CS532 : Database Systems

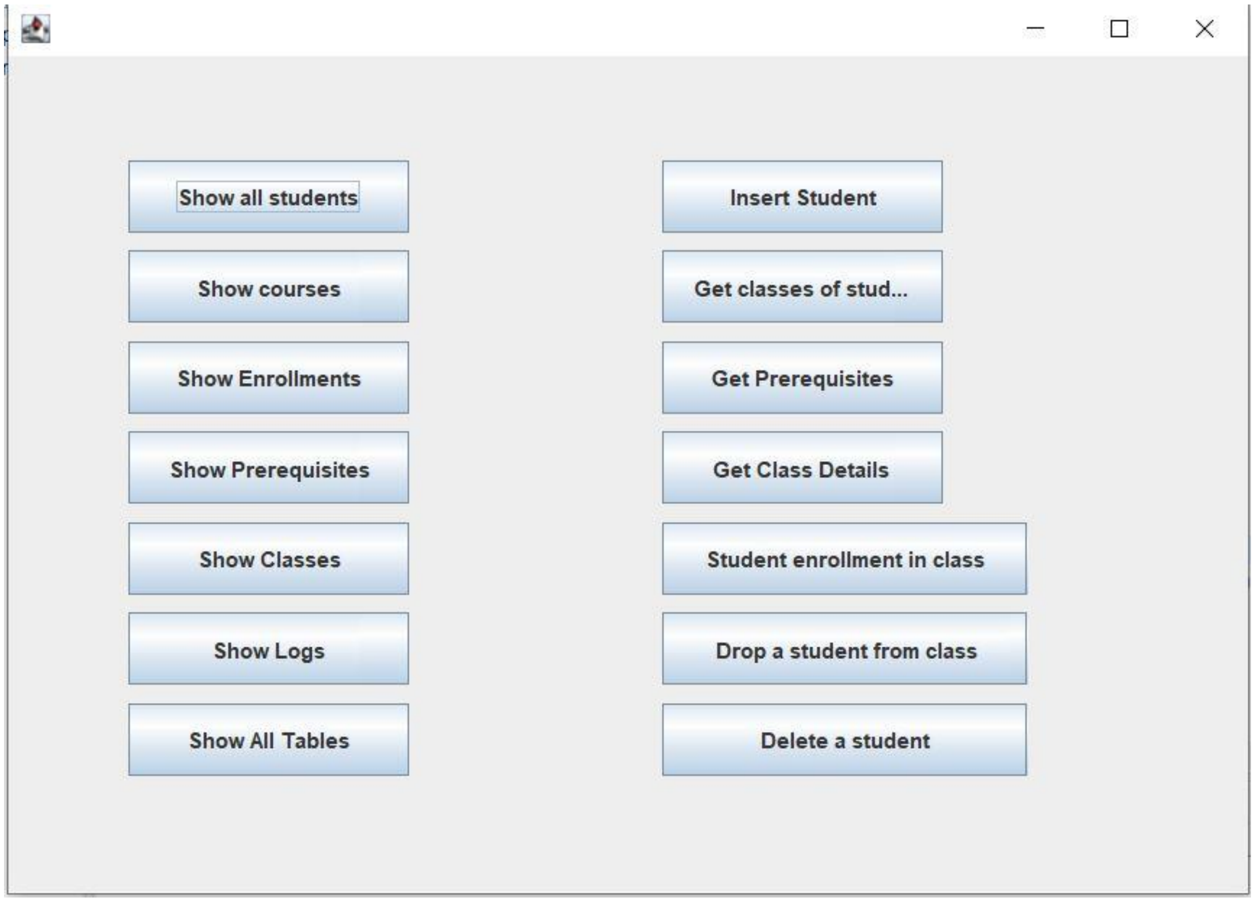
Project 2.

Team Members : Shubham Patwa, Juhi Yadav

Report :

We have created a graphical interface for this project.

On the left side, all tables can be shown while on the right side, all procedures are run as shown in the figure.



1. **Sequence for logs:**

```
create sequence seq1
increment by 1
start with 1000001
minvalue 000000
maxvalue 9999999
nocycle
cache 20;
```

Sequence was created as shown above.
Started with 1000001 since, a 7 digit sequence was needed.

2. The left side shows all tables:
Students, Courses, Classes, Prerequisistes, Enrollments, Logs

For this we have used the following procedures:

Show Students:

```
procedure
show_students(list out sys_refcursor)
is
begin
    open list for
    select * from students;

end;
```

Show Courses:

```
procedure
show_courses(list out sys_refcursor)
is
begin
    open list for
    select * from courses;

end;
```

Show Prerequisites:

```
procedure
show_prerequisites(list out sys_refcursor)
is
begin
    open list for
    select * from prerequisites;

end;
```

Similar procedure was followed for the remaining tables.
As we can see, we have used a cursor to return multiple rows to java console.

3. Insert into Students table:

```
procedure
insert_values(sid_in in students.sid%type,
firstname_in in students.firstname%type,
lastname_in in students.lastname%type,
status_in in students.status%type,
gpa_in in students.gpa%type,
email_in in students.email%type,message out varchar2) is
counter number;
begin
    select count(sid) into counter from students where sid = sid_in;
    if(counter>0) then
        message := 'Error  Sid exists';
    elsif( status_in not in ('freshman', 'sophomore', 'junior', 'senior', 'graduate')) then
        message := 'Error . status wrongly defined.';
    else
        insert into students values (sid_in, firstname_in, lastname_in, status_in, gpa_in,email_in);
        message := 'Successful';
    end if;
end;
```

I have made two checks here, where sid exists and whether the status follows the guidelines.

Rest of it, is just if else loop.

4. Get the student's associated data and classes that he/she is enrolled in:

```
procedure
get_classes(sid_in students.sid%type,message out varchar2,list out sys_refcursor)
is
counter number;
begin
    select count(students.sid) into counter from students where students.sid = sid_in;

    if (counter > 0) then
        select count(enrollments.sid) into counter from enrollments where enrollments.sid = sid_in;
        if (counter > 0) then
            open list for
            select s.sid,s.firstname,s.lastname,s.status,e.classid,concat (c.dept_code, c.course_no) as Course_id, c1.title from Students s,Enrollments e, Classes c, Courses c1 where s.sid = e.sid and e.classid = c.classid and c.dept_code = c1.dept_code and c.course_no = c1.course_no and e.sid = sid_in;
        else
            message := 'Student has not enrolled in any classes';
        end if;
    else
    message:= 'Invalid sid for student';
    end if;
end;
```

Similarly, here too, it goes through multiple If-else loops.

After all the conditions are checked, such as whether sid exists and whether that student has enrolled in any classes.

After that, the query is run.

