroll in.");
                return;
            }
```

```
            int selectedSessionId =
Convert.ToInt32(availableClassesDataGridView.SelectedRows[0].Cells["session_id"].Value);
            int selectedSemesterId =
Convert.ToInt32(availableClassesDataGridView.SelectedRows[0].Cells["semester_id"].Value);
            if (enrolledClassesDataGridView.Rows.Cast<DataGridViewRow>().Any(row =>
Convert.ToInt32(row.Cells["session_id"].Value) == selectedSessionId)) {
                MessageBox.Show("The student is already enrolled in this class.");
                return;
            }
            string insertQuery = "INSERT INTO Enrollment (student_id, session_id,
semester_id) VALUES (:studentId, :sessionId, :semesterId)";
            OracleParameter[] parameters = new OracleParameter[]
            {
        new OracleParameter("studentId", OracleDbType.Int32, studentId,
ParameterDirection.Input),
        new OracleParameter("sessionId", OracleDbType.Int32, selectedSessionId,
ParameterDirection.Input),
        new OracleParameter("semesterId", OracleDbType.Int32, selectedSemesterId,
ParameterDirection.Input)
            };
            int rowsAffected = SchoolDatabase.ExecuteNonQuery(insertQuery, parameters);
            if (rowsAffected > 0) {
                MessageBox.Show("Enrollment successful!");
                LoadEnrolledClasses();
                LoadAvailableClasses();
            }
            else {
                MessageBox.Show("An error occurred while enrolling the student.");
            }
        }

        private void removeButton_Click(object sender, EventArgs e) {
            if (enrolledClassesDataGridView.SelectedRows.Count == 0) {
                MessageBox.Show("Please select a class to remove the student from.");
                return;
            }
            int selectedSessionId =
Convert.ToInt32(enrolledClassesDataGridView.SelectedRows[0].Cells["session_id"].Value);
            string deleteQuery = "DELETE FROM Enrollment WHERE student_id = :studentId
AND session_id = :sessionId";
            OracleParameter[] parameters = new OracleParameter[]
            {
                new OracleParameter("studentId", OracleDbType.Int32, studentId,
ParameterDirection.Input),
                new OracleParameter("sessionId", OracleDbType.Int32, selectedSessionId,
ParameterDirection.Input)
            };
            int rowsAffected = SchoolDatabase.ExecuteNonQuery(deleteQuery, parameters);
            if (rowsAffected > 0) {
                MessageBox.Show("Enrollment removed successfully!");
                LoadEnrolledClasses();
                LoadAvailableClasses();
            }
            else {
                MessageBox.Show("An error occurred while removing the enrollment.");
            }
        }
    }
```

2.2.4. نماذج المقررات: إدارة المقررات و الصفوف

2.2.4.1. نموذج قائمة المقررات



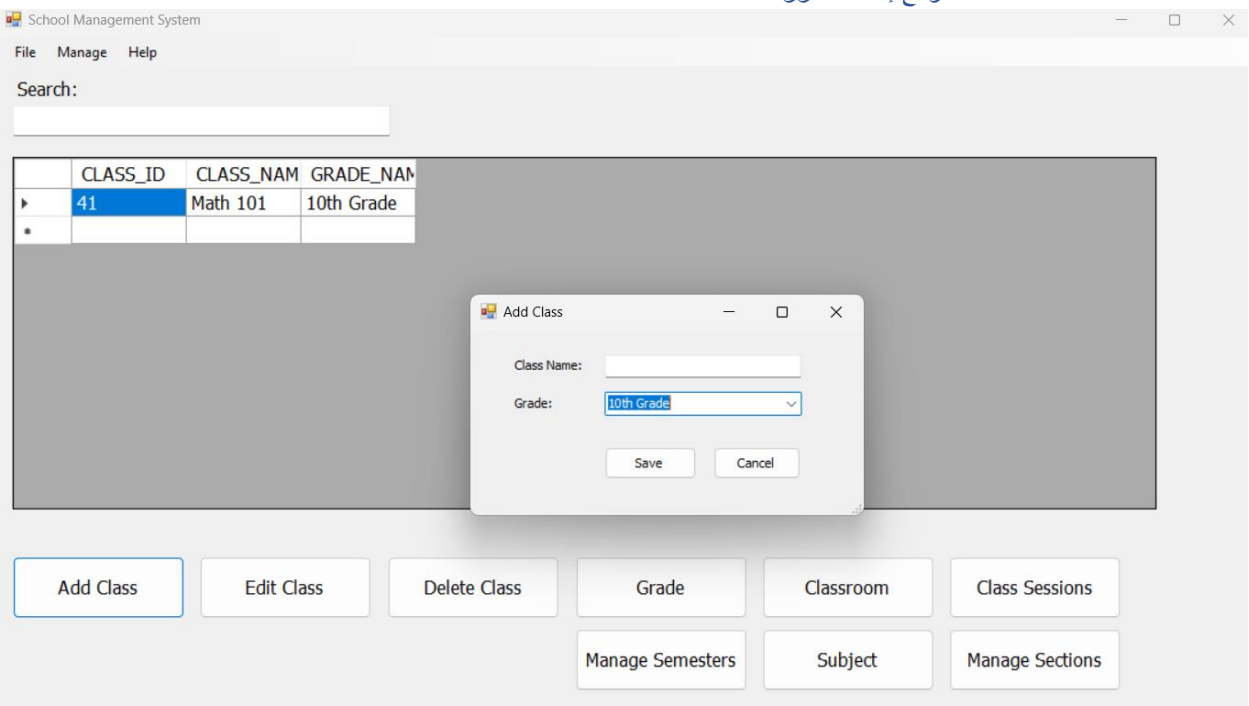
الكود:

```
public partial class ClassListForm : Form {
    public ClassListForm() {
        InitializeComponent();
    }

    private void ClassListForm_Load(object sender, EventArgs e) {
        string query = @"SELECT
                    C.class_id,
                    C.class_name,
                    G.grade_name
                FROM Class C
                JOIN Grade G ON C.grade_id = G.grade_id";

        DataTable classDataTable = SchoolDatabase.ExecuteQuery(query);
        classDataGridView.DataSource = classDataTable;
    }

    private void searchTextBox_TextChanged(object sender, EventArgs e) {
        string searchTerm = searchTextBox.Text.Trim();
        string query = @"SELECT * FROM Class WHERE class_name LIKE :searchTerm";
        OracleParameter searchTermParameter = new OracleParameter(":searchTerm",
OracleDbType.Varchar2);
        searchTermParameter.Value = "%" + searchTerm + "%";
        DataTable searchResults = SchoolDatabase.ExecuteQuery(query, new
OracleParameter[] { searchTermParameter });
        classDataGridView.DataSource = searchResults;
    }
    private void classroomButton_Click(object sender, EventArgs e) {
```

```
        ClassroomForm classroomForm = new ClassroomForm();
        classroomForm.ShowDialog();
    }

    private void deleteButton_Click(object sender, EventArgs e) {
        if (classDataGridView.SelectedRows.Count == 0) {
            MessageBox.Show("Please select a class to delete.");
            return;
        }
        int classId =
Convert.ToInt32(classDataGridView.SelectedRows[0].Cells["class_id"].Value);
        DialogResult result = MessageBox.Show(
            $"Are you sure you want to delete class with ID {classId}?",
            "Confirm Deletion",
            MessageBoxButtons.YesNo,
            MessageBoxIcon.Warning);
        if (result == DialogResult.No) {
            return;
        }
        try {
            OracleParameter classIdParameter = new OracleParameter("class_id", classId);
            int rowsAffected = SchoolDatabase.ExecuteNonQuery(
                "DELETE FROM Class WHERE class_id = :class_id",
                new OracleParameter[] { classIdParameter });
            if (rowsAffected > 0) {
                MessageBox.Show("Class deleted successfully.");

                RefreshClassList();
            }
            else {
                MessageBox.Show("An error occurred while deleting the class.");
            }
        }
        catch (Exception ex) {
            MessageBox.Show("Error: " + ex.Message);
        }
    }

    private void gradeButton_Click(object sender, EventArgs e) {
        GradeForm gradeForm = new GradeForm();
        gradeForm.ShowDialog();
    }

    private void editButton_Click(object sender, EventArgs e) {
        if (classDataGridView.SelectedRows.Count == 0) {
            MessageBox.Show("Please select a class to edit.");
            return;
        }
        int selectedClassId =
Convert.ToInt32(classDataGridView.SelectedRows[0].Cells["class_id"].Value);
        EditClassForm editClassForm = new EditClassForm(selectedClassId);
        DialogResult result = editClassForm.ShowDialog();
        if (result == DialogResult.OK) {
            RefreshClassList();
        }
    }

    private void addButton_Click(object sender, EventArgs e) {
```

```
        AddClassForm addClassForm = new AddClassForm();
        DialogResult result = addClassForm.ShowDialog();
        if (result == DialogResult.OK) {
            DataTable classDataTable = SchoolDatabase.ExecuteQuery("select * from
Class");
            classDataGridView.DataSource = classDataTable;
        }
    }

    private void RefreshClassList() {
        DataTable classDataTable = SchoolDatabase.ExecuteQuery("SELECT * FROM Class");
        classDataGridView.DataSource = classDataTable;
    }

    private void classSessionButton_Click(object sender, EventArgs e) {
        ClassSessionForm classSessionForm = new ClassSessionForm();
        classSessionForm.ShowDialog();
    }
    private void manageSectionsButton_Click(object sender, EventArgs e) {
        SectionForm sectionForm = new SectionForm();
        sectionForm.ShowDialog();
    }
    private void subjectButton_Click(object sender, EventArgs e) {
        SubjectForm subjectForm = new SubjectForm();
        subjectForm.ShowDialog();
    }
    private void semesterButton_Click(object sender, EventArgs e) {
        SemesterForm semesterForm = new SemesterForm();
        semesterForm.ShowDialog();
    }
}
```

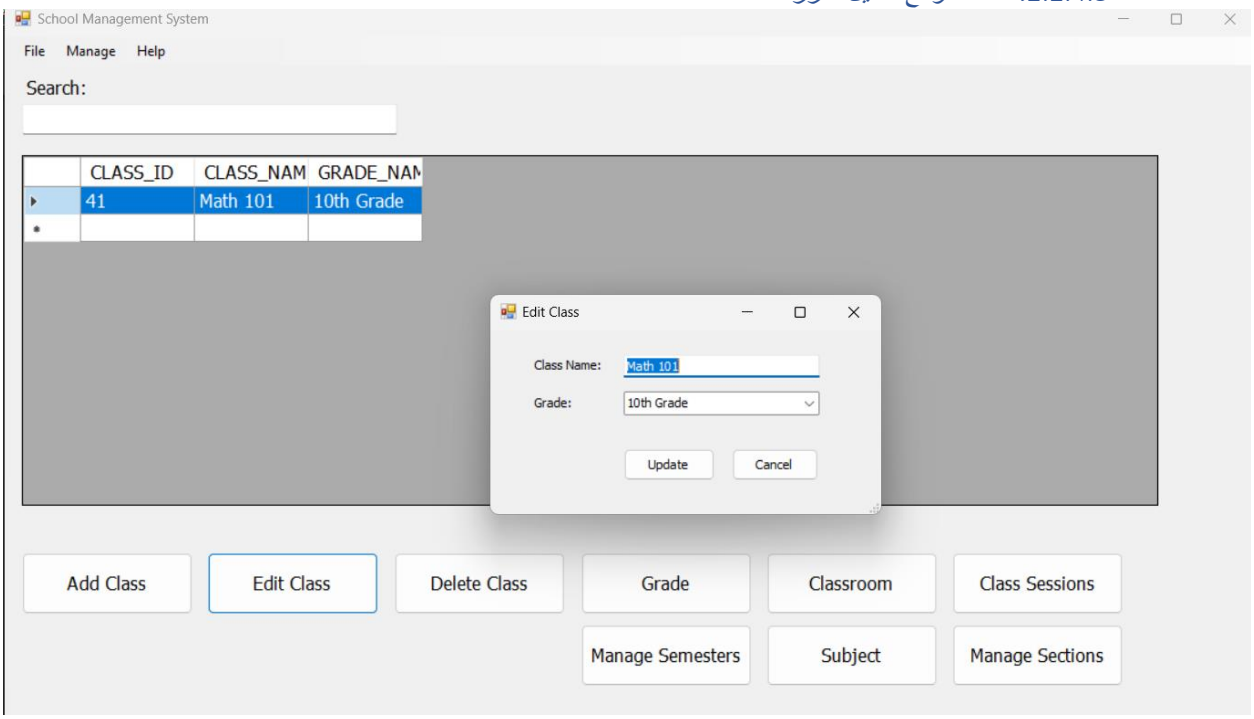
2.2.4.2. نموذج إضافة مقرر



الكود:

```
public partial class AddClassForm : Form {
    public AddClassForm() {
        InitializeComponent();
        LoadGradeLevels();
    }

    private void AddClassForm_Load(object sender, EventArgs e) {
    }

    private void LoadGradeLevels() {
        DataTable gradeLevels = SchoolDatabase.ExecuteQuery("SELECT * FROM Grade");
        gradeComboBox.DataSource = gradeLevels;
        gradeComboBox.DisplayMember = "grade_name";
        gradeComboBox.ValueMember = "grade_id";
    }

    private void saveButton_Click(object sender, EventArgs e) {
        if (string.IsNullOrEmpty(classNameTextBox.Text)) {
            MessageBox.Show("Please enter a class name.");
            return;
        }
        Classes newClass = new Classes {
            ClassName = classNameTextBox.Text,
            GradeId = Convert.ToInt32(gradeComboBox.SelectedValue)
        };
        string insertQuery = @"INSERT INTO Class (class_name, grade_id) VALUES
(:className, :gradeId)";
        OracleParameter[] parameters = new OracleParameter[]
        {
            new OracleParameter("className", OracleDbType.Varchar2, newClass.ClassName,
ParameterDirection.Input),
            new OracleParameter("gradeId", OracleDbType.Int32, newClass.GradeId,
ParameterDirection.Input)
        };
        int rowsAffected = SchoolDatabase.ExecuteNonQuery(insertQuery, parameters);
        if (rowsAffected > 0) {
            MessageBox.Show("Class added successfully!");
            this.DialogResult = DialogResult.OK;
            this.Close();
        }
        else {
            MessageBox.Show("An error occurred while adding the class.");
        }
    }

    private void cancelButton_Click(object sender, EventArgs e) {
        this.Close();
    }
}
```

2.2.4.3. نموذج تعديل مقرر



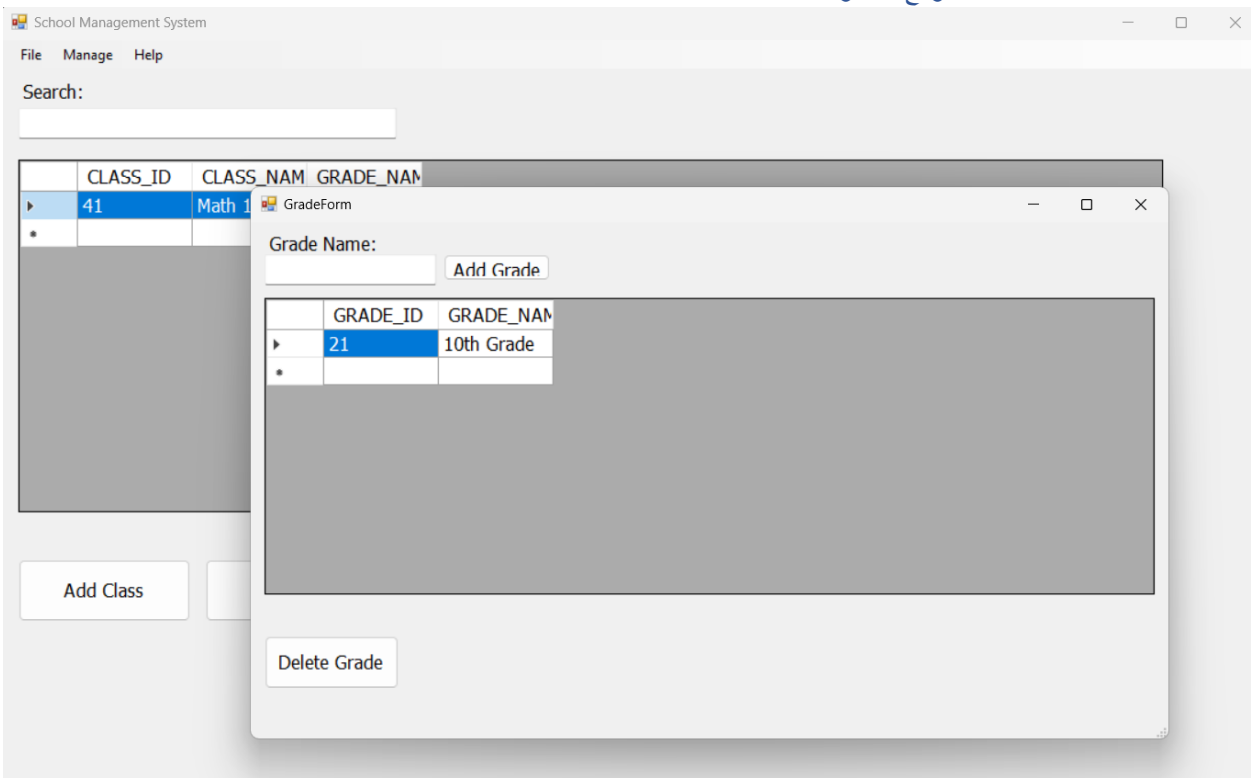
الكود:

