

# Course Registration Management System



## American International University-Bangladesh

**Course Name:** Advanced Database Management System

**Section:** B

**Department:** Computer Science and Engineering

**Submission Date:** August 28, 2023

**Project Name:** Course Registration Management System

### **Submitted By:**

Name	ID	Contribution	Percentage
Zuhair Ahmed	20-42115-1	Interface Design and Description, Schema Diagram, Table Creation, Data Insertion, Query Writing	25%
Nabila Chowdhury Joya	20-42268-1	Scenario Description, ER Diagram, Normalization, Table Creation, Query Writing	25%
Amit Podder	20-42273-1	Class Diagram, Use Case Diagram, Activity Diagram, Table Creation, Data Insertion, Normalization, Query Writing	25%
Md. Forhadul Islam	20-42091-1	Normalization, Data Insertion, Query Writing, Relational Algebra	25%

### **Submitted To:**

**JUENA AHMED NOSHIN**  
Assistant Professor, Computer Science

# Course Registration Management System

## Table of Contents

<b>Introduction .....</b>	<b>3</b>
<b>Project Proposal .....</b>	<b>3</b>
<b>Class Diagram .....</b>	<b>3</b>
<b>Use Case Diagram .....</b>	<b>4</b>
<b>Activity Diagram .....</b>	<b>5</b>
<b>Interface Design and Description.....</b>	<b>6</b>
<b>Scenario Description .....</b>	<b>10</b>
<b>ER Diagram .....</b>	<b>11</b>
<b>Normalization .....</b>	<b>12</b>
<b>Schema Diagram .....</b>	<b>17</b>
<b>Table Creation .....</b>	<b>18</b>
<b>Data Insertion .....</b>	<b>29</b>
<b>Query Writing.....</b>	<b>37</b>
<b>Relational Algebra .....</b>	<b>68</b>
<b>Conclusion .....</b>	<b>68</b>

# Course Registration Management System

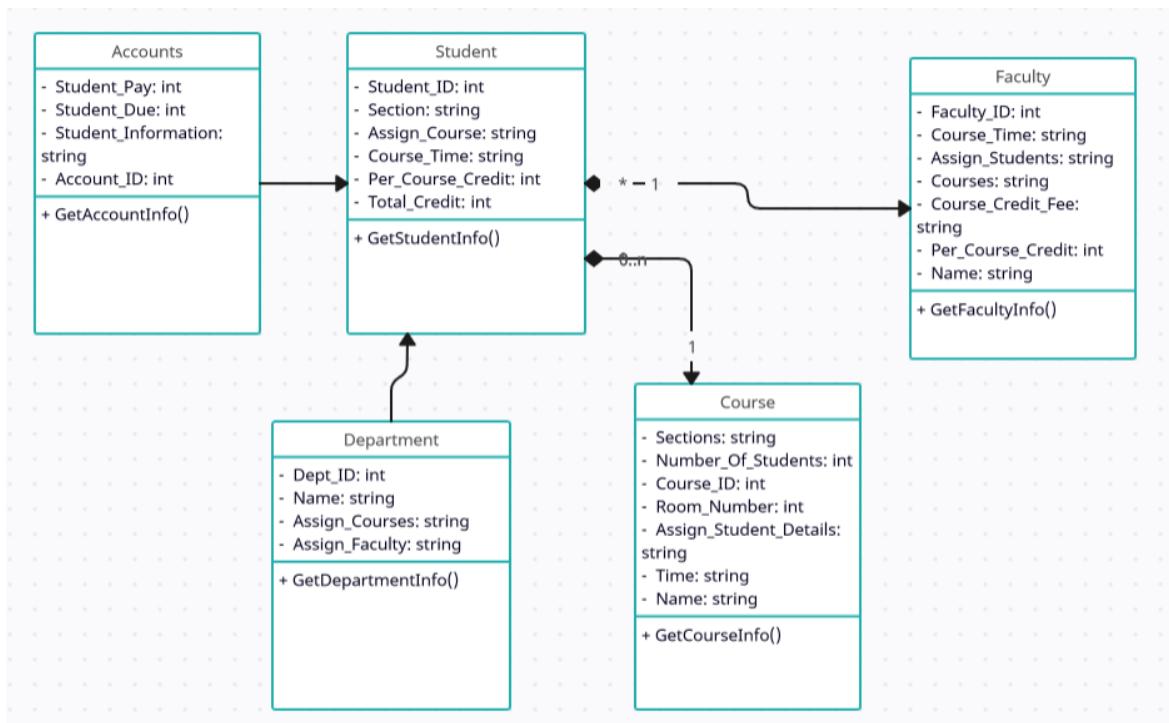
## Introduction:

A course registration management system is a software application designed to manage the process of student registration for courses offered by educational institutions. The system will provide a platform for students to view course offerings, select courses, register for courses, and manage their registration status. The system will also provide the Department with tools to manage course offerings, student registration, and student enrolment data.

## Project Proposal:

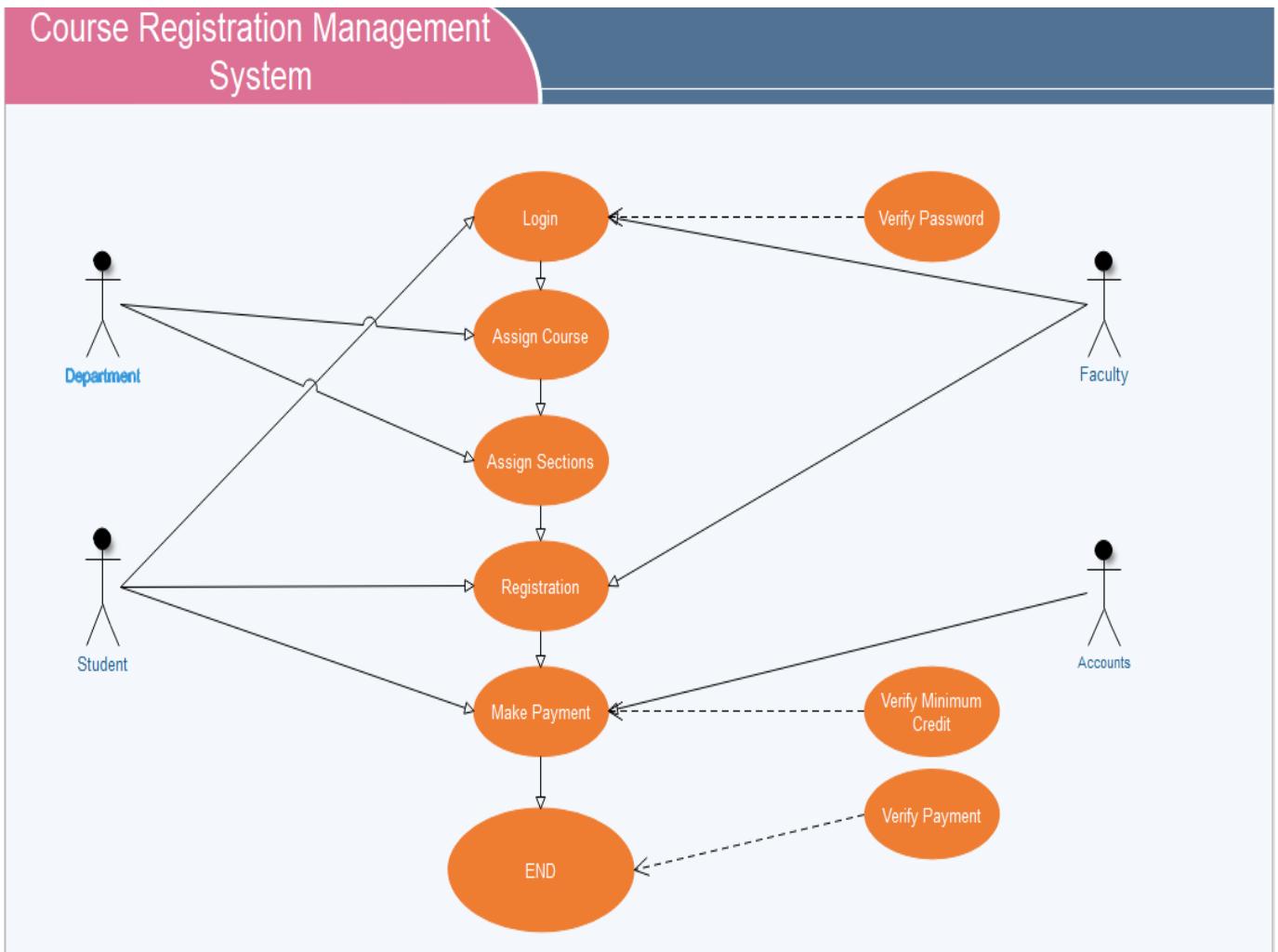
The course registration management system is a software application that will manage the registration process for students at educational institutes. This software will provide a user-friendly interface for students to view and select courses they will take. As well as allow administrative staff to access course offerings, student enrollment, and other related stuff. The faculty taking courses will also have the option to enroll at their preferred times and schedules. The course management system will be implemented using a combination of programming languages, database management systems, and web technologies. The specific tools and technologies used will depend on the requirements and preferences of the institute.

## Class Diagram:



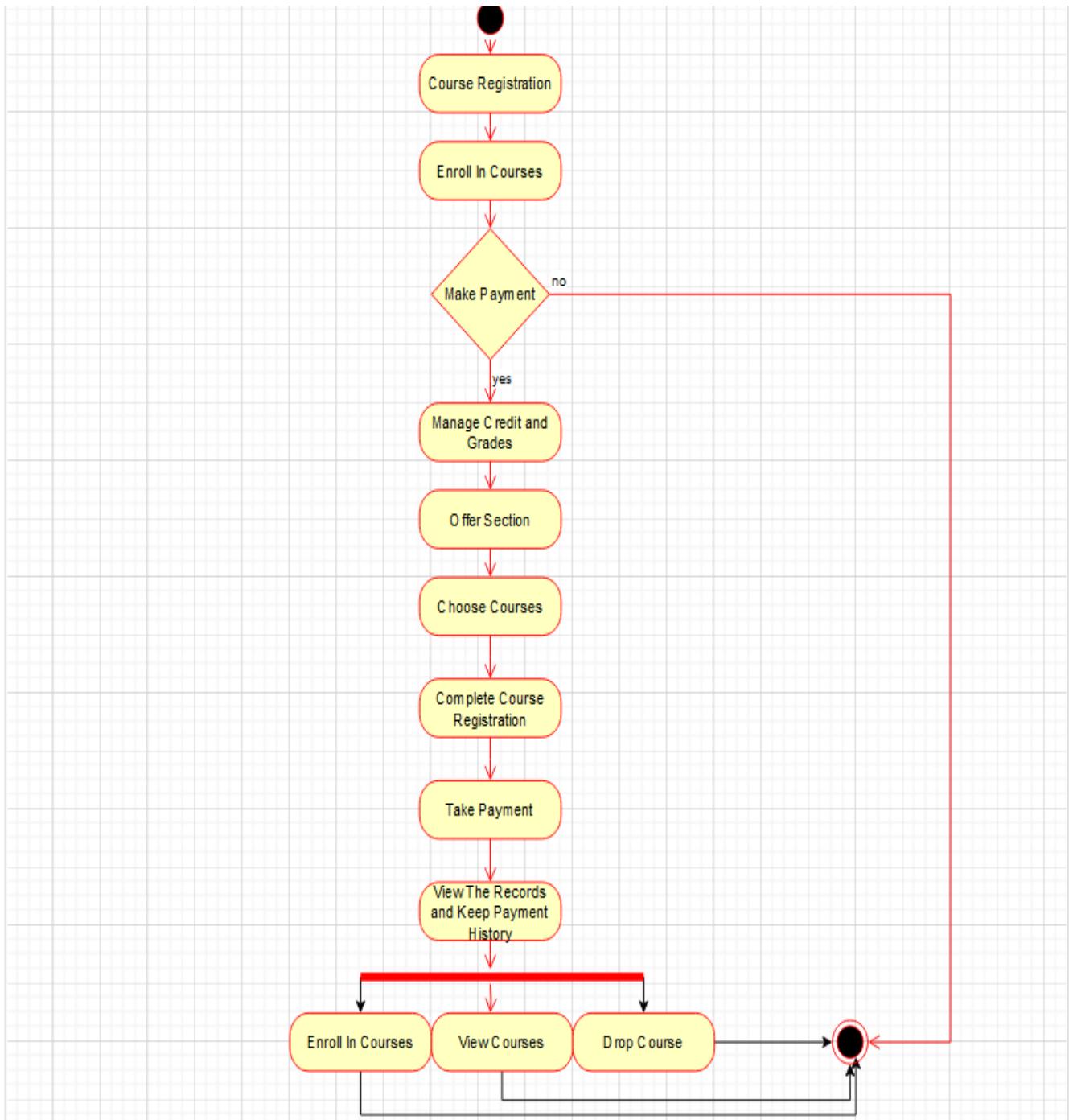
# Course Registration Management System

## Use Case Diagram:



# Course Registration Management System

## Activity Diagram:



# Course Registration Management System

## Interface Design and Description:

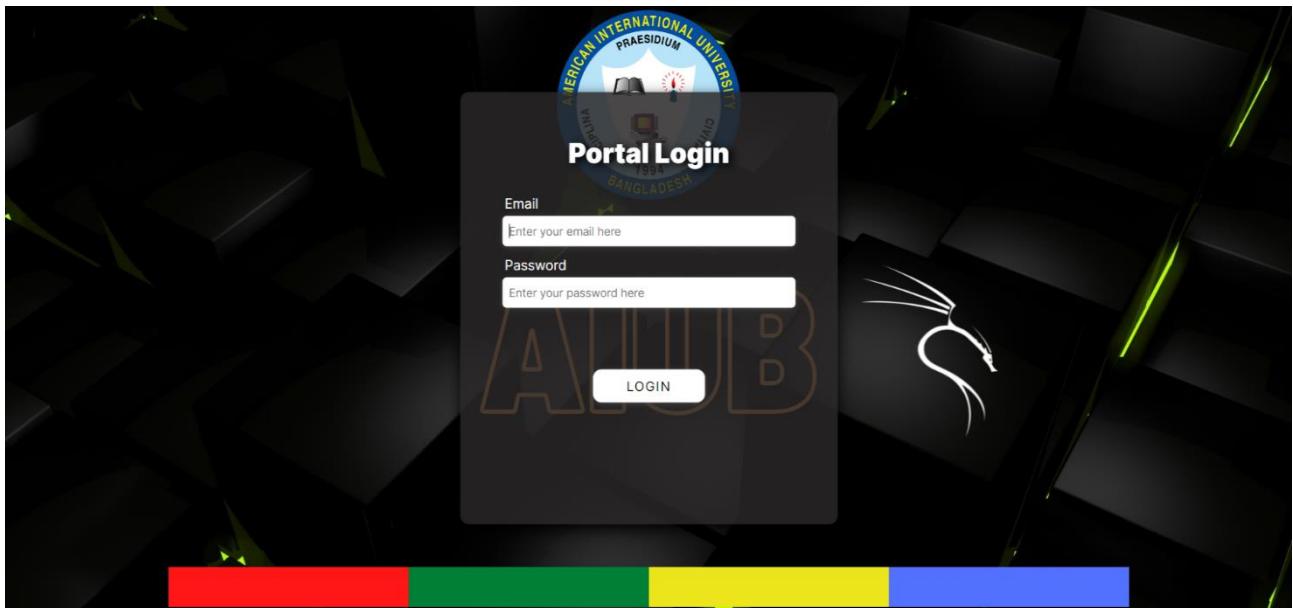


Figure 01: Portal Login

If the user is a student or faculty member, they will have access to a dedicated portal. This portal will automatically detect whether the user is a student or faculty member.

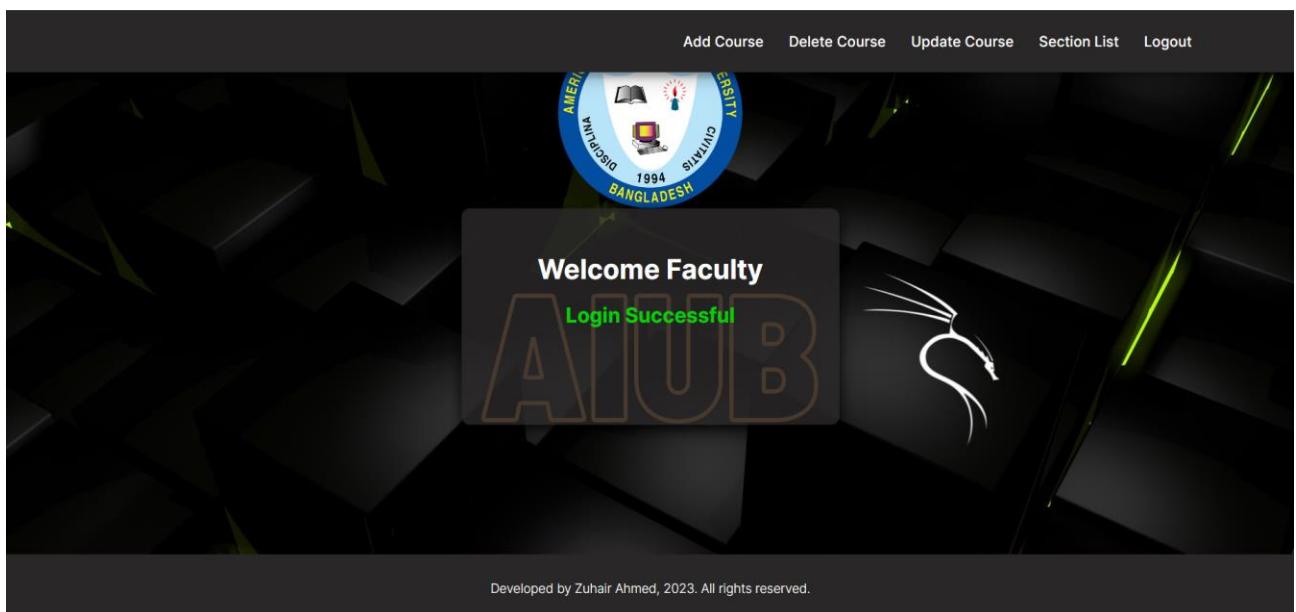


Figure 02: Faculty Dashboard

## Course Registration Management System

The Faculty Dashboard includes four buttons and a logout button. The “Add Course” button enables faculty to add new courses, while the “Delete Course” button is used to remove courses that are no longer offered, and “Update Course” is for new times or section updates. The section list displays the number of available sections and is not visible to students. Refer to [Figure: 06] for an interface view. Faculty members can log out at any time by clicking the "Logout" button.

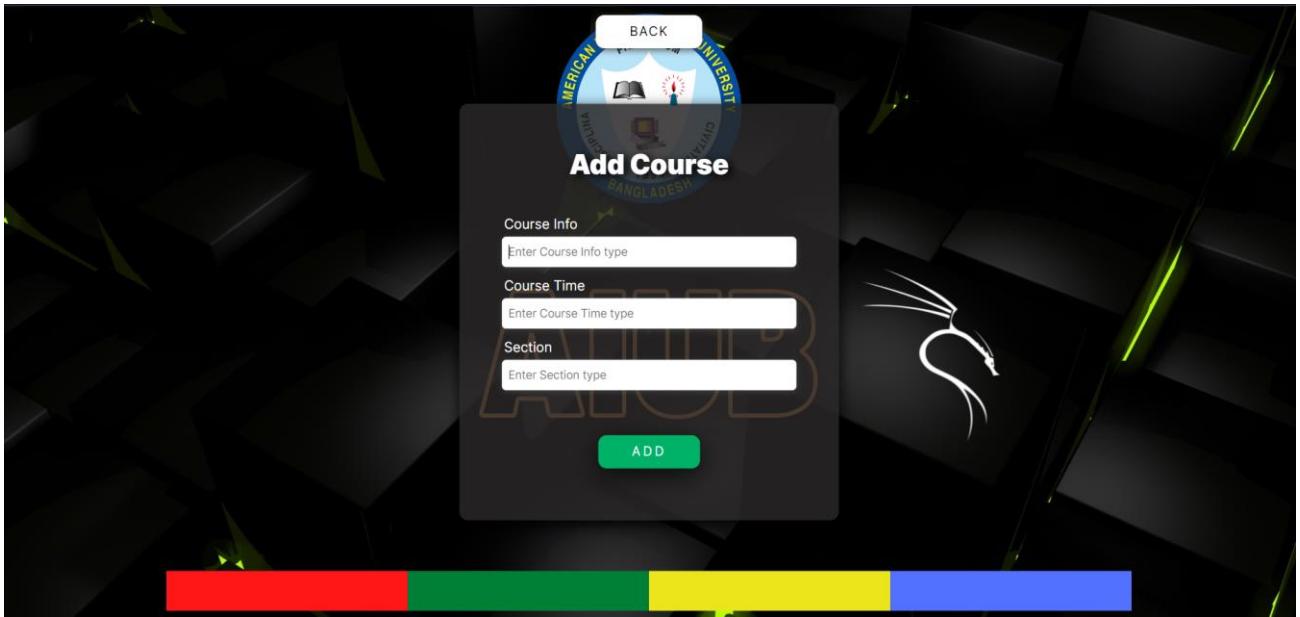


Figure 03: Add Course

A screenshot of a mobile application interface titled "Delete Course". At the top is a circular logo for "AMERICAN INTERNATIONAL UNIVERSITY - BANGLADESH". Below the title is a table with four columns: "Course Info", "Course Time", "Section", and "Action". The table contains five rows of data: CSC101 (10:30-12:30, A, DELETE), CSC102 (8.30-11.00, F, DELETE), CSC103 (10.30-12.00, B, DELETE), CSC104 (2.30-5.00, A, DELETE), and CSC105 (11.30-2.00, C, DELETE). A "BACK" button is located at the bottom left of the table. In the background, there is a large watermark of a white cat's head and the letters "AUDB". The bottom of the screen features a decorative bar divided into four colored segments: red, green, yellow, and blue.

