

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

Introduction	3
Project Proposal	3
Use Case Diagram	3
Activity Diagram	4
Class Diagram	5
Interface Design and description	6
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
Scenario description	10 ER
Diagram	11
Normalization	11
Schema Diagram	17
Table Creation	17
Data Insertion	28
SQL Query.....	36
Relational Algebra	56
Conclusion	57

Course Registration Management System

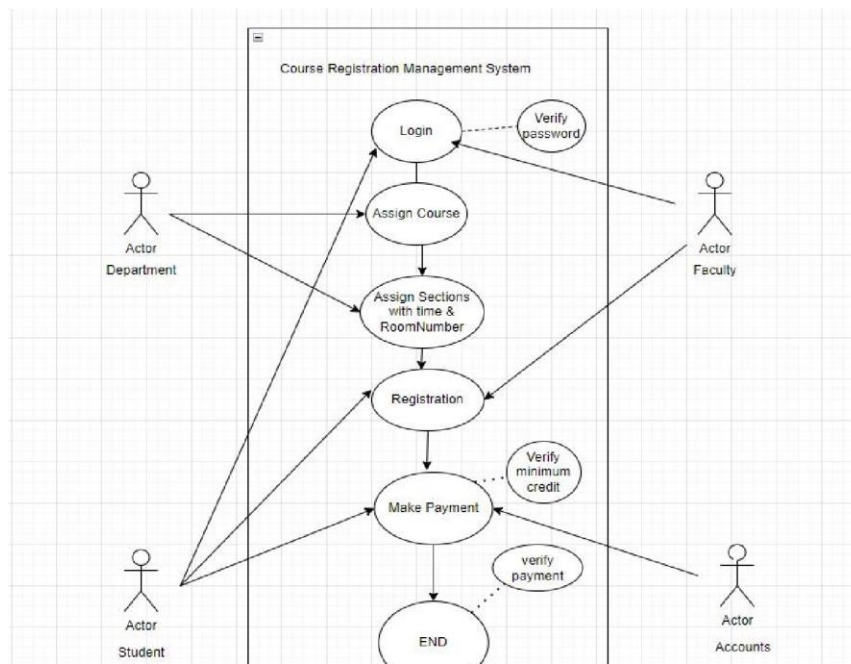
Introduction:

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.

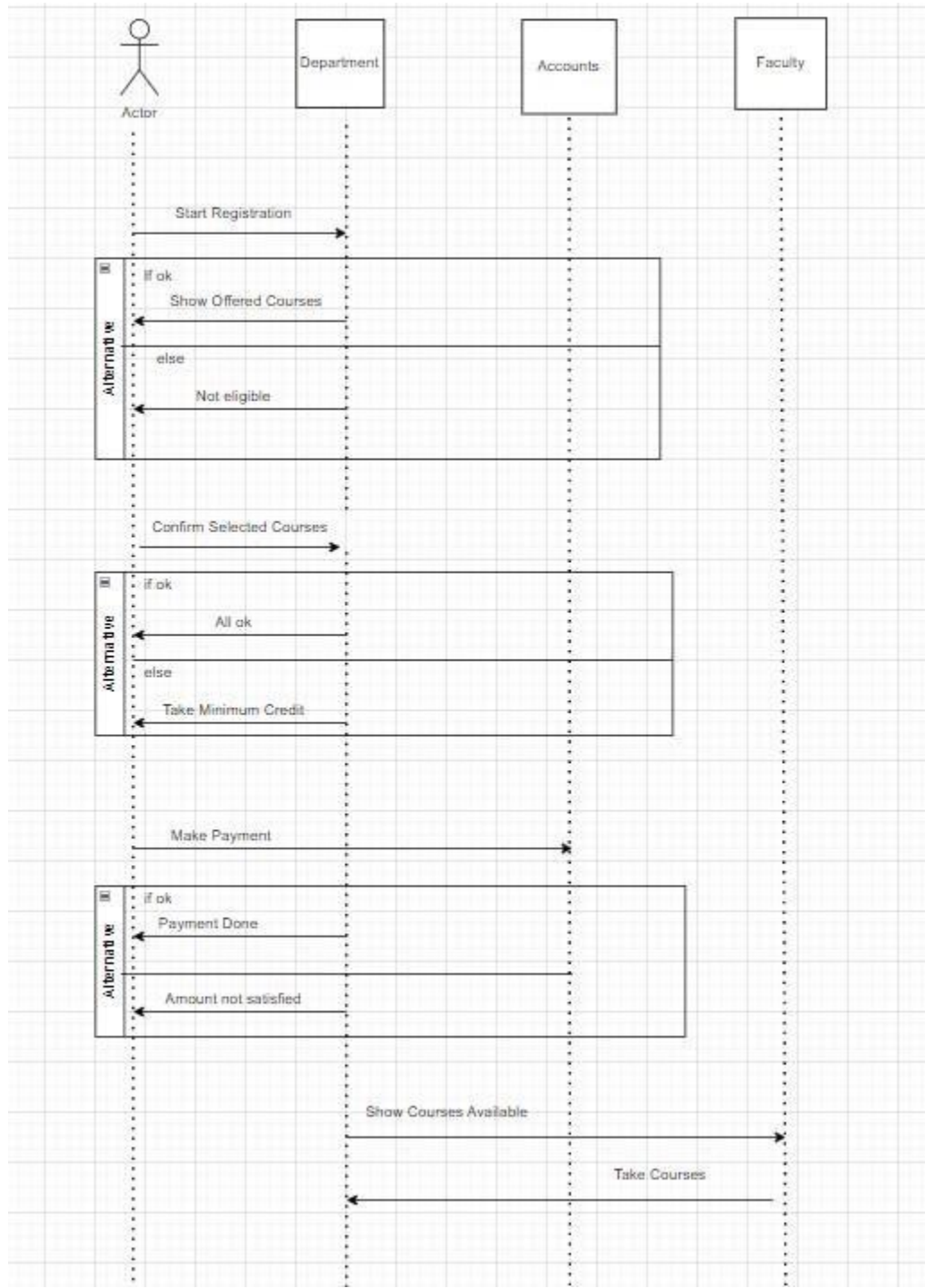
Use Case Diagram:



Course Registration Management System

Activity Diagram:

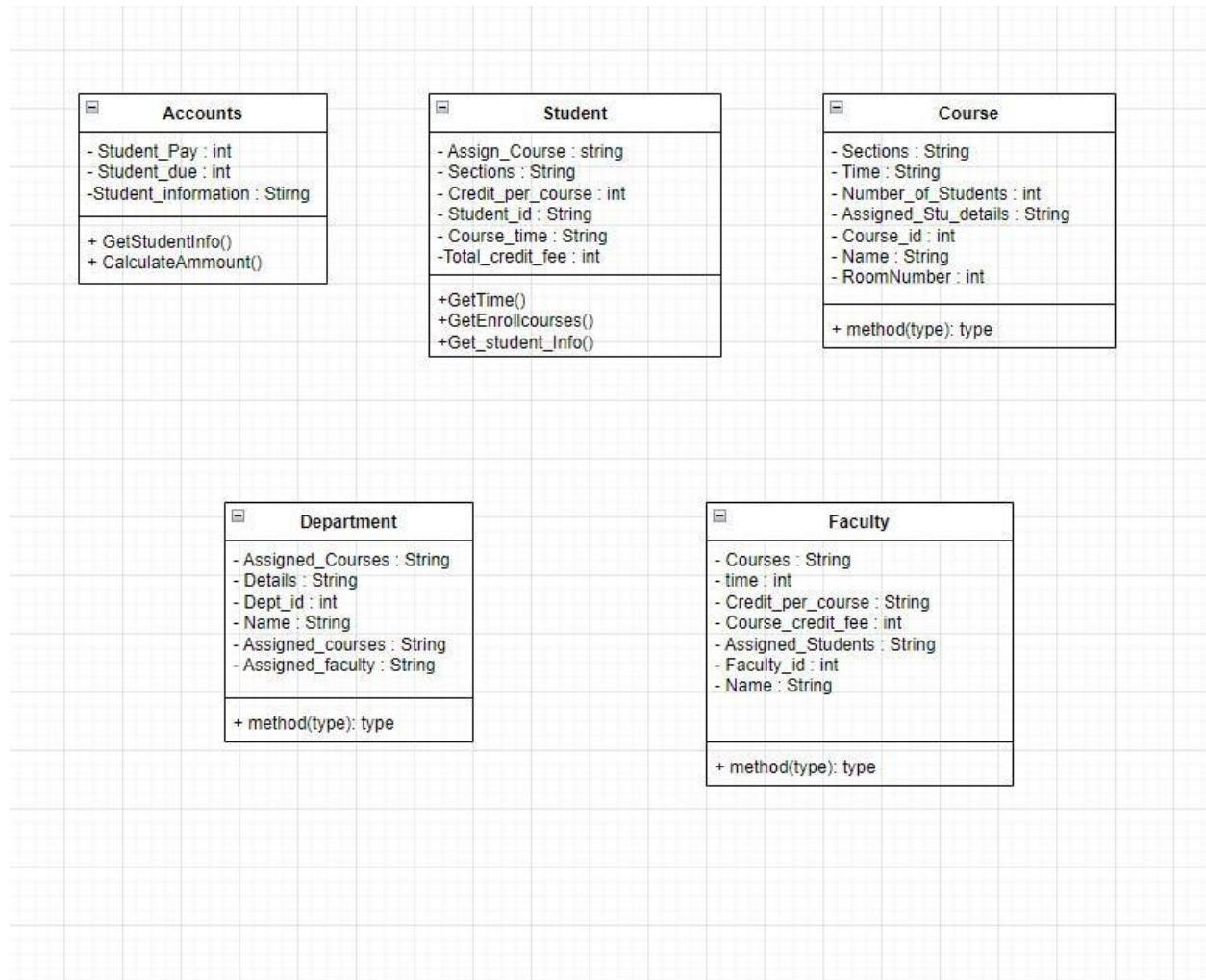
Course Registration Management System



Class Diagram:

Course Registration Management System

Figure: Class Diagram



Interface Design and Description:

Course Registration Management System

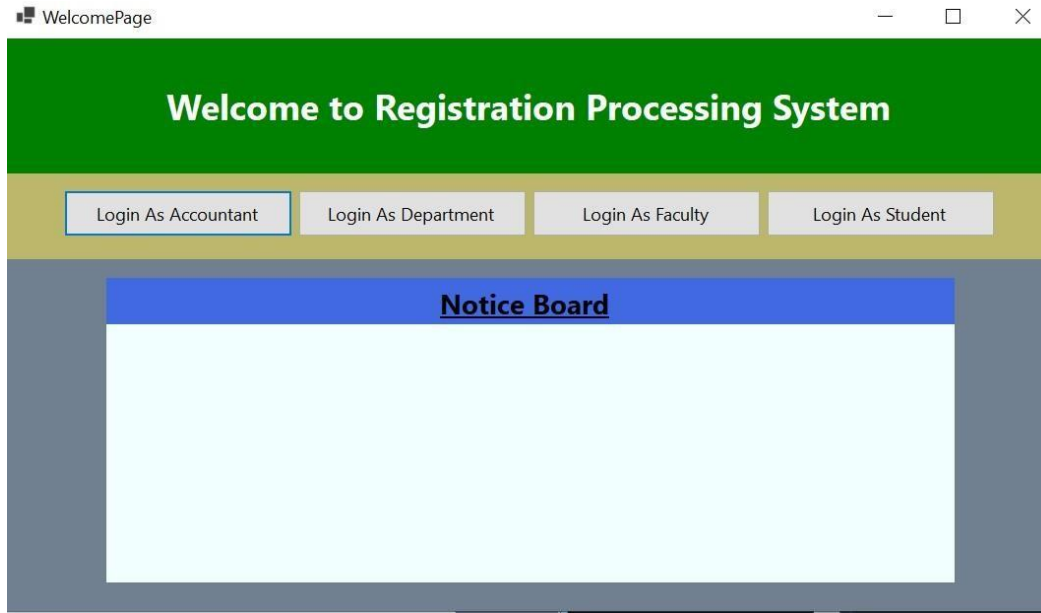


Figure 1: Welcome_Page

After opening the app's 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.

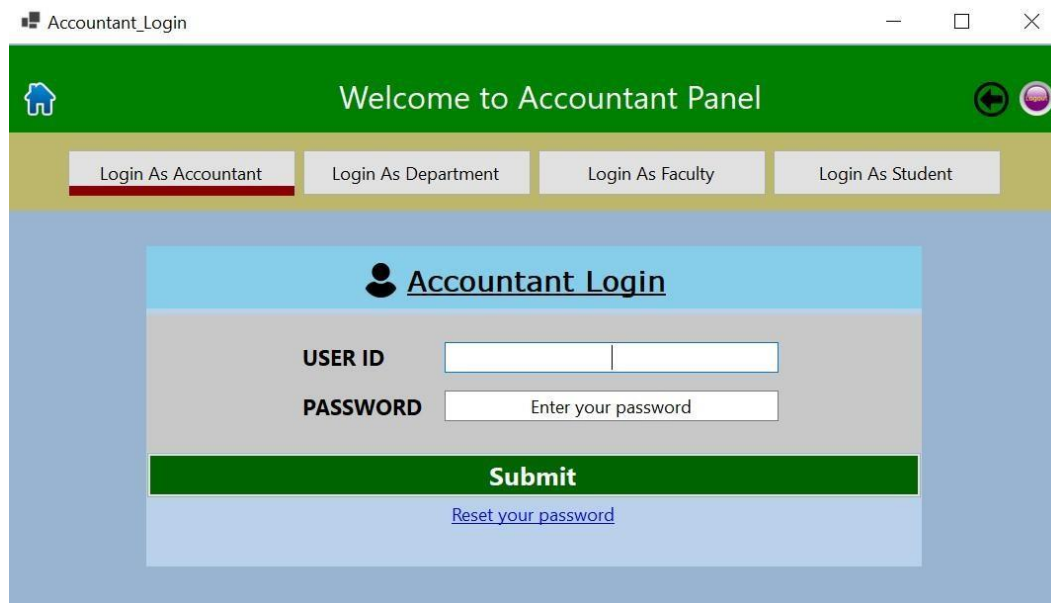


Figure 2: Accountant Login

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.

