

CS532 : Database Systems

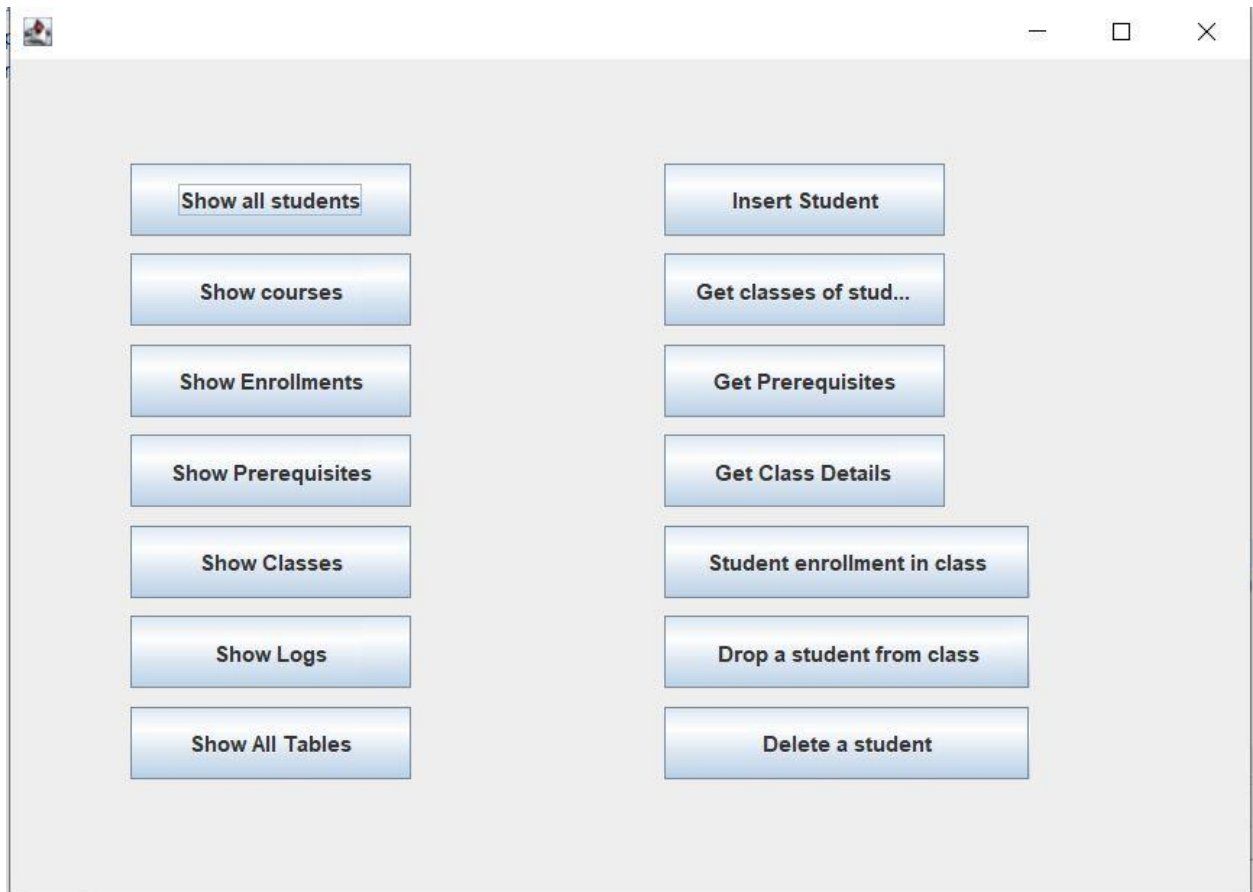
Project 2.

