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

**Project Name:** 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

# Course Registration Management System

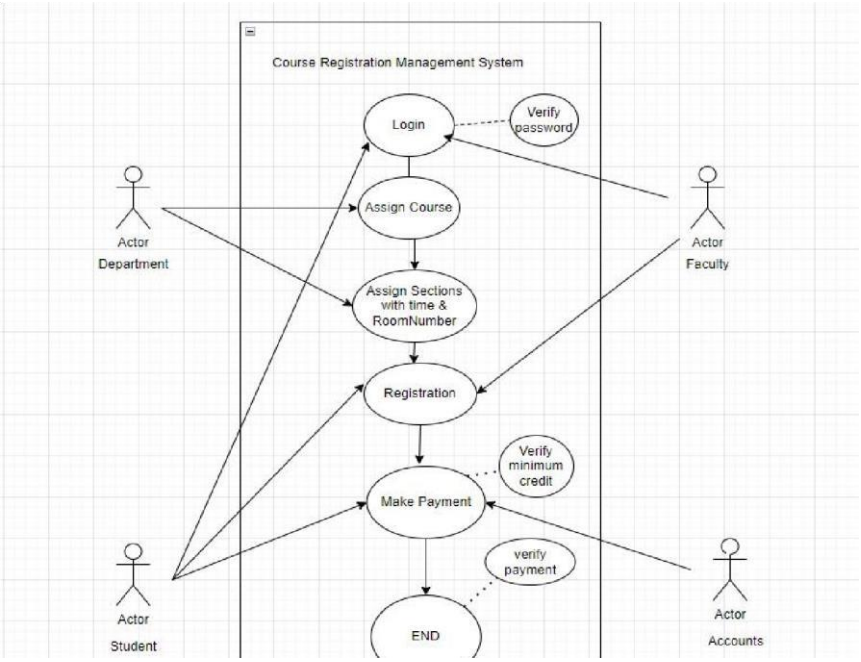
**Introduction:**

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.

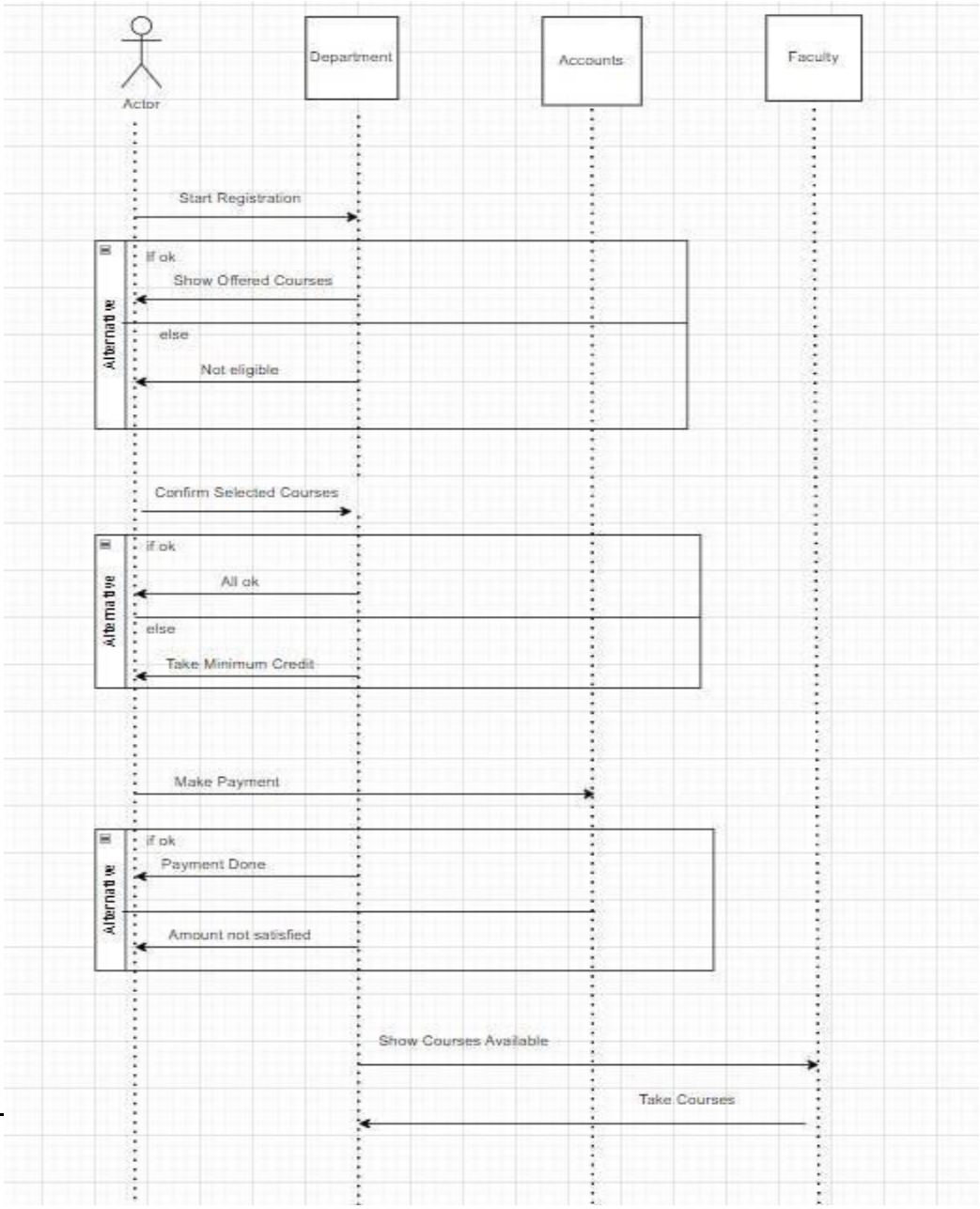
**Project Proposal:**

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.

**Use Case Diagram:**

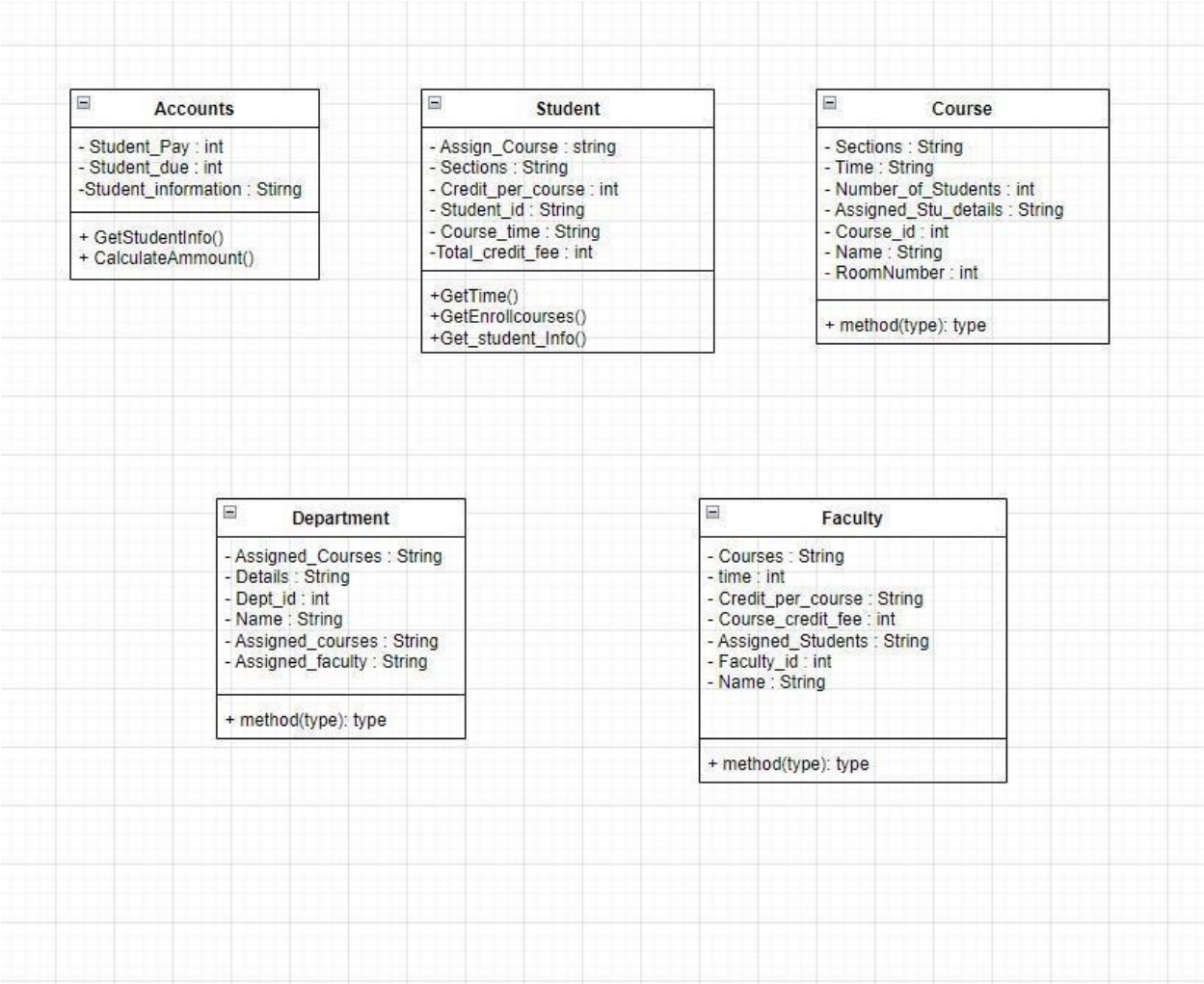


**Activity Diagram:**



**Class Diagram:**

Figure: Class Diagram



**Interface Design and Description:**

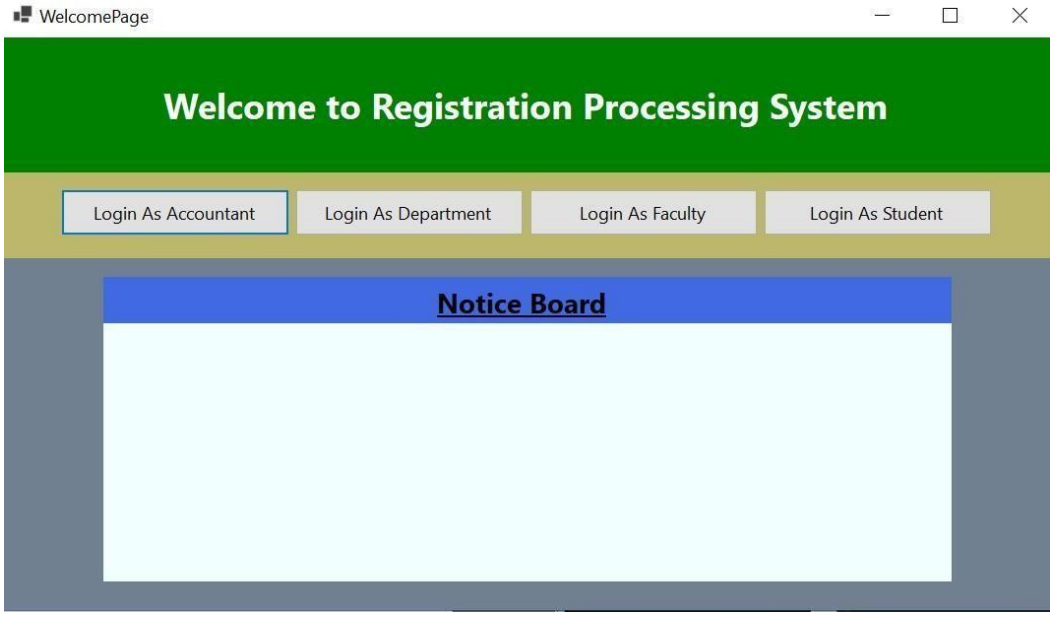



Figure 1: Welcome_Page

After opening the apps 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.

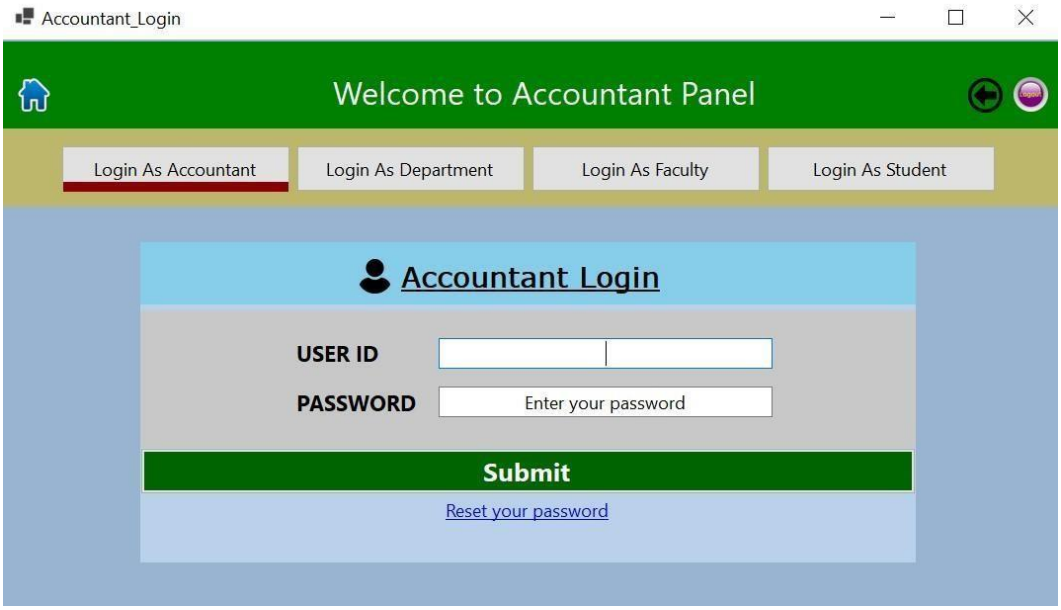



Figure 2: Accountant Login

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.