Course Registration Management System

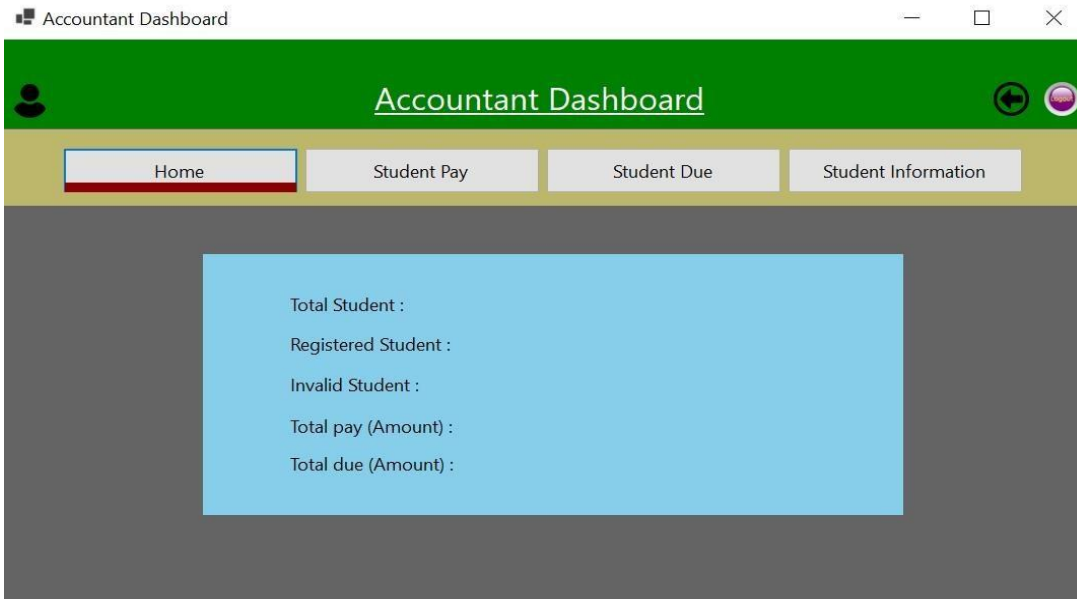


Figure 3: Accountant Dashboard

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.

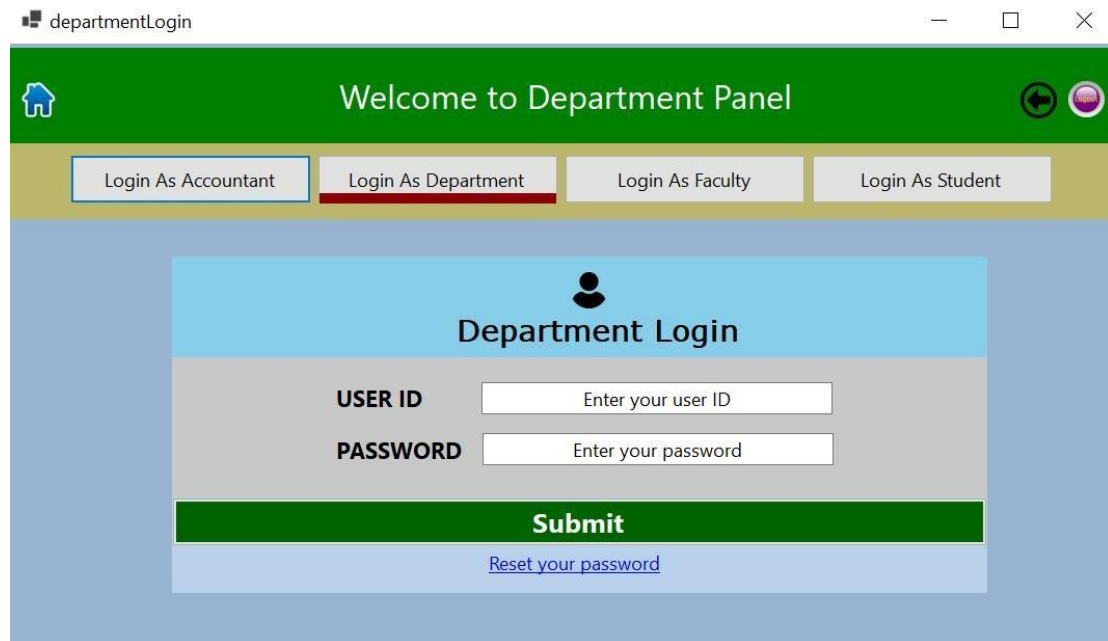


Figure 4: Department Login

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

Course Registration Management System

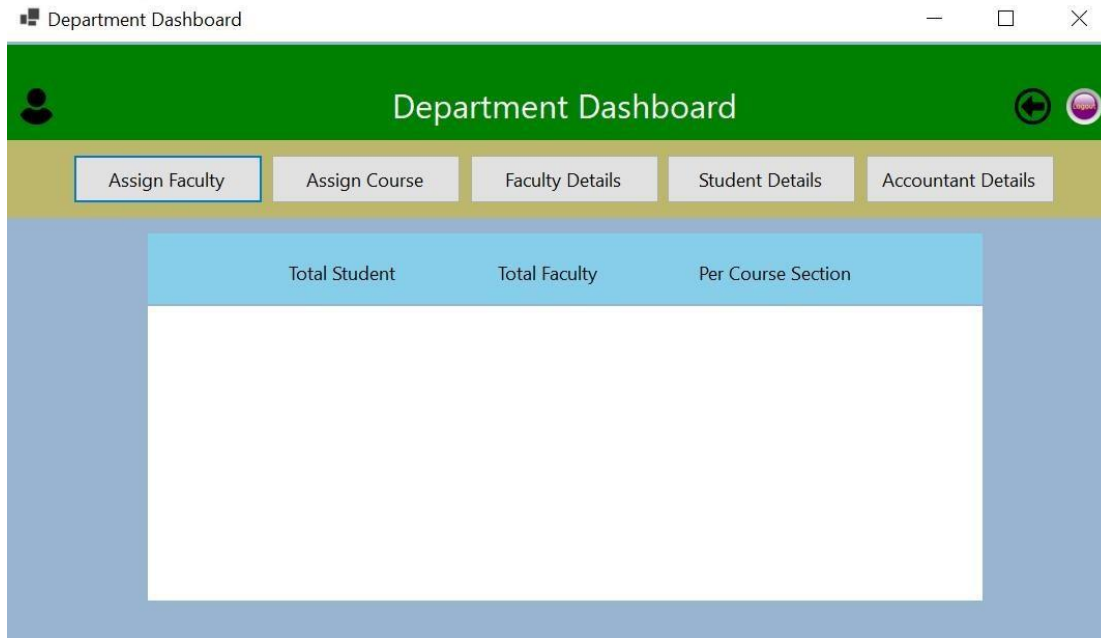


Figure 5: Department Dashboard

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.

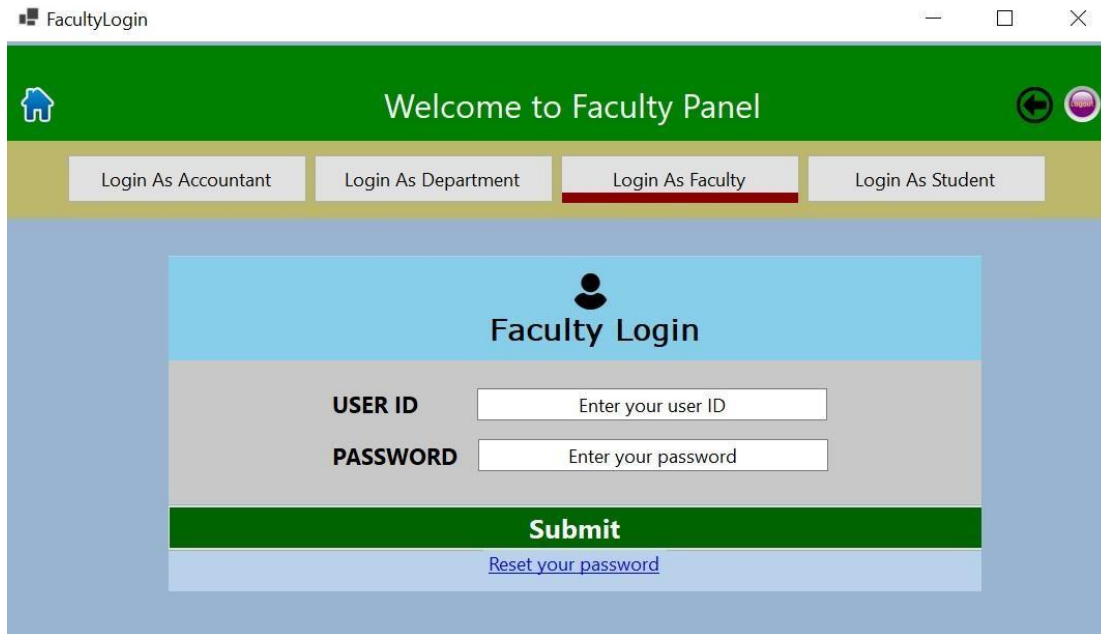


Figure 6: Faculty Login

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.

Course Registration Management System

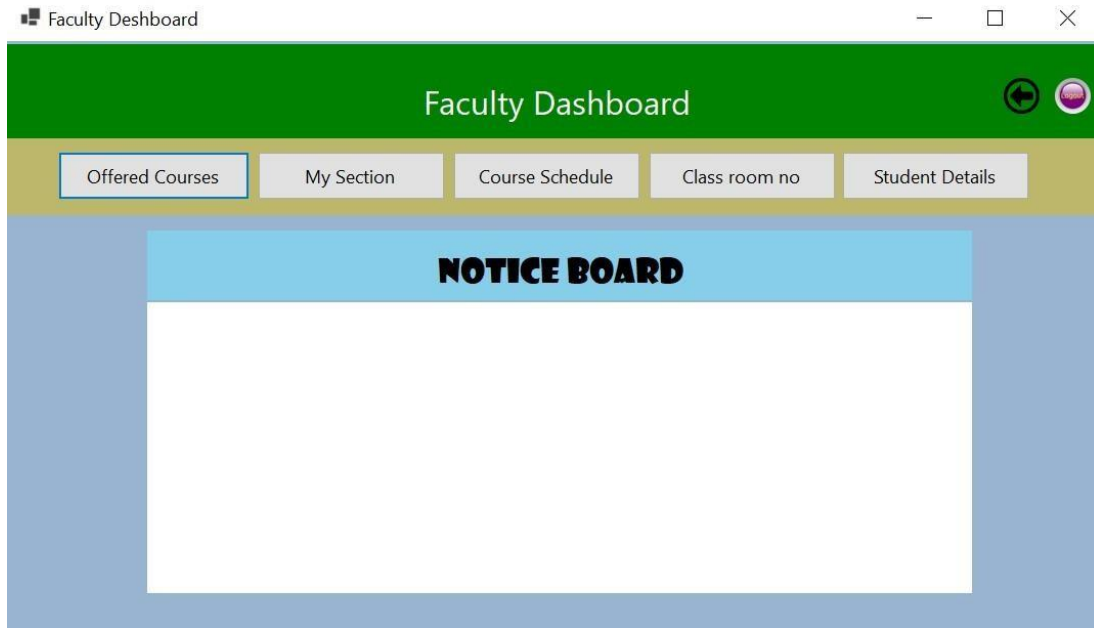


Figure 7: Faculty Dashboard

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.

