Project 2 report

Name: Sumeet Patle

B-Number: B00864468

Email: spatle1@binghamton.edu

==

Sequence for Log Entery for project(sequence file)

log_number:

Here creates “log_number” sequence which is a 6 digit number that starts from 100000 to 999999 and increments by 1

==

Trigger for procedure and function and log table (Trigger file)

TRIGGER logs_insert_students:

Trigger is created to insert an entry into the log table after a student is added to the students table.

TRIGGER logs_insert_enrollments:

Trigger is created to insert an entry into the log table after a student is a enrolled in a class.

TRIGGER logs_delete_student:

Trigger is created to insert an entry into the log table after a student is deleted from the students table.

TRIGGER logs_delete_enrollments:

This trigger is created to insert an entry into the log table after a student is dropped from a class

TRIGGER student_trigger_insert:

Trigger is created to insert the student from enrollments.

TRIGGER student_trigger_delete:

Trigger is created to delete the student from enrollments.

TRIGGER enrollments_trigger_insert:

Trigger is created to increment the class size by 1, when a student is enroll in a class.

TRIGGER enrollments_trigger_delete:

Trigger is created to decrement the class size by 1, when a student is dropped from a class

==

Package for project

All the package are in proj2_procedure.sql file

Which include procedure and function for project

==

Procedures and Function for project(Menu_Interface):

procedure show_students:

It is used to show the students table from the database.

procedure show_courses:

It is used to show the courses table from the database.

procedure show_classes:

It is used to show the classes table from the database.

procedure show_enrollments:

It is used to show the enrollments table from the database.

procedure show_prereq:

It is used to show the pre-requisites table from the database.

procedure show_logs:

It is used to show the logs table from the database.

Function show_students:

It is used to show the students table from the database.

function show_courses:

It is used to show the courses table from the database.

Function show_classes:

It is used to show the classes table from the database.

Function show_enrollments:

It is used to show the enrollments table from the database

procedure insert_student:

It takes sid, firstname, lastname, status, gpa and email as input and inserts into the student table.

procedure student_info:

It take sid as input and displays sid, lastname, status, classid, dept_code, course_no in the output.

function class_info:

It takes classid as input and displays classid, course title, sid, lastname and email as the output.

procedure course_info:

It takes deptcode and course_no as input and shows the dept_code, course_no and title of the direct/indirect pre-requisite courses.

procedure enroll_student:

It takes sid and classid as input and enrolls the student in that particular class, based on the conditions that are provided.

procedure delete_enrollment:

This procedure takes sid and classid as input and drops the student from that particular course, based on the conditions that are provided.

function delete_student:

It takes sid as input and deletes the the student from the student table as well as the enrollments table.

===

Execution for Project

Steps:

1. Enter into sql
2. Start project table to inter data or input in system (start proj2_tables)
3. Start sequence for log entery (start sequence)
4. Start trigger that trigger procedure and function (start trigger)
5. Start procedure and function (start proj2_procedure)
6. Start procedure and function body (start Menu_Interface)
7. Then exit from sql
8. Then compile java file(javac -cp /usr/lib/oracle/18.3/client64/lib/ojdbc8.jar proj2.java)
9. Then run java file (java -cp /usr/lib/oracle/18.3/client64/lib/ojdbc8.jar proj2.java)

In bracket command is there for run this project files

===

Source code:

----

proj2_tables.sql

drop table logs;

```
drop table prerequisites;

drop table enrollments;

drop table classes;

drop table courses;

drop table students;

drop sequence log_number;


create table students (sid char(5) primary key check (sid like 'B%'),

firstname varchar2(20) not null, lastname varchar2(20) not null, status
varchar2(10)

check (status in ('freshman', 'sophomore', 'junior', 'senior', 'graduate')),

gpa number(3,2) check (gpa between 0 and 4.0), email varchar2(20) unique);


create table courses (dept_code varchar2(4) not null, course_no number(3) not
null

check (course_no between 100 and 799), title varchar2(20) not null,

primary key (dept_code, course_no));


create table prerequisites (dept_code varchar2(4) not null,

course_no number(3) not null, pre_dept_code varchar2(4) not null,

pre_course_no number(3) not null,

primary key (dept_code, course_no, pre_dept_code, pre_course_no),

foreign key (dept_code, course_no) references courses on delete cascade,

foreign key (pre_dept_code, pre_course_no) references courses

on delete cascade);
```

```
create table classes (classid char(5) primary key check (classid like 'c%'),
dept_code varchar2(4) not null, course_no number(3) not null,
sect_no number(2), year number(4), semester varchar2(6)
check (semester in ('Spring', 'Fall', 'Summer')), limit number(3),
class_size number(3), foreign key (dept_code, course_no) references courses
on delete cascade, unique(dept_code, course_no, sect_no, year, semester),
check (class_size <= limit));

create table enrollments (sid char(5) references students, classid char(5)
references classes,
lgrade char check (lgrade in ('A', 'B', 'C', 'D', 'F', 'I', null)), primary key (sid, classid));

create table logs (logid number(20) primary key, who varchar2(40) not null, time
date not null,
table_name varchar2(20) not null, operation varchar2(20) not null, key_value
varchar2(14));


insert into students values ('B001', 'Anne', 'Broder', 'junior', 3.17,
'broder@bu.edu');
insert into students values ('B002', 'Terry', 'Buttler', 'senior', 3.0,
'buttler@bu.edu');
insert into students values ('B003', 'Tracy', 'Wang', 'senior', 4.0, 'wang@bu.edu');
insert into students values ('B004', 'Barbara', 'Callan', 'junior', 2.5,
'callan@bu.edu');
```

```
insert into students values ('B005', 'Jack', 'Smith', 'graduate', 3.0,
'smith@bu.edu');

insert into students values ('B006', 'Terry', 'Zillman', 'graduate', 4.0,
'zillman@bu.edu');

insert into students values ('B007', 'Becky', 'Lee', 'senior', 4.0, 'lee@bu.edu');

insert into students values ('B008', 'Tom', 'Baker', 'freshman', null,
'baker@bu.edu');


insert into courses values ('CS', 432, 'database systems');

insert into courses values ('Math', 314, 'discrete math');

insert into courses values ('CS', 240, 'data structure');

insert into courses values ('Math', 221, 'calculus I');

insert into courses values ('CS', 532, 'database systems');

insert into courses values ('CS', 552, 'operating systems');

insert into courses values ('BIOL', 425, 'molecular biology');


insert into prerequisites values ('CS', 432, 'CS', 240);

insert into prerequisites values ('CS', 532, 'CS', 432);

insert into prerequisites values ('Math', 314, 'Math', 221);

insert into prerequisites values ('CS', 432, 'Math', 314);

insert into prerequisites values ('CS', 552, 'CS', 240);


insert into classes values  ('c0001', 'CS', 432, 1, 2011, 'Spring', 3, 1);

insert into classes values  ('c0002', 'Math', 314, 1, 2010, 'Fall', 2, 2);

insert into classes values  ('c0003', 'Math', 314, 2, 2010, 'Fall', 3, 2);
insert into classes values  ('c0004', 'CS', 432, 1, 2010, 'Spring', 2, 1);
insert into classes values  ('c0005', 'CS', 240, 1, 2011, 'Spring', 5, 5);
insert into classes values  ('c0006', 'CS', 532, 1, 2011, 'Spring', 2, 1);
insert into classes values  ('c0007', 'Math', 221, 1, 2011, 'Spring', 6, 5);

insert into enrollments values  ('B001', 'c0001', 'A');
insert into enrollments values  ('B002', 'c0002', 'B');
insert into enrollments values  ('B006', 'c0007', 'A');
insert into enrollments values  ('B004', 'c0005', 'C');
insert into enrollments values  ('B005', 'c0005', 'B');
insert into enrollments values  ('B005', 'c0007', 'B');
insert into enrollments values  ('B006', 'c0003', 'A');
insert into enrollments values  ('B001', 'c0002', 'C');
insert into enrollments values  ('B003', 'c0005', null);
insert into enrollments values  ('B002', 'c0007', 'A');
insert into enrollments values  ('B001', 'c0007', 'B');
insert into enrollments values  ('B001', 'c0006', 'B');
insert into enrollments values  ('B001', 'c0005', 'A');
insert into enrollments values  ('B005', 'c0003', 'B');
insert into enrollments values  ('B005', 'c0004', 'D');
insert into enrollments values  ('B007', 'c0005', 'B');
insert into enrollments values  ('B008', 'c0007', 'A');
```

----

sequence.sql

```
create sequence log_number minvalue 100000 maxvalue 999999 increment by 1
start with 100000;

/*6 digit sequence for log entry */
```