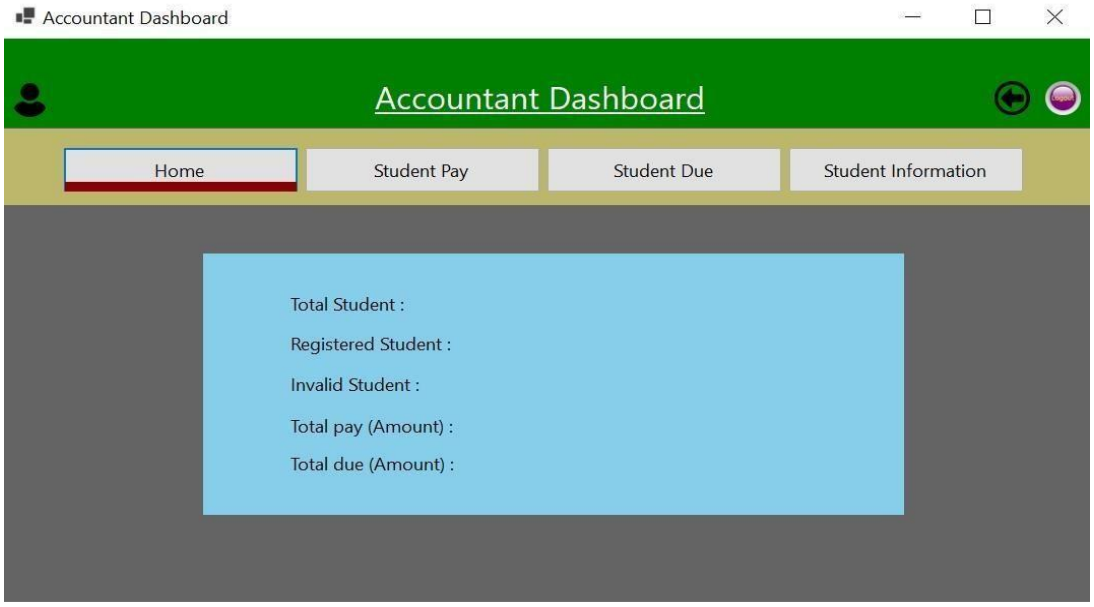


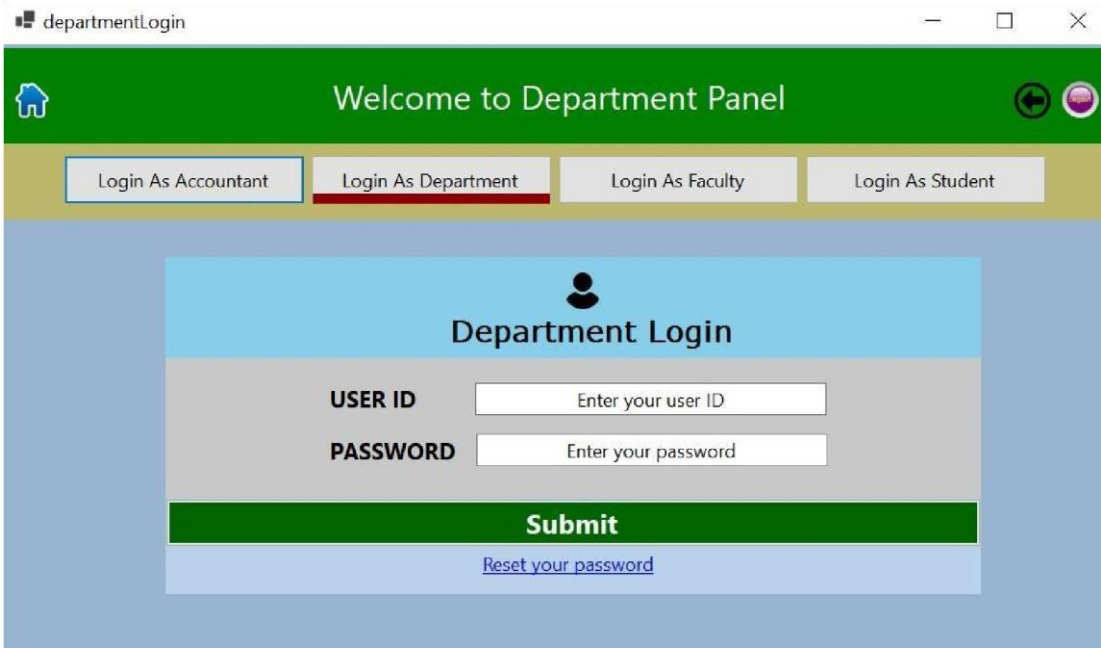

Figure 3: Accountant Dashboard

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

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 “Loging As Department” then the Department Login panel will be appear.



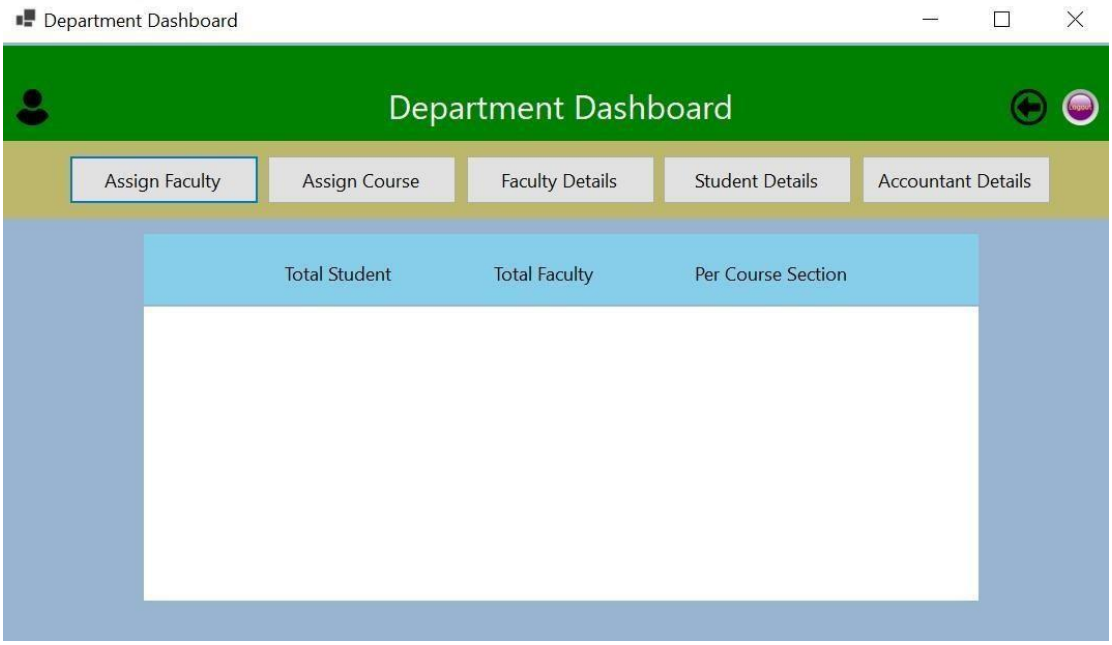



Figure 5: Department Dashboard

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.

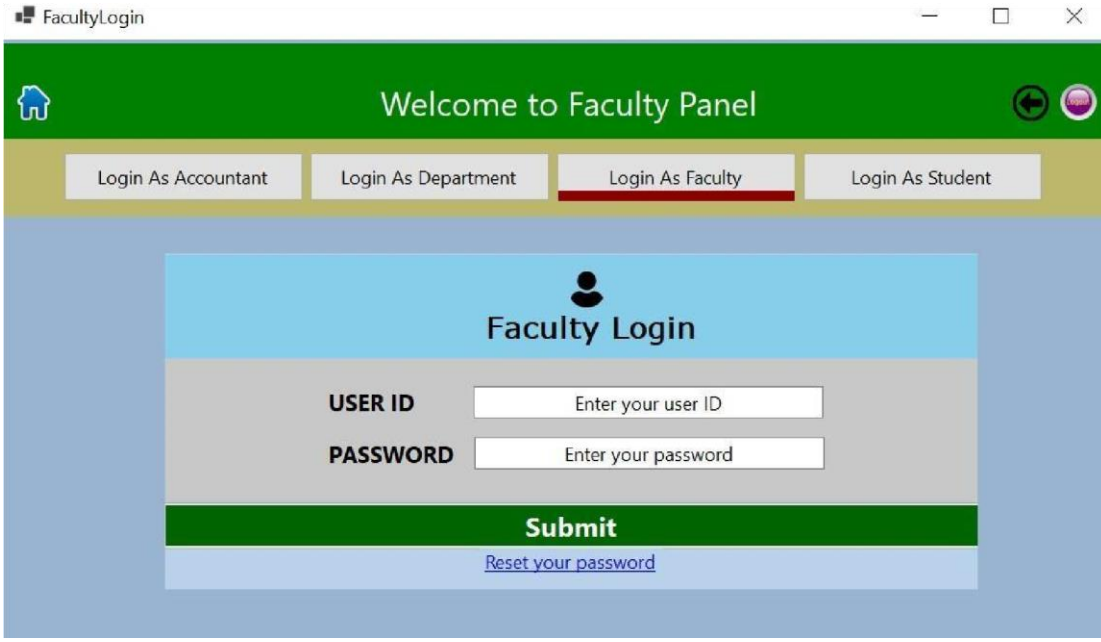



Figure 6: Faculty Login

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

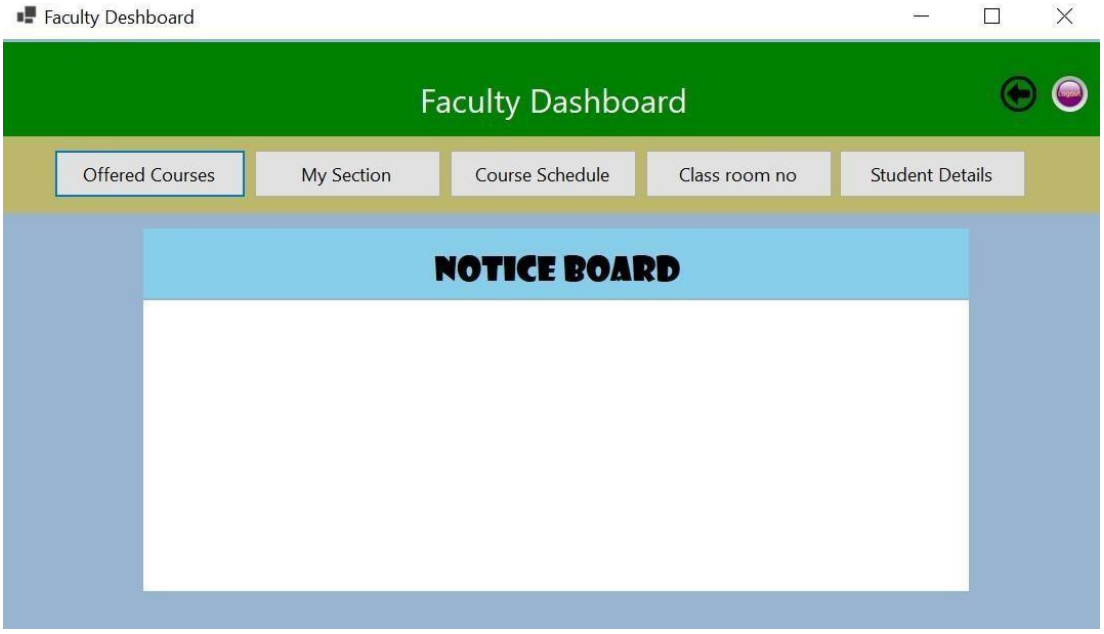



Figure 7: Faculty Dashboard

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.

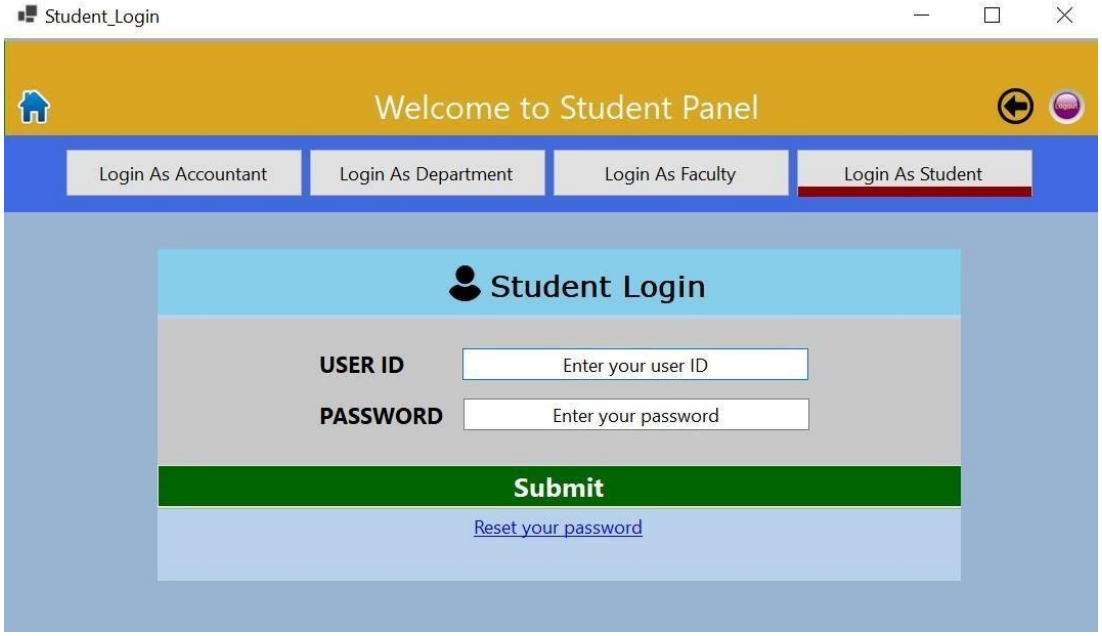



Figure 8: Student Login

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

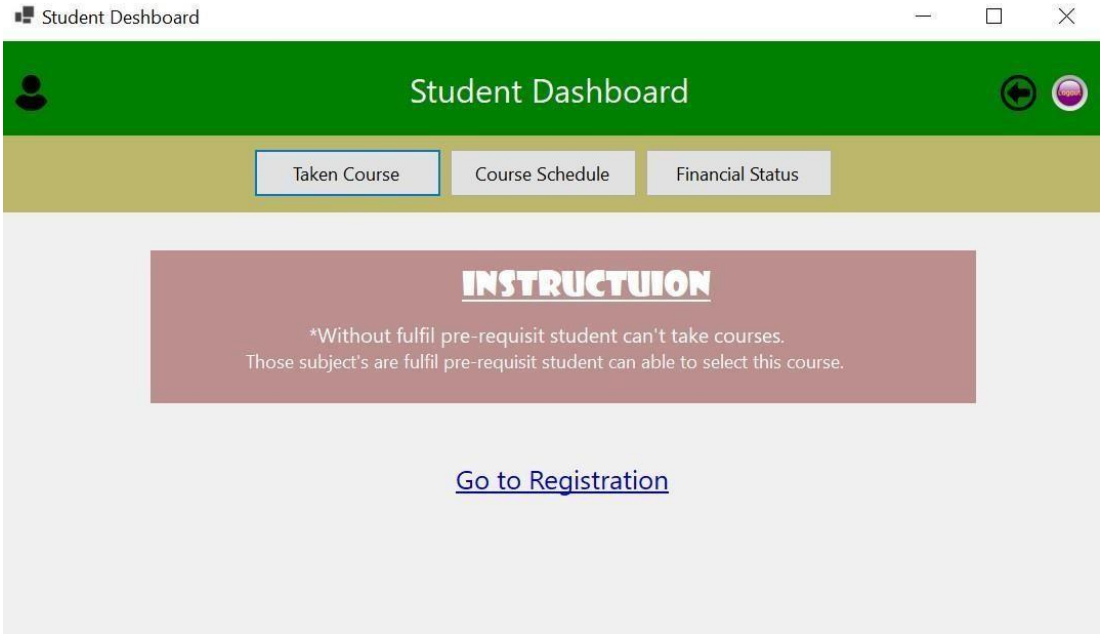



Figure 9: Student Dashboard

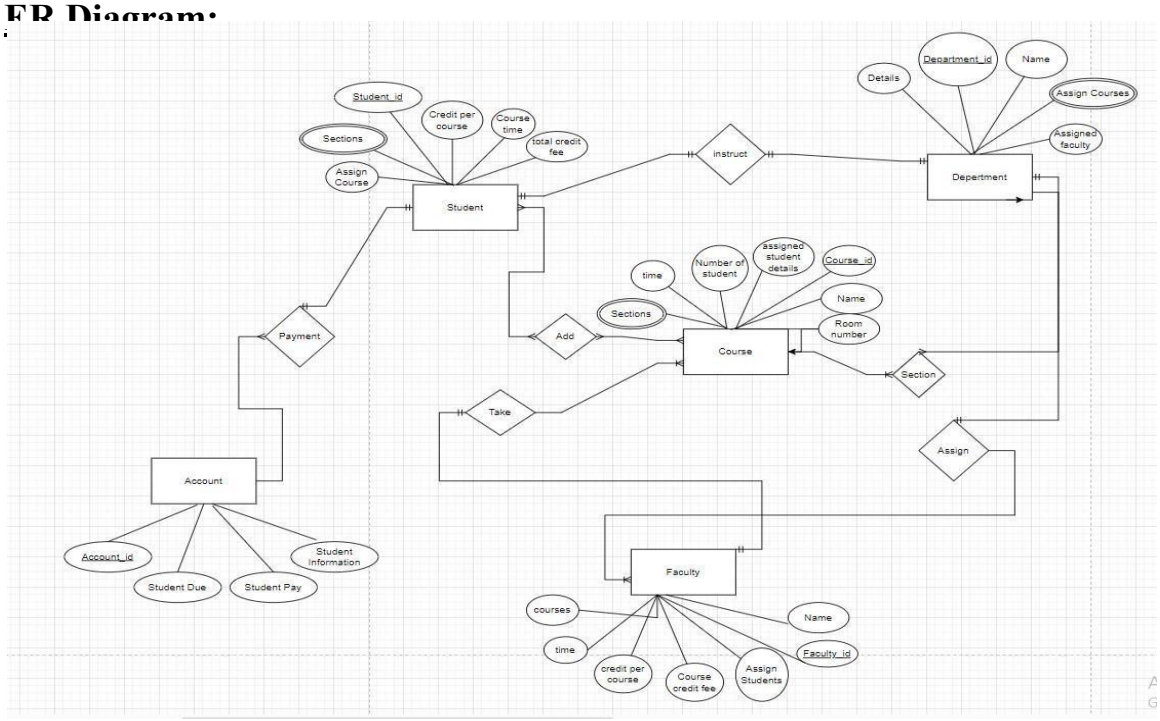
In the student dashboard there are 3 buttons. By pressing Taken Course user can see the courses which they are taken. Course schedule for seeing the schedule. Financial status for knowing the payable amount after confirming the registration process.

