

# Course Registration Management System



## American International University-Bangladesh

**Course name:** Advanced Database Management System

**Section:** B

**Department:** Computer Science and Engineering

**Group No:** 06

**Submission Date:** May 15, 2023

<b>Project Name:</b> Course Registration Management System.
---

### Submitted By:

Name	ID	Contribution
NOBONITA NONDE	20-43819-2	Normalization, Table Creation, Data Insertion, Interface description.

### 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>Use Case Diagram .....</b>	<b>3</b>
<b>Activity Diagram .....</b>	<b>4</b>
<b>Class Diagram .....</b>	<b>5</b>
<b>Interface Design and description .....</b>	<b>6</b>
Welcome page .....	6
Accountant Login .....	7
Accountant Dashboard .....	7
Department Login .....	8
Department Dashboard .....	8
Faculty Login.....	9
Faculty Dashboard .....	9
Student Login .....	10
Student Dashboard .....	10
<b>Scenario description .....</b>	<b>10 ER</b>
<b>Diagram .....</b>	<b>11</b>
<b>Normalization .....</b>	<b>11</b>
<b>Schema Diagram .....</b>	<b>17</b>
<b>Table Creation .....</b>	<b>17</b>
<b>Data Insertion .....</b>	<b>28</b>
<b>SQL Query.....</b>	<b>36</b>
<b>Relational Algebra .....</b>	<b>56</b>
<b>Conclusion .....</b>	<b>57</b>

# Course Registration Management System

## **Introduction:**

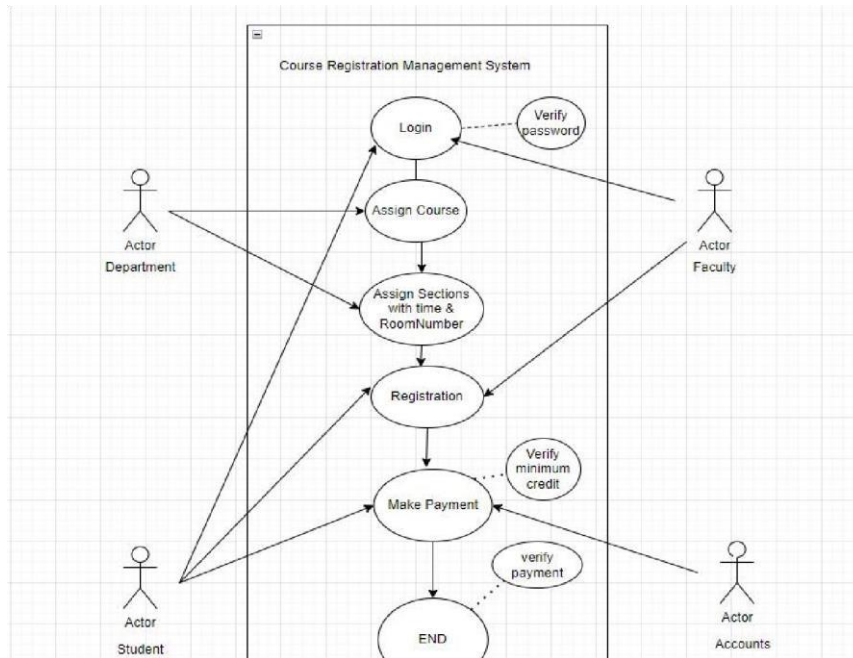
A course registration management system is a software application designed to manage the process of student registration for courses offered by educational institutions. The system will provide a platform for students to view course offerings, select courses, register for courses, and manage their registration status. The system will also provide the Department with tools to manage course offerings, student registration, and student enrollment 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 course offerings, student enrollment and other related stuff. The faculty who will be taking courses will also have the option to enroll in their preferred times and schedule. The course management system will be implemented using the combinations of programming languages, database management systems and web technologies. The specific tools and technologies used will depend on requirement and preferences of the institute.

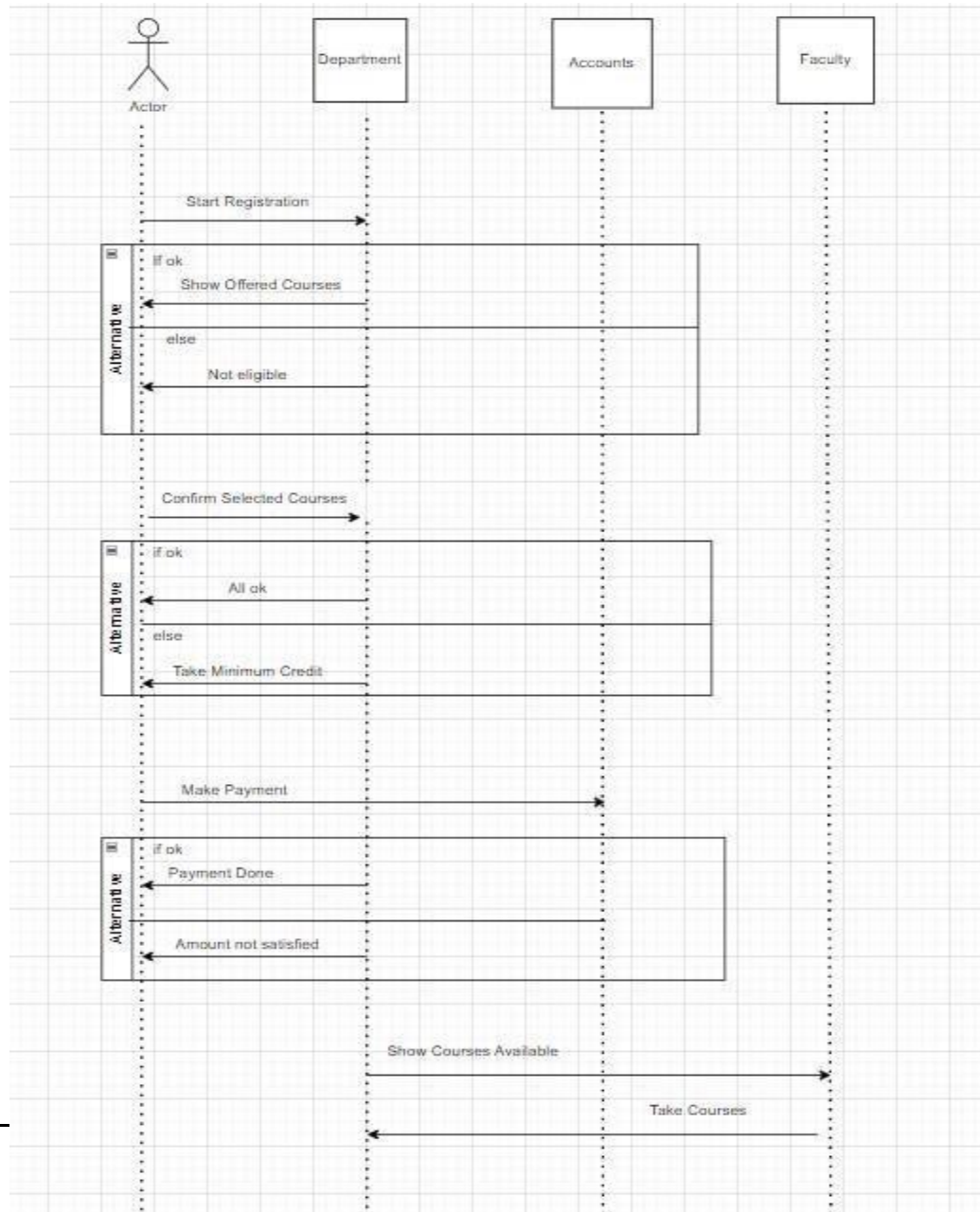
# Course Registration Management System

## Use Case Diagram:



# Course Registration Management System

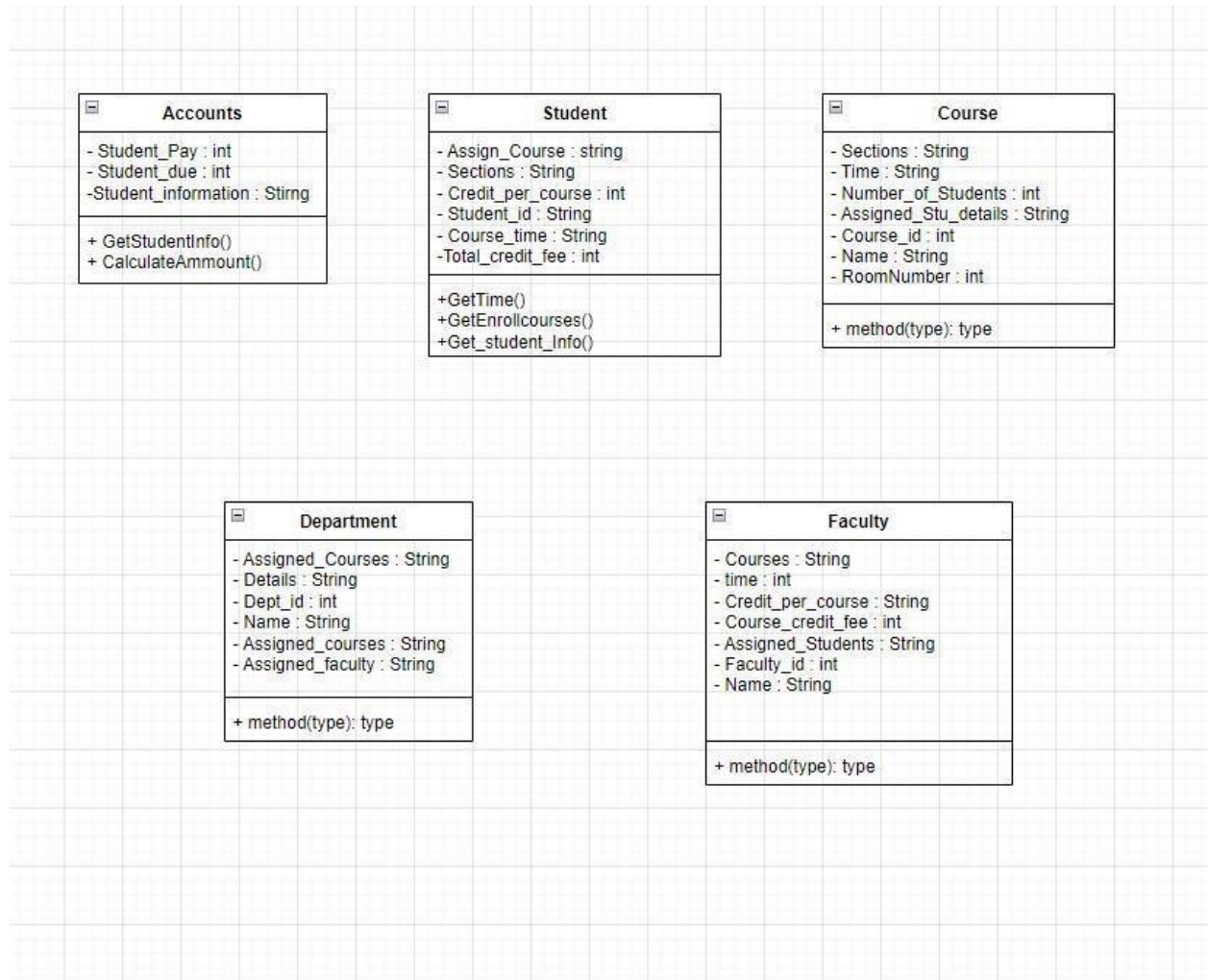
## Activity Diagram:



# Course Registration Management System

## Class Diagram:

Figure: Class Diagram



# Course Registration Management System

## **Interface Design and Description:**

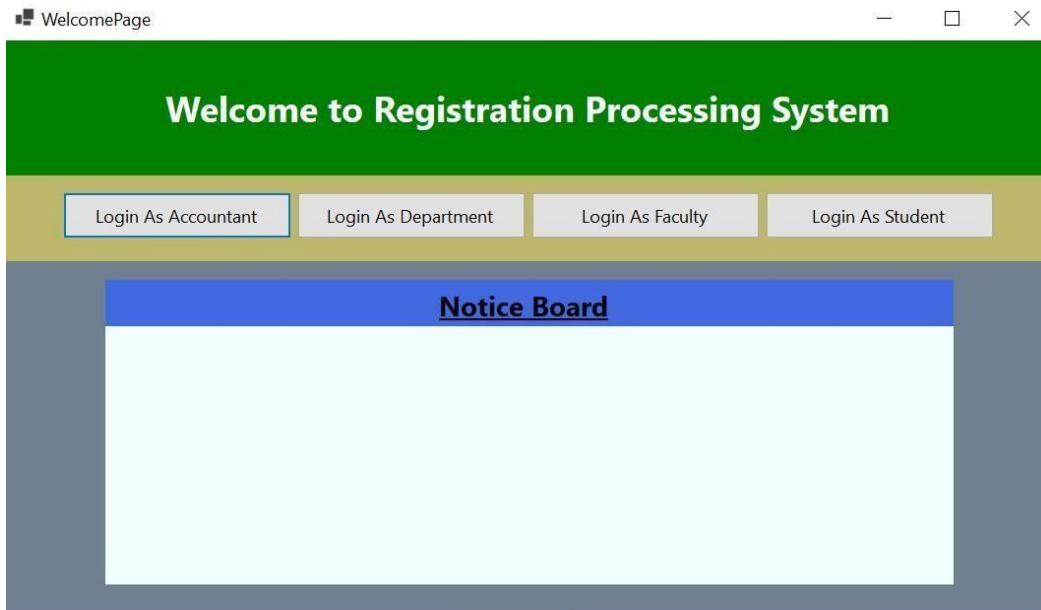
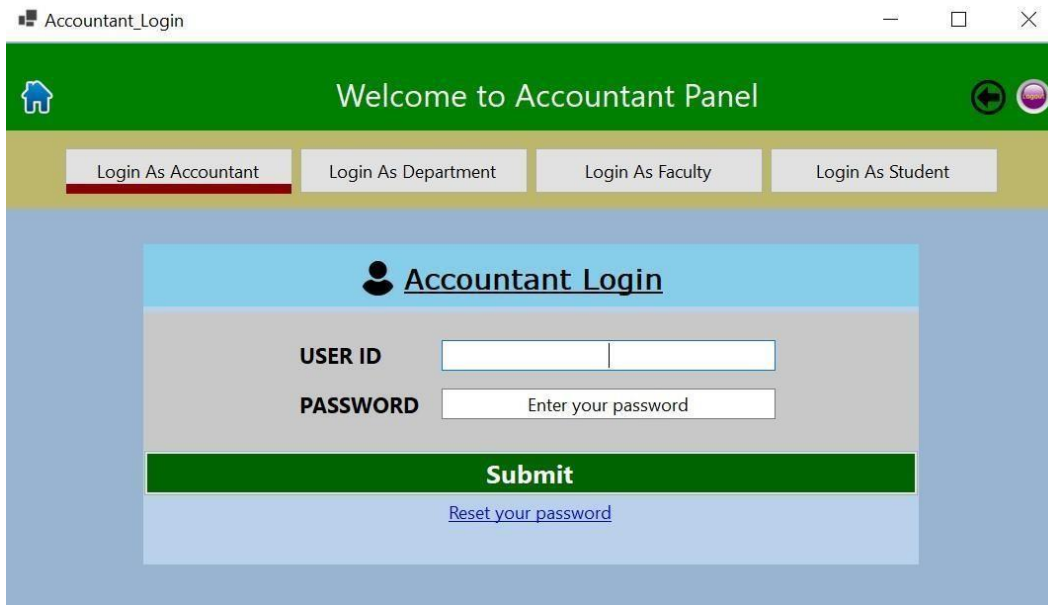


Figure 1: Welcome\_Page

After opening the app user will see 4 buttons for 4 types of users. “Login as Accountant” for Accountant login, “Login as Department” for department users. “Login as Faculty” and “Login As Student” is use for according to Faculty and Students. Different type of user use different login button for login their dashboard. There also a notice board, all kind of user will seem it.