----

trigger.sql

```
 /*trigger is created for each procedure and function */

create or replace trigger student_trigger_delete

after delete on students

for each row

begin

delete from enrollments where sid= :old.sid;

end;

/


create or replace trigger enrollments_trigger_delete

after delete on enrollments

for each row

begin

update classes set class_size=class_size-1 where classid=:old.classid;

end;

/


create or replace trigger enrollments_trigger_insert

after insert on enrollments
```

```
for each row

begin

update classes set class_size=class_size+1 where classid= :new.classid;

end;

/

/*trigger for log each procedure and function*/


CREATE OR REPLACE TRIGGER logs_insert_student

AFTER INSERT ON students

FOR EACH ROW

BEGIN

INSERT INTO logs VALUES(log_number.NEXTVAL, user, SYSDATE, 'Students',
'insert', :NEW.sid);

END;

/


CREATE OR REPLACE TRIGGER logs_delete_student

AFTER DELETE ON students

FOR EACH ROW

BEGIN

INSERT INTO logs VALUES(log_number.NEXTVAL, user, SYSDATE, 'Students',
'delete', :OLD.sid);

END;

/
```

```
CREATE OR REPLACE TRIGGER logs_insert_enrollments

AFTER INSERT ON enrollments

FOR EACH ROW

BEGIN

INSERT INTO logs VALUES(log_number.NEXTVAL, user, SYSDATE, 'Enrollments',
'insert', :NEW.sid || ' ' || :NEW.classid);

END;

/


CREATE OR REPLACE TRIGGER logs_delete_enrollments

AFTER DELETE ON enrollments

FOR EACH ROW

BEGIN

INSERT INTO logs VALUES(log_number.NEXTVAL, user, SYSDATE, 'Enrollments',
'delete', :OLD.sid || ' ' || :OLD.classid);

END;

/


show errors
```

----

proj2_prodecure.sql

```
create or replace package proj2_procedure as

procedure show_students(students_curs out sys_refcursor);

procedure show_courses(courses_curs out sys_refcursor);
procedure show_prereq(prereq_curs out sys_refcursor);
procedure show_classes(classes_curs out sys_refcursor);
procedure show_enrollments(enrollments_curs out sys_refcursor);
procedure show_logs(logs_curs out sys_refcursor);
/*function show_courses(courses_curs out sys_refcursor);
function show_students(students_curs out sys_refcursor);
function show_classes(classes_curs out sys_refcursor);
function show_enrollments(enrollments_curs out sys_refcursor);*/
procedure insert_student(studentid in students.sid%type,Fname in
students.firstname%type, Lname in students.lastname%type, Stat in
students.status%type,gp in students.gpa%type,Mail in
students.email%type);
end;
/
create or replace package body proj2_procedure as
procedure show_students(students_curs out sys_refcursor) as
begin
open students_curs for
select * from students;
end show_students;



procedure show_courses(courses_curs out sys_refcursor) as
begin
```

```
open courses_curs for
select * from courses;
end show_courses;


procedure show_prereq(prereq_curs out sys_refcursor) as
begin
open prereq_curs for
select * from prerequisites;
end show_prereq;


procedure show_classes(classes_curs out sys_refcursor) as
begin
open classes_curs for
select * from classes;
end show_classes;


procedure show_enrollments(enrollments_curs out sys_refcursor) as
begin
open enrollments_curs for
select * from enrollments;
```

```
end show_enrollments;

procedure show_logs(logs_curs out sys_refcursor) as
begin
open logs_curs for
select * from logs;
end show_logs;

procedure insert_student(studentid in students.sid%type,Fname in
students.firstname%type, Lname in students.lastname%type, Stat in
students.status%type,gp in students.gpa%type,Mail in
students.email%type)
is
begin
   insert into
students("SID","FIRSTNAME","LASTNAME","STATUS","GPA","EMAIL")
values(studentid,Fname,Lname,Stat,gp,Mail);
COMMIT;
end insert_student;

/*function show_courses(courses_curs out sys_refcursor) return courses_curs as
begin
open courses_curs for
select * from courses;
```

```
return courses_curs;
end show_courses;

function show_classes(classes_curs out sys_refcursor) as
begin
open classes_curs for
select * from classes;
end show_classes;

function show_enrollments(enrollments_curs out sys_refcursor) as
begin
open enrollments_curs for
select * from enrollments;
end show_enrollments;

function show_students(students_curs out sys_refcursor) as
begin
open students_curs for
select * from students;
end show_students;*/

end proj2_procedure;
/
show errors
```