5. Get the prerequisite courses of a course.

```
procedure
get_prerequisites(dept_code_in in prerequisites.dept_code%type,course_no_in in prerequisites.course_no%type,result out varchar2,checkvar in number,prereq out number)
is
CURSOR crefcur is
    select pre_dept_code,pre_course_no from prerequisites where dept_code = dept_code_in and course_no = course_no_in;
    course_row crefcur%rowtype;
    counter number;

    begin
        select count(*) into counter from Courses where dept_code = dept_code_in and course_no = course_no_in;
        open crefcur;
        fetch crefcur into course_row;
        if(crefcur%found) then
            while(crefcur%found) loop
                get_prerequisites(course_row.pre_dept_code, course_row.pre_course_no,result,0,prereq);
                if(result is NULL) then
                    result := course_row.pre_dept_code || ' ' || course_row.pre_course_no;
                    prereq := 1;
                else
                    result := result || ',' || course_row.pre_dept_code || ' ' ||  course_row.pre_course_no;
                    prereq := prereq+1;
                end if;
                fetch crefcur into course_row;
            end loop;
        elsif (counter = 1 and checkvar = 1) then
            result := 'No prerequisite';
            prereq := 0;
        elsif (counter < 1 and checkvar = 1) then
            result := 'This course does not exists';
            prereq := 0;
        end if;
end get_prerequisites;
```

This was complicated to design since the prerequisite course could also have a prerequisite and it could go upto 2 levels deep.

We implemented a recursive function for this, no other way was possible.

We made an additional functionality where it returns the total number of prerequisite courses. This helped in further queries.

Using a recursive procedure, the result was concatenated by a comma and a space in between.

This regex was used in query 7 and 8 for processing and checking regarding the prerequisite courses.

6. Display the students that are enrolled in a class and the class details:

```
procedure
query_6(class_id_in in Classes.classid%type,message out varchar2,list out sys_refcursor
)
is
counter number;
begin
    select count(classes.classid) into counter from Classes where Classes.classid = class_id_in;
    if (counter >0 ) then
        select count(s.sid) into counter from Students s,Enrollments e, Classes c, Courses c1 where s.sid = e.sid and e.classid = c.classid and c.course_no = c1.course_no and c.dept_code = c1.dept_code and c.classid = class_id_in;
        if (counter > 0) then
            open list for
            select s.sid,s.firstname,s.lastname,c.classid,c1.title,c.semester,c.year from Students s,Enrollments e, Classes c, Courses c1 where s.sid = e.sid and e.classid = c.classid and c.course_no = c1.course_no and c.dept_code = c1.dept_code and c.classid = class_id_in;
        else
            message := 'No students found';
        end if;
    else
    message := 'Invalid class id';
    end if;
end;
```

Class id is taken as input .

It first checks whether the class id exists.

After that, it checks whether there are students in the class.

If true, it returns all the students in a class.

7. **Enroll a student in a class:**

```
procedure
query_7(sid_in in Students.sid%type,class_id_in in Classes.classid%type,message out varchar2,truthvalue out number)
is
counter number;
limits number;
sizes number;
begin
    truthvalue := 0;
    select count(students.sid) into counter from students where students.sid = sid_in;

    if (counter > 0) then
        select count(classes.classid) into counter from Classes where classes.classid = class_id_in;
        if (counter > 0) then
            select limit into limits from Classes where classid = class_id_in;
            select class_size into sizes from Classes where classid = class_id_in;
            if (limits - sizes > 0) then
                select count(e.sid) into counter from Enrollments e where e.classid = class_id_in and e.sid = sid_in;
                if (counter > 0) then
                    message := 'Student is already enrolled in this class.';
                else
                    select count(*) into counter from (select count(e.classid) from Enrollments e,Classes c where c.classid = e.classid and e.sid = sid_in having count(*) > 1 group by (e.sid,c.year,c.semester));
                    if (counter > 0) then
                        message := 'You are overloaded';
                    else
                        message := 'All conditions satisifed.Proceed for prerequisite check';
                        truthvalue := 1;
                    end if;
                end if;


            else
                message := 'The class is full';
            end if;

        else
            message := 'class id is invalid';
        end if;

    else
        message := 'Sid not found';
    end if;
end query_7;
```

Firstly , the given required conditions are checked such as

Whether the sid exists, whether the class exists, whether the student is overloaded, etc.

Once the conditions are satisfied, we proceed and run prerequisites query and get the prerequisites.Then , for each prerequisite, we run another query to check the grade:

```
procedure
check_grade(sid_in in students.sid%type,
dept_code_in in Classes.dept_code%type,
course_no_in in Classes.course_no%type,message out varchar2,truthvalue out number) is
counter number;
begin
    truthvalue := 0;
    select count(e.lgrade) into counter from Classes c,Enrollments e where e.classid = c.classid and e.sid =sid_in and c.dept_code =dept_code_in and c.course_no=course_no_in;

    if(counter >0 ) then

        select count(e.lgrade) into counter from Classes c,Enrollments e where e.classid = c.classid and e.sid =sid_in and c.dept_code =dept_code_in and c.course_no=course_no_in and e.lgrade not in ('A','A-','B+','B','B-','C+','C','C-');
        if(counter > 0) then
            message := 'Student has failed in prerequisites.';
            truthvalue := 0;
        else
            message := 'Case passed';
            truthvalue :=1;
        end if;
    else
        message:= 'Prerequisite courses have not been completed';
        truthvalue := 0;
    end if;
end check_grade;
```

Once, all the conditions are satisfied, we run another query to enroll student into class.

```
procedure
final_enroll(sid_in in students.sid%type,
class_id_in in Classes.classid%type) is
counter number;
begin
    insert into enrollments values(sid_in, class_id_in, null);
    commit;
end;
```

8 Drop a student from a class:

```
procedure
query_8(sid_in in Students.sid%type,class_id_in in Classes.classid%type,message out varchar2,truthvalue out number,list out sys_refcursor,dept_code1 out varchar2,course_no1 out varchar2)
is
counter number;
begin
    truthvalue := 0;
    select count(sid) into counter from students where sid = sid_in;
    if(counter > 0) then
        select count(classid) into counter from Classes where classid = class_id_in;
        if (counter >0) then
            select count(*) into counter from enrollments where sid = sid_in and classid = class_id_in;
            if(counter >0 ) then
                truthvalue := 1;
                select c.dept_code into dept_code1 from Students s,Enrollments e, Classes c where s.sid = e.sid and e.classid = c.classid and e.sid = sid_in and e.classid = class_id_in;
                select c.course_no into course_no1 from Students s,Enrollments e, Classes c where s.sid = e.sid and e.classid = c.classid and e.sid = sid_in and e.classid = class_id_in;
                open list for
                select c.dept_code,c.course_no from Students s,Enrollments e, Classes c where s.sid = e.sid and e.classid = c.classid and e.sid = sid_in minus (select c.dept_code,c.course_no from Students s,Enrollments e, Classes c where s.sid = e.sid and e.classid = c.classid and e.sid = sid_in and e.classid = class_id_in);
            else
                message := 'Student not enrolled in class';
            end if;
        else
```

```
            message := 'Class id not found';
        end if;
    else
        message := 'sid not found';
    end if;
end query_8;
```

This is also a complex procedure.

Here too, initially all the conditions are checked such are sid exists or not, class id is valid or not.