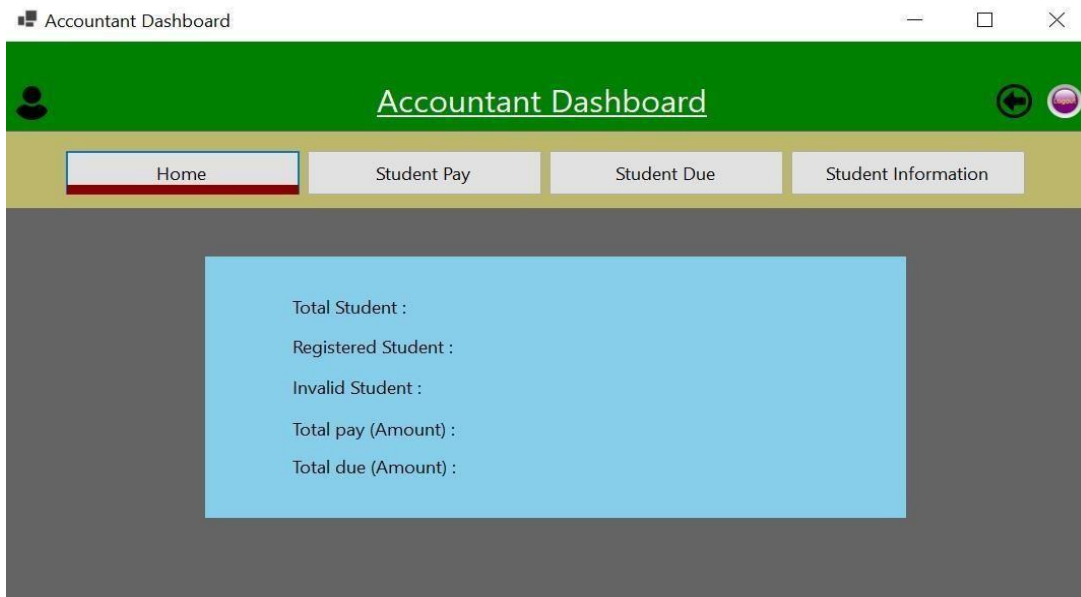
# Course Registration Management System



The screenshot shows a web application window titled "Accountant\_Login". The main header is green with a home icon on the left and navigation icons on the right. Below the header is a yellow navigation bar with four buttons: "Login As Accountant" (highlighted with a red underline), "Login As Department", "Login As Faculty", and "Login As Student". The main content area has a light blue background. In the center, there is a white box with a blue header "Accountant Login" featuring a user icon. Below this header are two input fields: "USER ID" and "PASSWORD". The "PASSWORD" field has a placeholder text "Enter your password". Below the input fields is a green "Submit" button. At the bottom of the white box is a blue link "Reset your password".

Figure 2: Accountant Login

If the user will be accountant, after selecting Login as Accountant, Accountant login panel will be open. Giving user id and password correctly, user will press submit to go to their dashboard.



The screenshot shows a web application window titled "Accountant Dashboard". The main header is green with a user icon on the left and navigation icons on the right. Below the header is a yellow navigation bar with four buttons: "Home" (highlighted with a red underline), "Student Pay", "Student Due", and "Student Information". The main content area has a dark gray background. In the center, there is a light blue box containing the following text:

- Total Student :
- Registered Student :
- Invalid Student :
- Total pay (Amount) :
- Total due (Amount) :

Figure 3: Accountant Dashboard

After login properly a page will be appeared to the accountant where has 4 buttons. Home,

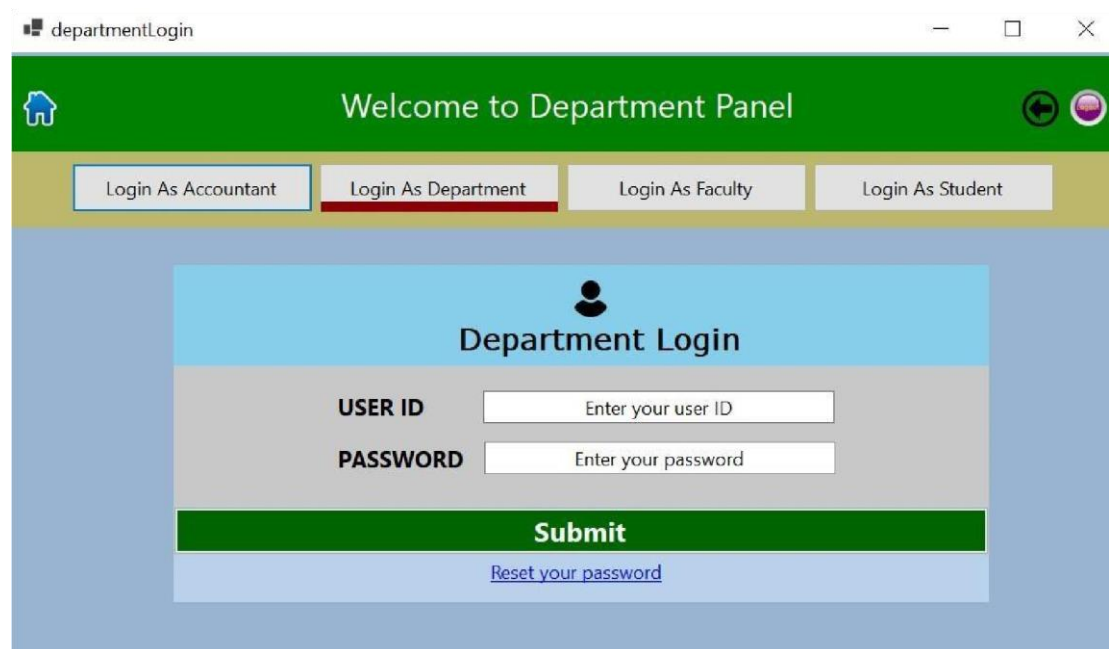


## Course Registration Management System

Student Pay, Student Due and Student Information. Just entering the accountant dashboard, Accountant will show some information. Such as Total student number, Registered student, Invalid student etc. Selecting student pay accountant can see which student are pay for courses and which amount. Selecting “Student Due” accountant can see How much a student have to pay.

Figure 4: Department Login

If the user will be a Department, in the starting page thy have to select “Logging As Department” then the Department Login panel will be appear.



The screenshot shows a web browser window titled "departmentLogin". The main header is green with a home icon on the left and "Welcome to Department Panel" in the center. Below the header is a navigation bar with four buttons: "Login As Accountant", "Login As Department" (which is highlighted with a red underline), "Login As Faculty", and "Login As Student". The main content area has a light blue background. In the center, there is a "Department Login" panel. This panel has a light blue header with a user icon. Below the header, there are two input fields: "USER ID" with the placeholder text "Enter your user ID" and "PASSWORD" with the placeholder text "Enter your password". Below these fields is a green "Submit" button. At the bottom of the panel, there is a link that says "Reset your password".

# Course Registration Management System

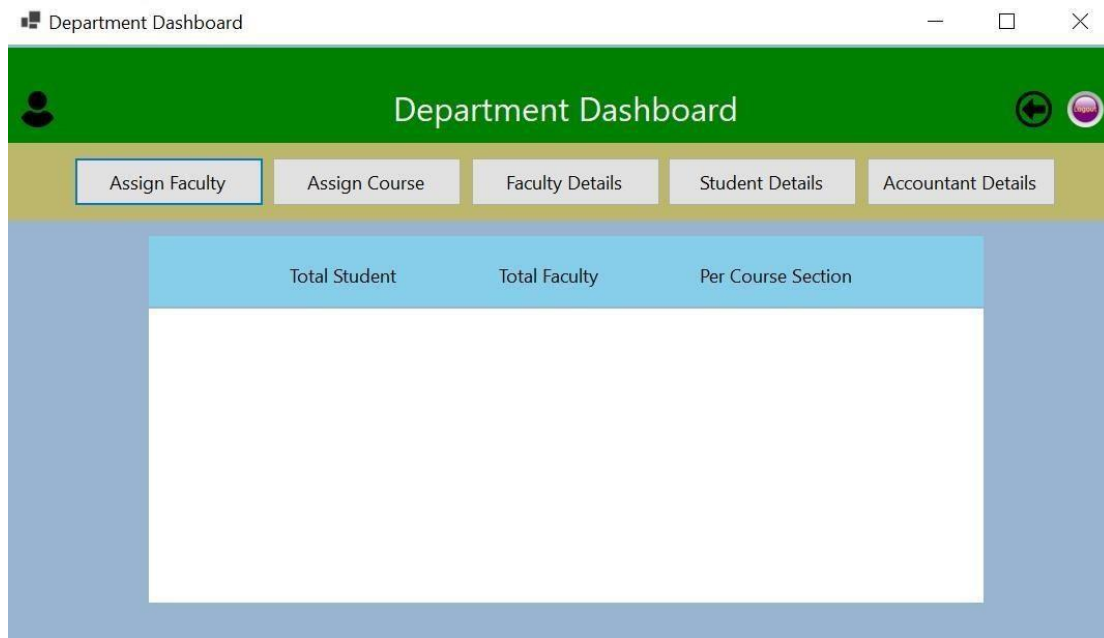


Figure 5: Department Dashboard

After proper login Department Dashboard will be appear where has 5 buttons. Assign Faculty is for assigning faculties for courses. Assign Course is use for assigning course for students. Other 3 buttons are use for seeing other 3 users details.

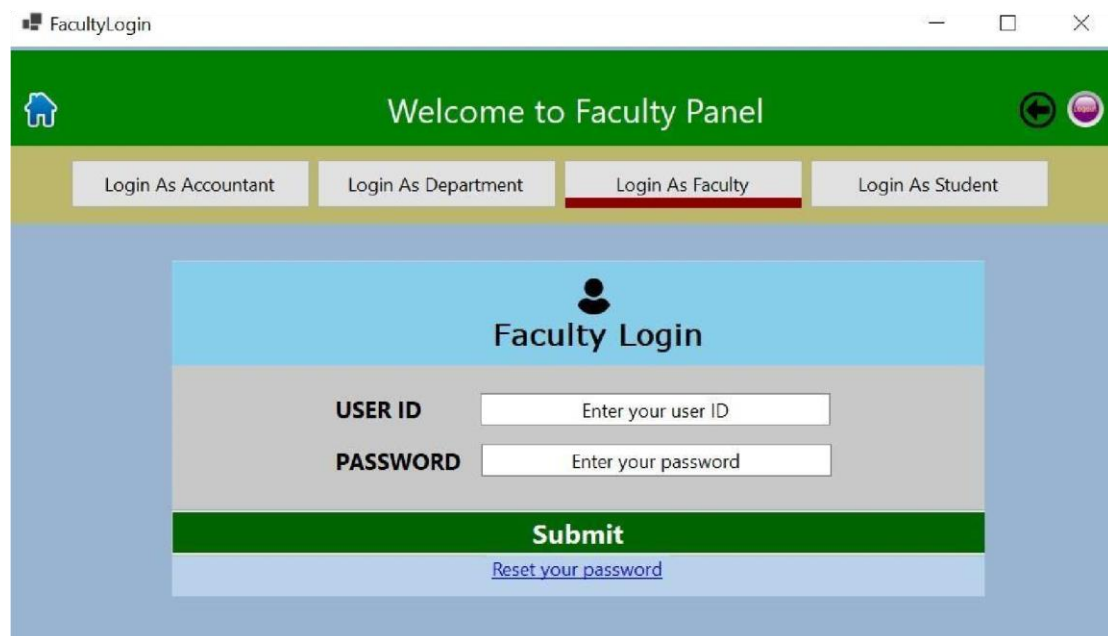


Figure 6: Faculty Login

# Course Registration Management System

If the user will be a Faculty, in the starting page they have to select “Logging As Faculty” then the Faculty Login panel will be appear. Clicking submit button then the user go to the Faculty Dashboard.

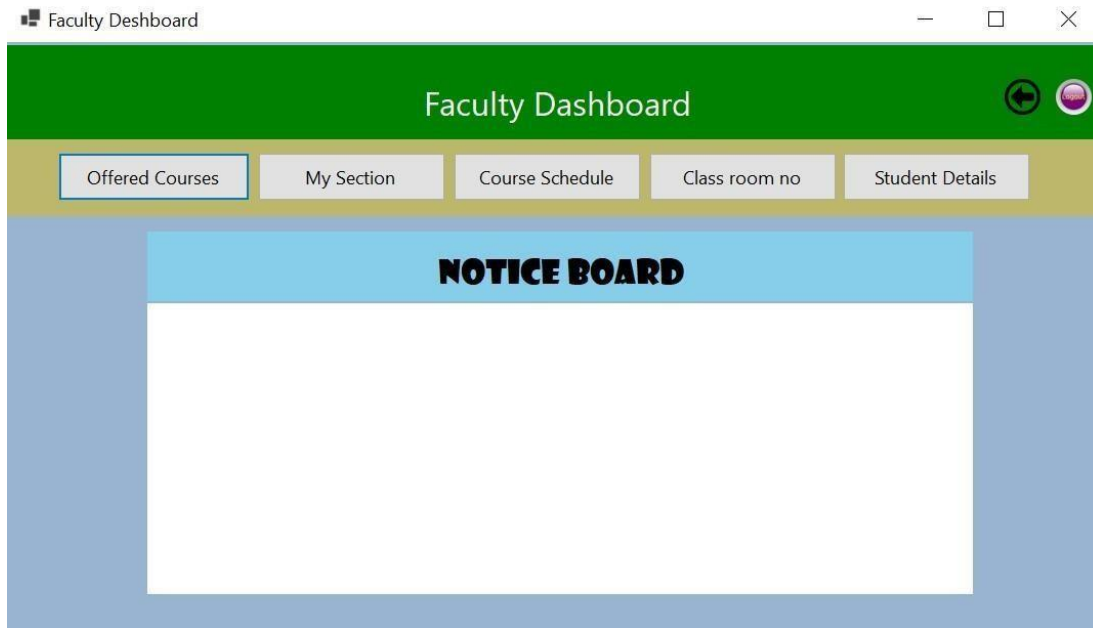


Figure 7: Faculty Dashboard

In the Faculty Dashboard there are 5 buttons. Offered Courses for use to see the offer course which they are declared by the department. My section is user for to see the courses which they are assigned. They can see their course schedule, Classroom no and student details by pressing different type for buttons.

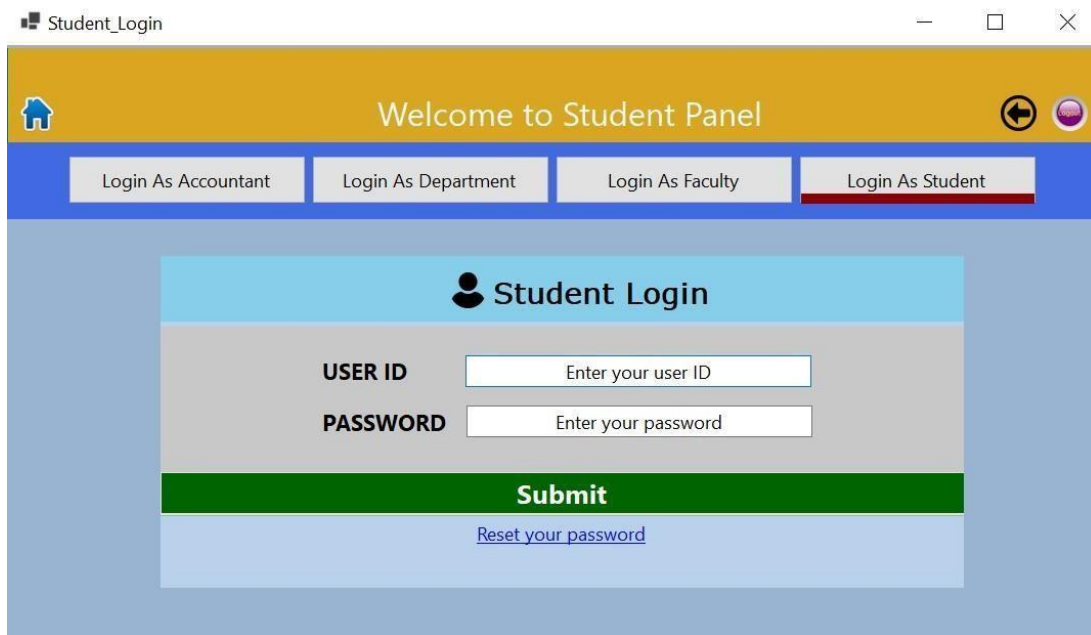


Figure 8: Student Login

# Course Registration Management System

If the user will be a student, in the starting page they have to select “Logging As Student” then the Student Login panel will be appear. Clicking submit button then the user go to the Student Dashboard.