Figure 04: Delete Course

## Course Registration Management System

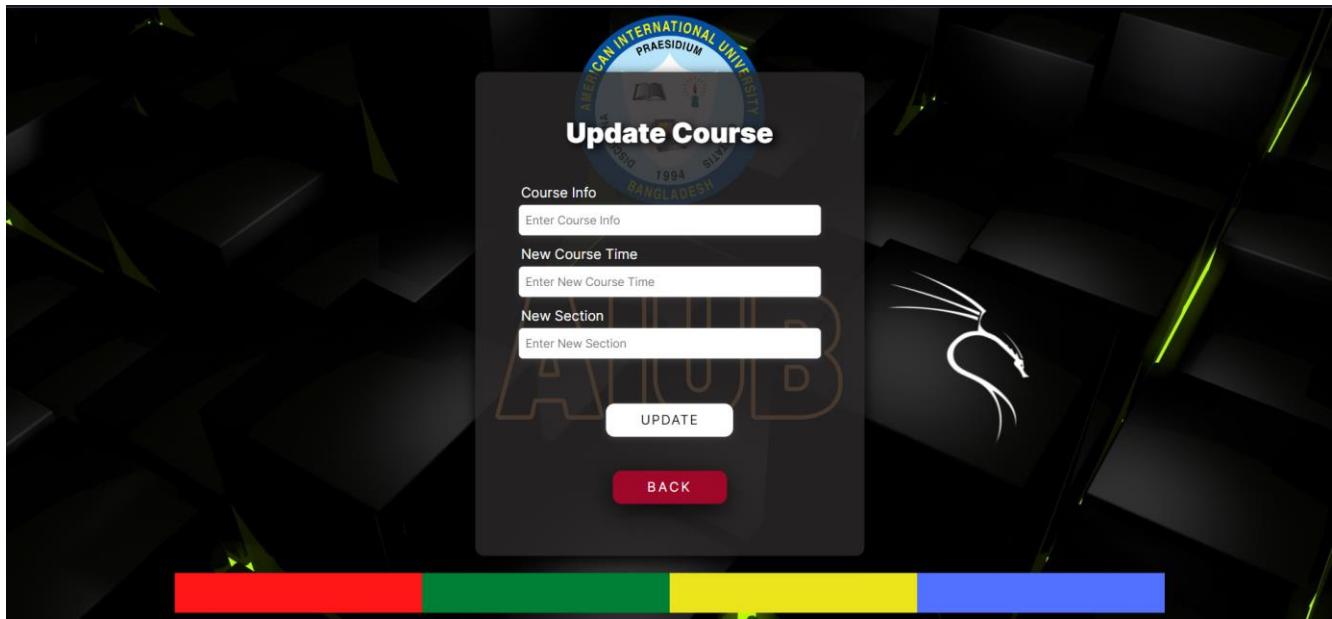


Figure 05: Update Course

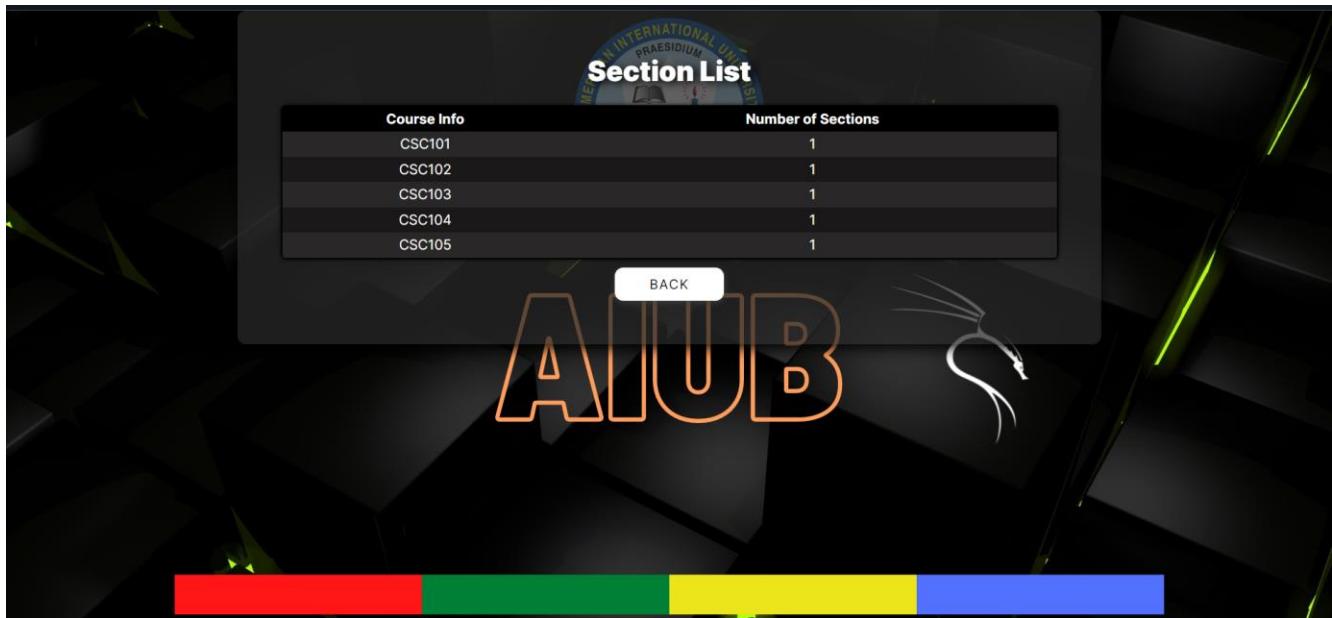


Figure 06: Section List

# Course Registration Management System

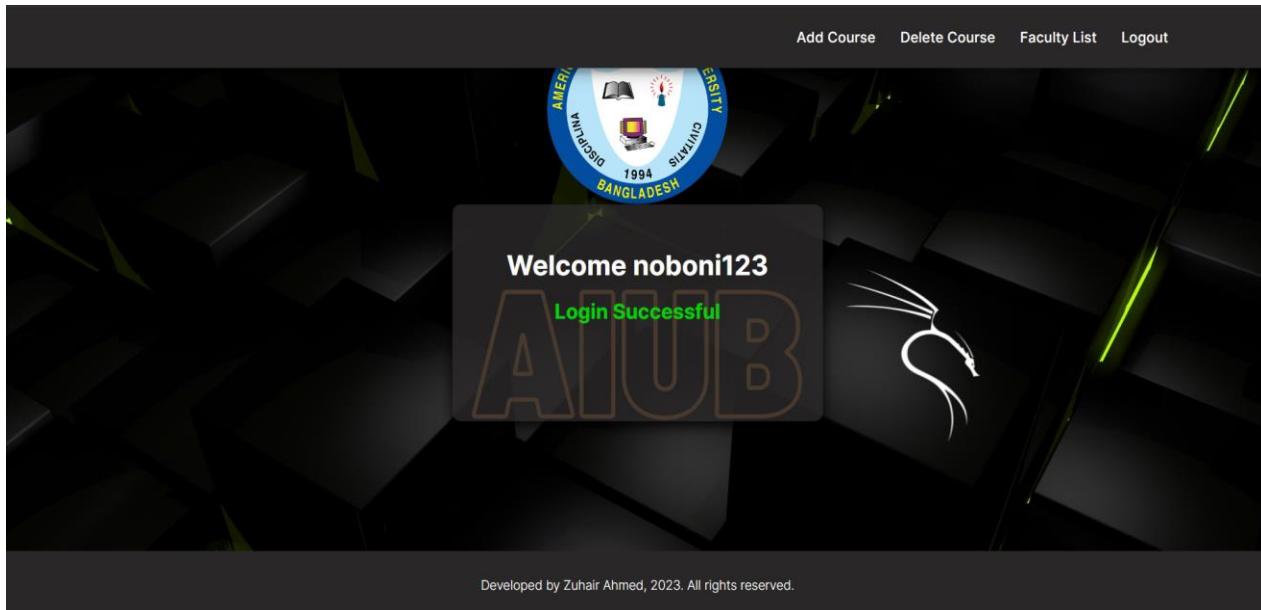


Figure 07: Student Dashboard

The student dashboard will display a welcome message with the student's username. To enroll in a course, the student can click "Add Course". If a course is deleted, the student can view it in the "Delete Course" section. If they wish to re-enroll in a previously deleted course, a fine will be imposed. The faculty list will display the departments of the course faculties. Students can log out at any time by clicking "Logout".

The Faculty List page shows a table with five rows of data. The table has three columns: "Course Id", "Class Info", and "Assigned Faculty".

Course Id	Class Info	Assigned Faculty
1200	Class101	FST
1202	Class102	BBA
1201	Class100	FST
1203	Class103	ENG
1204	Class104	ART

Figure 08: Faculty List

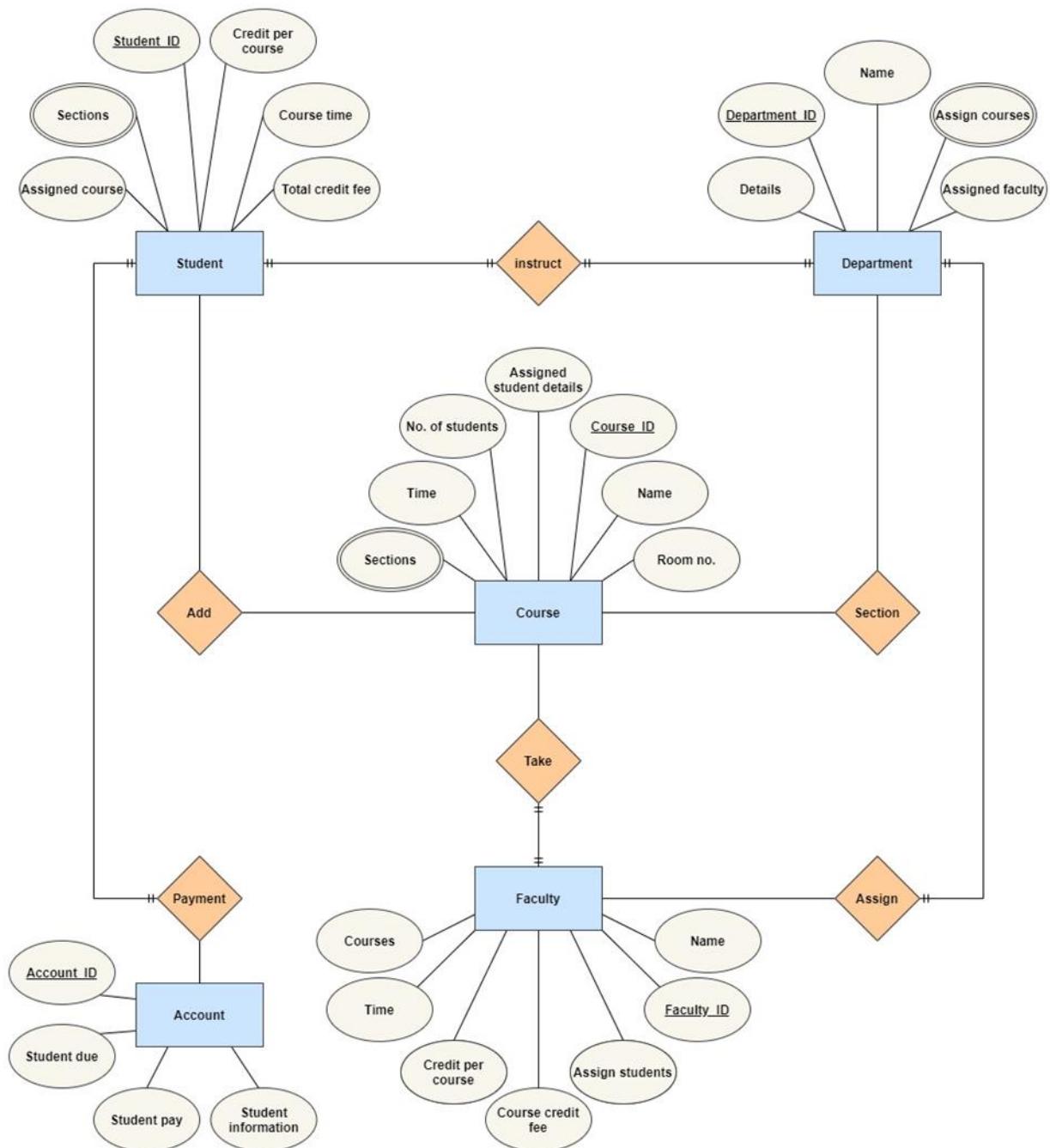
# Course Registration Management System

## **Scenario Description:**

In a university, a Course Management System has been established to efficiently manage various aspects of student enrollment, course assignments, faculty responsibilities, and financial transactions. The system is built upon a robust relational database structure, enabling seamless interactions between different components. The university's lifeblood, each student is uniquely identified by an "S\_info" and is associated with personal details, including name, username, password, and email. Students are assigned to classes and are enrolled in courses based on their academic department. Courses are the building blocks of education, identified by "Course\_info." Each course has associated details like its schedule "Course\_time" and class section "SECTION". They also have specific credit requirements and fees, outlined in the "DetailsCourse". Faculty members are responsible for teaching courses, managing student assignments, and handling various academic tasks. They are identified by a unique "Faculty\_id" and have a username, password, and email for system access. Faculty members are assigned to specific courses and oversee a group of students. Classes represent physical spaces where courses are conducted. Each class is identified by "Class\_info" and has a room number and section information. Academic departments are responsible for managing specific sets of courses and students. Each department is identified by a "Department\_id" and is associated with faculty, courses, and students. Financial transactions are recorded in the "Transaction" section. Each transaction is uniquely identified by "S\_transaction" and is related to a student's payment and due amounts. Accountants manage the financial records of students. They are identified by "Accountant\_id" and are associated with student information, transactions, students' IDs, and course information. There will be also a table containing detailed information about students, including their assigned courses, credit requirements, and course-related details. Each student is identified by a unique "Students\_id" and has links to courses, classes, and academic departments. Also, a table contains additional details about courses, such as the assigned faculty, class information, and more. Each course is identified by "Course\_id" and is linked to its related faculty, class, and course details.

# Course Registration Management System

## ER Diagram:



# Course Registration Management System

## **Normalization:**

**Payment-** Accountant\_id, Students\_pay, Students\_due, Students\_information,  
Assign\_Course, Sections, Credit\_per\_course, Studnets\_id, Course\_time, total\_credit\_fee

1NF- Sections are multivalued attributes.

Accountant\_id , Students\_pay, Students\_due, Students\_information,  
Assign\_Course, Sections, Credit\_per\_course, Studnets\_id, Course\_time,  
total\_credit\_fee

2NF-

Accountant\_id, Students\_pay, Students\_due, Students\_information  
Studnets\_id, Assign\_Course, Sections, Credit\_per\_course, Course\_time,  
total\_credit\_fee

3NF-

Accountant\_id, Students\_information.  
S\_transaction, Students\_pay, Students\_due.  
Studnets\_id, Assign\_Course, Credit\_per\_course, total\_credit\_fee.  
Course\_info, Sections, Course\_time.

## Final Table-

1. Accountant\_id, Students\_information, S\_transaction, Studnets\_id, Course\_info.
2. S\_transaction, Students\_pay, Students\_due.
3. Studnets\_id, Assign\_Course, Credit\_per\_course, total\_credit\_fee.
4. Course\_info, Sections, Course\_time.

**Instruct-** Assign\_Course, Sections, Credit\_per\_course, Students\_id, Course\_time, total\_credit\_fee, Details, Department\_id, Name, Assigned\_courses, Assigned\_faculty.

1NF- Sections and Assigned Courses are multivalued attributes.

Studnets\_id, Assign\_Course, Sections, Credit\_per\_course, Course\_time, total\_credit\_fee,  
Details, Department\_id, Name, Assigned\_courses, Assigned\_faculty.

2NF-

Studnets\_id, Assign\_Course, Sections, Credit\_per\_course, Course\_time,  
total\_credit\_fee.

Department\_id, Assigned\_courses, Details, Name, Assigned\_faculty.

3NF-

Studnets\_id, Assign\_Course, Credit\_per\_course, total\_credit\_fee.  
Course\_info, Sections, Course\_time.  
Department\_id, Assigned\_courses, Assigned\_faculty.  
S\_info, Details, Name.

# Course Registration Management System

## Final Table-

1. Students\_id, Assign Course, Credit per course, total credit fee, Course\_info, Department\_id, S\_info.
2. Course\_info, Sections, Course time.
3. Department\_id, Assigned course, Assigned faculty.
4. S\_info, Details, Name.

**Add-** Assign Course, Sections, Credit per course, Students\_id, Course time, total credit fee, Sections, time, Number of students, assigned student details, Course\_id, Name, Room number.

1NF- Sections are multivalued attributed.

Assign Course, Sections, Credit per course, Students\_id, Course time, total credit fee, sections, time, Number of students, assigned student details, Course\_id, Name, Room number.

2NF-

Students\_id, Assign Course, Sections, Credit per course, Course time, total credit fee.

Course\_id, Sections, time, Number of students, assigned student details, Name, Room number.

3NF-

Students\_id, Assign Course, Credit per course, total credit fee.

Course\_info, Sections, Course time.

Course\_id, time, Number of students, assigned student details, Name.

Class\_info, Sections, room number.

## Final Table-

1. Students\_id, Assign Course, Credit per course, total credit fee, Course\_info, Course\_id, Class\_info,
2. Course\_info, Sections, Course time.
3. Course\_id, time, Number of students, assigned student details, Name.
4. Class\_info, Sections, room number.

**Take-** Sections, time, Number of students, assigned student details, Course\_id, Name, Room number, Courses, time, credit per course, Course credit fee, assigned students, Faculty\_id, Name.

1NF- Sections and courses are multivalued attribute.

Sections, time, Number\_of\_students, assigned\_student\_details, Course\_id, Name, Room number, Courses, time, credit per course, Course credit fee, assigned students, Faculty\_id, Name.

# Course Registration Management System

2NF-

Course\_id, Sections, time, Number of students, assigned student details, Name, Room number.

Faculty\_id, Courses, time, credit per course, Course credit fee, assigned students, Name.

3NF-

Course\_id, time, Number of students, assigned student details, Name.

Class\_info, Sections, room number.

Faculty\_id, Courses, time, assigned students, Name.

Course\_d, credit per course, Course credit fee.

Final Table-

1. Course\_id, time, Number of students, assigned student details, Name, Class\_info,  
Faculty\_id, Course\_d,
2. Class\_info, Sections, room number.
3. Faculty\_id, Courses, time, assigned students, Name.
4. Course\_d, credit per course, Course credit fee.

**Section-** Details, Department\_id, Name, Assigned courses, Assigned faculty, Sections, time, Number of students, assigned student details, Course\_id, Name, Room number.

1NF- Assigned courses and sections are multivalued attribute.

Details, Department\_id, Name, Assigned faculty, Sections, time, Number of students, assigned student details, Course\_id, Name, Room number.

2NF-

Department\_id, Details, Name, Assigned courses, Assigned faculty.

Course\_id, Sections, time, Number of students, assigned student details, Name, Room number.

3NF-

Department\_id, Assigned course, Assigned faculty.

S\_info, Details, Name.

Course\_id, time, Number of students, assigned student details, Name.

Class\_info, Sections, room number.

# Course Registration Management System

Final Table-

1. Department\_id, Assigned course, Assigned faculty, s\_info, course\_id, class\_info
2. S\_info, Details, Name.
3. Course\_id, time, Number of students, assigned student details, Name.
4. Class\_info, Sections, room number.

**Assign-** Details, Department\_id, Name, Assigned courses, Assigned faculty, Courses, time, credit per course, Course credit fee, assigned students, Faculty\_id, Name.

1NF- Assigned Courses and courses are multivalued attributes.

Details, Department\_id, Name, Assigned courses, Assigned faculty, Courses, time, credit per course, Course credit fee, assigned students, Faculty\_id, Name.

2NF-

Department\_id, Details, Name, Assigned courses, Assigned faculty.

Faculty\_id, Courses, time, credit per course, Course credit fee, assigned students, Name.

3NF-

Department\_id, Assigned course, Assigned faculty.

S\_info, Details, Name.

Faculty\_id, Courses, time, assigned students, Name.

Course\_d, credit per course, Course credit fee.

Final table-

1. Department\_id, Assigned course, Assigned faculty, s\_info, faculty\_id, course\_d
2. S\_info, Details, Name.
3. Faculty\_id, Courses, time, assigned students, Name.
4. Course\_d, credit per course, Course credit fee.