## 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

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, Studnets_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, Studnets_id, Course_time, total_credit_fee.

2NF- Accountant_id, Students_pay, Students_due, Students_information. Studnets_id, Assign_Course, Sections, Credit_per_course, Course_time, total_credit_fee.

3NF- Accountant_id, Students information.
_S_transaction, Students_pay, Students_due.

Studnets_id, Assign Course, Credit per course, total credit fee.
Course_info, Sections, Course time.

## Final table-

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.

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.

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.

**After removing repetition final tables are-**

**1.**

Accountant_id, Students information, S_transaction, Studnets_id, Course_info.

**2.**

S_transaction, Students_pay, Students_due. **3.**

Studnets_id, Assign Course, Credit per course, total credit fee, Course_info, Department_id, S_info, Course_id, Class_info. **4.**

Course_info, Course time, Sections.

**5.**

Department_id, Assigned faculty, Assigned_courses, s_info, course_id, class_info, faculty_id, course_d. **6.**

S_info, Details, Name. **7.**

Course_id, time, Number of students, assigned student details, Name, Class_info, Faculty_id, Course_d, **8.**

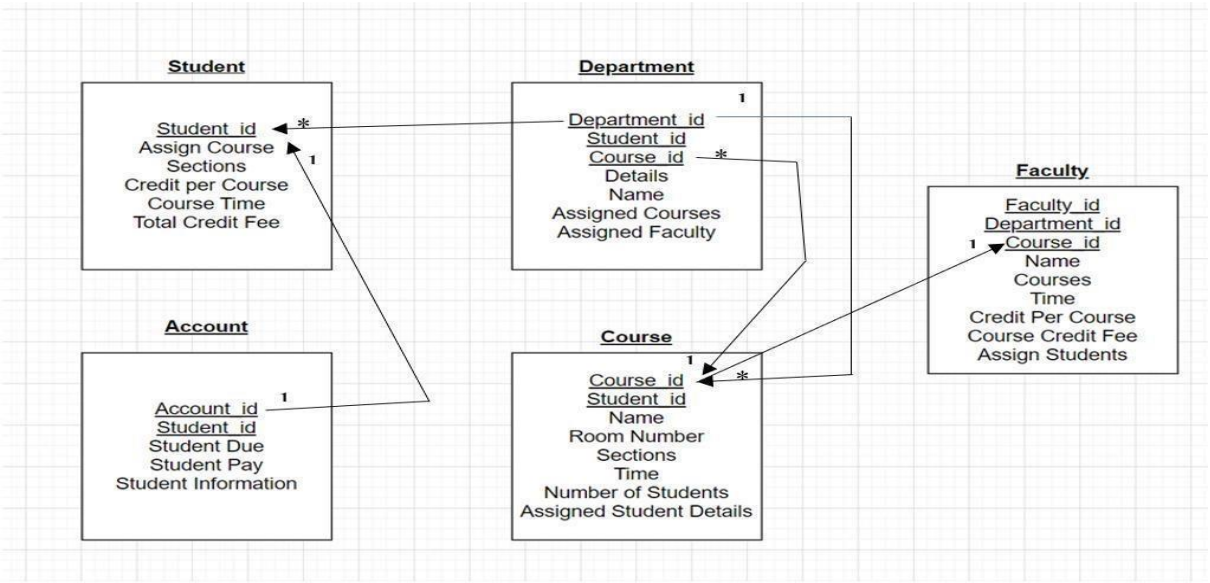
Class_info, room number, Sections.

**9.**

Faculty_id, Courses, time, assigned students, Name. **10.**

Course_d, credit per course, Course credit fee.

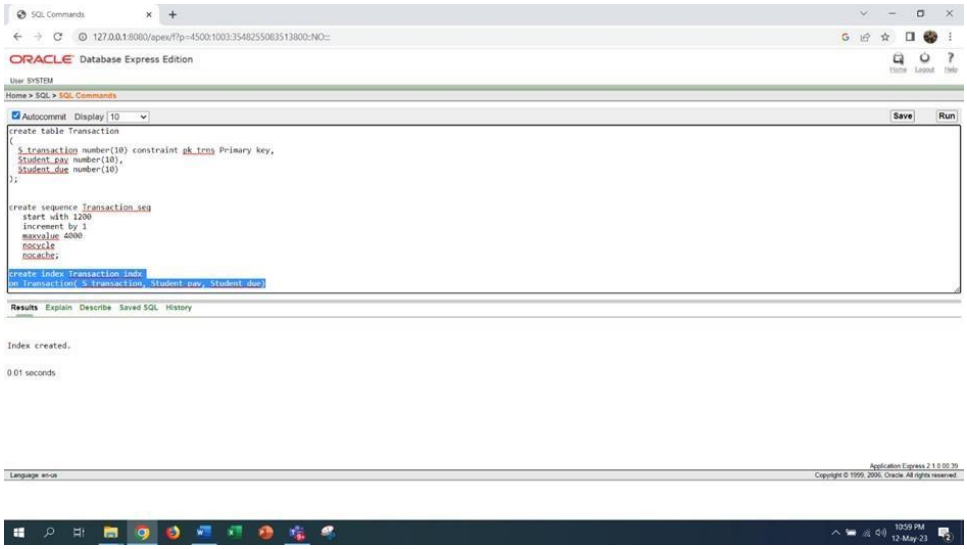
## Schema Diagram:

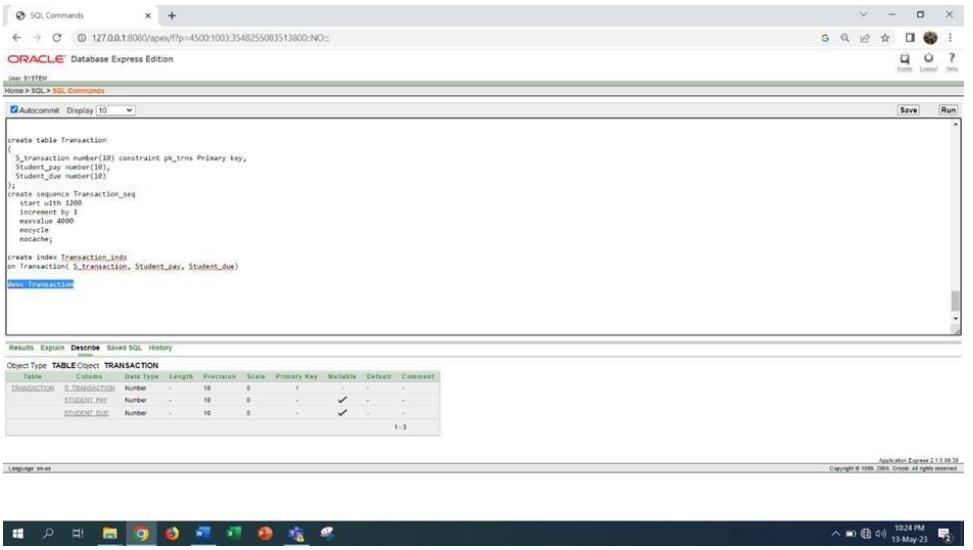



## Table Creation:

### 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_indx
on Transaction( S_transaction, Student_pay, Student_due)
```




## 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

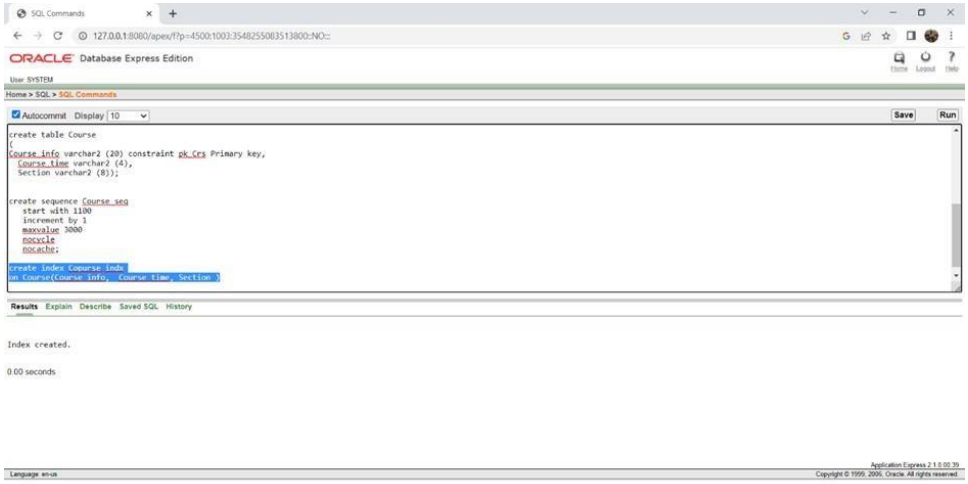
maxvalue 3000 nocycle

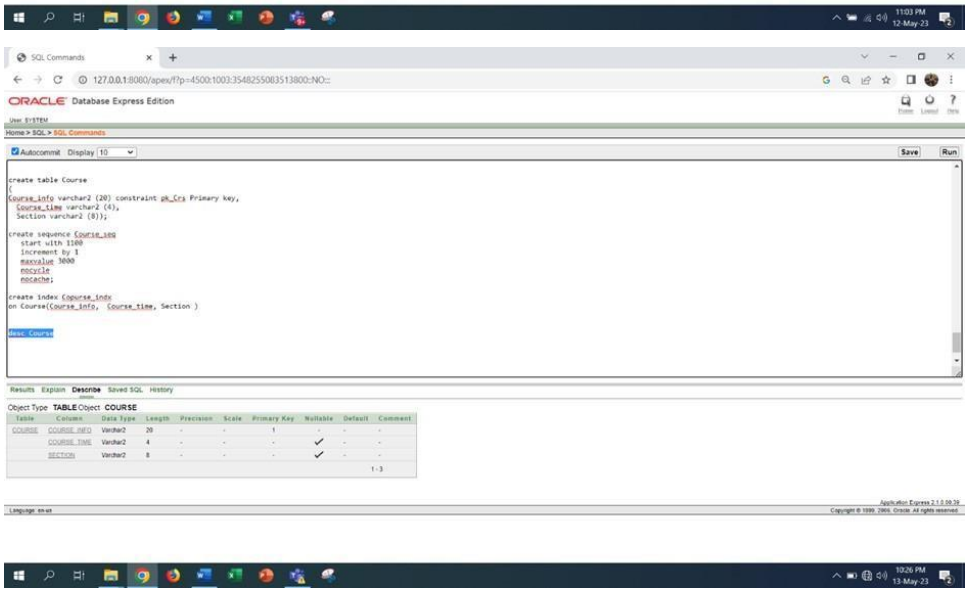
nocache; create index

Copurse_indx on

Course(Course_info,

Course_time, Section )




## 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

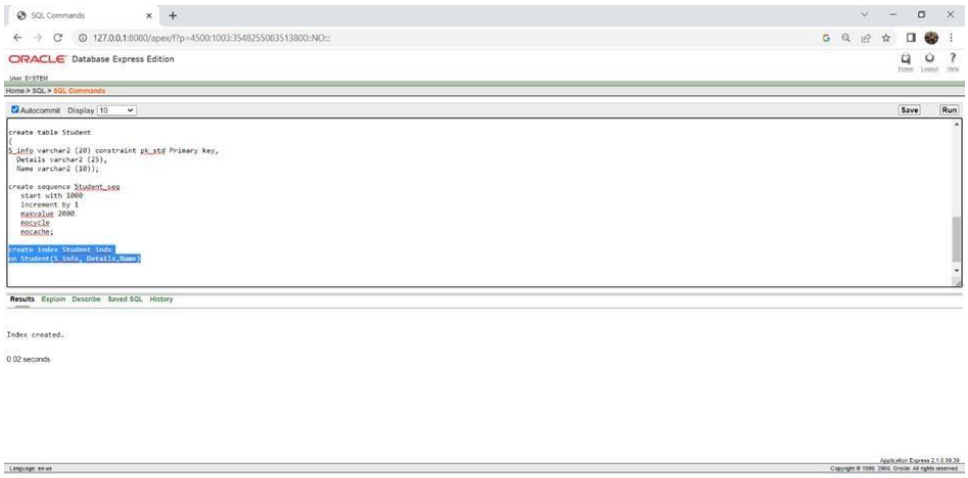
with 1000 increment by 1

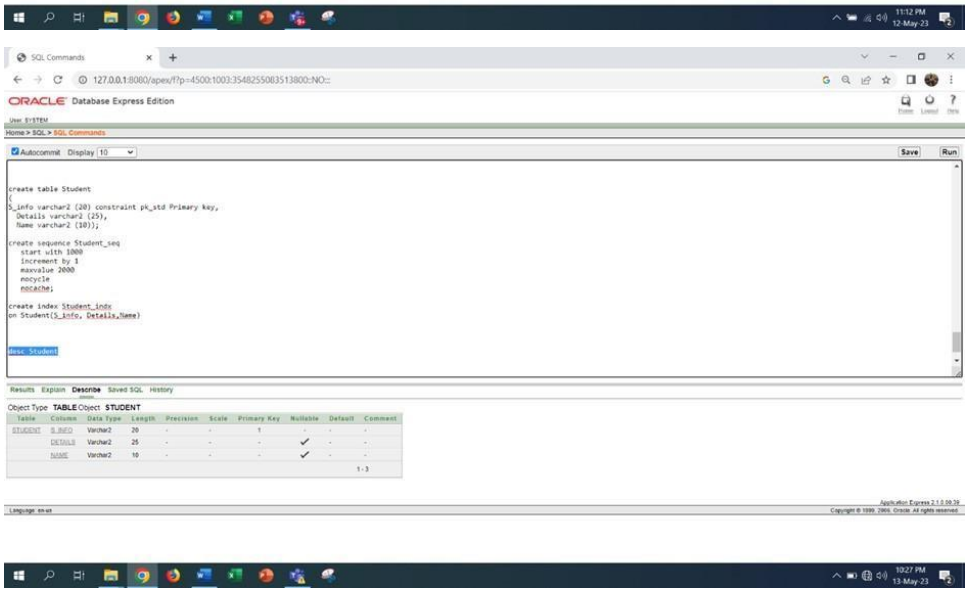
maxvalue 2000 nocycle

nocache; create index

Student_indx on Student(S_info,

Details,Name)




## Table Class

create table Class

(

Class_info varchar2 (20) constraint pk_Cls Primary key,

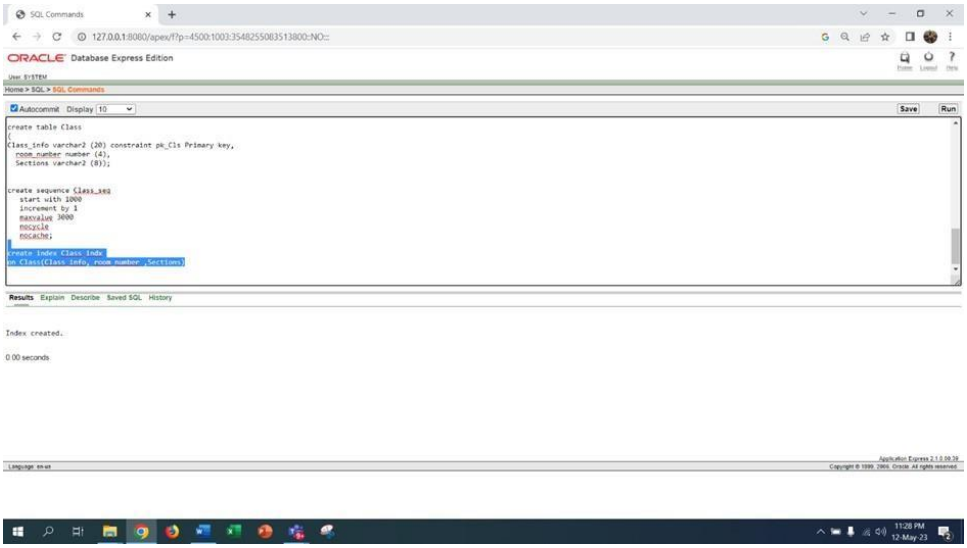
room_number number (4), Sections varchar2 (8)); create

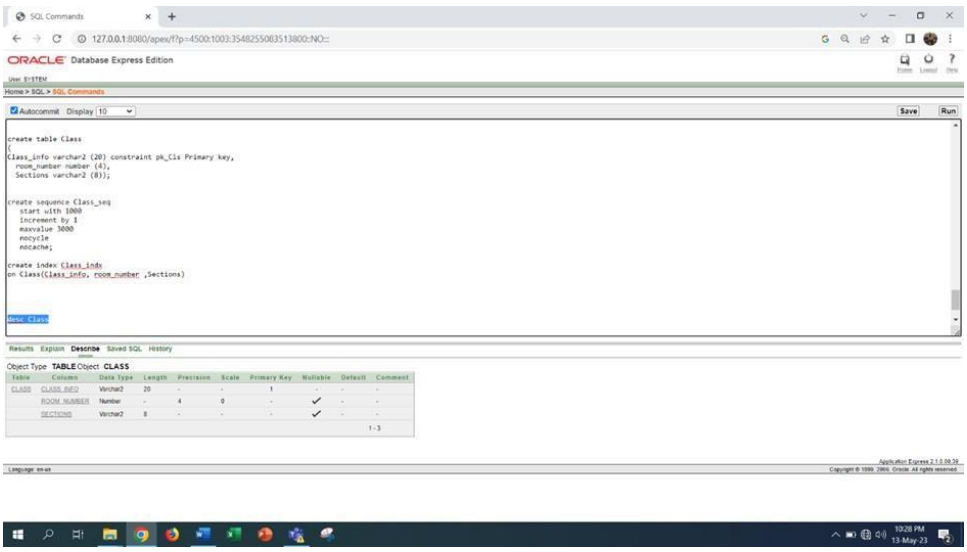
sequence Class_seq start with 1000 increment by 1

maxvalue 3000 nocycle nocache; create

index Class_indx on Class(Class_info,

room_number ,Sections)



## Table DetailsCourse

create table DetailsCourse

(

Course_d varchar2 (20) constraint pk_crsdtls Primary key,

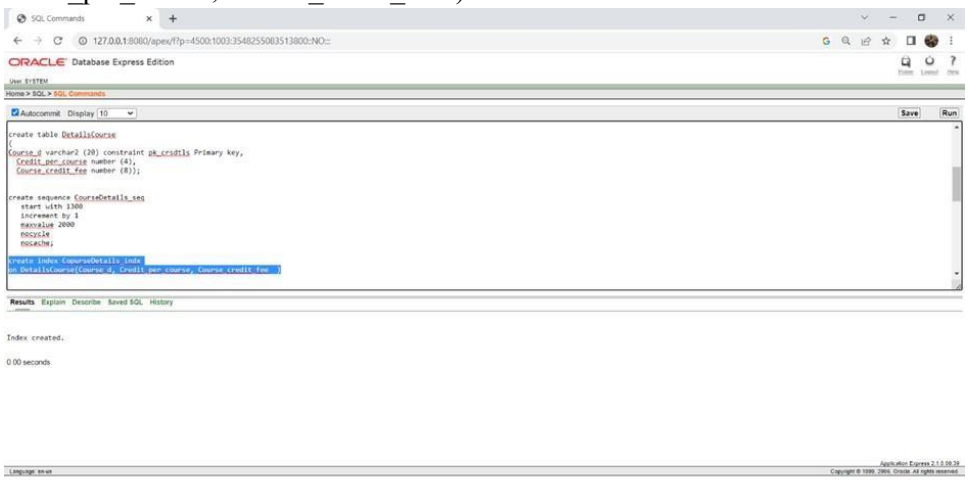
Credit_per_course number (4), Course_credit_fee number (8)); create

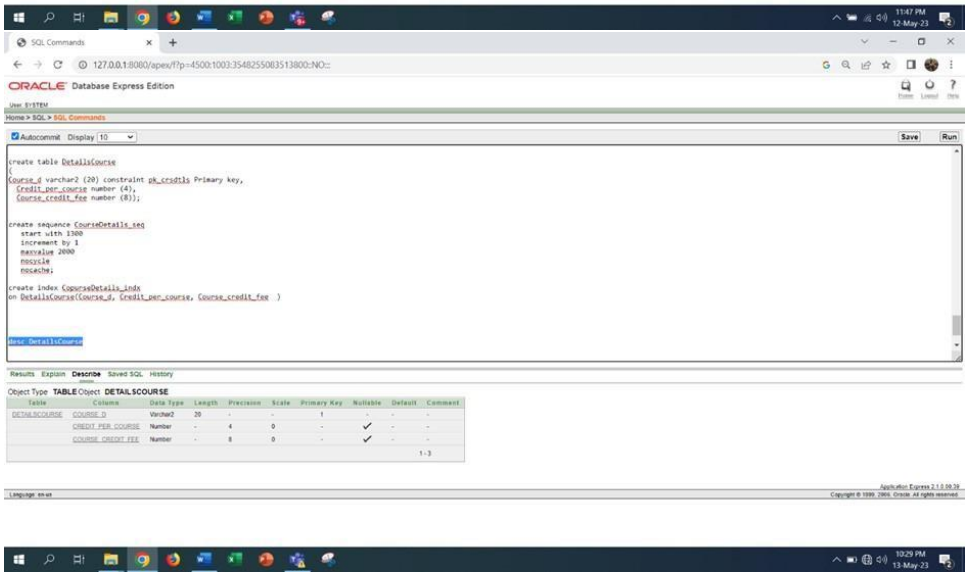
sequence CourseDetails_seq start with 1300 increment by

1 maxvalue 2000 nocycle nocache; create index

CopurseDetails_indx on DetailsCourse(Course_d,

Credit_per_course, Course_credit_fee )



## Table Faculty

create table Faculty

(

Faculty_id number (8) constraint pk_Flt Primary key,

Courses varchar2 (30),

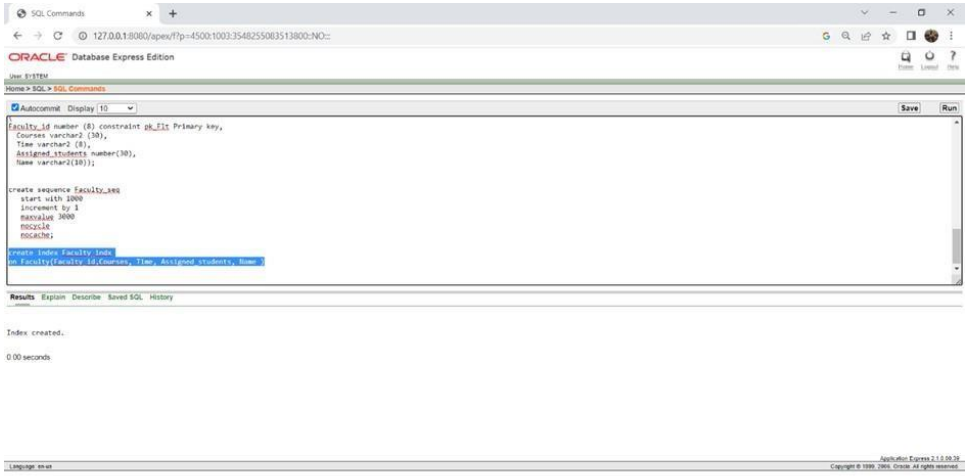
Time varchar2 (8),

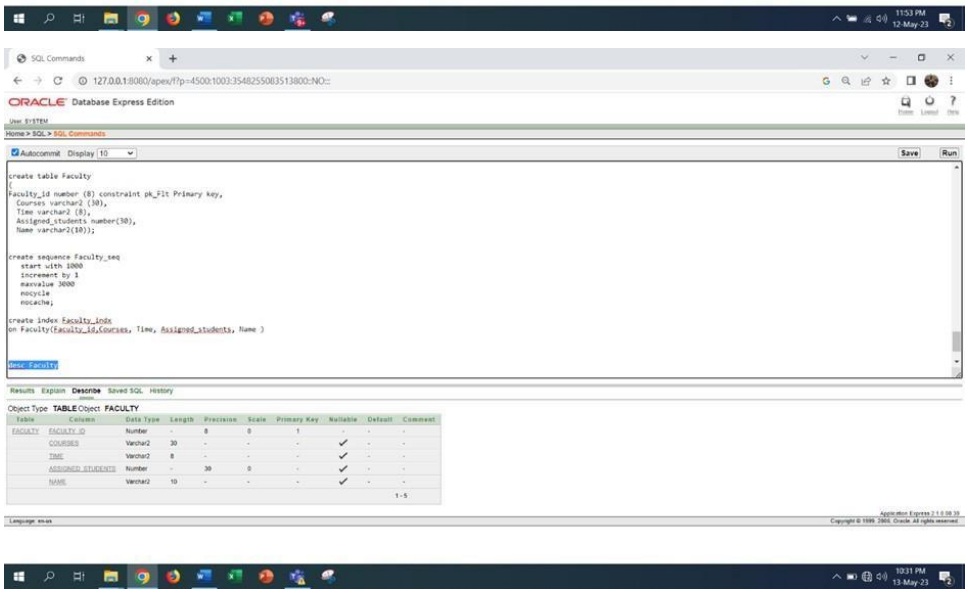
Assigned_students number(30), Name varchar2(10)); create

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 )



## Table Accountant

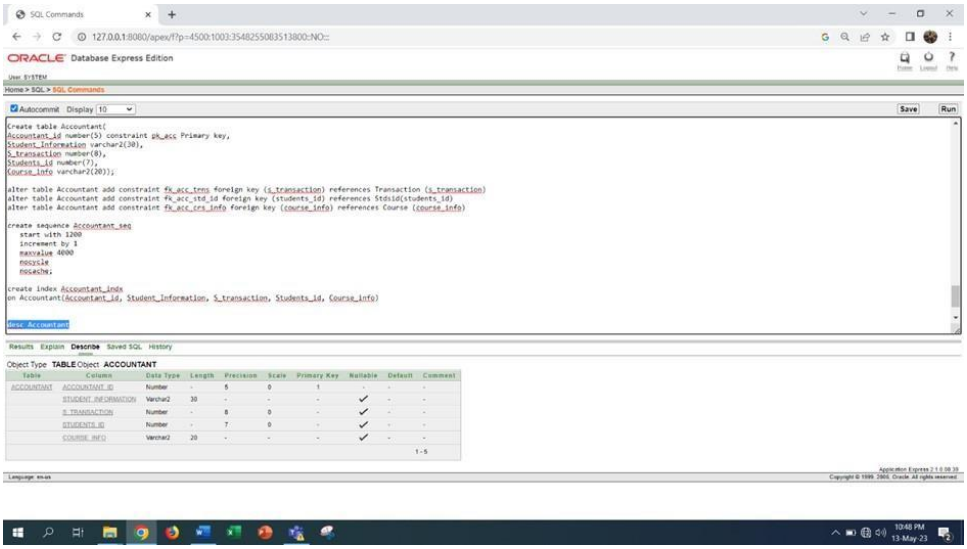
Create table Accountant(

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),

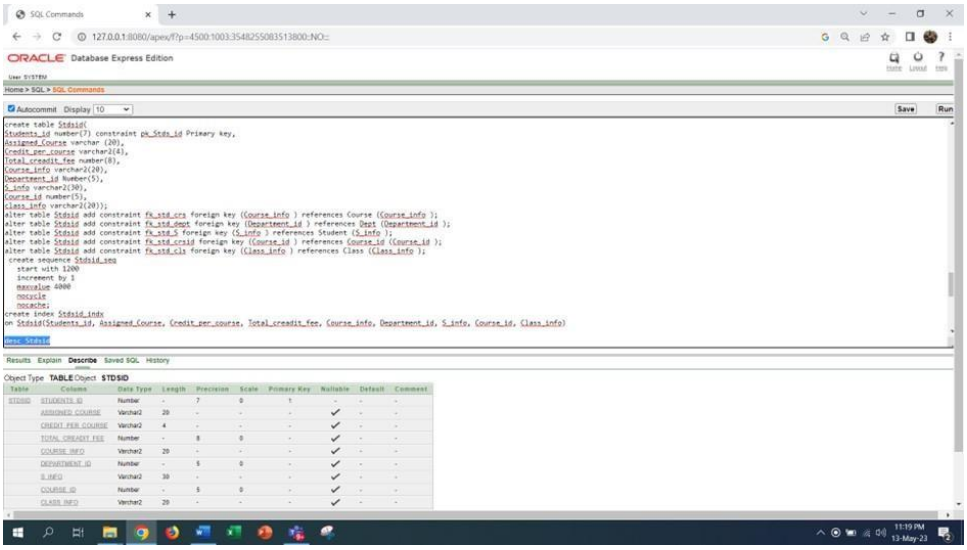
Credit_per_course varchar2(4),

Total_creadit_fee number(8),

Course_info varchar2(20),

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),

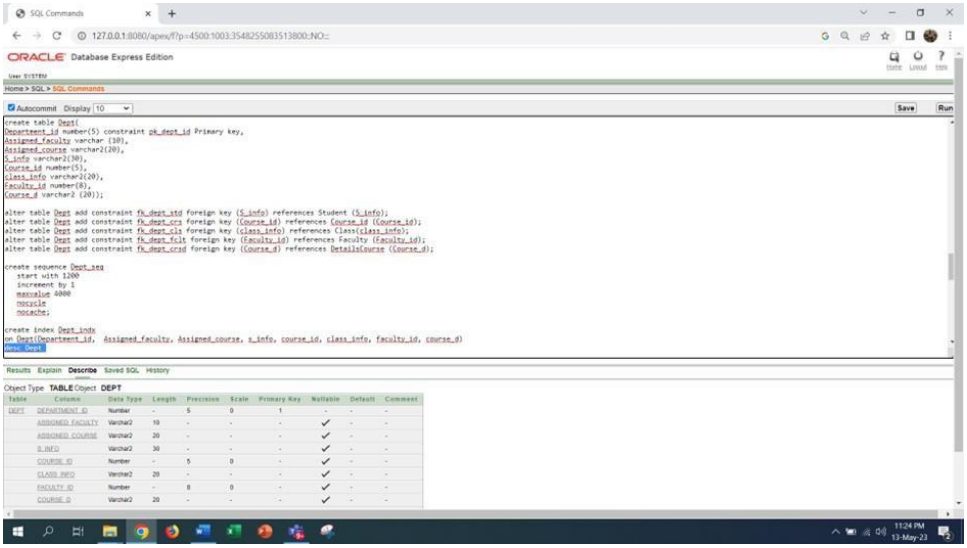
S_info varchar2(30), Course_id number(5), class_info varchar2(20), Faculty_id number(8), Course_d varchar2 (20)); alter table Dept add constraint fk_dept_std foreign key (S_info) references Student (S_info); alter table Dept add constraint fk_dept_crs foreign key (Course_id) references Course_id

(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

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),

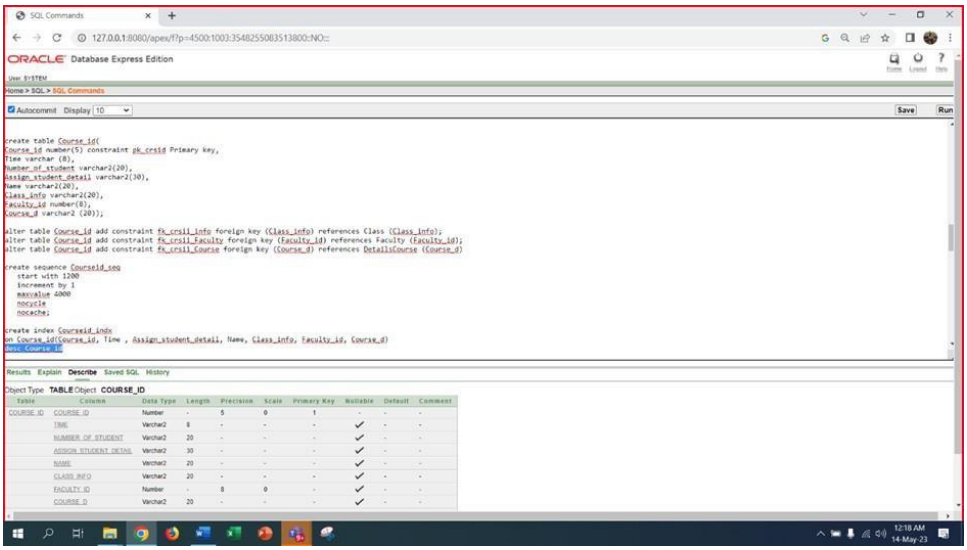
Class_info varchar2(20),

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)

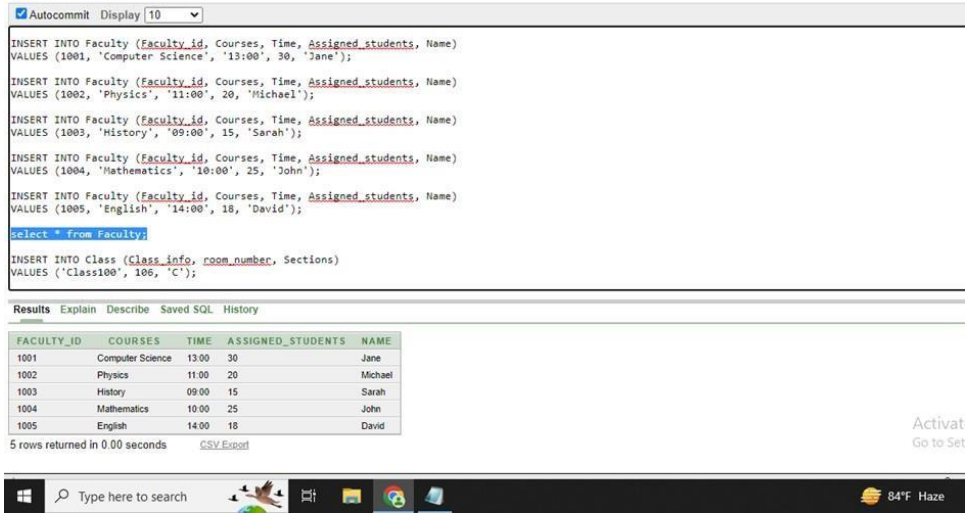
```
create sequence Courseid_seq
start with 1200    increment by
1    maxvalue 4000    nocycle
nocache;