```
public partial class EditClassForm : Form {
    private int classId;

    public EditClassForm(int classId) {
        InitializeComponent();
        this.classId = classId;
        LoadClassData();
    }

    private void EditClassForm_Load(object sender, EventArgs e) {
        DataTable grades = SchoolDatabase.ExecuteQuery("SELECT grade_id, grade_name FROM
Grade");
        gradeComboBox.DataSource = grades;
        gradeComboBox.DisplayMember = "grade_name";
        gradeComboBox.ValueMember = "grade_id";
    }

    private void LoadClassData() {

        string query = "SELECT * FROM Class WHERE class_id = :classId";
        OracleParameter parameter = new OracleParameter("classId", OracleDbType.Int32,
classId, ParameterDirection.Input);

        DataTable classData = SchoolDatabase.ExecuteQuery(query, new OracleParameter[] {
parameter });

        if (classData.Rows.Count > 0) {
            DataRow row = classData.Rows[0];
```

```
            classNameTextBox.Text = row["class_name"].ToString();


            string gradeQuery = "SELECT grade_name FROM Grade WHERE grade_id = :gradeId";
            OracleParameter gradeIdParameter = new OracleParameter("gradeId",
OracleDbType.Int32, row["grade_id"], ParameterDirection.Input);

            DataTable gradeData = SchoolDatabase.ExecuteQuery(gradeQuery, new
OracleParameter[] { gradeIdParameter });

            if (gradeData.Rows.Count > 0) {
                gradeComboBox.SelectedItem = gradeData.Rows[0]["grade_name"].ToString();
            }
            else {
                MessageBox.Show("Grade not found for this class.");
            }
        }
        else {
            MessageBox.Show("Class not found.");
        }
    }

    private void updateButton_Click(object sender, EventArgs e) {
        string gradeName =
((DataRowView)gradeComboBox.SelectedItem).Row["grade_name"].ToString();

        int gradeId = GetGradeIdFromComboBox(gradeName);

        if (gradeId == -1) {

            MessageBox.Show("Please select a valid grade from the list.");
            return;
        }


        Classes updatedClass = new Classes {
            ClassId = classId,
            ClassName = classNameTextBox.Text,
            GradeId = gradeId
        };

        string updateQuery = @"UPDATE Class
                               SET class_name = :className,
                                   grade_id = :gradeId
                               WHERE class_id = :classId";

        OracleParameter[] parameters = new OracleParameter[]
        {
            new OracleParameter("className", OracleDbType.Varchar2,
updatedClass.ClassName, ParameterDirection.Input),
            new OracleParameter("gradeId", OracleDbType.Int32, updatedClass.GradeId,
ParameterDirection.Input),
            new OracleParameter("classId", OracleDbType.Int32, updatedClass.ClassId,
ParameterDirection.Input)
        };

        int rowsAffected = SchoolDatabase.ExecuteNonQuery(updateQuery, parameters);
```

```
        if (rowsAffected > 0) {
            MessageBox.Show("Class updated successfully!");
            this.DialogResult = DialogResult.OK;
            this.Close();
        }
        else {
            MessageBox.Show("An error occurred while updating the class.");
        }
    }

    private void cancelButton_Click(object sender, EventArgs e) {
        this.Close();
    }

    private int GetGradeIdFromComboBox(string gradeName) {

        string query = "SELECT grade_id FROM Grade WHERE grade_name = :gradeName";
        OracleParameter parameter = new OracleParameter("gradeName",
OracleDbType.Varchar2, gradeName, ParameterDirection.Input);

        DataTable gradeData = SchoolDatabase.ExecuteQuery(query, new OracleParameter[] {
parameter });

        if (gradeData.Rows.Count > 0) {
            return Convert.ToInt32(gradeData.Rows[0]["grade_id"]);
        }
        else {

            MessageBox.Show($"The grade '{gradeName}' does not exist. Please add the
grade first or select a valid grade.");
            return -1;
        }
    }
}
```

2.2.4.4. نموذج الصفوف

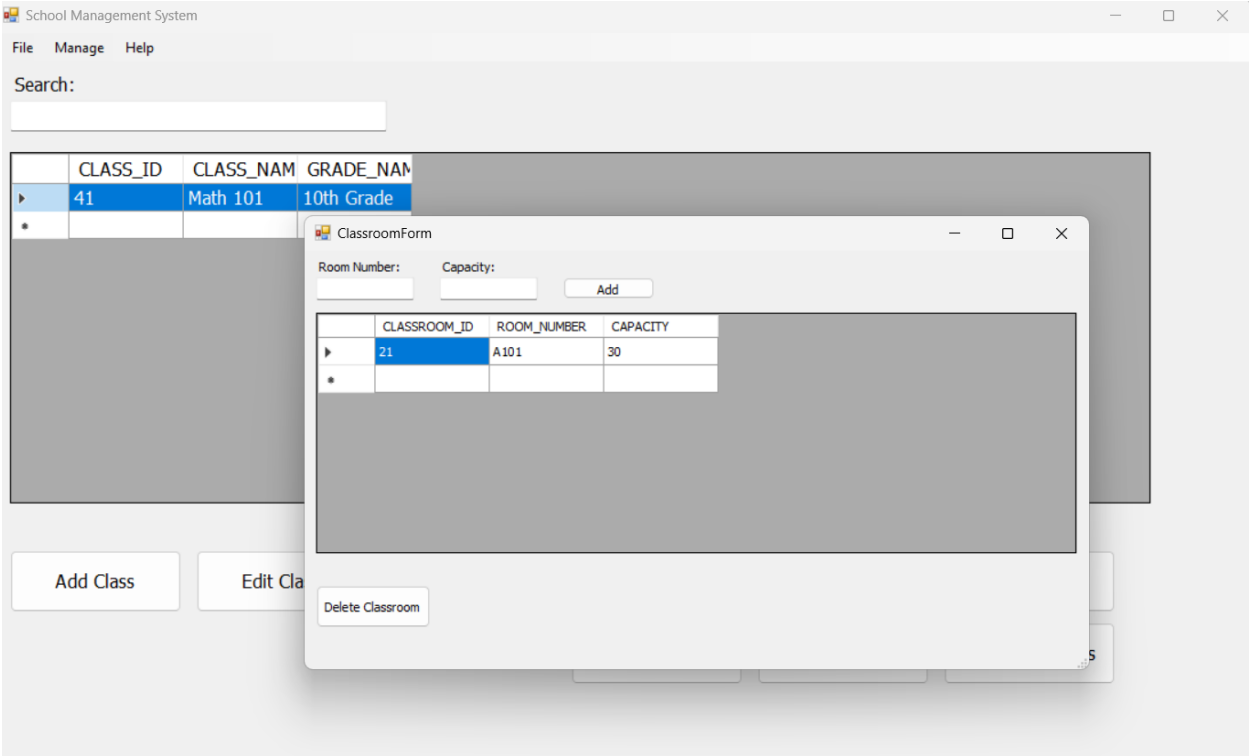


الكود:

```
public partial class GradeForm : Form {
    private void GradeForm_Load(object sender, EventArgs e) {

    }
    public GradeForm() {
        InitializeComponent();
        RefreshGradeList();
    }

    private void addGradeButton_Click(object sender, EventArgs e) {
        string gradeName = gradeNameTextBox.Text.Trim();
        if (string.IsNullOrEmpty(gradeName)) {
            MessageBox.Show("Please enter a grade name.");
            return;
        }

        string insertQuery = @"INSERT INTO Grade (grade_name) VALUES (:gradeName)";
        OracleParameter gradeNameParameter = new OracleParameter(":gradeName",
OracleDbType.Varchar2, gradeName, ParameterDirection.Input);
        int rowsAffected = SchoolDatabase.ExecuteNonQuery(insertQuery, new
OracleParameter[] { gradeNameParameter });

        if (rowsAffected > 0) {
            MessageBox.Show("Grade added successfully.");
```

```
            gradeNameTextBox.Clear();
            RefreshGradeList();
        }
        else {
            MessageBox.Show("An error occurred while adding the grade.");
        }
    }

    private void deleteGradeButton_Click(object sender, EventArgs e) {

        if (gradeDataGridView.SelectedRows.Count == 0) {
            MessageBox.Show("Please select a grade to delete.");
            return;
        }

        int gradeId =
Convert.ToInt32(gradeDataGridView.SelectedRows[0].Cells["grade_id"].Value);


        DialogResult result = MessageBox.Show(
            $"Are you sure you want to delete grade with ID {gradeId}?",
            "Confirm Deletion",
            MessageBoxButtons.YesNo,
            MessageBoxIcon.Warning);

        if (result == DialogResult.No) {
            return;
        }


        string deleteQuery = "DELETE FROM Grade WHERE grade_id = :gradeId";
        OracleParameter gradeIdParameter = new OracleParameter(":gradeId", gradeId);
        int rowsAffected = SchoolDatabase.ExecuteNonQuery(deleteQuery, new
OracleParameter[] { gradeIdParameter });

        if (rowsAffected > 0) {
            MessageBox.Show("Grade deleted successfully.");
            RefreshGradeList();
        }
        else {
            MessageBox.Show("An error occurred while deleting the grade.");
        }
    }

    private void RefreshGradeList() {
        DataTable gradeDataTable = SchoolDatabase.ExecuteQuery("SELECT * FROM Grade");
        gradeDataGridView.DataSource = gradeDataTable;
    }
}
```

5.2.4.2. نموذج غرف الصفوف



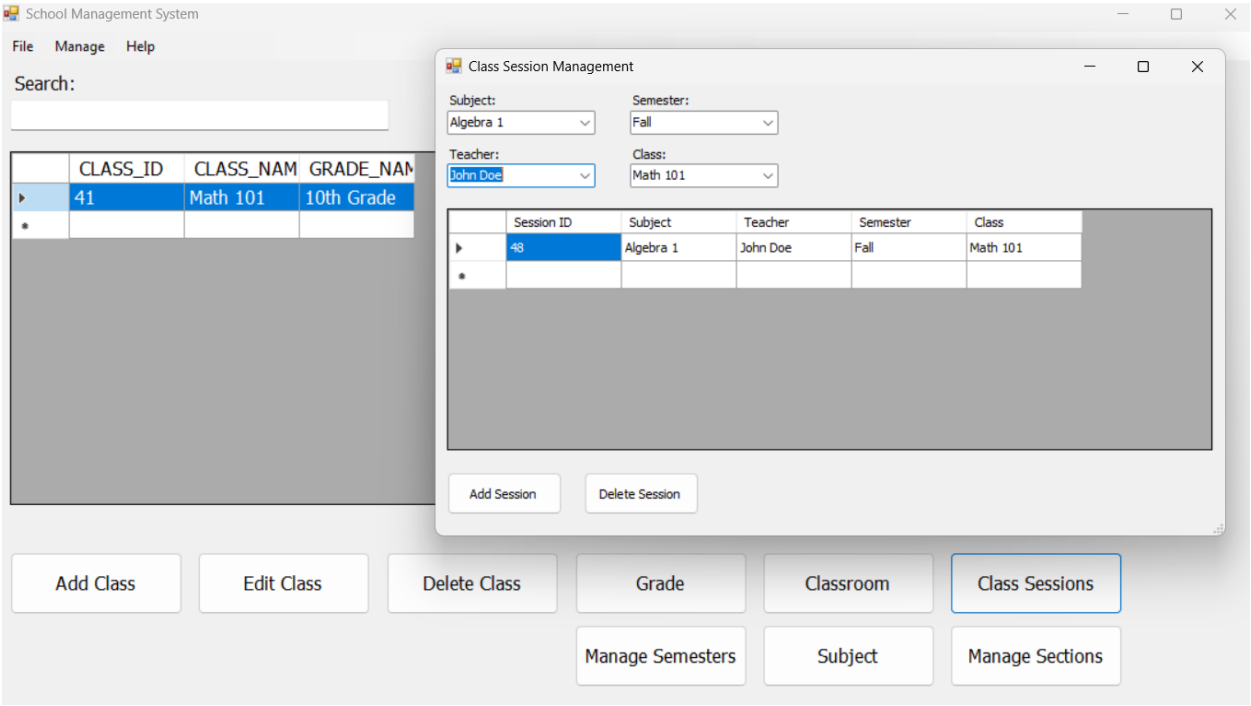
الكود:

```
public partial class ClassroomForm : Form {
    public ClassroomForm() {
        InitializeComponent();
        RefreshClassroomList();
    }

    private void addClassroomButton_Click(object sender, EventArgs e) {

        string roomNumber = roomNumberTextBox.Text.Trim();
        if (string.IsNullOrEmpty(roomNumber)) {
            MessageBox.Show("Please enter a room number.");
            return;
        }

        int capacity;
        if (!int.TryParse(capacityTextBox.Text, out capacity)) {
            MessageBox.Show("Please enter a valid capacity (integer).");
            return;
        }


        string insertQuery = @"INSERT INTO Classroom (room_number, capacity) VALUES
(:roomNumber, :capacity)";
```

```
        OracleParameter roomNumberParameter = new OracleParameter(":roomNumber",
OracleDbType.Varchar2, roomNumber, ParameterDirection.Input);
        OracleParameter capacityParameter = new OracleParameter(":capacity",
OracleDbType.Int32, capacity, ParameterDirection.Input);
        int rowsAffected = SchoolDatabase.ExecuteNonQuery(insertQuery, new
OracleParameter[] { roomNumberParameter, capacityParameter });

        if (rowsAffected > 0) {
            MessageBox.Show("Classroom added successfully.");
            roomNumberTextBox.Clear();
            capacityTextBox.Clear();
            RefreshClassroomList();
        }
        else {
            MessageBox.Show("An error occurred while adding the classroom.");
        }
    }

    private void deleteClassroomButton_Click(object sender, EventArgs e) {

        if (classroomDataGridView.SelectedRows.Count == 0) {
            MessageBox.Show("Please select a classroom to delete.");
            return;
        }

        int classroomId =
Convert.ToInt32(classroomDataGridView.SelectedRows[0].Cells["classroom_id"].Value);


        DialogResult result = MessageBox.Show(
            $"Are you sure you want to delete classroom with ID {classroomId}?",
            "Confirm Deletion",
            MessageBoxButtons.YesNo,
            MessageBoxIcon.Warning);

        if (result == DialogResult.No) {
            return;
        }


        string deleteQuery = "DELETE FROM Classroom WHERE classroom_id = :classroomId";
        OracleParameter classroomIdParameter = new OracleParameter(":classroomId",
classroomId);
        int rowsAffected = SchoolDatabase.ExecuteNonQuery(deleteQuery, new
OracleParameter[] { classroomIdParameter });

        if (rowsAffected > 0) {
            MessageBox.Show("Classroom deleted successfully.");
            RefreshClassroomList();
        }
        else {
            MessageBox.Show("An error occurred while deleting the classroom.");
        }
    }

    private void RefreshClassroomList() {
        DataTable classroomDataTable = SchoolDatabase.ExecuteQuery("SELECT * FROM
Classroom");
```

```
        classroomDataGridView.DataSource = classroomDataTable;
    }
}
```

6.4.2.2. نموذج جلسة المقرر



الكود:

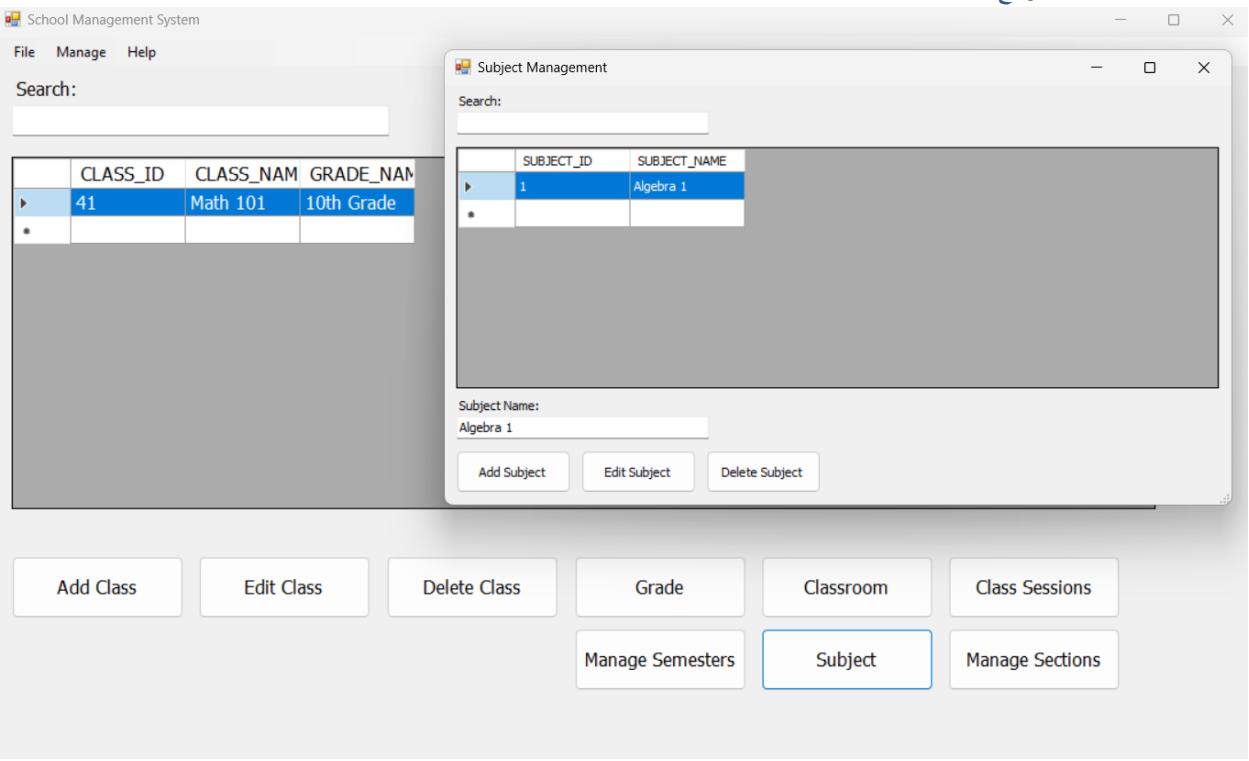
```
public partial class ClassSessionForm : Form {
    public ClassSessionForm() {
        InitializeComponent();
        LoadClassSessions();
        LoadSubjects();
        LoadTeachers();
        LoadSemesters();
        LoadClasses();
    }


    private void LoadClassSessions() {

        DataTable sessionData = SchoolDatabase.ExecuteQuery(
            @"SELECT
                cs.session_id,
                s.subject_name AS Subject,
                e.first_name || ' ' || e.last_name AS Teacher,
                sem.semester_name AS Semester,
                c.class_name AS Class
```