# Course Registration Management System

## Table after normalization

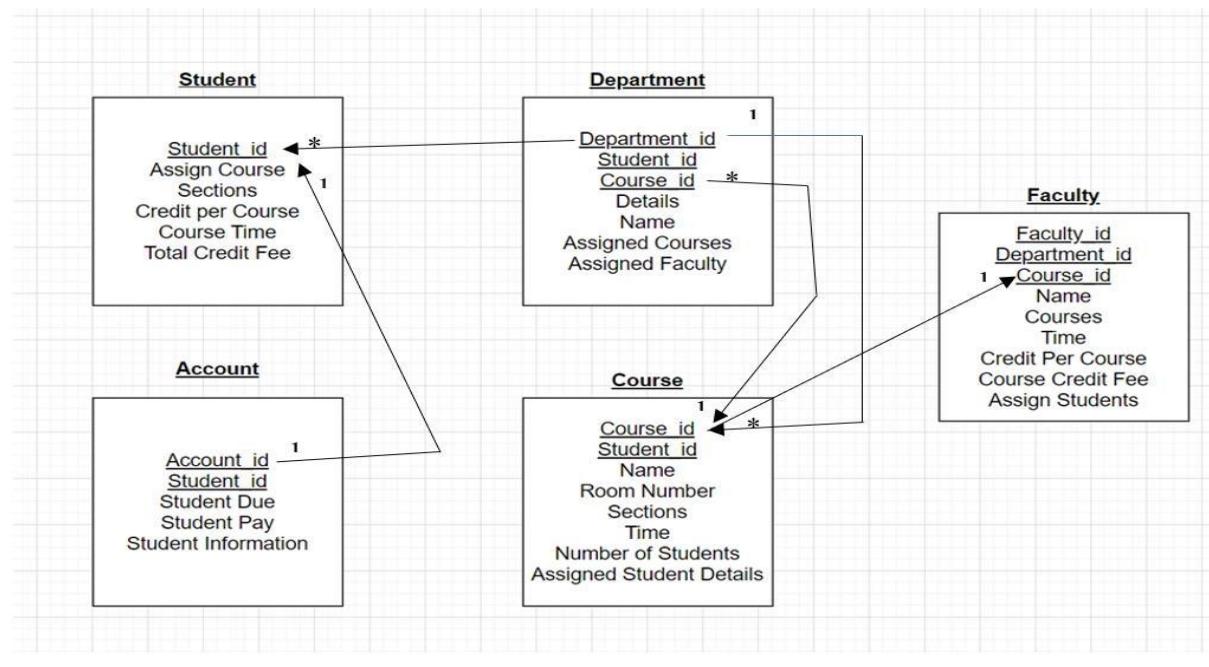
1. Accountant\_id, Students information, S\_transaction, Studnets\_id, Course\_info
2. S\_transaction, Students\_pay, Students\_due
3. Studnets\_id, Assign Course, Credit per course, total credit fee
4. Course\_info, Course time
5. Course\_info, Sections -**Composite PK**
6. Studnets\_id, Assign Course, Credit per course, total credit fee, Course\_info, Department\_id, S\_info
7. Course\_info, Course time
8. Course\_info, Sections -**Composite PK**
9. Department\_id, Assigned faculty
10. Department\_id, Assigned course -**Composite PK**
11. S\_info, Details, Name
12. Studnets\_id, Assign Course, Credit per course, total credit fee, Course\_info, Course\_id, Class\_info
13. Course\_info, Course time
14. Course\_info, Sections -**Composite PK**
15. Course\_id, time, Number of students, assigned student details, Name
16. Class\_info, room\_number
17. Class\_info, Sections -**Composite PK**
18. Course\_id, time, Number of students, assigned student details, Name, Class\_info, Faculty\_id, Course\_d
19. Class\_info, room\_number
20. Class\_info, Sections -**Composite PK**
21. Faculty\_id, Courses, time, assigned students, Name
22. Course\_d, credit per course, Course credit fee
23. Department\_id, Assigned faculty, s\_info, course\_id, class\_info
24. Department\_id, Assigned course -**Composite PK**
25. S\_info, Details, Name
26. Course\_id, time, Number of students, assigned student details, Name
27. Class\_info, room number
28. Class\_info, Sections -**Composite PK**
29. Department\_id, Assigned faculty, s\_info, faculty\_id, course\_d
30. Department\_id, Assigned course -**Composite PK**
31. S\_info, Details, Name
32. Faculty\_id, Courses, time, assigned students, Name
33. Course\_d, credit per course, Course credit fee

# Course Registration Management System

## After removing repetition final tables are

1. Accountant\_id, Students information, S\_transaction, Studnets\_id, Course\_info
2. S\_transaction, Students\_pay, Students\_due
3. Studnets\_id, Assign Course, Credit per course, total credit fee, Course\_info, Department\_id, S\_info, Course\_id, Class\_info
4. Course\_info, Course time, Sections
5. Department\_id, Assigned faculty, Assigned\_courses, s\_info, course\_id, class\_info, faculty\_id, course\_d
6. S\_info, Details, Name
7. Course\_id, time, Number of students, assigned student details, Name, Class\_info, Faculty\_id, Course\_d
8. Class\_info, room number, Sections
9. Faculty\_id, Courses, time, assigned students, Name
10. Course\_d, credit per course, Course credit fee

## Schema Diagram:



# Course Registration Management System

## Table Creation:

### Create User and Giving Permissions

```
CREATE USER ADMS IDENTIFIED BY "tiger";
```

```
GRANT CREATE VIEW, CONNECT, RESOURCE, CREATE SESSION, UNLIMITED TABLESPACE TO ADMS;
```

```
SQL> conn
Enter user-name: system
Enter password:
Connected.
SQL> CREATE USER ADMS IDENTIFIED BY "tiger";

User created.

SQL> GRANT CREATE VIEW, CONNECT, RESOURCE, CREATE SESSION, UNLIMITED TABLESPACE TO ADMS;

Grant succeeded.
```

## Transaction Table

```
CREATE TABLE Transaction (
    S_transaction NUMBER(10) CONSTRAINT pk_trns PRIMARY KEY,
    Student_pay NUMBER(10),
    Student_due NUMBER(10)
);
```

```
CREATE SEQUENCE Transaction_seq
START WITH 1200
INCREMENT BY 1
MAXVALUE 4000
NOCYCLE
NOCACHE;
```

```
CREATE INDEX Transaction_idx ON Transaction(S_transaction, Student_pay, Student_due);
```

## Course Registration Management System

```
SQL> CREATE TABLE Transaction (
  2   S_transaction NUMBER(10) CONSTRAINT pk_trns PRIMARY KEY,
  3   Student_pay NUMBER(10),
  4   Student_due NUMBER(10)
  5 );

Table created.

SQL>
SQL> CREATE SEQUENCE Transaction_seq
  2   START WITH 1200
  3   INCREMENT BY 1
  4   MAXVALUE 4000
  5   NOCYCLE
  6   NOCACHE;

Sequence created.

SQL>
SQL> CREATE INDEX Transaction_indx ON Transaction(S_transaction, Student_pay, Student_due);

Index created.
```

### Table Course

```
CREATE TABLE Course (
  Course_info VARCHAR2(20) CONSTRAINT pk_Crs PRIMARY KEY,
  Course_time VARCHAR2(4),
  Section VARCHAR2(8)
);
```

```
CREATE SEQUENCE Course_seq
START WITH 1100
INCREMENT BY 1
MAXVALUE 3000
NOCYCLE
NOCACHE;
```

```
CREATE INDEX Course_indx ON Course(Course_info, Course_time, Section);
```

# Course Registration Management System

```
SQL> CREATE TABLE Course (
 2   Course_info VARCHAR2(20) CONSTRAINT pk_Crs PRIMARY KEY,
 3   Course_time VARCHAR2(4),
 4   Section VARCHAR2(8)
 5 );

Table created.

SQL>
SQL> CREATE SEQUENCE Course_seq
 2   START WITH 1100
 3   INCREMENT BY 1
 4   MAXVALUE 3000
 5   NOCYCLE
 6   NOCACHE;

Sequence created.

SQL>
SQL> CREATE INDEX Course_indx ON Course(Course_info, Course_time, Section);

Index created.
```

## Table Student

```
CREATE TABLE Student (
  S_info VARCHAR2(20) CONSTRAINT pk_std PRIMARY KEY,
  Details VARCHAR2(25),
  Name VARCHAR2(10),
  Email VARCHAR2(20),
  Username VARCHAR2(10),
  Password VARCHAR2(20)
);
```

```
CREATE SEQUENCE Student_seq
  START WITH 1000
  INCREMENT BY 1
  MAXVALUE 2000
  NOCYCLE
  NOCACHE;
```

```
CREATE INDEX Student_indx ON Student(S_info, Details, Name);
```

# Course Registration Management System

```
SQL> CREATE TABLE Student (
  2   S_info VARCHAR2(20) CONSTRAINT pk_std PRIMARY KEY,
  3   Details VARCHAR2(25),
  4   Name VARCHAR2(10),
  5   Email VARCHAR2(50),
  6   Username VARCHAR2(20),
  7   Password VARCHAR2(20)
  8 );
Table created.

SQL>
SQL> CREATE SEQUENCE Student_seq
  2   START WITH 1000
  3   INCREMENT BY 1
  4   MAXVALUE 2000
  5   NOCYCLE
  6   NOCACHE;

Sequence created.

SQL>
SQL> CREATE INDEX Student_indx ON Student(S_info, Details, Name);

Index created.
```

## Table Class

```
CREATE TABLE Class (
  Class_info VARCHAR2(20) CONSTRAINT pk_Cls PRIMARY KEY,
  room_number NUMBER(4),
  Sections VARCHAR2(8)
);
```

```
CREATE SEQUENCE Class_seq
START WITH 1000
INCREMENT BY 1
MAXVALUE 3000
NOCYCLE
NOCACHE;
```

```
CREATE INDEX Class_indx ON Class(Class_info, room_number, Sections)
```

# Course Registration Management System

```
SQL> CREATE TABLE Class (
 2   Class_info VARCHAR2(20) CONSTRAINT pk_Cls PRIMARY KEY,
 3   room_number NUMBER(4),
 4   Sections VARCHAR2(8)
 5 );
Table created.

SQL>
SQL> CREATE SEQUENCE Class_seq
 2   START WITH 1000
 3   INCREMENT BY 1
 4   MAXVALUE 3000
 5   NOCYCLE
 6   NOCACHE;

Sequence created.

SQL>
SQL> CREATE INDEX Class_indx ON Class(Class_info, room_number, Sections);
Index created.
```

## Table Details Course

```
CREATE TABLE DetailsCourse (
  Course_d VARCHAR2(20) CONSTRAINT pk_crsdtls PRIMARY KEY,
  Credit_per_course NUMBER(4),
  Course_credit_fee NUMBER(8)
);

CREATE SEQUENCE CourseDetails_seq
START WITH 1300
INCREMENT BY 1
MAXVALUE 2000
NOCYCLE
NOCACHE;

CREATE INDEX CourseDetails_indx ON DetailsCourse(Course_d, Credit_per_course, Course_credit_fee);
```

# Course Registration Management System

```
SQL> CREATE TABLE DetailsCourse (
  2   Course_d VARCHAR2(20) CONSTRAINT pk_crsdtls PRIMARY KEY,
  3   Credit_per_course NUMBER(4),
  4   Course_credit_fee NUMBER(8)
  5 );

Table created.

SQL>
SQL> CREATE SEQUENCE CourseDetails_seq
  2   START WITH 1300
  3   INCREMENT BY 1
  4   MAXVALUE 2000
  5   NOCYCLE
  6   NOCACHE;

Sequence created.

SQL>
SQL> CREATE INDEX CourseDetails_indx ON DetailsCourse(Course_d, Credit_per_course, Course_credit_fee);

Index created.
```

## Table Faculty

```
CREATE TABLE Faculty (
  Faculty_id NUMBER(8) CONSTRAINT pk_Flt PRIMARY KEY,
  Courses VARCHAR2(30),
  Time VARCHAR2(8),
  Assigned_students NUMBER(30),
  Name VARCHAR2(10),
  Email VARCHAR2(20),
  Password VARCHAR2(20)
);
```

```
CREATE SEQUENCE Faculty_seq
START WITH 1000
INCREMENT BY 1
MAXVALUE 3000
NOCYCLE
NOCACHE;
```

```
CREATE INDEX Faculty_indx ON Faculty(Faculty_id, Courses, Time, Assigned_students, Name);
```

# Course Registration Management System

```
SQL> CREATE TABLE Faculty (
  2   Faculty_id NUMBER(8) CONSTRAINT pk_Flt PRIMARY KEY,
  3   Courses VARCHAR2(30),
  4   Time VARCHAR2(8),
  5   Assigned_students NUMBER(30),
  6   Name VARCHAR2(10),
  7   Email VARCHAR2(50),
  8   Username VARCHAR2(20),
  9   Password VARCHAR2(20)
10  );
Table created.

SQL>
SQL> CREATE SEQUENCE Faculty_seq
  2   START WITH 1000
  3   INCREMENT BY 1
  4   MAXVALUE 3000
  5   NOCYCLE
  6   NOCACHE;

Sequence created.

SQL>
SQL> CREATE INDEX Faculty_indx ON Faculty(Faculty_id, Courses, Time, Assigned_students, Name);

Index created.
```

## Table Accountant

```
CREATE TABLE Accountant (
  Accountant_id NUMBER(5) CONSTRAINT pk_acc PRIMARY KEY,
  Student_Information VARCHAR2(30),
  S_transaction NUMBER(8),
  Students_id NUMBER(7),
  Course_info VARCHAR2(20)
);

ALTER TABLE Accountant ADD CONSTRAINT fk_acc_trns FOREIGN KEY (S_transaction) REFERENCES Transaction (S_transaction);

ALTER TABLE Accountant ADD CONSTRAINT fk_acc_std_id FOREIGN KEY (Students_id) REFERENCES Stdsid(Students_id);

ALTER TABLE Accountant ADD CONSTRAINT fk_acc_crs_info FOREIGN KEY (Course_info) REFERENCES Course (Course_info);
```

```
CREATE SEQUENCE Accountant_seq
START WITH 1200
INCREMENT BY 1
MAXVALUE 4000
NOCYCLE
```

# Course Registration Management System

NOCACHE;

```
CREATE INDEX Accountant_indx ON Accountant(Accountant_id, Student_Information, S_transaction, Students_id, Course_info);
```

```
SQL> CREATE TABLE Accountant (
 2   Accountant_id NUMBER(5) CONSTRAINT pk_acc PRIMARY KEY,
 3   Student_Information VARCHAR2(30),
 4   S_transaction NUMBER(8),
 5   Students_id NUMBER(7),
 6   Course_info VARCHAR2(20)
 7 );
Table created.

SQL>
SQL> ALTER TABLE Accountant ADD CONSTRAINT fk_acc_trns FOREIGN KEY (S_transaction) REFERENCES Transaction (S_transaction);
Table altered.

SQL> CREATE SEQUENCE Accountant_seq
 2   START WITH 1200
 3   INCREMENT BY 1
 4   MAXVALUE 4000
 5   NOCYCLE
 6   NOCACHE;
Sequence created.

SQL>
SQL> CREATE INDEX Accountant_indx ON Accountant(Accountant_id, Student_Information, S_transaction, Students_id, Course_info);
Index created.
```

## Table Stdsid

```
CREATE TABLE Stdsid (
  Students_id NUMBER(7) CONSTRAINT pk_Stds_id PRIMARY KEY,
  Assigned_Course VARCHAR2(20),
  Credit_per_course VARCHAR2(4),
  Total_creadit_fee NUMBER(8),
  Course_info VARCHAR2(20),
  Department_id NUMBER(5),
  S_info VARCHAR2(30),
  Course_id NUMBER(5),
  class_info VARCHAR2(20)
);
```

```
ALTER TABLE Stdsid ADD CONSTRAINT fk_std_crs FOREIGN KEY (Course_info) REFERENCES Course (Course_info);
```

```
ALTER TABLE Stdsid ADD CONSTRAINT fk_std_dept FOREIGN KEY (Department_id) REFERENCES Dept (Department_id);
```

```
ALTER TABLE Stdsid ADD CONSTRAINT fk_std_S FOREIGN KEY (S_info) REFERENCES Student (S_info);
```

```
ALTER TABLE Stdsid ADD CONSTRAINT fk_std_crsid FOREIGN KEY (Course_id) REFERENCES Course_id (Course_id);
```

```
ALTER TABLE Stdsid ADD CONSTRAINT fk_std_cls FOREIGN KEY (class_info) REFERENCES Class (Class_info);
```

# Course Registration Management System

```
CREATE SEQUENCE Stdsid_seq
START WITH 1200
INCREMENT BY 1
MAXVALUE 4000
NOCYCLE
NOCACHE;
```

```
CREATE INDEX Stdsid_indx ON Stdsid(Students_id, Assigned_Course, Credit_per_course, Total_creadit_fee,
Course_info, Department_id, S_info, Course_id, class_info);
```

```
SQL> --Table Stdsid
SQL> CREATE TABLE Stdsid (
 2   Students_id NUMBER(7) CONSTRAINT pk_Stds_id PRIMARY KEY,
 3   Assigned_Course VARCHAR2(20),
 4   Credit_per_course VARCHAR2(4),
 5   Total_creadit_fee NUMBER(8),
 6   Course_info VARCHAR2(20),
 7   Department_id NUMBER(5),
 8   S_info VARCHAR2(30),
 9   Course_id NUMBER(5),
10   class_info VARCHAR2(20)
11 );
Table created.

SQL>
SQL> ALTER TABLE Stdsid ADD CONSTRAINT fk_std_crs FOREIGN KEY (Course_info) REFERENCES Course (Course_info);
Table altered.
```

```
SQL> CREATE SEQUENCE Stdsid_seq
 2   START WITH 1200
 3   INCREMENT BY 1
 4   MAXVALUE 4000
 5   NOCYCLE
 6   NOCACHE;

Sequence created.

SQL>
SQL> CREATE INDEX Stdsid_indx ON Stdsid(Students_id, Assigned_Course, Credit_per_course, Total_creadit_fee, Course_info,
  Department_id, S_info, Course_id, class_info);

Index created.
```

## Table Dept

```
CREATE TABLE Dept (
  Department_id NUMBER(5) CONSTRAINT pk_dept_id PRIMARY KEY,
  Assigned_faculty VARCHAR2(10),
  Assigned_course VARCHAR2(20),
  S_info VARCHAR2(30),
  Course_id NUMBER(5),
  class_info VARCHAR2(20),
  Faculty_id NUMBER(8),
  Course_d VARCHAR2(20)
```

# Course Registration Management System

);

```
ALTER TABLE Dept ADD CONSTRAINT fk_dept_std FOREIGN KEY (S_info) REFERENCES Student (S_info);
ALTER TABLE Dept ADD CONSTRAINT fk_dept_crs FOREIGN KEY (Course_id) REFERENCES Course_id (Course_id);
ALTER TABLE Dept ADD CONSTRAINT fk_dept_cls FOREIGN KEY (class_info) REFERENCES Class (Class_info);
ALTER TABLE Dept ADD CONSTRAINT fk_dept_fclt FOREIGN KEY (Faculty_id) REFERENCES Faculty (Faculty_id);
ALTER TABLE Dept ADD CONSTRAINT fk_dept_crsd FOREIGN KEY (Course_d) REFERENCES DetailsCourse (Course_d);
```

```
CREATE SEQUENCE Dept_seq
START WITH 1200
INCREMENT BY 1
MAXVALUE 4000
NOCYCLE
NOCACHE;
```

```
CREATE INDEX Dept_indx ON Dept(Department_id, Assigned_faculty, Assigned_course, S_info, Course_id, class_info, Faculty_id, Course_d);
```

```
SQL> --Table Dept
SQL> CREATE TABLE Dept (
 2   Department_id NUMBER(5) CONSTRAINT pk_dept_id PRIMARY KEY,
 3   Assigned_faculty VARCHAR2(10),
 4   Assigned_course VARCHAR2(20),
 5   S_info VARCHAR2(30),
 6   Course_id NUMBER(5),
 7   class_info VARCHAR2(20),
 8   Faculty_id NUMBER(8),
 9   Course_d VARCHAR2(20)
10  );
Table created.

SQL>
SQL> ALTER TABLE Dept ADD CONSTRAINT fk_dept_std FOREIGN KEY (S_info) REFERENCES Student (S_info);

Table altered.
```

```
SQL> CREATE SEQUENCE Dept_seq
 2   START WITH 1200
 3   INCREMENT BY 1
 4   MAXVALUE 4000
 5   NOCYCLE
 6   NOCACHE;

Sequence created.

SQL>
SQL> CREATE INDEX Dept_indx ON Dept(Department_id, Assigned_faculty, Assigned_course, S_info, Course_id, class_info, Faculty_id, Course_d);

Index created.
```