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)
```

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




INSERT INTO Class (Class_info, room_number, Sections)

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

INSERT INTO Class (Class_info, room_number, Sections)

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

INSERT INTO Class (Class_info, room_number, Sections)

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

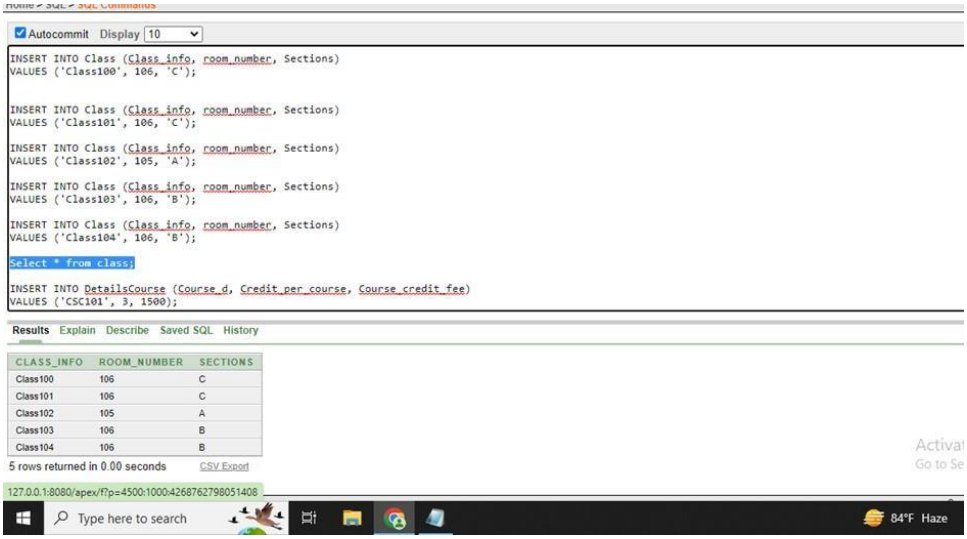
INSERT INTO Class (Class_info, room_number, Sections)

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

INSERT INTO Class (Class_info, room_number, Sections)

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

Select * from class;




INSERT INTO DetailsCourse (Course_d, Credit_per_course, Course_credit_fee)

VALUES ('CSC101', 3, 1500);

INSERT INTO DetailsCourse (Course_d, Credit_per_course, Course_credit_fee)

VALUES ('CSC102', 3, 1500);

INSERT INTO DetailsCourse (Course_d, Credit_per_course, Course_credit_fee)

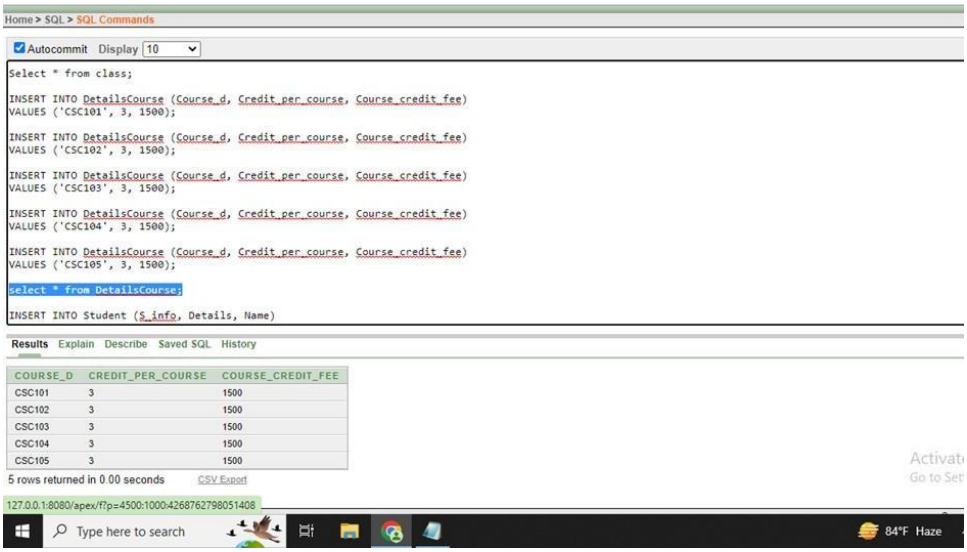
VALUES ('CSC103', 3, 1500);

INSERT INTO DetailsCourse (Course_d, Credit_per_course, Course_credit_fee)

VALUES ('CSC104', 3, 1500);

INSERT INTO DetailsCourse (Course_d, Credit_per_course, Course_credit_fee)

VALUES ('CSC105', 3, 1500); select * from DetailsCourse;




INSERT INTO Student (S_info, Details, Name)

VALUES ('0001', 'CSE', 'Noboni');

INSERT INTO Student (S_info, Details, Name)

VALUES ('0002', 'CSE', 'Succho');

INSERT INTO Student (S_info, Details, Name)

VALUES ('0003', 'SE', 'Fahim');

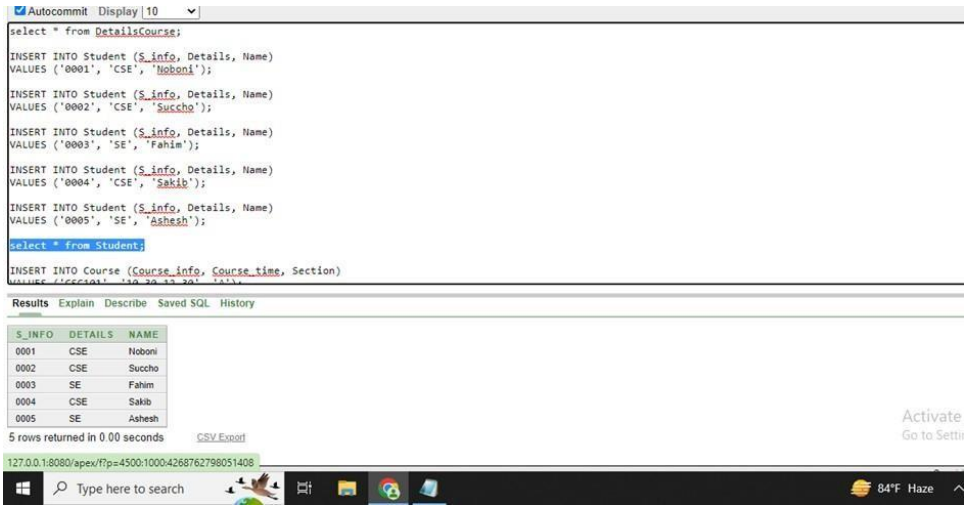
INSERT INTO Student (S_info, Details, Name)

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');

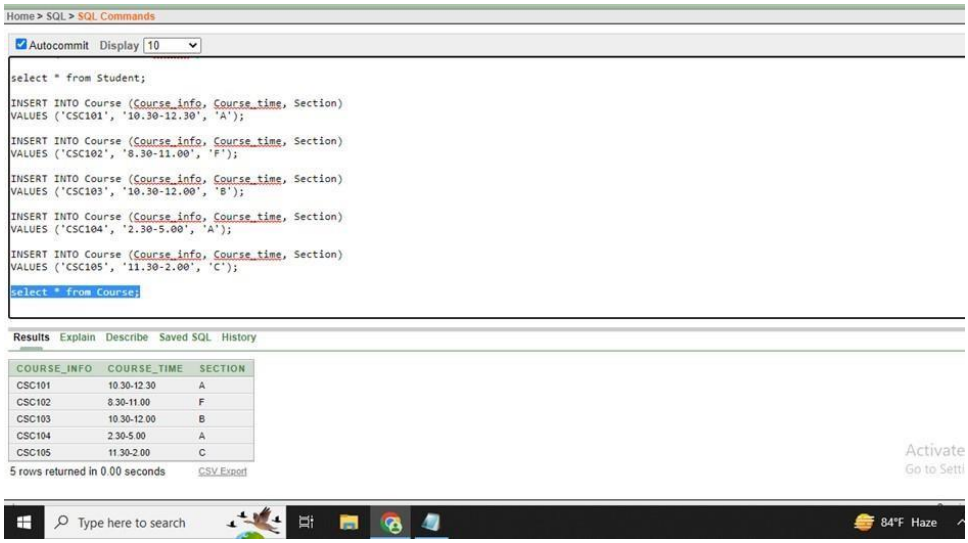
INSERT INTO Course (Course_info, Course_time, Section) VALUES

('CSC104', '2.30-5.00', 'A');

INSERT INTO Course (Course_info, Course_time, Section)

VALUES ('CSC105', '11.30-2.00', 'C'); select * from

Course;




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);

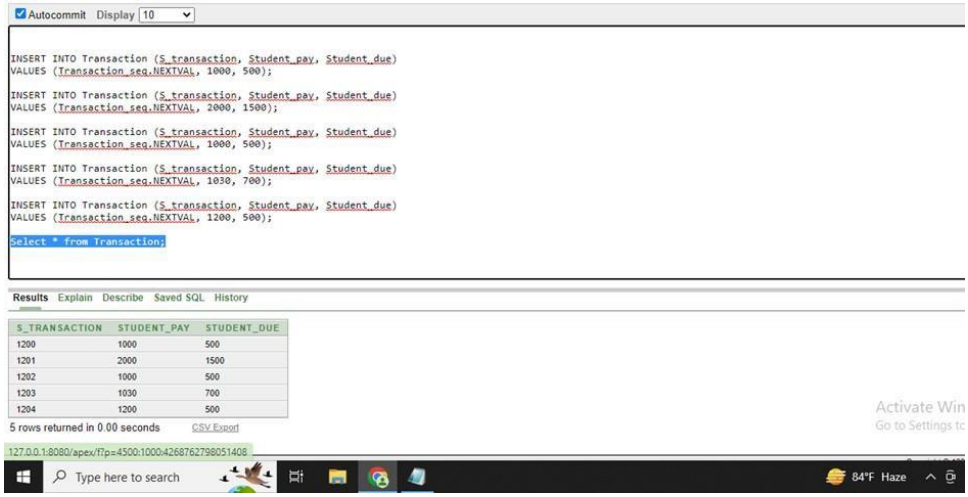
INSERT INTO Transaction (S_transaction, Student_pay, Student_due)

VALUES (Transaction_seq.NEXTVAL, 1030, 700);

INSERT INTO Transaction (S_transaction, Student_pay, Student_due)

VALUES (Transaction_seq.NEXTVAL, 1200, 500);

Select * from Transaction;



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');

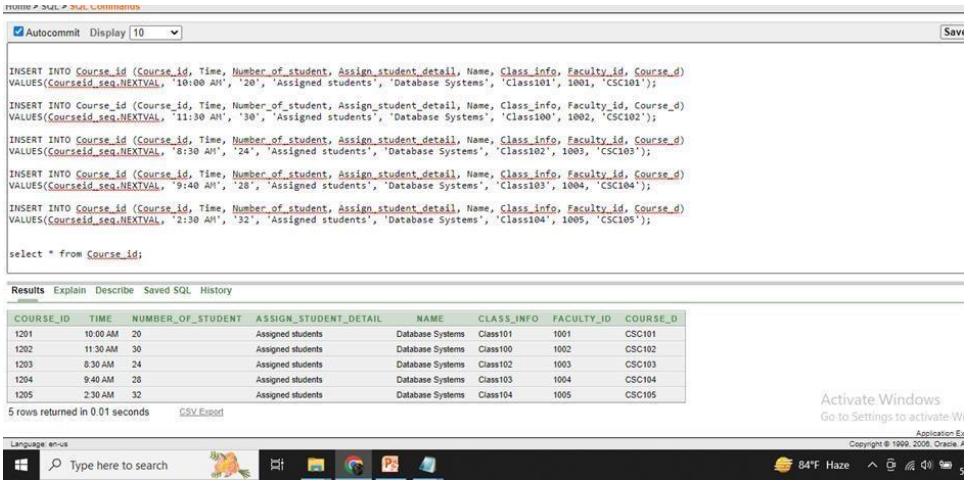
INSERT INTO Course_id (Course_id, Time, Number_of_student, Assign_student_detail, Name, Class_info, Faculty_id, Course_d)

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;




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

VALUES (101, 'FST', 'Biology', '0005', 1203, 'Class103', 1002, 'CSC101');

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

VALUES (102, 'BBA', 'Communications', '0003', 1202, 'Class104', 1003, 'CSC103');

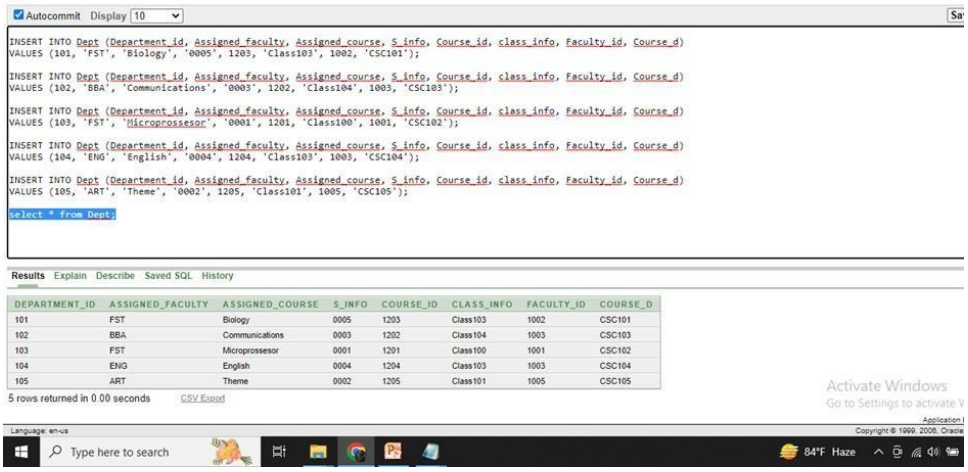
INSERT INTO Dept (Department_id, Assigned_faculty, Assigned_course, S_info, Course_id, class_info, Faculty_id, Course_d)

VALUES (103, 'FST', 'Microprossesor', '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');




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