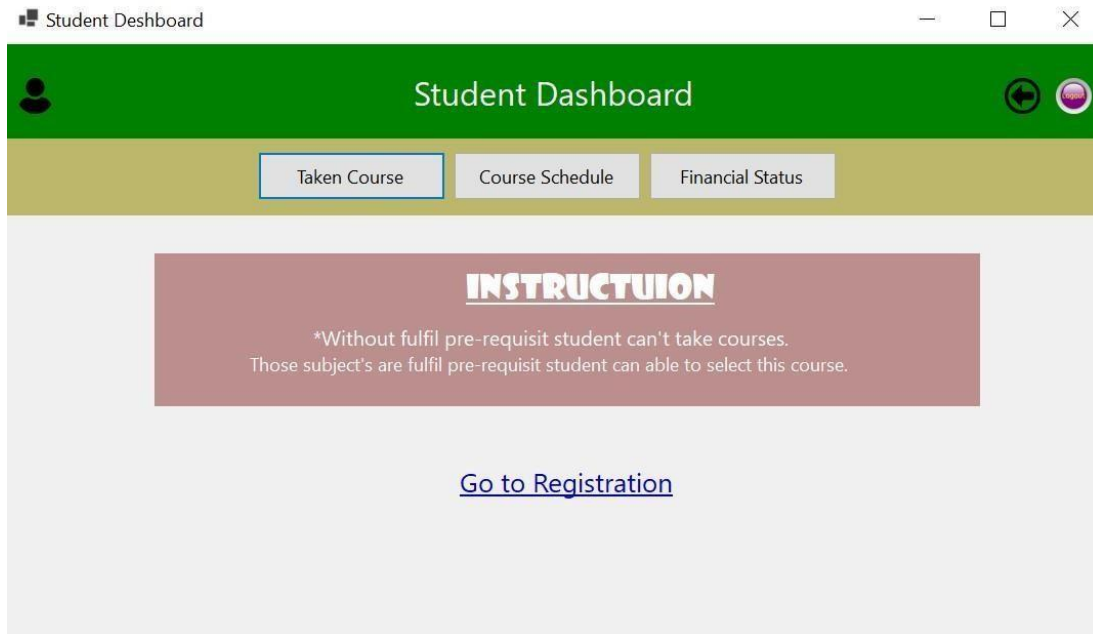


Figure 9: Student Dashboard

In the student dashboard there are 3 buttons. By pressing Taken Course user can see the courses which they are taken. Course schedule for seeing the schedule. Financial status for knowing the payable amount after confirming the registration process.

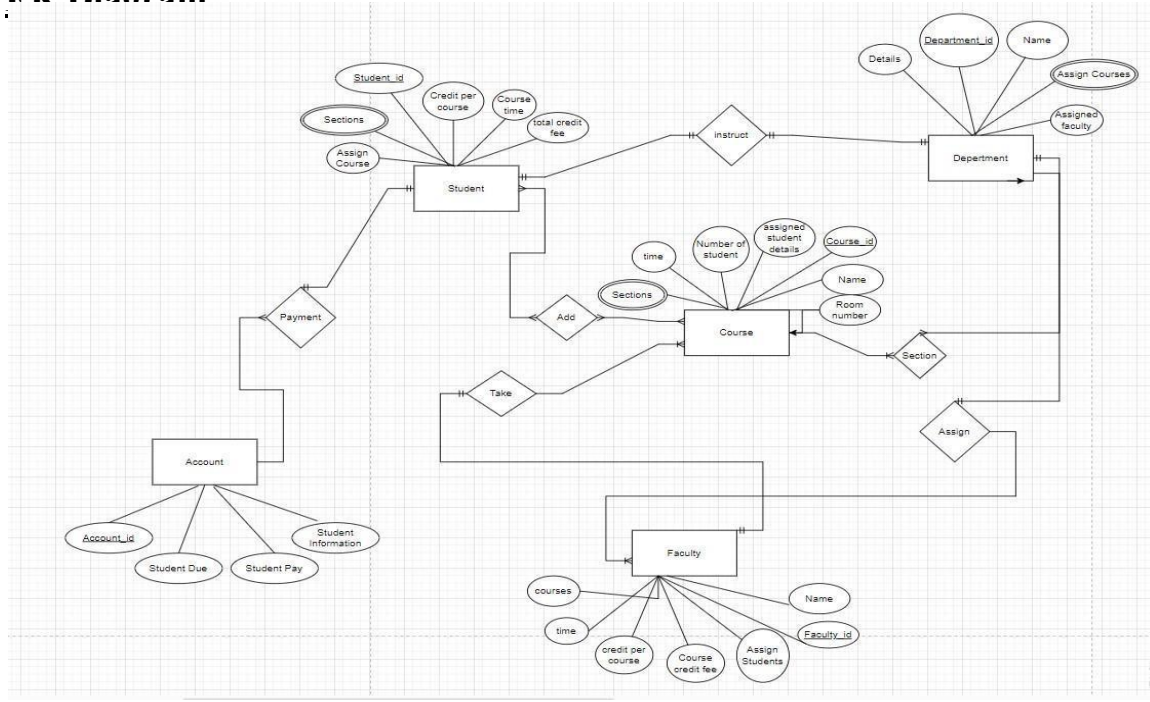
## **Scenario Description:**

In Course registration management system software students can enroll to courses of their interest , match their chosen times and faculties. Students after completing their registration and enrollment into their desired courses have to make payments. Students have the flexibility to pay their registration fee in several installments. The department can offer courses with septic blocks of times and sections so that students can enroll in the course with their desired times . Department can also manage the credits taken by the students in their particular semester and maintain their grades . Department after the enrollment of the students can offer those sections which are filled with desired students to faculty members who are eligible to take the courses. After the faculty member chooses their courses by their chosen times and period the department can complete the course registration. The Accountant can after the course registration is complete can take the payment from the student and keep records of those transactions. Accountants can also view the records of students' payment history. The faculty can enroll in courses offered by the Department and also see the number of students enrolled in it with room

# Course Registration Management System

number and time . Faculty can also decide to drop the course if their desired time and schedule is not eligible.

## ER Diagram:



## Normalization:

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

1NF- sections are multivalued attribute.

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

2NF- Accountant\_id, Students\_pay, Students\_due, Students\_information. Students\_id, Assign\_Course, Sections, Credit\_per\_course, Course\_time, total\_credit\_fee.

3NF- Accountant\_id, Students information. S transaction, Students\_pay, Students\_due.

## Course Registration Management System

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

### Final table-

Accountant\_id, Students information, S\_transaction, Studnets\_id, Course\_info,  
S\_transaction, Students\_pay, Students\_due.  
Studnets\_id, Assign Course, Credit per course, total credit fee. 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, Assigned Courses are multivalued attribute. 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.

### Final table-

1. Studnets\_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.

# Course Registration Management System

1NF- Sections, 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- Studnets\_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. Studnets\_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.

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.

# Course Registration Management System

## 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.

## Final table-

Department\_id, Assigned course, Assigned faculty, s\_info, course\_id, class\_info

S\_info, Details, Name.

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

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



# Course Registration Management System

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-

Department\_id, Assigned course, Assigned faculty, s\_info, faculty\_id, course\_d s\_info,  
Details, Name.  
faculty\_id, Courses, time, assigned students, Name.  
Course\_d, credit per course, Course credit fee.

Table from 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.

## Course Registration Management System

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.**
20. Faculty\_id, Courses, time, assigned students, Name.
21. Course\_d, credit per course, Course credit fee.
22. Department\_id, Assigned faculty, s\_info, course\_id, class\_info.
23. Department\_id, Assigned course. **-Composite PK.**
24. S\_info, Details, Name.
25. Course\_id, time, Number of students, assigned student details, Name.
26. Class\_info, room number.
27. Class\_info, Sections. **-Composite PK.**
28. Department\_id, Assigned faculty, s\_info, faculty\_id, course\_d
29. Department\_id, Assigned course. **-Composite PK.**
30. s\_info, Details, Name.
31. faculty\_id, Courses, time, assigned students, Name.
32. 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.

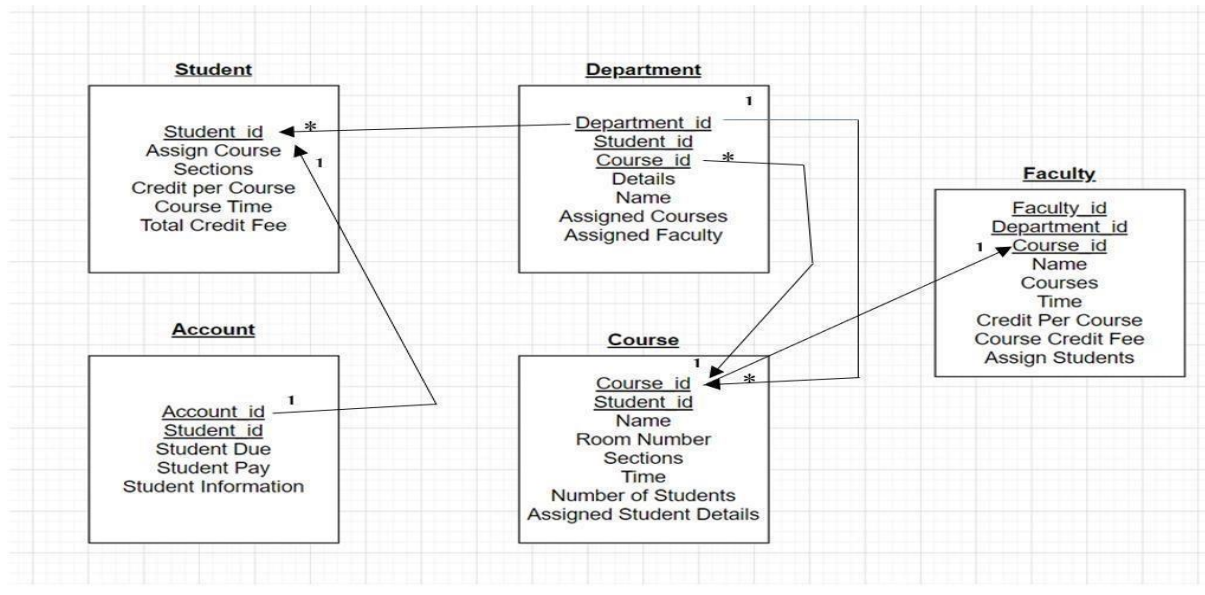
**9.**

Faculty\_id, Courses, time, assigned students, Name. **10.**

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

# Course Registration Management System

## Schema Diagram:

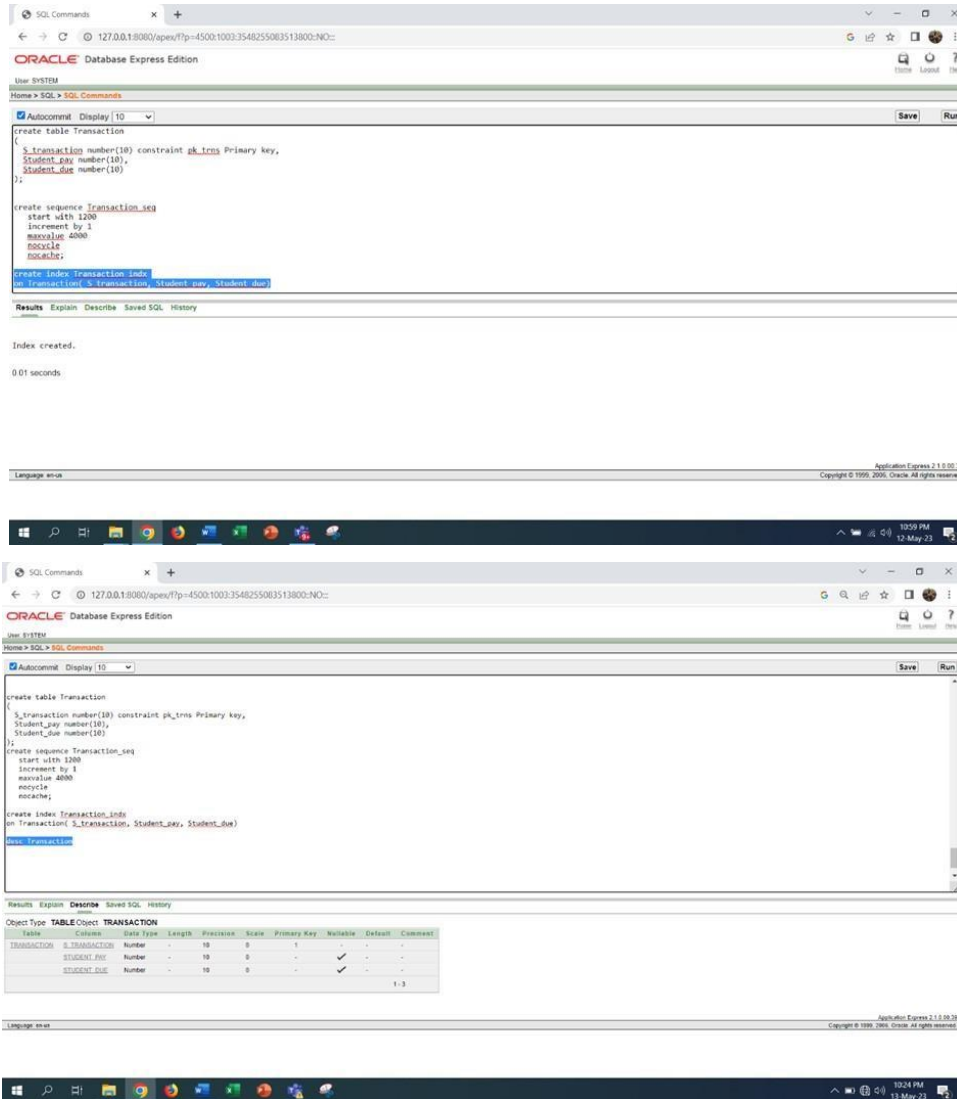


## Table Creation:

### Table Transaction

```
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



## 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

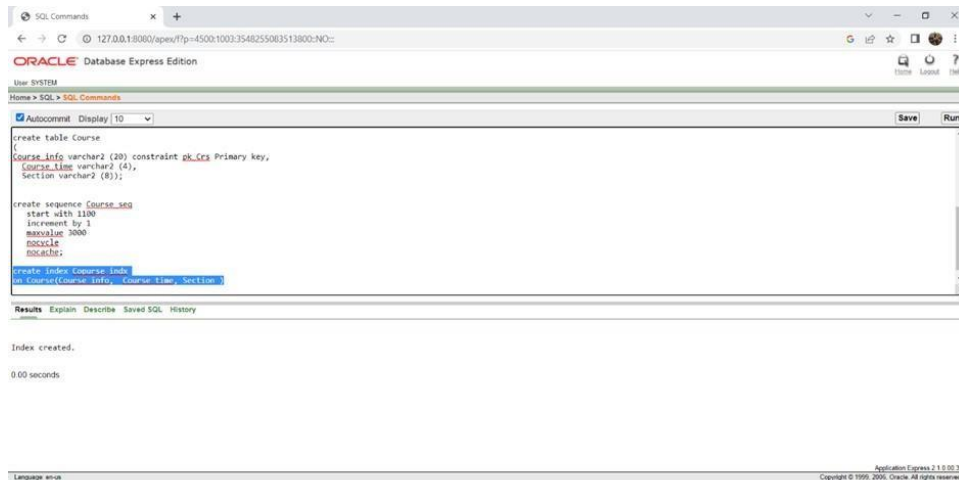
nocache; create index

# Course Registration Management System

Copurse\_idx on

Course(Course\_info,

Course\_time, Section )



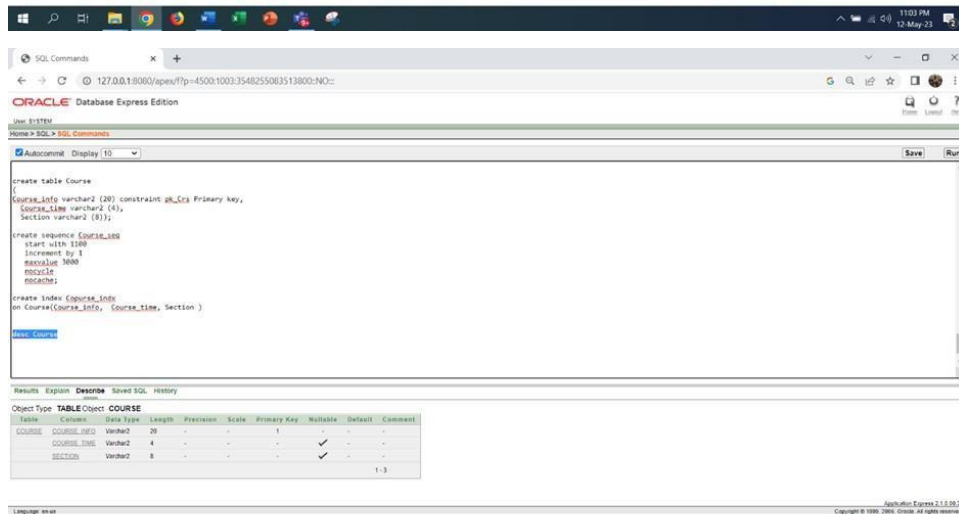
```
create table Course
(
  Course_info varchar2 (20) constraint pk_Crs Primary key,
  Course_time varchar2 (4),
  Section varchar2 (8));

create sequence Course_seq
start with 1000
increment by 1
maxvalue 3000
nocycle
nolatch;

create index Copurse_idx
on Course(Course_info, Course_time, Section );
```

Index created.

0.00 seconds



