



## AMERICAN INTERNATIONAL UNIVERSITY- BANGLADESH

### ADVANCE DATABASE MANAGEMENT SYSTEM

FALL 2024-25

#### Project Name

#### Course Registration Management System

Section: B

Group: 03

Submission Date: 03/ 02/ 2025

#### Supervised By

JUENA AHMED NOSHIN

Assistant Professor, Faculty, Department of Computer Science

#### Submitted By

ID	Name	Topic Name	Contribution
22-49415-3	S S Zobaer Ahmed	Backend construction, AI development and Integration, Documentation, Query Writing	25%
21-45834-3	Morshed Al-Jaber Bishal	Database Specialist, Design, Query Writing	25%
21-45837-3	SANVIRAJ AYNUL SIAM	Documentation, Design, Query writing	25%
21-45836-3	SHARUP PAUL	Query Writing, Relational Algebra	25%

## Table of Contents

<b>Project Updates</b> .....	3
Changes in Tables and ERD .....	3
Update on User Interface .....	4
Updated ERD .....	5
Updates to Query Writing .....	5
Joining With PL/SQL: .....	5
<b>Project Proposal</b> .....	7
Background of the problem .....	7
Solution to the problem.....	7
<b>Diagrams</b> .....	9
Activity Diagram: .....	9
Class Diagram: .....	9
Use Case Diagram: .....	10
<b>Schema Diagram</b> .....	10
<b>User Interface</b> .....	11
Interface Design .....	11
Admin Panel .....	11
Student Panel .....	12
Final term interface design code .....	13
<b>Database Connection</b> .....	13
Connection in C# .....	13
Connection in Python.....	14
<b>Query Writing</b> .....	14
Exception handling .....	14
Implicit Locking.....	15
Explicit Locking.....	16
<b>Relational Algebra</b> .....	17

# Project Updates

## Changes in Tables and ERD

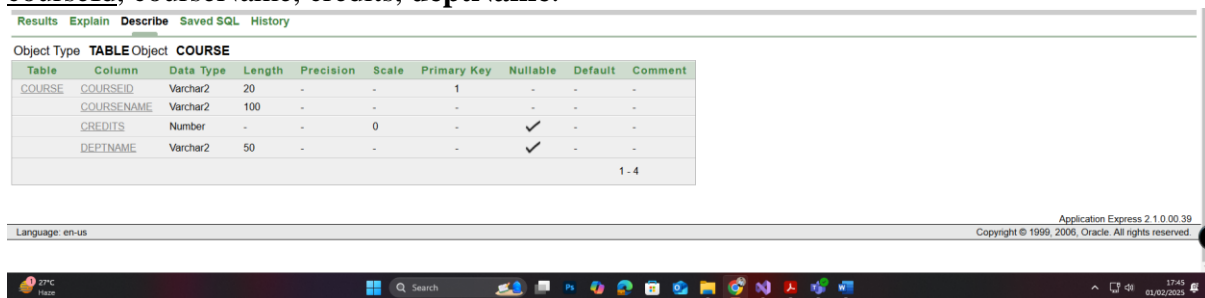
In the course table we remove an attribute or column named semesterOffered and also add a column named DeptName.

### Previous Course Table:

courseId, courseName, semesterOffered, credits.

### Updated Course Table:

courseId, courseName, credits, **deptName**.



The screenshot shows the Oracle SQL Developer interface with the 'Describe' tab selected for the 'COURSE' table. The table structure is as follows:

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
COURSE	COURSEID	Varchar2	20	-	-	1	-	-	-
	COURSENAME	Varchar2	100	-	-	-	-	-	-
	CREDITS	Number	-	-	0	-	✓	-	-
	DEPTNAME	Varchar2	50	-	-	-	✓	-	-

At the bottom right of the table, it says '1 - 4'. The status bar at the bottom indicates 'Language: en-us' and 'Application Express 2.1.0.00.39 Copyright © 1999, 2006, Oracle. All rights reserved.'

In ClassSchedule table we will add two new attributes named Capacity and Section

### Previous ClassSchedule Table:

classId, classDay, classTime, roomNo, semester, **courseId**.

### Updated ClassSchedule Table:

classId, classDay, classTime, roomNo, semester, **coursed**, capacity, Section.



The screenshot shows the Oracle SQL Developer interface with the 'Describe' tab selected for the 'CLASSSCHEDULE' table. The table structure is as follows:

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
CLASSSCHEDULE	CLASSID	Number	-	-	0	1	-	-	-
	CLASSDAY	Varchar2	20	-	-	-	✓	-	-
	CLASSTIME	Varchar2	30	-	-	-	✓	-	-
	ROOMNO	Varchar2	50	-	-	-	✓	-	-
	SEMESTER	Varchar2	20	-	-	-	✓	-	-
	COURSEID	Varchar2	20	-	-	-	✓	-	-
	CAPACITY	Number	-	-	-	-	✓	-	-
	SECTION	Varchar2	5	-	-	-	✓	-	-

At the bottom right of the table, it says '1 - 8'. The status bar at the bottom indicates 'Language: en-us' and 'Application Express 2.1.0.00.39 Copyright © 1999, 2006, Oracle. All rights reserved.'

We will add two new Entities or Tables

### 1. RegisteredCourse:

```
CREATE TABLE RegisteredCourse (  
  registrationId NUMBER PRIMARY KEY,  
  studentId NUMBER NOT NULL,  
  courseId VARCHAR2(20) NOT NULL,  
  semester VARCHAR2(50) NOT NULL,  
  year NUMBER(4) NOT NULL,
```

```

    enrollmentDate DATE NOT NULL,
    CONSTRAINT fk_student FOREIGN KEY (studentId) REFERENCES Student(studentId)
ON DELETE CASCADE,
    CONSTRAINT fk_course FOREIGN KEY (courseId) REFERENCES Courses(courseId)
ON DELETE CASCADE
);

```

Results Explain Describe Saved SQL History

Object Type TABLE Object REGISTEREDCOURSE

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
REGISTEREDCOURSE	REGISTRATIONID	Number	-	-	-	1	-	-	-
	STUDENTID	Number	-	-	-	-	-	-	-
	COURSEID	Varchar2	20	-	-	-	-	-	-
	SEMESTER	Varchar2	50	-	-	-	-	-	-
	YEAR	Number	-	4	0	-	-	-	-
	ENROLLMENTDATE	Date	7	-	-	-	-	-	-

1 - 6

Language: en-us

Application Express 2.1.0.00.39  
Copyright © 1999, 2006, Oracle. All rights reserved.