# Course Registration Management System

## Table Course Id

```
CREATE TABLE Course_id (
    Course_id NUMBER(5) CONSTRAINT pk_crsid PRIMARY KEY,
    Time VARCHAR2(8),
    Number_of_student VARCHAR2(20),
    Assign_student_detail VARCHAR2(30),
    Name VARCHAR2(20),
    Class_info VARCHAR2(20),
    Faculty_id NUMBER(8),
    Course_d VARCHAR2(20)
);

ALTER TABLE Course_id ADD CONSTRAINT fk_crsii_info FOREIGN KEY (Class_info) REFERENCES Class (Class_info);
ALTER TABLE Course_id ADD CONSTRAINT fk_crsii_Faculty FOREIGN KEY (Faculty_id) REFERENCES Faculty (Faculty_id);
ALTER TABLE Course_id ADD CONSTRAINT fk_crsii_Course FOREIGN KEY (Course_d) REFERENCES DetailsCourse (Course_d);

CREATE SEQUENCE Courseid_seq
START WITH 1200
INCREMENT BY 1
MAXVALUE 4000
NOCYCLE
NOCACHE;

CREATE INDEX Courseid_idx ON Course_id(Course_id, Time, Assign_student_detail, Name, Class_info, Faculty_id, Course_d);
```

# Course Registration Management System

```
SQL> CREATE TABLE Course_id (
 2   Course_id NUMBER(5) CONSTRAINT pk_crsid PRIMARY KEY,
 3   Time VARCHAR2(8),
 4   Number_of_student VARCHAR2(20),
 5   Assign_student_detail VARCHAR2(30),
 6   Name VARCHAR2(20),
 7   Class_info VARCHAR2(20),
 8   Faculty_id NUMBER(8),
 9   Course_d VARCHAR2(20)
10  );
Table created.

SQL>
SQL> ALTER TABLE Course_id ADD CONSTRAINT fk_crsii_info FOREIGN KEY (Class_info) REFERENCES Class (Class_info);
Table altered.

SQL> ALTER TABLE Course_id ADD CONSTRAINT fk_crsii_Faculty FOREIGN KEY (Faculty_id) REFERENCES Faculty (Faculty_id);
Table altered.

SQL> CREATE SEQUENCE Courseid_seq
 2   START WITH 1200
 3   INCREMENT BY 1
 4   MAXVALUE 4000
 5   NOCYCLE
 6   NOCACHE;
Sequence created.

SQL>
SQL> CREATE INDEX Courseid_indx ON Course_id(Course_id, Time, Assign_student_detail, Name, Class_info, Faculty_id, Course_d);
Index created.
```

## Data Insertion:

```
INSERT INTO Faculty (Faculty_id, Courses, Time, Assigned_students, Name, Username, Password, Email)
VALUES (1001, 'Computer Science', '13:00', 30, 'Jane', 'jane123', 'password123', 'jane@example.com');
```

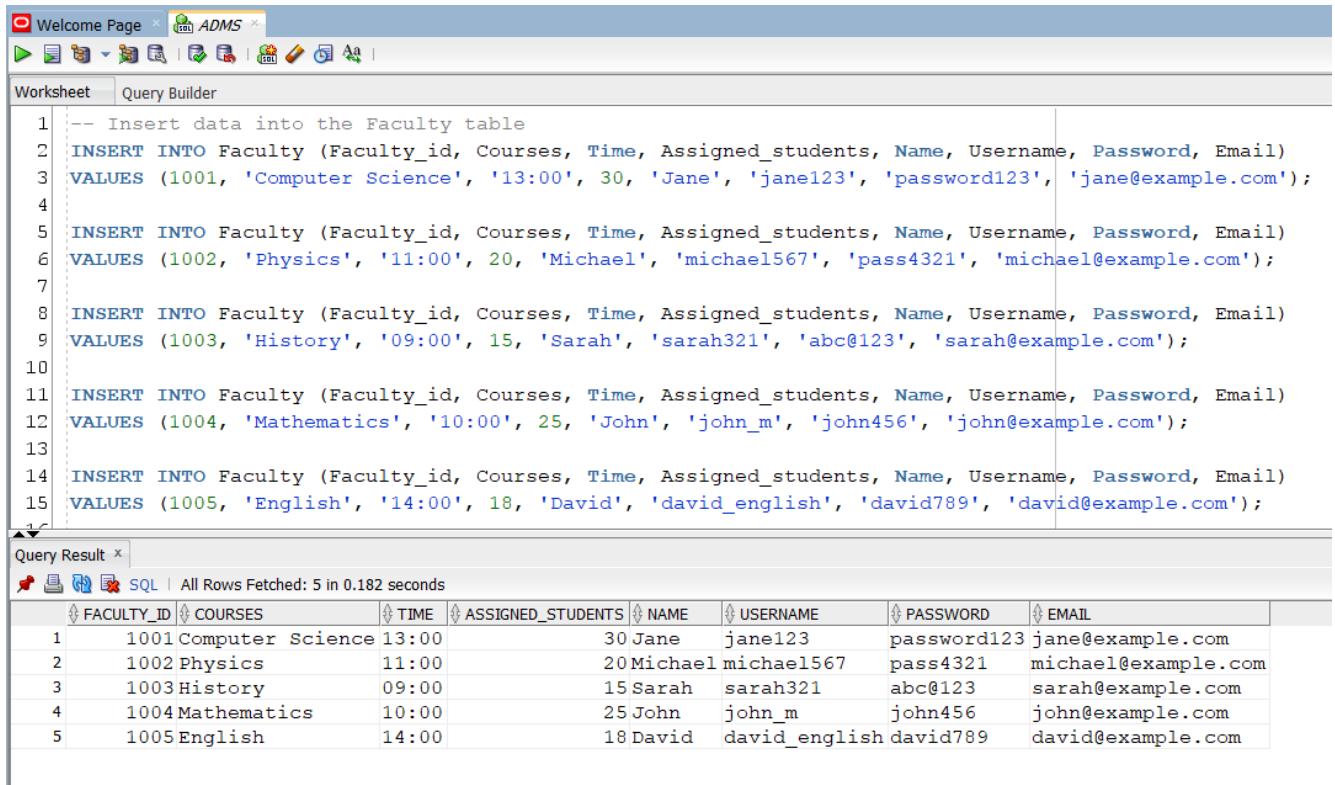
```
INSERT INTO Faculty (Faculty_id, Courses, Time, Assigned_students, Name, Username, Password, Email)
VALUES (1002, 'Physics', '11:00', 20, 'Michael', 'michael567', 'pass4321', 'michael@example.com');
```

```
INSERT INTO Faculty (Faculty_id, Courses, Time, Assigned_students, Name, Username, Password, Email)
VALUES (1003, 'History', '09:00', 15, 'Sarah', 'sarah321', 'abc@123', 'sarah@example.com');
```

```
INSERT INTO Faculty (Faculty_id, Courses, Time, Assigned_students, Name, Username, Password, Email)
VALUES (1004, 'Mathematics', '10:00', 25, 'John', 'john_m', 'john456', 'john@example.com');
```

```
INSERT INTO Faculty (Faculty_id, Courses, Time, Assigned_students, Name, Username, Password, Email)
VALUES (1005, 'English', '14:00', 18, 'David', 'david_english', 'david789', 'david@example.com');
```

# Course Registration Management System



The screenshot shows the ADMS (Advanced Database Management System) software interface. The top bar has tabs for 'Welcome Page' and 'ADMS'. Below the bar is a toolbar with various icons. The main area is divided into two sections: 'Worksheet' and 'Query Builder'. The 'Worksheet' section contains the following SQL code:

```
1 -- Insert data into the Faculty table
2 INSERT INTO Faculty (Faculty_id, Courses, Time, Assigned_students, Name, Username, Password, Email)
3 VALUES (1001, 'Computer Science', '13:00', 30, 'Jane', 'jane123', 'password123', 'jane@example.com');
4
5 INSERT INTO Faculty (Faculty_id, Courses, Time, Assigned_students, Name, Username, Password, Email)
6 VALUES (1002, 'Physics', '11:00', 20, 'Michael', 'michael1567', 'pass4321', 'michael@example.com');
7
8 INSERT INTO Faculty (Faculty_id, Courses, Time, Assigned_students, Name, Username, Password, Email)
9 VALUES (1003, 'History', '09:00', 15, 'Sarah', 'sarah321', 'abc@123', 'sarah@example.com');
10
11 INSERT INTO Faculty (Faculty_id, Courses, Time, Assigned_students, Name, Username, Password, Email)
12 VALUES (1004, 'Mathematics', '10:00', 25, 'John', 'john_m', 'john456', 'john@example.com');
13
14 INSERT INTO Faculty (Faculty_id, Courses, Time, Assigned_students, Name, Username, Password, Email)
15 VALUES (1005, 'English', '14:00', 18, 'David', 'david_english', 'david789', 'david@example.com');
```

The 'Query Result' section shows the output of the query:

FACULTY_ID	COURSES	TIME	ASSIGNED_STUDENTS	NAME	USERNAME	PASSWORD	EMAIL
1	1001 Computer Science	13:00	30	Jane	jane123	password123	jane@example.com
2	1002 Physics	11:00	20	Michael	michael1567	pass4321	michael@example.com
3	1003 History	09:00	15	Sarah	sarah321	abc@123	sarah@example.com
4	1004 Mathematics	10:00	25	John	john_m	john456	john@example.com
5	1005 English	14:00	18	David	david_english	david789	david@example.com

```
INSERT INTO Class (Class_info, room_number, Sections)
```

```
VALUES ('Class100', 106, 'C');
```

```
INSERT INTO Class (Class_info, room_number, Sections)
```

```
VALUES ('Class101', 106, 'C');
```

```
INSERT INTO Class (Class_info, room_number, Sections)
```

```
VALUES ('Class102', 105, 'A');
```

```
INSERT INTO Class (Class_info, room_number, Sections)
```

```
VALUES ('Class103', 106, 'B');
```

```
INSERT INTO Class (Class_info, room_number, Sections)
```

```
VALUES ('Class104', 106, 'B');
```

```
Select * from class;
```

# Course Registration Management System

```

    ✓ Autocommit Display 10
    INSERT INTO Class (Class_info, room_number, Sections)
    VALUES ('Class100', 106, 'C');

    INSERT INTO Class (Class_info, room_number, Sections)
    VALUES ('Class101', 106, 'C');

    INSERT INTO Class (Class_info, room_number, Sections)
    VALUES ('Class102', 105, 'A');

    INSERT INTO Class (Class_info, room_number, Sections)
    VALUES ('Class103', 106, 'B');

    INSERT INTO Class (Class_info, room_number, Sections)
    VALUES ('Class104', 106, 'B');

    Select * from class;

    INSERT INTO DetailsCourse (Course_d, Credit_per_course, Course_credit_fee)
    VALUES ('CSC101', 3, 1500);

    Results Explain Describe Saved SQL History
    CLASS_INFO ROOM_NUMBER SECTIONS
    Class100 106 C
    Class101 106 C
    Class102 105 A
    Class103 106 B
    Class104 106 B
    5 rows returned in 0.00 seconds CSV Export
  
```

Activation  
Go to Set

INSERT INTO DetailsCourse (Course\_d, Credit\_per\_course, Course\_credit\_fee)

VALUES ('CSC101', 3, 1500);

INSERT INTO DetailsCourse (Course\_d, Credit\_per\_course, Course\_credit\_fee)

VALUES ('CSC102', 3, 1500);

INSERT INTO DetailsCourse (Course\_d, Credit\_per\_course, Course\_credit\_fee)

VALUES ('CSC103', 3, 1500);

INSERT INTO DetailsCourse (Course\_d, Credit\_per\_course, Course\_credit\_fee)

VALUES ('CSC104', 3, 1500);

INSERT INTO DetailsCourse (Course\_d, Credit\_per\_course, Course\_credit\_fee)

VALUES ('CSC105', 3, 1500);

select \* from DetailsCourse;

```

    ✓ Autocommit Display 10
    Select * from class;

    INSERT INTO DetailsCourse (Course_d, Credit_per_course, Course_credit_fee)
    VALUES ('CSC101', 3, 1500);

    INSERT INTO DetailsCourse (Course_d, Credit_per_course, Course_credit_fee)
    VALUES ('CSC102', 3, 1500);

    INSERT INTO DetailsCourse (Course_d, Credit_per_course, Course_credit_fee)
    VALUES ('CSC103', 3, 1500);

    INSERT INTO DetailsCourse (Course_d, Credit_per_course, Course_credit_fee)
    VALUES ('CSC104', 3, 1500);

    INSERT INTO DetailsCourse (Course_d, Credit_per_course, Course_credit_fee)
    VALUES ('CSC105', 3, 1500);

    select * from DetailsCourse;

    INSERT INTO Student (S_info, Details, Name)

    Results Explain Describe Saved SQL History
    COURSE_D CREDIT_PER_COURSE COURSE_CREDIT_FEE
    CSC101 3 1500
    CSC102 3 1500
    CSC103 3 1500
    CSC104 3 1500
    CSC105 3 1500
    5 rows returned in 0.00 seconds CSV Export
  
```

Activation  
Go to Set

INSERT INTO Student (S\_info, Details, Name, Username, Password, Email)

VALUES ('0001', 'CSE', 'Noboni', 'noboni123', 'pass1234', 'noboni@aiub.edu');

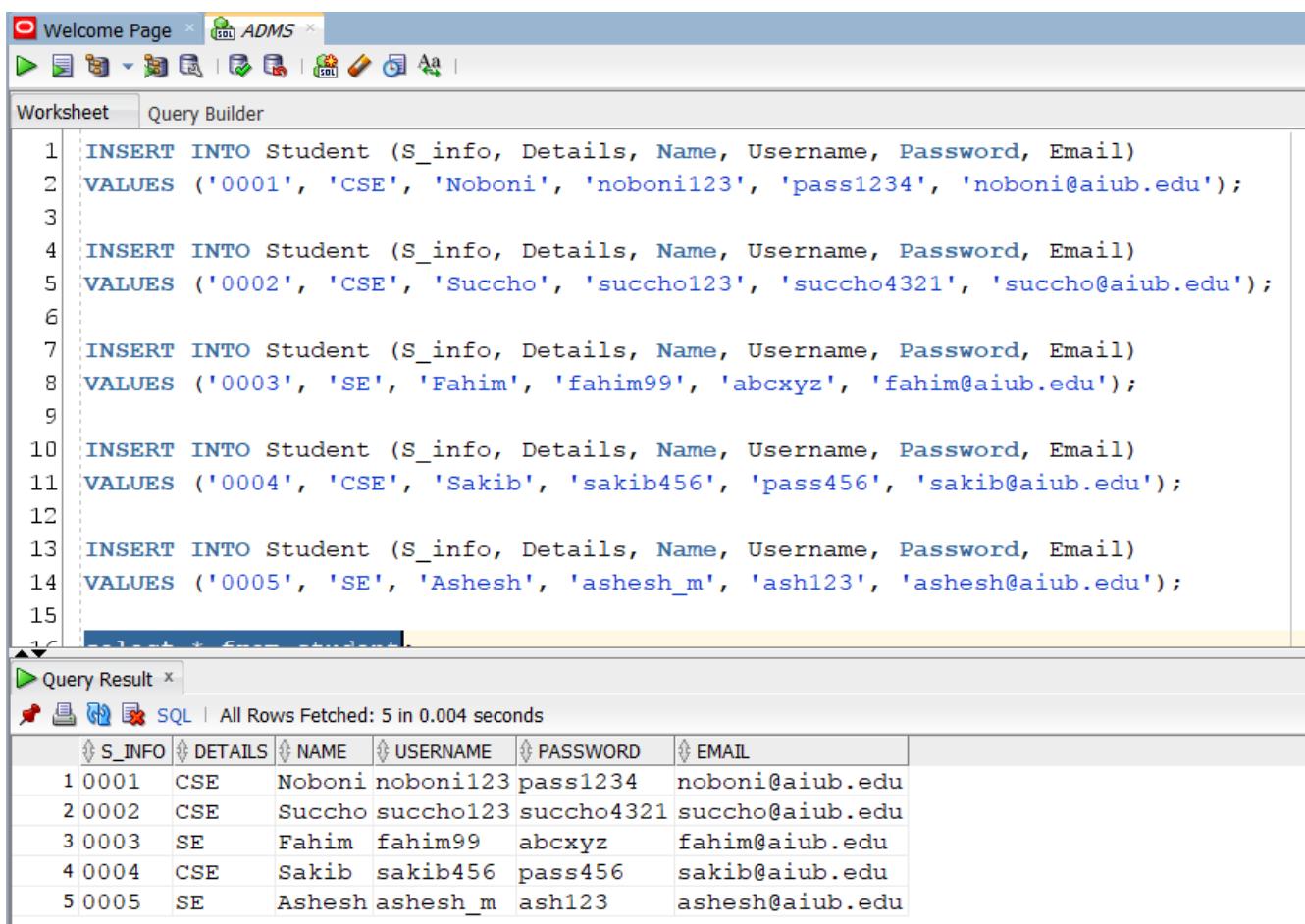
# Course Registration Management System

```
INSERT INTO Student (S_info, Details, Name, Username, Password, Email)
VALUES ('0002', 'CSE', 'Succho', 'succho123', 'succho4321', 'succho@aiub.edu');
```

```
INSERT INTO Student (S_info, Details, Name, Username, Password, Email)
VALUES ('0003', 'SE', 'Fahim', 'fahim99', 'abcxyz', 'fahim@aiub.edu');
```

```
INSERT INTO Student (S_info, Details, Name, Username, Password, Email)
VALUES ('0004', 'CSE', 'Sakib', 'sakib456', 'pass456', 'sakib@aiub.edu');
```

```
INSERT INTO Student (S_info, Details, Name, Username, Password, Email)
VALUES ('0005', 'SE', 'Ashesh', 'ashesh_m', 'ash123', 'ashesh@aiub.edu');
```



The screenshot shows the ADMS (Advanced Database Management System) interface. The top bar has tabs for 'Welcome Page' and 'ADMS'. Below the bar is a toolbar with various icons. The main area is divided into two panes: 'Worksheet' and 'Query Result'. The 'Worksheet' pane contains a code editor with numbered lines of SQL code for inserting data into the 'Student' table. The 'Query Result' pane displays a table with 5 rows of data, corresponding to the inserted records.

S_INFO	DETAILS	NAME	USERNAME	PASSWORD	EMAIL
1 0001	CSE	Noboni	noboni123	pass1234	noboni@aiub.edu
2 0002	CSE	Succho	succho123	succho4321	succho@aiub.edu
3 0003	SE	Fahim	fahim99	abcxyz	fahim@aiub.edu
4 0004	CSE	Sakib	sakib456	pass456	sakib@aiub.edu
5 0005	SE	Ashesh	ashesh_m	ash123	ashesh@aiub.edu

```
INSERT INTO Course (Course_info, Course_time, Section)
```

```
VALUES ('CSC101', '10.30-12.30', 'A');
```

```
INSERT INTO Course (Course_info, Course_time, Section)
```

```
VALUES ('CSC102', '8.30-11.00', 'F');
```

```
INSERT INTO Course (Course_info, Course_time, Section)
```

# Course Registration Management System

```
VALUES ('CSC103', '10.30-12.00', 'B');  
INSERT INTO Course (Course_info, Course_time, Section)  
VALUES ('CSC104', '2.30-5.00', 'A');  
INSERT INTO Course (Course_info, Course_time, Section)  
VALUES ('CSC105', '11.30-2.00', 'C');  
select * from Course;
```

The screenshot shows a SQL command window with the following details:

- Header:** Home > SQL > SQL Commands
- Toolbar:** Autocommit, Display 10
- Script Area:** Contains the provided SQL code.
- Results Area:** Shows the output of the query, displaying five rows of course information in a table.
- Table Headers:** COURSE\_INFO, COURSE\_TIME, SECTION
- Table Data:**

COURSE_INFO	COURSE_TIME	SECTION
CSC101	10.30-12.30	A
CSC102	8.30-11.00	F
CSC103	10.30-12.00	B
CSC104	2.30-5.00	A
CSC105	11.30-2.00	C
- Message:** 5 rows returned in 0.00 seconds
- Buttons:** Results, Explain, Describe, Saved SQL, History
- Right Panel:** Activate, Go to Settings

```
INSERT INTO Transaction (S_transaction, Student_pay, Student_due)  
VALUES (Transaction_seq.NEXTVAL, 1000, 500);  
INSERT INTO Transaction (S_transaction, Student_pay, Student_due)  
VALUES (Transaction_seq.NEXTVAL, 2000, 1500);  
INSERT INTO Transaction (S_transaction, Student_pay, Student_due)  
VALUES (Transaction_seq.NEXTVAL, 1000, 500);  
INSERT INTO Transaction (S_transaction, Student_pay, Student_due)  
VALUES (Transaction_seq.NEXTVAL, 1030, 700);  
INSERT INTO Transaction (S_transaction, Student_pay, Student_due)  
VALUES (Transaction_seq.NEXTVAL, 1200, 500);  
Select * from Transaction;
```

# Course Registration Management System

Autocommit Display 10

```

INSERT INTO Transaction (S_transaction, Student_pay, Student_due)
VALUES (Transaction_seq.NEXTVAL, 1000, 500);

INSERT INTO Transaction (S_transaction, Student_pay, Student_due)
VALUES (Transaction_seq.NEXTVAL, 2000, 1500);

INSERT INTO Transaction (S_transaction, Student_pay, Student_due)
VALUES (Transaction_seq.NEXTVAL, 1000, 500);

INSERT INTO Transaction (S_transaction, Student_pay, Student_due)
VALUES (Transaction_seq.NEXTVAL, 1030, 700);

INSERT INTO Transaction (S_transaction, Student_pay, Student_due)
VALUES (Transaction_seq.NEXTVAL, 1200, 500);

Select * from Transaction;