Menu_Interface.sql

```
set serveroutput on

/*Procedure to get student information*/

create or replace procedure student_info(studentid in students.sid%type,
showmessage OUT VARCHAR2,stud_info OUT sys_refcursor)

is

course_reg int;

student_exist1 int;

begin

select count(*) into student_exist1 from students s where s.sid=studentid;

if student_exist1 = 0

then

dbms_output.put_line('invalid sid');

showmessage := 'invalid sid';

else

select count(*) into course_reg from students s,enrollments e, classes c where
s.sid =studentid and e.sid = s.sid and e.classid = c.classid;

if course_reg = 0

then

dbms_output.put_line('The sid has not taken any course');

showmessage := 'The sid has not taken any course';

else
```

```
open stud_info for

select s.sid, s.lastname, s.status, c.classid, concat(c.dept_code,c.course_no)
course from students s,enrollments

e, classes c where s.sid=studentid and e.sid = s.sid and e.classid = c.classid;

end if;

end if;

end;

/

show errors


/****************************************************************
****************************************************************
***************

*/


/*Procedure to get course information*/

create or replace procedure course_info(deptcode in
prerequisites.dept_code%type,cnum in prerequisites.course_no%type,count1 in
number,result out varchar2)

is

course_exist1 number;

count2 number;

pre_code prerequisites.dept_code%type;

pre_num prerequisites.course_no%type;

pre_title courses.title%type;

CURSOR course1_info is
```

```
select pre_dept_code, pre_course_no, pre_title  FROM

prerequisites p, courses c WHERE p.dept_code=deptcode and p.course_no =
cnum and p.dept_code=c.dept_code and p.course_no=c.course_no;

detail course1_info%rowtype;

begin

select count(*) into course_exist1 from prerequisites p where
p.dept_code=deptcode and p.course_no=cnum;

open course1_info;

fetch course1_info into detail;

while(course1_info%found) loop

count2:=count1+1;

course_info(detail.pre_dept_code, detail.pre_course_no,count2,result);

if(result is NULL) then

result := detail.pre_dept_code || detail.pre_course_no || detail.pre_title;

else

result := result || ',' || detail.pre_dept_code || detail.pre_course_no ||
detail.pre_title;

end if;


fetch course1_info into detail;

end loop;

if(course_exist1=0 and count1=0)then

result := ' Does not exist';

end if;

end;
```

```
/

show errors


/*****************************************************************
*****************************************************************
***************

*/


/*function to get class information*/

create or replace procedure class_info(id in classes.classid%type,showmessage
OUT VARCHAR2,class1_info OUT sys_refcursor)

is

class_exist2 int;

student_exist2 int;

begin

select count(*) into class_exist2 from classes c where c.classid=id;

if class_exist2 = 0

then

dbms_output.put_line('invalid cid');

showmessage := 'invalid cid';

else

select count(*) into student_exist2 from enrollments e where e.classid=id;

if student_exist2 = 0

then

dbms_output.put_line('No student is enrolled in the class');
```

```
showmessage := 'No student is enrolled in the class';

else

open class1_info for

select c1.classid,c2.title,s.sid,s.lastname,s.email from classes c1,courses c2,
enrollments e, students s where c1.classid=id and c1.dept_code=c2.dept_code
and c1.course_no= c2.course_no and c1.classid=e.classid and e.sid=s.sid;

end if;

end if;

end;

/

show errors


/****************************************************************
*****************************************************************
***************

*/


/*Procedure to enroll student*/

create or replace procedure enroll_student(std_id in students.sid%type,cl_id in
classes.classid%type,showmessage OUT VARCHAR2)

is

student_exist3 int;

class_exist3 int;

count_wrong int;

check_student int;

count_enroll int;
```

```
dep_code classes.dept_code%type;

cnum classes.course_no%type;

pre_classes varchar2(100);

pre_count int;

pre_count2 int;

begin

select count(*) into student_exist3 from students s where std_id=s.sid;

select count(*) into class_exist3 from classes c where cl_id = c.classid;

select count(*) into count_wrong from classes c where c.class_size+1>limit and
cl_id = c.classid;

select count(*) into check_student from enrollments e where e.sid=std_id and
e.classid = cl_id;

select count(*) into count_enroll from enrollments e where e.sid=std_id;

select dept_code,course_no into dep_code,cnum from classes where
classid=cl_id;

course_info(dep_code, cnum, 0,pre_classes);


select count(*) into pre_count from enrollments e,classes c where sid = std_id
and e.classid = c.classid

and INSTR(pre_classes, c.dept_code || c.course_no) != 0 and e.lgrade not in
('A','A-','B+','B','B-','C+');