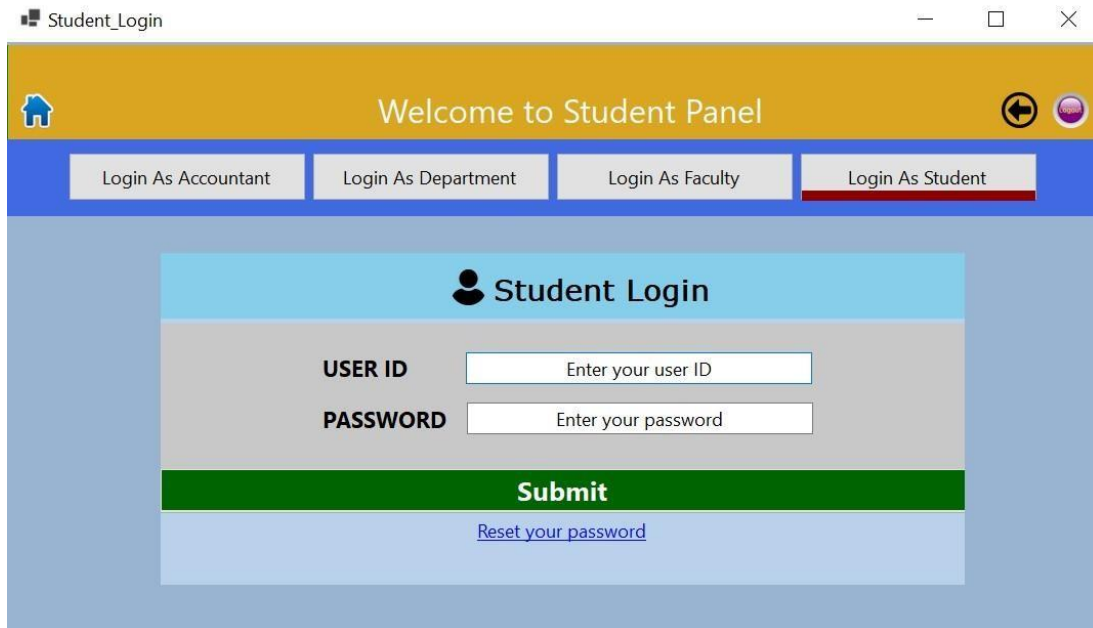


Figure 8: Student Login

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

Course Registration Management System

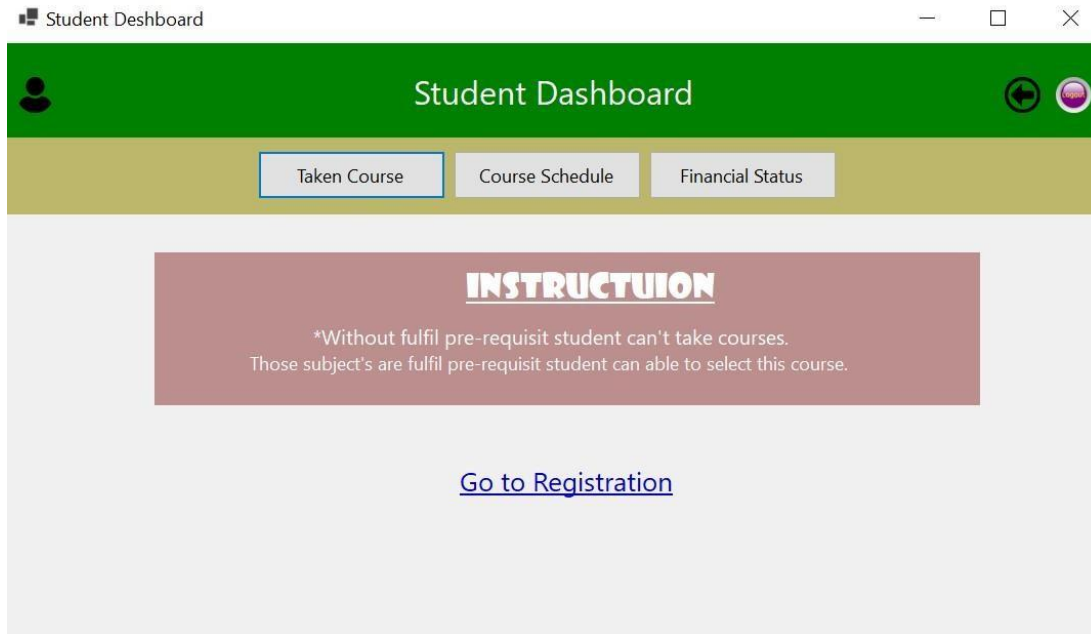


Figure 9: Student Dashboard

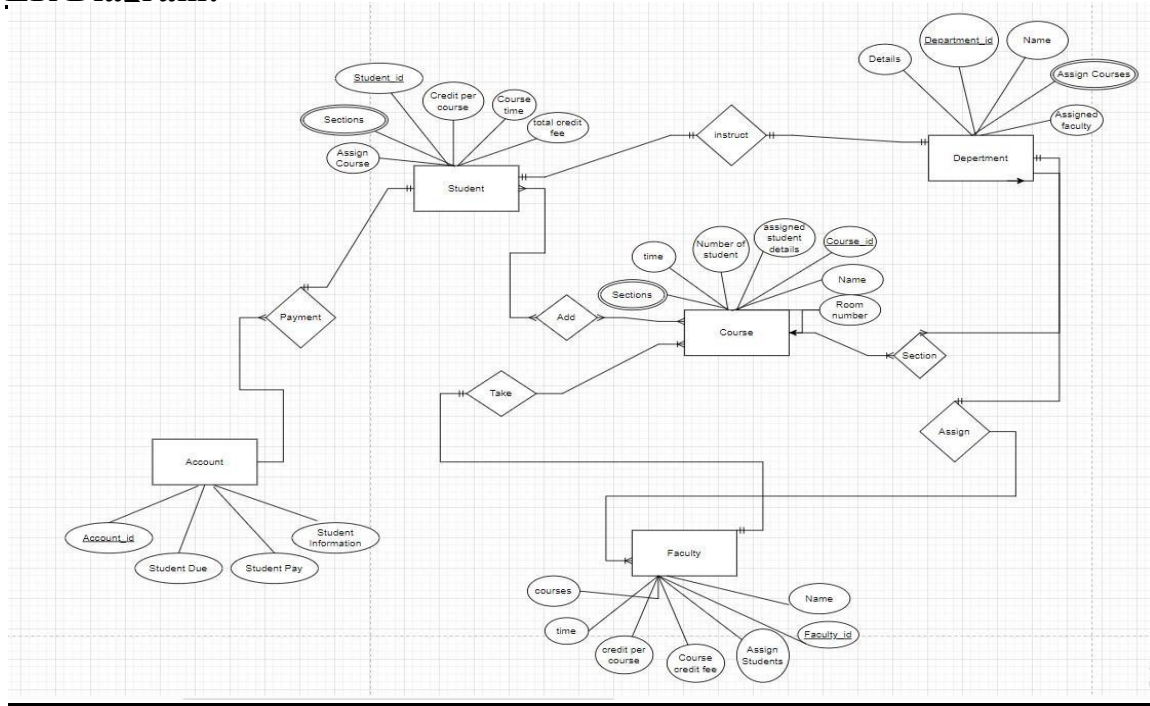
In the student dashboard there are 3 buttons. By pressing Taken Course user can see the courses which they are taken. Course schedule for seeing the schedule. Financial status for knowing the payable amount after confirming the registration process.

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.

Course Registration Management System

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.

Students_id, Assign Course, Credit per course, total credit fee.

Course_info, Sections, Course time.

Course Registration Management System

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.

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 Registration Management System

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.

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.

Course Registration Management System

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.

Course Registration Management System

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.**

Course Registration Management System

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.

Course Registration Management System

Schema Diagram:

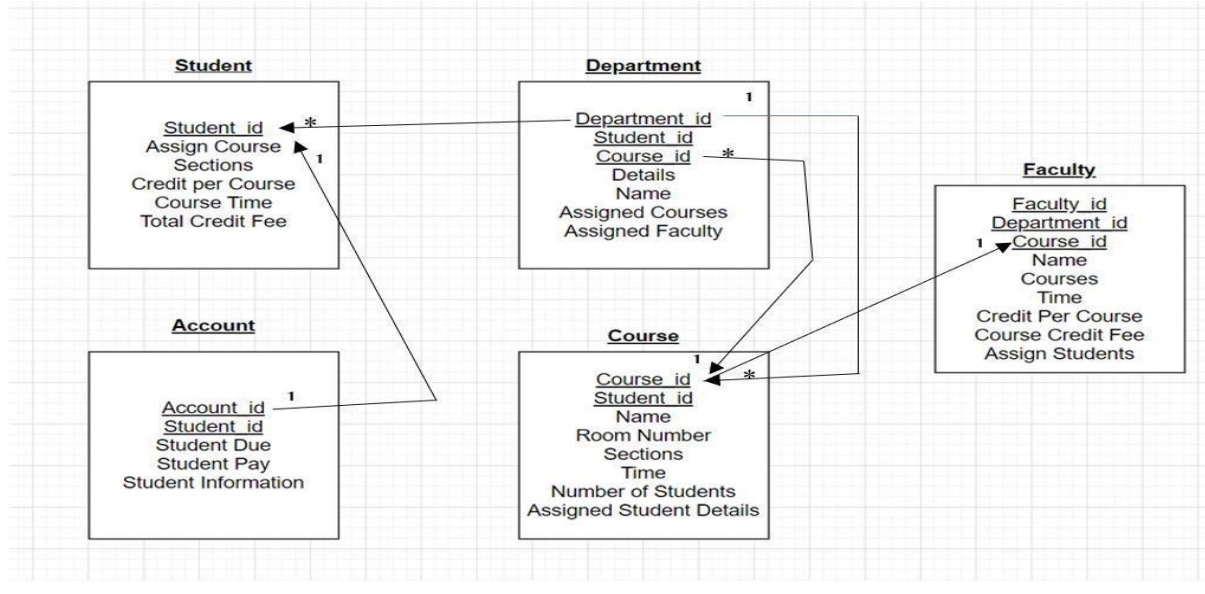
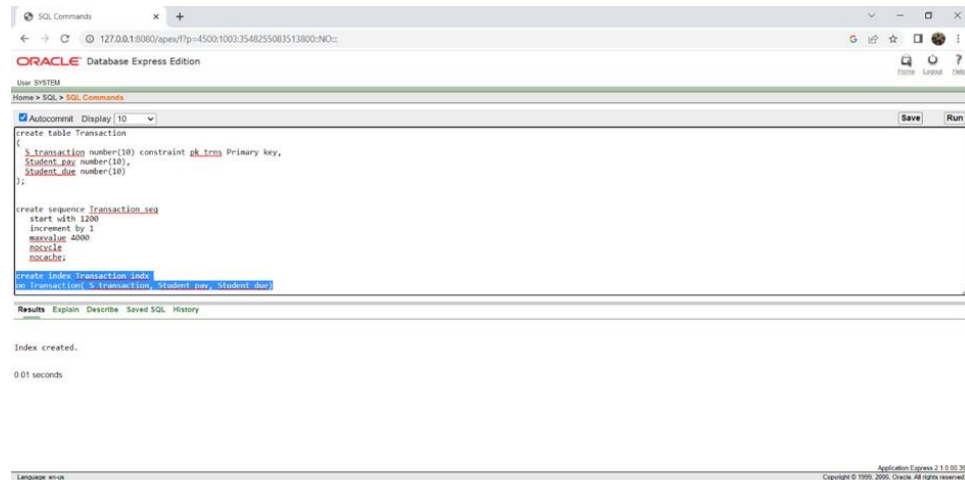


Table Creation:

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

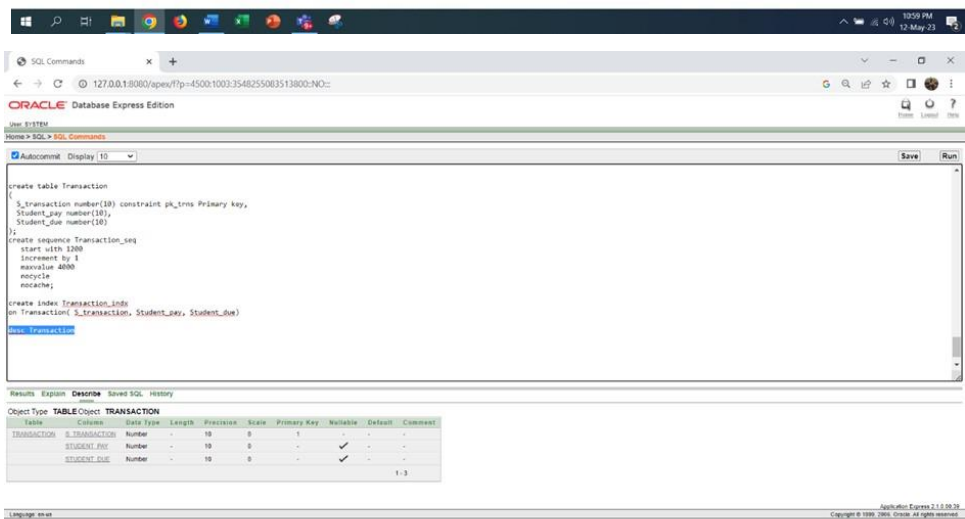


```
SQL Commands
127.0.0.1:8080/apex/?p=4500:1003:3548255063513800:NO:~
ORACLE Database Express Edition
User SYSTEM
Home > SQL > SQL Commands
Autocommit Display 10 Save Run
create table Transaction
(
  s_transaction number(10) constraint pk_trns Primary key,
  student_pay number(10),
  student_due number(10)
);

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);

Results Explain Describe Saved SQL History
Index created.
0.01 seconds
```



```
SQL Commands
127.0.0.1:8080/apex/?p=4500:1003:3548255063513800:NO:~
ORACLE Database Express Edition
User SYSTEM
Home > SQL > SQL Commands
Autocommit Display 10 Save Run
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);

Desc Transaction

Results Explain Describe Saved SQL History
Object Type TABLE Object TRANSACTION
Table Columns Data Type Length Precision Scale Primary Key Nullable Default Comment
TRANSACTION s_transaction Number 10 0 - 1 - -
STUDENT_PAY Number 10 0 - - - -
STUDENT_DUE Number 10 0 - - - -
1-3
```