## 2. PreRequisiteCourse:

```

CREATE TABLE PreRequisiteCourse (
    courseId VARCHAR2(20) NOT NULL,
    preRequisiteCourseID VARCHAR2(20),
    PRIMARY KEY (courseId),
    FOREIGN KEY (courseId) REFERENCES Course(courseId) ON DELETE CASCADE,
    FOREIGN KEY (preRequisiteCourseID) REFERENCES Course(courseId) ON DELETE
CASCADE
);

```

Results Explain Describe Saved SQL History

Object Type TABLE Object PREREQUISITECOURSE

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
PREREQUISITECOURSE	COURSEID	Varchar2	20	-	-	1	-	-	-
	PREREQUISITECOURSEID	Varchar2	20	-	-	-	✓	-	-

1 - 2

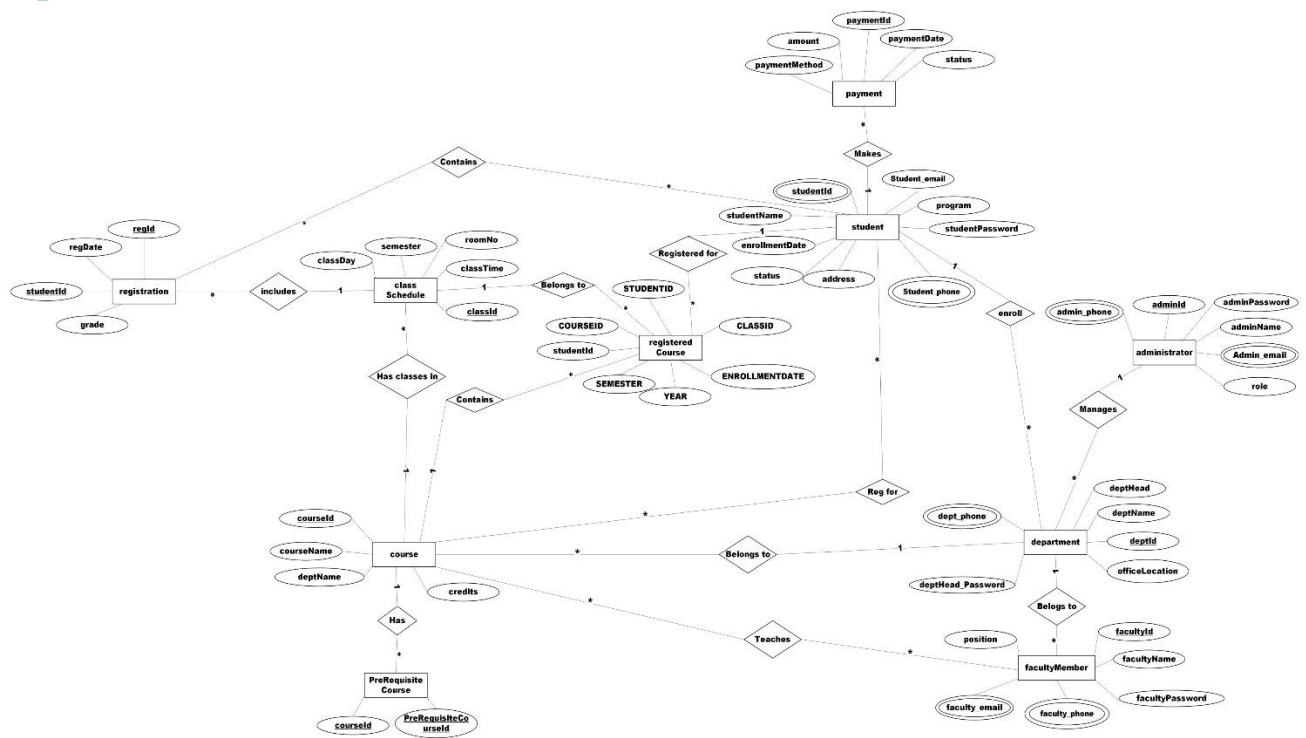
Language: en-us

Application Express 2.1.0.00.39  
Copyright © 1999, 2006, Oracle. All rights reserved.

## Update on User Interface

We have made slight improvements to our User Interface! Dive into the [User Interface](#) section to explore the fresh design and improved user experience.

## Updated ERD



Click to view : [ER Diagram CRMS](#)

## Updates to Query Writing

### Joining With PL/SQL:

1. Write a query to list students and their department names using PL/SQL

#### Query:

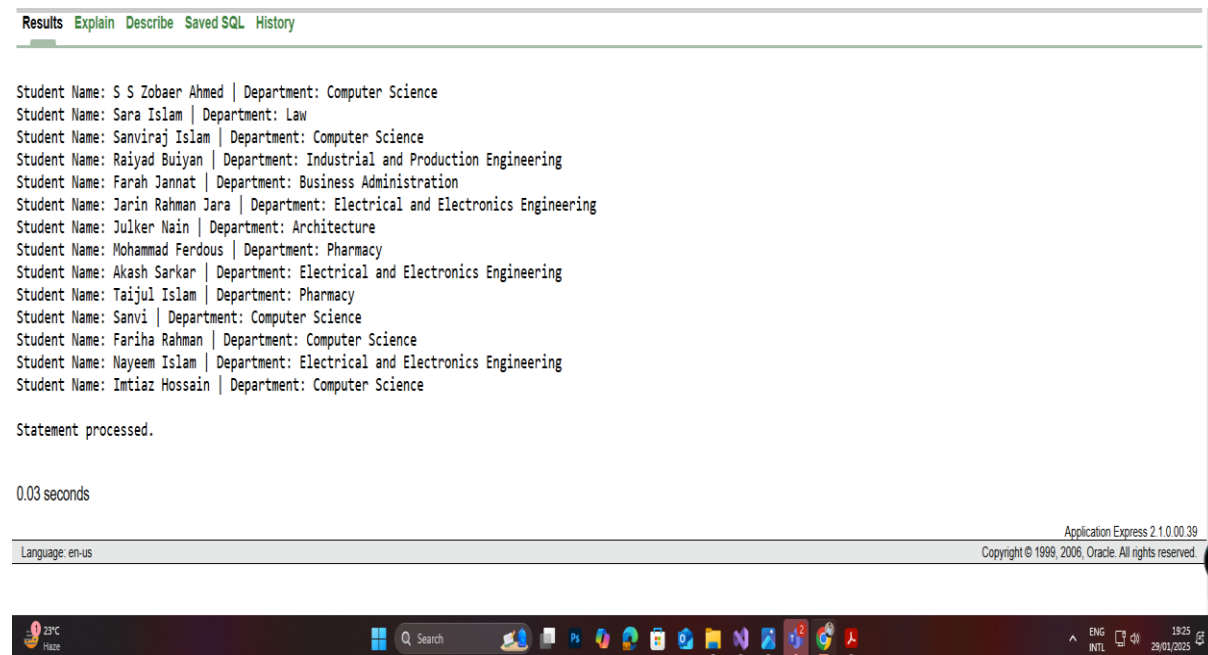
```

DECLARE
CURSOR student_cursor IS
  SELECT s.studentName, d.deptName
  FROM Student s
  JOIN Department d ON s.deptId = d.deptId;
v_studentName Student.studentName%TYPE;
v_deptName Department.deptName%TYPE;
BEGIN
  OPEN student_cursor;
  LOOP
    FETCH student_cursor INTO v_studentName, v_deptName;
    EXIT WHEN student_cursor%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE('Student Name: ' || v_studentName || ' | Department: ' ||
v_deptName);
  END LOOP;
  CLOSE student_cursor;