select count(*) into pre_count2 from enrollments e,classes c where sid = std_id
and e.classid = c.classid

and INSTR(pre_classes, c.dept_code || c.course_no) = 0;

if
```

```
student_exist3=0

then

showmessage := 'invalid sid.';

dbms_output.put_line('invalid sid.');


elsif class_exist3=0

then

showmessage := 'invalid classid';

dbms_output.put_line('invalid classid');


elsif count_wrong > 0

then

showmessage := 'class full';

dbms_output.put_line('class full');


elsif check_student>0

then

showmessage := 'already in the class';

dbms_output.put_line('already in this class');


elsif count_enroll>5

then

showmessage := 'overloaded!';

dbms_output.put_line('overloaded!');
```

```
elsif pre_count2>0
then
showmessage := 'Prerequisites courses have not been completed';
dbms_output.put_line('Prerequisites courses have not been completed');
elsif pre_count>0
then
showmessage := 'Prerequisites courses have not been completed';
dbms_output.put_line('Prerequisites courses have not been completed');


else
if count_enroll=4
then
dbms_output.put_line('Successfully enrolled, course count is 5');
insert into enrollments values (std_id,cl_id,null);
showmessage := 'Successfully enrolled, course count is 4';
else
showmessage := 'Successfully enrolled';
insert into enrollments values (std_id,cl_id,null);
end if;
end if;
end;
/
show errors
```

```
/**************************************************************
***************************************************************
**************
*/

/*Procedure to delete student enrollment*/
create or replace procedure delete_enrollment(std_id in students.sid%type,cl_id
in classes.classid%type, showmessage OUT VARCHAR2)
is
student_exist4 int;
class_exist4 int;
enroll_exist int;
course_count int;
enroll_exist1 int;
dep_code classes.dept_code%type;
cnum classes.course_no%type;
pre_count int;
begin
select count(*) into student_exist4 from students s where std_id=s.sid;
select count(*) into class_exist4 from classes c where cl_id = c.classid;
select count(*) into enroll_exist from enrollments e where e.sid=std_id and
e.classid=cl_id;
select count(*) into enroll_exist1 from enrollments e where e.sid=std_id;
select count(*) into course_count from enrollments e where e.classid=cl_id;
if
```

```
student_exist4=0

then

showmessage := 'invalid sid';

dbms_output.put_line('invalid sid');

commit;

elsif class_exist4=0

then

showmessage := 'invalid classid';

dbms_output.put_line('invalid classid');

commit;

elsif enroll_exist=0

then

showmessage := 'student not enrolled';

dbms_output.put_line('student not enrolled');

commit;

else


select dept_code,course_no into dep_code, cnum from classes where
classid=cl_id;


select count(*) into pre_count from classes cl,prerequisites p where cl.classid in

(select classid from enrollments e where e.classid != cl_id and e.sid=std_id)

and cl.dept_code=p.dept_code and cl.course_no = p.course_no and

p.pre_dept_code=dep_code and

p.pre_course_no=cnum;
```

```
if(pre_count=0) then
if(enroll_exist1=1) then
delete from enrollments e where e.sid = std_id and e.classid=cl_id;
showmessage := 'drop request rejected; must be enrolled in at least one class';
dbms_output.put_line('drop request rejected; must be enrolled in at least one
class');
elsif(course_count=1) then
delete from enrollments e where e.sid = std_id and e.classid=cl_id;
showmessage := 'no student in this class';
dbms_output.put_line('no student in this class');
else
delete from enrollments e where e.sid = std_id and e.classid=cl_id;
showmessage := 'The student dropped successfully';
dbms_output.put_line('The student is successfully dropped');
end if;
else
showmessage := 'drop request rejected due to prerequisite requirements';
dbms_output.put_line('drop request rejected due to prerequisite requirements');


end if;
end if;


end;
/
show errors
```

```
/*****************************************************************
*****************************************************************
***************
*/


/*function to delete student*/
create or replace procedure delete_student(std_id in students.sid%type)
is
student_exist5 int;
student_enrolled int;
begin
select count(*) into student_exist5 from students where sid=std_id;
if student_exist5=0
then
dbms_output.put_line('sid not found');
else
select count(*) into student_enrolled from students s, enrollments e
where s.sid = std_id and s.sid = e.sid;
if student_enrolled = 0
then
dbms_output.put_line('student is not enrolled in any courses');
delete from students where sid=std_id;
commit;
dbms_output.put_line('deleted sucessfully');
```

```
else
delete from enrollments where sid = std_id;
delete from students s where s.sid = std_id;
commit;
dbms_output.put_line('student deleted sucessfully from enrollments table ');
end if;
end if;
end;
/
show errors
```