```

Results Explain Describe Saved SQL History

S_TRANSACTION	STUDENT_PAY	STUDENT_DUE
1200	1000	500
1201	2000	1500
1202	1000	500
1203	1030	700
1204	1200	500

5 rows returned in 0.00 seconds CSV Export

Activate Win  
Go to Settings to

```

INSERT INTO Course_id (Course_id, Time, Number_of_student, Assign_student_detail, Name, Class_info, Faculty_id, Course_d)
VALUES(Courseid_seq.NEXTVAL, '10:00 AM', '20', 'Assigned students', 'Database Systems', 'Class101', 1001, 'CSC101');

INSERT INTO Course_id (Course_id, Time, Number_of_student, Assign_student_detail, Name, Class_info, Faculty_id, Course_d)
VALUES(Courseid_seq.NEXTVAL, '11:30 AM', '30', 'Assigned students', 'Database Systems', 'Class100', 1002, 'CSC102');

INSERT INTO Course_id (Course_id, Time, Number_of_student, Assign_student_detail, Name, Class_info, Faculty_id, Course_d)
VALUES(Courseid_seq.NEXTVAL, '8:30 AM', '24', 'Assigned students', 'Database Systems', 'Class102', 1003, 'CSC103');

INSERT INTO Course_id (Course_id, Time, Number_of_student, Assign_student_detail, Name, Class_info, Faculty_id, Course_d)
VALUES(Courseid_seq.NEXTVAL, '9:40 AM', '28', 'Assigned students', 'Database Systems', 'Class103', 1004, 'CSC104');

INSERT INTO Course_id (Course_id, Time, Number_of_student, Assign_student_detail, Name, Class_info, Faculty_id, Course_d)
VALUES(Courseid_seq.NEXTVAL, '2:30 AM', '32', 'Assigned students', 'Database Systems', 'Class104', 1005, 'CSC105');

select * from Course_id;

```

Autocommit Display 10

```

INSERT INTO Course_id (Course_id, Time, Number_of_student, Assign_student_detail, Name, Class_info, Faculty_id, Course_d)
VALUES(Courseid_seq.NEXTVAL, '10:00 AM', '20', 'Assigned students', 'Database Systems', 'Class101', 1001, 'CSC101');

INSERT INTO Course_id (Course_id, Time, Number_of_student, Assign_student_detail, Name, Class_info, Faculty_id, Course_d)
VALUES(Courseid_seq.NEXTVAL, '11:30 AM', '30', 'Assigned students', 'Database Systems', 'Class100', 1002, 'CSC102');

INSERT INTO Course_id (Course_id, Time, Number_of_student, Assign_student_detail, Name, Class_info, Faculty_id, Course_d)
VALUES(Courseid_seq.NEXTVAL, '8:30 AM', '24', 'Assigned students', 'Database Systems', 'Class102', 1003, 'CSC103');

INSERT INTO Course_id (Course_id, Time, Number_of_student, Assign_student_detail, Name, Class_info, Faculty_id, Course_d)
VALUES(Courseid_seq.NEXTVAL, '9:40 AM', '28', 'Assigned students', 'Database Systems', 'Class103', 1004, 'CSC104');

INSERT INTO Course_id (Course_id, Time, Number_of_student, Assign_student_detail, Name, Class_info, Faculty_id, Course_d)
VALUES(Courseid_seq.NEXTVAL, '2:30 AM', '32', 'Assigned students', 'Database Systems', 'Class104', 1005, 'CSC105');

select * from Course_id;

```

Results Explain Describe Saved SQL History

COURSE_ID	TIME	NUMBER_OF_STUDENT	ASSIGN_STUDENT_DETAIL	NAME	CLASS_INFO	FACULTY_ID	COURSE_D
1201	10:00 AM	20	Assigned students	Database Systems	Class101	1001	CSC101
1202	11:30 AM	30	Assigned students	Database Systems	Class100	1002	CSC102
1203	8:30 AM	24	Assigned students	Database Systems	Class102	1003	CSC103
1204	9:40 AM	28	Assigned students	Database Systems	Class103	1004	CSC104
1205	2:30 AM	32	Assigned students	Database Systems	Class104	1005	CSC105

5 rows returned in 0.01 seconds CSV Export

Activate Windows  
Go to Settings to activate Wi

Application Ex

# Course Registration Management System

```

INSERT INTO Dept (Department_id, Assigned_faculty, Assigned_course, S_info, Course_id, class_info, Faculty_id, Course_d)
VALUES (101, 'FST', 'Biology', '0005', 1203, 'Class103', 1002, 'CSC101');

INSERT INTO Dept (Department_id, Assigned_faculty, Assigned_course, S_info, Course_id, class_info, Faculty_id, Course_d)
VALUES (102, 'BBA', 'Communications', '0003', 1202, 'Class104', 1003, 'CSC103');

INSERT INTO Dept (Department_id, Assigned_faculty, Assigned_course, S_info, Course_id, class_info, Faculty_id, Course_d)
VALUES (103, 'FST', 'Microprocessor', '0001', 1201, 'Class100', 1001, 'CSC102');

INSERT INTO Dept (Department_id, Assigned_faculty, Assigned_course, S_info, Course_id, class_info, Faculty_id, Course_d)
VALUES (104, 'ENG', 'English', '0004', 1204, 'Class103', 1003, 'CSC104');

INSERT INTO Dept (Department_id, Assigned_faculty, Assigned_course, S_info, Course_id, class_info, Faculty_id, Course_d)
VALUES (105, 'ART', 'Theme', '0002', 1205, 'Class101', 1005, 'CSC105');

```

The screenshot shows a MySQL command-line interface. The SQL code is pasted into the command window, and the results are displayed in a table below.

DEPARTMENT_ID	ASSIGNED_FACULTY	ASSIGNED COURSE	S_INFO	COURSE_ID	CLASS_INFO	FACULTY_ID	COURSE_D
101	FST	Biology	0005	1203	Class103	1002	CSC101
102	BBA	Communications	0003	1202	Class104	1003	CSC103
103	FST	Microprocessor	0001	1201	Class100	1001	CSC102
104	ENG	English	0004	1204	Class103	1003	CSC104
105	ART	Theme	0002	1205	Class101	1005	CSC105

5 rows returned in 0.00 seconds [CSV Export](#)

```

INSERT INTO Stdsid (Students_id, Assigned_Course, Credit_per_course, Total_creadit_fee, Course_info, Department_id, S_info, Course_id, Class_info)
VALUES (Stdsid_seq.NEXTVAL, 'Computer Science', '3', 15, 'CSC103', 104, '0002', 1204, 'Class103');

INSERT INTO Stdsid (Students_id, Assigned_Course, Credit_per_course, Total_creadit_fee, Course_info, Department_id, S_info, Course_id, Class_info)
VALUES (Stdsid_seq.NEXTVAL, 'OOP2', '3', 18, 'CSC102', 103, '0003', 1201, 'Class100');

INSERT INTO Stdsid (Students_id, Assigned_Course, Credit_per_course, Total_creadit_fee, Course_info, Department_id, S_info, Course_id, Class_info)
VALUES (Stdsid_seq.NEXTVAL, 'Compailor', '3', 12, 'CSC105', 102, '0001', 1202, 'Class104');

INSERT INTO Stdsid (Students_id, Assigned_Course, Credit_per_course, Total_creadit_fee, Course_info, Department_id, S_info, Course_id, Class_info)
VALUES (Stdsid_seq.NEXTVAL, 'Database', '3', 09, 'CSC101', 105, '0005', 1203, 'Class101');

```

# Course Registration Management System

```
INSERT INTO Stdsid (Students_id, Assigned_Course, Credit_per_course, Total_creadit_fee, Course_info, Department_id, S_info, Course_id, Class_info)
VALUES (Stdsid_seq.NEXTVAL, 'Computer network', '3', 12, 'CSC104', 101, '0004', 1205, 'Class102');

select * from stdsid;
```

The screenshot shows a MySQL query editor interface. The SQL command window contains the following code:

```
INSERT INTO Stdsid (Students_id, Assigned_Course, Credit_per_course, Total_creadit_fee, Course_info, Department_id, S_info, Course_id, Class_info)
VALUES (Stdsid_seq.NEXTVAL, 'Computer Science', '3', 15, 'CSC103', 104, '0002', 1204, 'Class103');
INSERT INTO Stdsid (Students_id, Assigned_Course, Credit_per_course, Total_creadit_fee, Course_info, Department_id, S_info, Course_id, Class_info)
VALUES (Stdsid_seq.NEXTVAL, 'OOP2', '3', 18, 'CSC102', 103, '0003', 1201, 'Class100');
INSERT INTO Stdsid (Students_id, Assigned_Course, Credit_per_course, Total_creadit_fee, Course_info, Department_id, S_info, Course_id, Class_info)
VALUES (Stdsid_seq.NEXTVAL, 'Compiler', '3', 12, 'CSC105', 102, '0001', 1202, 'Class104');
INSERT INTO Stdsid (Students_id, Assigned_Course, Credit_per_course, Total_creadit_fee, Course_info, Department_id, S_info, Course_id, Class_info)
VALUES (Stdsid_seq.NEXTVAL, 'Database', '3', 9, 'CSC101', 105, '0005', 1203, 'Class101');
INSERT INTO Stdsid (Students_id, Assigned_Course, Credit_per_course, Total_creadit_fee, Course_info, Department_id, S_info, Course_id, Class_info)
VALUES (Stdsid_seq.NEXTVAL, 'Computer network', '3', 12, 'CSC104', 101, '0004', 1205, 'Class102');

select * from Stdsid;
```

The results window displays the inserted data:

STUDENTS_ID	ASSIGNED_COURSE	CREDIT_PER_COURSE	TOTAL_CREADIT_FEE	COURSE_INFO	DEPARTMENT_ID	S_INFO	COURSE_ID	CLASS_INFO
1201	Computer Science	3	15	CSC103	104	0002	1204	Class103
1203	OOP2	3	18	CSC102	103	0003	1201	Class100
1204	Compiler	3	12	CSC105	102	0001	1202	Class104
1206	Database	3	9	CSC101	105	0005	1203	Class101
1207	Computer network	3	12	CSC104	101	0004	1205	Class102

5 rows returned in 0.00 seconds [CSV Export](#)

```
INSERT INTO Accountant (Accountant_id, Student_Information, S_transaction, Students_id, Course_info)
VALUES (4450, 'payment', '1200', '1204', 'CSC104');

INSERT INTO Accountant (Accountant_id, Student_Information, S_transaction, Students_id, Course_info)
VALUES (4451, 'payment', '1202', '1207', 'CSC102');

INSERT INTO Accountant (Accountant_id, Student_Information, S_transaction, Students_id, Course_info)
VALUES (4452, 'payment', '1200', '1206', 'CSC101');

INSERT INTO Accountant (Accountant_id, Student_Information, S_transaction, Students_id, Course_info)
VALUES (4453, 'payment', '1203', '1203', 'CSC103');

INSERT INTO Accountant (Accountant_id, Student_Information, S_transaction, Students_id, Course_info)
VALUES (4454, 'payment', '1202', '1201', 'CSC105');

select * from Accountant;
```

# Course Registration Management System

The screenshot shows a MySQL command-line interface. The SQL tab contains the following code:

```
INSERT INTO Accountant (Accountant_id, Student_Information, S_transaction, Students_id, Course_info)
VALUES (4450, 'payment', '1200', '1204', 'CSC104');
INSERT INTO Accountant (Accountant_id, Student_Information, S_transaction, Students_id, Course_info)
VALUES (4451, 'payment', '1202', '1207', 'CSC102');
INSERT INTO Accountant (Accountant_id, Student_Information, S_transaction, Students_id, Course_info)
VALUES (4452, 'payment', '1200', '1206', 'CSC101');
INSERT INTO Accountant (Accountant_id, Student_Information, S_transaction, Students_id, Course_info)
VALUES (4453, 'payment', '1203', '1203', 'CSC103');
INSERT INTO Accountant (Accountant_id, Student_Information, S_transaction, Students_id, Course_info)
VALUES (4454, 'payment', '1202', '1201', 'CSC105');

Select * From Accountant;
```

The Results tab displays the following table:

ACCOUNTANT_ID	STUDENT_INFORMATION	S_TRANSACTION	STUDENTS_ID	COURSE_INFO
4450	payment	1200	1204	CSC104
4451	payment	1202	1207	CSC102
4452	payment	1200	1206	CSC101
4453	payment	1203	1203	CSC103
4454	payment	1202	1201	CSC105

5 rows returned in 0.00 seconds    CSV Export

Activate Windows  
Go to Settings to activate!  
Application

## Query Writing:

### SQL

#### Single-row function

1. Find the room number and section.

Ans:

Select concat (room\_number, sections)

From Class;

The screenshot shows a MySQL command-line interface. The SQL tab contains the following code:

```
Select concat (room_number,sections)From Class;
```

The Results tab displays the following table:

CONCAT(ROOM_NUMBER,SECTIONS)
106C
106C
105A
106B
106B

5 rows returned in 0.06 seconds    CSV Export

## Course Registration Management System

2. What will be the length of the name?

Ans:

Select name length(name)

From DetailsCourse;

Home > SQL > SQL Commands

Autocommit Display 10 ▾

```
Select length(COURSE_D) from DetailsCourse;
```

Results Explain Describe Saved SQL History

LENGTH(COURSE_D)
6
6
6
6
6

5 rows returned in 0.00 seconds [CSV Export](#)

3. Print the assigned courses.

Ans:

Select Assigned\_course,instr (Assigned\_course, 'C')

From stdsid;

Home > SQL > SQL Commands

Autocommit Display 10 ▾

```
Select Assigned_course,instr (Assigned_course, 'C')
From stdsid;
```

Results Explain Describe Saved SQL History

ASSIGNED COURSE	INSTR(ASSIGNED COURSE,'C')
Computer Science	1
OOP2	0
Compilator	1
Database	0
Computer network	1

5 rows returned in 0.01 seconds [CSV Export](#)

# Course Registration Management System

## Group function

1. Find the max due amount of students.

Ans:

```
select max(Student_due)
```

```
from Transaction;
```

The screenshot shows a MySQL command-line interface. The SQL command entered is: `select max(Student_due) from Transaction;`. The results section displays a single row with the value 1500 under the heading MAX(STUDENT\_DUE). Below the results, it says "1 rows returned in 0.06 seconds".

MAX(STUDENT_DUE)
1500

1 rows returned in 0.06 seconds CSV Export

2. Print the minimum student pay.

Ans:

```
select min(Student_pay)
```

```
from Transaction;
```

The screenshot shows a MySQL command-line interface. The SQL command entered is: `select min(Student_pay) from Transaction;`. The results section displays a single row with the value 1000 under the heading MIN(STUDENT\_PAY). Below the results, it says "1 rows returned in 0.00 seconds".

MIN(STUDENT_PAY)
1000

1 rows returned in 0.00 seconds CSV Export

## Course Registration Management System

3. How many students have been assigned by faculty?

Ans:

```
select max(Assigned_students)
from Faculty
group by Faculty_id;
```

The screenshot shows a MySQL command-line interface. The SQL command entered is:

```
select max(Assigned_students)
from Faculty
group by Faculty_id;
```

The results table shows the maximum number of assigned students for each faculty, with 5 rows returned in 0.02 seconds. The results are:

MAX(ASSIGNED_STUDENTS)
30
20
15
25
18

CSV Export

## Subquery

1. Print the names of students whose details match those of students whose names start with 'Su'.

Ans:

Select the name

From Student

Where details= (select details

From Student

Where Name Like 'Su%');

## Course Registration Management System

Home > SQL > **SQL Commands**

Autocommit Display 10 ▾

```
select name  
From Student  
Where details=(select details  
From Student  
Where Name Like 'Su%');
```

**Results Explain Describe Saved SQL History**

NAME
Noboni
Succho
Sakib

3 rows returned in 0.00 seconds [CSV Export](#)

2. Print the names and times of faculty members who have the same time as those teaching the 'COMPUTER SCIENCE' course.

Ans:

Select name,time  
From Faculty  
Where time=(select time  
From faculty  
Where Course\_name='COMPUTER SCIENCE');

Home > SQL > **SQL Commands**

Autocommit Display 10 ▾

```
Select time,name  
From Faculty  
Where time=(select time  
From faculty  
Where Courses= 'Computer Science')
```

**Results Explain Describe Saved SQL**

TIME	NAME
13:00	Jane

1 rows returned in 0.01 seconds [CSV](#)

## Course Registration Management System

3. Print the names and number of assigned students for faculty members who have the minimum number of assigned students among all the faculty members.

Ans:

Select name,assigned\_students

From faculty

Where assigned\_students=(select min(assigned\_students)

From Faculty)

The screenshot shows a MySQL query interface. The query entered is:

```
Autocommit: Display 10
Select name,assigned_students
From faculty
Where assigned_students=(select min(assigned_students)
From Faculty);
```

The results section shows a table with one row:

NAME	ASSIGNED_STUDENTS
Sarah	15

1 rows returned in 0.00 seconds [CSV Export](#)

## Joining

1. Find student details and department.

Ans:

```
SELECT Name,Details,Department_ID
FROM Stdsid t
JOIN Student s
ON t.S_INFO = s.S_INFO;
```

# Course Registration Management System

Home > SQL > SQL Commands

Autocommit Display 10

```
SELECT Name,Details,Department_ID
FROM StdSID t
JOIN Student s
ON t.S_INFO = s.S_INFO;
```

Results Explain Describe Saved SQL History

NAME	DETAILS	DEPARTMENT_ID
Succho	CSE	104
Fahim	SE	103
Noboni	CSE	102
Ashesh	SE	105
Sakib	CSE	101

5 rows returned in 0.02 seconds [CSV Export](#)

2. Get the course room number and faculty.

Ans:

```
SELECT Course_id.Course_d, Course_id.Class_info, Dept.Assigned_faculty
FROM Course_id
JOIN Dept
ON Course_id.Course_d = Dept.Course_d;
```

Home > SQL > SQL Commands

Autocommit Display 10

```
SELECT Course_id.Course_d, Course_id.Class_info, Dept.Assigned_faculty
FROM Course_id
JOIN Dept
ON Course_id.Course_d = Dept.Course_d;
```

Results Explain Describe Saved SQL History

COURSE_D	CLASS_INFO	ASSIGNED_FACULTY
CSC101	Class101	FST
CSC103	Class102	BBA
CSC102	Class100	FST
CSC104	Class103	ENG
CSC105	Class104	ART

5 rows returned in 0.02 seconds [CSV Export](#)

3. Find all information for all courses.

Ans:

```
SELECT *
FROM dept d
```

# Course Registration Management System

```
JOIN Stdsid s  
ON d.COURSE_D = s.COURSE_INFO  
JOIN course_id c  
ON c.course_id = s.course_id  
JOIN Student st  
ON st.S_INFO = s.S_INFO;
```

The screenshot shows the Oracle Database Express Edition interface. The SQL command window contains the following query:

```
SELECT *  
FROM dept d  
JOIN Stdsid s  
ON d.COURSE_D = s.COURSE_INFO  
JOIN course_id c  
ON c.course_id = s.course_id  
JOIN Student st  
ON st.S_INFO = s.S_INFO;
```

The results section displays the following data:

DEPARTMENT_ID	ASSIGNED_FACULTY	ASSIGNED_COURSE	S_INFO	COURSE_ID	CLASS_INFO	FACULTY_ID	COURSE_D	STUDENTS_ID	ASSIGNED_COURSE	CREDIT_PER_COURSE	TOTAL_CRE
103	FST	Microprocessor	0001	1201	Class100	1001	CSC102	1203	OOP2	3	18
105	ART	Theme	0002	1205	Class101	1005	CSC105	1204	Compiler	3	12
101	FST	Biology	0005	1203	Class103	1002	CSC101	1206	Database	3	9
102	BBA	Communications	0003	1202	Class104	1003	CSC103	1201	Computer Science	3	15
104	ENG	English	0004	1204	Class103	1003	CSC104	1207	Computer network	3	12

5 rows returned in 0.02 seconds [CSV Export](#)

## View

1. Create a view to get student names.

Ans:

```
CREATE VIEW student_names AS
```

```
SELECT NAME FROM Student;
```

```
SELECT * FROM student_names;
```

# Course Registration Management System

The screenshot shows the MySQL Workbench interface. In the top navigation bar, it says "Home > SQL > SQL Commands". Below that is a toolbar with "Autocommit" checked and a "Display" dropdown set to 10. The main area contains the following SQL code:

```
CREATE VIEW student_names AS
SELECT NAME FROM Student;

SELECT * FROM student_names;
```

Below the code, there are tabs for "Results", "Explain", "Describe", "Saved SQL", and "History". The "Results" tab is selected. The results table has a single column "NAME" with the following data:

NAME
Noboni
Succho
Fahim
Sakib
Ashesh

At the bottom, it says "5 rows returned in 0.00 seconds" and has a "CSV Export" link.