```
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 1000
increment by 1
maxvalue 3000
nocycle
nolatch;

create index Copurse_idx
on Course(Course_info, Course_time, Section );
```

Object Type: TABLE Object: COURSE

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
COURSE	COURSE_SEQ	Number	20			1	-	-	-
	COURSE_TIME	Varchar2	4			-	✓	-	-
	SECTION	Varchar2	8			-	✓	-	-

## Table Student

create table Student

(

S\_info varchar2 (20) constraint pk\_std Primary key,

Details varchar2 (25), Name

varchar2 (10)); create

sequence Student\_seq start

# Course Registration Management System

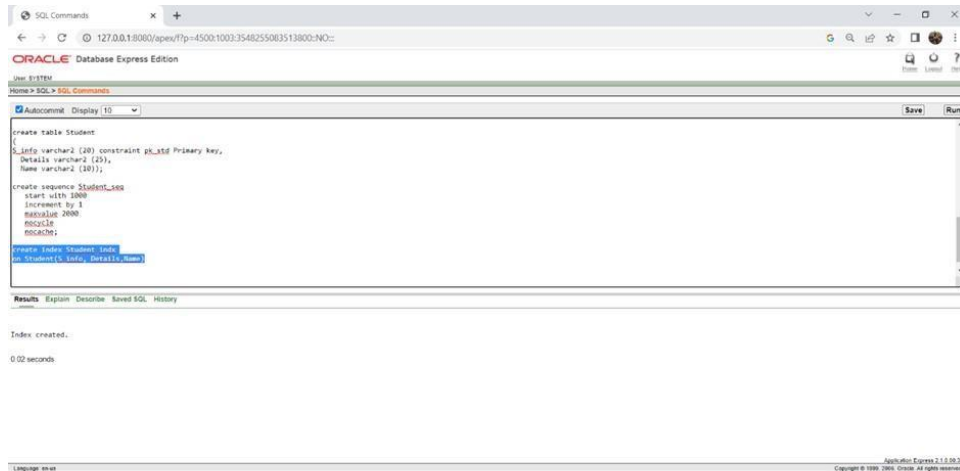
with 1000 increment by 1

maxvalue 2000 nocycle

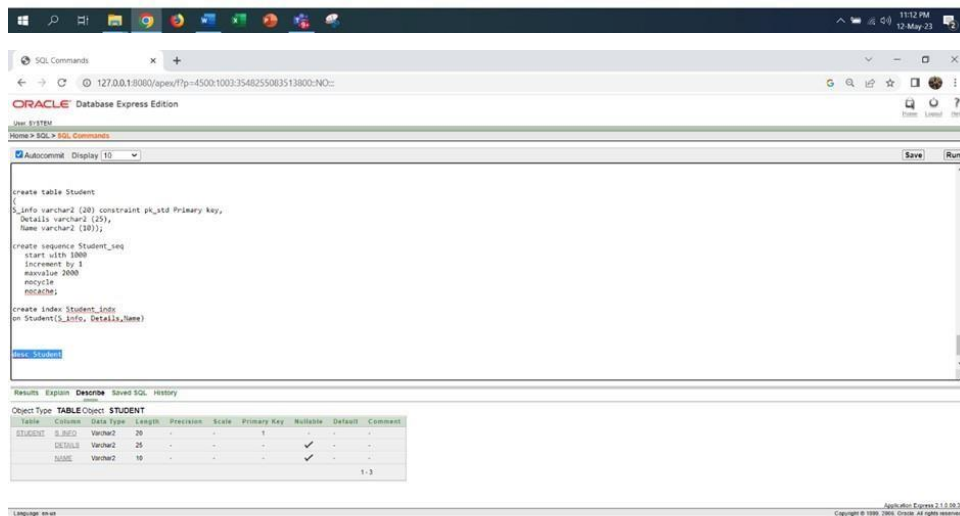
nocache; create index

Student\_indx on Student(S\_info,

Details,Name)



```
SQL Commands
127.0.0.1:8080/apex/f?p=4500:1002:3548255063513800::NO::
Oracle Database Express Edition
User: SYS
Home > SQL > SQL Commands
Autocommit: Display 10
Save Run
create table Student
(
  S_info varchar2(20) constraint pk_std Primary key,
  Details varchar2(25),
  Name varchar2(10));
create sequence Student_seq
start with 1000
increment by 1
maxvalue 2000
nocycle
nocache;
create index Student_indx
on Student(S_info, Details, Name);
Results Explain Describe Saved SQL History
Index created.
0.02 seconds
Language: SQL
Application Express 2.1.1.00.30
Copyright © 1996, 2006, Oracle. All rights reserved.
```



```
SQL Commands
127.0.0.1:8080/apex/f?p=4500:1002:3548255063513800::NO::
Oracle Database Express Edition
User: SYS
Home > SQL > SQL Commands
Autocommit: Display 10
Save Run
create table Student
(
  S_info varchar2(20) constraint pk_std Primary key,
  Details varchar2(25),
  Name varchar2(10));
create sequence Student_seq
start with 1000
increment by 1
maxvalue 2000
nocycle
nocache;
create index Student_indx
on Student(S_info, Details, Name);
Results Explain Describe Saved SQL History
Object Type: TABLE Object: STUDENT
Table Columns Data Type Length Precision Scale Primary Key Nullable Default Comment
STUDENT S_INFO Varchar2 20 - - - 1 - -
DETAILS Varchar2 25 - - - - - -
NAME Varchar2 10 - - - - - -
1-3
Language: SQL
Application Express 2.1.1.00.30
Copyright © 1996, 2006, Oracle. All rights reserved.
```

## Table Class

create table Class

(

# Course Registration Management System

```

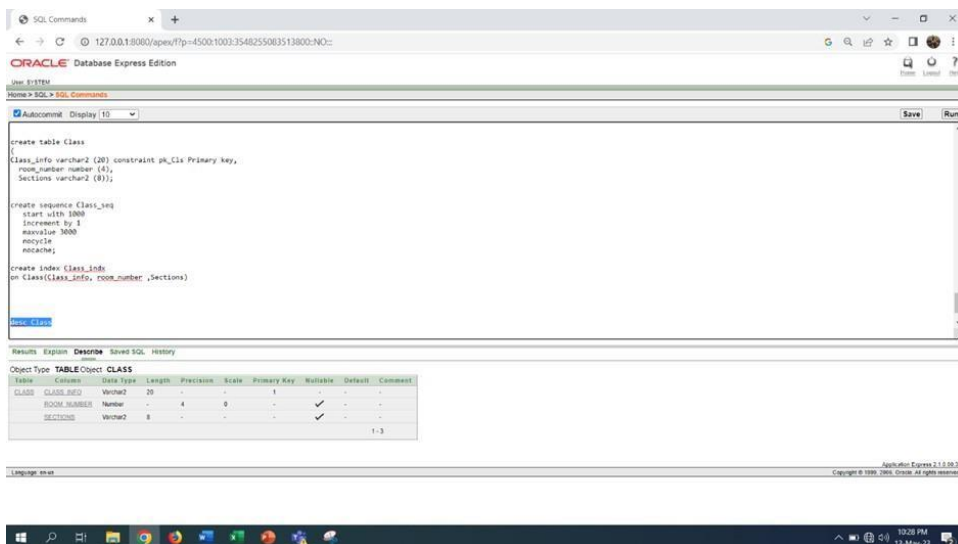
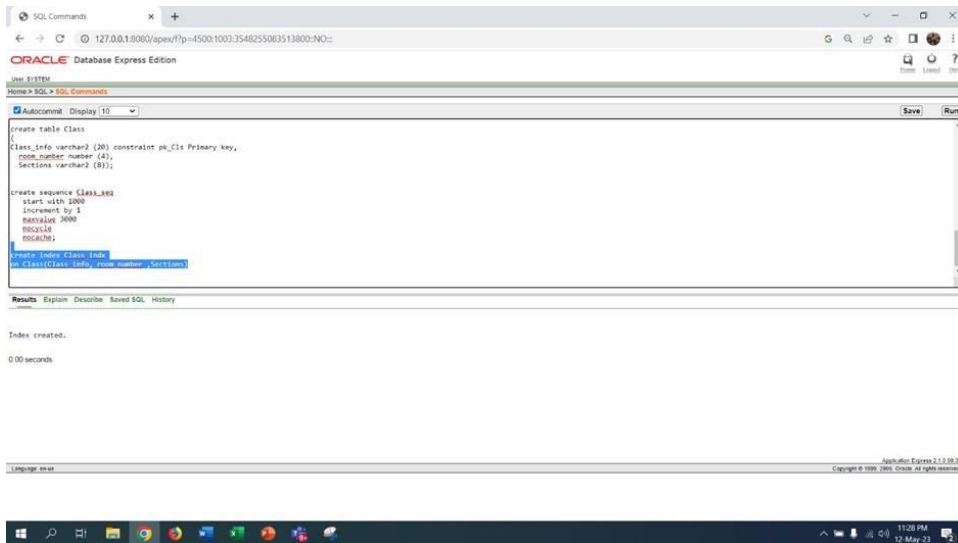
Class_info varchar2 (20) constraint pk_Cls Primary key,
room_number number (4), Sections varchar2 (8)); create
sequence Class_seq  start with 1000  increment by 1

```