Once all trivial conditions are met, it return the dept_code and course_no for that class to check for the prerequisite condition.

Then, using previous prerequisite query, all prerequisites are found.

Condition for not being a prerequisite is checked by using a loop.

Once, all the conditions are met, procedure is run.

```
procedure
drop_student(sid_in in students.sid%type,class_id_in in Classes.classid%type,message ou
t varchar2)
is
begin
    delete from enrollments where sid= sid_in and classid=class_id_in;
    commit;
    message := 'Sucessfully dropped';
end drop_student;
```

9. Delete a student from students' table:

```
procedure
delete_a_student(sid_in in students.sid%type,message out varchar2)
is
counter number;
begin
    select count(*) into counter from students where sid = sid_in;
    if (counter = 0) then
        message := 'sid not found';
    else
        delete from students where students.sid = sid_in;
        commit;
        message := 'Delete successful';
    end if;


end delete_a_student;
```

Conditions are checked that is existence of sid in table and then the sid is dropped successfully.

However, before dropping , triggers are executed.

10. Triggers :

Triggers are created whenever a student is added to or deleted from the students table, or when a student is successfully enrolled into or dropped from a class.

```
create or replace trigger student_insertion_trigger
after insert on students
for each row
declare
user_name varchar2(15);
begin
select user into user_name from dual;
insert into logs values(seq1.nextval,user_name,sysdate,'students','insert',:new.sid);
end;
/
show errors
```

```
create or replace trigger delete_a_student_trigger
before delete on students
for each row
declare
sid_in char(4);
user_name varchar2(15);
begin
sid_in := :old.sid;
delete from enrollments where sid = sid_in;
select user into user_name from dual;
insert into logs values(seq1.nextval,user_name,sysdate,'students','delete',sid_in);
end;
/
show errors
```

```
create or replace trigger enroll_a_student
after insert on enrollments
for each row
declare
user_name varchar2(15);
classid_in Classes.classid%type;
begin
classid_in := :new.classid;
update Classes set class_size = class_size +1 where classid = classid_in;
select user into user_name from dual;
insert into logs values(seq1.nextval,user_name,sysdate,'enrollments','insert',classid_i
n);
end;
/
show errors
```

```sql
create or replace trigger remove_from_classes_trigger
before delete on enrollments
for each row
declare
user_name varchar2(15);
classid_in Classes.classid%type;
begin
classid_in := :old.classid;
update Classes set class_size = class_size -1 where classid = classid_in;
select user into user_name from dual;
insert into logs values(seq1.nextval,user_name,sysdate,'enrollments','insert',classid_in);
end;
/
show errors
```

Java Code :

Driver :

```java
package project2;
import java.sql.Connection;
import java.sql.SQLException;

import oracle.jdbc.pool.OracleDataSource;
public class Driver {

    static Connection conn;
    public static void  start_connection(){

        try {
            OracleDataSource ds = new oracle.jdbc.pool.OracleDataSource();
            ds.setURL("jdbc:oracle:thin:@castor.cc.binghamton.edu:1521:ACAD111");
            conn = ds.getConnection("spatwa1", "98985432");
        }

            catch (SQLException ex) { System.out.println ("\n*** SQLException caught ***\n");}
            catch (Exception e) {System.out.println ("\n*** other Exception caught ***\n");}

        }


    public static void end_connection() {
        try {

            conn.close();
        }
        catch (SQLException ex) { System.out.println ("\n*** SQLException caught ***\n");}
        catch (Exception e) {System.out.println ("\n*** other Exception caught ***\n");}

    }




}
```

ClassDetails.java :

```
package project2;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.sql.CallableStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Types;

import oracle.jdbc.OracleTypes;

public class ClassDetails {

    public void getClassDetails() {

        try {

            BufferedReader  readKeyBoard;
               String          classid;
               readKeyBoard = new BufferedReader(new InputStreamReader(System.in));
               System.out.print("Please Enter class_id:");
               classid = readKeyBoard.readLine();
                CallableStatement cs = Driver.conn.prepareCall("begin project2.query_6(:1,:2,:3); end;");
                 cs.setString(1, classid);

                cs.registerOutParameter(2,Types.VARCHAR);
                cs.registerOutParameter(3, OracleTypes.CURSOR);

                cs.executeQuery();
            String message = cs.getString(2);
            if(message != null){
               System.out.println(message);
            }


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

                // print the results
                while (rs.next()) {
                    System.out.println(rs.getString(1) + "\t" +
                       rs.getString(2) + "\t" + rs.getString(3)  + "\t" +
                        rs.getString(4) +
                        "\t" +
                        rs.getString(5)+ "\t" +
                        rs.getString(6) +"\t" +
```

```java
                        rs.getString(7));
                }

                //close the result set, statement, and the connection
                cs.close();

        }
        catch (SQLException ex) { System.out.println ("\n*** SQLException caught ***\n" + ex.getMessage());}
           catch (Exception e) {System.out.println ("\n*** other Exception caught ***\n");}

    }


}
```

DeleteStudent.java:

```java
package project2;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.sql.CallableStatement;
import java.sql.Types;

public class DeleteStudent {


    public void deleteStudent() {

        try {

            BufferedReader  readKeyBoard;
            String          sid;
            readKeyBoard = new BufferedReader(new InputStreamReader(System.in));
            System.out.print("Please Enter sid:");
            sid = readKeyBoard.readLine();

            CallableStatement cs = Driver.conn.prepareCall("begin project2.delete_a_student(:1,:2); end;");

            cs.setString(1,  sid);
            cs.registerOutParameter(2, Types.VARCHAR);
            cs.executeQuery();
            System.out.println(cs.getString(2));
            cs.close();
        }

        catch (Exception e) {System.out.println ("\n*** other Exception caught ***\n");}

    }

}
```