2. Create a view to get Faculty names.

Ans:

```
CREATE VIEW faculty_names_view AS
```

```
SELECT Name FROM Faculty;
```

```
SELECT * FROM faculty_names_view;
```

The screenshot shows the MySQL Workbench interface. In the top navigation bar, it says "Home > SQL > SQL Commands". Below that is a toolbar with "Autocommit" checked and a "Display" dropdown set to 10. The main area contains the following SQL code:

```
CREATE VIEW faculty_names_view AS
SELECT Name FROM Faculty;

SELECT * FROM faculty_names_view;
```

Below the code, there are tabs for "Results", "Explain", "Describe", "Saved SQL", and "History". The "Results" tab is selected. The results table has a single column "NAME" with the following data:

NAME
Jane
Michael
Sarah
John
David

At the bottom, it says "5 rows returned in 0.00 seconds" and has a "CSV Export" link.

3. Create a view to calculate the total sections in a course.

Ans:

```
CREATE VIEW total_sections AS
```

```
SELECT Course_info, COUNT(*) as num_sections
```

# Course Registration Management System

```
FROM Course  
GROUP BY Course_info;
```

```
select * from course;
```

Home > SQL > SQL Commands

Autocommit Display 10 ▾

```
CREATE VIEW total_sections AS
SELECT Course_info, COUNT(*) as num_sections
FROM Course
GROUP BY Course_info;

select * from course;
```

Results Explain Describe Saved SQL History

COURSE_INFO	NUM_SECTIONS
CSC101	1
CSC102	1
CSC103	1
CSC104	1
CSC105	1

5 rows returned in 0.01 seconds [CSV Export](#)

## Synonym

1. Create a synonym for DetailsCourse.

Ans:

```
CREATE SYNONYM Details
```

```
FOR DetailsCourse;
```

# Course Registration Management System

The screenshot shows the Oracle SQL Developer interface. The 'Worksheet' tab is active, displaying the SQL query: 'select \* from details;'. Below it, the 'Query Result' tab shows the execution results:

COURSE_D	CREDIT_PER_COURSE	COURSE_CREDIT_FEE
1 CSC101	3	1500
2 CSC102	3	1500
3 CSC103	3	1500
4 CSC104	3	1500
5 CSC105	3	1500

All Rows Fetched: 5 in 0.012 seconds.

2. Create a synonym for Stdsid.

Ans:

CREATE SYNONYM Stdsid\_syn

FOR Stdsid;

The screenshot shows the Oracle SQL Developer interface. The 'Worksheet' tab is active, displaying the SQL query: 'select \* from Stdsid\_syn;'. Below it, the 'Query Result' tab shows the execution results:

STUDENTS_ID	ASSIGNED_COURSE	CREDIT_PER_COURSE	TOTAL_CREDIT_FEE	COURSE_INFO	DEPARTMENT_ID	S_INFO	COURSE_ID	CLASS_INFO
1	1200 Computer Science	3	15	CSC103	104 0002	1204 Class103		
2	1201 OOP2	3	18	CSC102	103 0003	1201 Class100		
3	1202 Compiler	3	12	CSC105	102 0001	1202 Class104		
4	1203 Database	3	9	CSC101	105 0005	1203 Class101		
5	1204 Computer network	3	12	CSC104	101 0004	1200 Class102		

All Rows Fetched: 5 in 0.015 seconds.

3. Create a synonym for Dept.

Ans:

CREATE SYNONYM Syn\_Dept

FOR Dept;

# Course Registration Management System

The screenshot shows the Oracle SQL Developer interface. The top tab bar has 'Worksheet' and 'Query Builder' tabs, with 'Worksheet' selected. Below the tabs is a code editor window containing the SQL query:

```
1 select * from syn_dept;
```

Below the code editor is a 'Query Result' window. The title bar says 'Query Result x' and 'SQL | All Rows Fetched: 5 in 0.004 seconds'. The result grid has the following columns: DEPARTMENT\_ID, ASSIGNED\_FACULTY, ASSIGNED\_COURSE, S\_INFO, COURSE\_ID, CLASS\_INFO, FACULTY\_ID, and COURSE\_D. The data is as follows:

DEPARTMENT_ID	ASSIGNED_FACULTY	ASSIGNED_COURSE	S_INFO	COURSE_ID	CLASS_INFO	FACULTY_ID	COURSE_D
1	101 FST	Biology	0005	1203	Class103	1002	CSC101
2	102 BBA	Communications	0003	1202	Class104	1003	CSC103
3	103 FST	Microprocessor	0001	1201	Class100	1001	CSC102
4	104 ENG	English	0004	1204	Class103	1003	CSC104
5	105 ART	Theme	0002	1200	Class101	1005	CSC105

## PL/SQL

### Function

1. Write a function to calculate the total credit fee.

Ans:

```
CREATE OR REPLACE FUNCTION calculate_credit_fee(p_credit_per_course IN NUMBER,
p_num_courses IN NUMBER)

RETURN NUMBER IS

v_total_fee NUMBER;

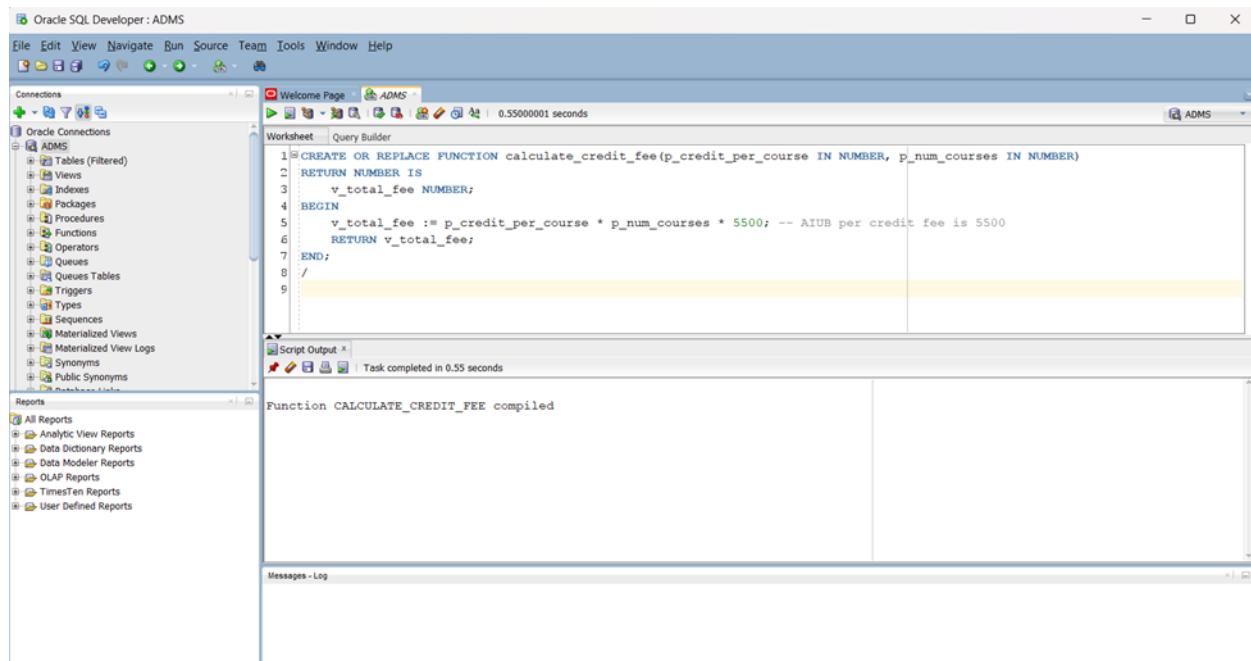
BEGIN

v_total_fee := p_credit_per_course * p_num_courses * 5500; -- AIUB per credit fee is 5500

RETURN v_total_fee;

END;
```

# Course Registration Management System

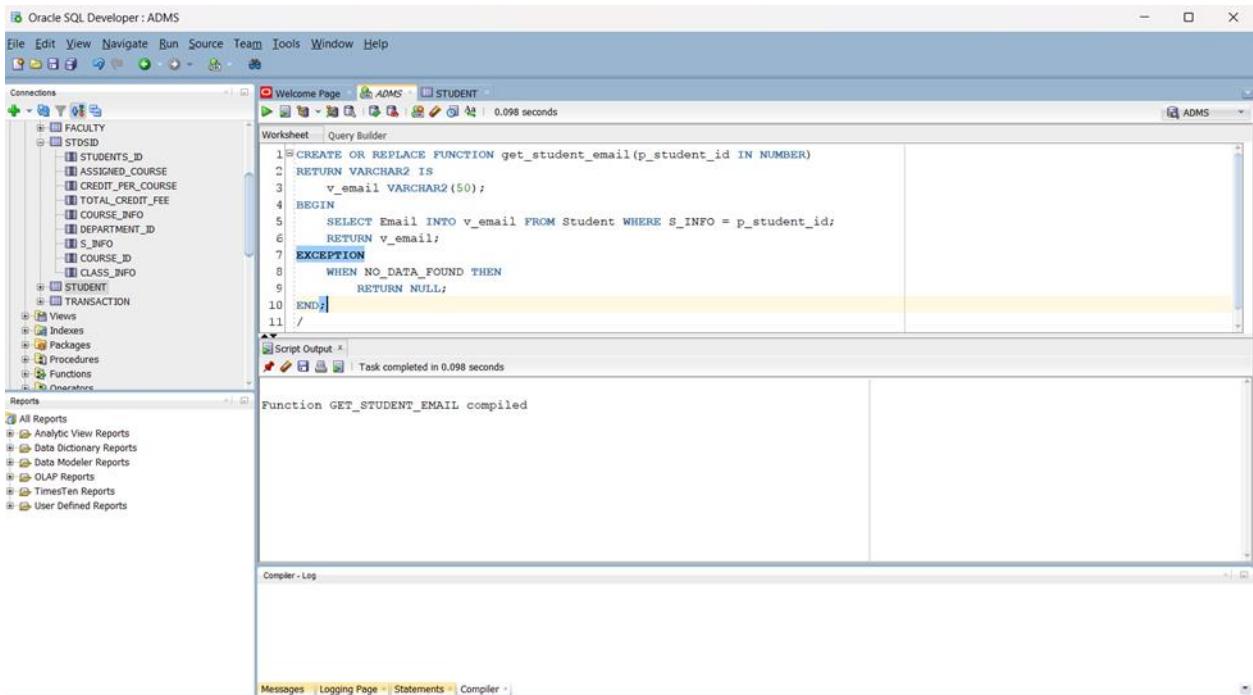


2. Write a function to get student email.

Ans:

```
CREATE OR REPLACE FUNCTION get_student_email(p_student_id IN NUMBER)
RETURN VARCHAR2 IS
    v_email VARCHAR2(50);
BEGIN
    SELECT Email INTO v_email FROM Student WHERE S_INFO = p_student_id;
    RETURN v_email;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN NULL;
END;
/
```

# Course Registration Management System

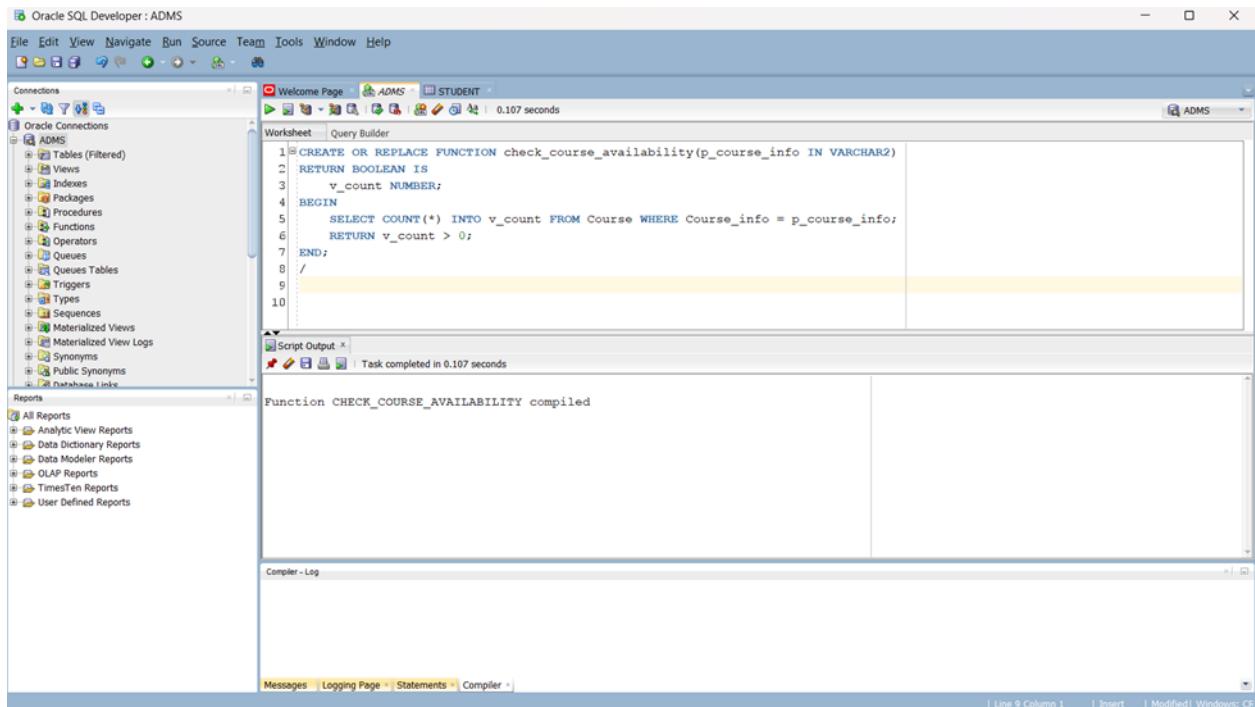


3. Write a function to check course availability.

Ans:

```
CREATE OR REPLACE FUNCTION check_course_availability(p_course_info IN VARCHAR2)
RETURN BOOLEAN IS
  v_count NUMBER;
BEGIN
  SELECT COUNT(*) INTO v_count FROM Course WHERE Course_info = p_course_info;
  RETURN v_count > 0;
END;
/
```

# Course Registration Management System



## Procedure

1. Write a procedure for updating the course.

Ans:

```
create or replace PROCEDURE update_course(
```

```
  p_course_info IN VARCHAR2,
  p_new_course_time IN VARCHAR2,
  p_new_section IN VARCHAR2
)
```

IS

BEGIN

```
  UPDATE Course
```

```
  SET Course_time = p_new_course_time,
```

```
  Section = p_new_section
```

```
  WHERE Course_info = p_course_info;
```

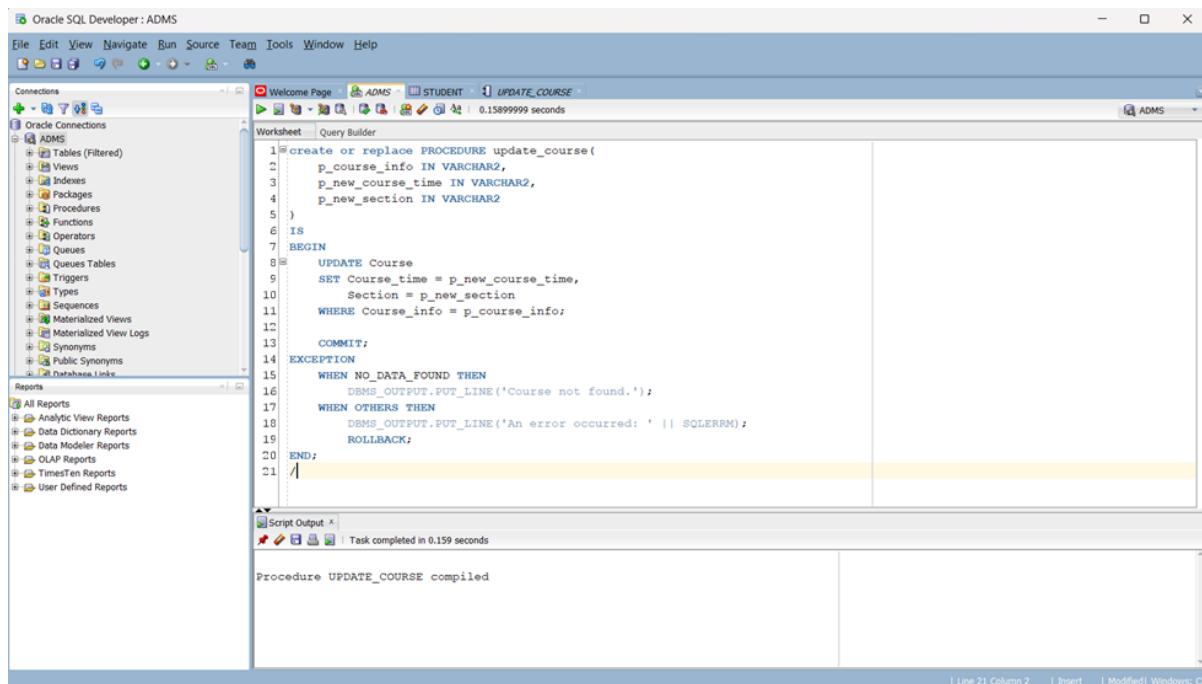
```
  COMMIT;
```

```
EXCEPTION
```

# Course Registration Management System

```
WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Course not found.');
WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
    ROLLBACK;
END;
/

```



2. Write a procedure to update the student's password.

Ans:

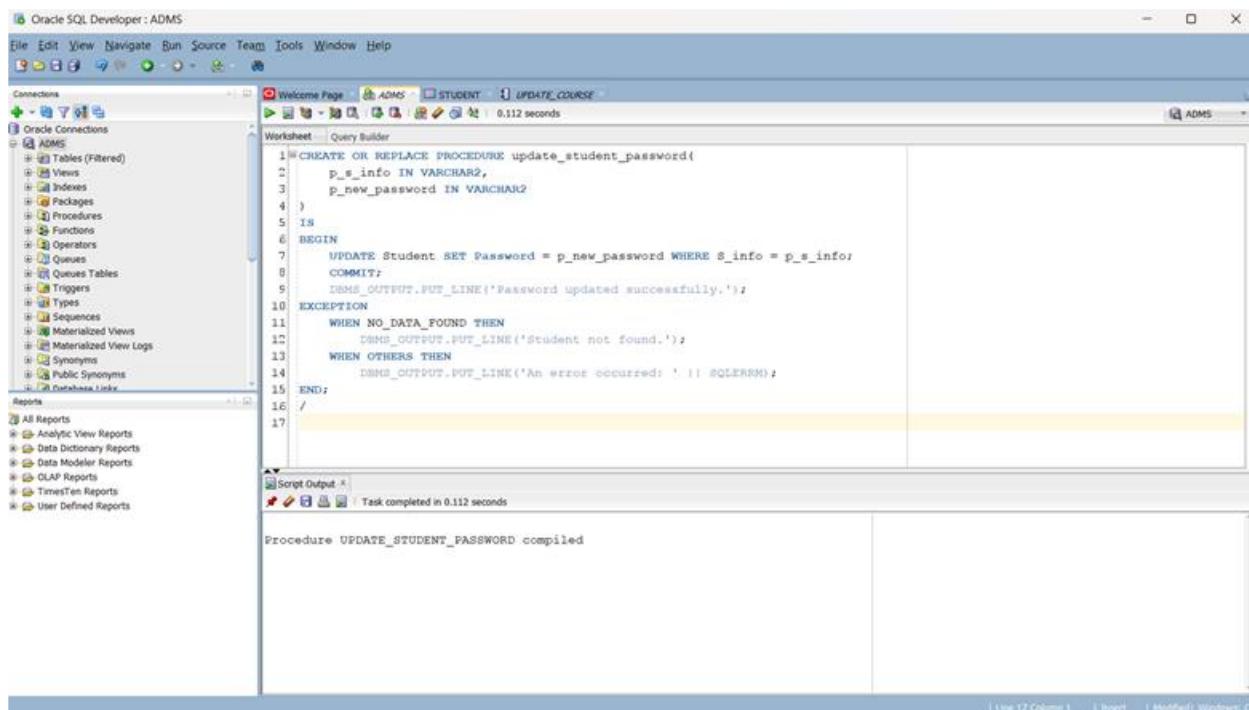
```
CREATE OR REPLACE PROCEDURE update_student_password(
    p_s_info IN VARCHAR2,
    p_new_password IN VARCHAR2
)
IS
BEGIN
    UPDATE Student SET Password = p_new_password WHERE S_info = p_s_info;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Password updated successfully.');

```

# Course Registration Management System

EXCEPTION

```
WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Student not found.');
WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END;
/
```



3. Write a procedure to assign faculty to the course.

Ans:

```
CREATE OR REPLACE PROCEDURE assign_faculty_to_course(p_faculty_id IN NUMBER,
p_course_info IN VARCHAR2)
```

IS

BEGIN

```
    UPDATE Faculty SET Courses = p_course_info WHERE Faculty_id = p_faculty_id;
    COMMIT;
```

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

# Course Registration Management System

```
RAISE;  
END;  
/  
;
```

The screenshot shows the Oracle SQL Developer interface. The left sidebar displays the database schema for the 'ADMS' connection, including tables, views, packages, and procedures. The central workspace contains a 'Worksheet' tab with the following PL/SQL code:

```
1 CREATE OR REPLACE PROCEDURE assign_faculty_to_course(p_faculty_id IN NUMBER, p_course_info IN VARCHAR2)
2 IS
3 BEGIN
4     UPDATE Faculty SET Courses = p_course_info WHERE Faculty_id = p_faculty_id;
5     COMMIT;
6     EXCEPTION
7     WHEN OTHERS THEN
8         ROLLBACK;
9         RAISE;
10 END;
11 /
12
```

The code is highlighted in yellow, indicating it has been selected or is being edited. Below the worksheet, the 'Script Output' pane shows the message: "Procedure ASSIGN\_FACULTY\_TO\_COURSE compiled". The status bar at the bottom right indicates the task completed in 0.076 seconds.