```

END;

/



2. Write a query to list faculty members and their respective department names using PL/SQL

**Query:**

DECLARE

CURSOR faculty\_cursor IS

SELECT f.facultyName, d.deptName

FROM FacultyMember f

JOIN Department d ON f.deptId = d.deptId;

v\_facultyName FacultyMember.facultyName%TYPE;

v\_deptName Department.deptName%TYPE;

BEGIN

OPEN faculty\_cursor;

LOOP

FETCH faculty\_cursor INTO v\_facultyName, v\_deptName;

EXIT WHEN faculty\_cursor%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Faculty Name: ' || v\_facultyName || ' | Department: ' ||

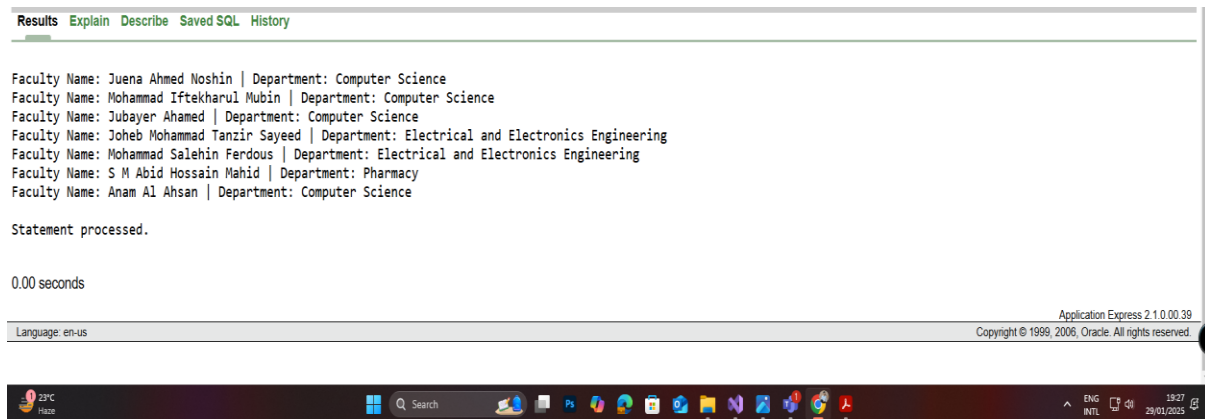
v\_deptName);

END LOOP;

CLOSE faculty\_cursor;

END;

/



# Project Proposal

## Background of the problem

Course registration is an important process for students in universities. It helps them enroll in the right courses so they can complete their degrees on time. However, many students face difficulties during registration, which can lead to **missing out on courses** and **delays in graduation**.

### Main Problems Students Face:

1. **Limited Seats in Courses:** Some students can't get into the courses they need because seats fill up too fast.
2. **No Immediate Help:** Academic advisors are often too busy to help every student during registration.
3. **Confusion About Course Selection:** Many students, especially first-year students, struggle to choose the right courses.
4. **Time Conflicts:** Sometimes, two or more chosen courses have overlapping class times.
5. **Slow Decision-Making:** If students can't decide quickly, they may lose their chance to enroll in available courses.
6. **System Issues:** The registration website may slow down or crash when too many students use it at the same time.

### The Biggest Problem:

Some students **fail to register for any courses** due to these issues, which affects their academic progress and causes stress

## Solution to the problem

### Proposed Solution: Virtual Advisor Chatbot using AI

To solve this issue, we plan to add a **Virtual Advisor Chatbot** powered by **Artificial**

**Intelligence (AI).** This chatbot will use **Natural Language Processing (NLP)** with the **OpenAI API** to help students register for courses smoothly.

**How the Virtual Advisor Chatbot Will Help:**

1. **Suggesting Courses:** It will recommend the best courses based on a student's academic progress and available options.
2. **Finding Alternatives:** If a course is full, the chatbot will suggest similar courses.
3. **Checking Requirements:** It will make sure students meet the prerequisites for a course before they enroll.
4. **Avoiding Time Conflicts:** It will help students choose courses that don't overlap in schedule.
5. **Available Anytime:** Unlike human advisors, the chatbot will be available **24/7** to assist students.
6. **Notifying About Seat Availability:** If a course is full, it will alert students when a seat opens up.
7. **Answering Questions:** The chatbot will provide information about course rules, credit limits, and registration deadlines.
8. **Helping Universities Plan Better:** The AI can predict which courses will be in high demand so universities can **offer more seats** in those courses.