Proj2.java

```
jk import java.sql.*;
import oracle.jdbc.*;
import java.math.*;
import java.io.*;
import java.awt.*;
import oracle.jdbc.pool.OracleDataSource;



public class proj2
{

   public static void main(String[] args)  throws SQLException
   {
   try
   {
   //Connection to Oracle server
   OracleDataSource ds = new oracle.jdbc.pool.OracleDataSource();
   ds.setURL("jdbc:oracle:thin:@castor.cc.binghamton.edu:1521:ACAD111");
   Connection conn = ds.getConnection("spatle1","sumeet01111995");
   int User_Selection = -1;
   BufferedReader readKeyBoard = new BufferedReader(new
InputStreamReader(System.in));
   String choice;

   //Available options
   while(User_Selection != 0)
   {
   System.out.println("\nPlease select one of the following options:");
   System.out.println("1- Display all the tables in the database");
   System.out.println("2- Add a student");
   System.out.println("3- Get the information for student");
   System.out.println("4- Show the prerequisites courses for a given course");
   System.out.println("5- Show the information and the list of students for a given
class");
   System.out.println("6- Enroll a student into a class");
   System.out.println("7- Drop a student from a class");
   System.out.println("8- Delete a student");
   System.out.println("0- Exit");

   choice = readKeyBoard.readLine();
   User_Selection = Integer.parseInt(choice);
   //Function call based on choice

   if(User_Selection == 1)
   {

   // print all the tables
   printTables(conn);
   }
   else if(User_Selection == 2)
   {
   //add student
   addStudent(conn);
   }
   else if(User_Selection == 3)
   {
   //getStudentInfo
   getStudentInfo(conn);
   }
   else if(User_Selection == 4)
   {
```

```
getCoursePrereq(conn);
}
else if(User_Selection == 5)
{
//showcourseinfo
classInfoStudents(conn);
}
else if(User_Selection == 6)
{
enrollStudent(conn);
}
//drop student
else if(User_Selection == 7)
{
dropStudent(conn);
}
// delete student from student table
else if(User_Selection == 8)
{
deleteStudent(conn);
}


}
conn.close();
```

```
   }
   catch(SQLException ex){
   System.out.println("SQL Exception");
   }
   catch(Exception e){
   System.out.println("Exception");
   }
   }
public static void printTables(Connection conn) throws SQLException
{
/*The proj2_procedure package is used to get the information of the tables in the
database*/
try{
//Using the show_students procedure in the proj2_procedure package for
students table
CallableStatement cs = conn.prepareCall("begin
proj2_procedure.show_students(?); end;");
cs.registerOutParameter(1,OracleTypes.CURSOR);
//Printing Student table
// here execute and retrieve the result set
cs.execute();
ResultSet rs = (ResultSet)cs.getObject(1);
System.out.println("STUDENTS TABLE");
System.out.println("------------------------------------------------------------------------------
-----------------------------------------------------------");
```

```
System.out.println("SID\t\t\tFIRSTNAME\t\tLASTNAME\t\tSTATUS\t\t\tGPA\t\tE
MAIL");

System.out.println("-------------------------------------------------------------------------------
-------------------------------------------------------");

while (rs.next())

{

System.out.println(rs.getString(1)+"\t\t\t"+rs.getString(2)+"\t\t\t"+rs.getString(3)
+"\t\t\t"+rs.getString(4)+"\t\t\t"+rs.getString(5)+"\t\t"+rs.getString(6));

}


//Using the show_courses procedure in the proj2_procedure package for courses
table

cs = conn.prepareCall("begin proj2_procedure.show_courses(?); end;");

cs.registerOutParameter(1,OracleTypes.CURSOR);

//Printing Course table

// here execute and retrieve the result set

cs.execute();

rs = (ResultSet)cs.getObject(1);

System.out.println("\nCOURSES TABLE");

System.out.println("-------------------------------------------------------------------------------
---------------------------");

System.out.println("DEPT_CODE\t\tCOURSE_NO\t\tTITLE");

System.out.println("-------------------------------------------------------------------------------
---------------------------");

while (rs.next())

{
```

```
System.out.println(rs.getString(1)+"\t\t\t"+rs.getString(2)+"\t\t\t"+rs.getString(3))
;

}


//Using the show_prereq procedure in the proj2_procedure package for
prerequisites table

cs = conn.prepareCall("begin proj2_procedure.show_prereq(?); end;");

cs.registerOutParameter(1,OracleTypes.CURSOR);

//Printing Prerequisites table

// execute and retrieve the result set

cs.execute();

rs = (ResultSet)cs.getObject(1);

System.out.println("\nPREREQUISITES TABLE");

System.out.println("----------------------------------------------------------------------------------
---------------------------");

System.out.println("DEPT_CODE\t\tCOURSE_NO\t\tPRE_DEPT_CODE\t\tPRE_CO
URSE_NO");

System.out.println("----------------------------------------------------------------------------------
---------------------------");

while (rs.next())

{

System.out.println(rs.getString(1)+"\t\t\t"+rs.getString(2)+"\t\t\t"+rs.getString(3)
+"\t\t\t"+rs.getString(4));

}
```

```
//Using the show_classes procedure in the proj2_procedure package for classes
table

cs = conn.prepareCall("begin proj2_procedure.show_classes(?); end;");

cs.registerOutParameter(1,OracleTypes.CURSOR);