```

maxvalue 3000  nocycle  nocache; create
index  Class_indx  on  Class(Class_info,
room_number ,Sections)

```





# Course Registration Management System

## Table DetailsCourse

create table DetailsCourse

(

Course\_d varchar2 (20) constraint pk\_crstdtls Primary key,

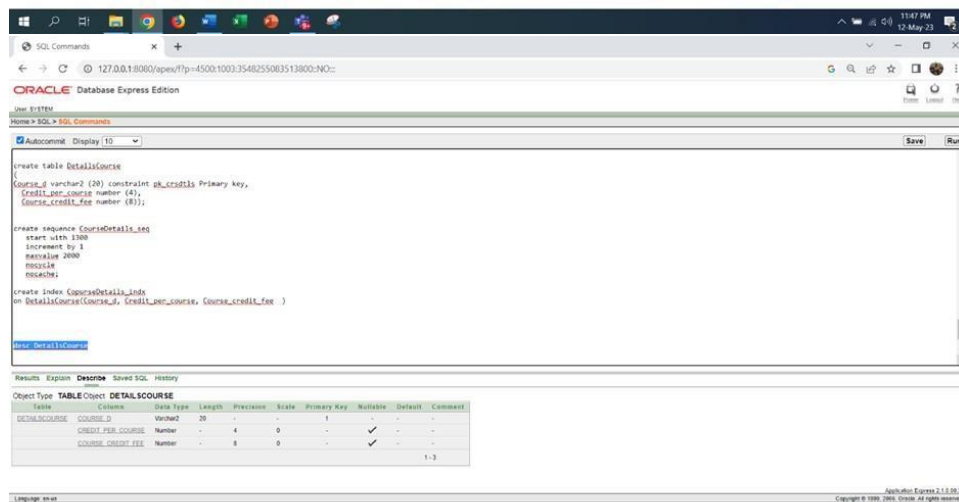
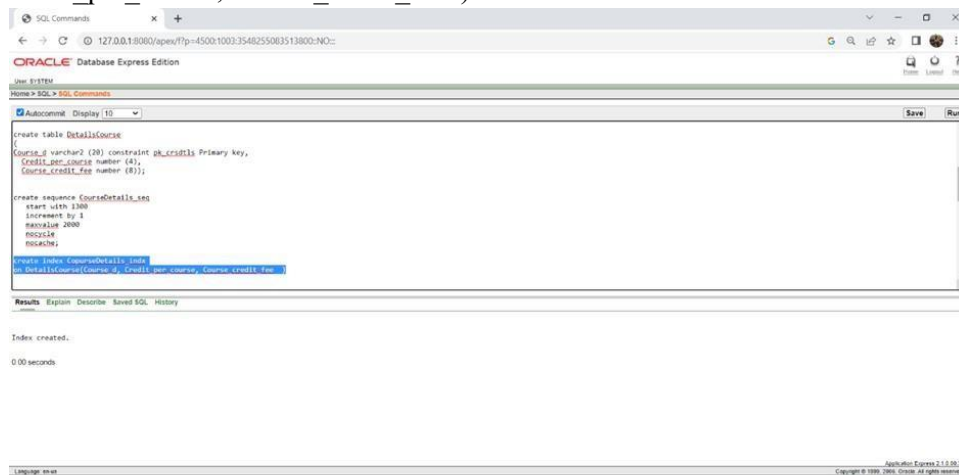
Credit\_per\_course number (4), Course\_credit\_fee number (8)); create

sequence CourseDetails\_seq start with 1300 increment by

1 maxvalue 2000 nocycle nocache; create index

CopurseDetails\_indx on DetailsCourse(Course\_d,

Credit\_per\_course, Course\_credit\_fee )



# Course Registration Management System

## Table Faculty

create table Faculty

(

Faculty\_id number (8) constraint pk\_Flt Primary key,

Courses varchar2 (30),

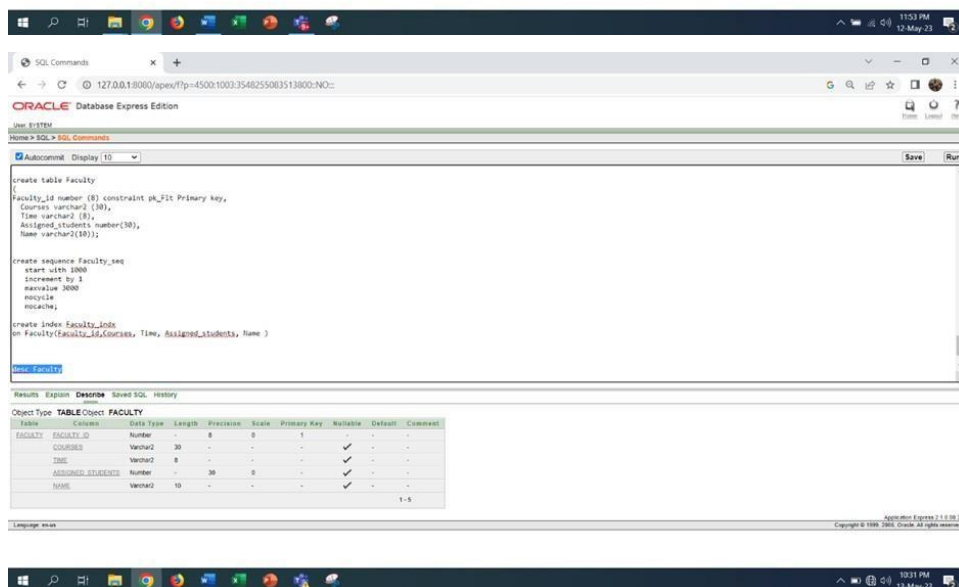
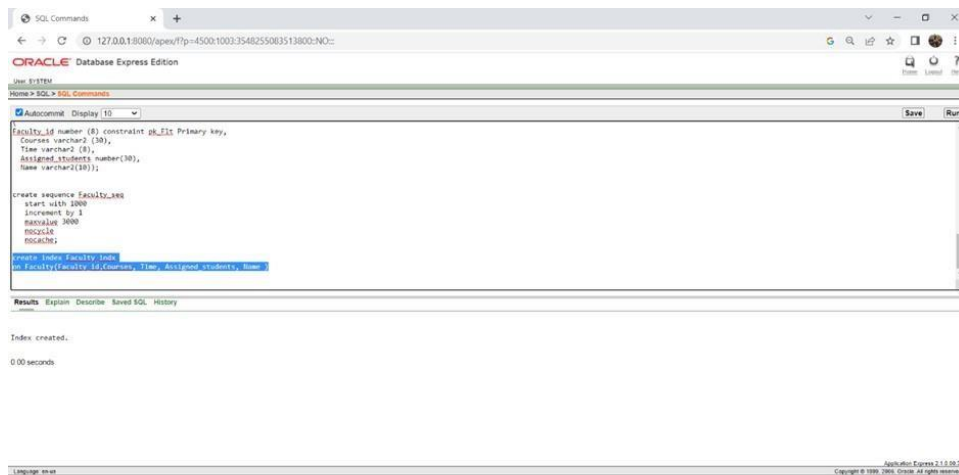
Time varchar2 (8),

Assigned\_students number(30), Name varchar2(10)); 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

## 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

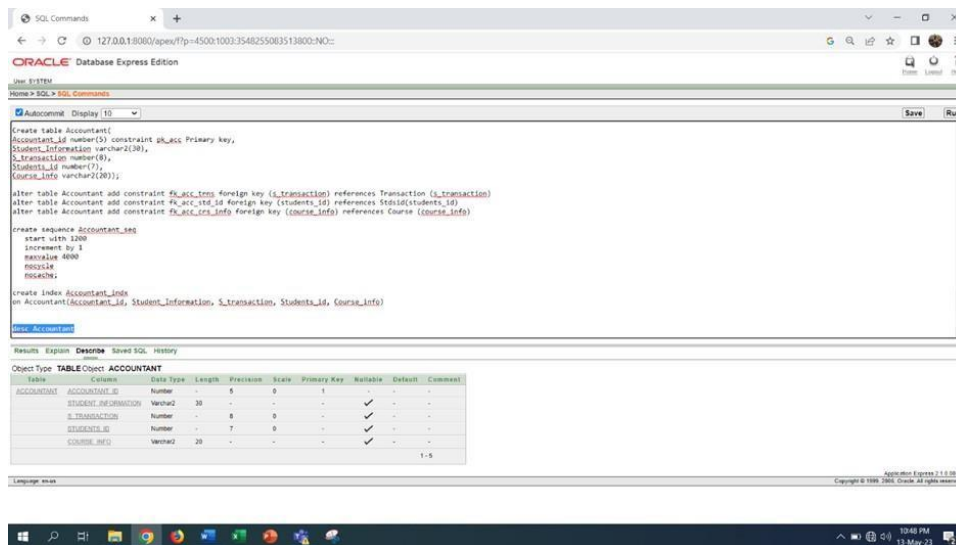
(s\_transaction) alter table Accountant add constraint fk\_acc\_std\_id foreign key (students\_id) references

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 nocache; create index Accountant\_indx on Accountant(Accountant\_id, Student\_Information,

S\_transaction, Students\_id, Course\_info)



## Table Stdsid

create table Stdsid(

Students\_id number(7) constraint pk\_Stds\_id Primary key,

Assigned\_Course varchar (20),

Credit\_per\_course varchar2(4),

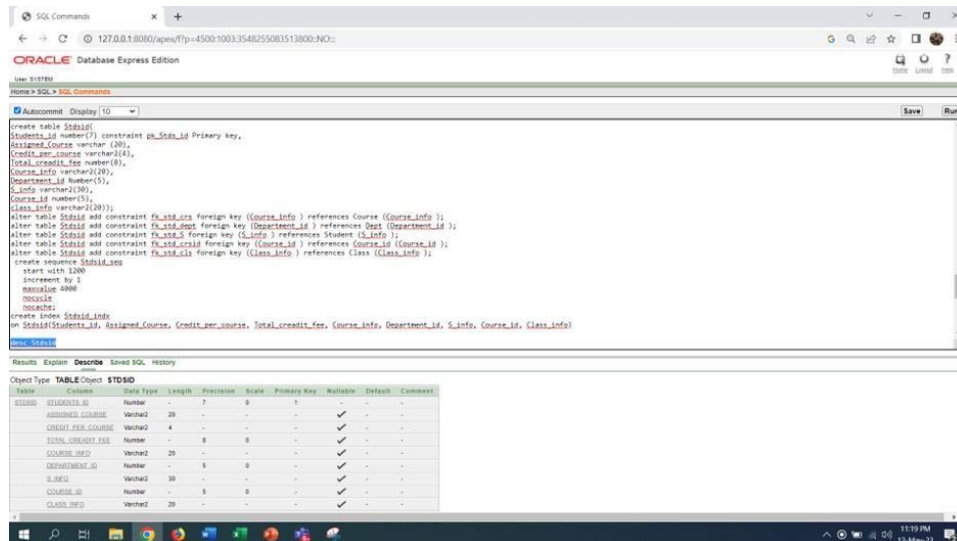
Total\_creadit\_fee number(8),

Course\_info varchar2(20),

# Course Registration Management System

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 ); 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)



## Table Dept

create table Dept(

Department\_id number(5) constraint pk\_dept\_id Primary key,

Assigned\_faculty varchar (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)); alter table Dept add constraint fk\_dept\_std foreign key (S\_info) references Student (S\_info); alter table Dept

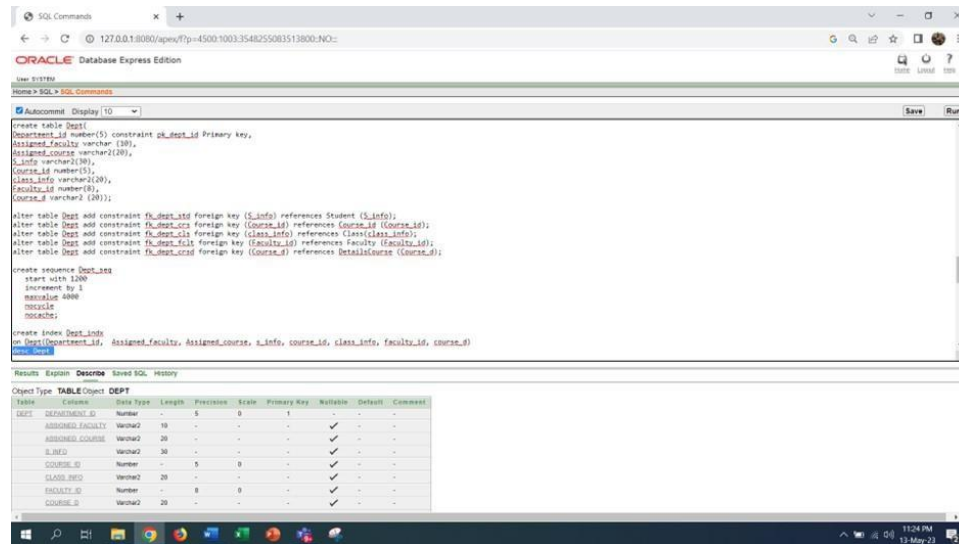
add constraint fk\_dept\_crs foreign key (Course\_id) references Course\_id

(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

# Course Registration Management System

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) desc Dept



## Table Course\_id

create table Course\_id(

Course\_id number(5) constraint pk\_crsid Primary key,

Time varchar (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)

# Course Registration Management System

create sequence Courseid\_seq

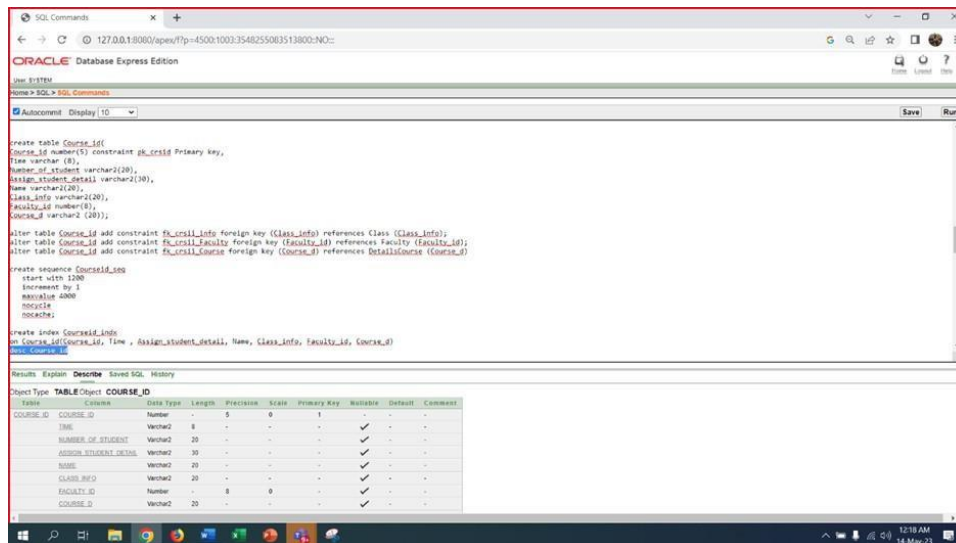
start with 1200 increment by

1 maxvalue 4000 nocycle

nocache;

create index Courseid\_indx

on Course\_id(Course\_id, Time , Assign\_student\_detail, Name, Class\_info, Faculty\_id, Course\_d)



## Data insertion

INSERT INTO Faculty (Faculty\_id, Courses, Time, Assigned\_students, Name)

VALUES (1001, 'Computer Science', '13:00', 30, 'Jane');

INSERT INTO Faculty (Faculty\_id, Courses, Time, Assigned\_students, Name)

VALUES (1002, 'Physics', '11:00', 20, 'Michael');

INSERT INTO Faculty (Faculty\_id, Courses, Time, Assigned\_students, Name)

VALUES (1003, 'History', '09:00', 15, 'Sarah');

INSERT INTO Faculty (Faculty\_id, Courses, Time, Assigned\_students, Name)

VALUES (1004, 'Mathematics', '10:00', 25, 'John');

INSERT INTO Faculty (Faculty\_id, Courses, Time, Assigned\_students, Name)

# Course Registration Management System

VALUES (1005, 'English', '14:00', 18, 'David'); select \* from Faculty;

The screenshot shows a database management interface with a SQL editor and a results pane. The SQL editor contains the following queries:

```
INSERT INTO Faculty (Faculty_id, Courses, Time, Assigned_students, Name)
VALUES (1001, 'Computer Science', '13:00', 30, 'Jane');

INSERT INTO Faculty (Faculty_id, Courses, Time, Assigned_students, Name)
VALUES (1002, 'Physics', '11:00', 20, 'Michael');

INSERT INTO Faculty (Faculty_id, Courses, Time, Assigned_students, Name)
VALUES (1003, 'History', '09:00', 15, 'Sarah');

INSERT INTO Faculty (Faculty_id, Courses, Time, Assigned_students, Name)
VALUES (1004, 'Mathematics', '10:00', 25, 'John');

INSERT INTO Faculty (Faculty_id, Courses, Time, Assigned_students, Name)
VALUES (1005, 'English', '14:00', 18, 'David');

select * from Faculty;
```

The results pane displays a table with 5 rows and 5 columns: FACULTY\_ID, COURSES, TIME, ASSIGNED\_STUDENTS, and NAME. The data is as follows:

FACULTY_ID	COURSES	TIME	ASSIGNED_STUDENTS	NAME
1001	Computer Science	13:00	30	Jane
1002	Physics	11:00	20	Michael
1003	History	09:00	15	Sarah
1004	Mathematics	10:00	25	John
1005	English	14:00	18	David

Below the table, it says "5 rows returned in 0.00 seconds" and "CSV Export".

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

The screenshot shows the SQL Developer interface with the following SQL commands executed:

```
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;
```

The Results pane shows the following data:

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

```
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;
```

The screenshot shows the SQL Developer interface with the following SQL commands executed:

```
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;
```

The Results pane shows the following data:

COURSE_D	CREDIT_PER_COURSE	COURSE_CREDIT_FEE
CSC101	3	1500
CSC102	3	1500
CSC103	3	1500
CSC104	3	1500
CSC105	3	1500

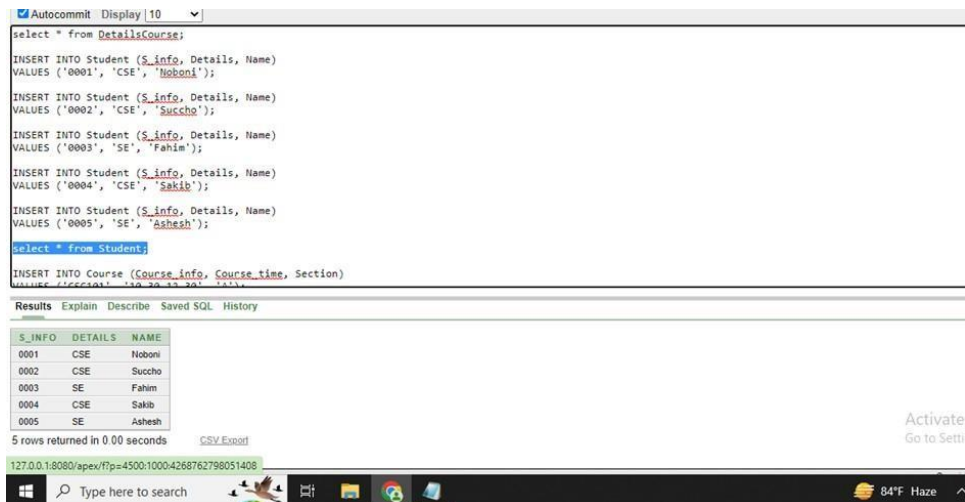
5 rows returned in 0.00 seconds

```
INSERT INTO Student (S_info, Details, Name)
```



# Course Registration Management System

```
VALUES ('0001', 'CSE', 'Noboni');
INSERT INTO Student (S_info, Details, Name)
VALUES ('0002', 'CSE', 'Succcho');
INSERT INTO Student (S_info, Details, Name)
VALUES ('0003', 'SE', 'Fahim');
INSERT INTO Student (S_info, Details, Name)
VALUES ('0004', 'CSE', 'Sakib');
INSERT INTO Student (S_info, Details, Name)
VALUES ('0005', 'SE', 'Ashesh'); select * from
Student;
```



```
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)
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;
```

# Course Registration Management System

Home > SQL > SQL Commands

☒ Autocommit Display 10

```
select * from Student;

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)
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;
```

Results Explain Describe Saved SQL History

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

5 rows returned in 0.00 seconds CSV Export

Activate Win 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;
```

☒ 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

127.0.0.1:8080/apex/?p=4500:1000:4268762798051406

Activate Win Go to Settings

# Course Registration Management System

```
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;

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

```
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');
```

# Course Registration Management System

```
INSERT INTO Dept (Department_id, Assigned_faculty, Assigned_course, S_info, Course_id, class_info, Faculty_id, Course_d)
```

```
VALUES (103, 'FST', 'Microprozessor', '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');
```

Autocommit Display 10