## Record

1. Write a record to store faculty information.

Ans:

```
DECLARE
```

```
    TYPE FacultyRecord IS RECORD (
        faculty_id Faculty.Faculty_id%TYPE,
        faculty_name Faculty.Name%TYPE,
        faculty_courses Faculty.Courses%TYPE
    );
```

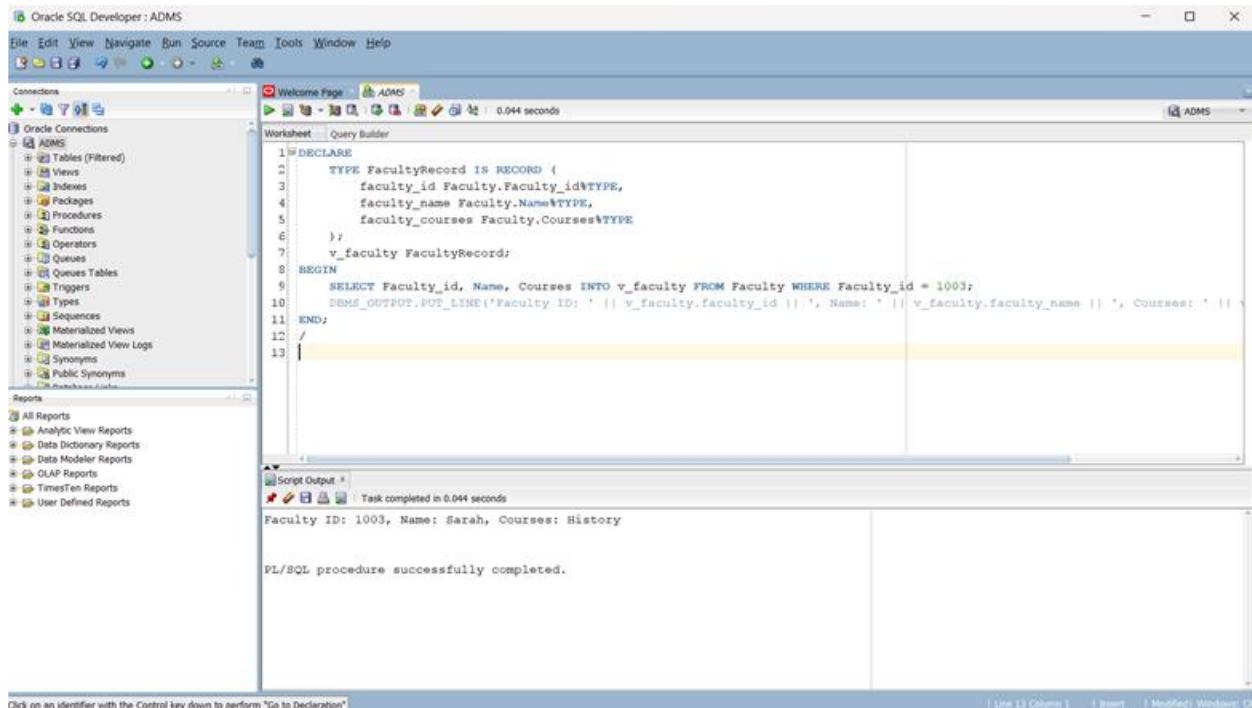
```
    v_faculty FacultyRecord;
```

```
BEGIN
```

```
    SELECT Faculty_id, Name, Courses INTO v_faculty FROM Faculty WHERE Faculty_id = 1003;
```

# Course Registration Management System

```
DBMS_OUTPUT.PUT_LINE('Faculty ID: ' || v_faculty.faculty_id || ', Name: ' ||  
v_faculty.faculty_name || ', Courses: ' || v_faculty.faculty_courses);  
END;  
/
```



2. Write a record to store course information.

Ans:

DECLARE

```
TYPE CourseRecord IS RECORD (  
  course_code Course.Course_info%TYPE,  
  course_time Course.Course_time%TYPE,  
  course_section Course.Section%TYPE  
);  
v_course CourseRecord;
```

BEGIN

```
  SELECT Course_info, Course_time, Section INTO v_course FROM Course WHERE Course_info =  
'CSC101';
```

```
  DBMS_OUTPUT.PUT_LINE('Course Code: ' || v_course.course_code || ', Time: ' ||  
v_course.course_time || ', Section: ' || v_course.course_section);
```

# Course Registration Management System

END;

/

The screenshot shows the Oracle SQL Developer interface. The 'Connections' sidebar shows a connection named 'ADMS'. The 'Worksheet' tab displays a PL/SQL block:

```
1 DECLARE
2   TYPE CourseRecord IS RECORD (
3     course_code Course.Course_info%TYPE,
4     course_time Course.Course_time%TYPE,
5     course_section Course.Section%TYPE
6   );
7   v_course CourseRecord;
8 BEGIN
9   SELECT Course_info, Course_time, Section INTO v_course FROM Course WHERE Course_info = 'CSC101';
10  DBMS_OUTPUT.PUT_LINE('Course Code: ' || v_course.course_code || ', Time: ' || v_course.course_time || ', Section: ' || v_
11 END;
12 /
13
14
```

The 'Script Output' window below the worksheet shows the results of the execution:

```
Task completed in 0.051 seconds
Course Code: CSC101, Time: 10:30-12:30, Section: A
PL/SQL procedure successfully completed.
```

3. Write a record to store student information.

Ans:

DECLARE

```
TYPE StudentRecord IS RECORD (
  student_id Student.s_info%TYPE,
  student_name Student.Name%TYPE,
  student_email Student.Email%TYPE
);
```

```
v_student StudentRecord;
```

BEGIN

```
SELECT s_info, Name, Email INTO v_student FROM Student WHERE s_info = 0001;
DBMS_OUTPUT.PUT_LINE('Student ID: ' || v_student.student_id || ', Name: ' ||
v_student.student_name || ', Email: ' || v_student.student_email);
```

END;

/

# Course Registration Management System

The screenshot shows the Oracle SQL Developer interface. The left sidebar displays the 'Connections' tree, which includes an 'ADMS' connection node. The main workspace contains a 'Worksheet' tab with a 'Query Builder' section. The code in the worksheet is:

```
1 DECLARE
2   TYPE StudentRecord IS RECORD (
3     student_id Student.S_info%TYPE,
4     student_name Student.Name%TYPE,
5     student_email Student.Email%TYPE
6   );
7   v_student StudentRecord;
8 BEGIN
9   SELECT s_info, Name, Email INTO v_student FROM Student WHERE s_info = 0001;
10  DBMS_OUTPUT.PUT_LINE('Student ID: ' || v_student.student_id || ', Name: ' || v_
11 END;
12 /
13
```

The 'Script Output' pane at the bottom shows the results of the execution:

```
Student ID: 0001, Name: Noboni, Email: noboni@example.com
PL/SQL procedure successfully completed.
```

## Cursor

1. Write a cursor to list students in a course.

Ans:

DECLARE

CURSOR student\_cursor IS

```
SELECT S_info, Name FROM Student WHERE S_info IN (SELECT S_info FROM Stdsid WHERE
Course_info = 'CSC101');
```

```
    v_student_id Student.S_info%TYPE;
```

```
    v_student_name Student.Name%TYPE;
```

BEGIN

```
    OPEN student_cursor;
```

```
LOOP
```

```
    FETCH student_cursor INTO v_student_id, v_student_name;
```

```
    EXIT WHEN student_cursor%NOTFOUND;
```

```
    DBMS_OUTPUT.PUT_LINE('Student ID: ' || v_student_id || ', Name: ' || v_student_name);
```

```
END LOOP;
```

# Course Registration Management System

```
CLOSE student_cursor;  
END;  
/  
;
```

The screenshot shows the Oracle SQL Developer interface. The left pane displays the 'Connections' tree, with 'ADMS' selected. The central 'Worksheet' pane contains the following PL/SQL code:

```
1 DECLARE  
2   CURSOR student_cursor IS  
3     SELECT S_info, Name FROM Student WHERE S_info IN (SELECT S_info FROM Stdsid WHERE Course_info = 'CSC101');  
4   v_student_id Student.S_info%TYPE;  
5   v_student_name Student.Name%TYPE;  
6 BEGIN  
7   OPEN student_cursor;  
8   LOOP  
9     FETCH student_cursor INTO v_student_id, v_student_name;  
10    EXIT WHEN student_cursor%NOTFOUND;  
11    DBMS_OUTPUT.PUT_LINE('Student ID: ' || v_student_id || ', Name: ' || v_student_name);  
12  END LOOP;  
13  CLOSE student_cursor;  
14 END;  
15 /  
16
```

The 'Script Output' pane at the bottom shows the results of the execution:

```
Student ID: 0005, Name: Ashesh  
PL/SQL procedure successfully completed.
```

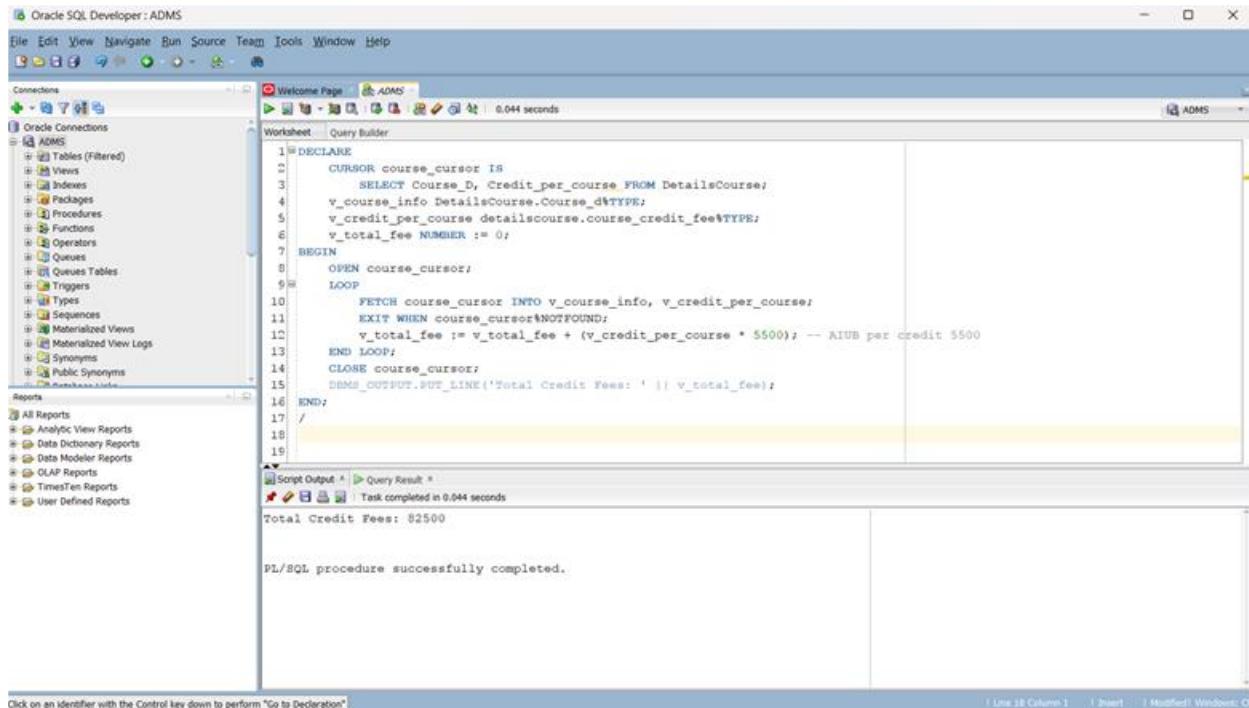
2. Write a cursor to calculate total credit fees.

Ans:

```
DECLARE  
CURSOR course_cursor IS  
SELECT Course_D, Credit_per_course FROM DetailsCourse;  
v_course_info DetailsCourse.Course_d%TYPE;  
v_credit_per_course detailscourse.course_credit_fee%TYPE;  
v_total_fee NUMBER := 0;  
BEGIN  
OPEN course_cursor;  
LOOP  
FETCH course_cursor INTO v_course_info, v_credit_per_course;  
EXIT WHEN course_cursor%NOTFOUND;  
v_total_fee := v_total_fee + (v_credit_per_course * 5500); -- AIUB per credit 5500
```

# Course Registration Management System

```
END LOOP;  
CLOSE course_cursor;  
DBMS_OUTPUT.PUT_LINE('Total Credit Fees: ' || v_total_fee);  
END;  
/  
END;
```



3. Write a cursor to update students' email domains.

Ans:

```
DECLARE  
CURSOR student_cursor IS  
SELECT s_info, Email FROM Student WHERE Email LIKE '%@example.com';  
    v_student_id student.s_info%TYPE;  
    v_student_email Student.Email%TYPE;  
BEGIN  
    OPEN student_cursor;  
    LOOP  
        FETCH student_cursor INTO v_student_id, v_student_email;  
        EXIT WHEN student_cursor%NOTFOUND;
```

# Course Registration Management System

```
UPDATE Student SET Email = REPLACE(v_student_email, '@example.com', '@aiub.edu') WHERE
s_info = v_student_id;
COMMIT;
END LOOP;
CLOSE student_cursor;
END;
/
```

The screenshot shows the Oracle SQL Developer interface. The 'Connections' sidebar is open, showing a connection named 'ADMS'. The 'Worksheet' tab is active, displaying a PL/SQL script in the 'Query Builder' pane:

```
1  DECLARE
2      CURSOR student_cursor IS
3          SELECT s_info, Email FROM Student WHERE Email LIKE '%@example.com';
4          v_student_id student.s_info%TYPE;
5          v_student_email Student.Email%TYPE;
6      BEGIN
7          OPEN student_cursor;
8          LOOP
9              FETCH student_cursor INTO v_student_id, v_student_email;
10             EXIT WHEN student_cursor%NOTFOUND;
11             UPDATE Student SET Email = REPLACE(v_student_email, '@example.com', '@aiub.edu') WHERE s_info = v_student_id;
12             COMMIT;
13         END LOOP;
14         CLOSE student_cursor;
15     END;
16 /
17
```

The 'Script Output' pane at the bottom shows the message: "PL/SQL procedure successfully completed." The status bar at the bottom right indicates "Task completed in 0.045 seconds".

S_INFO	DETAILS	NAME	USERNAME	PASSWORD	EMAIL
1 0001	CSE	Noboni	noboni123	pass1234	noboni@aiub.edu
2 0002	CSE	Succho	succho123	succho4321	succho@aiub.edu
3 0003	SE	Fahim	fahim99	abcxyz	fahim@aiub.edu
4 0004	CSE	Sakib	sakib456	pass456	sakib@aiub.edu

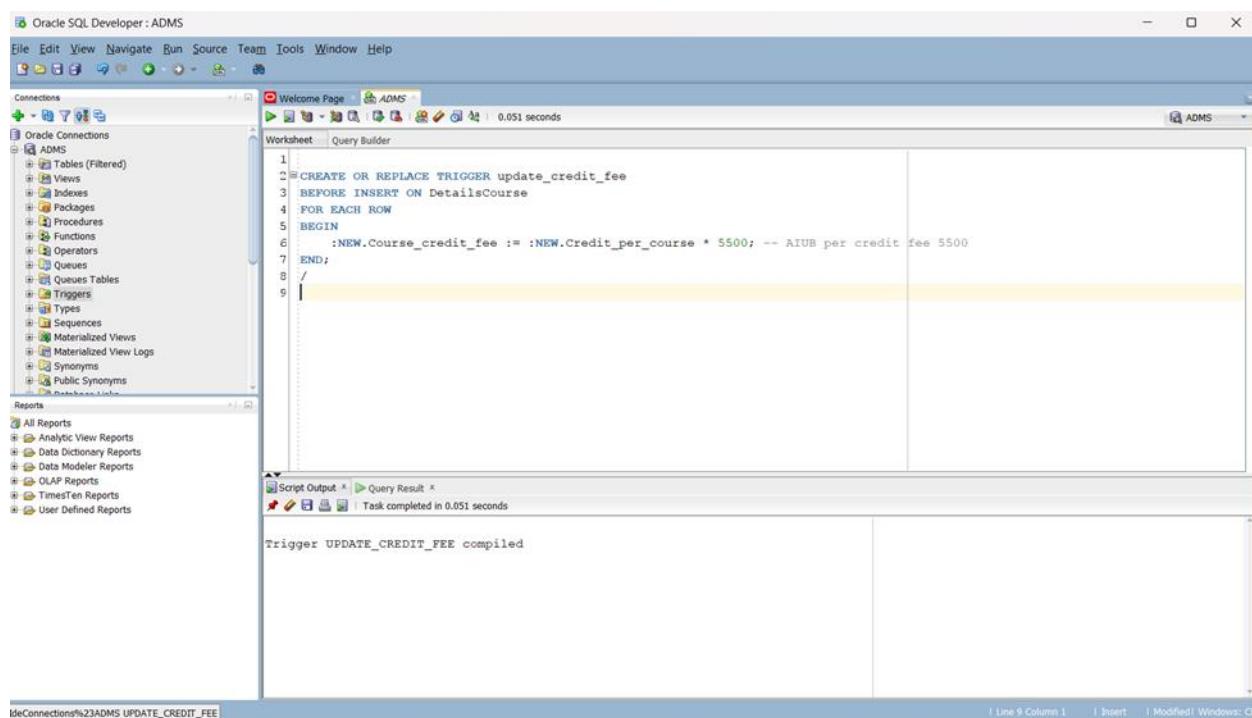
# Course Registration Management System

## Trigger

1. Write a trigger to update the course credit fee.

Ans:

```
CREATE OR REPLACE TRIGGER update_credit_fee
BEFORE INSERT ON DetailsCourse
FOR EACH ROW
BEGIN
:NEW.Course_credit_fee := :NEW.Credit_per_course * 5500; -- AIUB per credit fee 5500
END;
/
```



2. Write a trigger to enforce the maximum number of students in a class.

Ans:

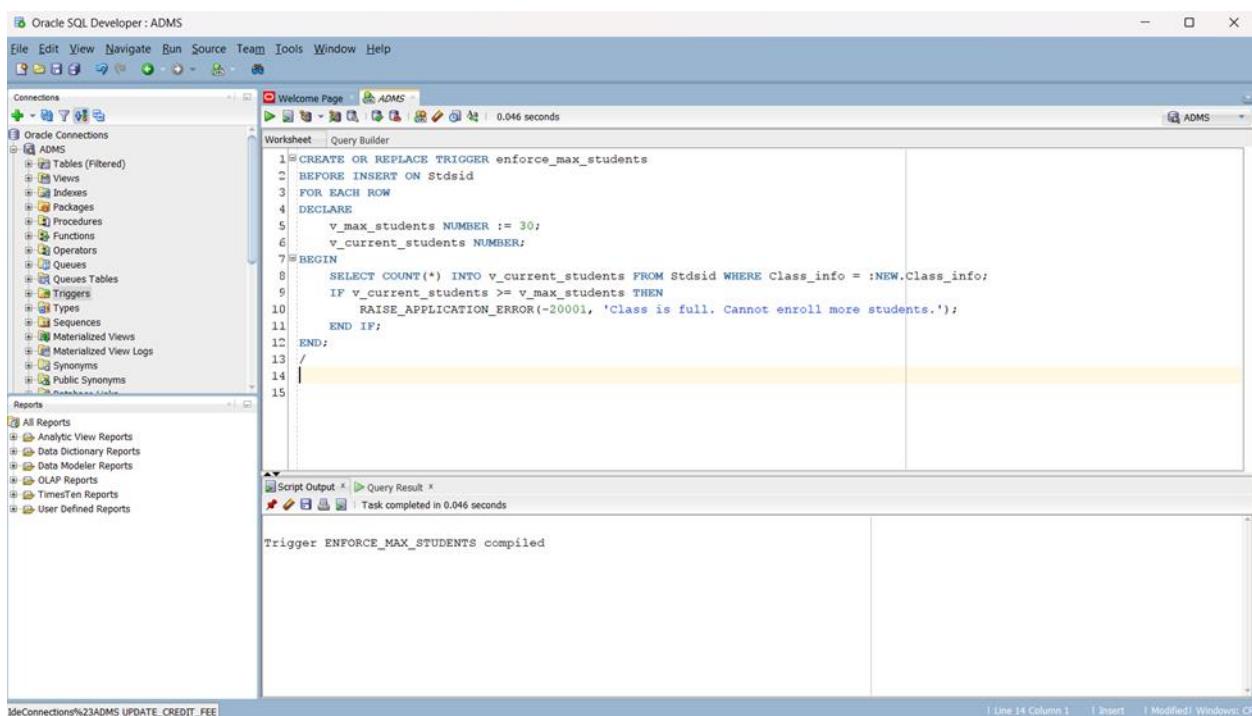
```
CREATE OR REPLACE TRIGGER enforce_max_students
BEFORE INSERT ON Stdsid
FOR EACH ROW
DECLARE
v_max_students NUMBER := 30;
```