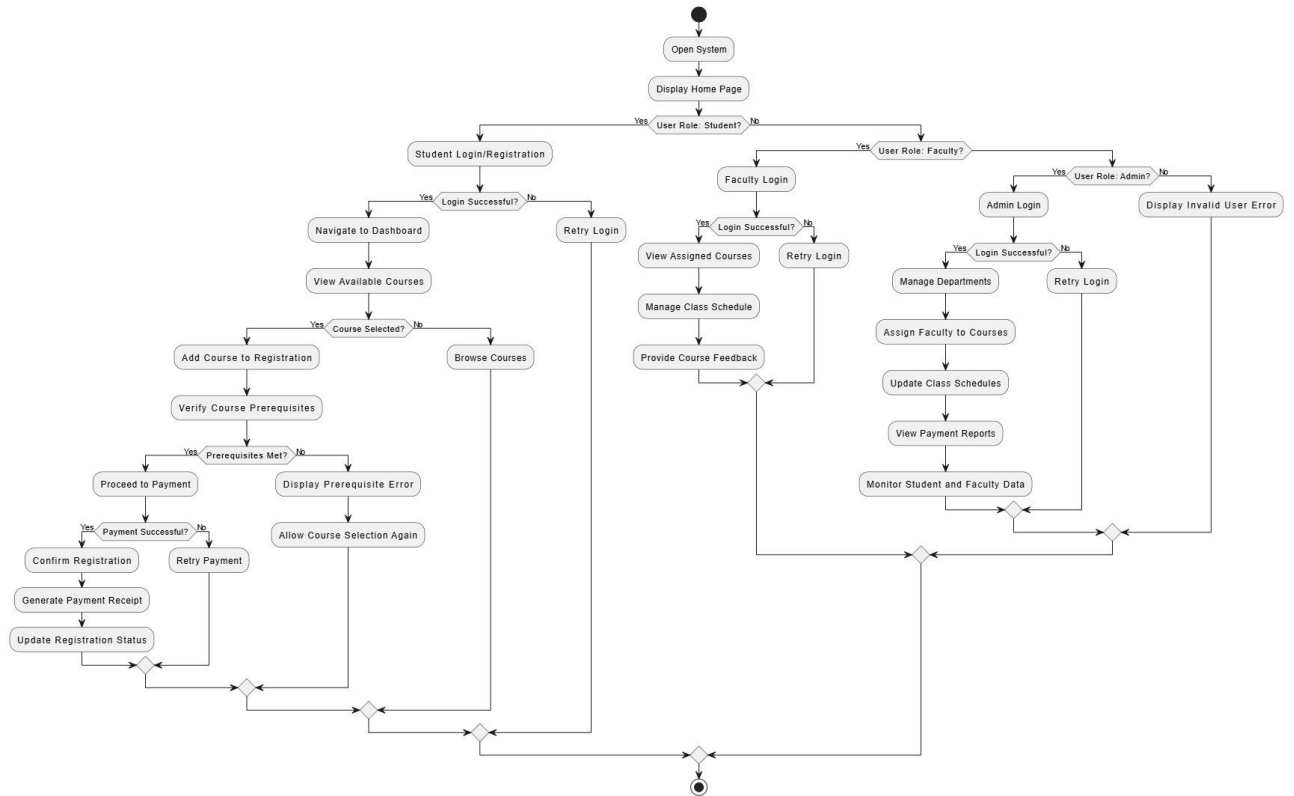
**Benefits of This Solution:**

1. **More Students Get Registered:** The chatbot will help students quickly find available **courses** so they don't miss out.
2. **Less Pressure on Human Advisors:** Since the chatbot answers common questions, advisors can focus on more complex issues.
3. **Faster Decision-Making:** Students won't have to waste time searching for courses manually.
4. **Better Student Experience:** A smooth and stress-free registration process
5. **Smarter Course Planning:** Universities will know in advance which courses need more seats.

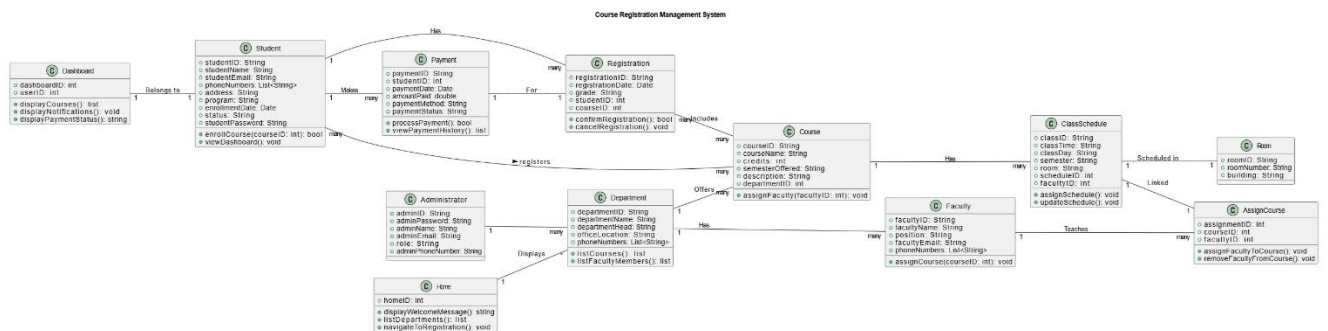


# Diagrams

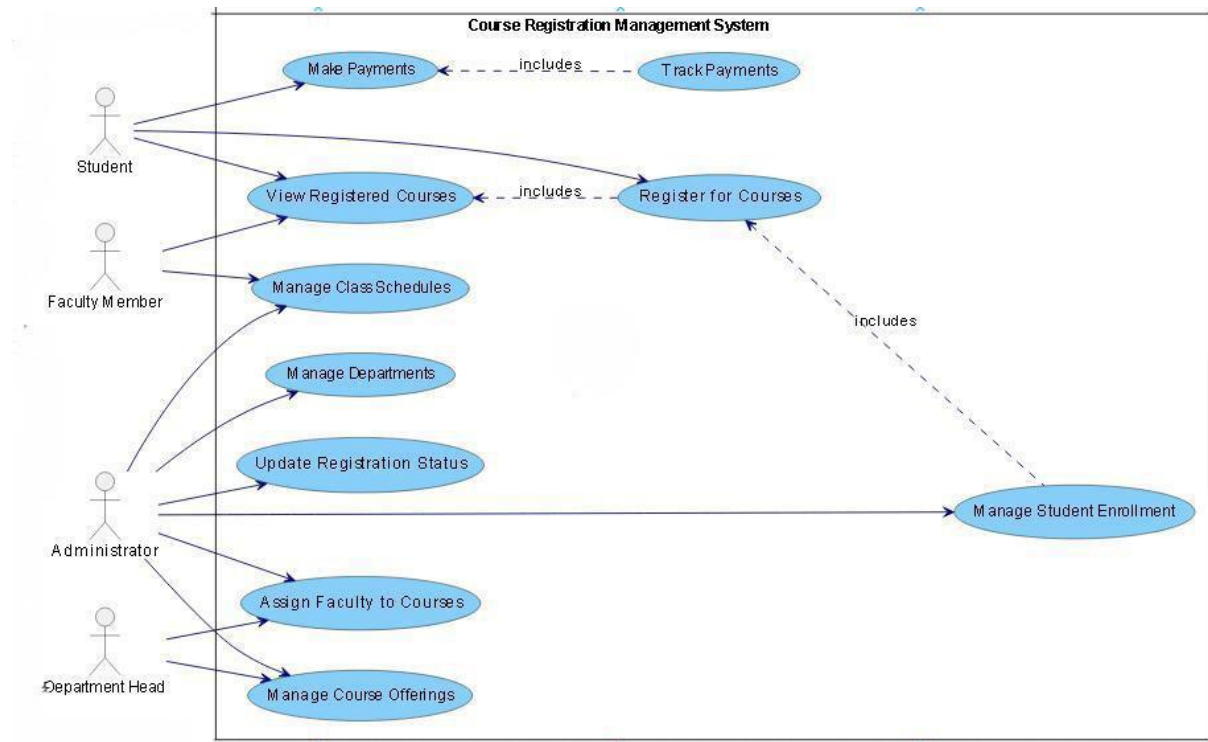
## Activity Diagram:



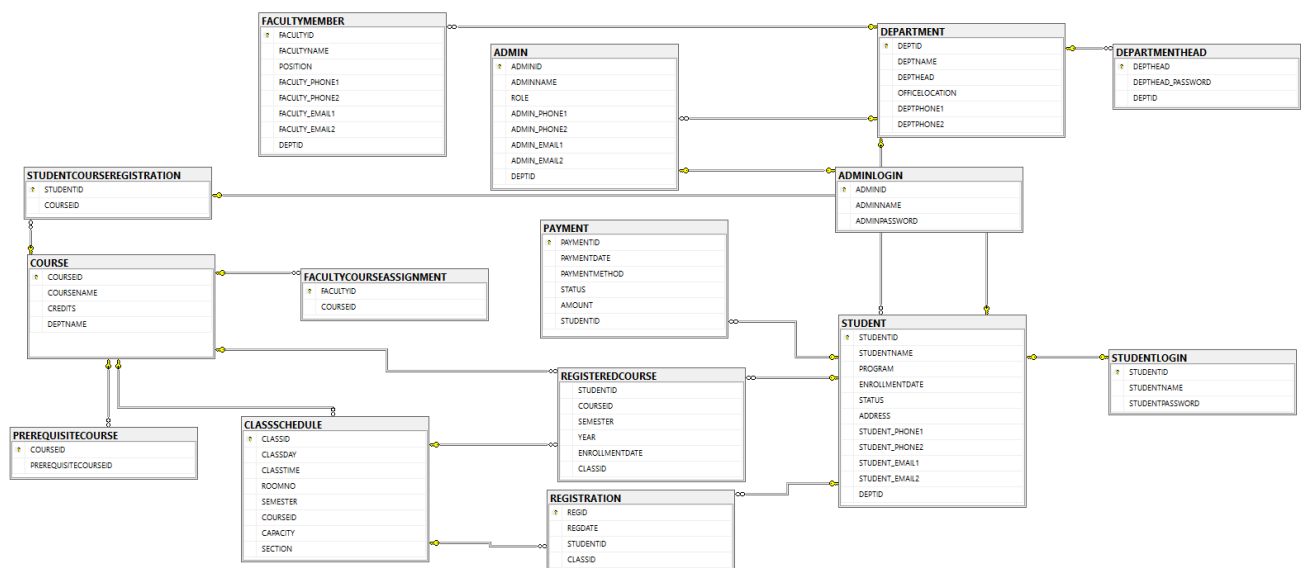
## Class Diagram:



## Use Case Diagram:



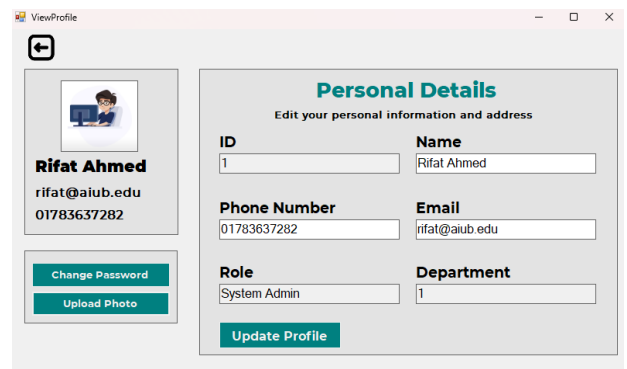
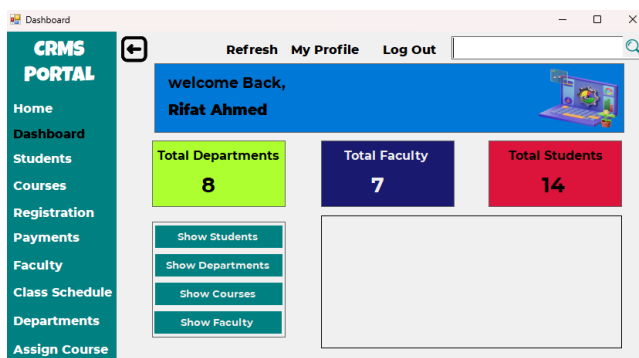
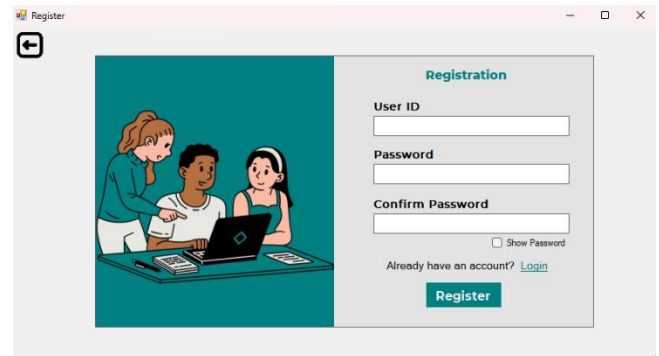
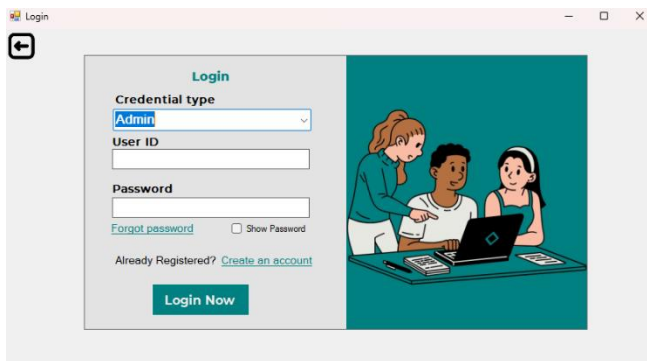
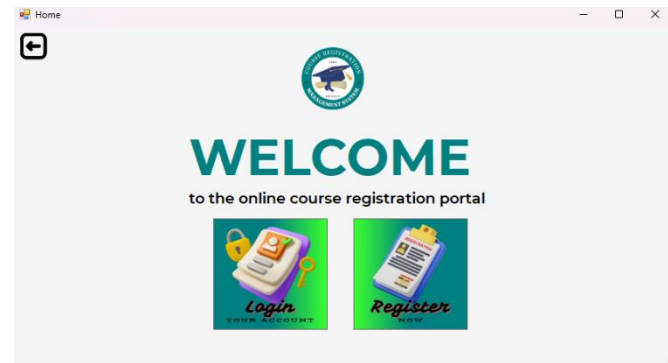
## Schema Diagram



# User Interface

## Interface Design

### Admin Panel



CRMS PORTAL

Home

Dashboard

Students

Courses

Registration

Payments

Faculty

Class Schedule

Departments

Assign Course

Add New Students

ID

Name

Program

03 February 2025

Status

Address

Phone Number

Email

Computer Science

Insert

Update

Delete Student

ID

Search

Delete

STUDENTID	STUDENTNAM	PROGRAM
1	S S Zobaer A...	CSE
2	Sara Islam	LLB
3	Sanviraj Islam	CSE
4	Riyaduzzama...	IPE
5	Farah Jannat	BBA
6	Jarin Rahma...	EEE
7	Morshed Al J...	Arch
8	Mohammad ...	B. Pharm
9	Akash Sarkar	EEE