```

INSERT INTO Dept (Department_id, Assigned_faculty, Assigned_course, S_info, Course_id, class_info, Faculty_id, Course_d)
VALUES (103, 'FST', 'Microprozessor', '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');

select * from Dept;

```

Results Explain Describe Saved SQL History

DEPARTMENT_ID	ASSIGNED_FACULTY	ASSIGNED_COURSE	S_INFO	COURSE_ID	CLASS_INFO	FACULTY_ID	COURSE_D
103	FST	Microprozessor	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 Stdssid (Students_id, Assigned_Course, Credit_per_course, Total_creadit_fee, Course_info, Department_id, S_info, Course_id, Class_info)
```

```
VALUES (Stdssid_seq.NEXTVAL, 'Computer Science', '3', 15, 'CSC103', 104, '0002', 1204, 'Class103');
```

```
INSERT INTO Stdssid (Students_id, Assigned_Course, Credit_per_course, Total_creadit_fee, Course_info, Department_id, S_info, Course_id, Class_info)
```

```
VALUES (Stdssid_seq.NEXTVAL, 'OOP2', '3', 18, 'CSC102', 103, '0003', 1201, 'Class100');
```

```
INSERT INTO Stdssid (Students_id, Assigned_Course, Credit_per_course, Total_creadit_fee, Course_info, Department_id, S_info, Course_id, Class_info)
```

```
VALUES (Stdssid_seq.NEXTVAL, 'Compailor', '3', 12, 'CSC105', 102, '0001', 1202, 'Class104');
```

```
INSERT INTO Stdssid (Students_id, Assigned_Course, Credit_per_course, Total_creadit_fee, Course_info, Department_id, S_info, Course_id, Class_info)
```

```
VALUES (Stdssid_seq.NEXTVAL, 'Database', '3', 09, 'CSC101', 105, '0005', 1203, 'Class101');
```

```
INSERT INTO Stdssid (Students_id, Assigned_Course, Credit_per_course, Total_creadit_fee, Course_info, Department_id, S_info, Course_id, Class_info)
```

```
VALUES (Stdssid_seq.NEXTVAL, 'Computer network', '3', 12, 'CSC104', 101, '0004', 1205, 'Class102');
```

```
* from stdssid;
```

# Course Registration Management System

Home > SQL > SQL Commands

Autocommit Display 10

```

INSERT INTO Stdsid (Students_id, Assigned_Course, Credit_per_course, Total_credit_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_credit_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_credit_fee, Course_info, Department_id, S_info, Course_id, Class_info)
VALUES (Stdsid_seq.NEXTVAL, 'Compallor', '3', 12, 'CSC105', 102, '0001', 1202, 'Class104');
INSERT INTO Stdsid (Students_id, Assigned_Course, Credit_per_course, Total_credit_fee, Course_info, Department_id, S_info, Course_id, Class_info)
VALUES (Stdsid_seq.NEXTVAL, 'Database', '3', 09, 'CSC101', 105, '0005', 1203, 'Class101');
INSERT INTO Stdsid (Students_id, Assigned_Course, Credit_per_course, Total_credit_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;
    
```

Results Explain Describe Saved SQL History

STUDENTS_ID	ASSIGNED_COURSE	CREDIT_PER_COURSE	TOTAL_CREDIT_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	Compallor	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;

Home > SQL > SQL Commands

Autocommit Display 10

```

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;
    
```

Results Explain Describe Saved SQL History

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

# Course Registration Management System

## SQL -Query

### single-row function -3

1. Select concat (room\_number,sections)

From Class

User: SCOTT

Home > SQL > **SQL Commands**

☒ Autocommit Display 10 ▼

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

**Results** Explain Describe Saved SQL History

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

5 rows returned in 0.06 seconds [CSV Export](#)

2. Select name length(name)

From DetailsCourse

User: SCOTT

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](#)



# Course Registration Management System

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

From stdsid

The screenshot shows the Oracle SQL Developer interface. At the top, there's a breadcrumb 'Home > SQL > SQL Commands'. Below it, a toolbar has 'Autocommit' checked and 'Display' set to 10. The SQL editor contains the query: `Select Assigned course,instr (Assigned course, 'C')` and `From stdsid;`. Below the editor, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is active, displaying a table with two columns: 'ASSIGNED\_COURSE' and 'INSTR(ASSIGNED\_COURSE,'C')'. The table contains five rows of data. At the bottom, it says '5 rows returned in 0.01 seconds' and provides a 'CSV Export' link.

ASSIGNED_COURSE	INSTR(ASSIGNED_COURSE,'C')
Computer Science	1
OOP2	0
Compailor	1
Database	0
Computer network	1

## Group function -3

1. select max(Student\_due) from

Transaction

The screenshot shows the Oracle SQL Developer interface. At the top, there's a breadcrumb 'Home > SQL > SQL Commands'. Below it, a toolbar has 'Autocommit' checked and 'Display' set to 10. The SQL editor contains the query: `select max(Student_due) from Transaction;`. Below the editor, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is active, displaying a table with one column: 'MAX(STUDENT\_DUE)'. The table contains one row with the value 1500. At the bottom, it says '1 rows returned in 0.06 seconds' and provides a 'CSV Export' link.

MAX(STUDENT_DUE)
1500

2. select min(Student\_pay) from

# Course Registration Management System

Transaction

ORACLE® Database Express Edition

User: SCOTT

Home > SQL > SQL Commands

☒ Autocommit Display 10

```
select min(Student_pay) from Transaction;
```

Results Explain Describe Saved SQL History

MIN(STUDENT_PAY)
1000

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

3. select max(Assigned\_students) from

Faculty group by Faculty\_id

Home > SQL > SQL Commands

☒ Autocommit Display 10

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

Results Explain Describe Saved SQL History

MAX(ASSIGNED_STUDENTS)
30
20
15
25
18

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

## Subquery -3

1. Select name From Student

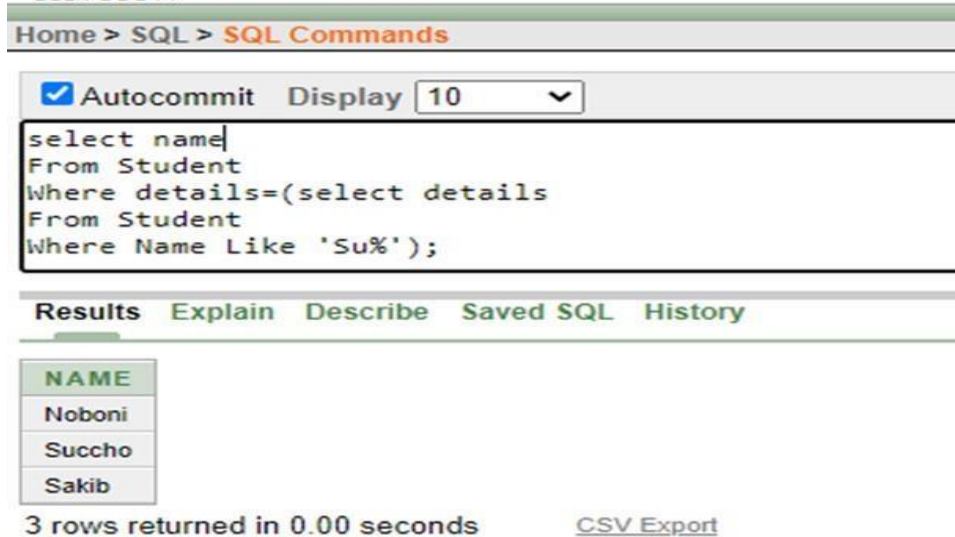


# Course Registration Management System

Where details=(select details

From Student

Where Name Like 'Su%')



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

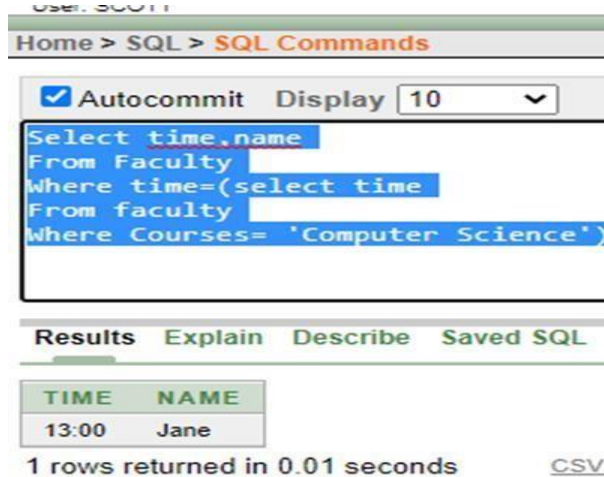
3 rows returned in 0.00 seconds [CSV Export](#)

2. 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](#)

3. Select name,assigned\_students

From faculty

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

From Faculty)

# Course Registration Management System

Home > SQL > SQL Commands

☒ Autocommit Display 10

```
Select name,assigned_students  
From faculty  
Where assigned_students=(select min(assigned_students)  
From Faculty);
```

Results Explain Describe Saved SQL History

NAME	ASSIGNED_STUDENTS
Sarah	15

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

## Joining -3

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;
```

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 course room number and faculty .

Ans :

```
SELECT Course_id.Course_d, Course_id.Class_info, Dept.Assigned_faculty
```

```
FROM Course_id
```

# Course Registration Management System

JOIN Dept

ON Course\_id.Course\_d = Dept.Course\_d;

User: SCOTT

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

JOIN Stdsid s

ON d.COURSE\_D = s.COURSE\_INFO

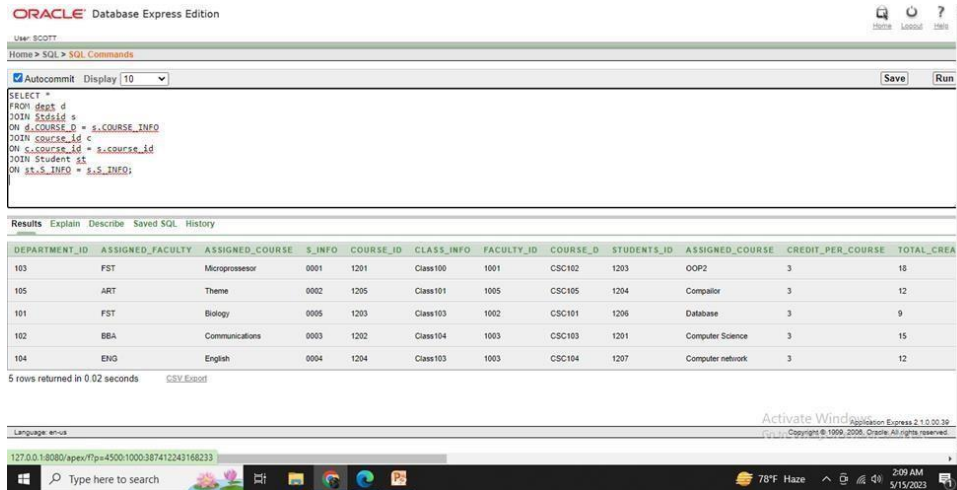
JOIN course\_id c

ON c.course\_id = s.course\_id

JOIN Student st

ON st.S\_INFO = s.S\_INFO;

# Course Registration Management System



Oracle Database Express Edition interface showing a SQL query and its results. The query is a join between department, student, and course tables. The results table shows 5 rows of 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_CREA
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

## view -3

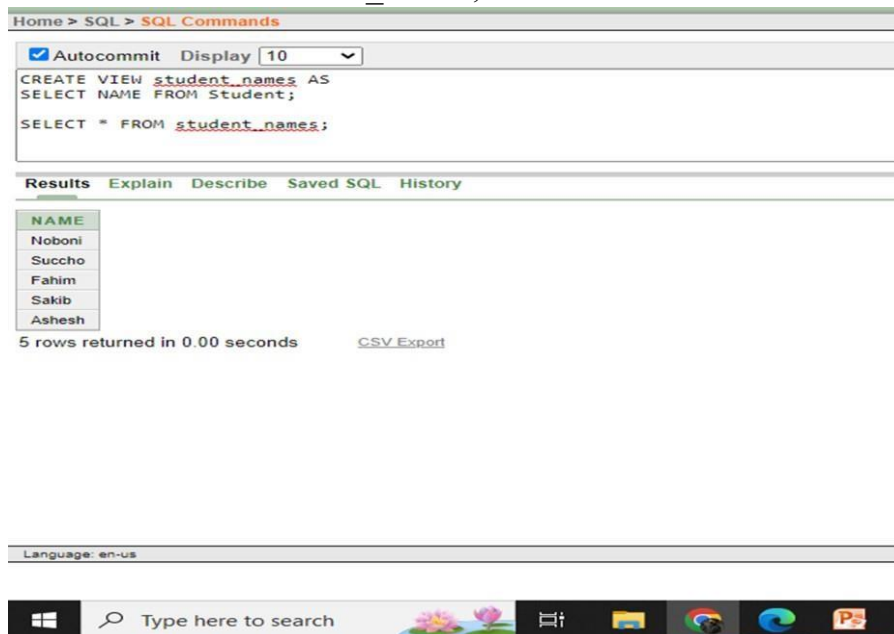
1. Create a view to get student names

Ans :

CREATE VIEW student\_names AS

SELECT NAME FROM Student;

SELECT \* FROM student\_names;



Oracle Database Express Edition interface showing the creation and execution of a view. The SQL commands are: CREATE VIEW student\_names AS SELECT NAME FROM Student; and SELECT \* FROM student\_names;. The results table shows 5 rows of student names.

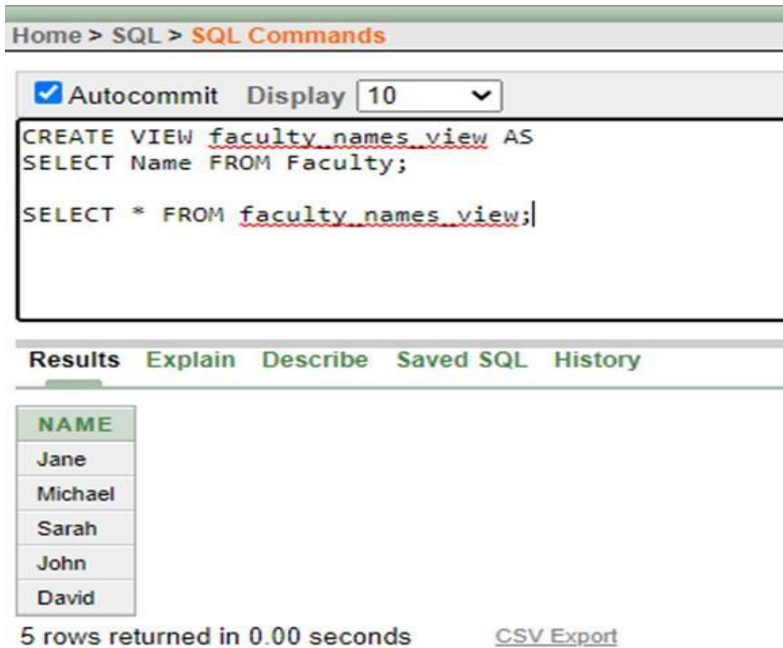
NAME
Noboni
Succho
Fahim
Sakib
Ashesh

# Course Registration Management System

## 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 a web-based SQL interface. At the top, there's a breadcrumb trail: "Home > SQL > SQL Commands". Below this, there's a toolbar with a checked "Autocommit" checkbox and a "Display" dropdown set to "10". The main text area contains the following SQL commands:

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

Below the text area, there's a row of tabs: "Results", "Explain", "Describe", "Saved SQL", and "History". The "Results" tab is selected, showing a table with one column, "NAME", and five rows of data:

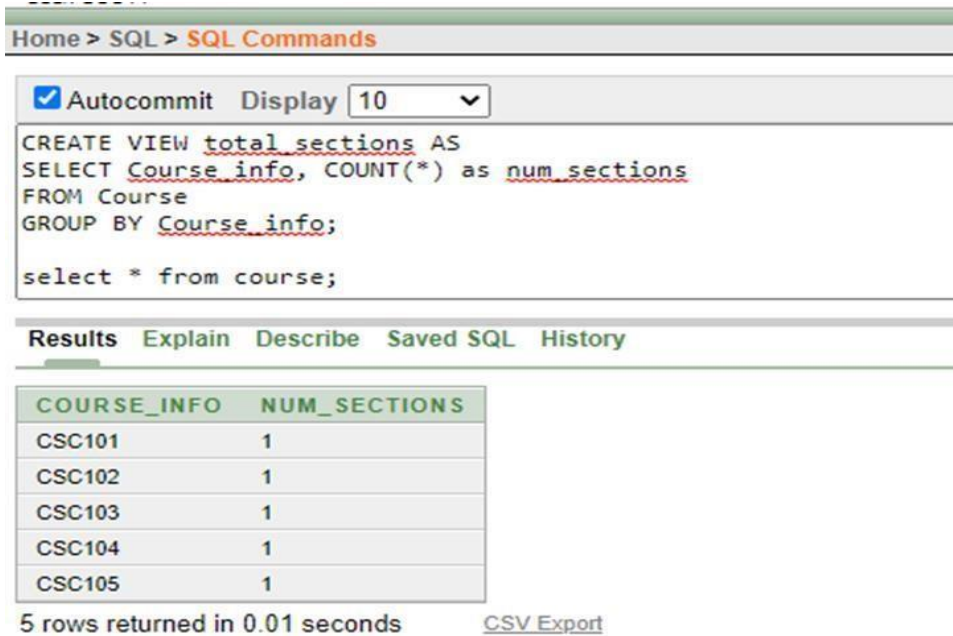
NAME
Jane
Michael
Sarah
John
David

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

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

Ans : CREATE VIEW total\_sections AS  
SELECT Course\_info, COUNT(\*) as num\_sections  
FROM Course  
GROUP BY Course\_info;  
select \* from course;

# Course Registration Management System



The screenshot shows an SQL command window with the following content:

```
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 SYNONYM Details FOR DetailsCourse;
- 2  
CREATE SYNONYM Stdsid\_syn  
FOR Stdsid;
3.  
CREATE SYNONYM Syn\_Dept FOR Dept;

## PL/SQL -3 function

1. Write a function to find the maximum credit of a student Ans  
:  
CREATE OR REPLACE FUNCTION find\_max\_credit  
RETURN NUMBER

# Course Registration Management System

```
IS max_credit
NUMBER;
BEGIN
    SELECT MAX(Total_credit_fee)
    INTO max_credit
    FROM StdSID;

    DBMS_OUTPUT.PUT_LINE('The maximum credit is: ' || max_credit);

    RETURN max_credit;
END;

DECLARE          max_credit
NUMBER; BEGIN    max_credit
:= find_max_credit();
END;
```

The screenshot shows the Oracle SQL Developer interface. The top bar indicates 'Home > SQL > SQL Commands'. Below this, there's a toolbar with 'Autocommit' checked and 'Display' set to 10. The main text area contains the following PL/SQL code:

```
CREATE OR REPLACE FUNCTION find_max_credit
RETURN NUMBER
IS
    max_credit NUMBER;
BEGIN
    SELECT MAX(Total_credit_fee)
    INTO max_credit
    FROM StdSID;

    DBMS_OUTPUT.PUT_LINE('The maximum credit is: ' || max_credit);

    RETURN max_credit;
END;

DECLARE
    max_credit NUMBER;
BEGIN
    max_credit := find_max_credit();
END;
```

Below the code editor, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is active, showing the output of the execution:

```
The maximum credit is: 18
Statement processed.
0.00 seconds
```

At the bottom, there's a taskbar with several open documents: 'Non EQUERY.docx', 'Relational Algebra.docx', and 'project DB.docx'. The Windows taskbar is visible at the very bottom.

2. Write a Function to find student with minimum credit Ans

# Course Registration Management System

```
:  
CREATE OR REPLACE FUNCTION find_student_min_credit RETURN VARCHAR2 IS  
v_student_id VARCHAR2(10); v_min_credit NUMBER;  
BEGIN  
    SELECT Students_id, Total_creadit_fee  
    INTO v_student_id, v_min_credit  
    FROM stdsid  
    WHERE Total_creadit_fee = (SELECT MIN(Total_creadit_fee) FROM stdsid);  
  
    RETURN 'Student with minimum credit: ' || v_student_id || ' (' || v_min_credit || ')';  
END;  
  
DECLARE                                v_output  
VARCHAR2(100); BEGIN                v_output  
:= find_student_min_credit();  
    DBMS_OUTPUT.PUT_LINE(v_output);  
END;
```

```
CREATE OR REPLACE FUNCTION find_student_min_credit RETURN VARCHAR2 IS  
v_student_id VARCHAR2(10);  
v_min_credit NUMBER;  
BEGIN  
    SELECT Students_id, Total_creadit_fee  
    INTO v_student_id, v_min_credit  
    FROM stdsid  
    WHERE Total_creadit_fee = (SELECT MIN(Total_creadit_fee) FROM stdsid);  
  
    RETURN 'Student with minimum credit: ' || v_student_id || ' (' || v_min_credit || ')';  
END;  
  
DECLARE  
v_output VARCHAR2(100);  
BEGIN  
v_output := find_student_min_credit();  
DBMS_OUTPUT.PUT_LINE(v_output);  
END;
```

Results Explain Describe Saved SQL History

Student with minimum credit: 1206 (9)

Statement processed.

0.00 seconds

127.0.0.1:8080/apex/f?p=4500:1000:2593007992652739

Non EQUERY.docx

Relational Algebra.docx

project DB.docx



3. Write a function to find the maximum Studnet due.

Ans :

```
CREATE OR REPLACE FUNCTION find_max_student_due RETURN NUMBER IS  
max_due NUMBER(10);  
BEGIN
```



# Course Registration Management System

```
SELECT MAX(Student_due)
INTO max_due
FROM Transaction;
DBMS_OUTPUT.PUT_LINE('The maximum Student_due is: ' || max_due); RETURN
max_due;
END;
```

```
Declare a number; Begin a :=
find_max_student_due();
dbms_output.put_line(a); end;
```

The screenshot displays the Oracle SQL Developer environment. At the top, the 'SQL Commands' tab is active. A PL/SQL script is entered in the main editor, which includes a function definition and a block to execute it. The script is as follows:

```
CREATE OR REPLACE FUNCTION find_max_student_due RETURN NUMBER IS
max_due NUMBER(10);
BEGIN
SELECT MAX(Student_due)
INTO max_due
FROM Transaction;
DBMS_OUTPUT.PUT_LINE('The maximum Student_due is: ' || max_due);
RETURN max_due;
END;

Declare
a number;
Begin
a := find_max_student_due();
dbms_output.put_line(a);
end;
```

Below the editor, the 'Results' tab shows the output of the function call. A table with one row is displayed:

MAX(STUDENT_DUE)
1500

Below the table, it states '1 rows returned in 0.03 seconds' and provides a 'CSV Export' link. The bottom of the screenshot shows the Windows taskbar with various application icons and the system clock.

## -3 procedure

1. Find the faculty name with only one letter containing 'L'.

Ans :

```
CREATE OR REPLACE PROCEDURE find_faculty_with_one_l
IS
v_name Faculty.Name%TYPE;
BEGIN
-- Loop through each record in the Faculty table
```

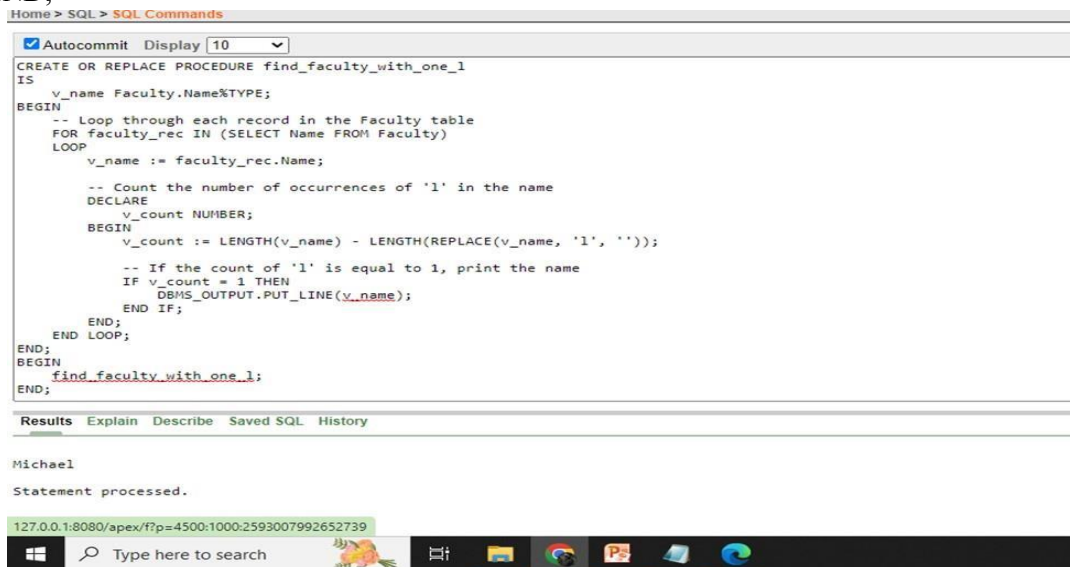
# Course Registration Management System

```
FOR faculty_rec IN (SELECT Name FROM Faculty) LOOP
    v_name := faculty_rec.Name;

    -- Count the number of occurrences of 'l' in the name
    DECLARE          v_count NUMBER;          BEGIN

        v_count := LENGTH(v_name) - LENGTH(REPLACE(v_name, 'l', ''));

        -- If the count of 'l' is equal to 1, print the name
    IF v_count = 1 THEN
        DBMS_OUTPUT.PUT_LINE(v_name);
    END IF;
    END;
END LOOP;
END;
BEGIN
    find_faculty_with_one_l;
END;
```



The screenshot shows a web-based SQL interface. At the top, there's a header "Home > SQL > SQL Commands". Below it, a toolbar includes "Autocommit" (checked), "Display" (set to 10), and a dropdown menu. The main text area contains the PL/SQL code for the procedure `find_faculty_with_one_l`, which iterates through the `Faculty` table, counts the occurrences of the letter 'l' in each name, and prints the name if the count is exactly 1. The code is syntactically highlighted. Below the code area, there are tabs for "Results", "Explain", "Describe", "Saved SQL", and "History". The "Results" tab is active, showing the name "Michael" and the message "Statement processed." Below this, a status bar displays the URL "127.0.0.1:8080/apex/?p=4500:1000:2593007992652739". At the bottom, there is a Windows taskbar with a search bar and several application icons.

## 2. Find the Class with maximum Roomnumber.

Ans ;

```
CREATE OR REPLACE PROCEDURE find_max_room_class
IS
    max_room_num NUMBER(4);
    max_room_class VARCHAR2(20);
BEGIN
    SELECT MAX(ROOM_NUMBER) INTO max_room_num FROM Class;
```

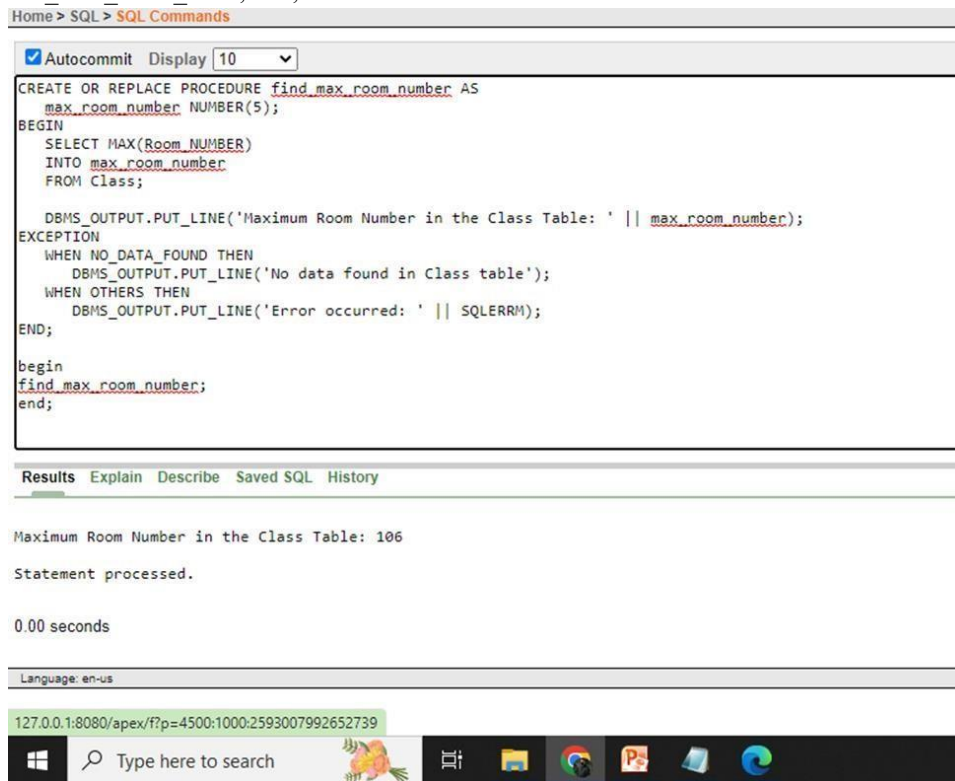
# Course Registration Management System

```
SELECT CLASS_INFO INTO max_room_class FROM Class WHERE ROOM_NUMBER = max_room_num;
```

```
DBMS_OUTPUT.PUT_LINE('The class with the maximum room number is ' || max_room_class); END;
```

begin

find\_max\_room\_class; end;



```
Home > SQL > SQL Commands

Autocommit Display 10

CREATE OR REPLACE PROCEDURE find_max_room_number AS
    max_room_number NUMBER(5);
BEGIN
    SELECT MAX(Room_NUMBER)
    INTO max_room_number
    FROM Class;

    DBMS_OUTPUT.PUT_LINE('Maximum Room Number in the Class Table: ' || max_room_number);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No data found in Class table');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error occurred: ' || SQLERRM);
END;

begin
find_max_room_number;
end;
```

Results Explain Describe Saved SQL History

Maximum Room Number in the Class Table: 106

Statement processed.

0.00 seconds

Language: en-us

127.0.0.1:8080/apex/f?p=4500:1000:2593007992652739

3. Write a procedure to find student who has a letter 'a' in their name.

Ans :

```
CREATE OR REPLACE PROCEDURE find_students_with_a AS student_name
student.Name%TYPE;
```

```
BEGIN
```

```
FOR r IN (SELECT Name FROM student WHERE Name LIKE '%a%') LOOP
student_name := r.Name; dbms_output.put_line(student_name);
```

```
END LOOP;
```

```
END;
```

```
BEGIN
```

## Course Registration Management System

```
find_students_with_a;  
END;
```

```
CREATE OR REPLACE PROCEDURE find_students_with_a AS  
  student_name student.Name%TYPE;  
BEGIN  
  FOR r IN (SELECT Name FROM student WHERE Name LIKE '%a%') LOOP  
    student_name := r.Name;  
    dbms_output.put_line(student_name);  
  END LOOP;  
END;  
  
BEGIN  
  find_students_with_a;  
END;
```

Results Explain Describe Saved SQL History

Fahim  
Sakib

Statement processed.