# Course Registration Management System

```
v_current_students NUMBER;  
BEGIN  
    SELECT COUNT(*) INTO v_current_students FROM Stdsid WHERE Class_info =  
    :NEW.Class_info;  
    IF v_current_students >= v_max_students THEN  
        RAISE_APPLICATION_ERROR(-20001, 'Class is full. Cannot enroll more students.');
```

END IF;

```
END;  
/  
Oracle SQL Developer : ADMS
```



3. Write a trigger to change the faculty email domain.

Ans:

```
CREATE OR REPLACE TRIGGER change_faculty_email_domain
```

```
BEFORE UPDATE ON Faculty
```

```
FOR EACH ROW
```

```
BEGIN
```

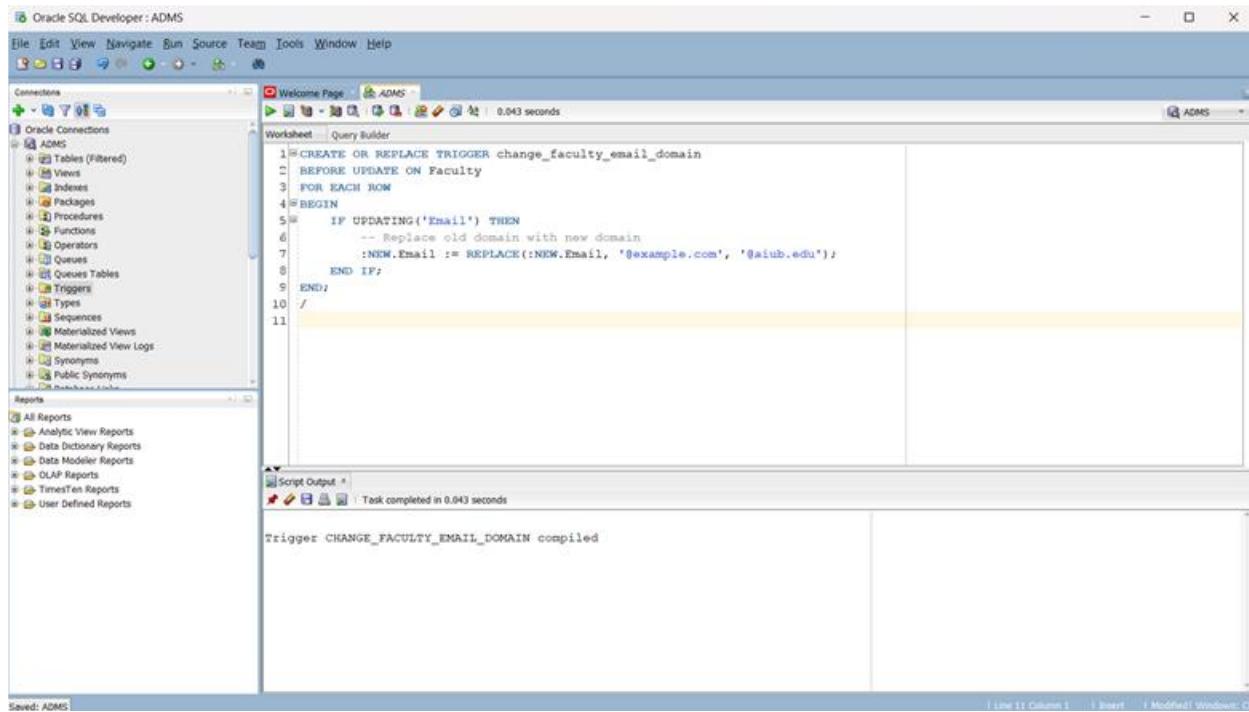
```
    IF UPDATING('Email') THEN
```

```
        -- Replace old domain with new domain
```

```
:NEW.Email := REPLACE(:NEW.Email, '@example.com', '@aiub.edu');
```

# Course Registration Management System

```
END IF;  
END;  
/
```



The screenshot shows the Oracle SQL Developer interface. The left sidebar displays the 'Connections' tree, which includes 'ADMS' under 'Oracle Connections'. The 'Reports' section is also visible. The main area is titled 'Worksheet - Query Builder' and contains the following PL/SQL code:

```
1 CREATE OR REPLACE TRIGGER change_faculty_email_domain  
2 BEFORE UPDATE ON Faculty  
3 FOR EACH ROW  
4 BEGIN  
5 IF UPDATING('Email') THEN  
6 -- Replace old domain with new domain  
7 :NEW.Email := REPLACE(:NEW.Email, '@example.com', '@aiub.edu');  
8 END IF;  
9 END;  
10 /
```

The bottom right corner of the worksheet shows the message: 'Trigger CHANGE\_FACULTY\_EMAIL\_DOMAIN compiled'.

## Package

1. Write a package to insert the course.

Ans:

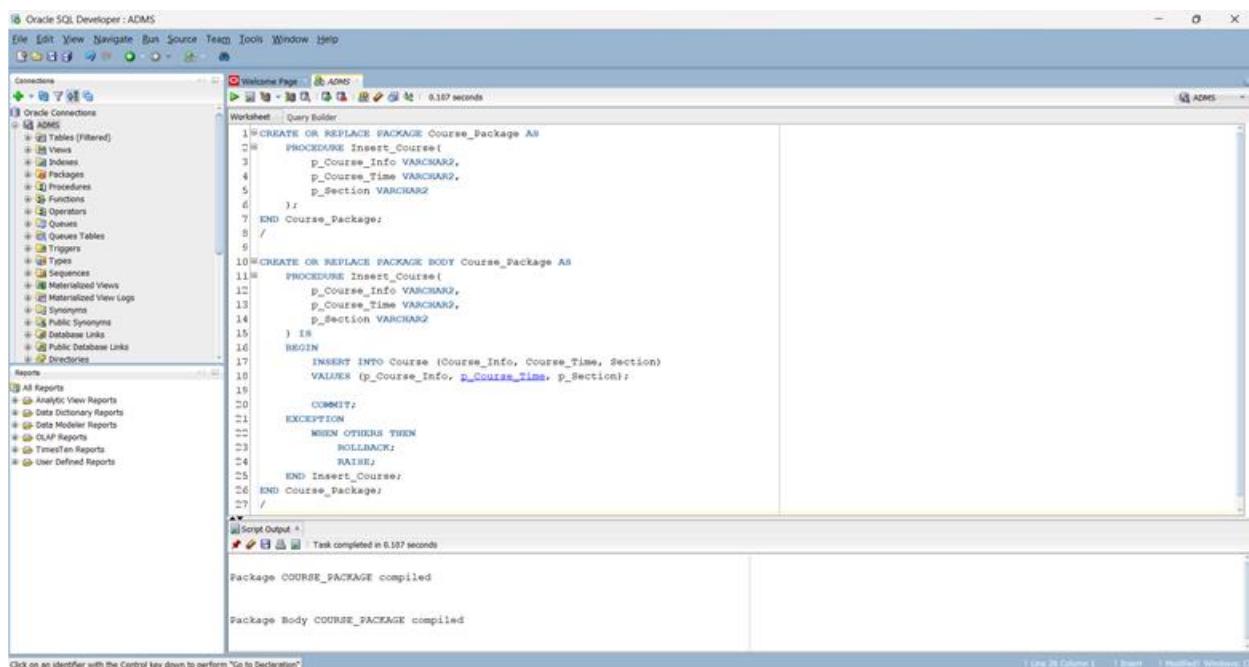
```
CREATE OR REPLACE PACKAGE Course_Package AS  
PROCEDURE Insert_Course(  
    p_Course_Info VARCHAR2,  
    p_Course_Time VARCHAR2,  
    p_Section VARCHAR2  
);  
END Course_Package;  
/
```

```
CREATE OR REPLACE PACKAGE BODY Course_Package AS
```

# Course Registration Management System

```
PROCEDURE Insert_Course(
    p_Course_Info VARCHAR2,
    p_Course_Time VARCHAR2,
    p_Section VARCHAR2
) IS
BEGIN
    INSERT INTO Course (Course_Info, Course_Time, Section)
    VALUES (p_Course_Info, p_Course_Time, p_Section);

    COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        RAISE;
END Insert_Course;
END Course_Package;
/
```



The screenshot shows the Oracle SQL Developer interface with the following details:

- Connections:** Oracle Connections - ADMSP.
- Worksheet:** Query Builder - Welcome Page, ADMSP. The code is displayed in the worksheet.
- Script Output:** Task completed in 0.187 seconds.
- Messages:** Package COURSE\_PACKAGE compiled and Package Body COURSE\_PACKAGE compiled.
- Status Bar:** Click on an identifier with the Control key down to perform "Go to Declaration". Line 28 Column 1, 2 Rows, 1 Modified, Windows.

```
1 CREATE OR REPLACE PACKAGE Course_Package AS
2     PROCEDURE Insert_Course(
3         p_Course_Info VARCHAR2,
4         p_Course_Time VARCHAR2,
5         p_Section VARCHAR2
6     );
7     END Course_Package;
8 /
9
10 CREATE OR REPLACE PACKAGE BODY Course_Package AS
11     PROCEDURE Insert_Course(
12         p_Course_Info VARCHAR2,
13         p_Course_Time VARCHAR2,
14         p_Section VARCHAR2
15     ) IS
16     BEGIN
17         INSERT INTO Course (Course_Info, Course_Time, Section)
18         VALUES (p_Course_Info, p_Course_Time, p_Section);
19
20         COMMIT;
21     EXCEPTION
22         WHEN OTHERS THEN
23             ROLLBACK;
24             RATHER;
25     END Insert_Course;
26 END Course_Package;
27 /
```

## Course Registration Management System

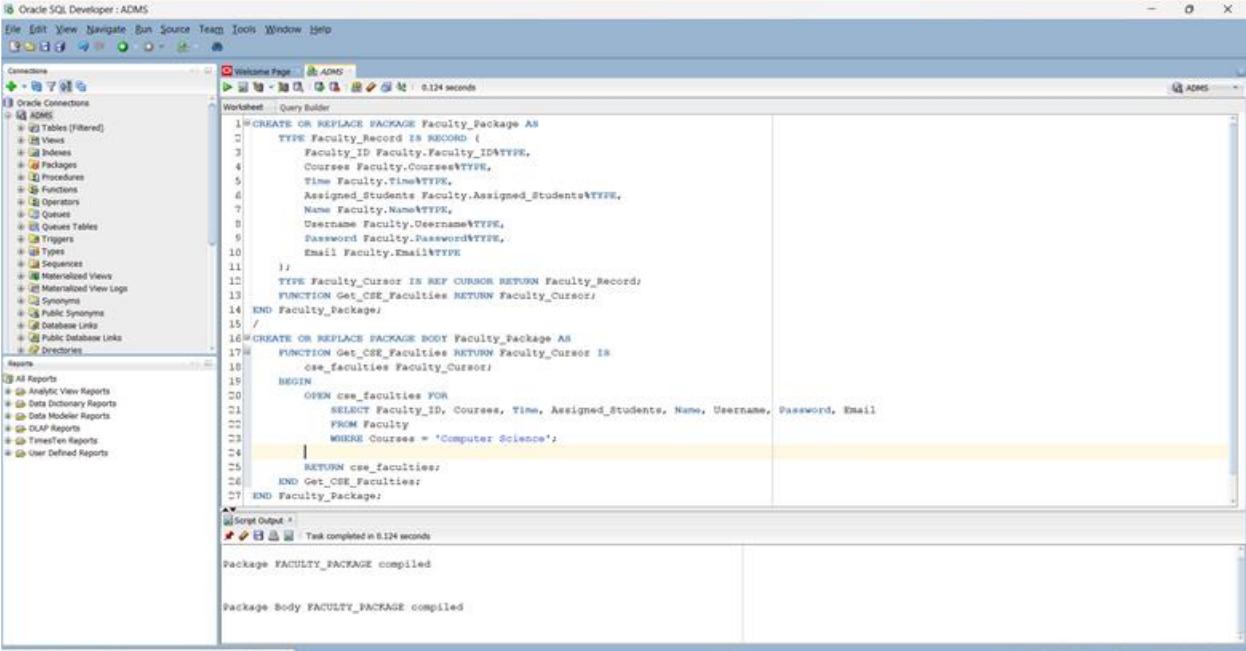
2. Write a package to retrieve only computer science faculties.

Ans:

```
CREATE OR REPLACE PACKAGE Faculty_Package AS
TYPE Faculty_Record IS RECORD (
    Faculty_ID Faculty.Faculty_ID%TYPE,
    Courses Faculty.Courses%TYPE,
    Time Faculty.Time%TYPE,
    Assigned_Students Faculty.Assigned_Students%TYPE,
    Name Faculty.Name%TYPE,
    Username Faculty.Username%TYPE,
    Password Faculty.Password%TYPE,
    Email Faculty.Email%TYPE
);
TYPE Faculty_Cursor IS REF CURSOR RETURN Faculty_Record;
FUNCTION Get_CSE_Faculties RETURN Faculty_Cursor;
END Faculty_Package;
/
CREATE OR REPLACE PACKAGE BODY Faculty_Package AS
FUNCTION Get_CSE_Faculties RETURN Faculty_Cursor IS
cse_faculties Faculty_Cursor;
BEGIN
    OPEN cse_faculties FOR
        SELECT Faculty_ID, Courses, Time, Assigned_Students, Name, Username, Password, Email
        FROM Faculty
        WHERE Courses = 'Computer Science';

    RETURN cse_faculties;
END Get_CSE_Faculties;
END Faculty_Package;
/
```

# Course Registration Management System



The screenshot shows the Oracle SQL Developer interface with the title bar "Oracle SQL Developer : ADMS". The left sidebar shows "Connections" with one entry "ADMS". The main area is titled "Worksheet - Query Builder". The code in the worksheet is:

```
1 CREATE OR REPLACE PACKAGE Faculty_Package AS
2     TYPE Faculty_Record IS RECORD (
3         Faculty_ID Faculty.Faculty_ID%TYPE,
4         Courses Faculty.Courses%TYPE,
5         Time Faculty.Time%TYPE,
6         Assigned_Students Faculty.Assigned_Students%TYPE,
7         Name Faculty.Name%TYPE,
8         Username Faculty.Username%TYPE,
9         Password Faculty.Password%TYPE,
10        Email Faculty.Email%TYPE
11    );
12    TYPE Faculty_Cursor IS REF CURSOR RETURN Faculty_Record;
13    FUNCTION Get_CSE_Faculties RETURN Faculty_Cursor;
14 END Faculty_Package;
15 /
16 CREATE OR REPLACE PACKAGE BODY Faculty_Package AS
17     FUNCTION Get_CSE_Faculties RETURN Faculty_Cursor IS
18        cse_faculties Faculty_Cursor;
19    BEGIN
20        OPEN cse_faculties FOR
21            SELECT Faculty_ID, Courses, Time, Assigned_Students, Name, Username, Password, Email
22            FROM Faculty
23            WHERE Courses = 'Computer Science';
24    END;
25    RETURN cse_faculties;
26    END Get_CSE_Faculties;
27 END Faculty_Package;
```

The status bar at the bottom right indicates "1 line 24 columns 8 2 rows 1 Modified Windows XP".

3. Write a package to retrieve the assigned course by student name.

Ans:

```
CREATE OR REPLACE PACKAGE Student_Course_Package AS
FUNCTION Get_Assigned_Course(
    p_Student_Name Student.Name%TYPE
) RETURN VARCHAR2;
END Student_Course_Package;
/
```

```
CREATE OR REPLACE PACKAGE BODY Student_Course_Package AS
FUNCTION Get_Assigned_Course(
    p_Student_Name Student.Name%TYPE
) RETURN VARCHAR2 IS
    v_Assigned_Course VARCHAR2(20);
BEGIN
    SELECT DISTINCT ci.Course_d INTO v_Assigned_Course
    FROM Stdsid s
    JOIN Course_id ci ON s.Course_id = ci.Course_id
```

# Course Registration Management System

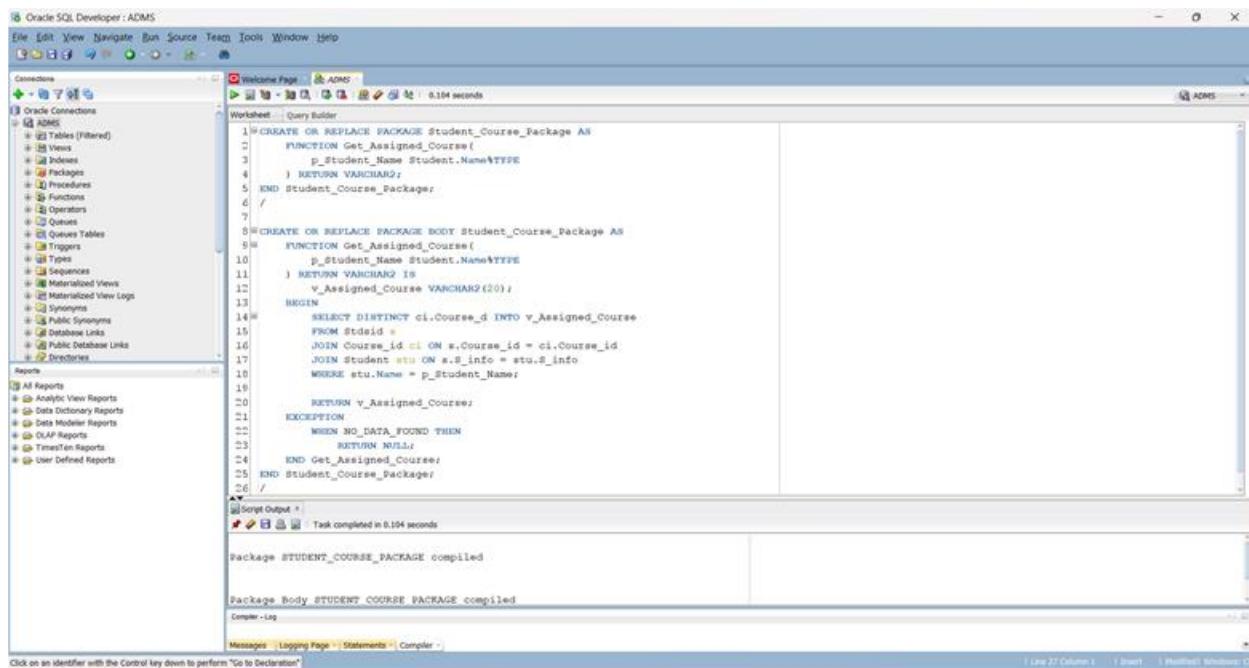
```
JOIN Student stu ON s.S_info = stu.S_info
WHERE stu.Name = p_Student_Name;

RETURN v_Assigned_Course;

EXCEPTION
WHEN NO_DATA_FOUND THEN
RETURN NULL;
END Get_Assigned_Course;

END Student_Course_Package;
```

/



The screenshot shows the Oracle SQL Developer interface with the following details:

- Connections:** A tree view showing various database objects like Tables, Indexes, Packages, Procedures, Functions, Operators, Queues, Views, Triggers, Sequences, Materialized Views, Materialized View Logs, Synonyms, Public Synonyms, Database Links, and Directories.
- Worksheet:** The main area where the PL/SQL code is written. The code is a package named "Student\_Course\_Package". It contains a function "Get\_Assigned\_Course" which joins the "Student" and "Course" tables to find a student's assigned course based on their name. It also handles the "NO\_DATA\_FOUND" exception by returning null if no course is found.
- Script Output:** A panel at the bottom showing the results of the compilation. It displays two messages:
  - "Package STUDENT\_COURSE\_PACKAGE compiled"
  - "Package Body STUDENT\_COURSE PACKAGE compiled"
- Status Bar:** At the bottom, it says "Click on an identifier with the Control key down to perform 'Go to Declaration'".
- Bottom Navigation:** Buttons for "Messages", "Logging Page", "Statements", and "Compiler".

# Course Registration Management System

## **Relational Algebra**

1. Display all the info whose Student\_Pay is greater than 1000.

Ans:  $\sigma_{\text{Student\_Pay} > 1000}(\text{Transaction})$

2. Display all the information from students whose details are in CSE.

Ans:  $\sigma_{\text{details} = \text{CSE}}(\text{Student})$

3. Display all the information from a student whose name starts with S.

Ans:  $\sigma_{\text{S\_name like 'S%'}}(\text{Student})$

4. Display all S-Transactions whose student pay is greater than 1000.

Ans:  $\prod_{\text{S-Transaction}} (\sigma_{\text{Student\_Pay} > 1000}(\text{Transaction}))$

5. Display all the room numbers and sections whose section is B from the class table.

Ans:  $\prod_{\text{room\_number, Section}}(\text{class})$

## **Conclusion:**

The planned course registration management system aims to offer educational institutions a streamlined and productive tool for overseeing student course registration. This system will enhance the registration process for students and administrators, minimize mistakes, and provide up-to-date details on course availability and enrolment status. It will be created using web-based technologies and hosted on a cloud-based server to ensure easy access and scalability.