Drop.java:

```java
package project2;
```

```java

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.sql.CallableStatement;
import java.sql.ResultSet;
import java.sql.Types;

import oracle.jdbc.OracleTypes;

public class Drop {

    public void dropAStudent() {

        try {
            BufferedReader  readKeyBoard;
            String          sid,classid;
            readKeyBoard = new BufferedReader(new InputStreamReader(System.in));
            System.out.print("Please Enter sid:");
            sid = readKeyBoard.readLine();
            readKeyBoard = new BufferedReader(new InputStreamReader(System.in));
            System.out.print("Please Enter classid:");
            classid = readKeyBoard.readLine();


            CallableStatement cs = Driver.conn.prepareCall("begin project2.query_8(:1,:2,:3,:4,:5,:6,:7); end;");

            cs.setString(1,  sid);
            cs.setString(2,  classid);
            cs.registerOutParameter(3,Types.VARCHAR);
            cs.registerOutParameter(4, OracleTypes.NUMBER);
            cs.registerOutParameter(5,OracleTypes.CURSOR);
            cs.registerOutParameter(6, Types.VARCHAR);
            cs.registerOutParameter(7,Types.VARCHAR);
            cs.executeQuery();
            String message = cs.getString(3);
            String truthvalue = cs.getString(4);
            if(Integer.valueOf(truthvalue) == 0) {
                System.out.println(message);
                return ;
            }
            else {

                String dept_code = cs.getString(6);
                String course_no = cs.getString(7);
                ResultSet rs = (ResultSet)cs.getObject(5);

                while (rs.next()) {

                    GetPrerequisites getPreq = new GetPrerequisites();
                    String prereq = getPreq.getPrerequisites(rs.getString(1), rs.getString(2));
                    int i = Integer.valueOf(getPreq.getNewvar());
                    if(i!=0) {
                        String[] parts = prereq.split(",");
                        for(String x : parts) {
                            String[] part = x.split(" ");
                            if(part[0].equals(dept_code) && part[1].equals(course_no)) {
                                System.out.println("Cannot drop course. It is a prerequisite of an already enrolled course");
                                return;
                            }
                            else {

                            }
                        }
                    }

                }
                cs = Driver.conn.prepareCall("begin project2.drop_student(:1,:2,:3); end;");

                cs.setString(1,  sid);
                cs.setString(2,  classid);
                cs.registerOutParameter(3,Types.VARCHAR);
                cs.executeQuery();
                System.out.println(cs.getString(3));


                cs.close();


            }







        }


        catch (Exception e) {System.out.println ("\n*** other Exception caught ***\n");}


    }


}
```

Enroll.java :

```
package project2;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.sql.CallableStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.Types;

import oracle.jdbc.OracleTypes;

public class Enroll {

    public void enrollAStudent() {

        try {
```

```java

            BufferedReader  readKeyBoard;
            String          sid,classid;
            readKeyBoard = new BufferedReader(new InputStreamReader(System.in));
            System.out.print("Please Enter sid:");
            sid = readKeyBoard.readLine();
            readKeyBoard = new BufferedReader(new InputStreamReader(System.in));
            System.out.print("Please Enter classid:");
            classid = readKeyBoard.readLine();
            CallableStatement cs = Driver.conn.prepareCall("begin project2.query_7(:1,:2,:3,:4); end;");
            cs.setString(1,sid);
            cs.setString(2, classid);
            cs.registerOutParameter(3,Types.VARCHAR);
            cs.registerOutParameter(4, OracleTypes.NUMBER);
            cs.executeQuery();
            String message = cs.getString(3);
            String value = cs.getString(4);
            if( Integer.valueOf(value) == 0) {
                System.out.println(message);
                return ;
            }
            else {

                 Statement stmt = Driver.conn.createStatement ();

                  // Save result
                ResultSet rset;
                rset = stmt.executeQuery ("SELECT dept_code,course_no FROM Classes where classid='"+classid+"'");
                rset.next ();
                String dept_code = rset.getString(1);
                    String course_no = rset.getString(2);


                    GetPrerequisites getPre = new GetPrerequisites();
                    String prereq = getPre.getPrerequisites(dept_code, course_no);
                    int i = Integer.valueOf(getPre.getNewvar());
                    if(i != 0) {
                        String[] parts = prereq.split(",");
                        for(String x: parts) {
                            String[] part = x.split(" ");
                            cs = Driver.conn.prepareCall("begin project2.check_grade(:1,:2,:3,:4,:5); end;");
                            cs.setString(1,sid);
                            cs.setString(2, part[0]);
                            cs.setString(3,part[1]);
                            cs.registerOutParameter(4,Types.VARCHAR);
                            cs.registerOutParameter(5, OracleTypes.NUMBER);
                            cs.executeQuery();
                            String message1 =  cs.getString(4);
                            String value1 = cs.getString(5);
                            if(Integer.valueOf(value1) == 0) {
                                System.out.println(message1);
                                return;
                            }
                            else {
                                continue;
                            }





                        }
                    }
                    cs = Driver.conn.prepareCall("begin project2.final_enroll(:1,:2); end;");
                    cs.setString(1,sid);
                    cs.setString(2, classid);
                    cs.executeQuery();
                    System.out.println("Enrollment Successful");



            }




        }
        catch (SQLException ex) { System.out.println ("\n*** SQLException caught ***\n" + ex.getMessage());}
        catch (Exception e) {System.out.println ("\n*** other Exception caught ***\n"); }
    }

}
```

GetClasses.java:

```java
package project2;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.sql.CallableStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Types;

import oracle.jdbc.OracleTypes;

public class GetClasses {



    public void getClasses() {
        try {

            BufferedReader  readKeyBoard;
            String          sid;
            readKeyBoard = new BufferedReader(new InputStreamReader(System.in));
            System.out.print("Please Enter SID:");
            sid = readKeyBoard.readLine();
            CallableStatement cs = Driver.conn.prepareCall("begin project2.get_classes(:1,:2,:3); end;");
            cs.setString(1, sid);
            cs.registerOutParameter(2,Types.VARCHAR);
            cs.registerOutParameter(3, OracleTypes.CURSOR);
            cs.executeQuery();
            String message = cs.getString(2);
            if(message != null){
                System.out.println(message);
            }
            ResultSet rs = (ResultSet)cs.getObject(3);

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

            cs.close();}
        catch (SQLException ex) { System.out.println ("\n*** SQLException caught ***\n" + ex.getMessage());}
        catch (Exception e) {System.out.println ("\n*** other Exception caught ***\n");}
    }


}
```

GetPrerequisites:

```
package project2;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.sql.CallableStatement;
import java.sql.SQLException;
import java.sql.Types;

import oracle.jdbc.OracleTypes;

public class GetPrerequisites {
    String newvar;
    public void getPrerequisites() {
        try {
        String dept_code,course_no,checkvar;

          BufferedReader  readKeyBoard;

          readKeyBoard = new BufferedReader(new InputStreamReader(System.in));
           System.out.print("Please Enter dept_code:");
           dept_code = readKeyBoard.readLine();
           readKeyBoard = new BufferedReader(new InputStreamReader(System.in));
           System.out.print("Please Enter course no:");
          course_no = readKeyBoard.readLine();
          checkvar = "1";
            CallableStatement cs = Driver.conn.prepareCall("begin project2.get_prerequisites(:1,:2,:3,:4,:5); end;");
             cs.setString(1,dept_code);
             cs.setString(2, course_no);
             cs.setString(4, checkvar);


            cs.registerOutParameter(3,Types.VARCHAR);
            cs.registerOutParameter(5, OracleTypes.NUMBER);

            cs.executeQuery();
         String message = cs.getString(3);

         if(message != null){
            System.out.println(message);
         }
            cs.close();
    }
    catch (SQLException ex) { System.out.println ("\n*** SQLException caught ***\n" + ex.getMessage());}
       catch (Exception e) {System.out.println ("\n*** other Exception caught ***\n");}




    }

    public String getPrerequisites(String dept_code,String course_no) {
        String message = null;
        try {


            String checkvar = "1";
            CallableStatement cs = Driver.conn.prepareCall("begin project2.get_prerequisites(:1,:2,:3,:4,:5); end;");
             cs.setString(1,dept_code);
             cs.setString(2, course_no);
             cs.setString(4, checkvar);


            cs.registerOutParameter(3,Types.VARCHAR);
            cs.registerOutParameter(5, OracleTypes.NUMBER);

            cs.executeQuery();
      message = cs.getString(3);
         newvar = cs.getString(5);

            cs.close();
            return message;
    }

    catch (SQLException ex) { System.out.println ("\n*** SQLException caught ***\n" + ex.getMessage());}
       catch (Exception e) {System.out.println ("\n*** other Exception caught ***\n");

       }
        return message;


    }

    public String getNewvar() {
        return newvar;
    }