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

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

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

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

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

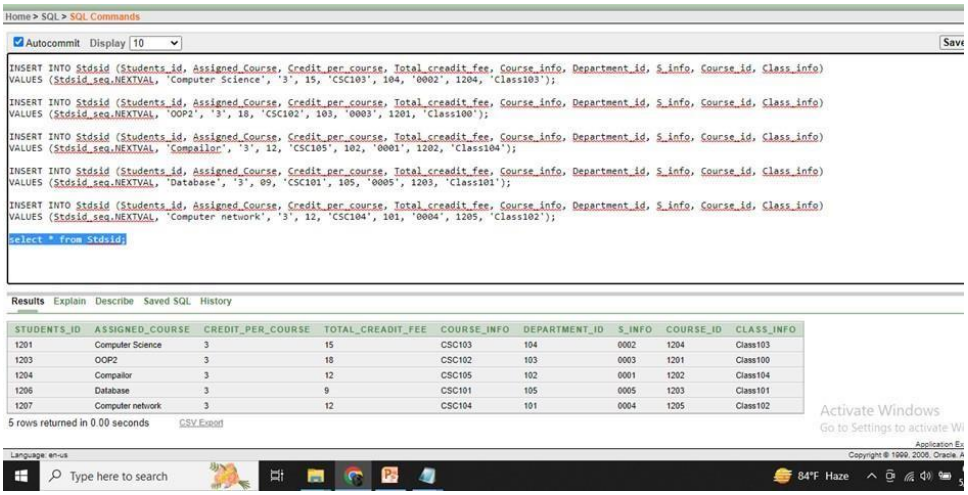
INSERT INTO Stdsid (Students_id, Assigned_Course, Credit_per_course, Total_creadit_fee, Course_info, Department_id, S_info, Course_id, Class_info)

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

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

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

* from stdsid;




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)

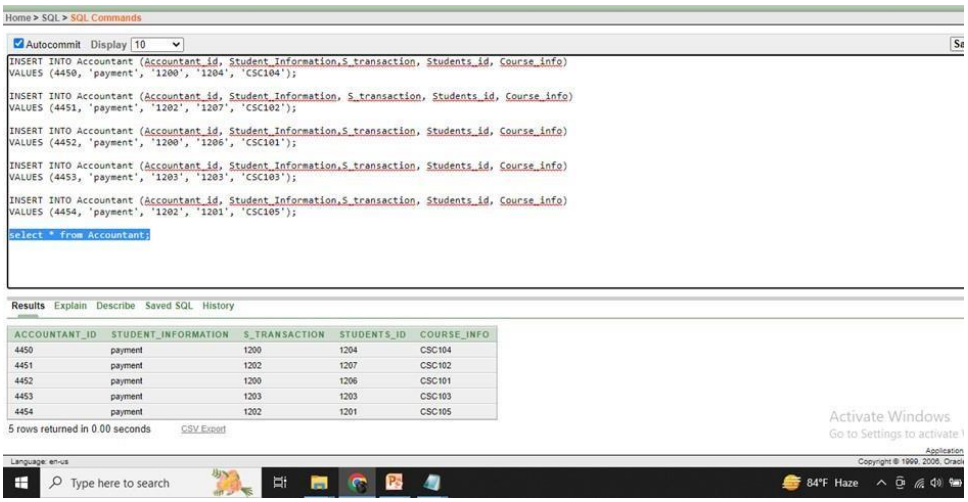
VALUES (4452, 'payment', '1200', '1206', 'CSC101');

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;

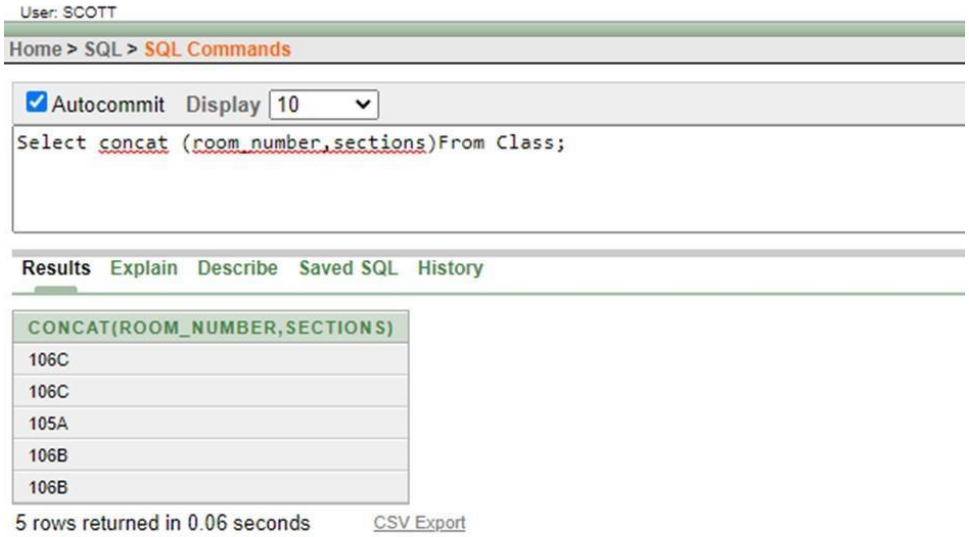


## SQL -Query

## single-row function -3

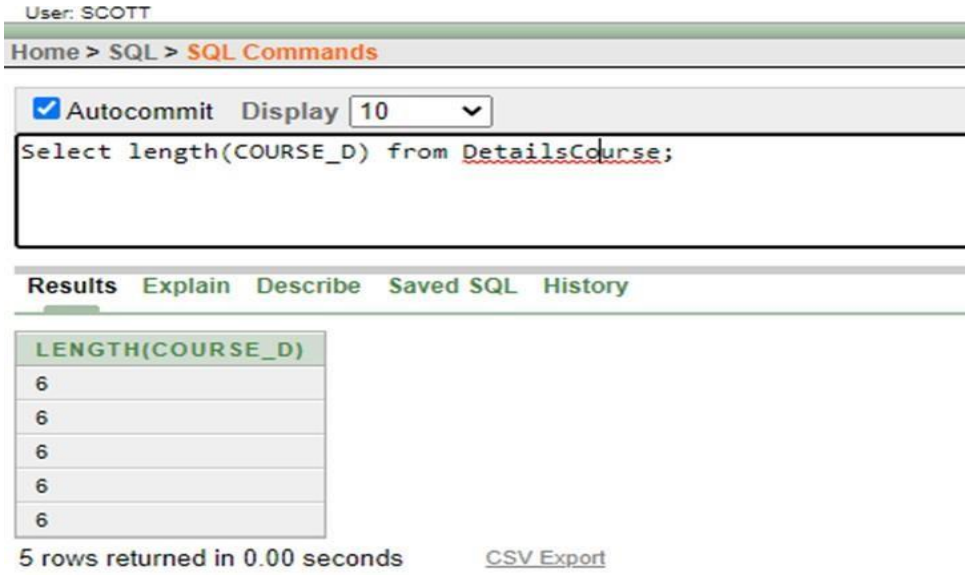
1. Select concat (room_number,sections)

From Class



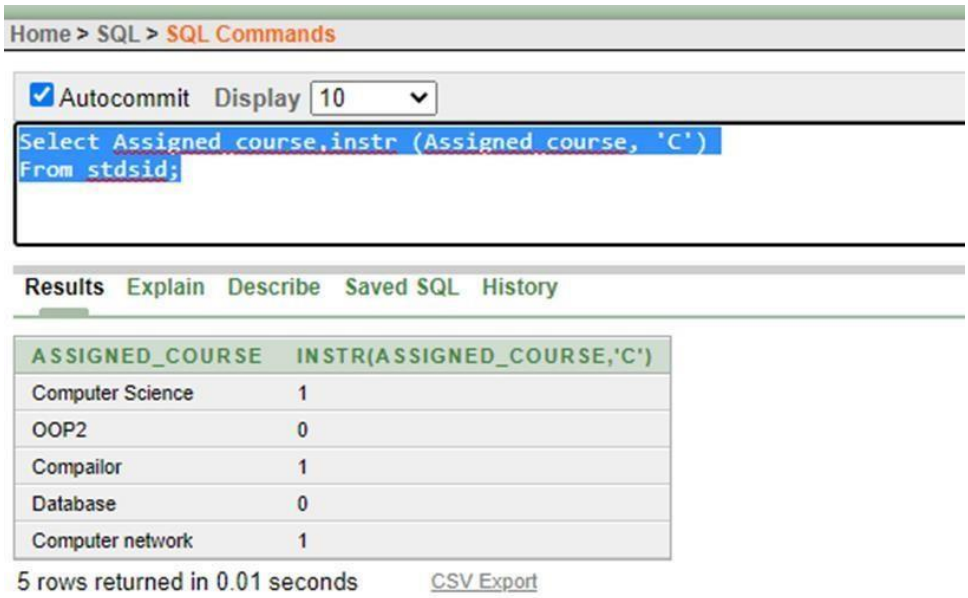

2. Select name length(name)

From DetailsCourse



3. Select Assigned_course,instr (Assigned_course, ‘C’)

From stdsid



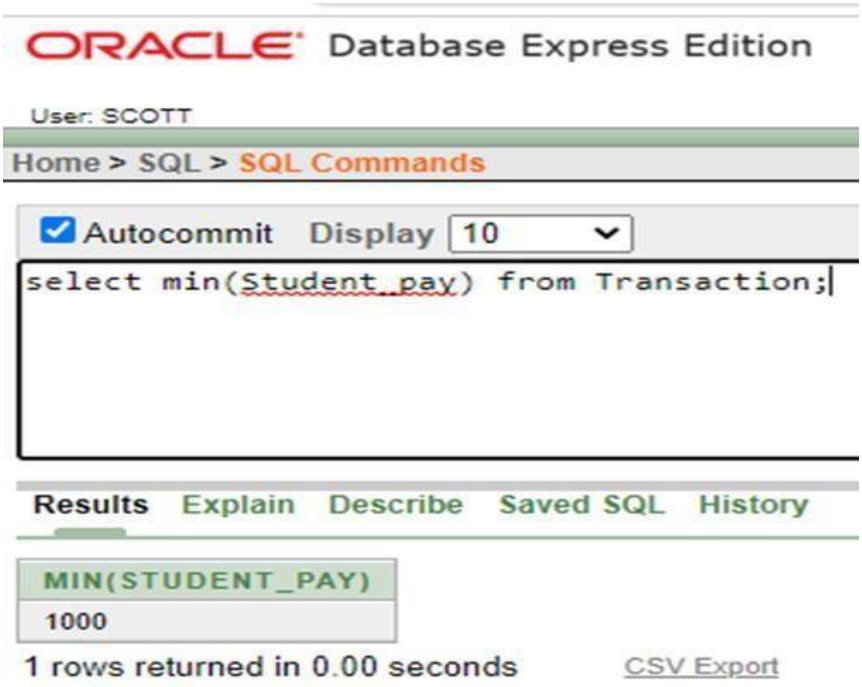

## Group function -3

1. select max(Student_due) from

Transaction

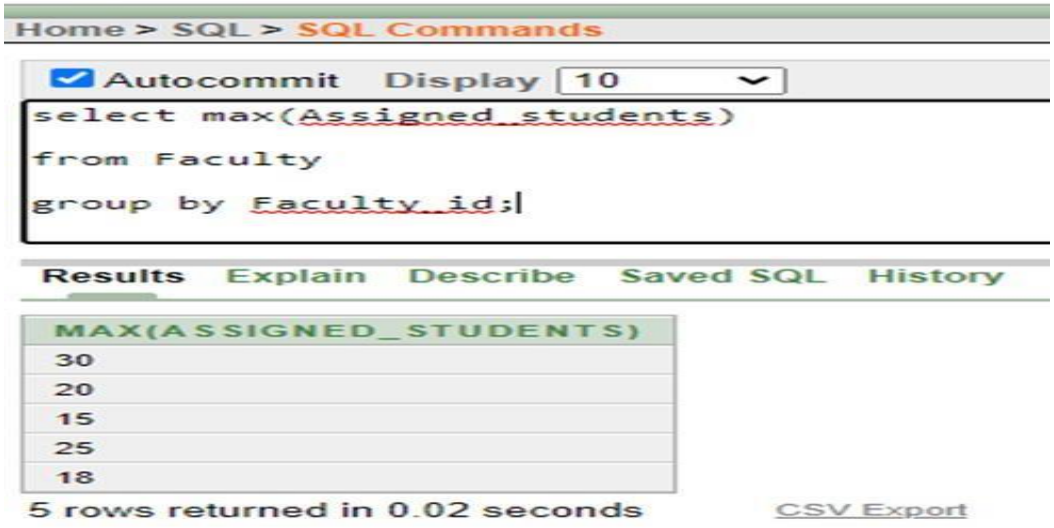

2. select min(Student_pay) from

Transaction




3. select max(Assigned_students) from

Faculty group by Faculty_id



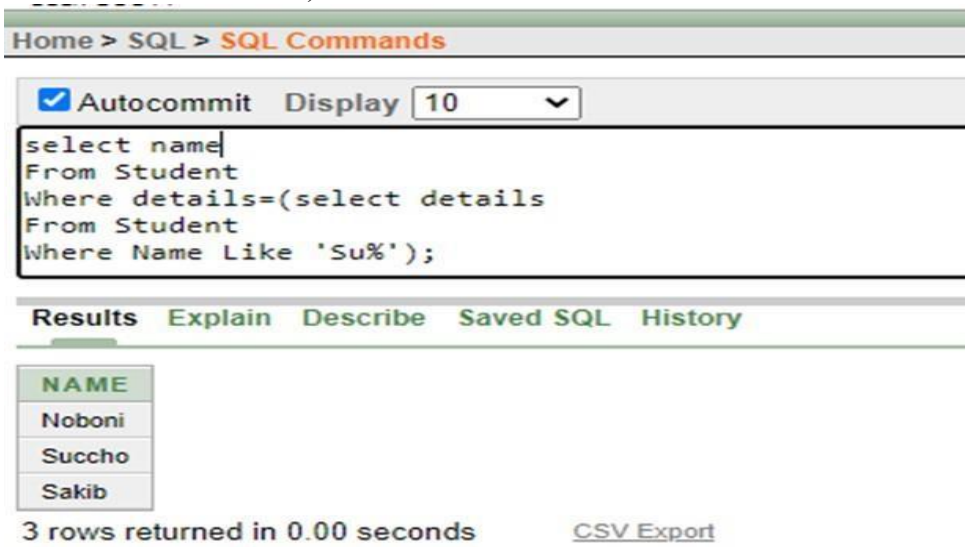

## Subquery -3

1. Select name From Student

Where details=(select details

From Student

Where Name Like 'Su%')

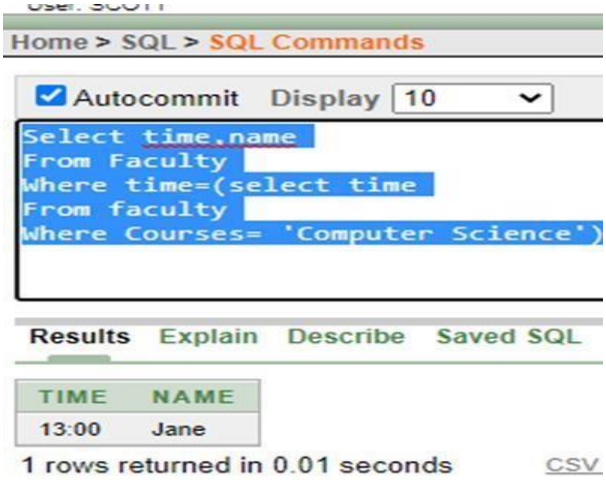



2. Select name,time From Faculty

Where time=(select time

From faculty

Where Course_name='COMPUTER SCIENCE')




3. Select name,assigned_students

From faculty

Where assigned_students=(select min(assigned_students)

From Faculty)

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 .

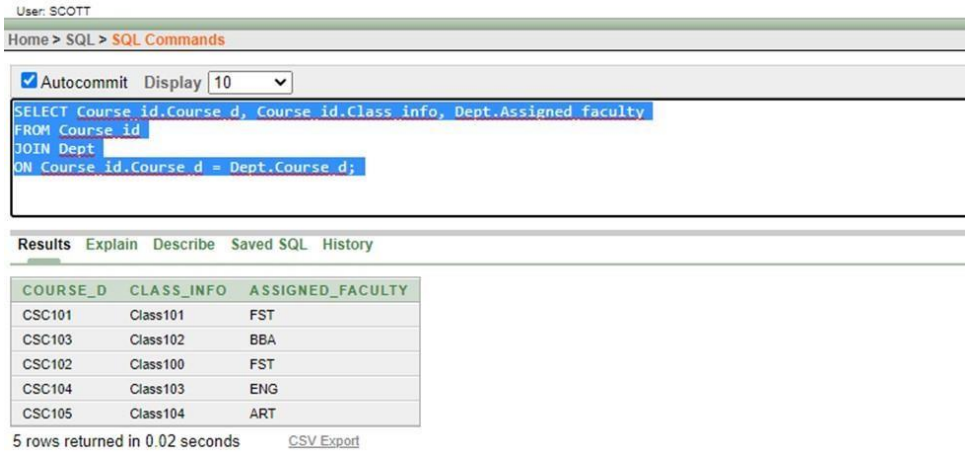
Ans :

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

FROM Course_id

JOIN Dept

ON Course_id.Course_d = Dept.Course_d;




3. Find all information for all courses.

Ans :

SELECT *

FROM dept d

JOIN Stdsid s

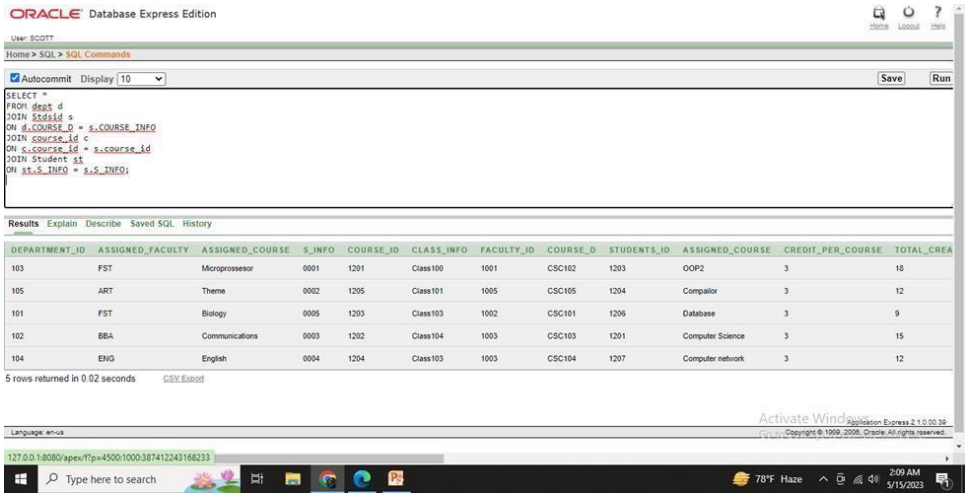
ON d.COURSE_D = s.COURSE_INFO

JOIN course_id c

ON c.course_id = s.course_id

JOIN Student st

ON st.S_INFO = s.S_INFO;




**view -3**

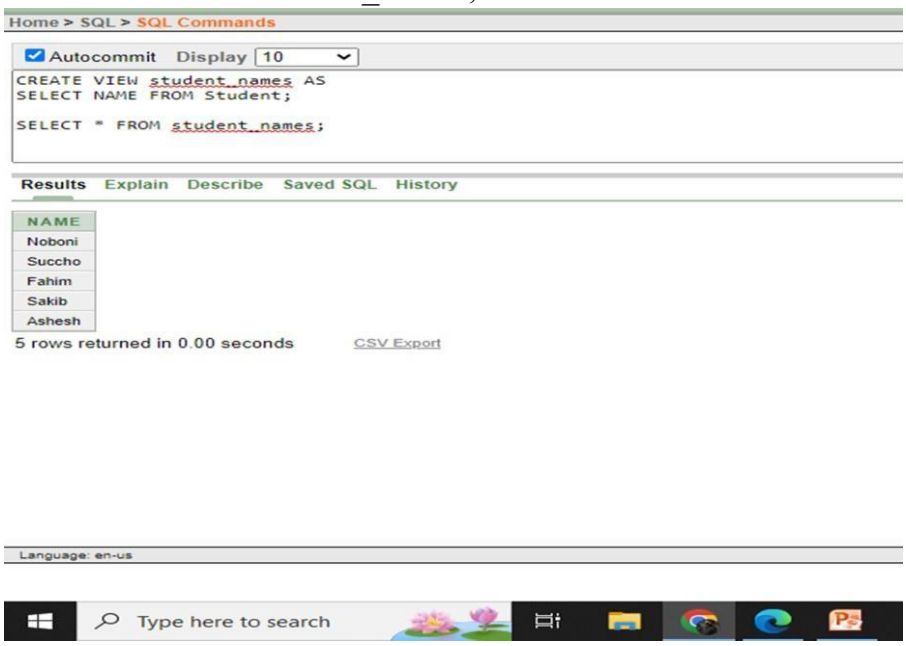
1. Create a view to get student names

Ans :

CREATE VIEW student_names AS

SELECT NAME FROM Student;

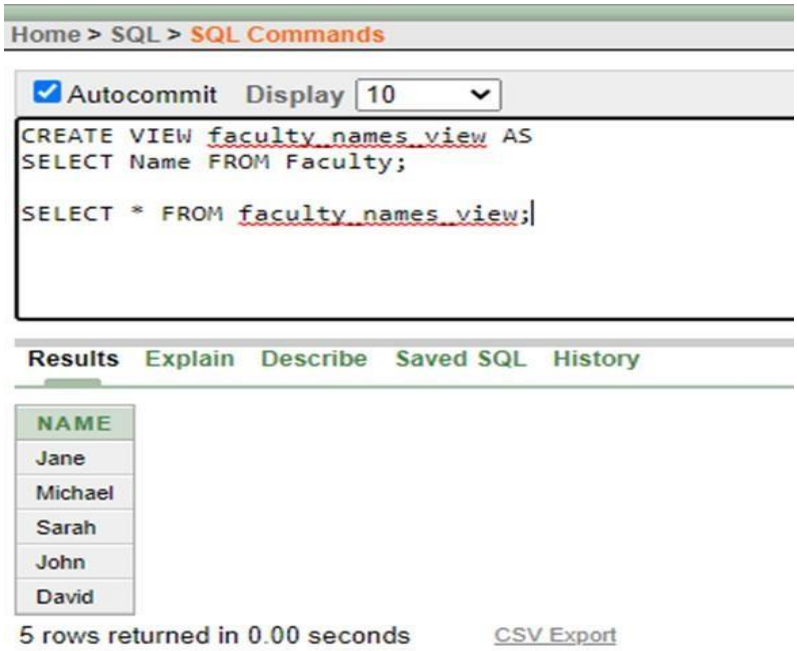
SELECT * FROM student_names;



2. Create a view to get Faculty names .

Ans : CREATE VIEW faculty_names_view AS
SELECT Name FROM Faculty;

SELECT * FROM faculty_names_view;




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;

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;

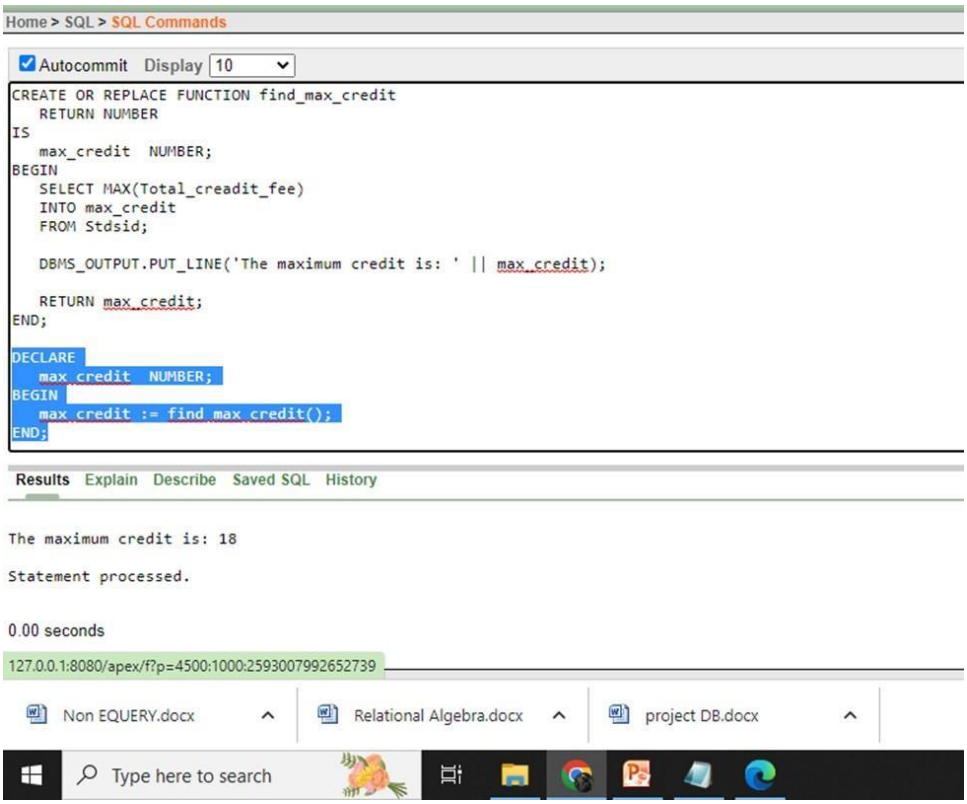
3.
CREATE SYNONYM Syn_Dept FOR Dept;

**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

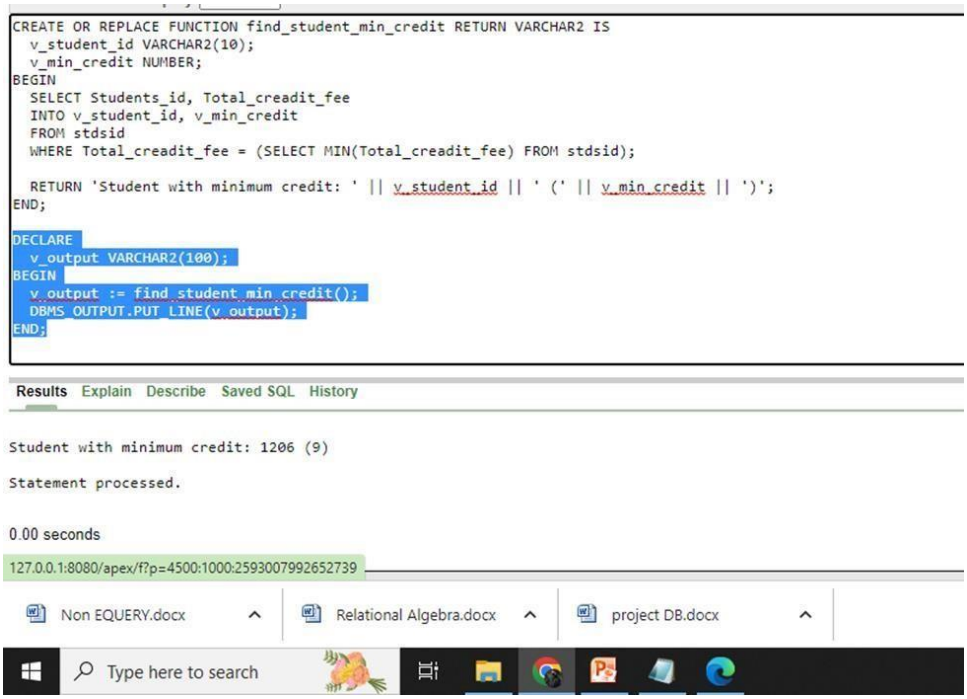
```
IS    max_credit
NUMBER;
BEGIN
  SELECT MAX(Total_creadit_fee)
  INTO max_credit
  FROM Stdsid;