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

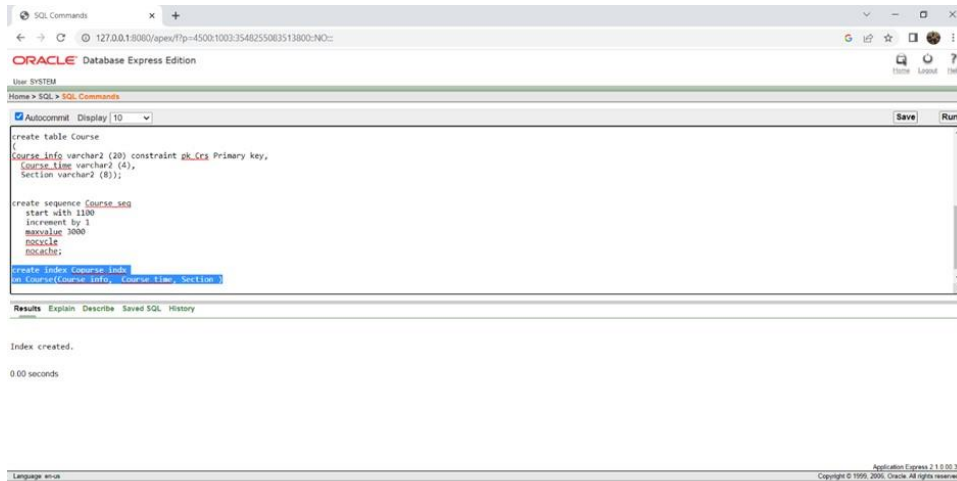
by 1 maxvalue 3000

nocycle nocache; create

index Copurse_indx on

Course Registration Management System

Course(Course_info,
Course_time, Section)

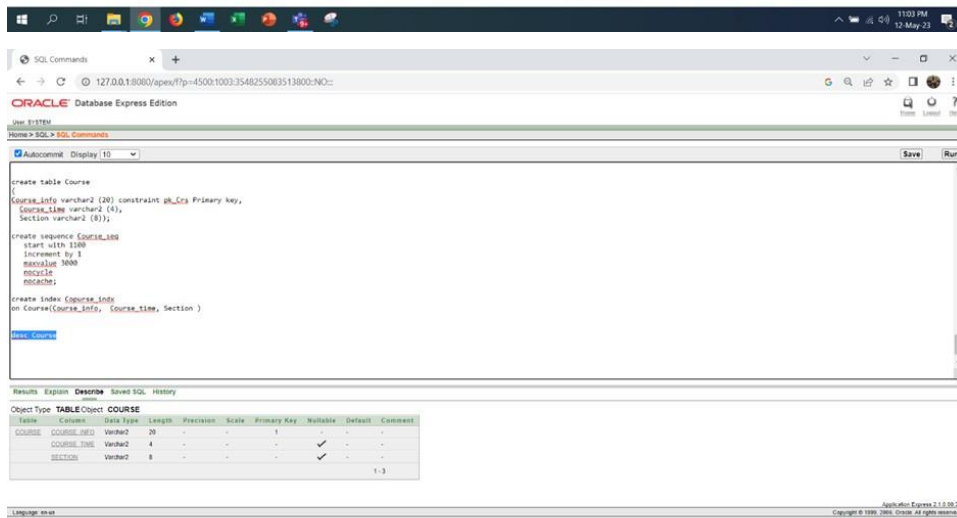


```
SQL Command
127.0.0.1:8080/apex/?p=4500:1003:3548255083513800:NO::
ORACLE Database Express Edition
Use SYSTEM
Home > SQL > SQL Commands
Autocommit Display 10 Save Run
create table Course
(
  Course_info varchar2 (20) constraint pk_Crs Primary key,
  Course_time varchar2 (4),
  Section varchar2 (8)),

create sequence Course_seq
start with 1100
Increment by 1
maxvalue 3000
nocycle
nocalendar;

create index Course_index
on Course(Course_info, Course_time, Section )

Results Explain Describe Saved SQL History
Index created.
0.00 seconds
Language menu
Application Express 2.1.10.30
Copyright © 1999, 2005, Oracle. All rights reserved.
```



```
SQL Command
127.0.0.1:8080/apex/?p=4500:1003:3548255083513800:NO::
ORACLE Database Express Edition
Use SYSTEM
Home > SQL > SQL Commands
Autocommit Display 10 Save Run
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
nocalendar;

create index Course_index
on Course(Course_info, Course_time, Section )

Results Explain Describe Saved SQL History
Object Type TABLE Object COURSE
Table Column Data Type Length Precision Scale Primary Key Nullable Default Comment
COURSE Course_info Varchar2 20 - - - 1 - -
COURSE Course_time Varchar2 4 - - - - - -
COURSE Section Varchar2 8 - - - - - -
1-3
Language menu
Application Express 2.1.10.30
Copyright © 1999, 2005, Oracle. All rights reserved.
```

Table Student

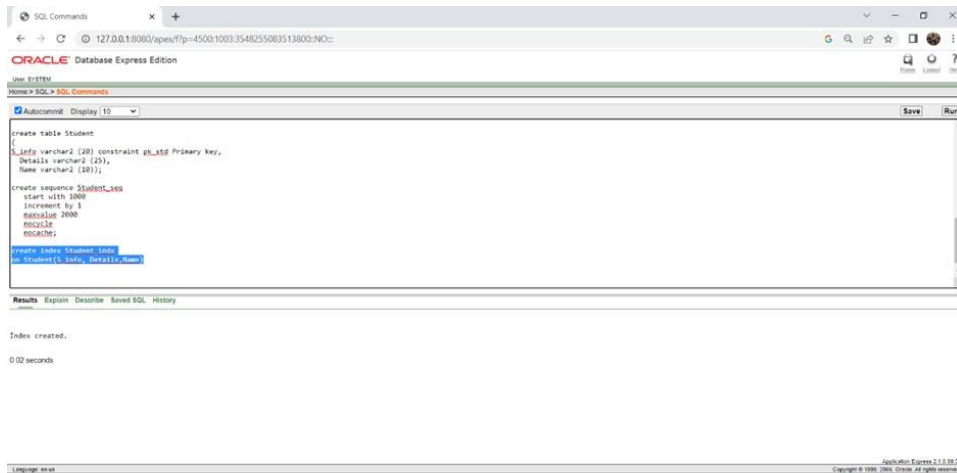
create table Student

(
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

Course Registration Management System

nocache; create index

Student_idx on Student(S_info,
Details,Name)



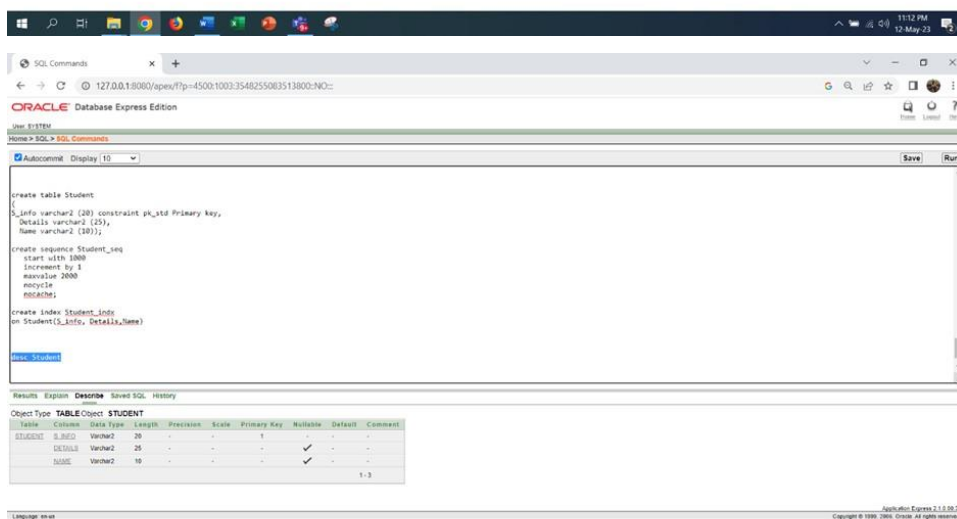
The screenshot shows the SQL Developer interface with the following SQL commands entered in the 'SQL Commands' window:

```
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_idx
on Student(S_info, Details, Name);
```

The 'Results' pane shows the message: 'Index created.' and the execution time is '0.02 seconds'.



The screenshot shows the SQL Developer interface with the same SQL commands as above. Below the commands, the 'Results' pane displays the table structure for the 'STUDENT' table:

Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
STUDENT S_INFO	Varchar2	20	-	-	1	-	-	-
DETAILS	Varchar2	25	-	-	-	✓	-	-
NAME	Varchar2	10	-	-	-	✓	-	-

The 'Results' pane also shows the message: 'Index created.' and the execution time is '0.02 seconds'.

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

Course Registration Management System

maxvalue 3000 nocycle nocache;

create index Class_idx on Class(Class_info,
room_number,Sections)

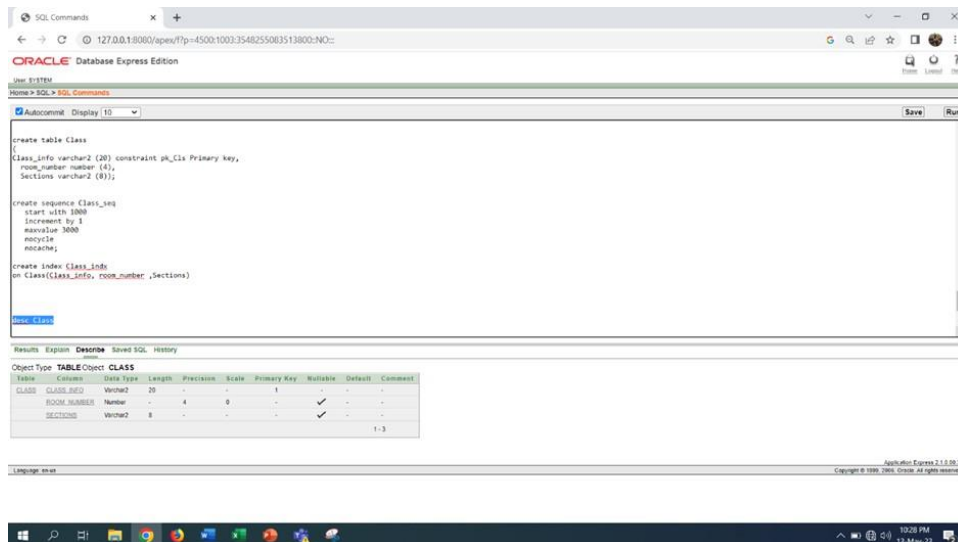
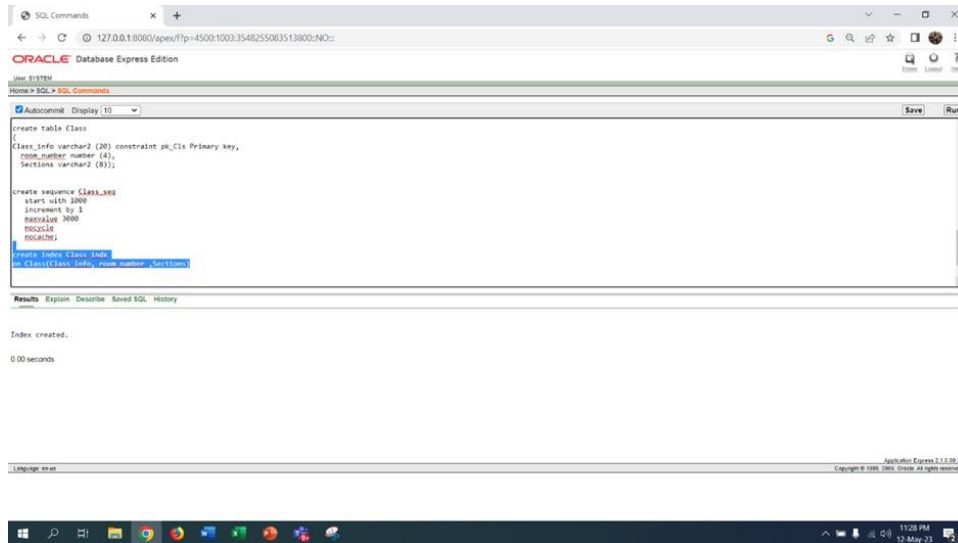