```

```java
}
```

InsertIntoStudent:

```java
package project2;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.sql.CallableStatement;
import java.sql.SQLException;
import java.sql.Types;

public class InsertIntoStudent {

    String status;

    public void insert() {


        BufferedReader  readKeyBoard;
        String          sid,firstname,lastname, status_in, gpa,email ;
        readKeyBoard = new BufferedReader(new InputStreamReader(System.in));
        System.out.print("Please Enter SID:");
        try {
        sid = readKeyBoard.readLine();
        readKeyBoard = new BufferedReader(new InputStreamReader(System.in));
        System.out.print("Please Enter firstname:");
        firstname = readKeyBoard.readLine();
        readKeyBoard = new BufferedReader(new InputStreamReader(System.in));
        System.out.print("Please Enter lastname:");
        lastname = readKeyBoard.readLine();
        readKeyBoard = new BufferedReader(new InputStreamReader(System.in));
        System.out.print("Please Enter level of education :");
        status_in = readKeyBoard.readLine();
        readKeyBoard = new BufferedReader(new InputStreamReader(System.in));
        System.out.print("Please Enter GPA:");
        gpa = readKeyBoard.readLine();
        readKeyBoard = new BufferedReader(new InputStreamReader(System.in));
        System.out.print("Please Enter email:");
        email = readKeyBoard.readLine();

        CallableStatement cs = Driver.conn.prepareCall("begin project2.insert_values(:1
,:2,:3,:4,:5,:6,:7); end;");

        //set the in parameter (the first parameter)
        cs.setString(1, sid);
        cs.setString(2, firstname);
        cs.setString(3, lastname);
        cs.setString(4, status_in);
        cs.setString(5, gpa);
        cs.setString(6, email);
        cs.registerOutParameter(7, Types.VARCHAR);

        cs.executeQuery();
        status = cs.getString(7);
         System.out.println(status);
        cs.close();


        }
        catch (SQLException ex) { System.out.println ("\n*** SQLException caught ***\n" + status);}
        catch (Exception e) {System.out.println ("\n*** other Exception caught ***\n"); }





    }

}
```

ShowQueries.java:

```java
package project2;

import java.sql.CallableStatement;
import java.sql.ResultSet;

import oracle.jdbc.OracleTypes;

public class ShowQueries {



    public void show_students() {
        try {
            CallableStatement cs = Driver.conn.prepareCall("begin project2.show_students(:1); end;");
            cs.registerOutParameter(1, OracleTypes.CURSOR);
            cs.execute();
            ResultSet rs = (ResultSet)cs.getObject(1);
            System.out.println("SHOW STUDENTS QUERY :");
        while (rs.next()) {
            System.out.println(rs.getString(1) + "\t" +
                rs.getString(2) + "\t" + rs.getString(3) +  "\t" +
                rs.getString(4) + "\t" + rs.getString(5) +  "\t" +    rs.getString(6));

        }
        System.out.println("\n");
        cs.close();

        }
        catch(Exception e1) {
            e1.printStackTrace();
        }


    }


    public void show_courses() {
        try {
            CallableStatement cs = Driver.conn.prepareCall("begin project2.show_courses(:1); end;");
            cs.registerOutParameter(1, OracleTypes.CURSOR);
            cs.execute();
            ResultSet rs = (ResultSet)cs.getObject(1);
            System.out.println("SHOW COURSES QUERY :");
        while (rs.next()) {
            System.out.println(rs.getString(1) + "\t" +
                rs.getString(2) + "\t" + rs.getString(3));
        }
        System.out.println("\n");
        cs.close();
        }
        catch(Exception e1) {
            e1.printStackTrace();
        }
    }

    public void show_enrollments() {
        try {
            CallableStatement cs = Driver.conn.prepareCall("begin project2.show_enrollm
ents(:1); end;");
            cs.registerOutParameter(1, OracleTypes.CURSOR);
            cs.execute();
            ResultSet rs = (ResultSet)cs.getObject(1);
            System.out.println("SHOW Enrollments QUERY :");
        while (rs.next()) {
            System.out.println(rs.getString(1) + "\t" +
                rs.getString(2) + "\t" + rs.getString(3));
        }
        System.out.println("\n");
        cs.close();
        }
        catch(Exception e1) {
            e1.printStackTrace();
        }


    }

    public void show_prerequisites() {
        try {
            CallableStatement cs = Driver.conn.prepareCall("begin project2.show_prerequ
isites(:1); end;");
            cs.registerOutParameter(1, OracleTypes.CURSOR);
            cs.execute();
            ResultSet rs = (ResultSet)cs.getObject(1);
            System.out.println("SHOW Prerequisites QUERY :");
        while (rs.next()) {
            System.out.println(rs.getString(1) + "\t" +
                rs.getString(2) + "\t" + rs.getString(3) + "\t" + rs.getString(4));
        }
        System.out.println("\n");
        cs.close();
        }
        catch(Exception e1) {
            e1.printStackTrace();
        }
    }

    public void show_classes() {
        try {
            CallableStatement cs = Driver.conn.prepareCall("begin project2.show_classes
(:1); end;");
            cs.registerOutParameter(1, OracleTypes.CURSOR);
            cs.execute();
```

```
                    ResultSet rs = (ResultSet)cs.getObject(1);
                    System.out.println("SHOW Classes:");
                    while (rs.next()) {
                        System.out.println(rs.getString(1) + "\t" +
                            rs.getString(2) + "\t" + rs.getString(3) +  "\t" +
                            rs.getString(4) +
                            "\t" + rs.getString(5) + "\t" +
                            rs.getString(6) +
                            "\t" + rs.getString(7) + "\t" +
                            rs.getString(8));
                    }
                System.out.println("\n");
                cs.close();
                }
                catch(Exception e1) {
                    e1.printStackTrace();
                }

            }

            public void show_logs() {
                try {
                    CallableStatement cs = Driver.conn.prepareCall("begin project2.show_logs(:1); end;");
                    cs.registerOutParameter(1, OracleTypes.CURSOR);
                    cs.execute();
                    ResultSet rs = (ResultSet)cs.getObject(1);
                    System.out.println("SHOW Logs:");
                    while (rs.next()) {
                        System.out.println(rs.getString(1) + "\t" +
                            rs.getString(2) + "\t" + rs.getString(3) +  "\t" +
                            rs.getString(4) +
                            "\t" + rs.getString(5) + "\t" +
                            rs.getString(6));
                    }
                System.out.println("\n");
                cs.close();
                }
                catch(Exception e1) {
                    e1.printStackTrace();
                }


            }
        }
```