0.01 seconds

### -3 record -3 cursor

1. Write a cursor to check if student due more than 2000.

Ans :

DECLARE

v\_due Transaction.student\_due%TYPE;

CURSOR c\_transaction IS

SELECT student\_due

FROM Transaction

WHERE student\_due > 2000;

BEGIN

OPEN c\_transaction;

FETCH c\_transaction INTO v\_due;

IF c\_transaction%FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('There are transactions with student\_due more than 2000.');

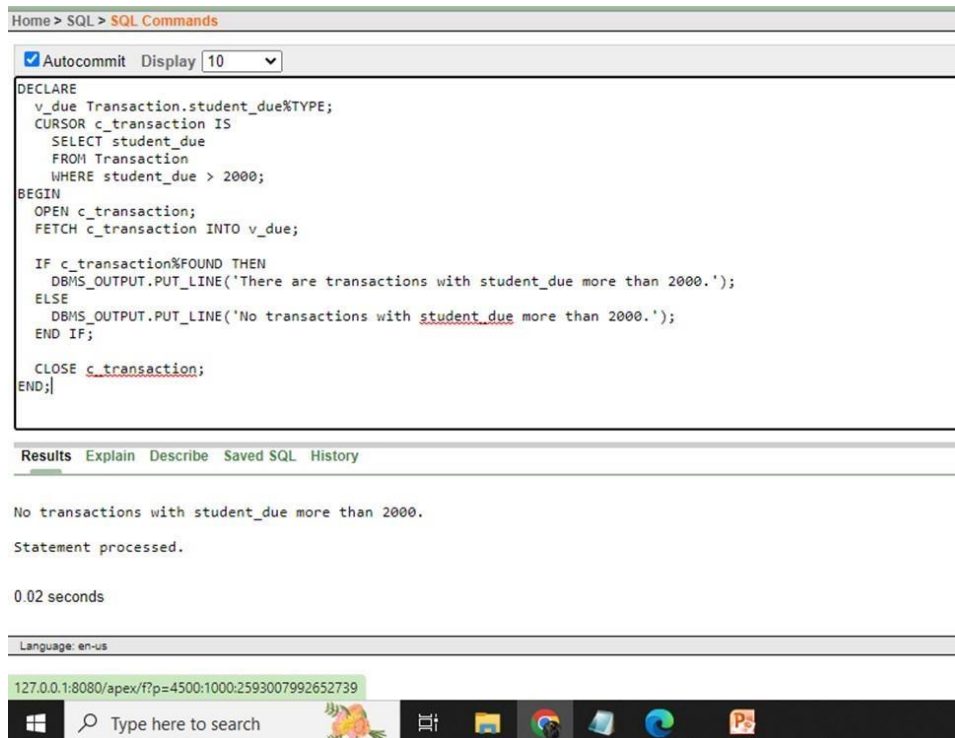
ELSE

DBMS\_OUTPUT.PUT\_LINE('No transactions with student\_due more than 2000.');

END IF;

# Course Registration Management System

```
CLOSE c_transaction;  
END;
```



The screenshot shows the Oracle SQL Developer interface. At the top, there's a breadcrumb 'Home > SQL > SQL Commands'. Below it, a toolbar has 'Autocommit' checked and a 'Display' dropdown set to '10'. The main editor contains a PL/SQL script:

```
DECLARE  
  v_due Transaction.student_due%TYPE;  
  CURSOR c_transaction IS  
    SELECT student_due  
    FROM Transaction  
    WHERE student_due > 2000;  
BEGIN  
  OPEN c_transaction;  
  FETCH c_transaction INTO v_due;  
  
  IF c_transaction%FOUND THEN  
    DBMS_OUTPUT.PUT_LINE('There are transactions with student_due more than 2000.');
```

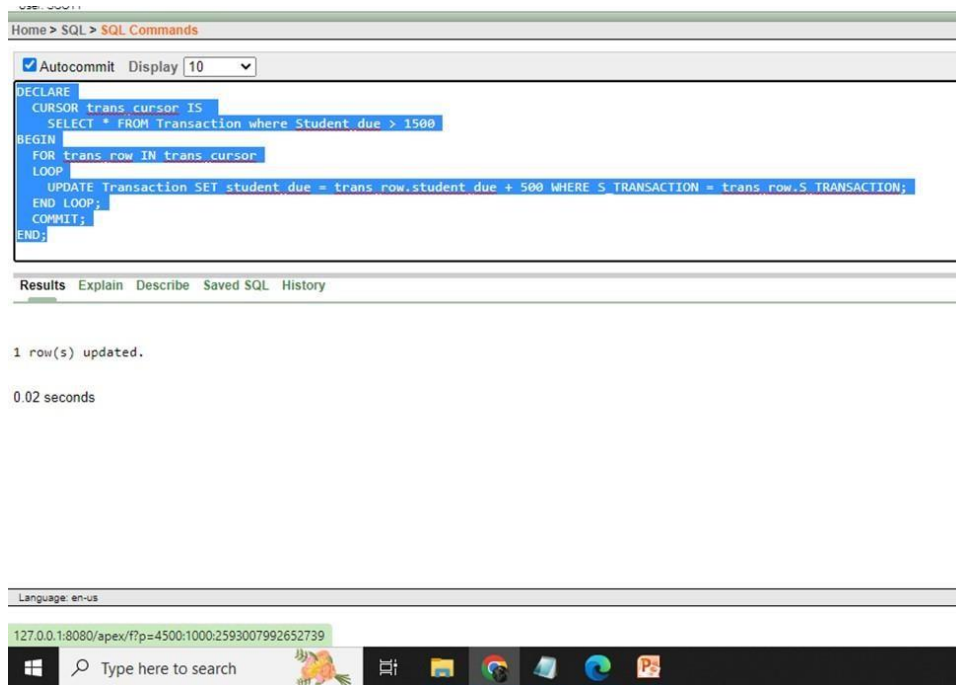
The script continues with an ELSE branch, an END IF statement, and a CLOSE c\_transaction; statement, ending with END;|. Below the editor, a tab bar shows 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is active, displaying the output: 'No transactions with student\_due more than 2000.', 'Statement processed.', and '0.02 seconds'. At the bottom, a status bar shows 'Language: en-us' and a taskbar with various application icons.

2. Write a cursor to update student transactions where students owe more than 1500.

Ans :

```
DECLARE  
  CURSOR trans_cursor IS  
    SELECT * FROM Transaction where Student_due > 1500  
BEGIN  
  FOR trans_row IN trans_cursor  
  LOOP  
    UPDATE Transaction SET student_due = trans_row.student_due + 500 WHERE  
S_TRANSACTION = trans_row.S_TRANSACTION;  
  END LOOP;  
  COMMIT;  
END;
```

# Course Registration Management System

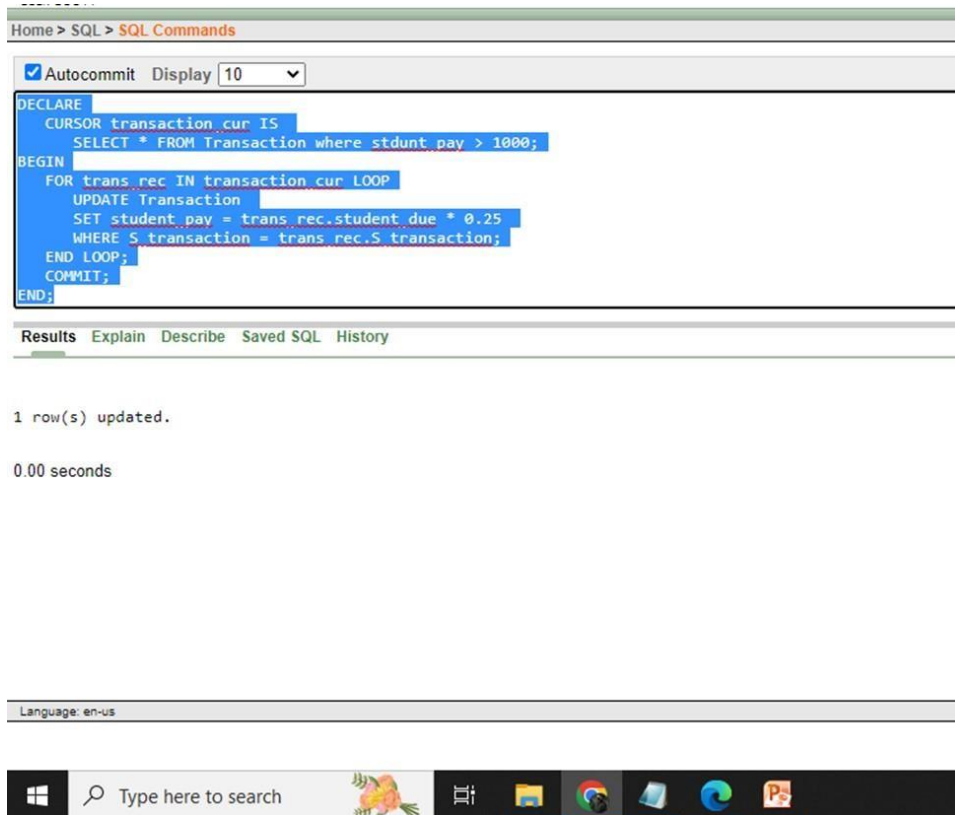


3. Write a cursor to give a 25 percent discount to students who paid more than 1000

Ans :

```
DECLARE
CURSOR transaction_cur IS
SELECT * FROM Transaction where stdunt_pay > 1000; BEGIN
FOR trans_rec IN transaction_cur LOOP
UPDATE Transaction
SET student_pay = trans_rec.student_due * 0.25
WHERE S_transaction = trans_rec.S_transaction;
END LOOP;
COMMIT;
END;
```

# Course Registration Management System



```
Home > SQL > SQL Commands

Autocommit Display 10

DECLARE
  CURSOR transaction_cur IS
    SELECT * FROM Transaction where stdunt_pay > 1000;
BEGIN
  FOR trans_rec IN transaction_cur LOOP
    UPDATE Transaction
      SET student_pay = trans_rec.student_due * 0.25
    WHERE S.transaction = trans_rec.S.transaction;
  END LOOP;
  COMMIT;
END;
```

Results Explain Describe Saved SQL History

1 row(s) updated.

0.00 seconds

Language: en-us

## -3 trigger

1.

```
CREATE OR REPLACE TRIGGER dept_trigger
BEFORE INSERT OR UPDATE OR DELETE ON Dept FOR
EACH ROW BEGIN
  dbms_output.put_line('Trigger fired for Department_id: ' || :OLD.Department_id);
END;
```

2.

```
CREATE OR REPLACE TRIGGER stdsid_trigger
AFTER INSERT ON Stdsid FOR
EACH ROW DECLARE
  v_total_credit_fee Stdsid.Total_creadit_fee%TYPE;
BEGIN
  SELECT SUM(Credit_per_course)
  INTO v_total_credit_fee
  FROM Stdsid
  WHERE Students_id = :NEW.Students_id;
```

## Course Registration Management System

```
UPDATE Stdscid
```

```
SET Total_credit_fee = v_total_credit_fee
```

```
WHERE Students_id = :NEW.Students_id;
```

```
dbms_output.put_line('Total credit fee updated for Students_id: ' || :NEW.Students_id);
```

```
END;
```

```
/
```

3.

```
CREATE OR REPLACE TRIGGER crsdtls_trigger
```

```
BEFORE INSERT OR UPDATE ON DetailsCourse
```

```
FOR EACH ROW
```

```
BEGIN
```

```
:NEW.Course_credit_fee := :NEW.Credit_per_course * 1000;
```

```
dbms_output.put_line('Course credit fee calculated for Course_d: ' || :NEW.Course_d);
```

```
END;
```

```
/
```

### **-3 package**

1.

```
CREATE OR REPLACE PACKAGE Faculty_pkg AS
```

```
FUNCTION get_faculty_data RETURN SYS_REFCURSOR;
```

```
PROCEDURE insert_faculty_record (
```

```
p_courses IN VARCHAR2,    p_time
```

```
IN                          VARCHAR2,
```

```
p_assigned_students IN NUMBER,
```

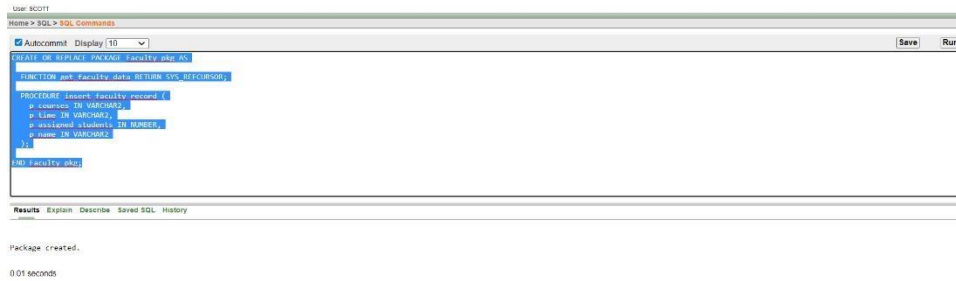
```
p_name IN VARCHAR2
```

```
);
```

```
END Faculty_pkg;
```



# Course Registration Management System



2.

CREATE OR REPLACE PACKAGE Student\_Pkg AS

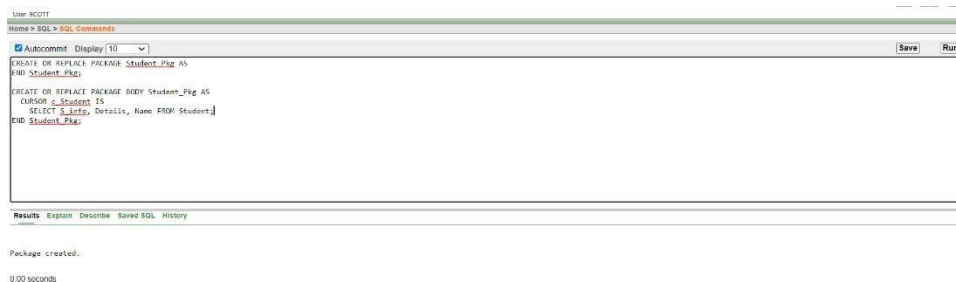
END Student\_Pkg;

CREATE OR REPLACE PACKAGE BODY Student\_Pkg AS

CURSOR c\_Student IS

SELECT S\_info, Details, Name FROM Student;

END Student\_Pkg;



3.

CREATE OR REPLACE PACKAGE DetailsCourse\_pkg AS

END DetailsCourse\_pkg;

CREATE OR REPLACE PACKAGE BODY DetailsCourse\_pkg AS

PROCEDURE insert\_course(course\_d IN VARCHAR2, credit\_per\_course IN NUMBER,  
course\_credit\_fee IN NUMBER) IS

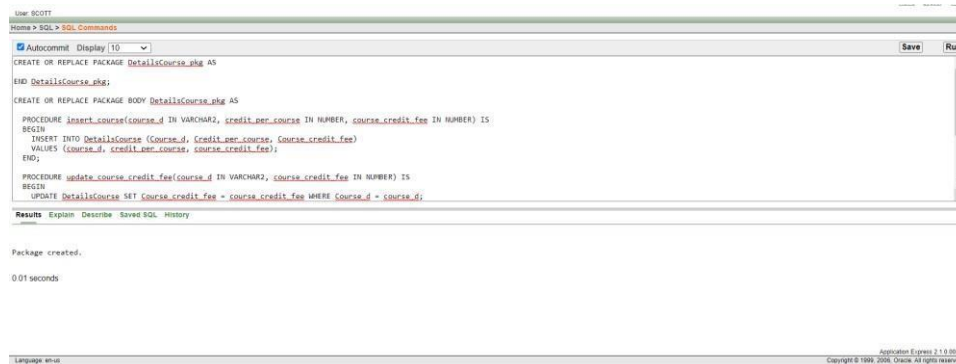
BEGIN

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

# Course Registration Management System

```
VALUES (course_d, credit_per_course, course_credit_fee);
END;
PROCEDURE update_course_credit_fee(course_d IN VARCHAR2, course_credit_fee IN
NUMBER) IS
BEGIN
    UPDATE DetailsCourse SET Course_credit_fee = course_credit_fee WHERE Course_d = course_d; END;

PROCEDURE delete_course(course_d IN VARCHAR2) IS
BEGIN
    DELETE FROM DetailsCourse WHERE Course_d = course_d;
END;
END DetailsCourse_pkg;
```



The screenshot shows a SQL Developer window with the following content:

```
SQL Command
CREATE OR REPLACE PACKAGE DetailsCourse_pkg AS
END DetailsCourse_pkg;

CREATE OR REPLACE PACKAGE BODY DetailsCourse_pkg AS
PROCEDURE insert_course(course_d IN VARCHAR2, credit_per_course IN NUMBER, course_credit_fee IN NUMBER) IS
BEGIN
    INSERT INTO DetailsCourse (Course_d, Credit_per_course, Course_credit_fee)
    VALUES (course_d, credit_per_course, course_credit_fee);
END;

PROCEDURE update_course_credit_fee(course_d IN VARCHAR2, course_credit_fee IN NUMBER) IS
BEGIN
    UPDATE DetailsCourse SET Course_credit_fee = course_credit_fee WHERE Course_d = course_d;
END;
END;

Results Explain Describe Saved SQL History
```

Package created.  
0.01 seconds

Application Express 2.1.9.0.0.38  
Copyright © 1999, 2008, Oracle. All rights reserved.

## Relational Algebra (Write down the question and also the answer.) -

1. Display all the info whose Student\_Pay is greater than 1000.  
Ans :  $\sigma$  Student\_Pay > 1000(Transaction).
2. Display all the information from students whose details are in CSE.      Ans  
:  $\sigma$  details = CSE(Student).
3. Display all the information from a student whose name starts with S.  
Ans :  $\sigma$  S\_name like 'S%' (Student).

## Course Registration Management System

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

$\Pi$  S-Transaction [ $\sigma$  Student\_Pay > 1000(Transaction)].

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

Ans :  $\Pi$  room\_number , Section(class).

### **Conclusion:**

The proposed course registration management system will provide educational institutions with an efficient and effective tool to manage student course registration. The system will improve the registration process for both students and administrators, reduce errors, and provide real-time information on course availability and enrolment status. The system will be developed using web-based technologies and hosted on a cloud-based server to ensure accessibility and scalability.