```
        FROM ClassSession cs
        JOIN Subject s ON cs.subject_id = s.subject_id
        JOIN Employee e ON cs.teacher_id = e.employee_id
        JOIN Semester sem ON cs.semester_id = sem.semester_id
        JOIN Class c ON cs.class_id = c.class_id");

    classSessionDataGridView.DataSource = sessionData;


    classSessionDataGridView.Columns["session_id"].HeaderText = "Session ID";
    classSessionDataGridView.Columns["Subject"].HeaderText = "Subject";
    classSessionDataGridView.Columns["Teacher"].HeaderText = "Teacher";
    classSessionDataGridView.Columns["Semester"].HeaderText = "Semester";
    classSessionDataGridView.Columns["Class"].HeaderText = "Class";
}

private void LoadSubjects() {
    DataTable subjectData = SchoolDatabase.ExecuteQuery("SELECT * FROM Subject");
    subjectComboBox.DataSource = subjectData;
    subjectComboBox.DisplayMember = "subject_name";
    subjectComboBox.ValueMember = "subject_id";
}

private void LoadTeachers() {

    DataTable teacherData = SchoolDatabase.ExecuteQuery(
        @"SELECT e.employee_id, e.first_name || ' ' || e.last_name AS EmployeeName
            FROM Employee e
            INNER JOIN EmployeeRole er ON e.employee_id = er.employee_id
            INNER JOIN Role r ON er.role_id = r.role_id
            WHERE r.role_name = 'Teacher'");

    teacherComboBox.DataSource = teacherData;
    teacherComboBox.DisplayMember = "EmployeeName";
    teacherComboBox.ValueMember = "employee_id";
}

private void LoadSemesters() {
    DataTable semesterData = SchoolDatabase.ExecuteQuery("SELECT * FROM Semester");
    semesterComboBox.DataSource = semesterData;
    semesterComboBox.DisplayMember = "semester_name";
    semesterComboBox.ValueMember = "semester_id";
}

private void LoadClasses() {
    DataTable classData = SchoolDatabase.ExecuteQuery("SELECT * FROM Class");
    classComboBox.DataSource = classData;
    classComboBox.DisplayMember = "class_name";
    classComboBox.ValueMember = "class_id";
}

private void addSessionButton_Click(object sender, EventArgs e) {

    int subjectId = Convert.ToInt32(subjectComboBox.SelectedValue);
    int teacherId = Convert.ToInt32(teacherComboBox.SelectedValue);
    int semesterId = Convert.ToInt32(semesterComboBox.SelectedValue);
    int classId = Convert.ToInt32(classComboBox.SelectedValue);
```

```
        string insertQuery = @"INSERT INTO ClassSession (subject_id, teacher_id,
semester_id, class_id)
                        VALUES (:subjectId, :teacherId, :semesterId, :classId)";

        OracleParameter[] parameters = new OracleParameter[]
        {
            new OracleParameter("subjectId", subjectId),
            new OracleParameter("teacherId", teacherId),
            new OracleParameter("semesterId", semesterId),
            new OracleParameter("classId", classId)
        };

        int rowsAffected = SchoolDatabase.ExecuteNonQuery(insertQuery, parameters);

        if (rowsAffected > 0) {
            MessageBox.Show("Class session added successfully.");
            LoadClassSessions();
        }
        else {
            MessageBox.Show("An error occurred while adding the class session.");
        }
    }

    private void deleteSessionButton_Click(object sender, EventArgs e) {

        if (classSessionDataGridView.SelectedRows.Count == 0) {
            MessageBox.Show("Please select a class session to delete.");
            return;
        }

        int sessionId =
Convert.ToInt32(classSessionDataGridView.SelectedRows[0].Cells["session_id"].Value);


        DialogResult result = MessageBox.Show(
            $"Are you sure you want to delete class session with ID {sessionId}?",
            "Confirm Deletion",
            MessageBoxButtons.YesNo,
            MessageBoxIcon.Warning);

        if (result == DialogResult.No) {
            return;
        }


        string deleteQuery = "DELETE FROM ClassSession WHERE session_id = :sessionId";
        OracleParameter sessionIdParameter = new OracleParameter(":sessionId",
sessionId);
        int rowsAffected = SchoolDatabase.ExecuteNonQuery(deleteQuery, new
OracleParameter[] { sessionIdParameter });

        if (rowsAffected > 0) {
            MessageBox.Show("Class session deleted successfully.");
            LoadClassSessions();
        }
        else {
            MessageBox.Show("An error occurred while deleting the class session.");
```

```
        }
    }

    private void ClassSessionForm_Load(object sender, EventArgs e) {

    }
}
```

2.2.4.7. نموذج المادة



الكود:

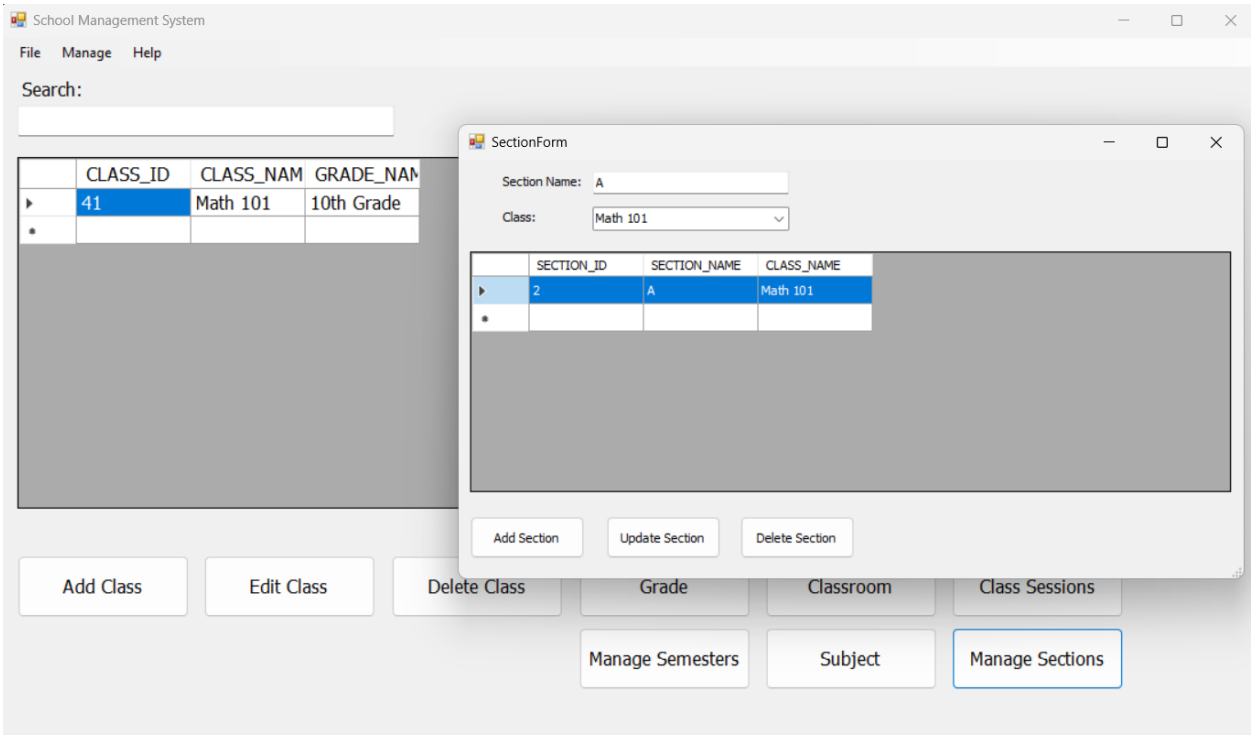
```
public partial class SubjectForm : Form {
    public SubjectForm() {
        InitializeComponent();
    }

    private void SubjectListForm_Load(object sender, EventArgs e) {

        DataTable subjectDataTable = SchoolDatabase.ExecuteQuery("SELECT * FROM
Subject");


        subjectDataGridView.DataSource = subjectDataTable;
    }

    private void searchTextBox_TextChanged(object sender, EventArgs e) {

        string searchTerm = searchTextBox.Text.Trim();
```

```
        string query = @"SELECT * FROM Subject WHERE subject_name LIKE :searchTerm";


        OracleParameter searchTermParameter = new OracleParameter(":searchTerm",
OracleDbType.Varchar2);
        searchTermParameter.Value = "%" + searchTerm + "%";


        DataTable searchResults = SchoolDatabase.ExecuteQuery(query, new
OracleParameter[] { searchTermParameter });


        subjectDataGridView.DataSource = searchResults;
    }

    private void addSubjectButton_Click(object sender, EventArgs e) {

        string newSubjectName = subjectNameTextBox.Text.Trim();

        if (string.IsNullOrEmpty(newSubjectName)) {
            MessageBox.Show("Please enter a subject name.");
            return;
        }


        string insertQuery = @"INSERT INTO Subject (subject_name) VALUES (:subjectName)";

        OracleParameter subjectNameParameter = new OracleParameter(":subjectName",
OracleDbType.Varchar2);
        subjectNameParameter.Value = newSubjectName;

        int rowsAffected = SchoolDatabase.ExecuteNonQuery(insertQuery, new
OracleParameter[] { subjectNameParameter });

        if (rowsAffected > 0) {
            MessageBox.Show("Subject added successfully!");
            RefreshSubjectList();
            subjectNameTextBox.Clear();
        }
        else {
            MessageBox.Show("An error occurred while adding the subject.");
        }
    }

    private void editSubjectButton_Click(object sender, EventArgs e) {

        if (subjectDataGridView.SelectedRows.Count == 0) {
            MessageBox.Show("Please select a subject to edit.");
            return;
        }


        int subjectId =
Convert.ToInt32(subjectDataGridView.SelectedRows[0].Cells["subject_id"].Value);
```

```
        string currentSubjectName =
subjectDataGridView.SelectedRows[0].Cells["subject_name"].Value.ToString();


        string updateQuery = @"UPDATE Subject
                               SET subject_name = :subjectName
                               WHERE subject_id = :subjectId";

        OracleParameter subjectNameParameter = new OracleParameter(":subjectName",
OracleDbType.Varchar2);
        subjectNameParameter.Value = subjectNameTextBox.Text.Trim();

        OracleParameter subjectIdParameter = new OracleParameter(":subjectId",
OracleDbType.Int32);
        subjectIdParameter.Value = subjectId;

        int rowsAffected = SchoolDatabase.ExecuteNonQuery(updateQuery, new
OracleParameter[] { subjectNameParameter, subjectIdParameter });

        if (rowsAffected > 0) {
            MessageBox.Show("Subject updated successfully!");
            RefreshSubjectList();
            subjectNameTextBox.Clear();
        }
        else {
            MessageBox.Show("An error occurred while updating the subject.");
        }
    }

    private void deleteSubjectButton_Click(object sender, EventArgs e) {

        if (subjectDataGridView.SelectedRows.Count == 0) {
            MessageBox.Show("Please select a subject to delete.");
            return;
        }
        int subjectId =
Convert.ToInt32(subjectDataGridView.SelectedRows[0].Cells["subject_id"].Value);


        DialogResult result = MessageBox.Show(
            $"Are you sure you want to delete subject with ID {subjectId}?",
            "Confirm Deletion",
            MessageBoxButtons.YesNo,
            MessageBoxIcon.Warning);

        if (result == DialogResult.No) {
            return;
        }


        try {

            OracleParameter subjectIdParameter = new OracleParameter("subject_id",
subjectId);


            int rowsAffected = SchoolDatabase.ExecuteNonQuery(
                "DELETE FROM Subject WHERE subject_id = :subject_id",
```

```
                new OracleParameter[] { subjectIdParameter });

            if (rowsAffected > 0) {
                MessageBox.Show("Subject deleted successfully.");

                RefreshSubjectList();
            }
            else {
                MessageBox.Show("An error occurred while deleting the subject.");
            }
        }
        catch (Exception ex) {
            MessageBox.Show("Error: " + ex.Message);
        }
    }

    private void RefreshSubjectList() {
        DataTable subjectDataTable = SchoolDatabase.ExecuteQuery("SELECT * FROM
Subject");
        subjectDataGridView.DataSource = subjectDataTable;
    }

    private void subjectDataGridView_SelectionChanged(object sender, EventArgs e) {
        if (subjectDataGridView.SelectedRows.Count > 0) {

            string selectedSubjectName =
subjectDataGridView.SelectedRows[0].Cells["subject_name"].Value.ToString();


            subjectNameTextBox.Text = selectedSubjectName;
        }
    }
}
```

8.4.2.2. نموذج الشعبة



الكود:

```
public partial class SectionForm : Form {
    public SectionForm() {
        InitializeComponent();
        LoadClassData();
    }

    private void SectionForm_Load(object sender, EventArgs e) {
        LoadSectionData();
    }

    private void LoadSectionData() {

        string query = @"SELECT s.*, c.class_name
                FROM Section s
                JOIN Class c ON s.class_id = c.class_id";


        DataTable sectionDataTable = SchoolDatabase.ExecuteQuery(query);


        sectionDataGridView.DataSource = sectionDataTable;

        sectionDataGridView.Columns["class_id"].Visible = false;
    }

    private void LoadClassData() {
```

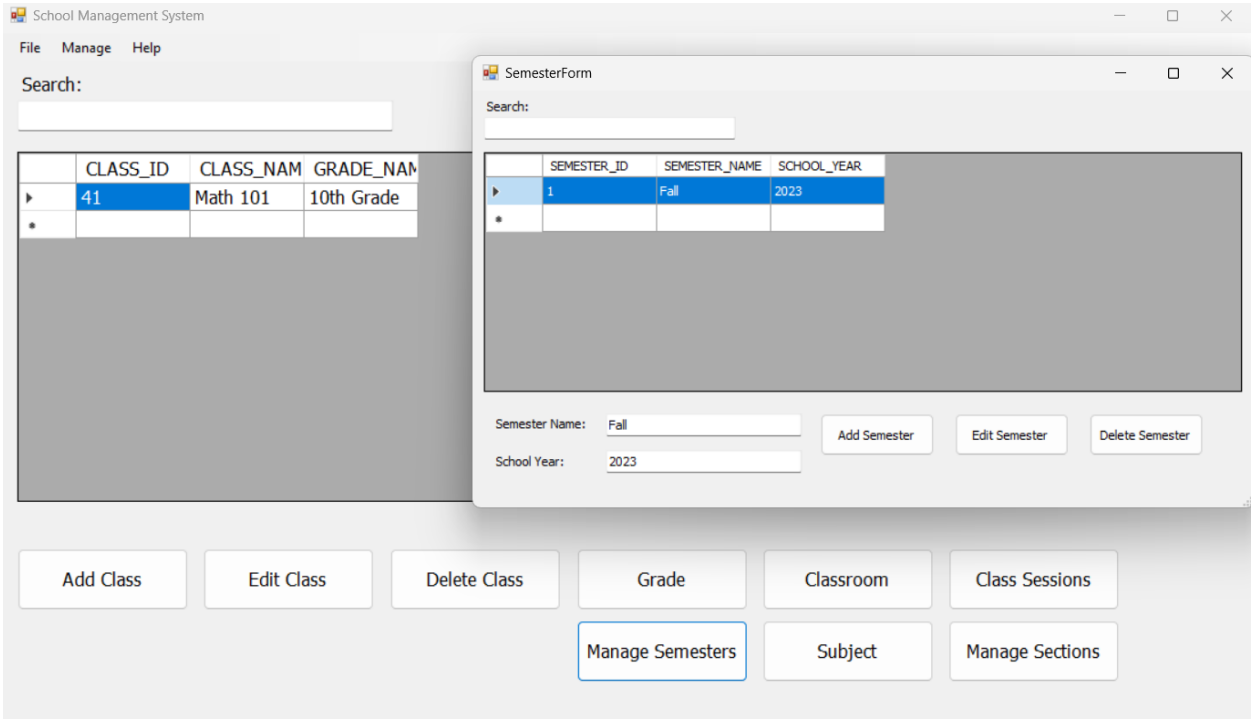
```
        DataTable classData = SchoolDatabase.ExecuteQuery("SELECT class_name FROM
Class");


        classComboBox.Items.Clear();
        foreach (DataRow row in classData.Rows) {
            classComboBox.Items.Add(row["class_name"].ToString());
        }
    }

    private void sectionDataGridView_SelectionChanged(object sender, EventArgs e) {
        if (sectionDataGridView.SelectedRows.Count > 0) {
            DataRowView selectedRow =
(DataRowView)sectionDataGridView.SelectedRows[0].DataBoundItem;
            sectionNameTextBox.Text = selectedRow["section_name"].ToString();


            int classId = Convert.ToInt32(selectedRow["class_id"]);


            string className = GetClassNameFromId(classId);


            classComboBox.SelectedItem = className;
        }
    }

    private void addButton_Click(object sender, EventArgs e) {
        string sectionName = sectionNameTextBox.Text;
        string className = classComboBox.SelectedItem.ToString();


        if (string.IsNullOrEmpty(sectionName) || string.IsNullOrEmpty(className)) {
            MessageBox.Show("Please fill in all fields.");
            return;
        }


        int classId = GetClassIdFromName(className);


        string insertQuery = "INSERT INTO Section (section_name, class_id) VALUES
(:sectionName, :classId)";
        OracleParameter[] parameters = new OracleParameter[]
        {
            new OracleParameter("sectionName", OracleDbType.Varchar2, sectionName,
ParameterDirection.Input),
            new OracleParameter("classId", OracleDbType.Int32, classId,
ParameterDirection.Input)
        };

        int rowsAffected = SchoolDatabase.ExecuteNonQuery(insertQuery, parameters);

        if (rowsAffected > 0) {
            MessageBox.Show("Section added successfully.");