CRMS PORTAL

Home

Dashboard

Students

Courses

Registration

Payments

Faculty

Class Schedule

Departments

Assign Course

Add New Course

Course Id

Course Name

Credits

Department

Computer Science

Insert

Update

Delete

Course

Course Id

Search

COURSEID	COURSENAM	CREDITS
NULL	No Prerequis...	0
ENG1101	ENGLISH RE...	3
CSC1101	INTRODUCTL...	1
CSC1103	INTRODUCTL...	1
CSC1102	INTRODUCTL...	3
CSC1204	DISCRETE M...	3
MAT1102	DIFFERENTIAL...	3
PHY1101	PHYSICS 1	3

CRMS PORTAL

Home

Dashboard

Students

Courses

Registration

Payments

Faculty

Class Schedule

Departments

Assign Course

Registration

Registration Id

registration Date

03 February 2025

Student ID

1

Class Id

1

Insert

Update

Search Registration

Registration Id

Search

Drop

REGID	REGDATE	STUDENTID	CL
1	29/01/2025	1	1
2	29/01/2025	2	9

CRMS PORTAL

Home

Dashboard

Students

Courses

Registration

Payments

Faculty

Class Schedule

Departments

Assign Course

Payment

Payment Id

Payment Date

03 February 2025

Payment Method

Status

Amount

Student Id

1

Insert

Search Payment

Payment Id

Search

PAYMENTID	PAYMENTDAT	PAYMENTM
1	2023-01-01	Bank Transfer
2	2023-02-01	Mobile Banki
3	2023-03-01	Credit Card
4	2023-04-01	Cash
5	2023-05-01	Bank Transfer
6	2023-06-01	Mobile Banki
7	2023-07-01	Credit Card
8	2023-08-01	Cash

CRMS PORTAL

Home

Dashboard

Students

Courses

Registration

Payments

Faculty

Class Schedule

Departments

Assign Course

Add New Department

Department Id

Department Name

Department Head

Juena Ahmed Noshin

Department Location

Department Phone

Department Phone(Optional)

Insert

Update

Search Department

Department Id

Search

DEPTID	DEPTNAME
1	Computer Science
2	Law
3	Industrial and Production Engine
4	Business Administration
5	Electrical and Electronics Enginee
6	Architecture
7	Pharmacy
8	Chemistry

## Student Panel

studentDashboard

CRMS PORTAL

Home

Dashboard

Students

Courses

Registration

Payments

Faculty

Class Schedule

Departments

Assign Course

Personal Details

Edit your personal information and address

ID

1

Name

S S Zobaer Ahmed

Phone Number

01713293375

Email

zobaer@student.aiub.edu

Program

CSE

Status

Active

Update Profile

CourseRegistration

CRMS PORTAL

Home

Dashboard

Students

Courses

Registration

Payments

Faculty

Class Schedule

Departments

Assign Course

Select Courses

Select	CLASSID	CLASSDAY	CLASSTIME	ROOMNO	SEMESTER	COURSEID	CAPACITY
<input type="checkbox"/>	16	ST	16:00-17:30	P1616	Spring 2025	PHY1101	39
<input type="checkbox"/>	17	MW	08:00-09:30	Q1717	Spring 2025	PHY1101	40
<input type="checkbox"/>	8	M	09:00-12:00	H808	Spring 2025	CSC1101	38
<input type="checkbox"/>	9	T	13:00-16:00	J909	Spring 2025	CSC1103	40
<input type="checkbox"/>	10	CT	14:00-17:30	J1010	Spring 2025	CSC1103	38

Add the course

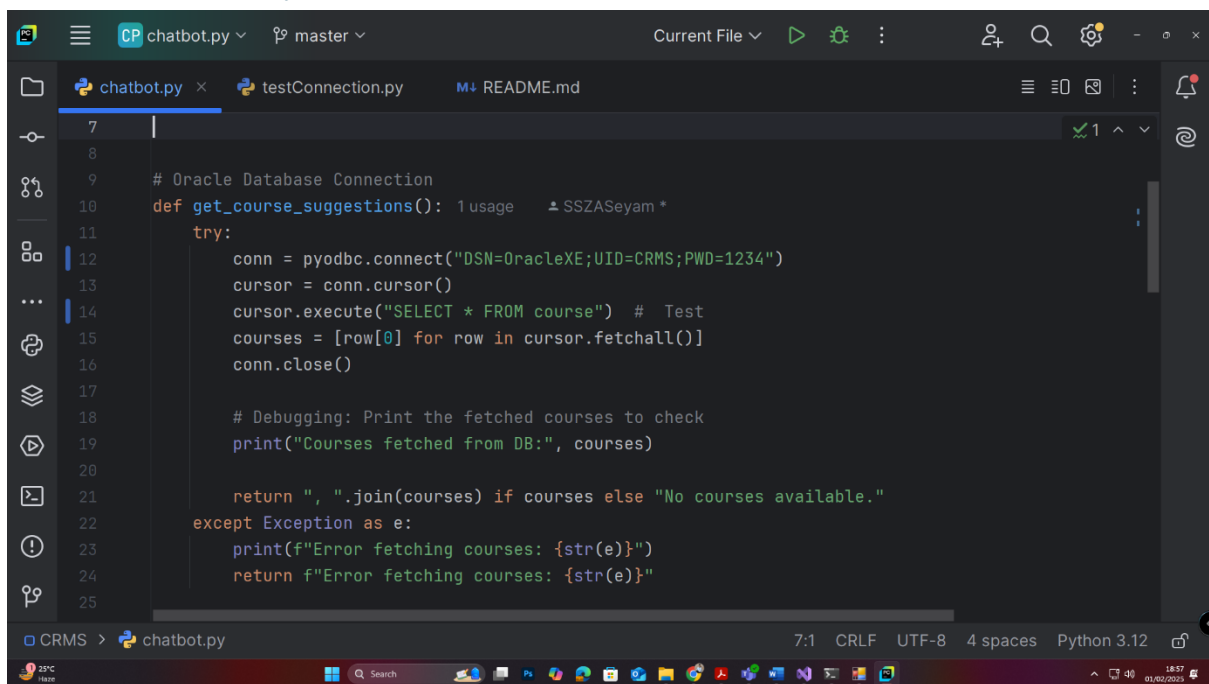
Registered Courses

Select	STUDENTID	COURSEID	SEMESTER	YEAR	ENROLLMENTE	CLASSID
<input type="checkbox"/>	1	ENG1101	Spring 2025	2025	02/02/2025	4
<input type="checkbox"/>	1	EEE2103	Spring 2025	2025	02/02/2025	36
<input type="checkbox"/>	1	MAT1102	Spring 2025	2025	02/02/2025	14