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:

```
2. create sequence seq1
3. increment by 1
4. start with 1000001
5. minvalue 000000
6. maxvalue 9999999
7. nocycle
8. 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:

```
1. procedure
2. show_students(list out sys_refcursor)
3. is
4. begin
5.     open list for
6.     select * from students;
7.
8. end;
```

Show Courses:

```
1. procedure
2. show_courses(list out sys_refcursor)
3. is
4. begin
5.     open list for
6.     select * from courses;
7.
8. end;
```

Show Prerequisites:

```
1. procedure
2. show_prerequisites(list out sys_refcursor)
3. is
4. begin
5.     open list for
6.     select * from prerequisites;
7.
8. 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:

```
4. procedure
5. insert_values(sid_in in students.sid%type,
6. firstname_in in students.firstname%type,
7. lastname_in in students.lastname%type,
8. status_in in students.status%type,
```

```

9. gpa_in in students.gpa%type,
10. email_in in students.email%type,message out varchar2) is
11. counter number;
12. begin
13.     select count(sid) into counter from students where sid = sid_in;
14.     if(counter>0) then
15.         message := 'Error Sid exists';
16.     elsif( status_in not in ('freshman', 'sophomore', 'junior', 'senior', 'graduate
17.         ')) then
18.         message := 'Error . status wrongly defined.';
19.     else
20.         insert into students values (sid_in, firstname_in, lastname_in, status_in,
21.         gpa_in,email_in);
22.         message := 'Successful';
23.     end if;
24. 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:

```

1. procedure
2. get_classes(sid_in students.sid%type,message out varchar2,list out sys_refcursor)
3. is
4. counter number;
5. begin
6.     select count(students.sid) into counter from students where students.sid = sid_in;
7.     if (counter > 0) then
8.         select count(enrollments.sid) into counter from enrollments where enrollments.s
9.         id = sid_in;
10.        if (counter > 0) then
11.            open list for
12.            select s.sid,s.firstname,s.lastname,s.status,e.classid,concat (c.dept_code,
13.            c.course_no) as Course_id, c1.title from Students s,Enrollments e, Classes c, Courses c
14.            1 where s.sid = e.sid and e.classid = c.classid and c.dept_code = c1.dept_code and c.co
15.            urse_no = c1.course_no and e.sid = sid_in;
16.        else
17.            message := 'Student has not enrolled in any classes';
18.        end if;
19.    else
20.        message:= 'Invalid sid for student';
21.    end if;
22. 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.

```
1. procedure
2. get_prerequisites(dept_code_in in prerequisites.dept_code%type, course_no_in in prerequi
   sites.course_no%type, result out varchar2, checkvar in number, prereq out number)
3. is
4. CURSOR crefcur is
5.     select pre_dept_code, pre_course_no from prerequisites where dept_code = dept_code_i
   n and course_no = course_no_in;
6.     course_row crefcur%rowtype;
7.     counter number;
8.
9.     begin
10.         select count(*) into counter from Courses where dept_code = dept_code_in and co
   urse_no = course_no_in;
11.         open crefcur;
12.         fetch crefcur into course_row;
13.         if(crefcur%found) then
14.             while(crefcur%found) loop
15.                 get_prerequisites(course_row.pre_dept_code, course_row.pre_course_no, re
   sult, 0, prereq);
16.                 if(result is NULL) then
17.                     result := course_row.pre_dept_code || ' ' || course_row.pre_course_
   no;
18.                     prereq := 1;
19.                 else
20.                     result := result || ', ' || course_row.pre_dept_code || ' ' || cour
   se_row.pre_course_no;
21.                     prereq := prereq+1;
22.                 end if;
23.                 fetch crefcur into course_row;
24.             end loop;
25.         elsif (counter = 1 and checkvar = 1) then
26.             result := 'No prerequisite';
27.             prereq := 0;
28.         elsif (counter < 1 and checkvar = 1) then
29.             result := 'This course does not exists';
30.             prereq := 0;
31.         end if;
32.     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:

```
1. procedure
2. query_6(class_id_in in Classes.classid%type,message out varchar2,list out sys_refcursor
   )
3. is
4. counter number;
5. begin
6.     select count(classes.classid) into counter from Classes where Classes.classid = cla
   ss_id_in;
7.     if (counter > 0 ) then
8.         select count(s.sid) into counter from Students s,Enrollments e, Classes c, Cour
   ses 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;
9.         if (counter > 0) then
10.             open list for
11.             select s.sid,s.firstname,s.lastname,c.classid,c1.title,c.semester,c.year fr
   om 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;
12.         else
13.             message := 'No students found';
14.         end if;
15.     else
16.         message := 'Invalid class id';
17.     end if;
18. 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:

```
1. procedure
2. query_7(sid_in in Students.sid%type,class_id_in in Classes.classid%type,message out var
   char2,truthvalue out number)
3. is
4. counter number;
5. limits number;
6. sizes number;
7. begin
8.     truthvalue := 0;
9.     select count(students.sid) into counter from students where students.sid = sid_in;
10.     if (counter > 0) then
11.         select count(classes.classid) into counter from Classes where classes.classid =
   class_id_in;
```

```

12.         if (counter > 0) then
13.             select limit into limits from Classes where classid = class_id_in;
14.             select class_size into sizes from Classes where classid = class_id_in;
15.             if (limits - sizes > 0) then
16.                 select count(e.sid) into counter from Enrollments e where e.classid = c
lass_id_in and e.sid = sid_in;
17.                 if (counter > 0) then
18.                     message := 'Student is already enrolled in this class.';
19.                 else
20.                     select count(*) into counter from (select count(e.classid) from Enr
ollments e,Classes c where c.classid = e.classid and e.sid = sid_in having count(*) > 1
group by (e.sid,c.year,c.semester));
21.                     if (counter > 0) then
22.                         message := 'You are overloaded';
23.                     else
24.                         message := 'All conditions satisfied.Proceed for prerequisite c
heck';
25.                         truthvalue := 1;
26.                     end if;
27.                 end if;
28.
29.
30.             else
31.                 message := 'The class is full';
32.             end if;
33.
34.         else
35.             message := 'class id is invalid';
36.         end if;
37.
38.     else
39.         message := 'Sid not found';
40.     end if;
41. 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:

```

1. procedure
2. check_grade(sid_in in students.sid%type,
3. dept_code_in in Classes.dept_code%type,
4. course_no_in in Classes.course_no%type,message out varchar2,truthvalue out number) is
5. counter number;
6. begin
7.     truthvalue := 0;
8.     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;
9.     if(counter >0 ) then
10.
11.         select count(e.lgrade) into counter from Classes c,Enrollments e where e.classi
d = 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-');
12.         if(counter > 0) then
13.             message := 'Student has failed in prerequisites.';

```

```

14.         truthvalue := 0;
15.     else
16.         message := 'Case passed';
17.         truthvalue := 1;
18.     end if;
19. else
20.     message:= 'Prerequisite courses have not been completed';
21.     truthvalue := 0;
22. end if;
23. end check_grade;

```

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

```

1. procedure
2. final_enroll(sid_in in students.sid%type,
3. class_id_in in Classes.classid%type) is
4. counter number;
5. begin
6.     insert into enrollments values(sid_in, class_id_in, null);
7.     commit;
8. end;

```

8 Drop a student from a class:

```

1. procedure
2. query_8(sid_in in Students.sid%type,class_id_in in Classes.classid%type,message out var
3. char2,truthvalue out number,list out sys_refcursor,dept_code1 out varchar2,course_no1 o
4. ut varchar2)
5. is
6. counter number;
7. begin
8.     truthvalue := 0;
9.     select count(sid) into counter from students where sid = sid_in;
10.    if(counter > 0) then
11.        select count(classid) into counter from Classes where classid = class_id_in;
12.        if (counter >0) then
13.            select count(*) into counter from enrollments where sid = sid_in and classi
14.            d = class_id_in;
15.            if(counter >0 ) then
16.                truthvalue := 1;
17.                select c.dept_code into dept_code1 from Students s,Enrollments e, Class
18.                es c where s.sid = e.sid and e.classid = c.classid and e.sid = sid_in and e.classid = c
19.                lass_id_in;
20.                select c.course_no into course_no1 from Students s,Enrollments e, Class
21.                es c where s.sid = e.sid and e.classid = c.classid and e.sid = sid_in and e.classid = c
22.                lass_id_in;
23.                open list for
24.                select c.dept_code,c.course_no from Students s,Enrollments e, Classes c
25.                where s.sid = e.sid and e.classid = c.classid and e.sid = sid_in minus (select c.dept_
26.                code,c.course_no from Students s,Enrollments e, Classes c where s.sid = e.sid and e.cla
27.                ssid = c.classid and e.sid = sid_in and e.classid = class_id_in);
28.            else
29.                message := 'Student not enrolled in class';
30.            end if;
31.        else

```

```

22.         message := 'Class id not found';
23.     end if;
24. else
25.         message := 'sid not found';
26.     end if;
27. end query_8;

```

This is also a complex procedure.

Here too, initially all the conditions are checked such as 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.