            LoadSectionData();
        }
```

```
        else {
            MessageBox.Show("An error occurred while adding the section.");
        }
    }

    private void deleteButton_Click(object sender, EventArgs e) {
        if (sectionDataGridView.SelectedRows.Count == 0) {
            MessageBox.Show("Please select a section to delete.");
            return;
        }

        int sectionId =
Convert.ToInt32(sectionDataGridView.SelectedRows[0].Cells["section_id"].Value);

        DialogResult result = MessageBox.Show(
            $"Are you sure you want to delete section with ID {sectionId}?",
            "Confirm Deletion",
            MessageBoxButtons.YesNo,
            MessageBoxIcon.Warning);

        if (result == DialogResult.Yes) {

            string deleteQuery = "DELETE FROM Section WHERE section_id = :sectionId";
            OracleParameter sectionIdParameter = new OracleParameter("sectionId",
OracleDbType.Int32, sectionId, ParameterDirection.Input);
            int rowsAffected = SchoolDatabase.ExecuteNonQuery(deleteQuery, new
OracleParameter[] { sectionIdParameter });

            if (rowsAffected > 0) {
                MessageBox.Show("Section deleted successfully.");
                LoadSectionData();
            }
            else {
                MessageBox.Show("An error occurred while deleting the section.");
            }
        }
    }

    private void updateButton_Click(object sender, EventArgs e) {
        if (sectionDataGridView.SelectedRows.Count == 0) {
            MessageBox.Show("Please select a section to update.");
            return;
        }

        int sectionId =
Convert.ToInt32(sectionDataGridView.SelectedRows[0].Cells["section_id"].Value);
        string sectionName = sectionNameTextBox.Text;
        string className = classComboBox.SelectedItem.ToString();


        if (string.IsNullOrEmpty(sectionName) || string.IsNullOrEmpty(className)) {
            MessageBox.Show("Please fill in all fields.");
            return;
        }


        int classId = GetClassIdFromName(className);
```

```
        string updateQuery = "UPDATE Section SET section_name = :sectionName, class_id =
:classId WHERE section_id = :sectionId";
        OracleParameter[] parameters = new OracleParameter[]
        {
            new OracleParameter("sectionName", OracleDbType.Varchar2, sectionName,
ParameterDirection.Input),
            new OracleParameter("classId", OracleDbType.Int32, classId,
ParameterDirection.Input),
            new OracleParameter("sectionId", OracleDbType.Int32, sectionId,
ParameterDirection.Input)
        };

        int rowsAffected = SchoolDatabase.ExecuteNonQuery(updateQuery, parameters);

        if (rowsAffected > 0) {
            MessageBox.Show("Section updated successfully.");
            LoadSectionData();
        }
        else {
            MessageBox.Show("An error occurred while updating the section.");
        }
    }


    private int GetClassIdFromName(string className) {
        string selectClassIdQuery = "SELECT class_id FROM Class WHERE class_name =
:className";
        OracleParameter classNameParameter = new OracleParameter("className",
OracleDbType.Varchar2, className, ParameterDirection.Input);

        DataTable classIdData = SchoolDatabase.ExecuteQuery(selectClassIdQuery, new
OracleParameter[] { classNameParameter });

        if (classIdData.Rows.Count > 0) {
            return Convert.ToInt32(classIdData.Rows[0]["class_id"]);
        }
        else {
            MessageBox.Show($"Class '{className}' not found.");
            return -1;
        }
    }

    private string GetClassNameFromId(int classId) {
        string selectClassNameQuery = "SELECT class_name FROM Class WHERE class_id =
:classId";
        OracleParameter classIdParameter = new OracleParameter("classId",
OracleDbType.Int32, classId, ParameterDirection.Input);

        DataTable classNameData = SchoolDatabase.ExecuteQuery(selectClassNameQuery, new
OracleParameter[] { classIdParameter });

        if (classNameData.Rows.Count > 0) {
            return classNameData.Rows[0]["class_name"].ToString();
        }
        else {
            MessageBox.Show($"Class with ID {classId} not found.");
            return string.Empty;
```

```
        }
    }
}
```

9.2.4.2. نموذج الفصل الدراسي



الكود:

```
public partial class SemesterForm : Form {
    public SemesterForm() {
        InitializeComponent();
        LoadSemesterData();
    }

    private void SemesterForm_Load(object sender, EventArgs e) {
    }

    private void LoadSemesterData() {

        DataTable semesterDataTable = SchoolDatabase.ExecuteQuery("SELECT * FROM
Semester");


        semesterDataGridView.DataSource = semesterDataTable;
    }

    private void addButton_Click(object sender, EventArgs e) {
```

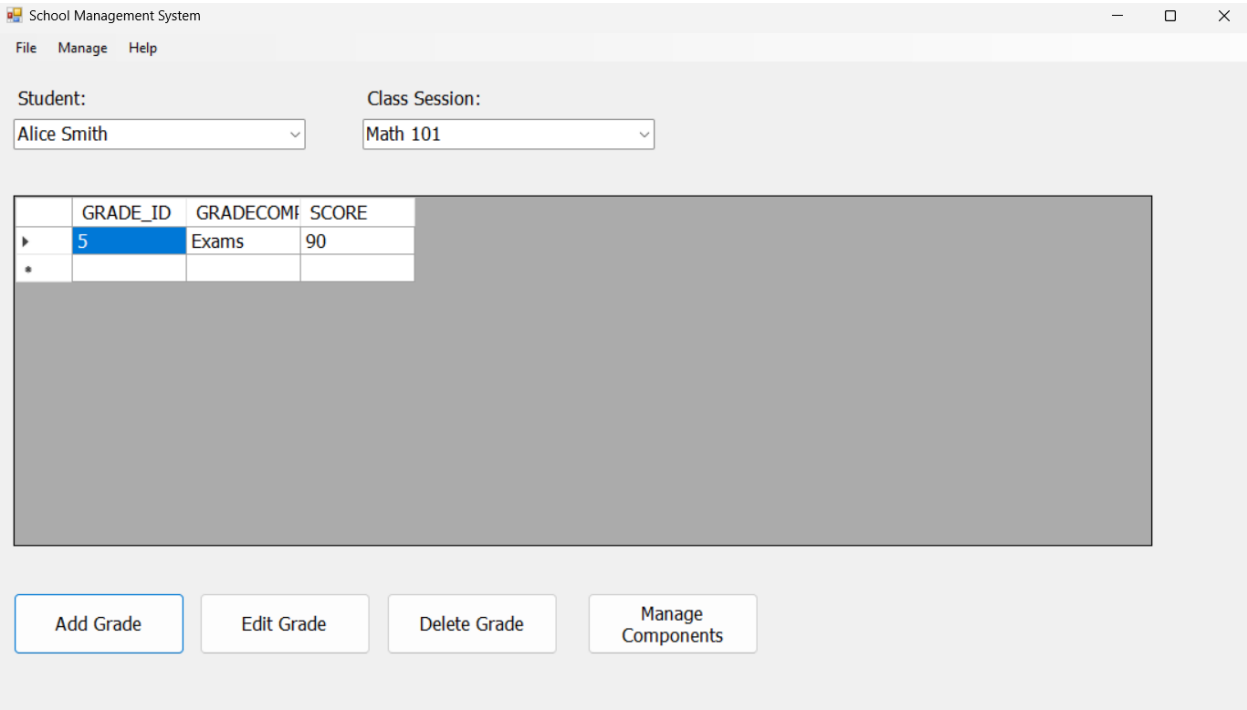
```
        string semesterName = semesterNameTextBox.Text.Trim();
        int schoolYear;
        if (!int.TryParse(schoolYearTextBox.Text.Trim(), out schoolYear)) {
            MessageBox.Show("Please enter a valid school year (number).");
            return;
        }


        if (string.IsNullOrEmpty(semesterName)) {
            MessageBox.Show("Please enter a semester name.");
            return;
        }


        string insertQuery = "INSERT INTO Semester (semester_name, school_year) VALUES
(:semesterName, :schoolYear)";
        OracleParameter[] parameters = new OracleParameter[]
        {
            new OracleParameter("semesterName", OracleDbType.Varchar2, semesterName,
ParameterDirection.Input),
            new OracleParameter("schoolYear", OracleDbType.Int32, schoolYear,
ParameterDirection.Input)
        };

        int rowsAffected = SchoolDatabase.ExecuteNonQuery(insertQuery, parameters);

        if (rowsAffected > 0) {
            MessageBox.Show("Semester added successfully.");

            LoadSemesterData();
        }
        else {
            MessageBox.Show("An error occurred while adding the semester.");
        }
    }

    private void updateButton_Click(object sender, EventArgs e) {

        if (semesterDataGridView.SelectedRows.Count == 0) {
            MessageBox.Show("Please select a semester to update.");
            return;
        }

        int semesterId =
Convert.ToInt32(semesterDataGridView.SelectedRows[0].Cells["semester_id"].Value);
        string semesterName = semesterNameTextBox.Text.Trim();
        int schoolYear;
        if (!int.TryParse(schoolYearTextBox.Text.Trim(), out schoolYear)) {
            MessageBox.Show("Please enter a valid school year (number).");
            return;
        }


        if (string.IsNullOrEmpty(semesterName)) {
            MessageBox.Show("Please enter a semester name.");
            return;
        }
```

```
        string updateQuery = "UPDATE Semester SET semester_name = :semesterName,
school_year = :schoolYear WHERE semester_id = :semesterId";
        OracleParameter[] parameters = new OracleParameter[]
        {
            new OracleParameter("semesterName", OracleDbType.Varchar2, semesterName,
ParameterDirection.Input),
            new OracleParameter("schoolYear", OracleDbType.Int32, schoolYear,
ParameterDirection.Input),
            new OracleParameter("semesterId", OracleDbType.Int32, semesterId,
ParameterDirection.Input)
        };

        int rowsAffected = SchoolDatabase.ExecuteNonQuery(updateQuery, parameters);

        if (rowsAffected > 0) {
            MessageBox.Show("Semester updated successfully.");
            LoadSemesterData();
        }
        else {
            MessageBox.Show("An error occurred while updating the semester.");
        }
    }

    private void deleteButton_Click(object sender, EventArgs e) {

        if (semesterDataGridView.SelectedRows.Count == 0) {
            MessageBox.Show("Please select a semester to delete.");
            return;
        }

        int semesterId =
Convert.ToInt32(semesterDataGridView.SelectedRows[0].Cells["semester_id"].Value);


        DialogResult result = MessageBox.Show(
            $"Are you sure you want to delete semester with ID {semesterId}?",
            "Confirm Deletion",
            MessageBoxButtons.YesNo,
            MessageBoxIcon.Warning);

        if (result == DialogResult.Yes) {

            string deleteQuery = "DELETE FROM Semester WHERE semester_id = :semesterId";
            OracleParameter semesterIdParameter = new OracleParameter("semesterId",
OracleDbType.Int32, semesterId, ParameterDirection.Input);
            int rowsAffected = SchoolDatabase.ExecuteNonQuery(deleteQuery, new
OracleParameter[] { semesterIdParameter });

            if (rowsAffected > 0) {
                MessageBox.Show("Semester deleted successfully.");
                LoadSemesterData();
            }
            else {
                MessageBox.Show("An error occurred while deleting the semester.");
            }
        }
    }
```

```
    private void semesterDataGridView_SelectionChanged(object sender, EventArgs e) {

        if (semesterDataGridView.SelectedRows.Count > 0) {
            DataRowView selectedRow =
(DataRowView)semesterDataGridView.SelectedRows[0].DataBoundItem;
            semesterNameTextBox.Text = selectedRow["semester_name"].ToString();
            schoolYearTextBox.Text = selectedRow["school_year"].ToString();
        }
    }
}
```

2.2.5. نماذج الدرجات: إدارة درجات الطلاب

2.2.5.1. نموذج قائمة الدرجات



الكود:

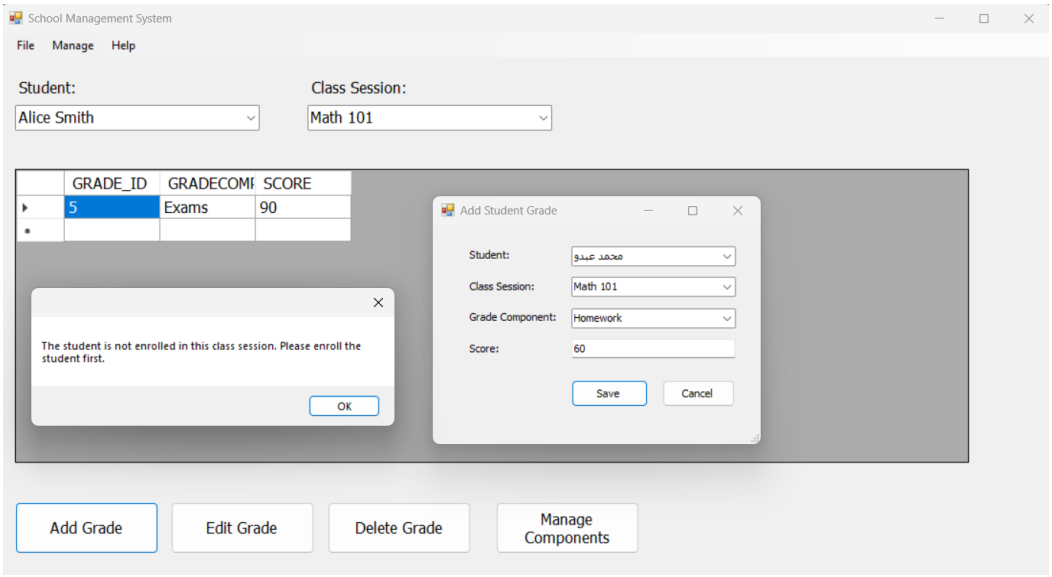
```
public partial class StudentGradesForm : Form {
    public StudentGradesForm() {
        InitializeComponent();
        LoadStudents();
        LoadClassSessions();
    }

    private void StudentGradesForm_Load(object sender, EventArgs e) {
    }

    private void LoadStudents() {
```

```
        DataTable students = SchoolDatabase.ExecuteQuery("SELECT student_id, first_name
|| ' ' || last_name AS student_name FROM Student");


        studentComboBox.DataSource = students;
        studentComboBox.DisplayMember = "student_name";
        studentComboBox.ValueMember = "student_id";
    }

    private void LoadClassSessions() {

        DataTable classSessions = SchoolDatabase.ExecuteQuery(
            "SELECT cs.session_id, c.class_name " +
            "FROM ClassSession cs " +
            "JOIN Class c ON cs.class_id = c.class_id");


        classSessionComboBox.DataSource = classSessions;
        classSessionComboBox.DisplayMember = "class_name";
        classSessionComboBox.ValueMember = "session_id";
    }

    private void studentComboBox_SelectedIndexChanged(object sender, EventArgs e) {

        RefreshGrades();
    }

    private void classSessionComboBox_SelectedIndexChanged(object sender, EventArgs e) {

        RefreshGrades();
    }

    private void RefreshGrades() {

        int studentId = Convert.ToInt32(studentComboBox.SelectedValue);
        int classSessionId = Convert.ToInt32(classSessionComboBox.SelectedValue);


        string query = @"
                        SELECT
                            sg.grade_id,
                            gc.component_name AS GradeComponent,
                            sg.score AS Score
                        FROM
                            StudentGrade sg
                        JOIN
                            Enrollment e ON sg.enrollment_id = e.enrollment_id
                        JOIN
                            GradeComponent gc ON sg.component_id = gc.component_id
                        WHERE
                            e.student_id = :studentId AND e.session_id = :classSessionId
                    ";