//Printing Classes table

// execute and retrieve the result set

cs.execute();

rs = (ResultSet)cs.getObject(1);

System.out.println("\nCLASSES TABLE");

System.out.println("-----------------------------------------------------------------------------------
----------------------------------------------");

System.out.println("CLASSID\tDEPT_CODE\tCOURSE_NO\tSECT_NO\t\tYEAR\t\tS
EMESTER\tLIMIT\t\tCLASS_SIZE");

System.out.println("-----------------------------------------------------------------------------------
----------------------------------------------");

while (rs.next())

{

System.out.println(rs.getString(1)+"\t"+rs.getString(2)+"\t\t"+rs.getString(3)+"\t\t
"+rs.getString(4)+"\t\t"+rs.getString(5)+"\t\t"+rs.getString(6)+"\t\t"+rs.getString(7
)+"\t\t"+rs.getString(8));

}

//Using the show_enrollments procedure in the proj2_procedure package for
enrollments table

cs = conn.prepareCall("begin proj2_procedure.show_enrollments(?); end;");

cs.registerOutParameter(1,OracleTypes.CURSOR);

// execute and retrieve the result set
```

```
cs.execute();

rs = (ResultSet)cs.getObject(1);

System.out.println("\nENROLLMENTS TABLE");

System.out.println("--------------------------------------------------------------------------------
--------------------------");

System.out.println("SID\t\t\tCLASSID\t\t\tLGRADE");

System.out.println("--------------------------------------------------------------------------------
--------------------------");

while (rs.next())

{

System.out.println(rs.getString
(1)+"\t\t\t"+rs.getString(2)+"\t\t\t"+rs.getString(3));

}

//Using the show_logs procedure in the proj2_procedure package for logs table

cs = conn.prepareCall("begin proj2_procedure.show_logs(?); end;");

cs.registerOutParameter(1,OracleTypes.CURSOR);

cs.execute();

rs = (ResultSet)cs.getObject(1);

System.out.println("\nLOGS TABLE");

System.out.println("--------------------------------------------------------------------------------
--------------------------");

System.out.printf("logid\twho\t\ttime\t\t\ttable_name\toperation\tkey_value\n
");

System.out.println("--------------------------------------------------------------------------------
--------------------------");

while(rs.next())
```

```
{
System.out.println(rs.getString(1)+"\t"+rs.getString(2)+"\t"+rs.getString(3)+"\t"+r
s.getString(4)+"\t"+rs.getString(5)+"\t\t"+rs.getString(6));
}
rs.close();
}
catch(SQLException ex){
System.out.println("SQL Exception in Print table function:");
System.out.println(ex);
}

return;
}

//-----------------------------------------------------------------------------------------------------
------------

public static void addStudent(Connection conn) throws SQLException
{
/*This function is used to add a student into student table using insert_student in
the proj2_procedure*/
try{
// Fetching the inputs
BufferedReader readkey = new BufferedReader(new
InputStreamReader(System.in));
System.out.println("Enter the student information:");
System.out.println("SID:");
String sid = readkey.readLine();
System.out.println("First Name:");
String firstName = readkey.readLine();
System.out.println("Last Name:");
String lastName = readkey.readLine();
System.out.println("Status (freshman, sophomore, junior, senior, graduate):");
String status = readkey.readLine();
System.out.println("GPA (0 to 4):");
String g = readkey.readLine();
double gpa = Double.parseDouble(g);
System.out.println("Email:");
String email = readkey.readLine();

//Call insert_student procedure:
CallableStatement cs = conn.prepareCall("begin
proj2_procedure.insert_student(:1,:2,:3,:4,:5,:6); end;");
//set the parameters
cs.setString(1,sid);
cs.setString(2,firstName);
cs.setString(3,lastName);
cs.setString(4,status);
cs.setDouble(5,gpa);
cs.setString(6,email);
cs.execute();
cs.close();
}
catch(SQLException ex){
System.out.println("SQL Exception in addstudent function");
System.out.println(ex);
}
catch(Exception e){System.out.println("Exception in addStudent");}
return;
}

//-----------------------------------------------------------------------------------------------------
-----------

public static void getStudentInfo(Connection conn) throws SQLException
{
/*This function in being used to get the infomation of a given student*/
try
{
BufferedReader readKeyBoard = new BufferedReader(new
InputStreamReader(System.in));
System.out.println("Enter the Student ID");
String sid = readKeyBoard.readLine();
CallableStatement cs = conn.prepareCall("begin student_info(:1,:2,:3); end;");
cs.setString(1,sid);
cs.registerOutParameter(2,Types.VARCHAR);
cs.registerOutParameter(3,OracleTypes.CURSOR);

cs.execute();


String showmessage = null;

showmessage =  cs.getString(2);

ResultSet rs = (ResultSet)cs.getObject(3);


//condition to check error in pl/sql

if(showmessage ==  null)

{

//Print the results of procedure

System.out.println("SID\t\tLNAME\t\tSTATUS\t\tCLASSID\t\tCOURSE");

System.out.println("------------------------------------------------------------------------------
---------------------------------");

while(rs.next())

{

System.out.println(rs.getString(1) + "\t\t" + rs.getString(2) + "\t\t" +
rs.getString(3) + "\t\t" + rs.getString(4)+ "\t\t"+ rs.getString(5));

}

}

else

{

System.out.println(showmessage);

}