```

1. procedure
2. drop_student(sid_in in students.sid%type,class_id_in in Classes.classid%type,message out
   t varchar2)
3. is
4. begin
5.     delete from enrollments where sid= sid_in and classid=class_id_in;
6.     commit;
7.     message := 'Sucessfully dropped';
8. end drop_student;

```

9. Delete a student from students' table:

```

1. procedure
2. delete_a_student(sid_in in students.sid%type,message out varchar2)
3. is
4. counter number;
5. begin
6.     select count(*) into counter from students where sid = sid_in;
7.     if (counter = 0) then
8.         message := 'sid not found';
9.     else
10.        delete from students where students.sid = sid_in;
11.        commit;
12.        message := 'Delete successful';
13.    end if;
14.
15.
16. 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.

```
1. create or replace trigger student_insertion_trigger
2. after insert on students
3. for each row
4. declare
5. user_name varchar2(15);
6. begin
7. select user into user_name from dual;
8. insert into logs values(seq1.nextval,user_name,sysdate,'students','insert',:new.sid);
9. end;
10. /
11. show errors
```

```
1. create or replace trigger delete_a_student_trigger
2. before delete on students
3. for each row
4. declare
5. sid_in char(4);
6. user_name varchar2(15);
7. begin
8. sid_in := :old.sid;
9. delete from enrollments where sid = sid_in;
10. select user into user_name from dual;
11. insert into logs values(seq1.nextval,user_name,sysdate,'students','delete',sid_in);
12. end;
13. /
14. show errors
```

```
1. create or replace trigger enroll_a_student
2. after insert on enrollments
3. for each row
4. declare
5. user_name varchar2(15);
6. classid_in Classes.classid%type;
7. begin
8. classid_in := :new.classid;
9. update Classes set class_size = class_size +1 where classid = classid_in;
10. select user into user_name from dual;
11. insert into logs values(seq1.nextval,user_name,sysdate,'enrollments','insert',classid_in);
12. end;
13. /
14. show errors
```

```

1. create or replace trigger remove_from_classes_trigger
2. before delete on enrollments
3. for each row
4. declare
5. user_name varchar2(15);
6. classid_in Classes.classid%type;
7. begin
8. classid_in := :old.classid;
9. update Classes set class_size = class_size -1 where classid = classid_in;
10. select user_name into user_name from dual;
11. insert into logs values(seq1.nextval,user_name,sysdate,'enrollments','insert',classid_i
    n);
12. end;
13. /
14. show errors

```

Java Code :

Driver :

```

1. package project2;
2. import java.sql.Connection;
3. import java.sql.SQLException;
4.
5. import oracle.jdbc.pool.OracleDataSource;
6. public class Driver {
7.
8.     static Connection conn;
9.     public static void start_connection(){
10.
11.         try {
12.             OracleDataSource ds = new oracle.jdbc.pool.OracleDataSource();
13.             ds.setURL("jdbc:oracle:thin:@castor.cc.binghamton.edu:1521:ACAD111");
14.             conn = ds.getConnection("spatwa1", "98985432");
15.         }
16.
17.         catch (SQLException ex) { System.out.println ("\n*** SQLException caught **
    *\n");}
18.         catch (Exception e) {System.out.println ("\n*** other Exception caught ***\
    n");}
19.
20.     }
21.
22.
23.     public static void end_connection() {
24.         try {
25.
26.             conn.close();
27.         }

```

```

28.         catch (SQLException ex) { System.out.println ("\n*** SQLException caught ***\n"
);}
29.         catch (Exception e) {System.out.println ("\n*** other Exception caught ***\n");
}
30.
31.     }
32.
33.
34.
35.
36. }