        OracleParameter[] parameters = new OracleParameter[]
        {
    new OracleParameter("studentId", OracleDbType.Int32, studentId,
ParameterDirection.Input),
```

```
        new OracleParameter("classSessionId", OracleDbType.Int32, classSessionId,
ParameterDirection.Input)
        };


        Console.WriteLine(query);

        DataTable grades = SchoolDatabase.ExecuteQuery(query, parameters);


        gradesDataGridView.DataSource = grades;
    }

    private void addGradeButton_Click(object sender, EventArgs e) {

        AddStudentGradeForm addGradeForm = new AddStudentGradeForm();
        addGradeForm.ShowDialog();


        RefreshGrades();
    }

    private void editGradeButton_Click(object sender, EventArgs e) {

        if (gradesDataGridView.SelectedRows.Count == 0) {
            MessageBox.Show("Please select a grade to edit.");
            return;
        }


        int gradeId =
Convert.ToInt32(gradesDataGridView.SelectedRows[0].Cells["grade_id"].Value);


        EditStudentGradeForm editGradeForm = new EditStudentGradeForm(gradeId);
        editGradeForm.ShowDialog();


        RefreshGrades();
    }

    private void deleteGradeButton_Click(object sender, EventArgs e) {

        if (gradesDataGridView.SelectedRows.Count == 0) {
            MessageBox.Show("Please select a grade to delete.");
            return;
        }


        int gradeId =
Convert.ToInt32(gradesDataGridView.SelectedRows[0].Cells["grade_id"].Value);


        DialogResult result = MessageBox.Show("Are you sure you want to delete this
grade?", "Confirm Deletion", MessageBoxButtons.YesNo, MessageBoxIcon.Warning);

        if (result == DialogResult.Yes) {
```

```
            try {

                OracleParameter gradeIdParameter = new OracleParameter("gradeId",
OracleDbType.Int32, gradeId, ParameterDirection.Input);


                int rowsAffected = SchoolDatabase.ExecuteNonQuery(
                    "DELETE FROM StudentGrade WHERE grade_id = :gradeId",
                    new OracleParameter[] { gradeIdParameter });

                if (rowsAffected > 0) {
                    MessageBox.Show("Grade deleted successfully.");

                    RefreshGrades();
                }
                else {
                    MessageBox.Show("An error occurred while deleting the grade.");
                }
            }
            catch (Exception ex) {
                MessageBox.Show("Error: " + ex.Message);
            }
        }
    }
    private void manageComponentsButton_Click(object sender, EventArgs e) {

        GradeComponentsForm gradeComponentsForm = new GradeComponentsForm();


        gradeComponentsForm.ShowDialog();
    }
}
```

2.2.5.2. نموذج إضافة علامة لطالب



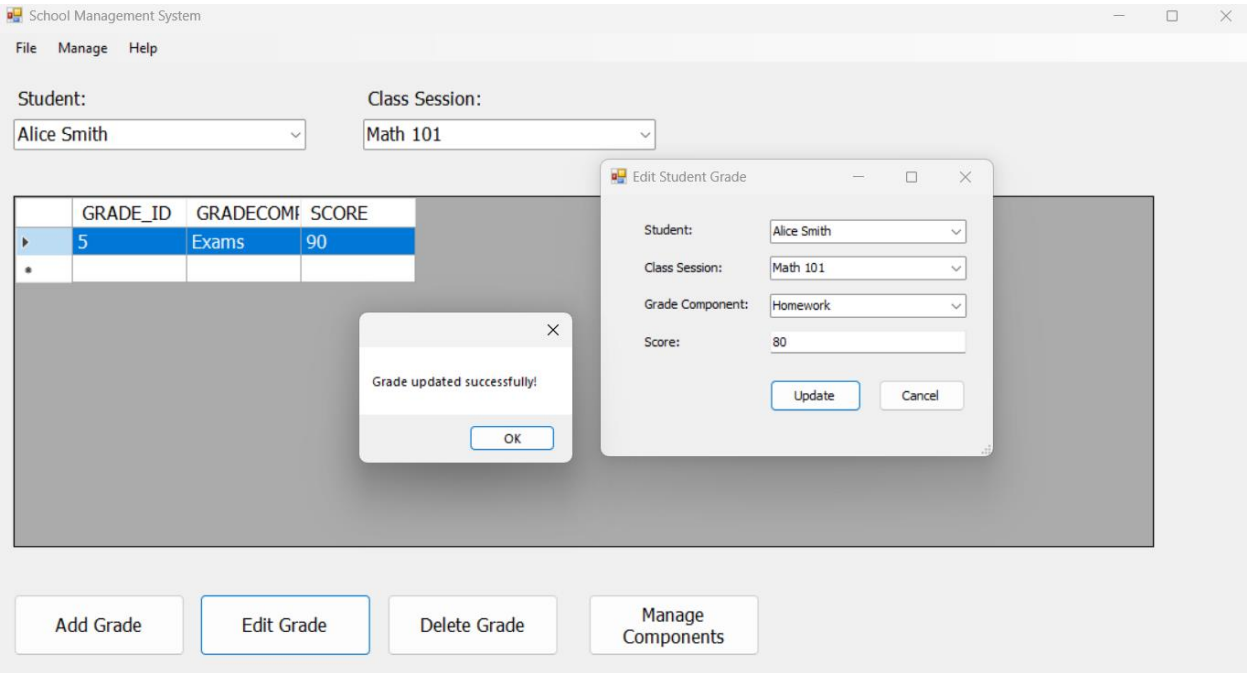
الكود:

```
public partial class AddStudentGradeForm : Form {
    public AddStudentGradeForm() {
        InitializeComponent();
        LoadStudents();
        LoadClassSessions();
        LoadGradeComponents();
    }

    private void LoadStudents() {

        DataTable students = SchoolDatabase.ExecuteQuery("SELECT student_id, first_name
|| ' ' || last_name AS student_name FROM Student");


        studentComboBox.DataSource = students;
        studentComboBox.DisplayMember = "student_name";
        studentComboBox.ValueMember = "student_id";
    }

    private void LoadClassSessions() {

        DataTable classSessions = SchoolDatabase.ExecuteQuery(
            "SELECT cs.session_id, c.class_name " +
            "FROM ClassSession cs " +
            "JOIN Class c ON cs.class_id = c.class_id");


        classSessionComboBox.DataSource = classSessions;
        classSessionComboBox.DisplayMember = "class_name";
        classSessionComboBox.ValueMember = "session_id";
    }

    private void LoadGradeComponents() {

        DataTable gradeComponents = SchoolDatabase.ExecuteQuery("SELECT component_id,
component_name FROM GradeComponent");


        gradeComponentComboBox.DataSource = gradeComponents;
        gradeComponentComboBox.DisplayMember = "component_name";
        gradeComponentComboBox.ValueMember = "component_id";
    }

    private void saveButton_Click(object sender, EventArgs e) {

        if (string.IsNullOrEmpty(scoreTextBox.Text)) {
            MessageBox.Show("Please enter a score.");
            return;
        }

        if (!decimal.TryParse(scoreTextBox.Text, out decimal score)) {
            MessageBox.Show("Please enter a valid numeric score.");
            return;
        }
```

```
        int studentId = Convert.ToInt32(studentComboBox.SelectedValue);
        int classSessionId = Convert.ToInt32(classSessionComboBox.SelectedValue);
        int gradeComponentId = Convert.ToInt32(gradeComponentComboBox.SelectedValue);


        string checkEnrollmentQuery = "SELECT 1 FROM Enrollment WHERE student_id =
:studentId AND session_id = :classSessionId";
        OracleParameter[] checkEnrollmentParameters = new OracleParameter[]
        {
            new OracleParameter("studentId", OracleDbType.Int32, studentId,
ParameterDirection.Input),
            new OracleParameter("classSessionId", OracleDbType.Int32, classSessionId,
ParameterDirection.Input)
        };

        DataTable enrollmentCheck = SchoolDatabase.ExecuteQuery(checkEnrollmentQuery,
checkEnrollmentParameters);

        if (enrollmentCheck.Rows.Count == 0) {
            MessageBox.Show("The student is not enrolled in this class session. Please
enroll the student first.");
            return;
        }


        string getEnrollmentIdQuery = "SELECT enrollment_id FROM Enrollment WHERE
student_id = :studentId AND session_id = :classSessionId";
        OracleParameter[] getEnrollmentIdParameters = new OracleParameter[]
        {
            new OracleParameter("studentId", OracleDbType.Int32, studentId,
ParameterDirection.Input),
            new OracleParameter("classSessionId", OracleDbType.Int32, classSessionId,
ParameterDirection.Input)
        };

        DataTable enrollmentIdData = SchoolDatabase.ExecuteQuery(getEnrollmentIdQuery,
getEnrollmentIdParameters);

        if (enrollmentIdData.Rows.Count == 0) {
            MessageBox.Show("Error retrieving enrollment ID.");
            return;
        }

        int enrollmentId = Convert.ToInt32(enrollmentIdData.Rows[0]["enrollment_id"]);


        string insertQuery = @"INSERT INTO StudentGrade (enrollment_id, component_id,
score) VALUES (:enrollmentId, :gradeComponentId, :score)";
        OracleParameter[] parameters = new OracleParameter[]
        {
            new OracleParameter("enrollmentId", OracleDbType.Int32, enrollmentId,
ParameterDirection.Input),
            new OracleParameter("gradeComponentId", OracleDbType.Int32, gradeComponentId,
ParameterDirection.Input),
            new OracleParameter("score", OracleDbType.Decimal, score,
ParameterDirection.Input)
        };
```

```
        int rowsAffected = SchoolDatabase.ExecuteNonQuery(insertQuery, parameters);

        if (rowsAffected > 0) {
            MessageBox.Show("Grade added successfully!");
            this.DialogResult = DialogResult.OK;
            this.Close();
        }
        else {
            MessageBox.Show("An error occurred while adding the grade.");
        }
    }

    private void cancelButton_Click(object sender, EventArgs e) {
        this.Close();
    }

    private void AddStudentGradeForm_Load(object sender, EventArgs e) {

    }
}
```

3.2.2.5. نموذج تعديل على درجة طالب



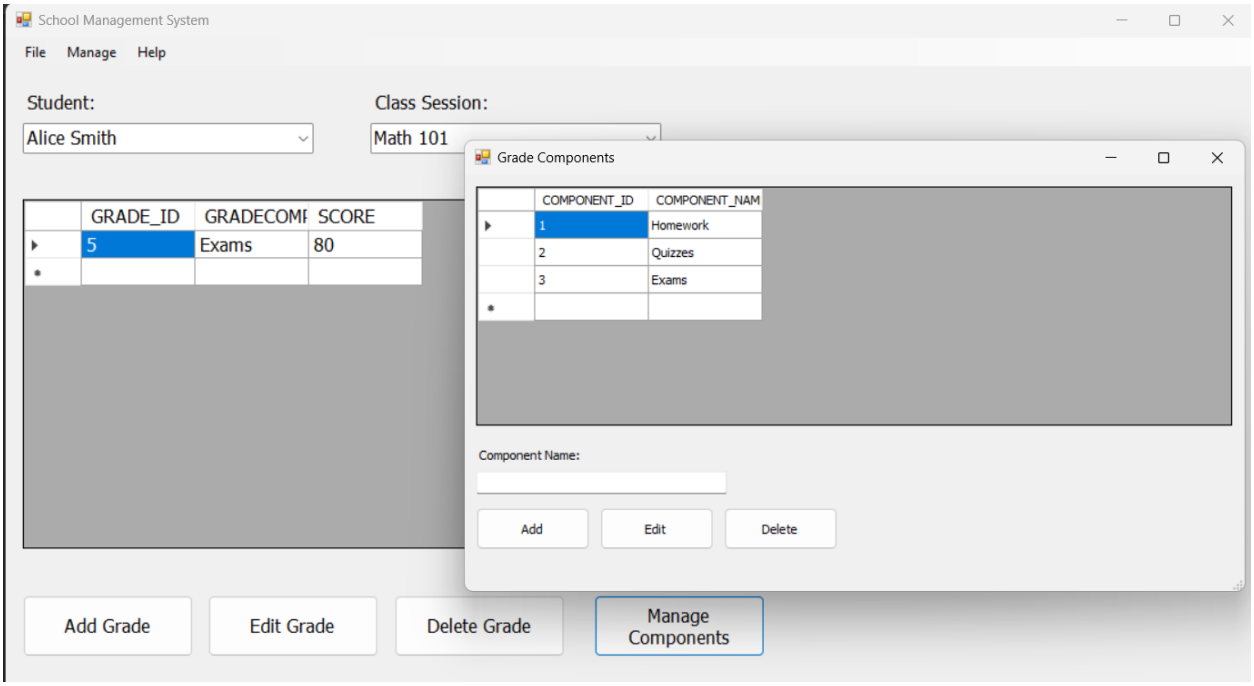
الكود:

```
public partial class EditStudentGradeForm : Form {
    private int gradeId;

    public EditStudentGradeForm(int gradeId) {
        InitializeComponent();
        this.gradeId = gradeId;
        LoadGradeData();
        LoadStudents();
        LoadClassSessions();
        LoadGradeComponents();
    }

    private void LoadStudents() {

        DataTable students = SchoolDatabase.ExecuteQuery("SELECT student_id, first_name
|| ' ' || last_name AS student_name FROM Student");


        studentComboBox.DataSource = students;
        studentComboBox.DisplayMember = "student_name";
        studentComboBox.ValueMember = "student_id";
    }

    private void LoadClassSessions() {

        DataTable classSessions = SchoolDatabase.ExecuteQuery(
            "SELECT cs.session_id, c.class_name " +
            "FROM ClassSession cs " +
            "JOIN Class c ON cs.class_id = c.class_id");


        classSessionComboBox.DataSource = classSessions;
        classSessionComboBox.DisplayMember = "class_name";
        classSessionComboBox.ValueMember = "session_id";
    }

    private void LoadGradeComponents() {

        DataTable gradeComponents = SchoolDatabase.ExecuteQuery("SELECT component_id,
component_name FROM GradeComponent");


        gradeComponentComboBox.DataSource = gradeComponents;
        gradeComponentComboBox.DisplayMember = "component_name";
        gradeComponentComboBox.ValueMember = "component_id";
    }

    private void LoadGradeData() {
        string query = "SELECT sg.grade_id, sg.score, e.student_id, e.session_id,
sg.component_id " +
                       "FROM StudentGrade sg " +
                       "JOIN Enrollment e ON sg.enrollment_id = e.enrollment_id " +
                       "WHERE sg.grade_id = :gradeId";

        OracleParameter parameter = new OracleParameter("gradeId", OracleDbType.Int32,
gradeId, ParameterDirection.Input);
```

```
        DataTable gradeData = SchoolDatabase.ExecuteQuery(query, new OracleParameter[] {
parameter });

        if (gradeData.Rows.Count > 0) {
            DataRow row = gradeData.Rows[0];




            scoreTextBox.Text = row["score"].ToString();
            studentComboBox.SelectedValue = row["student_id"];
            classSessionComboBox.SelectedValue = row["session_id"];
            gradeComponentComboBox.SelectedValue = row["component_id"];
        }
        else {
            MessageBox.Show("Grade not found.");
        }
    }

    private void updateButton_Click(object sender, EventArgs e) {

        if (string.IsNullOrEmpty(scoreTextBox.Text)) {
            MessageBox.Show("Please enter a score.");
            return;
        }

        if (!decimal.TryParse(scoreTextBox.Text, out decimal score)) {
            MessageBox.Show("Please enter a valid numeric score.");
            return;
        }


        string updateQuery = @"UPDATE StudentGrade SET score = :score WHERE grade_id =
:gradeId";

        OracleParameter[] parameters = new OracleParameter[]
        {
            new OracleParameter("score", OracleDbType.Decimal, score,
ParameterDirection.Input),
            new OracleParameter("gradeId", OracleDbType.Int32, gradeId,
ParameterDirection.Input)
        };

        int rowsAffected = SchoolDatabase.ExecuteNonQuery(updateQuery, parameters);

        if (rowsAffected > 0) {
            MessageBox.Show("Grade updated successfully!");
            this.DialogResult = DialogResult.OK;
            this.Close();
        }
        else {
            MessageBox.Show("An error occurred while updating the grade.");
        }
    }

    private void cancelButton_Click(object sender, EventArgs e) {
        this.Close();
    }
```