Table DetailsCourse

create table DetailsCourse

(

Course Registration Management System

Course_d varchar2 (20) constraint pk_crstdtls Primary key,

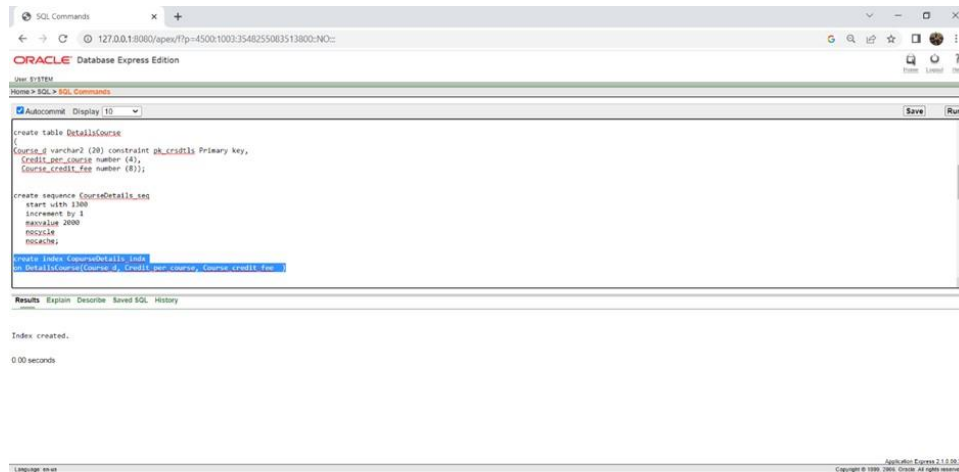
Credit_per_course number (4), Course_credit_fee number (8));

create sequence CourseDetails_seq start with 1300 increment by

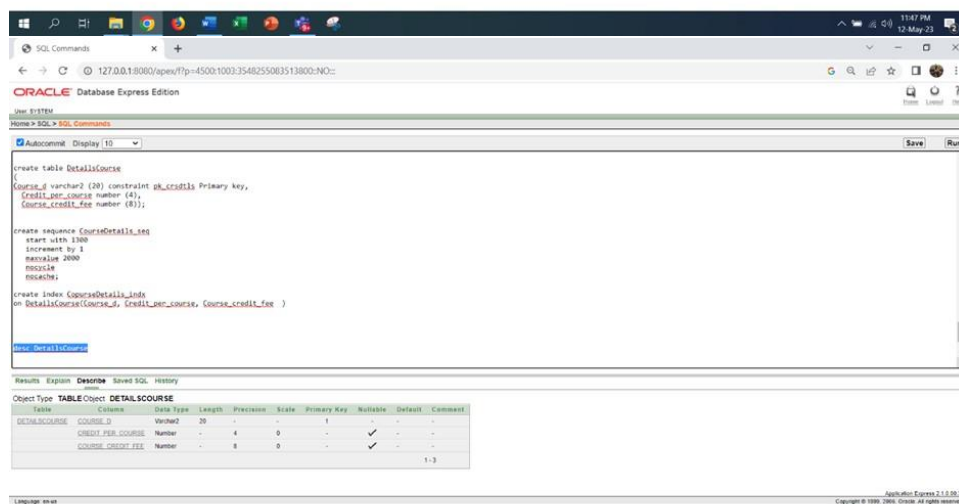
1 maxvalue 2000 nocycle nocache; create index

CopurseDetails_indx on DetailsCourse(Course_d,

Credit_per_course, Course_credit_fee)



```
SQL Commands
127.0.0.1:8080/apex/?p=4500:1003:3548255083513800:NO-
ORACLE Database Express Edition
User: SYS
Home > SQL > SQL Commands
Autocommit: Display 10
Save Run
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 );
Results Explain Describe Saved SQL History
Index created.
0.00 seconds
Language: sql
Application Express 2.1.1.0.0.20
Copyright © 1999, 2009, Oracle. All rights reserved.
```



```
SQL Commands
127.0.0.1:8080/apex/?p=4500:1003:3548255083513800:NO-
ORACLE Database Express Edition
User: SYS
Home > SQL > SQL Commands
Autocommit: Display 10
Save Run
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 );
View DetailsCourse
Results Explain Describe Saved SQL History
Object Type: TABLE Object: DETAILSCOURSE
Table
Columns Data Type Length Precision Scale Primary Key Nullable Default Comment
DETAILSCOURSE Course_d Varchar2 20 - - - 1 - -
DETAILSCOURSE Credit_per_course Number 4 0 - - - - -
DETAILSCOURSE Course_credit_fee Number 8 0 - - - - -
1-3
Language: sql
Application Express 2.1.1.0.0.20
Copyright © 1999, 2009, Oracle. All rights reserved.
```

Table Faculty

create table Faculty

(

Faculty_id number (8) constraint pk_Flt Primary key,

Courses varchar2 (30),

Course Registration Management System

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_idx
on Faculty(Faculty_id,Courses, Time, Assigned_students, Name)

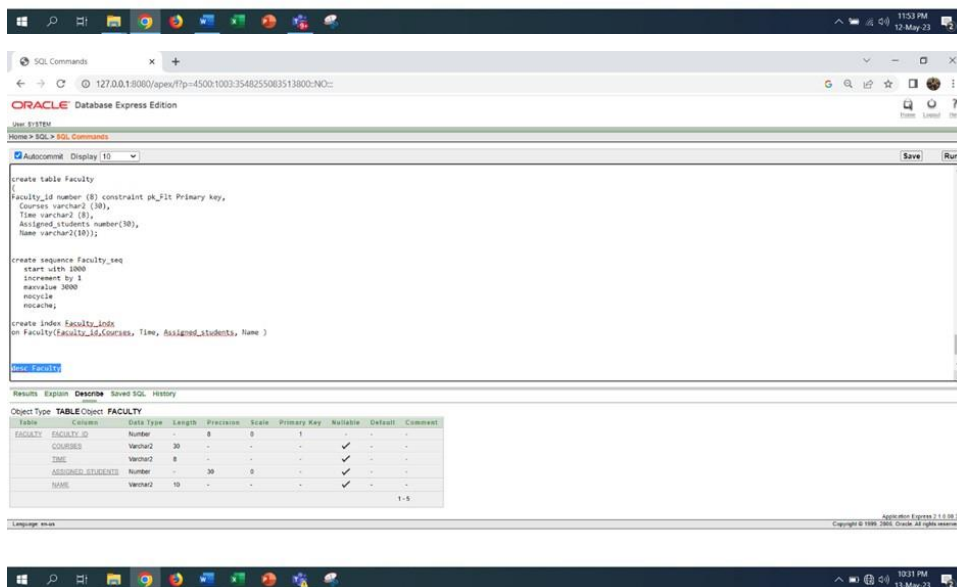
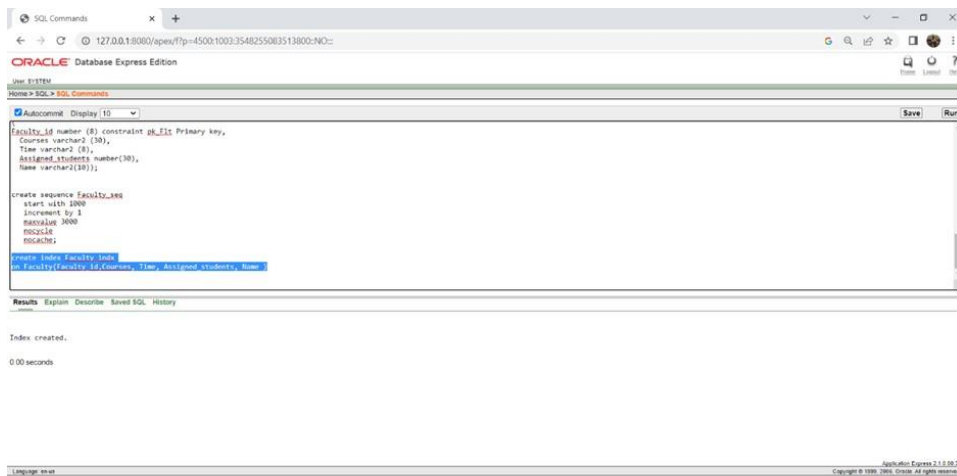


Table Accountant

Create table Accountant(

Course Registration Management System

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)

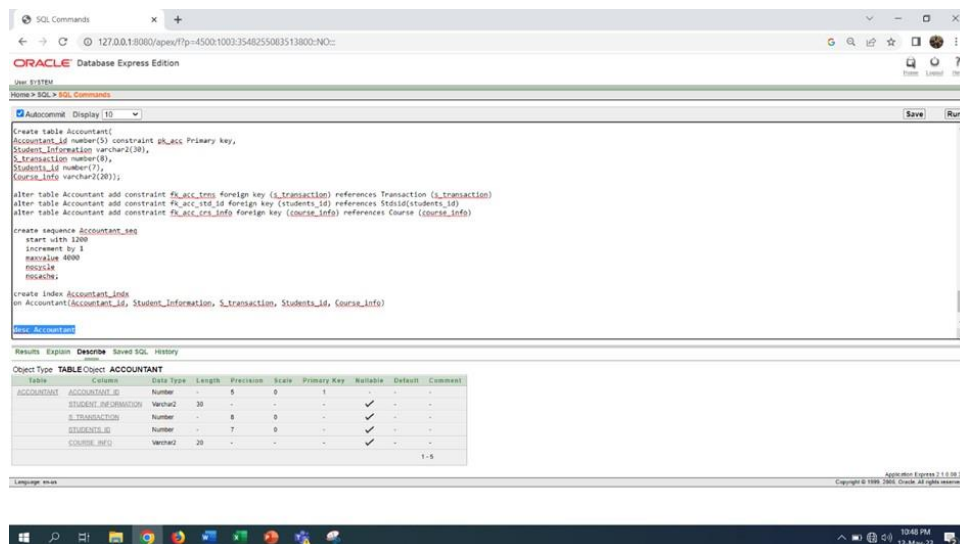


Table Stdsid

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 Registration Management System

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_idx

on StdSID(Students_id, Assigned_Course, Credit_per_course, Total_credit_fee, Course_info, Department_id, S_info, Course_id, Class_info)

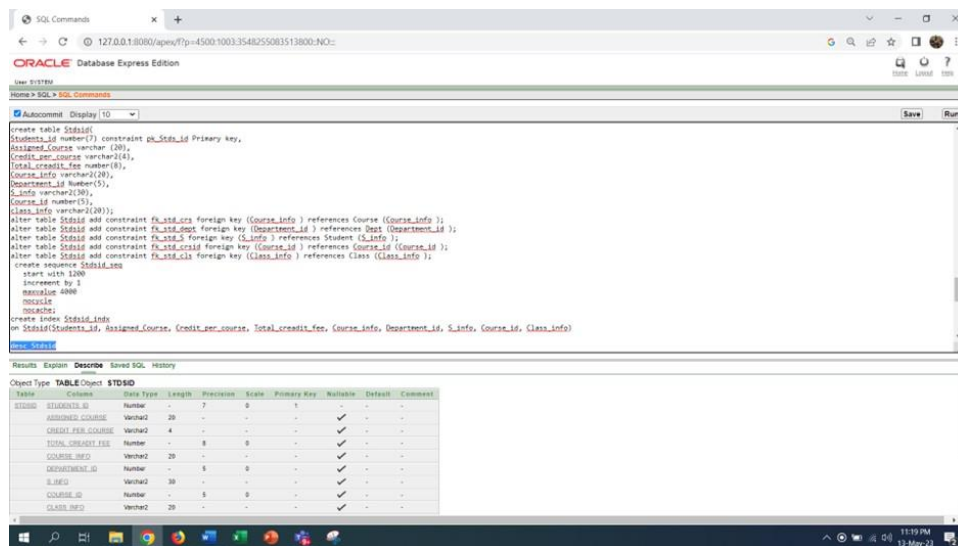


Table Dept

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

Course Registration Management System

```
(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
```

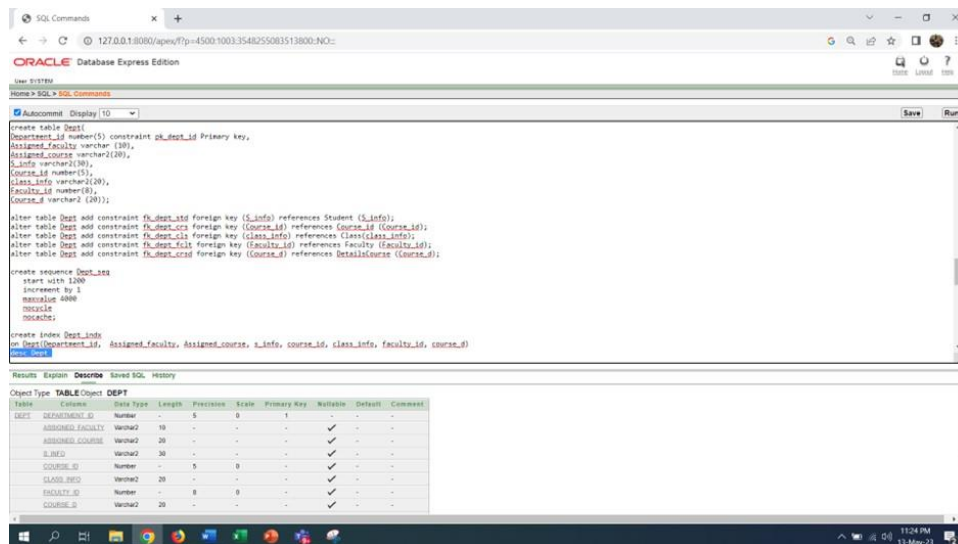


Table Course_id

```
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));
```

Course Registration Management System

```
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
```