  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;
```




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

:

```
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;
```



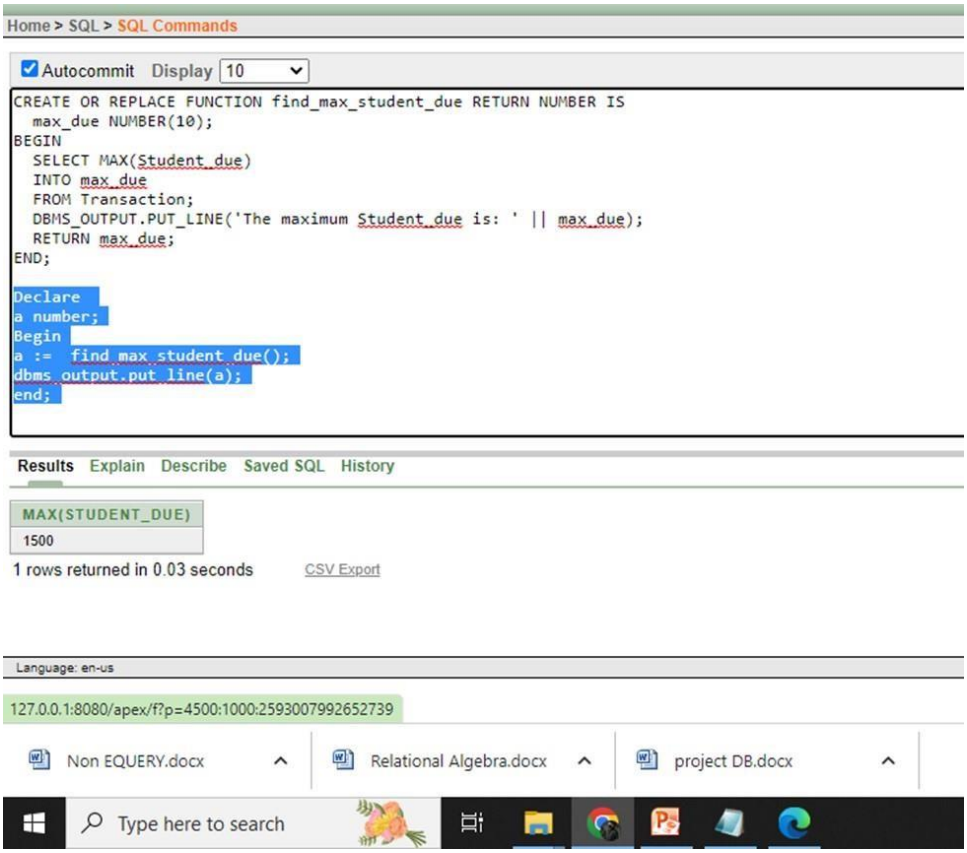

3. Write a function to find the maximum Studnet due.

Ans :

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