Start.java:

```
package project2;
import oracle.jdbc.*;
import java.math.*;
import java.io.*;
import java.awt.*;
import oracle.jdbc.pool.OracleDataSource;

import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JButton;
import java.awt.event.ActionListener;
```

```java
import java.sql.CallableStatement;
import java.sql.ResultSet;
import java.sql.Statement;
import java.awt.event.ActionEvent;
import javax.swing.SwingConstants;

public class Start {

    private JFrame frame;
    ShowQueries show;

    /**
     * Launch the application.
     */
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    Start window = new Start();
                    window.frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    /**
     * Create the application.
     */
    public Start() {
        initialize();
        Driver.start_connection();
     show = new ShowQueries();
    }

    /**
     * Initialize the contents of the frame.
     */
    private void initialize() {
        frame = new JFrame();
        frame.setBounds(100, 100, 700, 500);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.getContentPane().setLayout(null);

        JButton btnNewButton = new JButton("Show all students");
        btnNewButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                    show.show_students();
            }
        });
        btnNewButton.setBounds(67, 58, 155, 40);
        frame.getContentPane().add(btnNewButton);

        JButton btnNewButton_1 = new JButton("Show courses");
        btnNewButton_1.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
            show.show_courses();



            }
        });
        btnNewButton_1.setBounds(67, 108, 155, 40);
        frame.getContentPane().add(btnNewButton_1);

        JButton button = new JButton("Show Enrollments");
        button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {

                show.show_enrollments();
            }
        });
        button.setBounds(67, 158, 155, 40);
        frame.getContentPane().add(button);

        JButton button_1 = new JButton("Show Prerequisites");
        button_1.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                show.show_prerequisites();
            }
        });
        button_1.setBounds(67, 208, 155, 40);
        frame.getContentPane().add(button_1);

        JButton button_2 = new JButton("Show Classes");
        button_2.addActionListener(new ActionListener() {
                    public void actionPerformed(ActionEvent e) {
                        show.show_classes();




                    }
                });
                button_2.setBounds(67, 258, 155, 40);
                frame.getContentPane().add(button_2);

                JButton button_3 = new JButton("Show Logs");
                button_3.addActionListener(new ActionListener() {
                    public void actionPerformed(ActionEvent e) {
                        show.show_logs();

                    }
                });
                button_3.setBounds(67, 308, 155, 40);
                frame.getContentPane().add(button_3);

                JButton button_4 = new JButton("Insert Student");
                button_4.addActionListener(new ActionListener() {
                    public void actionPerformed(ActionEvent e) {
                        InsertIntoStudent insert = new InsertIntoStudent();
                        insert.insert();



                    }
                });
                button_4.setBounds(362, 58, 155, 40);
                frame.getContentPane().add(button_4);

                JButton btnGetClassesOf = new JButton("Get classes of student ");
                btnGetClassesOf.setHorizontalAlignment(SwingConstants.LEFT);
                btnGetClassesOf.addActionListener(new ActionListener() {
                    public void actionPerformed(ActionEvent e) {
                        GetClasses getClass = new GetClasses();
                        getClass.getClasses();



                    }
                });
                btnGetClassesOf.setBounds(362, 108, 155, 40);
                frame.getContentPane().add(btnGetClassesOf);

                JButton btnGetPrere = new JButton("Get Prerequisites");
                btnGetPrere.addActionListener(new ActionListener() {
                    public void actionPerformed(ActionEvent e) {

                        GetPrerequisites getPrereq = new GetPrerequisites();
                        getPrereq.getPrerequisites();

                    }
                });
                btnGetPrere.setBounds(362, 158, 155, 40);
                frame.getContentPane().add(btnGetPrere);

                JButton button_5 = new JButton("Get Class Details");
                button_5.addActionListener(new ActionListener() {
                    public void actionPerformed(ActionEvent e) {
                        ClassDetails getClassDetails = new ClassDetails();
                        getClassDetails.getClassDetails();

                    }
                });
                button_5.setBounds(362, 208, 155, 40);
                frame.getContentPane().add(button_5);

                JButton btnEnrollAStudent = new JButton("Student enrollment in class");
                btnEnrollAStudent.addActionListener(new ActionListener() {
                    public void actionPerformed(ActionEvent e) {
                        Enroll enrollStudent = new Enroll();
                        enrollStudent.enrollAStudent();

                    }
                });
                btnEnrollAStudent.setBounds(362, 258, 202, 40);
                frame.getContentPane().add(btnEnrollAStudent);

                JButton button_6 = new JButton("Drop a student from class");
                button_6.addActionListener(new ActionListener() {
                    public void actionPerformed(ActionEvent e) {
                        Drop dropper = new Drop();
                        dropper.dropAStudent();
                    }
                });
                button_6.setBounds(362, 308, 202, 40);
                frame.getContentPane().add(button_6);

                JButton btnDeleteAStudent = new JButton("Delete a student");
                btnDeleteAStudent.addActionListener(new ActionListener() {
                    public void actionPerformed(ActionEvent e) {
```

```java

                        DeleteStudent del = new DeleteStudent();
                        del.deleteStudent();
                    }
                });
                btnDeleteAStudent.setBounds(362, 358, 202, 40);
                frame.getContentPane().add(btnDeleteAStudent);

                JButton button_7 = new JButton("Show All Tables");
                button_7.addActionListener(new ActionListener() {
                    public void actionPerformed(ActionEvent e) {
                        show.show_students();
                        show.show_courses();
                        show.show_enrollments();
                        show.show_prerequisites();
                        show.show_classes();
                        show.show_logs();

                    }
                });
                button_7.setBounds(67, 358, 155, 40);
                frame.getContentPane().add(button_7);
            }
        }
```