```

```
cs.close();
}
catch(SQLException ex){
System.out.println("SQLException in getStudentInfo function");
}
catch(Exception e){
System.out.println("Exception in getStudentInfo");
}

return;
}

//----------------------------------------------------------------------------------------------------
-----------------

public static void classInfoStudents(Connection conn)
{
try
{
BufferedReader readKeyBoard = new BufferedReader(new
InputStreamReader(System.in));
System.out.println("Enter the class ID");
String cid = readKeyBoard.readLine();
CallableStatement cs = conn.prepareCall("begin class_info(:1,:2,:3); end;");
cs.setString(1,cid);
cs.registerOutParameter(2,Types.VARCHAR);

cs.registerOutParameter(3,OracleTypes.CURSOR);

cs.execute();


String showmessage = null;

showmessage =  cs.getString(2);

ResultSet rs = (ResultSet)cs.getObject(3);


if(showmessage == null)

{

//Print the results class info

System.out.println("CLASSID\t\tTITLE\t\tSID\t\t\tLNAME\t\tEMAIL");

System.out.println("--------------------------------------------------------------------------------
----------------------------------------------");

while(rs.next())

{

System.out.println(rs.getString(1) + "\t" + rs.getString(2) + "\t\t" + rs.getString(3)
+ "\t\t" + rs.getString(4)+ "\t\t"+ rs.getString(5));

}

}

else

{

System.out.println(showmessage);

}

cs.close();
```

```
}
catch(SQLException ex){
System.out.println("SQL Exception in classInfoStudents function");
}
catch(Exception e){
System.out.println("Exception in classInfoStudents");
}

return;
}

//---------------------------------------------------------------------------------------------

public static void deleteStudent(Connection conn) throws SQLException
{
/*This function is being used to delete a student*/
try
{
BufferedReader readKey = new BufferedReader(new
InputStreamReader(System.in));
System.out.println("Enter the student SID to delete");
String sid = readKey.readLine();
CallableStatement cs = conn.prepareCall("begin delete_student(:1); end;");
cs.setString(1,sid);
cs.execute();
cs.close();
}
catch(SQLException ex){
System.out.println("SQLException in delete Student function");
}
catch(Exception e){
System.out.println("Exception in delete Student function");
}
}

//----------------------------------------------------------------------------------------------------

public static void getCoursePrereq(Connection conn) throws SQLException
{
try
{
BufferedReader readKey = new BufferedReader(new
InputStreamReader(System.in));
System.out.println("Enter the DeptCode");
String deptCode = readKey.readLine();
System.out.println("Enter the course number");
String coursen = readKey.readLine();
int courseNo = Integer.parseInt(coursen);
int count = 0;
String result;
```

```
CallableStatement cs = conn.prepareCall("begin course_info(:1,:2,:3,:4); end;");

cs.setString(1,deptCode);

cs.setInt(2,courseNo);

cs.setInt(3,count);

cs.registerOutParameter(4, Types.VARCHAR);

cs.execute();

result = cs.getString(4);

//Print the results

System.out.println("PRE-REQUISITE COURSES");

System.out.println("-------------------------------------------------------------------------------------
-------------");

System.out.println(result);


cs.close();

}

catch(SQLException ex){

System.out.println("SQL Exception in getCoursePrereq funtion");

}

catch(Exception e){

System.out.println("Exception in getCoursePrereq function");

}

}


//-----------------------------------------------------------------------------------------------------------
```

```
public static void enrollStudent(Connection conn) throws SQLException

{

try

{

BufferedReader readKeyBoard = new BufferedReader(new
InputStreamReader(System.in));

System.out.println("Enter the student ID");

String sid = readKeyBoard.readLine();

System.out.println("Enter the class ID");

String cid = readKeyBoard.readLine();

CallableStatement cs = conn.prepareCall("begin enroll_student(:1,:2,:3); end;");

cs.setString(1,sid);

cs.setString(2,cid);

cs.registerOutParameter(3,Types.VARCHAR);

cs.execute();


String showmessage = null;

showmessage =  cs.getString(3);

System.out.println(showmessage);


cs.close();

}

catch(SQLException ex){

System.out.println("SQL Exception in enrollment function");

}
```

```
catch(Exception e){

System.out.println("Exception in enrollement");

}


return;

}


//----------------------------------------------------------------------------------------------------
----


public static void dropStudent(Connection conn) throws SQLException

{

try

{

BufferedReader readKeyBoard = new BufferedReader(new
InputStreamReader(System.in));

System.out.println("Enter the student ID");

String sid = readKeyBoard.readLine();

System.out.println("Enter the class ID");

String cid = readKeyBoard.readLine();

CallableStatement cs = conn.prepareCall("begin delete_enrollment(:1,:2,:3);
end;");

cs.setString(1,sid);

cs.setString(2,cid);

cs.registerOutParameter(3,Types.VARCHAR);
cs.execute();

String showmessage = null;
showmessage =  cs.getString(3);
System.out.println(showmessage);

cs.close();
}
catch(SQLException ex){
System.out.println("SQL Exception in dropstudent function");
}
catch(Exception e){
System.out.println("Exception in dropstudent");
}

return;
}
}
```