Remove the course



## Connection in Python



The screenshot shows a VS Code editor with a file named `chatbot.py` open. The code is a Python script that connects to an Oracle database using `pyodbc`. It defines a function `get_course_suggestions()` that attempts to fetch all rows from a table named `course`. If successful, it prints the fetched courses and returns them as a string. If an exception occurs, it prints an error message and returns it. The script includes comments for debugging and usage.

```
7 |
8 |
9 | # Oracle Database Connection
10 | def get_course_suggestions(): 1 usage SSZASeyam *
11 |     try:
12 |         conn = pyodbc.connect("DSN=OracleXE;UID=CRMS;PWD=1234")
13 |         cursor = conn.cursor()
14 |         cursor.execute("SELECT * FROM course") # Test
15 |         courses = [row[0] for row in cursor.fetchall()]
16 |         conn.close()
17 |
18 |         # Debugging: Print the fetched courses to check
19 |         print("Courses fetched from DB:", courses)
20 |
21 |         return ", ".join(courses) if courses else "No courses available."
22 |     except Exception as e:
23 |         print(f"Error fetching courses: {str(e)}")
24 |         return f"Error fetching courses: {str(e)}"
25 |
```

## Query Writing

### Exception handling

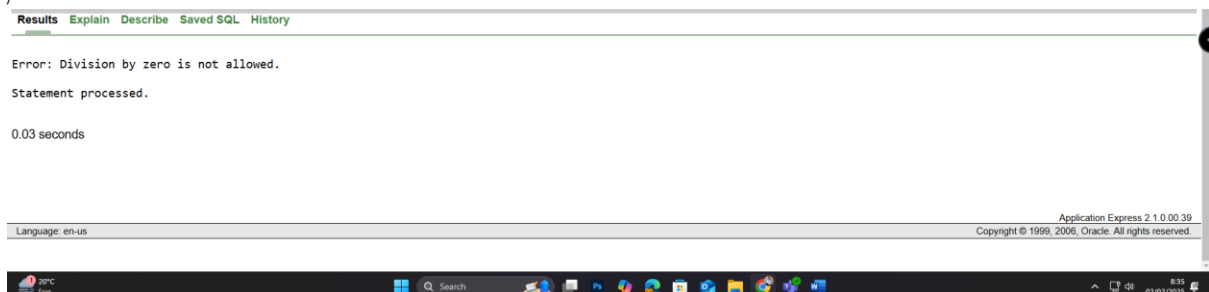
**Question 1:** Write a PL/SQL block to handle division by zero.

**Answer:**

```
DECLARE
    num1 NUMBER := 10;
    num2 NUMBER := 0;
    result NUMBER;
BEGIN
    result := num1 / num2;
    DBMS_OUTPUT.PUT_LINE('Result: ' || result);
EXCEPTION
    WHEN ZERO_DIVIDE THEN
        DBMS_OUTPUT.PUT_LINE('Error: Division by zero is not allowed.');
```

END;

/



The screenshot shows the Oracle SQL Developer Results window. It displays the output of the PL/SQL block execution. The first line is "Error: Division by zero is not allowed.", followed by "Statement processed." and "0.03 seconds". The window also shows the "Results" tab and the "Language: en-us" setting.

```
Results Explain Describe Saved SQL History

Error: Division by zero is not allowed.
Statement processed.
0.03 seconds

Language: en-us
Application Express 2.1.0.00.39
Copyright © 1999, 2006, Oracle. All rights reserved.
```

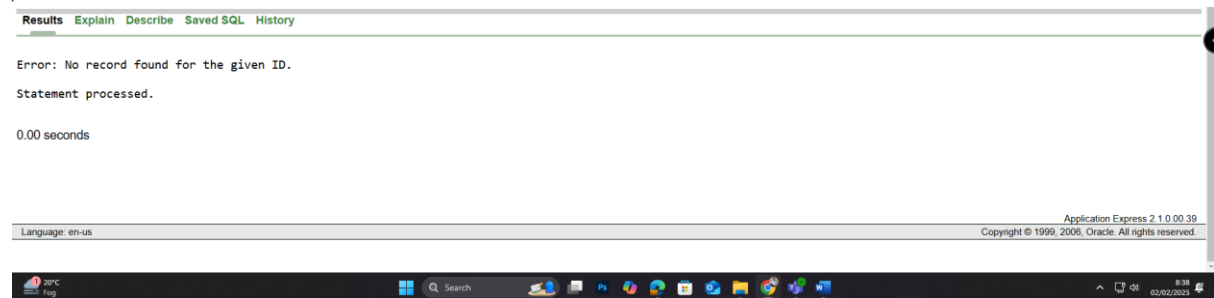
**Question 2:** Write a PL/SQL block to handle a NO\_DATA\_FOUND exception when querying an empty table.

**Answer:**

```
DECLARE
    deptName VARCHAR2(100);
BEGIN
    SELECT DEPTNAME INTO deptName FROM DEPARTMENT WHERE DEPTID = 0;
    DBMS_OUTPUT.PUT_LINE('Name: ' || deptName);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Error: No record found for the given ID.');
```

```
END;
```

```
/
```



## Implicit Locking

**Question 1:** Write a PL/SQL block to demonstrate implicit locking from student table using an UPDATE statement.

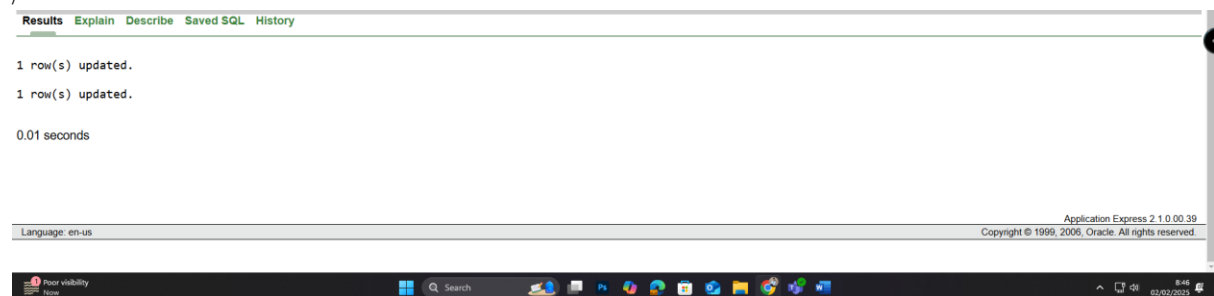
**Answer:**

```
DECLARE
    v_rows NUMBER;
BEGIN
    UPDATE student
    SET studentname = 'Sanvi Raj'
    WHERE studentid = 7;

    v_rows := SQL%ROWCOUNT;
    DBMS_OUTPUT.PUT_LINE(v_rows || ' row(s) updated.');
```

```
END;
```

```
/
```