Proj2.sql:

```sql
drop table logs;
drop table prerequisites;
drop table enrollments;
drop table classes;
drop table courses;
drop table students;

create table students (sid char(4) primary key check (sid like 'B%'),
firstname varchar2(15) not null, lastname varchar2(15) 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(4) 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(7) primary key, who varchar2(10) not null, time date not null,
table_name varchar2(20) not null, operation varchar2(6) not null, key_value varchar2(14));


insert into students values ('B001', 'Anne', 'Broder', 'junior', 3.6, 'broder@bu.edu');
insert into students values ('B002', 'Terry', 'Buttler', 'senior', 3.7, '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 ('CS', 352, 'operating systems');

insert into prerequisites values ('CS',532,'CS',432);
insert into prerequisites values ('CS',552, 'CS', 352);
insert into prerequisites values ('CS',352, 'CS', 252);
insert into prerequisites values ('CS',252, 'CS', 152);



insert into classes values  ('c0001', 'CS', 432, 1, 2020, 'Fall', 35, 34);
insert into classes values  ('c0002', 'Math', 314, 1, 2020, 'Fall', 25, 24);
insert into classes values  ('c0003', 'CS', 432, 1, 2019, 'Spring', 30, 30);
insert into classes values  ('c0004', 'CS', 240, 1, 2019, 'Fall', 40, 39);
insert into classes values  ('c0005', 'CS', 532, 1, 2019, 'Spring', 29, 28);
insert into classes values  ('c0006', 'Math', 221, 1, 2020, 'Spring', 30, 30);
insert into classes values  ('c0007', 'CS', 352, 1, 2019, 'Spring', 20, 19);
insert into enrollments values  ('B001', 'c0001', 'A');
insert into enrollments values  ('B002', 'c0002', 'B');
insert into enrollments values  ('B003', 'c0004', 'A');
insert into enrollments values  ('B004', 'c0004', 'C');
insert into enrollments values  ('B004', 'c0005', 'B');
insert into enrollments values  ('B005', 'c0006', 'B');
insert into enrollments values  ('B006', 'c0006', 'A');
insert into enrollments values  ('B001', 'c0002', 'C');
insert into enrollments values  ('B003', 'c0005', 'D');
insert into enrollments values  ('B007', 'c0007', 'A');
insert into enrollments values  ('B001', 'c0003', 'B');
insert into enrollments values  ('B001', 'c0006', 'B');
insert into enrollments values  ('B001', 'c0004', 'A');
insert into enrollments values  ('B001', 'c0005', 'B');
insert into enrollments values  ('B008', 'c0007', 'A');
insert into enrollments values  ('B008', 'c0005', 'B');

create sequence seq1
        increment by 1
        start with 1000001
        minvalue 000000
        maxvalue 9999999
        nocycle
        cache 20;



        create or replace trigger student_insertion_trigger
        after insert on students
        for each row
        declare
        user_name varchar2(15);
        begin
        select user into user_name from dual;
        insert into logs values(seq1.nextval,user_name,sysdate,'students','insert',:new.sid);
        end;
        /
        show errors

        create or replace trigger delete_a_student_trigger
        before delete on students
        for each row
        declare
        sid_in char(4);
        user_name varchar2(15);
        begin
        sid_in := :old.sid;
        delete from enrollments where sid = sid_in;
        select user into user_name from dual;
        insert into logs values(seq1.nextval,user_name,sysdate,'students','delete',sid_in);
        end;
        /
        show errors

        create or replace trigger enroll_a_student
        after insert on enrollments
        for each row
        declare
        user_name varchar2(15);
```

```
classid_in Classes.classid%type;
begin
classid_in := :new.classid;
update Classes set class_size = class_size +1 where classid = classid_in;
select user into user_name from dual;
insert into logs values(seq1.nextval,user_name,sysdate,'enrollments','insert',classid_in);
end;
/
show errors

create or replace trigger remove_from_classes_trigger
before delete on enrollments
for each row
declare
user_name varchar2(15);
classid_in Classes.classid%type;
begin
classid_in := :old.classid;
update Classes set class_size = class_size -1 where classid = classid_in;
select user into user_name from dual;
insert into logs values(seq1.nextval,user_name,sysdate,'enrollments','insert',classid_in);
end;
/
show errors





create or replace package project2 as

procedure
show_students(list out sys_refcursor);

procedure
show_courses(list out sys_refcursor);

procedure
show_prerequisites(list out sys_refcursor);

procedure
show_classes(list out sys_refcursor);

procedure
show_enrollments(list out sys_refcursor);

procedure
show_logs(list out sys_refcursor);

procedure
insert_values(sid_in in students.sid%type,
firstname_in in students.firstname%type,
lastname_in in students.lastname%type,
status_in in students.status%type,
gpa_in in students.gpa%type,
email_in in students.email%type,message out varchar2);

procedure
```

```
get_classes(sid_in students.sid%type,message out varchar2,list out sys_refcursor);

procedure
get_prerequisites(dept_code_in in prerequisites.dept_code%type,course_no_in in prerequisites.course_no%type,result out varchar2,checkvar in number,prereq out number);

procedure
query_6(class_id_in in Classes.classid%type,message out varchar2,list out sys_refcursor);

procedure
query_7(sid_in in Students.sid%type,class_id_in in Classes.classid%type,message out varchar2,truthvalue out number);

procedure
final_enroll(sid_in in students.sid%type,
class_id_in in Classes.classid%type);

procedure
check_grade(sid_in in students.sid%type,
dept_code_in in Classes.dept_code%type,
course_no_in in Classes.course_no%type,message out varchar2,truthvalue out number);

procedure
query_8(sid_in in Students.sid%type,class_id_in in Classes.classid%type,message out varchar2,truthvalue out number,list out sys_refcursor,dept_code1 out varchar2,course_no1 out varchar2);

procedure
drop_student(sid_in in students.sid%type,class_id_in in Classes.classid%type,message out varchar2);

procedure
delete_a_student(sid_in in students.sid%type,message out varchar2);


end project2;
/
show errors

create or replace package body project2 as

procedure
show_students(list out sys_refcursor)
is
begin
    open list for
    select * from students;

end show_students;

procedure
show_courses(list out sys_refcursor)
is
begin
    open list for
    select * from courses;

```

```
        end show_courses;

        procedure
        show_prerequisites(list out sys_refcursor)
        is
        begin
            open list for
            select * from prerequisites;

        end show_prerequisites;

        procedure
        show_classes(list out sys_refcursor)
        is
        begin
            open list for
            select * from classes;

        end show_classes;

        procedure
        show_enrollments(list out sys_refcursor)
        is
        begin
            open list for
            select * from enrollments;

        end show_enrollments;

        procedure
        show_logs(list out sys_refcursor)
        is
        begin
            open list for
            select * from logs;

        end show_logs;


        procedure
        insert_values(sid_in in students.sid%type,
        firstname_in in students.firstname%type,
        lastname_in in students.lastname%type,
        status_in in students.status%type,
        gpa_in in students.gpa%type,
        email_in in students.email%type,message out varchar2) is
        counter number;
        begin
            select count(sid) into counter from students where sid = sid_in;
            if(counter>0) then
                message := 'Error  Sid exists';
            elsif( status_in not in ('freshman', 'sophomore', 'junior', 'senior', 'graduate')) then
                message := 'Error . status wrongly defined.';
            else
                insert into students values (sid_in, firstname_in, lastname_in, status_in, gpa_in,email_in);
                message := 'Successful';
            end if;
        end insert_values;