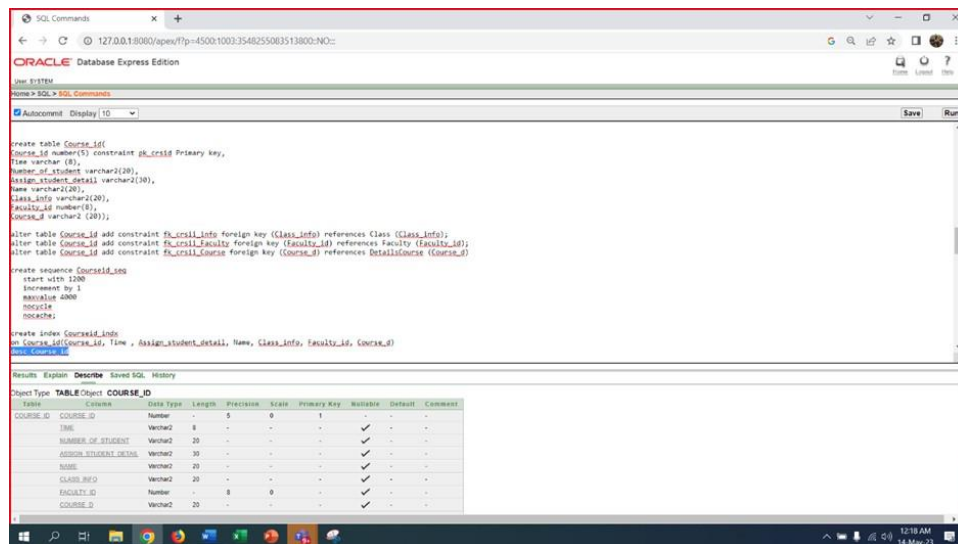
```
start with 1200 increment
```

```
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)
```

Course Registration Management System

```
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 screenshot shows a database management interface with a SQL editor and a results window. 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 window displays the following table:

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

5 rows returned in 0.00 seconds

```
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 in the script editor:

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

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

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

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

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

Select * from class;

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

The Results tab displays the following data for the 'class' table:

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 in the script editor:

```
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)
```

The Results tab displays the following data for the 'DetailsCourse' table:

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', '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;
```

The screenshot shows the Oracle SQL Developer interface. The top pane contains the following SQL script:

```
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;
```

The bottom pane shows the results of the last query, which is a table with 5 rows:

S_INFO	DETAILS	NAME
0001	CSE	Noboni
0002	CSE	Succho
0003	SE	Fahim
0004	CSE	Sakib
0005	SE	Ashesh

Below the table, it says "5 rows returned in 0.00 seconds". There is also a "CSV Export" button. The status bar at the bottom shows the connection string: "127.0.0.1:8080/apex/?p=4500:1000:4268762798051408".

```
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

The screenshot shows the Oracle SQL Developer interface. The top bar indicates 'Home > SQL > SQL Commands'. Below the toolbar, the SQL editor contains the following commands:

```
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;
```

The 'Results' pane below shows the output of the 'select * from Course;' query:

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. A 'CSV Export' button is visible at the bottom right of the results pane.

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;

The screenshot shows the Oracle SQL Developer interface. The top bar indicates 'Home > SQL > SQL Commands'. Below the toolbar, the SQL editor contains the following commands:

```
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;
```

The 'Results' pane below shows the output of the 'Select * from Transaction;' query:

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. A 'CSV Export' button is visible at the bottom right of the results pane.

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');
```

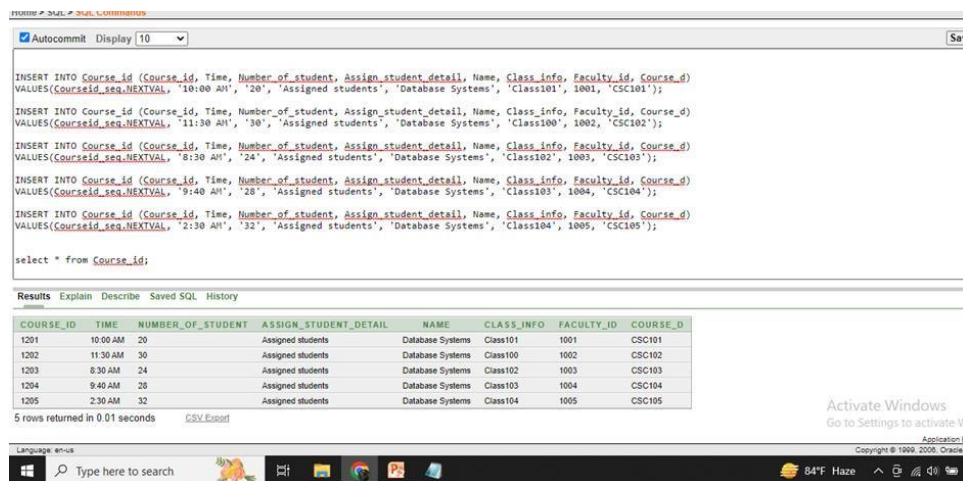
```
INSERT INTO Course_id (Course_id, Time, Number_of_student, Assign_student_detail, Name, Class_info, Faculty_id, Course_d)
```

```
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;



The screenshot shows a SQL query execution window with the following SQL code:

```
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;
```

The results are displayed in a table with the following columns: COURSE_ID, TIME, NUMBER_OF_STUDENT, ASSIGN_STUDENT_DETAIL, NAME, CLASS_INFO, FACULTY_ID, and COURSE_D. The table contains 5 rows of data.

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

5 rows returned in 0.01 seconds

```
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)
```

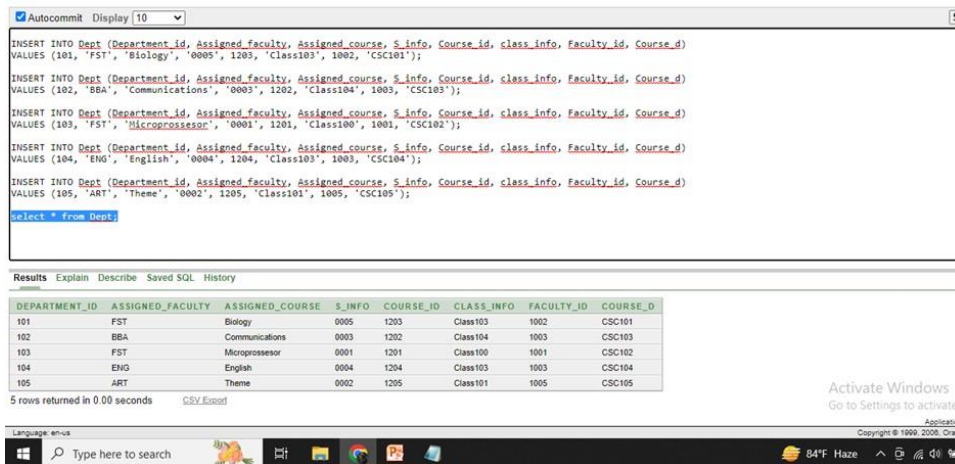
```
VALUES (103, 'FST', 'Microprossesor', '0001', 1201, 'Class100', 1001, 'CSC102');
```

```
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');
```



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

```
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'); select * from stdssid;
```

Course Registration Management System

Home > SQL > SQL Commands

☒ Autocommit Display 10 Save

```

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](#)

Activate Windows
Go to Settings to activate Windows

Application
Copyright © 1999, 2008, Oracle A

Language: en-us
Type here to search

84°F Haze

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 Save

```

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](#)

Activate Windows
Go to Settings to activate Windows

Application
Copyright © 1999, 2008, Oracle A

Language: en-us
Type here to search

84°F Haze

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](#)

3. Select Assigned_course,instr (Assigned_course, 'C')

Course Registration Management System

From stdsid

The screenshot shows the Oracle SQL Developer interface. At the top, the breadcrumb navigation is 'Home > SQL > SQL Commands'. Below this, there is a toolbar with 'Autocommit' checked and a 'Display' dropdown set to '10'. The SQL editor contains the following query:

```
Select Assigned_course,instr (Assigned_course, 'C')
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:

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

At the bottom of the results, it says '5 rows returned in 0.01 seconds' and there is a 'CSV Export' link.

Group function -3

1. select max(Student_due) from

Transaction

The screenshot shows the Oracle SQL Developer interface. At the top, the breadcrumb navigation is 'Home > SQL > SQL Commands'. Below this, there is a toolbar with 'Autocommit' checked and a 'Display' dropdown set to '10'. The SQL editor contains the following 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 of data:

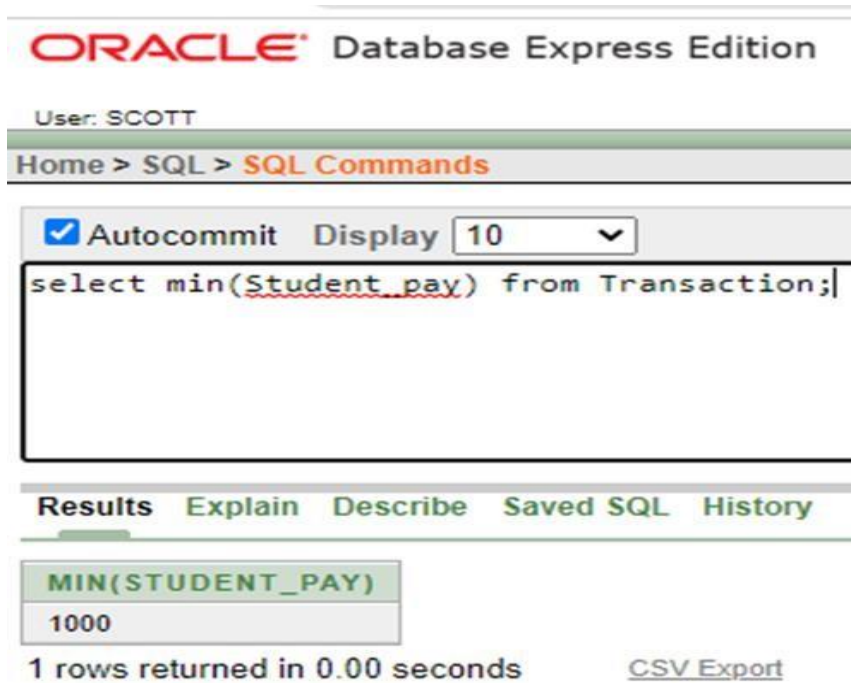
MAX(STUDENT_DUE)
1500

At the bottom of the results, it says '1 rows returned in 0.06 seconds' and there is a 'CSV Export' link.

Course Registration Management System

2. select min(Student_pay) from

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

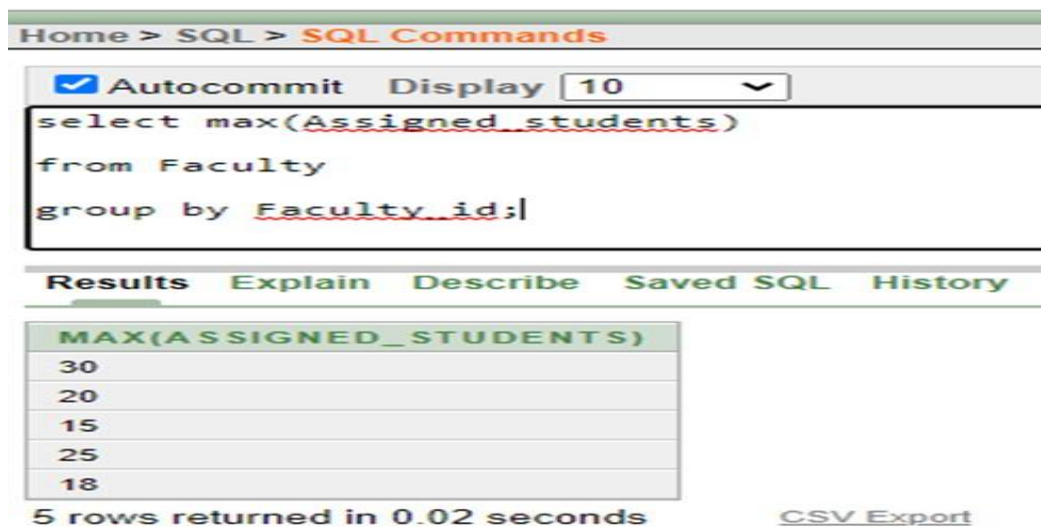
MIN(STUDENT_PAY)
1000

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](#)

Course Registration Management System

Subquery -3

1. Select name

From Student

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

Course Registration Management System

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

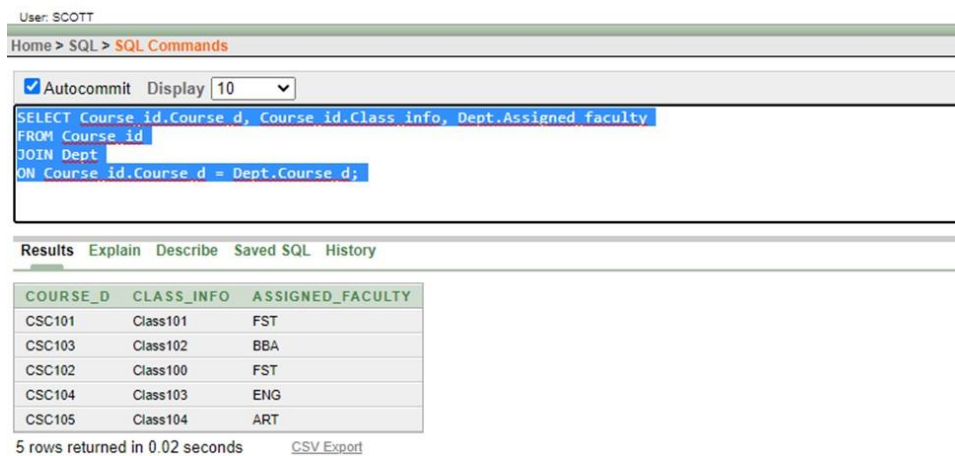
5 rows returned in 0.02 seconds [CSV Export](#)

2. Get course room number and faculty .

Course Registration Management System

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;
```



The screenshot shows a web-based SQL interface. At the top, it says 'User: SCOTT'. Below that is a breadcrumb 'Home > SQL > SQL Commands'. There's a toolbar with 'Autocommit' checked and 'Display' set to '10'. The SQL command window contains the following query:

```
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;
```

Below the command window is a tabbed interface with 'Results' selected. It displays a table with 5 rows and 3 columns: COURSE_D, CLASS_INFO, and ASSIGNED_FACULTY.