```
    private void EditStudentGradeForm_Load(object sender, EventArgs e) {

    }
}
```

2.2.5.4. نموذج مكون العلامة



الكود:

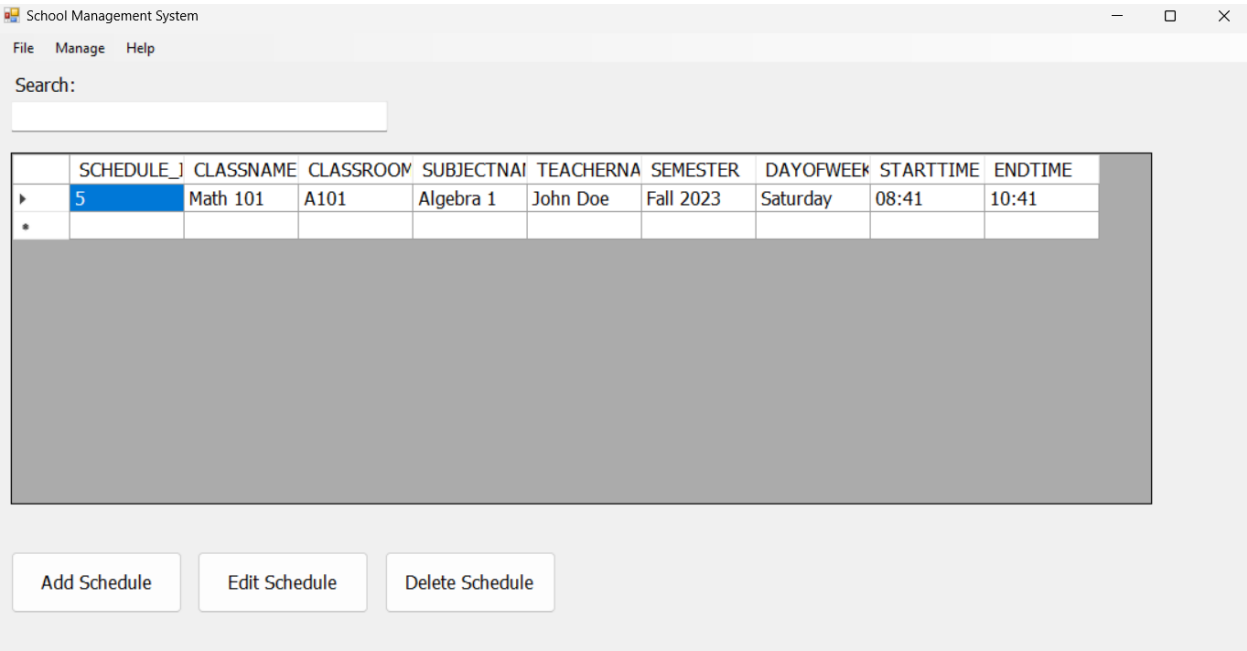
```
public partial class GradeComponentsForm : Form {
    private int selectedComponentId;

    public GradeComponentsForm() {
        InitializeComponent();
        LoadGradeComponents();
    }

    private void LoadGradeComponents() {

        DataTable gradeComponents = SchoolDatabase.ExecuteQuery("SELECT * FROM
GradeComponent");


        gradeComponentsDataGridView.DataSource = gradeComponents;
    }

    private void addButton_Click(object sender, EventArgs e) {
```

```
        if (string.IsNullOrEmpty(componentNameTextBox.Text)) {
            MessageBox.Show("Please enter a component name.");
            return;
        }


        string insertQuery = @"INSERT INTO GradeComponent (component_name) VALUES
(:componentName)";

        OracleParameter[] parameters = new OracleParameter[]
        {
            new OracleParameter("componentName", OracleDbType.Varchar2,
componentNameTextBox.Text, ParameterDirection.Input)
        };

        int rowsAffected = SchoolDatabase.ExecuteNonQuery(insertQuery, parameters);

        if (rowsAffected > 0) {
            MessageBox.Show("Grade component added successfully!");
            LoadGradeComponents();
            componentNameTextBox.Clear();
        }
        else {
            MessageBox.Show("An error occurred while adding the grade component.");
        }
    }

    private void editButton_Click(object sender, EventArgs e) {

        string updateQuery = @"UPDATE GradeComponent SET component_name = :componentName
WHERE component_id = :componentId";

        OracleParameter[] parameters = new OracleParameter[]
        {
            new OracleParameter("componentName", OracleDbType.Varchar2,
componentNameTextBox.Text, ParameterDirection.Input),
            new OracleParameter("componentId", OracleDbType.Int32, selectedComponentId,
ParameterDirection.Input)
        };

        int rowsAffected = SchoolDatabase.ExecuteNonQuery(updateQuery, parameters);

        if (rowsAffected > 0) {
            MessageBox.Show("Grade component updated successfully!");
            LoadGradeComponents();
            componentNameTextBox.Clear();
            selectedComponentId = 0;
        }
        else {
            MessageBox.Show("An error occurred while updating the grade component.");
        }
    }

    private void deleteButton_Click(object sender, EventArgs e) {

        if (gradeComponentsDataGridView.SelectedRows.Count == 0) {
            MessageBox.Show("Please select a grade component to delete.");
            return;
```

```
        }

        int componentId =
Convert.ToInt32(gradeComponentsDataGridView.SelectedRows[0].Cells["component_id"].Value);


        DialogResult result = MessageBox.Show(
            $"Are you sure you want to delete the grade component with ID
{componentId}?",
            "Confirm Deletion",
            MessageBoxButtons.YesNo,
            MessageBoxIcon.Warning);

        if (result == DialogResult.Yes) {

            string deleteQuery = @"DELETE FROM GradeComponent WHERE component_id =
:componentId";

            OracleParameter[] parameters = new OracleParameter[]
            {
                new OracleParameter("componentId", OracleDbType.Int32, componentId,
ParameterDirection.Input)
            };

            int rowsAffected = SchoolDatabase.ExecuteNonQuery(deleteQuery, parameters);

            if (rowsAffected > 0) {
                MessageBox.Show("Grade component deleted successfully!");
                LoadGradeComponents();
            }
            else {
                MessageBox.Show("An error occurred while deleting the grade component.");
            }
        }
    }

    private void gradeComponentsDataGridView_SelectionChanged(object sender, EventArgs e)
{

        if (gradeComponentsDataGridView.SelectedRows.Count > 0) {
            selectedComponentId =
Convert.ToInt32(gradeComponentsDataGridView.SelectedRows[0].Cells["component_id"].Value);
            componentNameTextBox.Text =
gradeComponentsDataGridView.SelectedRows[0].Cells["component_name"].Value.ToString();
        }
        else {
            selectedComponentId = 0;
            componentNameTextBox.Clear();
        }
    }

    private void GradeComponentsForm_Load(object sender, EventArgs e) {

    }
}
```

2.2.6. نماذج جداول الاسبوعي: إدارة جداول الاسبوعية

2.2.6.1. نموذج جداول الاسبوعي



الكود:

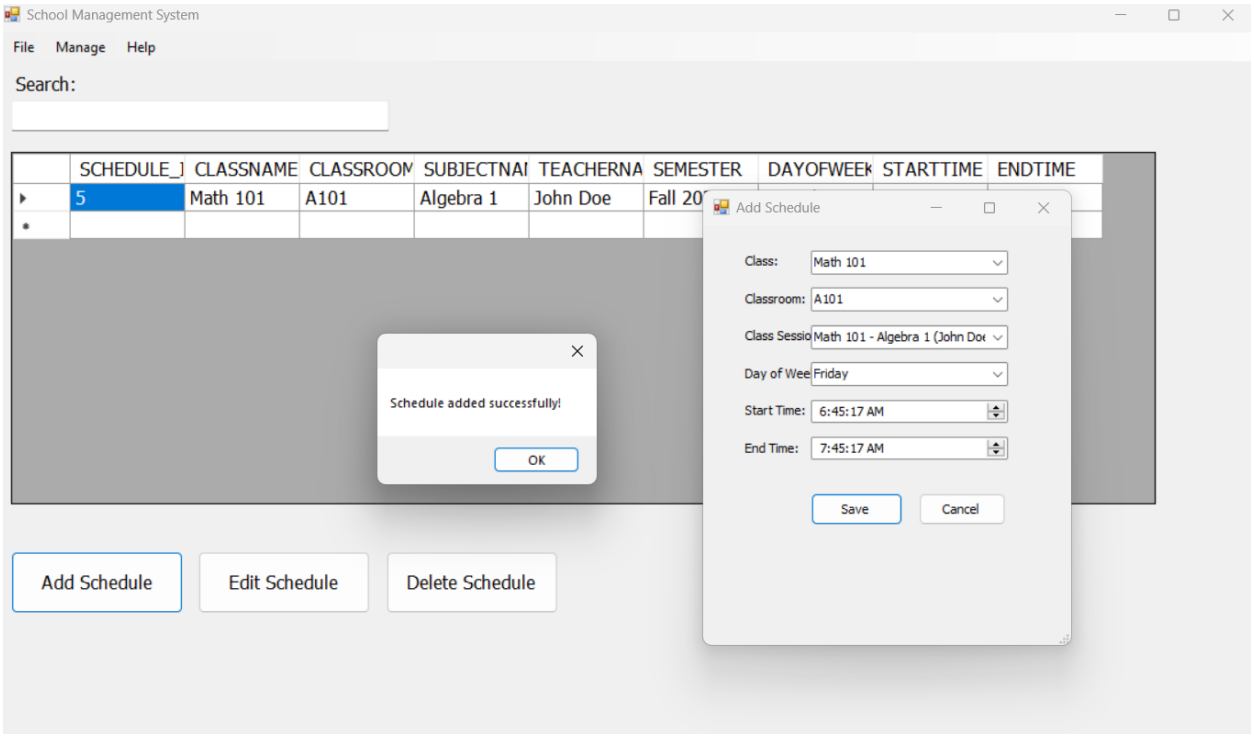
```
public partial class ScheduleListForm : Form {
        public ScheduleListForm() {
            InitializeComponent();
        }

        private void ScheduleListForm_Load(object sender, EventArgs e) {
            LoadSchedule();
        }

        private void LoadSchedule() {

            string query = @"SELECT
                                s.schedule_id,
                                c.class_name AS ClassName,
                                r.room_number AS Classroom,
                                su.subject_name AS SubjectName,
                                e.first_name || ' ' || e.last_name AS TeacherName,
                                se.semester_name || ' ' || se.school_year AS Semester,
                                s.day_of_week AS DayOfWeek,
                                TO_CHAR(s.start_time, 'HH24:MI') AS StartTime,
                                TO_CHAR(s.end_time, 'HH24:MI') AS EndTime
                            FROM
                                Schedule s
                            JOIN
```

```
                                Class c ON s.class_id = c.class_id
                            JOIN
                                Classroom r ON s.classroom_id = r.classroom_id
                            JOIN
                                ClassSession cs ON s.session_id = cs.session_id
                            JOIN
                                Employee e ON cs.teacher_id = e.employee_id
                            JOIN
                                Semester se ON cs.semester_id = se.semester_id
                            JOIN
                                Subject su ON cs.subject_id = su.subject_id";

            DataTable scheduleDataTable = SchoolDatabase.ExecuteQuery(query);


            scheduleDataGridView.DataSource = scheduleDataTable;
        }

        private void searchTextBox_TextChanged(object sender, EventArgs e) {

            string searchTerm = searchTextBox.Text.Trim();


            string query = @"SELECT * FROM Schedule WHERE class_id IN (SELECT class_id
FROM Class WHERE class_name LIKE :searchTerm)
                            OR classroom_id IN (SELECT classroom_id FROM Classroom WHERE
room_number LIKE :searchTerm)
                            OR session_id IN (SELECT session_id FROM ClassSession WHERE
teacher_id IN (SELECT employee_id FROM Employee WHERE first_name LIKE :searchTerm OR
last_name LIKE :searchTerm))";


            OracleParameter searchTermParameter = new OracleParameter(":searchTerm",
OracleDbType.Varchar2);
            searchTermParameter.Value = "%" + searchTerm + "%";


            DataTable searchResults = SchoolDatabase.ExecuteQuery(query, new
OracleParameter[] { searchTermParameter });


            scheduleDataGridView.DataSource = searchResults;
        }

        private void addScheduleButton_Click(object sender, EventArgs e) {
            AddScheduleForm addScheduleForm = new AddScheduleForm();
            DialogResult result = addScheduleForm.ShowDialog();
            if (result == DialogResult.OK) {
                LoadSchedule();
            }
        }

        private void deleteScheduleButton_Click(object sender, EventArgs e) {

            if (scheduleDataGridView.SelectedRows.Count == 0) {
                MessageBox.Show("Please select a schedule to delete.");
                return;
            }
```

```
            int scheduleId =
Convert.ToInt32(scheduleDataGridView.SelectedRows[0].Cells["schedule_id"].Value);


            DialogResult result = MessageBox.Show(
                $"Are you sure you want to delete schedule with ID {scheduleId}?",
                "Confirm Deletion",
                MessageBoxButtons.YesNo,
                MessageBoxIcon.Warning);

            if (result == DialogResult.No) {
                return;
            }


            try {

                OracleParameter scheduleIdParameter = new OracleParameter("schedule_id",
scheduleId);


                int rowsAffected = SchoolDatabase.ExecuteNonQuery(
                    "DELETE FROM Schedule WHERE schedule_id = :schedule_id",
                    new OracleParameter[] { scheduleIdParameter });

                if (rowsAffected > 0) {
                    MessageBox.Show("Schedule deleted successfully.");

                    LoadSchedule();
                }
                else {
                    MessageBox.Show("An error occurred while deleting the schedule.");
                }
            }
            catch (Exception ex) {
                MessageBox.Show("Error: " + ex.Message);
            }
        }

        private void editScheduleButton_Click(object sender, EventArgs e) {

            if (scheduleDataGridView.SelectedRows.Count == 0) {
                MessageBox.Show("Please select a schedule to edit.");
                return;
            }


            int selectedScheduleId =
Convert.ToInt32(scheduleDataGridView.SelectedRows[0].Cells["schedule_id"].Value);


            EditScheduleForm editScheduleForm = new EditScheduleForm(selectedScheduleId);


            DialogResult result = editScheduleForm.ShowDialog();
```

```
            if (result == DialogResult.OK) {
                LoadSchedule();
            }
        }

    }
```

2.2.6.2. نموذج إضافة جداول الاسبوعي



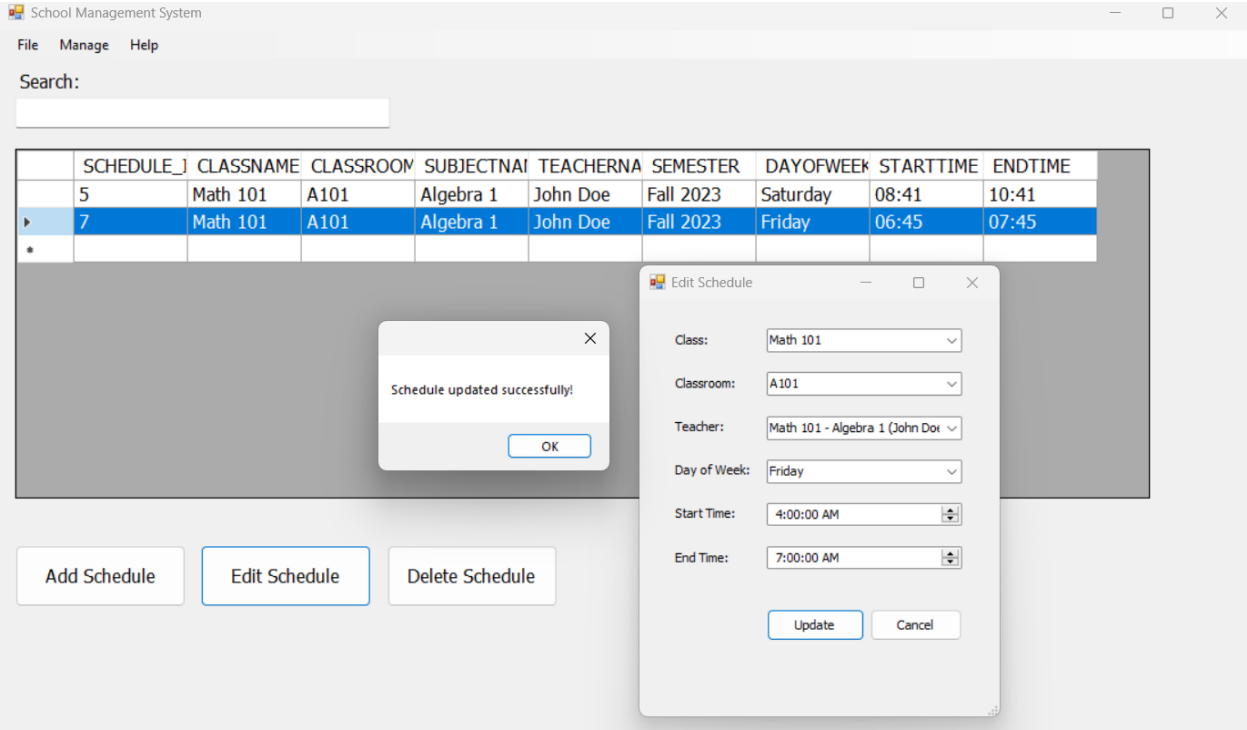
الكود:

```
public partial class AddScheduleForm : Form {
    public AddScheduleForm() {
        InitializeComponent();
        LoadClassData();
        LoadClassroomData();
        LoadClassSessionData();
    }

    private void LoadClassData() {
        DataTable classData = SchoolDatabase.ExecuteQuery("SELECT * FROM Class");
        classComboBox.DataSource = classData;
        classComboBox.DisplayMember = "class_name";
        classComboBox.ValueMember = "class_id";
    }

    private void LoadClassroomData() {
        DataTable classroomData = SchoolDatabase.ExecuteQuery("SELECT * FROM Classroom");
```