**Question 2:** Write a PL/SQL block to demonstrate implicit locking from faculty member table using DELETE.

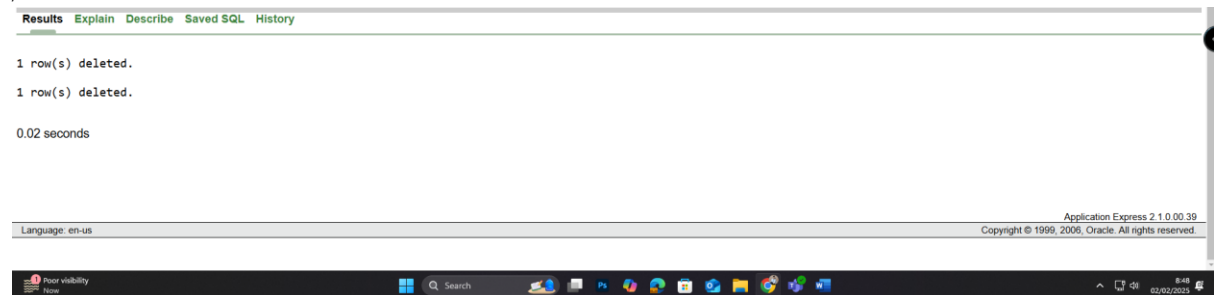
**Answer:**

```
DECLARE
    v_rows NUMBER;
BEGIN
    DELETE FROM facultyMember
    WHERE facultyid = 5;

    v_rows := SQL%ROWCOUNT;
    DBMS_OUTPUT.PUT_LINE(v_rows || ' row(s) deleted.');
```

END;

/



## Explicit Locking

**Question 1:** Write a PL/SQL block to demonstrate explicit locking on Department table using SELECT FOR UPDATE .

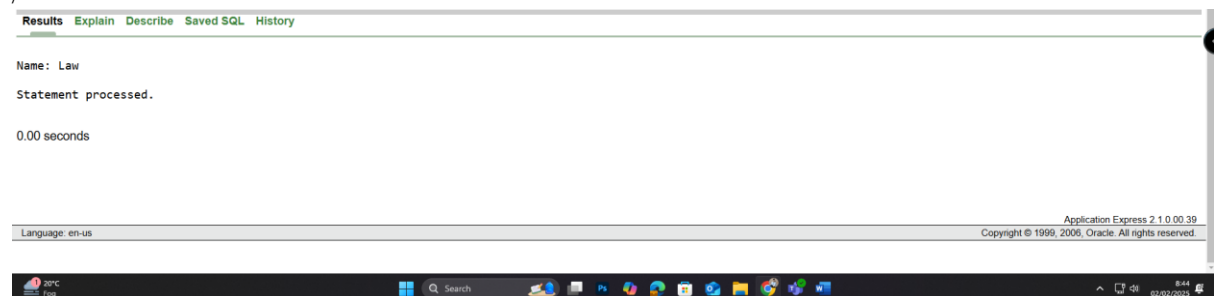
**Answer:**

```
DECLARE
    dept_Name VARCHAR2(50);
BEGIN
    SELECT deptname
    INTO dept_Name
    FROM Department
    WHERE Deptid = 2
    FOR UPDATE;

    DBMS_OUTPUT.PUT_LINE('Name: ' || dept_Name);
```

END;

/



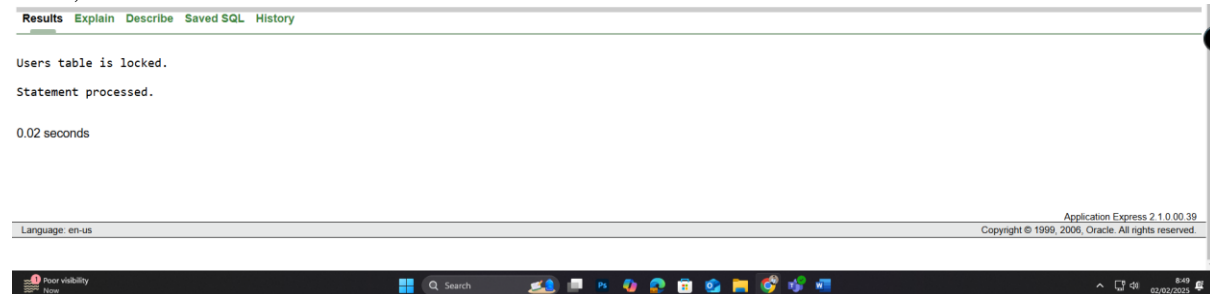


**Question 2:** Write a PL/SQL block to demonstrate the use of LOCK TABLE for explicit locking.

**Answer:**

```
BEGIN
    LOCK TABLE registeredcourse IN EXCLUSIVE MODE;
    DBMS_OUTPUT.PUT_LINE('Users table is locked.');
```

```
END; /
```



## Relational Algebra

### 1. Retrieve all students from the "Computer Science" department

**Relational Algebra:**

$$\sigma_{\text{DEPTNAME} = \text{'Computer Science'}}(\text{STUDENT})$$

**Explanation:**

This selection ( $\sigma$ ) retrieves all students whose DEPTNAME is "Computer Science" from the STUDENT table.

### 2. Retrieve student names and emails who are registered for a course

**Relational Algebra:**

$$\pi_{\text{STUDENTNAME}, \text{STUDENT\_EMAIL1}}(\text{REGISTEREDCOURSE} \bowtie \text{STUDENT})$$

**Explanation:**

This natural join ( $\bowtie$ ) combines REGISTEREDCOURSE and STUDENT on STUDENTID. The projection ( $\pi$ ) retrieves only STUDENTNAME and STUDENT\_EMAIL1.

### 3. Find all students who have registered for at least two different courses

**Relational Algebra:**

$$\pi_{\text{STUDENTID}}(\text{REGISTEREDCOURSE} \bowtie \rho_{C1}(\text{REGISTEREDCOURSE})) - \pi_{\text{STUDENTID}}(\text{REGISTEREDCOURSE})$$

**Explanation:**

This self-join ( $\bowtie$ ) finds students who have more than one course registered. The set difference ( $-$ ) ensures only students with at least two courses are selected.

### 4. Find students who have registered for "Introduction to Database"

**Relational Algebra:**

$$\pi_{\text{STUDENTNAME}}(\sigma_{\text{COURSENAME} = \text{'Introduction to Database'}}(\text{COURSE}) \bowtie \text{REGISTEREDCOURSE} \bowtie \text{STUDENT})$$

**Explanation:**

The **selection** ( $\sigma$ ) retrieves only rows where COURSENAME = 'Introduction to database'.

The **join** ( $\bowtie$ ) ensures we get students registered for this course.

The **projection** ( $\pi$ ) returns only STUDENTNAME.

**5. Find all students who have registered for a course or taken a prerequisite course****Relational Algebra:**

$\Pi \text{ STUDENTID}(\text{REGISTEREDCOURSE}) \cup \pi \text{ STUDENTID}$   
 $(\text{PREREQUISITECOURSE} \bowtie \text{REGISTEREDCOURSE})$

**Explanation:**

The union ( $\cup$ ) retrieves students who have either registered for a course or have completed at least one prerequisite course.

The join ( $\bowtie$ ) ensures we fetch prerequisite-related students from REGISTEREDCOURSE.