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

At the bottom, it says '5 rows returned in 0.02 seconds' and there is a 'CSV Export' link.

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. The SQL command window shows a query that joins department, student, and course information. The results window displays a table with 5 rows of data.

```
SELECT *
FROM dept d
JOIN ststdid s
ON s.course_id = s.course_info
JOIN course_id c
ON c.course_id = s.course_id
JOIN Student st
ON st.s_info = s.s_info;
```

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. The SQL command window shows the creation of a view named 'student_names' and a query to select all data from it. The results window displays a table with 5 rows of student names.

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

SELECT * FROM student_names;
```

NAME
Noboni
Succho
Fahim
Sakib
Ashesh

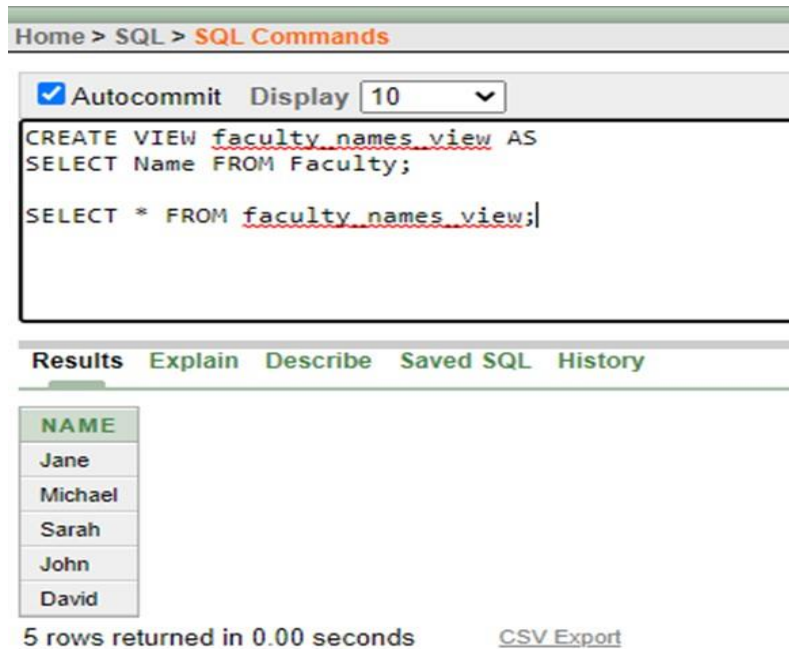
Course Registration Management System

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 SQL command window with the following content:

```
Home > SQL > SQL Commands
```

Autocommit is checked. Display is set to 10.

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

SELECT * FROM faculty_names_view;
```

Below the command window, there are tabs for Results, Explain, Describe, Saved SQL, and History. The Results tab is selected, showing a table with the following data:

NAME
Jane
Michael
Sarah
John
David

At the bottom, it says "5 rows returned in 0.00 seconds" and there is 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
IS max_credit
NUMBER;
```

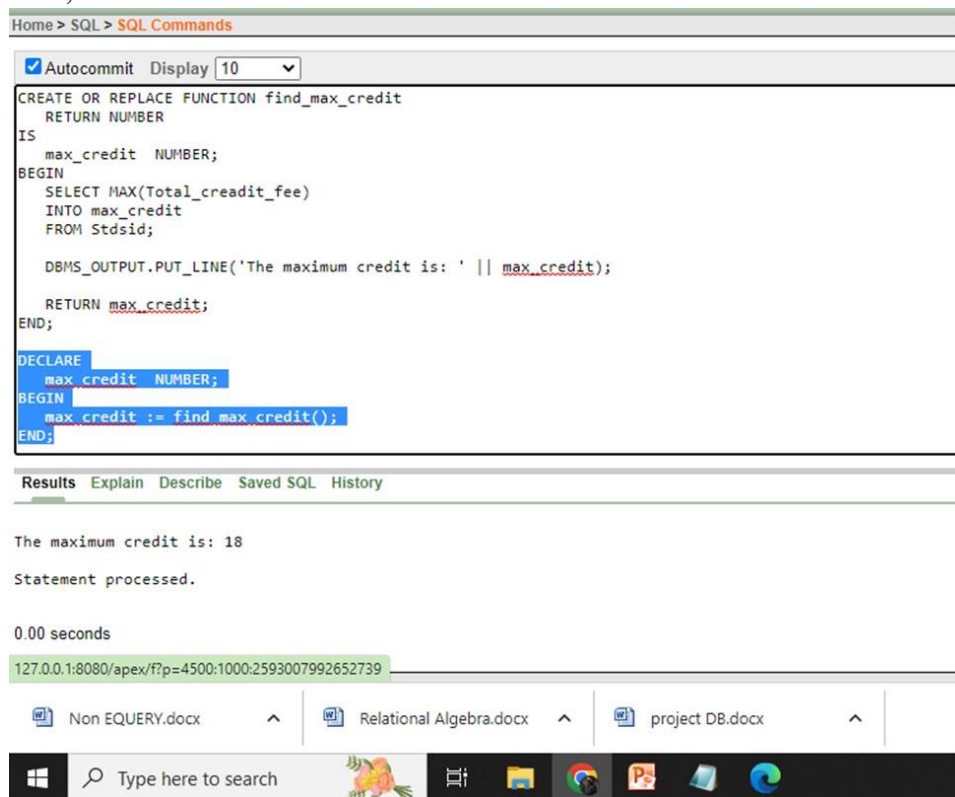
Course Registration Management System

```
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;
```



```
Home > SQL > SQL Commands
Autocommit Display 10
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;
```

Results Explain Describe Saved SQL History

The maximum credit is: 18

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

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

:

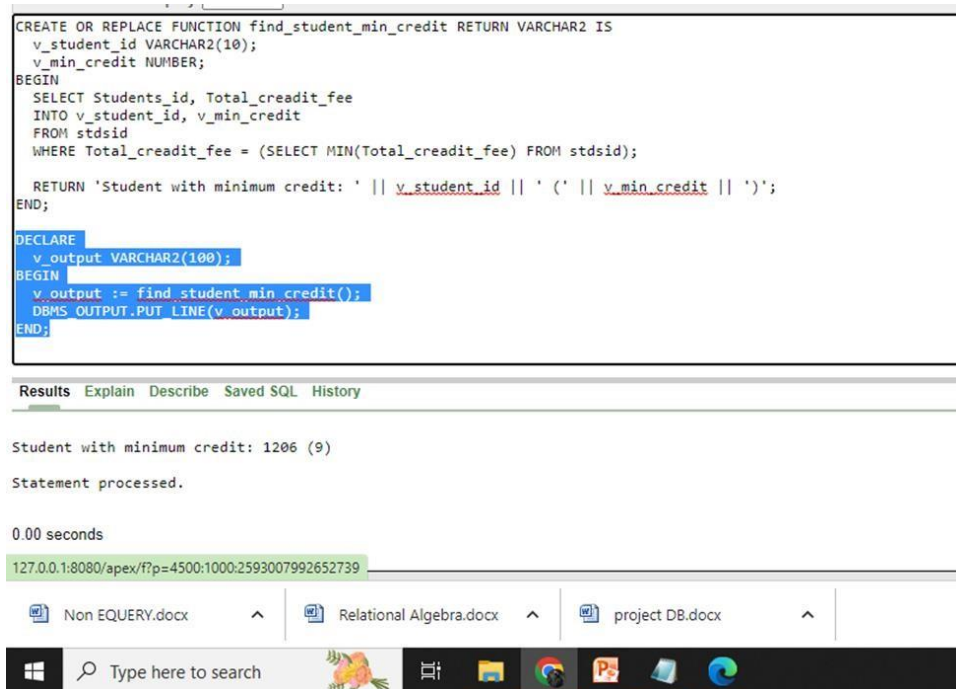
```
CREATE OR REPLACE FUNCTION find_student_min_credit RETURN VARCHAR2 IS
  v_student_id VARCHAR2(10); v_min_credit NUMBER;
BEGIN
```

Course Registration Management System

```
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;
```



The screenshot displays the Oracle SQL Developer environment. The top pane shows a PL/SQL script with the following code:

```
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;
```

Below the script, the 'Results' pane shows the output of the function call:

```
Student with minimum credit: 1206 (9)
```

The status bar at the bottom indicates 'Statement processed.' and '0.00 seconds'.

3. Write a function to find the maximum Student 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);
```

Course Registration Management System

```
RETURN max_due;  
END;
```

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

The screenshot shows the Oracle SQL Developer interface. At the top, the breadcrumb is 'Home > SQL > SQL Commands'. Below it, there's a toolbar with 'Autocommit' checked and 'Display' set to 10. The main text area contains the following SQL code:

```
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 code editor, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is active, showing a table with one row:

MAX(STUDENT_DUE)
1500

Below the table, it says '1 rows returned in 0.03 seconds' and there is a 'CSV Export' link. At the bottom of the window, there's a taskbar with several icons, including a search bar and application icons for Word, Excel, and others.

-3 procedure

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

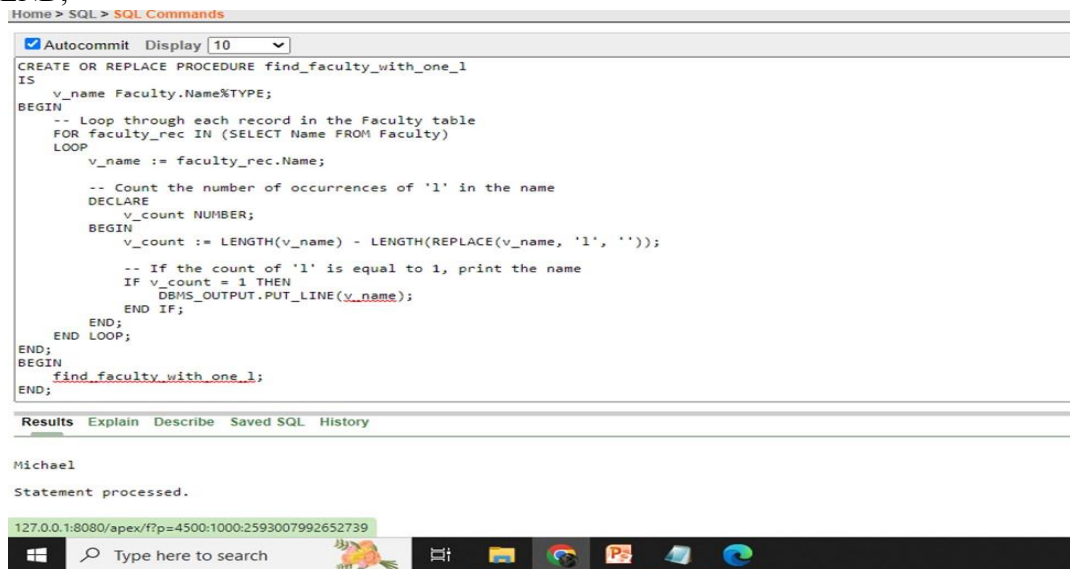
Ans :

```
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
```

Course Registration Management System

```
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;
```



```
Home > SQL > SQL Commands
Autocommit Display 10
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;
```

Results Explain Describe Saved SQL History

Michael

Statement processed.

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

2. Find the Class with maximum Roomnumber.

Ans ;

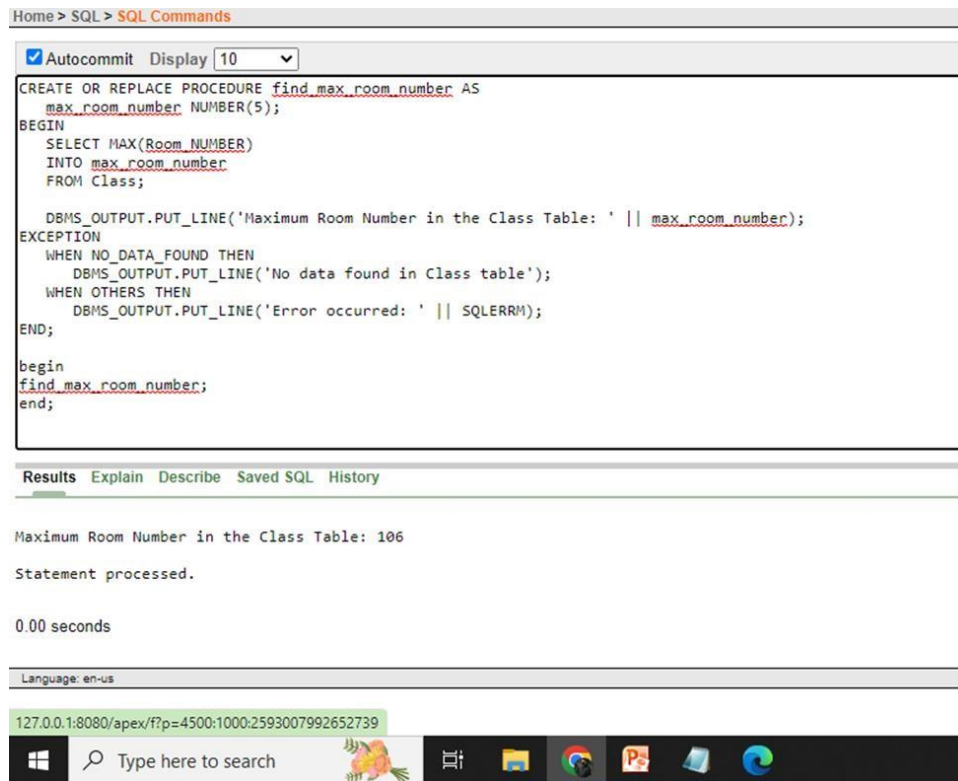
```
CREATE OR REPLACE PROCEDURE find_max_room_class
IS
    max_room_num NUMBER(4);
    max_room_class VARCHAR2(20);
BEGIN
    SELECT MAX(ROOM_NUMBER) INTO max_room_num FROM Class;