```
        classroomComboBox.DataSource = classroomData;
        classroomComboBox.DisplayMember = "room_number";
        classroomComboBox.ValueMember = "classroom_id";
    }

    private void LoadClassSessionData() {
        DataTable classSessionData = SchoolDatabase.ExecuteQuery(@"
            SELECT
                cs.session_id,
                (c.class_name || ' - ' || s.subject_name || ' (' || e.first_name || ' '
|| e.last_name || ')' || ' (Semester ' || se.semester_name || ')') AS session_details
            FROM
                ClassSession cs
            JOIN
                Class c ON cs.class_id = c.class_id
            JOIN
                Subject s ON cs.subject_id = s.subject_id
            JOIN
                Employee e ON cs.teacher_id = e.employee_id
            JOIN
                Semester se ON cs.semester_id = se.semester_id
        ");

        classSessionComboBox.DataSource = classSessionData;
        classSessionComboBox.DisplayMember = "session_details";
        classSessionComboBox.ValueMember = "session_id";
    }

    private void saveScheduleButton_Click(object sender, EventArgs e) {

        if (classComboBox.SelectedValue == null || classroomComboBox.SelectedValue ==
null ||
            classSessionComboBox.SelectedValue == null || dayOfWeekComboBox.SelectedItem
== null) {
            MessageBox.Show("Please select all required values.");
            return;
        }


        Schedule newSchedule = new Schedule {
            ClassId = Convert.ToInt32(classComboBox.SelectedValue),
            ClassroomId = Convert.ToInt32(classroomComboBox.SelectedValue),
            SessionId = Convert.ToInt32(classSessionComboBox.SelectedValue),
            DayOfTheWeek = dayOfWeekComboBox.SelectedItem.ToString(),
            StartTime = startTimePicker.Value.TimeOfDay,
            EndTime = endTimePicker.Value.TimeOfDay
        };


        string insertQuery = @"INSERT INTO Schedule (class_id, classroom_id, session_id,
day_of_week, start_time, end_time)
                      VALUES (:classId, :classroomId, :sessionId, :dayOfWeek,
:startTime, :endTime)";


        DateTime startTime = startTimePicker.Value.Date.Add(newSchedule.StartTime);
        DateTime endTime = endTimePicker.Value.Date.Add(newSchedule.EndTime);
```

```
        OracleParameter[] parameters = new OracleParameter[]
        {
    new OracleParameter("classId", OracleDbType.Int32, newSchedule.ClassId,
ParameterDirection.Input),
    new OracleParameter("classroomId", OracleDbType.Int32, newSchedule.ClassroomId,
ParameterDirection.Input),
    new OracleParameter("sessionId", OracleDbType.Int32, newSchedule.SessionId,
ParameterDirection.Input),
    new OracleParameter("dayOfWeek", OracleDbType.Varchar2, newSchedule.DayOfTheWeek,
ParameterDirection.Input),
    new OracleParameter("startTime", OracleDbType.Date, startTime,
ParameterDirection.Input),
    new OracleParameter("endTime", OracleDbType.Date, endTime, ParameterDirection.Input)
        };

        int rowsAffected = SchoolDatabase.ExecuteNonQuery(insertQuery, parameters);


        if (rowsAffected > 0) {
            MessageBox.Show("Schedule added successfully!");
            this.DialogResult = DialogResult.OK;
            this.Close();
        }
        else {
            MessageBox.Show("An error occurred while adding the schedule.");
        }
    }

    private void cancelScheduleButton_Click(object sender, EventArgs e) {
        this.Close();
    }
}
```

2.2.6.3. نموذج تعديل جداول الاسبوعي



الكود:

```
public partial class EditScheduleForm : Form {
    private int scheduleId;

    public EditScheduleForm(int scheduleId) {
        InitializeComponent();
        this.scheduleId = scheduleId;
        LoadScheduleData();
        LoadClassData();
        LoadClassroomData();
        LoadClassSessionData();
    }

    private void LoadScheduleData() {

        string query = "SELECT * FROM Schedule WHERE schedule_id = :scheduleId";
        OracleParameter parameter = new OracleParameter("scheduleId", OracleDbType.Int32,
scheduleId, ParameterDirection.Input);

        DataTable scheduleData = SchoolDatabase.ExecuteQuery(query, new OracleParameter[]
{ parameter });

        if (scheduleData.Rows.Count > 0) {
            DataRow row = scheduleData.Rows[0];
            classComboBox.SelectedValue = row["class_id"];
            classroomComboBox.SelectedValue = row["classroom_id"];
            classSessionComboBox.SelectedValue = row["session_id"];
            dayOfWeekComboBox.SelectedItem = row["day_of_week"];
            startTimePicker.Value = DateTime.Parse(row["start_time"].ToString());
            endTimePicker.Value = DateTime.Parse(row["end_time"].ToString());
```

```
        }
        else {
            MessageBox.Show("Schedule not found.");
        }
    }

    private void LoadClassData() {
        DataTable classData = SchoolDatabase.ExecuteQuery("SELECT * FROM Class");
        classComboBox.DataSource = classData;
        classComboBox.DisplayMember = "class_name";
        classComboBox.ValueMember = "class_id";
    }

    private void LoadClassroomData() {
        DataTable classroomData = SchoolDatabase.ExecuteQuery("SELECT * FROM Classroom");
        classroomComboBox.DataSource = classroomData;
        classroomComboBox.DisplayMember = "room_number";
        classroomComboBox.ValueMember = "classroom_id";
    }

    private void LoadClassSessionData() {
        DataTable classSessionData = SchoolDatabase.ExecuteQuery(@"
            SELECT
                cs.session_id,
                (c.class_name || ' - ' || s.subject_name || ' (' || e.first_name || ' '
|| e.last_name || ')' || ' (Semester ' || se.semester_name || ')') AS session_details
            FROM
                ClassSession cs
            JOIN
                Class c ON cs.class_id = c.class_id
            JOIN
                Subject s ON cs.subject_id = s.subject_id
            JOIN
                Employee e ON cs.teacher_id = e.employee_id
            JOIN
                Semester se ON cs.semester_id = se.semester_id
        ");

        classSessionComboBox.DataSource = classSessionData;
        classSessionComboBox.DisplayMember = "session_details";
        classSessionComboBox.ValueMember = "session_id";
    }

    private void updateScheduleButton_Click(object sender, EventArgs e) {

        if (classComboBox.SelectedValue == null || classroomComboBox.SelectedValue ==
null ||
            classSessionComboBox.SelectedValue == null || dayOfWeekComboBox.SelectedItem
== null) {
            MessageBox.Show("Please select all required values.");
            return;
        }


        Schedule updatedSchedule = new Schedule {
            ScheduleId = scheduleId,
            ClassId = Convert.ToInt32(classComboBox.SelectedValue),
            ClassroomId = Convert.ToInt32(classroomComboBox.SelectedValue),
```

```
            SessionId = Convert.ToInt32(classSessionComboBox.SelectedValue),
            DayOfTheWeek = dayOfWeekComboBox.SelectedItem.ToString(),
            StartTime = startTimePicker.Value.TimeOfDay,
            EndTime = endTimePicker.Value.TimeOfDay
        };

        string updateQuery = @"UPDATE Schedule
                        SET class_id = :classId,
                            classroom_id = :classroomId,
                            session_id = :sessionId,
                            day_of_week = :dayOfWeek,
                            start_time = :startTime,
                            end_time = :endTime
                        WHERE schedule_id = :scheduleId";


        DateTime startTime = startTimePicker.Value.Date.Add(updatedSchedule.StartTime);
        DateTime endTime = endTimePicker.Value.Date.Add(updatedSchedule.EndTime);

        OracleParameter[] parameters = new OracleParameter[]
        {
    new OracleParameter("classId", OracleDbType.Int32, updatedSchedule.ClassId,
ParameterDirection.Input),
    new OracleParameter("classroomId", OracleDbType.Int32, updatedSchedule.ClassroomId,
ParameterDirection.Input),
    new OracleParameter("sessionId", OracleDbType.Int32, updatedSchedule.SessionId,
ParameterDirection.Input),
    new OracleParameter("dayOfWeek", OracleDbType.Varchar2, updatedSchedule.DayOfTheWeek,
ParameterDirection.Input),
    new OracleParameter("startTime", OracleDbType.Date, startTime,
ParameterDirection.Input),
    new OracleParameter("endTime", OracleDbType.Date, endTime, ParameterDirection.Input),
    new OracleParameter("scheduleId", OracleDbType.Int32, updatedSchedule.ScheduleId,
ParameterDirection.Input)
        };

        int rowsAffected = SchoolDatabase.ExecuteNonQuery(updateQuery, parameters);


        if (rowsAffected > 0) {
            MessageBox.Show("Schedule updated successfully!");
            this.DialogResult = DialogResult.OK;
            this.Close();
        }
        else {
            MessageBox.Show("An error occurred while updating the schedule.");
        }
    }

    private void cancelScheduleButton_Click(object sender, EventArgs e) {
        this.Close();
    }
}
```

2.3. ادخال القيم الي Oracle بستخدام C#

لتسهيل رأيت المدخلات وما يتم ادخال ضمن القاعدة البيانات تم إضافة طريقة تقم بتخزين أوامر الادخال في ملف نصي

```
public static int ExecuteNonQuery(string query, OracleParameter[] parameters = null) {
    int rowsAffected = 0;

    using (OracleConnection connection = new OracleConnection(connectionString)) {
        using (OracleCommand command = new OracleCommand(query, connection)) {
            if (parameters != null) {
                command.Parameters.AddRange(parameters);
            }

            try {
                connection.Open();
                rowsAffected = command.ExecuteNonQuery();

                // Log the INSERT statement with values
                if (query.TrimStart().ToUpper().StartsWith("INSERT INTO")) {
                    LogInsertStatement(command);
                }
            }
            catch (OracleException ex) {
                MessageBox.Show("Oracle Error: " + ex.Message);
                throw;
            }
            catch (Exception ex) {
                MessageBox.Show("Error: " + ex.Message);
                throw;
            }
        }
    }

    return rowsAffected;
}

private static void LogInsertStatement(OracleCommand command) {
    string logFilePath = "insert_log.txt";

    string insertStatement = command.CommandText;
    foreach (OracleParameter parameter in command.Parameters) {
        // Replace parameter placeholders with actual values
        insertStatement = insertStatement.Replace(
            parameter.ParameterName,
            parameter.Value == DBNull.Value ? "NULL" :
$"'{parameter.Value.ToString().Replace("'", "''")}'");
    }

    using (StreamWriter writer = new StreamWriter(logFilePath, true)) {
        writer.WriteLine($"{insertStatement};\n");
    }
}
```

بداية قمت بحذف التسلسلات (Sequence) وإعادة بنائها مرة اخرة مع المحفز ليبداء تعداد المعرفات من الصفر:

```
DROP SEQUENCE Employee_seq;

DROP SEQUENCE Role_seq;

DROP SEQUENCE Employee_Role_seq;

DROP SEQUENCE Grade_seq;

DROP SEQUENCE Class_seq;

DROP SEQUENCE Section_seq;

DROP SEQUENCE Classroom_seq;

DROP SEQUENCE Subject_seq;

DROP SEQUENCE Class_Session_seq;

DROP SEQUENCE Student_seq;

DROP SEQUENCE Semester_seq;

DROP SEQUENCE Schedule_seq;

DROP SEQUENCE Enrollment_seq;

DROP SEQUENCE Grade_Component_seq;

DROP SEQUENCE Student_Grade_seq;

DROP SEQUENCE Note_seq;
```

وهذه هي أوامر الادخال التي تم إدخالها, وتم أيضا تحسينها ليتم قبلها كأمر SQL في Oracle:

```
-- Inserting into Employee Table

INSERT INTO Employee (first_name, last_name, date_of_birth, phone_number, address,
hire_date)

VALUES ('الحسن' ,'أحمد', TO_DATE('1985-03-15', 'YYYY-MM-DD'), '0111111111', 'دمشق',
TO_DATE('2010-08-15', 'YYYY-MM-DD'));


INSERT INTO Employee (first_name, last_name, date_of_birth, phone_number, address,
hire_date)

VALUES ('علي' ,'محمد', TO_DATE('1990-07-20', 'YYYY-MM-DD'), '0122222222', 'دمشق',
TO_DATE('2015-01-20', 'YYYY-MM-DD'));


INSERT INTO Employee (first_name, last_name, date_of_birth, phone_number, address,
hire_date)

VALUES ('الزهر' ,'فاطمة', TO_DATE('1988-11-05', 'YYYY-MM-DD'), '0133333333', 'دمشق',
TO_DATE('2012-05-10', 'YYYY-MM-DD'));
```

```
INSERT INTO Employee (first_name, last_name, date_of_birth, phone_number, address,
hire_date)

VALUES ('ابراهيم' ,'سعيد', TO_DATE('1980-05-22', 'YYYY-MM-DD'), '0144444444', 'دمشق',
TO_DATE('2018-09-10', 'YYYY-MM-DD'));


INSERT INTO Employee (first_name, last_name, date_of_birth, phone_number, address,
hire_date)

VALUES ('حسن' ,'ريم', TO_DATE('1982-12-12', 'YYYY-MM-DD'), '0155555555', 'دمشق',
TO_DATE('2020-02-15', 'YYYY-MM-DD'));


INSERT INTO Employee (first_name, last_name, date_of_birth, phone_number, address,
hire_date)

VALUES ('عمر' ,'وائل', TO_DATE('1978-01-08', 'YYYY-MM-DD'), '0166666666', 'دمشق',
TO_DATE('2016-07-01', 'YYYY-MM-DD'));


-- Inserting into Role Table

INSERT INTO Role (role_name)

VALUES ('معلم');


INSERT INTO Role (role_name)

VALUES ('مدير');


INSERT INTO Role (role_name)

VALUES ('سكرتير');


-- Inserting into EmployeeRole Table

INSERT INTO EmployeeRole (employee_id, role_id)

VALUES (1, 1);


INSERT INTO EmployeeRole (employee_id, role_id)

VALUES (2, 2);


INSERT INTO EmployeeRole (employee_id, role_id)

VALUES (3, 3);
```

```
INSERT INTO EmployeeRole (employee_id, role_id)
VALUES (4, 1);

INSERT INTO EmployeeRole (employee_id, role_id)
VALUES (5, 1);

INSERT INTO EmployeeRole (employee_id, role_id)
VALUES (6, 1);

-- Inserting into Grade Table
INSERT INTO Grade (grade_name)
VALUES ('الصف الثامن');

INSERT INTO Grade (grade_name)
VALUES ('الصف التاسع');

INSERT INTO Grade (grade_name)
VALUES ('الصف العاشر');

-- Inserting into Class Table
INSERT INTO Class (class_name, grade_id)
VALUES ('الرياضيات', 1);

INSERT INTO Class (class_name, grade_id)
VALUES ('اللغة العربية', 1);

INSERT INTO Class (class_name, grade_id)
VALUES ('العلوم', 2);

-- Inserting into Section Table
INSERT INTO Section (section_name, class_id)
VALUES ('أ', 1);
```

```
INSERT INTO Section (section_name, class_id)
VALUES ('1 ,'ب);

INSERT INTO Section (section_name, class_id)
VALUES ('2 ,'أ);

INSERT INTO Section (section_name, class_id)
VALUES ('2 ,'ب);

-- Inserting into Classroom Table
INSERT INTO Classroom (room_number, capacity)
VALUES ('101', 30);

INSERT INTO Classroom (room_number, capacity)
VALUES ('102', 25);

INSERT INTO Classroom (room_number, capacity)
VALUES ('201', 35);

-- Inserting into Subject Table
INSERT INTO Subject (subject_name)
VALUES ('الرياضيات');

INSERT INTO Subject (subject_name)
VALUES ('اللغة العربية');

INSERT INTO Subject (subject_name)
VALUES ('العلوم');

-- Inserting into Student Table
INSERT INTO Student (first_name, last_name, date_of_birth, gender, address, parent_name,
parent_contact, enrollment_date)
```

```
VALUES ('محمد' ,'علي', TO_DATE('2010-09-01', 'YYYY-MM-DD'), 'Male', ' ,'محمد علي' ,'دمشق، سوريا'
0111111111'', TO_DATE('2023-08-15', 'YYYY-MM-DD'));


INSERT INTO Student (first_name, last_name, date_of_birth, gender, address, parent_name,
parent_contact, enrollment_date)

VALUES ('احمد' ,'سارة', TO_DATE('2010-10-10', 'YYYY-MM-DD'), 'Female', ' احمد' ,'دمشق، سوريا'
0122222222' ,'سارة', TO_DATE('2023-08-20', 'YYYY-MM-DD'));


INSERT INTO Student (first_name, last_name, date_of_birth, gender, address, parent_name,
parent_contact, enrollment_date)

VALUES ('حسن' ,'محمد', TO_DATE('2011-01-25', 'YYYY-MM-DD'), 'Male', ' حسن' ,'دمشق، سوريا'
0133333333' ,'محمد', TO_DATE('2023-08-25', 'YYYY-MM-DD'));


-- Inserting into Semester Table

INSERT INTO Semester (semester_name, school_year)

VALUES ('الخريف' ,'2023');


INSERT INTO Semester (semester_name, school_year)

VALUES ('الربيع' ,'2023');


-- Inserting into ClassSession Table

INSERT INTO ClassSession (subject_id, teacher_id, semester_id, class_id)

VALUES (1, 1, 1, 1);


INSERT INTO ClassSession (subject_id, teacher_id, semester_id, class_id)

VALUES (2, 1, 1, 2);


INSERT INTO ClassSession (subject_id, teacher_id, semester_id, class_id)

VALUES (3, 1, 2, 3);