```
  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;
```



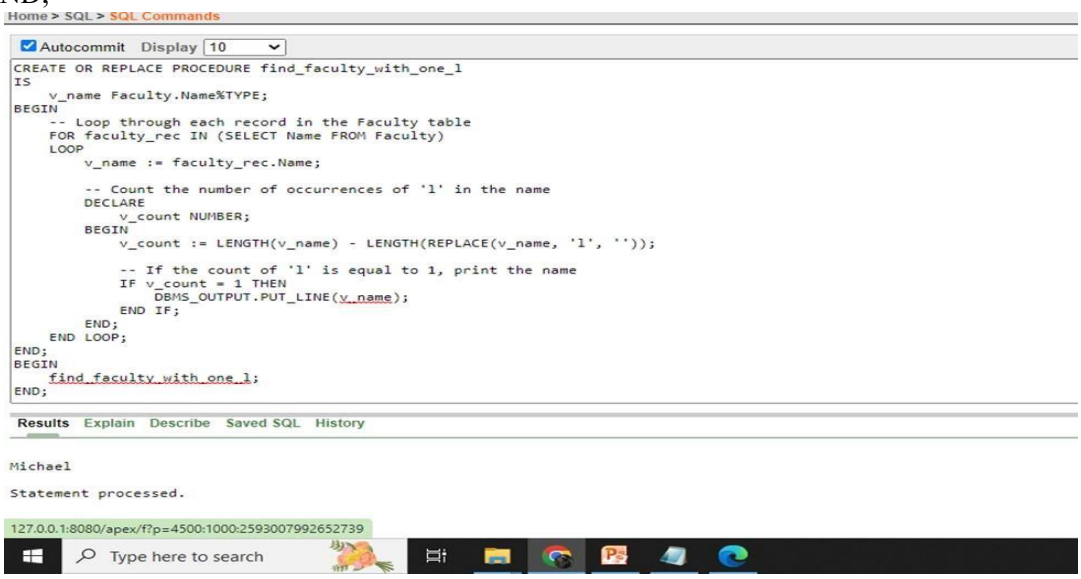

## -3 procedure

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
```

```
   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;
```




## 2. Find the Class with maximum Roomnumber.

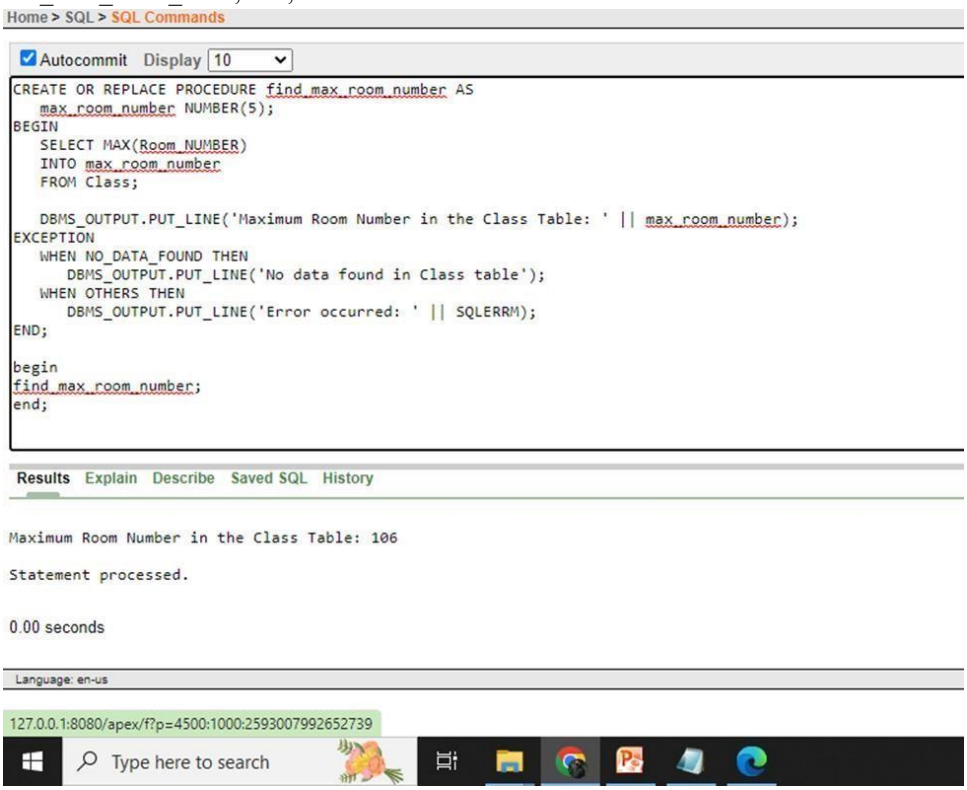
## Ans ;

```
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;
```

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;




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

```
  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
```

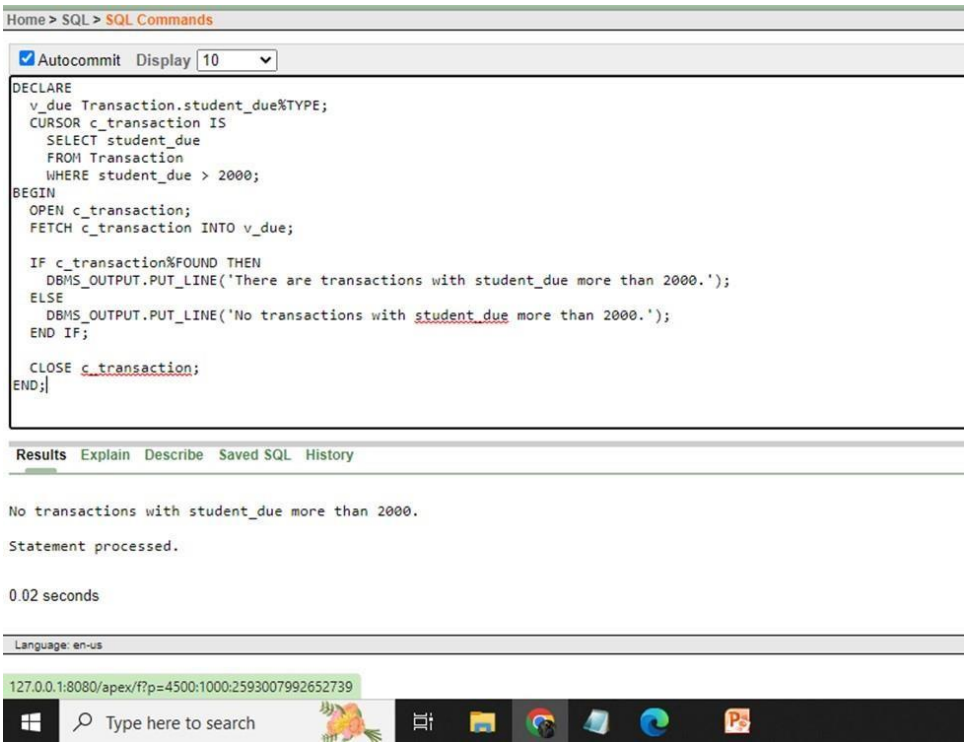
**-3 record -3 cursor**

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;
```

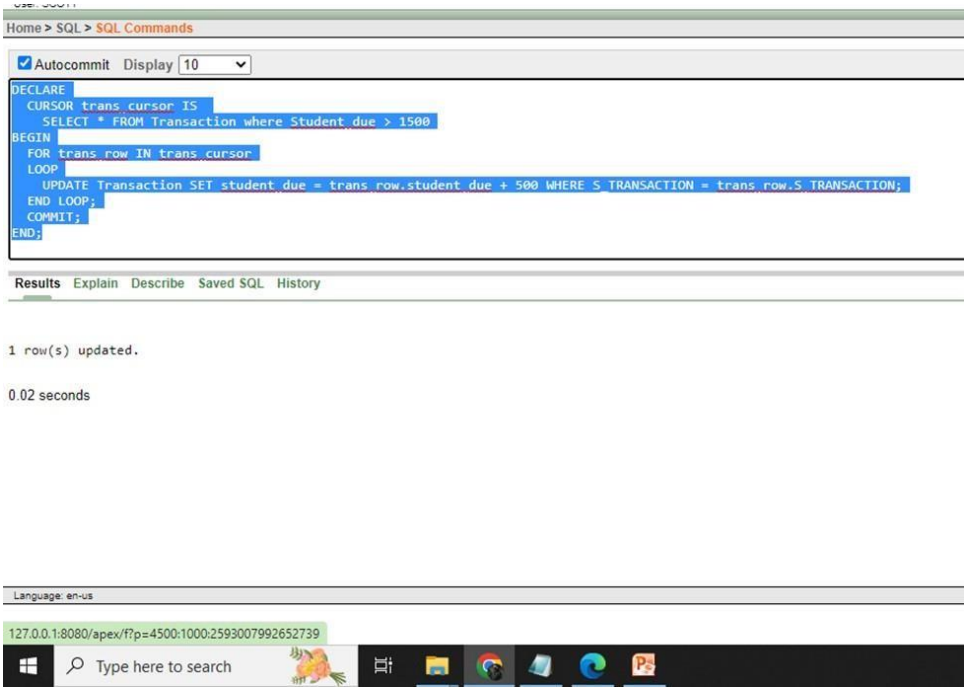
```
  CLOSE c_transaction;
END;
```




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

Ans :

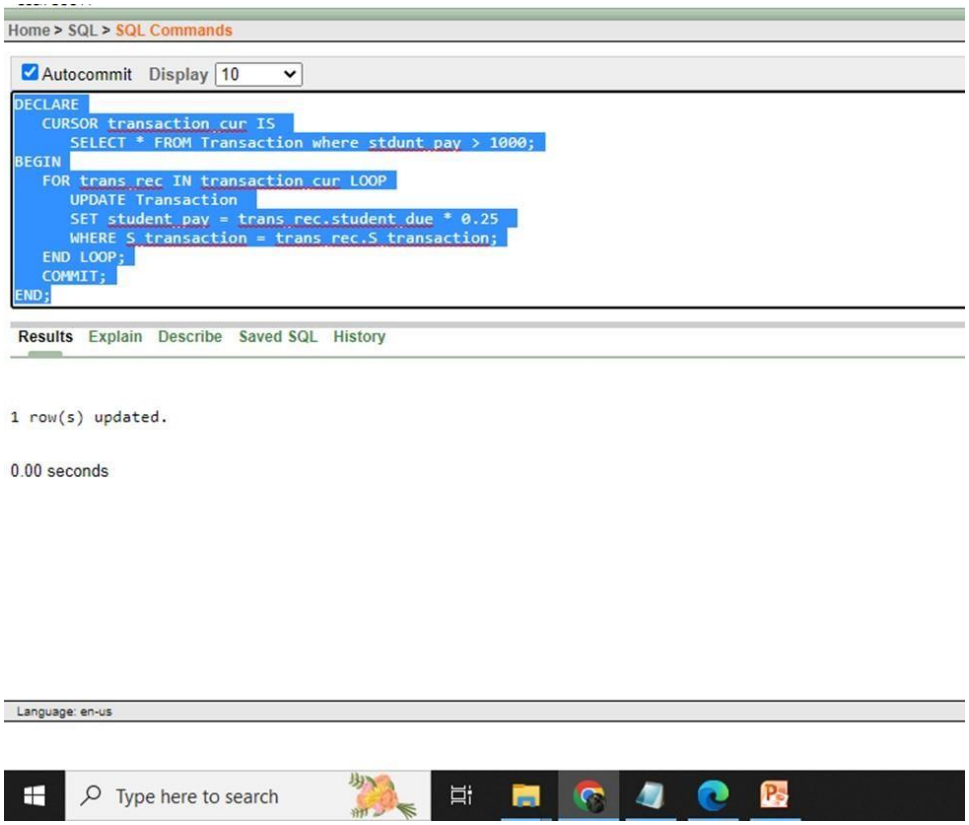
```
DECLARE
  CURSOR trans_cursor IS
    SELECT * FROM Transaction where Student_due > 1500
BEGIN
  FOR trans_row IN trans_cursor
  LOOP
    UPDATE Transaction SET student_due = trans_row.student_due + 500 WHERE
S_TRANSACTION = trans_row.S_TRANSACTION;
  END LOOP;
  COMMIT;
END;
```




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;
```




## -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;
```

```
   UPDATE Stdsid
   SET Total_creadit_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;
/
```

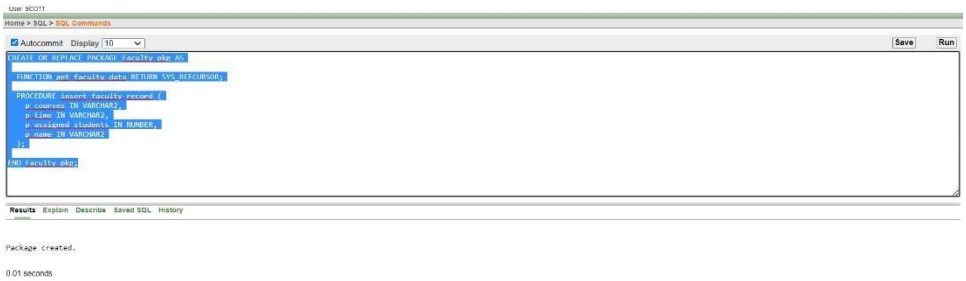
3.

```
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;
```




2.

CREATE OR REPLACE PACKAGE Student_Pkg AS

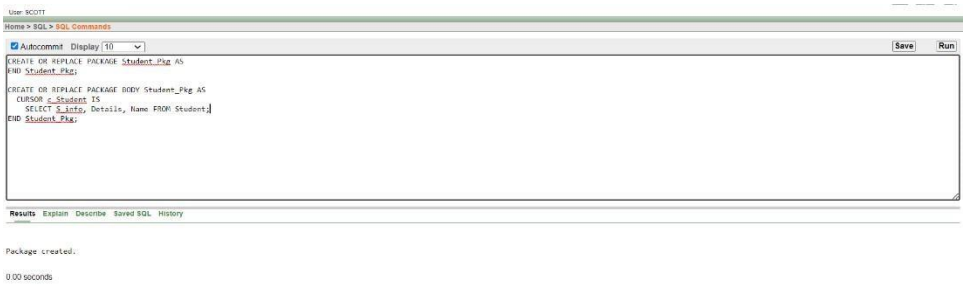
END Student_Pkg;

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;

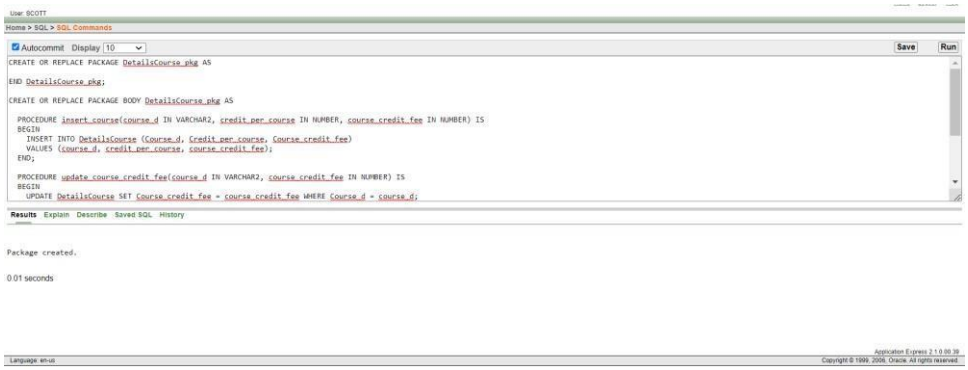
CREATE OR REPLACE PACKAGE BODY DetailsCourse_pkg AS

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)

```
  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;
```




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

1. Display all the info whose Student_Pay is greater than 1000.

Ans : σ Student_Pay > 1000(Transaction).

2.Display all the information from students whose details are in CSE. Ans : σ details = CSE(Student).

3. Display all the information from a student whose name starts with S.
   Ans : σ S_name like 'S%' (Student).

4. Display all S-Transactions whose student pay is greater than 1000. Ans :
   $\prod$ S-Transaction [$\sigma$ Student_Pay > 1000(Transaction)].

5. Display all the room numbers and sections whose section is B from the class table.
   Ans : $\prod$ 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.