```

ClassDetails.java :

```

1. package project2;
2.
3. import java.io.BufferedReader;
4. import java.io.InputStreamReader;
5. import java.sql.CallableStatement;
6. import java.sql.ResultSet;
7. import java.sql.SQLException;
8. import java.sql.Types;
9.
10. import oracle.jdbc.OracleTypes;
11.
12. public class ClassDetails {
13.
14.     public void getClassDetails() {
15.
16.         try {
17.
18.             BufferedReader readKeyBoard;
19.             String classid;
20.             readKeyBoard = new BufferedReader(new InputStreamReader(System.in));
21.             System.out.print("Please Enter class_id:");
22.             classid = readKeyBoard.readLine();
23.             CallableStatement cs = Driver.conn.prepareCall("begin project2.query_6(
:1,:2,:3); end;");
24.             cs.setString(1, classid);
25.
26.             cs.registerOutParameter(2,Types.VARCHAR);
27.             cs.registerOutParameter(3, OracleTypes.CURSOR);
28.
29.             cs.executeQuery();
30.             String message = cs.getString(2);
31.             if(message != null){
32.                 System.out.println(message);
33.             }
34.
35.
36.             ResultSet rs = (ResultSet)cs.getObject(3);
37.
38.             // print the results
39.             while (rs.next()) {
40.                 System.out.println(rs.getString(1) + "\t" +
41.                     rs.getString(2) + "\t" + rs.getString(3) + "\t" +
42.                     rs.getString(4) +
43.                     "\t" +
44.                     rs.getString(5)+ "\t" +
45.                     rs.getString(6) + "\t" +

```

```

46.         rs.getString(7));
47.     }
48.
49.     //close the result set, statement, and the connection
50.     cs.close();
51.
52. }
53.     catch (SQLException ex) { System.out.println ("\n*** SQLException caught ***\n"
+ ex.getMessage());}
54.     catch (Exception e) {System.out.println ("\n*** other Exception caught ***\n
");}
55.
56. }
57.
58.
59. }

```

DeleteStudent.java:

```

1. package project2;
2.
3. import java.io.BufferedReader;
4. import java.io.InputStreamReader;
5. import java.sql.CallableStatement;
6. import java.sql.Types;
7.
8. public class DeleteStudent {
9.
10.
11.     public void deleteStudent() {
12.
13.         try {
14.
15.             BufferedReader readKeyBoard;
16.             String sid;
17.             readKeyBoard = new BufferedReader(new InputStreamReader(System.in));
18.             System.out.print("Please Enter sid:");
19.             sid = readKeyBoard.readLine();
20.
21.             CallableStatement cs = Driver.conn.prepareCall("begin project2.delete_a_stu
dent(:1,:2); end;");
22.
23.             cs.setString(1, sid);
24.             cs.registerOutParameter(2, Types.VARCHAR);
25.             cs.executeQuery();
26.             System.out.println(cs.getString(2));
27.             cs.close();
28.         }
29.
30.         catch (Exception e) {System.out.println ("\n*** other Exception caught ***\n");
}
31.
32.     }
33.
34. }

```

Drop.java:

```

1. package project2;

```

```

2.
3. import java.io.BufferedReader;
4. import java.io.InputStreamReader;
5. import java.sql.CallableStatement;
6. import java.sql.ResultSet;
7. import java.sql.Types;
8.
9. import oracle.jdbc.OracleTypes;
10.
11. public class Drop {
12.
13.     public void dropAStudent() {
14.
15.         try {
16.             BufferedReader readKeyBoard;
17.             String sid, classid;
18.             readKeyBoard = new BufferedReader(new InputStreamReader(System.in));
19.             System.out.print("Please Enter sid:");
20.             sid = readKeyBoard.readLine();
21.             readKeyBoard = new BufferedReader(new InputStreamReader(System.in));
22.             System.out.print("Please Enter classid:");
23.             classid = readKeyBoard.readLine();
24.
25.
26.             CallableStatement cs = Driver.conn.prepareCall("begin project2.query_8(:1,:
27. 2,:3,:4,:5,:6,:7); end;");
28.
29.             cs.setString(1, sid);
30.             cs.setString(2, classid);
31.             cs.registerOutParameter(3, Types.VARCHAR);
32.             cs.registerOutParameter(4, OracleTypes.NUMBER);
33.             cs.registerOutParameter(5, OracleTypes.CURSOR);
34.             cs.registerOutParameter(6, Types.VARCHAR);
35.             cs.registerOutParameter(7, Types.VARCHAR);
36.             cs.executeQuery();
37.             String message = cs.getString(3);
38.             String truthvalue = cs.getString(4);
39.             if(Integer.valueOf(truthvalue) == 0) {
40.                 System.out.println(message);
41.                 return ;
42.             }
43.             else {
44.
45.                 String dept_code = cs.getString(6);
46.                 String course_no = cs.getString(7);
47.                 ResultSet rs = (ResultSet)cs.getObject(5);
48.
49.                 while (rs.next()) {
50.
51.                     GetPrerequisites getPreq = new GetPrerequisites();
52.                     String prereq = getPreq.getPrerequisites(rs.getString(1), rs.getStr
53. ing(2));
54.                     int i = Integer.valueOf(getPreq.getNewvar());
55.                     if(i!=0) {
56.                         String[] parts = prereq.split(",");
57.                         for(String x : parts) {
58.                             String[] part = x.split(" ");
59.                             if(part[0].equals(dept_code) && part[1].equals(course_no))
60. {
61.                                 System.out.println("Cannot drop course. It is a prerequ
62 isite of an already enrolled course");
63.                             }
64.                         }
65.                     }
66.                 }
67.             }
68.         }
69.         catch (Exception e) {
70.             System.out.println(e.getMessage());
71.         }
72.     }
73. }