```

```

procedure
get_classes(sid_in students.sid%type,message out varchar2,list out sys_refcursor)
is
counter number;
begin
    select count(students.sid) into counter from students where students.sid = sid_in;
    if (counter > 0) then
        select count(enrollments.sid) into counter from enrollments where enrollments.sid = sid_in;
        if (counter > 0) then
            open list for
            select s.sid,s.firstname,s.lastname,s.status,e.classid,concat (c.dept_code,c.course_no) as Course_id, c1.title from Students s,Enrollments e, Classes c, Courses c1 where s.sid = e.sid and e.classid = c.classid and c.dept_code = c1.dept_code and c.course_no = c1.course_no and e.sid = sid_in;
        else
            message := 'Student has not enrolled in any classes';
        end if;
    else
    message:= 'Invalid sid for student';
    end if;
end get_classes;

procedure
get_prerequisites(dept_code_in in prerequisites.dept_code%type,course_no_in in prerequisites.course_no%type,result out varchar2,checkvar in number,prereq out number)
is
CURSOR crefcur is
    select pre_dept_code,pre_course_no from prerequisites where dept_code = dept_code_in and course_no = course_no_in;
    course_row crefcur%rowtype;
    counter number;

    begin
        select count(*) into counter from Courses where dept_code = dept_code_in and course_no = course_no_in;
        open crefcur;
        fetch crefcur into course_row;
        if(crefcur%found) then
            while(crefcur%found) loop
                get_prerequisites(course_row.pre_dept_code, course_row.pre_course_no,result,0,prereq);
                if(result is NULL) then
                    result := course_row.pre_dept_code || ' ' || course_row.pre_course_no;
                    prereq := 1;
                else
                    result := result || ',' || course_row.pre_dept_code || ' ' ||  course_row.pre_course_no;
                    prereq := prereq+1;
                end if;
                fetch crefcur into course_row;
            end loop;
        elsif (counter = 1 and checkvar = 1) then
            result := 'No prerequisite';
            prereq := 0;
        elsif (counter < 1 and checkvar = 1) then
            result := 'This course does not exists';
```

```
      prereq := 0;
    end if;
end get_prerequisites;

procedure
query_6(class_id_in in Classes.classid%type,message out varchar2,list out sys_refcursor)
is
counter number;
begin
  select count(classes.classid) into counter from Classes where Classes.classid = class_id_in;
  if (counter >0 ) then
    select count(s.sid) into counter from Students s,Enrollments e, Classes c, Courses c1 where s.sid = e.sid and e.classid = c.classid and c.course_no = c1.course_no and c.dept_code = c1.dept_code and c.classid = class_id_in;
    if (counter > 0) then
      open list for
      select s.sid,s.firstname,s.lastname,c.classid,c1.title,c.semester,c.year from Students s,Enrollments e, Classes c, Courses c1 where s.sid = e.sid and e.classid = c.classid and c.course_no = c1.course_no and c.dept_code = c1.dept_code and c.classid = class_id_in;
    else
      message := 'No students found';
    end if;
  else
  message := 'Invalid class id';
  end if;
end query_6;


procedure
query_7(sid_in in Students.sid%type,class_id_in in Classes.classid%type,message out varchar2,truthvalue out number)
is
counter number;
limits number;
sizes number;
begin
  truthvalue := 0;
  select count(students.sid) into counter from students where students.sid = sid_in;
  if (counter > 0) then
    select count(classes.classid) into counter from Classes where classes.classid = class_id_in;
    if (counter > 0) then
      select limit into limits from Classes where classid = class_id_in;
      select class_size into sizes from Classes where classid = class_id_in;
      if (limits - sizes > 0) then
        select count(e.sid) into counter from Enrollments e where e.classid = class_id_in and e.sid = sid_in;
        if (counter > 0) then
          message := 'Student is already enrolled in this class.';
        else
          select count(*) into counter from (select count(e.classid) from Enrollments e,Classes c where c.classid = e.classid and e.sid = sid_in having count(*) > 1 group by (e.sid,c.year,c.semester));
          if (counter > 0) then
            message := 'You are overloaded';
          else
```

```
                                message := 'All conditions satisifed.Proceed for prerequisite check';
                                truthvalue := 1;
                            end if;
                        end if;


                    else
                        message := 'The class is full';
                    end if;

                else
                    message := 'class id is invalid';
                end if;

            else
                message := 'Sid not found';
            end if;
        end query_7;

        procedure
        final_enroll(sid_in in students.sid%type,
        class_id_in in Classes.classid%type) is
        counter number;
        begin
            insert into enrollments values(sid_in, class_id_in, null);
            commit;
        end;

        procedure
        check_grade(sid_in in students.sid%type,
        dept_code_in in Classes.dept_code%type,
        course_no_in in Classes.course_no%type,message out varchar2,truthvalue out number) is
        counter number;
        begin
            truthvalue := 0;
            select count(e.lgrade) into counter from Classes c,Enrollments e where e.classid = c.classid and e.sid =sid_in and c.dept_code =dept_code_in and c.course_no=course_no_in;
            if(counter >0 ) then

                select count(e.lgrade) into counter from Classes c,Enrollments e where e.classid = c.classid and e.sid =sid_in and c.dept_code =dept_code_in and c.course_no=course_no_in and e.lgrade not in ('A','A-','B+','B','B-','C+','C','C-');
                if(counter > 0) then
                    message := 'Student has failed in prerequisites.';
                    truthvalue := 0;
                else
                    message := 'Case passed';
                    truthvalue :=1;
                end if;
            else
                message:= 'Prerequisite courses have not been completed';
                truthvalue := 0;
            end if;
        end check_grade;

        procedure
```

```
query_8(sid_in in Students.sid%type,class_id_in in Classes.classid%type,message out varchar2,truthvalue out number,list out sys_refcursor,dept_code1 out varchar2,course_no1 out varchar2)
is
counter number;
begin
    truthvalue := 0;
    select count(sid) into counter from students where sid = sid_in;
    if(counter > 0) then
        select count(classid) into counter from Classes where classid = class_id_in;
        if (counter >0) then
            select count(*) into counter from enrollments where sid = sid_in and classid = class_id_in;
            if(counter >0 ) then
                truthvalue := 1;
                select c.dept_code into dept_code1 from Students s,Enrollments e, Classes c where s.sid = e.sid and e.classid = c.classid and e.sid = sid_in and e.classid = class_id_in;
                select c.course_no into course_no1 from Students s,Enrollments e, Classes c where s.sid = e.sid and e.classid = c.classid and e.sid = sid_in and e.classid = class_id_in;
                open list for
                select c.dept_code,c.course_no from Students s,Enrollments e, Classes c where s.sid = e.sid and e.classid = c.classid and e.sid = sid_in minus (select c.dept_code,c.course_no from Students s,Enrollments e, Classes c where s.sid = e.sid and e.classid = c.classid and e.sid = sid_in and e.classid = class_id_in);
            else
                message := 'Student not enrolled in class';
            end if;
        else
            message := 'Class id not found';
        end if;
    else
        message := 'sid not found';
    end if;
end query_8;

procedure
drop_student(sid_in in students.sid%type,class_id_in in Classes.classid%type,message out varchar2)
is
begin
    delete from enrollments where sid= sid_in and classid=class_id_in;
    commit;
    message := 'Sucessfully dropped';
end drop_student;


procedure
delete_a_student(sid_in in students.sid%type,message out varchar2)
is
counter number;
begin
    select count(*) into counter from students where sid = sid_in;
    if (counter = 0) then
        message := 'sid not found';
    else
        delete from students where students.sid = sid_in;
        commit;
        message := 'Delete successful';
    end if;


end delete_a_student;

end project2;
/
show errors
```