INSERT INTO ClassSession (subject_id, teacher_id, semester_id, class_id)

VALUES (2, 4, 1, 2);


INSERT INTO ClassSession (subject_id, teacher_id, semester_id, class_id)
```

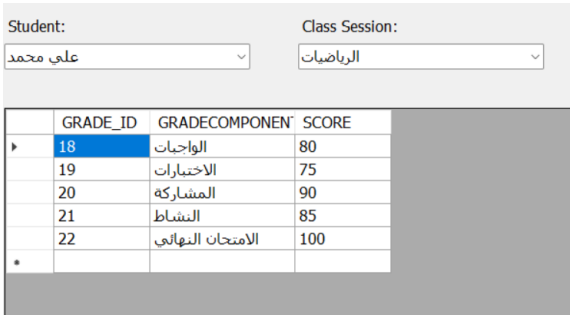
```
VALUES (3, 5, 2, 3);


INSERT INTO ClassSession (subject_id, teacher_id, semester_id, class_id)

VALUES (1, 6, 1, 1);


-- Inserting into Schedule Table

INSERT INTO Schedule (class_id, classroom_id, session_id, day_of_week, start_time,
end_time)

VALUES (1, 1, 2, 'Sunday', TO_DATE('08:00', 'HH24:MI'), TO_DATE('09:00', 'HH24:MI'));


INSERT INTO Schedule (class_id, classroom_id, session_id, day_of_week, start_time,
end_time)

VALUES (2, 2, 3, 'Monday', TO_DATE('09:00', 'HH24:MI'), TO_DATE('10:00', 'HH24:MI'));


INSERT INTO Schedule (class_id, classroom_id, session_id, day_of_week, start_time,
end_time)

VALUES (3, 3, 4, 'Tuesday', TO_DATE('10:00', 'HH24:MI'), TO_DATE('11:00', 'HH24:MI'));


-- Inserting into Enrollment Table

INSERT INTO Enrollment (student_id, session_id, semester_id)

VALUES (1, 2, 1);


INSERT INTO Enrollment (student_id, session_id, semester_id)

VALUES (1, 3, 1);


INSERT INTO Enrollment (student_id, session_id, semester_id)

VALUES (2, 3, 1);


INSERT INTO Enrollment (student_id, session_id, semester_id)

VALUES (3, 4, 2);


-- Inserting into GradeComponent Table

INSERT INTO GradeComponent (component_name)

VALUES ('الواجبات');
```

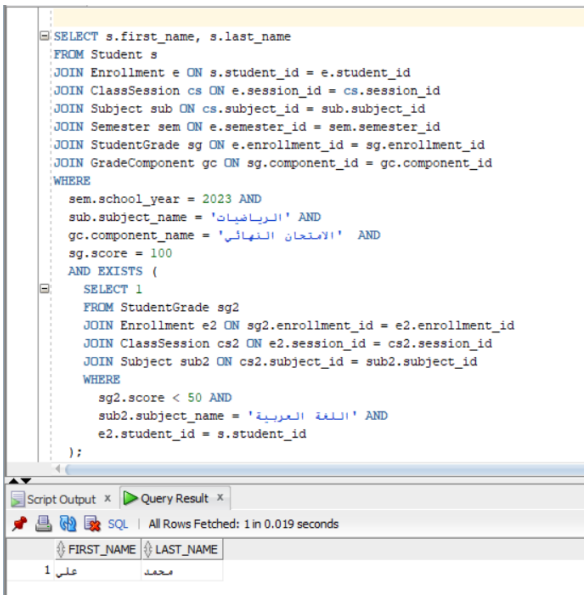
```
INSERT INTO GradeComponent (component_name)
VALUES ('الاختبارات');

INSERT INTO GradeComponent (component_name)
VALUES ('المشاركة');

INSERT INTO GradeComponent (component_name)
VALUES ('النشاط');

INSERT INTO GradeComponent (component_name)
VALUES ('الامتحان النهائي');

-- Inserting into StudentGrade Table
INSERT INTO StudentGrade (enrollment_id, component_id, score)
VALUES (1, 1, 80.00);

INSERT INTO StudentGrade (enrollment_id, component_id, score)
VALUES (1, 2, 75.00);

INSERT INTO StudentGrade (enrollment_id, component_id, score)
VALUES (1, 3, 90.00);

INSERT INTO StudentGrade (enrollment_id, component_id, score)
VALUES (1, 4, 85.00);

INSERT INTO StudentGrade (enrollment_id, component_id, score)
VALUES (1, 5, 95.00);

INSERT INTO StudentGrade (enrollment_id, component_id, score)
VALUES (7, 5, 40.00);
```

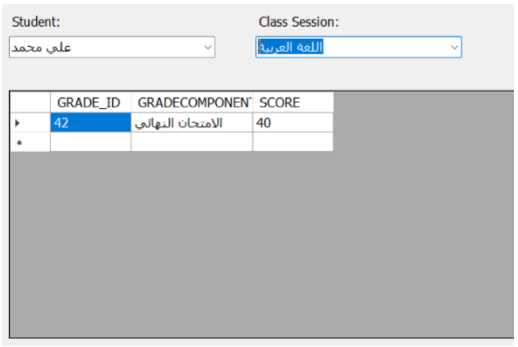
```
INSERT INTO StudentGrade (enrollment_id, component_id, score)
VALUES (2, 1, 70.00);

INSERT INTO StudentGrade (enrollment_id, component_id, score)
VALUES (2, 2, 65.00);

INSERT INTO StudentGrade (enrollment_id, component_id, score)
VALUES (2, 3, 80.00);

INSERT INTO StudentGrade (enrollment_id, component_id, score)
VALUES (2, 4, 75.00);

INSERT INTO StudentGrade (enrollment_id, component_id, score)
VALUES (2, 5, 85.00);

INSERT INTO StudentGrade (enrollment_id, component_id, score)
VALUES (3, 1, 90.00);

INSERT INTO StudentGrade (enrollment_id, component_id, score)
VALUES (3, 2, 85.00);

INSERT INTO StudentGrade (enrollment_id, component_id, score)
VALUES (3, 3, 95.00);

INSERT INTO StudentGrade (enrollment_id, component_id, score)
VALUES (3, 4, 90.00);

INSERT INTO StudentGrade (enrollment_id, component_id, score)
VALUES (3, 5, 100.00);
```

3. الاستعلامات

3.1. الاستعلام الأول: فحص رسوب اللغة العربية لنفس الطالب الذي حصل على علامة كاملة في الرياضيات







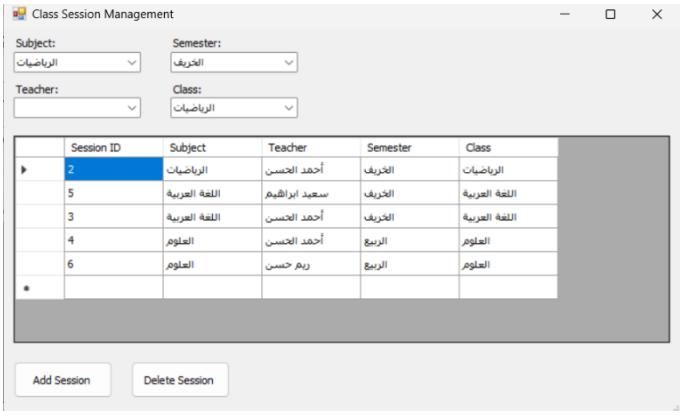

```
SELECT s.first_name, s.last_name
FROM Student s
JOIN Enrollment e ON s.student_id = e.student_id
JOIN ClassSession cs ON e.session_id = cs.session_id
JOIN Subject sub ON cs.subject_id = sub.subject_id
JOIN Semester sem ON e.semester_id = sem.semester_id
JOIN StudentGrade sg ON e.enrollment_id = sg.enrollment_id
JOIN GradeComponent gc ON sg.component_id = gc.component_id
WHERE
  sem.school_year = 2023 AND
  sub.subject_name = 'الرياضيات' AND
  gc.component_name = 'الامتحان النهائي' AND
  sg.score = 100
  AND EXISTS (
    SELECT 1
    FROM StudentGrade sg2
```

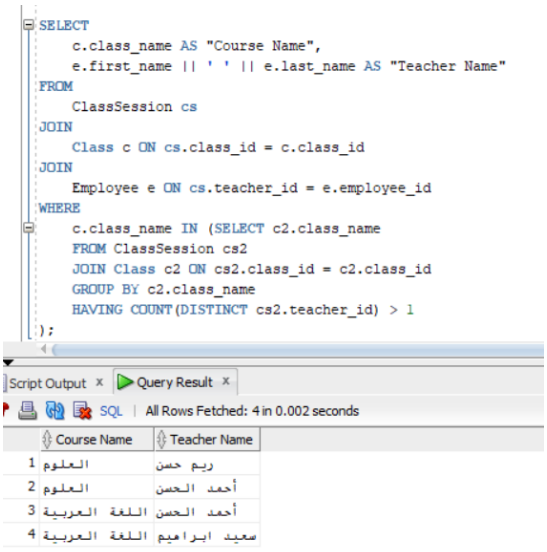
```
  JOIN Enrollment e2 ON sg2.enrollment_id = e2.enrollment_id
  JOIN ClassSession cs2 ON e2.session_id = cs2.session_id
  JOIN Subject sub2 ON cs2.subject_id = sub2.subject_id
  WHERE
    sg2.score < 50 AND
    sub2.subject_name = 'اللغة العربية' AND
    e2.student_id = s.student_id
);
```

شرح:

1. **اختيار أسماء الطلاب :**يبدأ الاستعلام باختيار اسم الطالب الأول واسم العائلة من جدول الطلاب
2. **الربط بين الجداول :**يستخدم الاستعلام العديد من عمليات الربط بين الجداول:
   - التسجيل :("e") لربط الطلاب بدروسهم المسجلة.
   - جلسة الصف :("cs") للحصول على اسم المادة في الجلسة.
   - المادة :("sub") لتحديد المادة المحددة (الرياضيات في هذه الحالة).
   - الفصل الدراسي :("sem") لتصفية النتائج بناءً على السنة الدراسية.(2023/2024)
   - درجة الطالب :("sg") لجلب درجات الطالب في كل مكون.
   - مكون الدرجة :("gc") لتصفية النتائج بناءً على مكون الدرجة (الامتحان النهائي في هذه الحالة).
3. **شروط التصفية :**تُطبّق شروط التصفية في:"WHERE"
   - sem.school_year = 2023: 2024/2023 لتصفية النتائج على أساس السنة الدراسية.
   - sub.subject_name = 'الرياضيات': 'الرياضيات' لتصفية النتائج لطلاب مادة الرياضيات.
   - gc.component_name = 'الامتحان النهائي ا': لتصفية النتائج بناءً على مكون الامتحان النهائي.
   - sg.score = 100: في (%100) لتصفية النتائج لطلاب الذين حصلوا على علامة كاملة الامتحان النهائي.
4. **الاستعلام الفرعي لفشل اللغة العربية (مع التصحيح) :**يفحص الاستعلام الفرعي (EXISTS) ما إذا كان نفس الطالب قد رسب في مادة اللغة العربية:
   - يربط الاستعلام الفرعي بين الجداول بطريقة مشابهة للاستعلام الرئيسي، ويفلتر النتائج لمادة اللغة العربية (sub2.subject_name = 'اللغة العربية').
   - sg2.score < 50: 50 لفحص ما إذا كانت درجة الطالب في اللغة العربية أقل من %.
   - **e2.student_id = s.student_id هذا يضمن أن الاستعلام الفرعي يفحص الرسوب في اللغة العربية لنفس الطالب الموجود في الاستعلام الرئيسي.**
   - أن يكون الطالب قد حصل على علامة كاملة في الرياضيات ورسب في اللغة العربية ليتم تضمينه في النتيجة.EXISTS يضمن شرط

3.2. الاستعلام الثاني: إرجاع قائمة بأسماء الدورات التي يدرسها أكثر من معلم






```
SELECT
    c.class_name AS "Course Name",
    e.first_name || ' ' || e.last_name AS "Teacher Name"
FROM
    ClassSession cs
JOIN
    Class c ON cs.class_id = c.class_id
JOIN
    Employee e ON cs.teacher_id = e.employee_id
WHERE
    c.class_name IN (SELECT c2.class_name
    FROM ClassSession cs2
    JOIN Class c2 ON cs2.class_id = c2.class_id
    GROUP BY c2.class_name
    HAVING COUNT(DISTINCT cs2.teacher_id) > 1
);
```

الشرح:

- **طريقة العمل:**

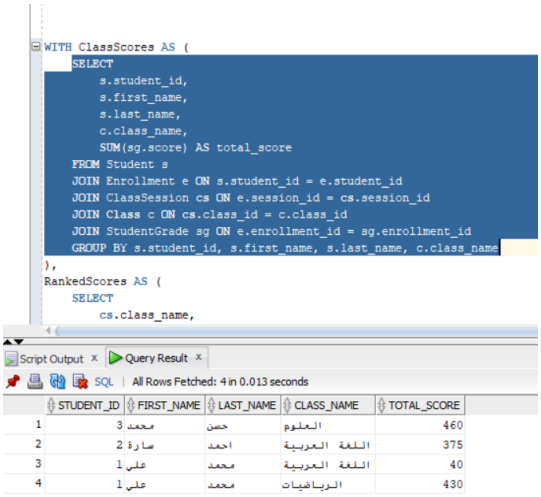
1. **الاستعلام الفرعي:**

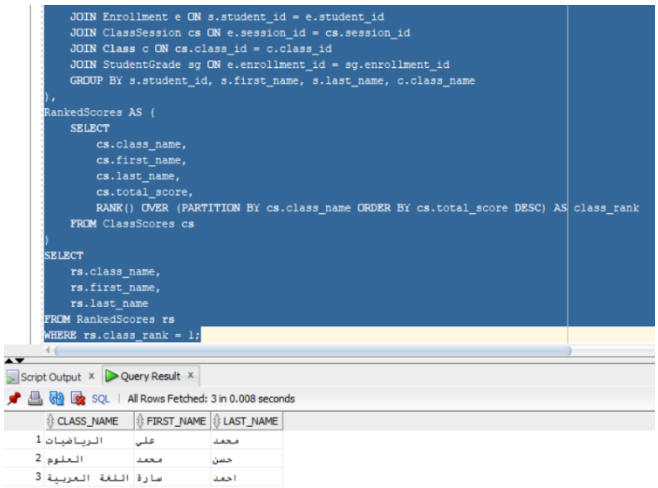
- يبدأ الكود باستعلام فرعي (بين قوسين) يُستخدم لتحديد أسماء الدورات التي يدرسها أكثر من معلم.
- يقوم الاستعلام الفرعي بالآتي:
  - اختيار اسم الدورة
  - ربط جدول `ClassSession` (مع تغيير اسمه مؤقتًا إلى `cs2`) مع جدول `Class`
  - تجميع النتائج حسب اسم الدورة.
  - فلترة النتائج لإرجاع فقط أسماء الدورات التي تم تعيين أكثر من مُعلم مُختلف لها

2. **الاستعلام الرئيسي:**

- يقوم الاستعلام الرئيسي بتحديد اسم الدورة
- يتم ربط هذه الجداول باستخدام `JOIN` بناءً على مفاتيح الربط بينها (`class_id` بين `ClassSession` و `Class`، و `teacher_id` بين `ClassSession` و `Employee`).
- فلترة النتائج بحيث يتم عرض فقط الصفوف التي يكون فيها اسم الدورة

3.3. استعلام الثالث: أسماء الطلاب الذين حصلوا على أعلى درجات في كل صف






```
WITH ClassScores AS (
    SELECT
```

```
        s.student_id,
        s.first_name,
        s.last_name,
        c.class_name,
        SUM(sg.score) AS total_score
    FROM Student s
    JOIN Enrollment e ON s.student_id = e.student_id
    JOIN ClassSession cs ON e.session_id = cs.session_id
    JOIN Class c ON cs.class_id = c.class_id
    JOIN StudentGrade sg ON e.enrollment_id = sg.enrollment_id
    GROUP BY s.student_id, s.first_name, s.last_name, c.class_name
),
RankedScores AS (
    SELECT
        cs.class_name,
        cs.first_name,
        cs.last_name,
        cs.total_score,
        RANK() OVER (PARTITION BY cs.class_name ORDER BY cs.total_score DESC) AS
class_rank
    FROM ClassScores cs
)
SELECT
    rs.class_name,
    rs.first_name,
    rs.last_name
FROM RankedScores rs
WHERE rs.class_rank = 1;
```

شرح:

1. **الخطوة الأولى: جمع درجات كل طالب في كل صف**
   - يبدأ الاستعلام بجمع جميع درجات كل طالب في كل صف باستخدام ClassScores CTE
   - تُجمع درجات كل طالب في كل صف باستخدام .SUM(sg.score)

2. **الخطوة الثانية: ترتيب الطلاب في كل صف**

- بعد ذلك، يتم ترتيب الطلاب في كل صف بناءً على مجموع درجاتهم باستخدام RankedScores CTE
- يتم استخدام الدالة RANK() لتحديد ترتيب كل طالب في صفه.

3. **الخطوة الثالثة: عرض أسماء الطلاب الأعلى رتبة**

- أخيرًا، يتم عرض أسماء الطلاب الذين حصلوا على الترتيب الأول (الرتبة الأعلى) في كل صف، وذلك باستخدام عبارة SELECT.