    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;
```


Course Registration Management System



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

```
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;
```

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

```
Maximum Room Number in the Class Table: 106
Statement processed.
0.00 seconds
```

At the bottom, there's a status bar showing 'Language: en-us' and a URL '127.0.0.1:8080/apex/f?p=4500:1000:2593007992652739'. The Windows taskbar is visible at the very bottom.

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;
```

Course Registration Management System

```
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

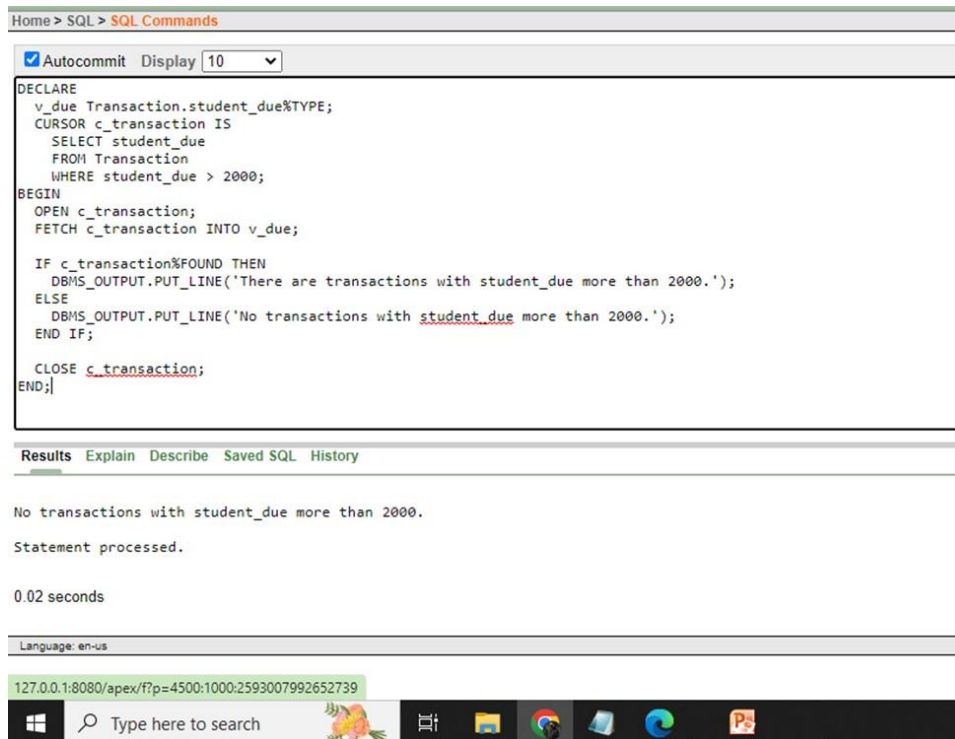
DBMS_OUTPUT.PUT_LINE('No transactions with student_due more than 2000.');

END IF;

CLOSE c_transaction;

END;

Course Registration Management System



```
Home > SQL > SQL Commands

Autocommit Display 10

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.');
```

Results Explain Describe Saved SQL History

No transactions with student_due more than 2000.

Statement processed.

0.02 seconds

Language: en-us

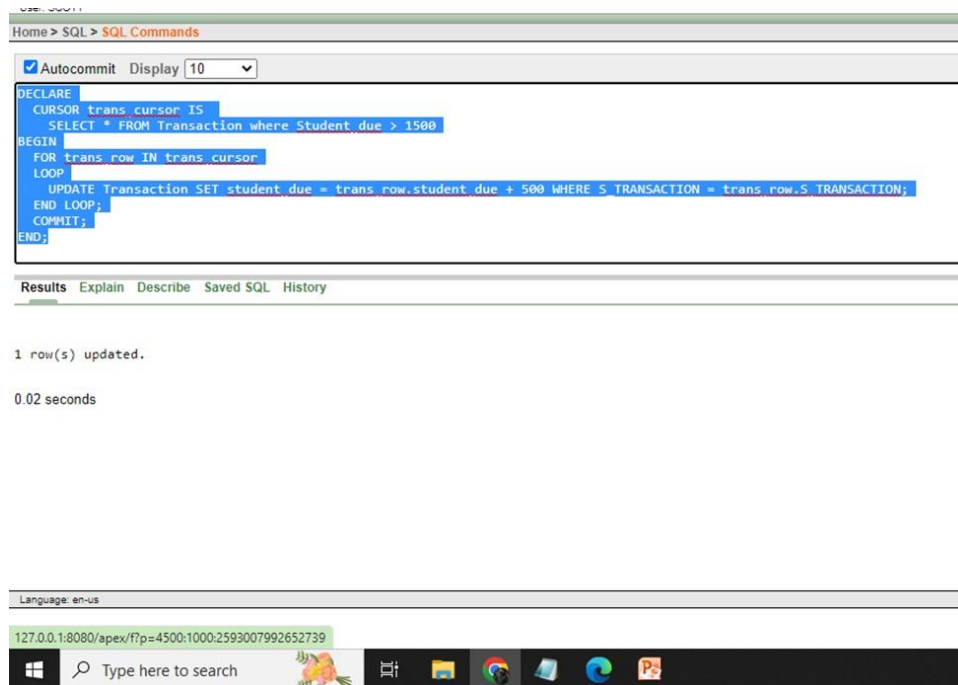
127.0.0.1:8080/apex/f?p=4500:1000:2593007992652739

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

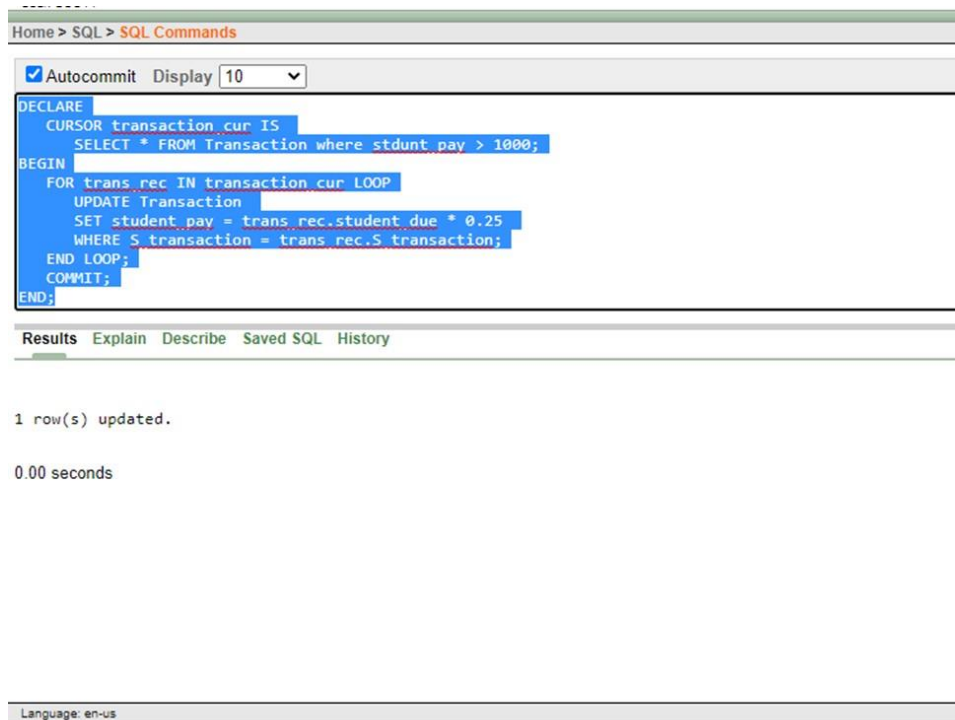


3. Write a cursor to give a 25 percent discount to students who paid more than 1000

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_credit_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 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,
```

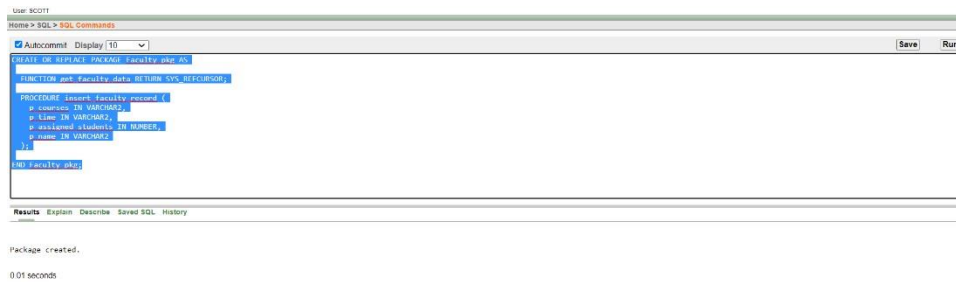
```
p_assigned_students IN NUMBER,
```

```
p_name IN VARCHAR2
```

```
);
```

```
END Faculty_pkg;
```

Course Registration Management System



The screenshot shows the SQL Developer interface with the 'SQL Commands' window. The code entered is:

```
CREATE OR REPLACE PACKAGE Faculty_pkg AS
FUNCTION get_Faculty_data RETURN SYS_REFCURSOR;
PROCEDURE insert_Faculty_member(
p_course IN VARCHAR2,
p_cred IN VARCHAR2,
p_student_id IN NUMBER,
p_fee IN VARCHAR2);
END Faculty_pkg;
```

The 'Results' pane below shows the message 'Package created.' and the execution time '0.01 seconds'.

2.

CREATE OR REPLACE PACKAGE Student_Pkg AS

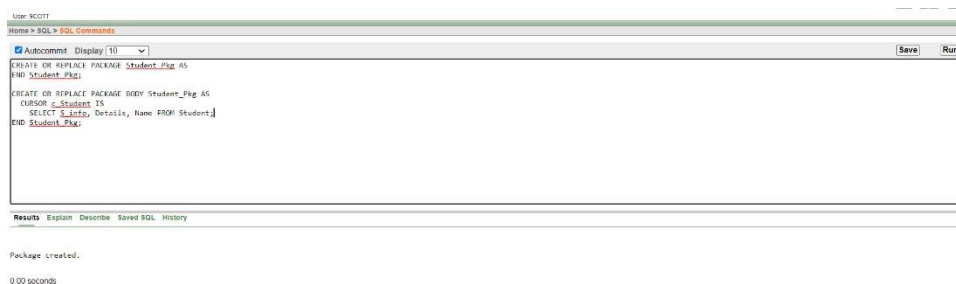
END Student_Pkg;

CREATE OR REPLACE PACKAGE BODY Student_Pkg AS

CURSOR c_Student IS

SELECT S_info, Details, Name FROM Student;

END Student_Pkg;



The screenshot shows the SQL Developer interface with the 'SQL Commands' window. The code entered is:

```
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;
```

The 'Results' pane below shows the message 'Package created.' and the execution time '0.00 seconds'.

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

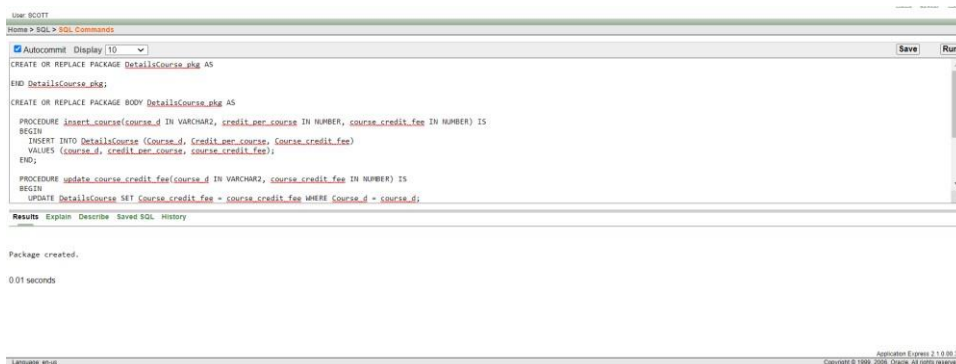
BEGIN

INSERT INTO DetailsCourse (Course_d, Credit_per_course, Course_credit_fee)

VALUES (course_d, credit_per_course, course_credit_fee);

Course Registration Management System

```
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
: Π S-Transaction [σ Student_Pay > 1000(Transaction)].

Course Registration Management System

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.