```

```

59.             return;
60.         }
61.         else {
62.
63.         }
64.     }
65. }
66.
67. }
68.     cs = Driver.conn.prepareStatement("begin project2.drop_student(:1,:2,:3); en
d;");
69.
70.     cs.setString(1, sid);
71.     cs.setString(2, classid);
72.     cs.registerOutParameter(3,Types.VARCHAR);
73.     cs.executeQuery();
74.     System.out.println(cs.getString(3));
75.
76.
77.     cs.close();
78.
79.
80. }
81.
82.
83.
84.
85.
86.
87. }
88.
89.
90.     catch (Exception e) {System.out.println ("\n*** other Exception caught ***\n");
}
91.
92.
93. }
94.
95.
96. }

```

Enroll.java :

```

1. package project2;
2.
3. import java.io.BufferedReader;
4. import java.io.InputStreamReader;
5. import java.sql.CallableStatement;
6. import java.sql.ResultSet;
7. import java.sql.SQLException;
8. import java.sql.Statement;
9. import java.sql.Types;
10.
11. import oracle.jdbc.OracleTypes;
12.
13. public class Enroll {
14.
15.     public void enrollASStudent() {
16.
17.         try {

```

```

18.
19.         BufferedReader  readKeyBoard;
20.         String          sid,classid;
21.         readKeyBoard = new BufferedReader(new InputStreamReader(System.in));
22.         System.out.print("Please Enter sid:");
23.         sid = readKeyBoard.readLine();
24.         readKeyBoard = new BufferedReader(new InputStreamReader(System.in));
25.         System.out.print("Please Enter classid:");
26.         classid = readKeyBoard.readLine();
27.         CallableStatement cs = Driver.conn.prepareCall("begin project2.query_7(:1,:
2, :3, :4); end;");
28.         cs.setString(1,sid);
29.         cs.setString(2, classid);
30.         cs.registerOutParameter(3,Types.VARCHAR);
31.         cs.registerOutParameter(4, OracleTypes.NUMBER);
32.         cs.executeQuery();
33.         String message = cs.getString(3);
34.         String value = cs.getString(4);
35.         if( Integer.valueOf(value) == 0) {
36.             System.out.println(message);
37.             return ;
38.         }
39.         else {
40.
41.             Statement stmt = Driver.conn.createStatement ();
42.
43.             // Save result
44.             ResultSet rset;
45.             rset = stmt.executeQuery ("SELECT dept_code,course_no FROM Classes wher
e classid='"+classid+"'");
46.             rset.next ();
47.             String dept_code = rset.getString(1);
48.             String course_no = rset.getString(2);
49.
50.
51.             GetPrerequisites getPre = new GetPrerequisites();
52.             String prereq = getPre.getPrerequisites(dept_code, course_no);
53.             int i = Integer.valueOf(getPre.getNewvar());
54.             if(i != 0) {
55.                 String[] parts = prereq.split(",");
56.                 for(String x: parts) {
57.                     String[] part = x.split(" ");
58.                     cs = Driver.conn.prepareCall("begin project2.check_grade(:1
, :2, :3, :4, :5); end;");
59.                     cs.setString(1,sid);
60.                     cs.setString(2, part[0]);
61.                     cs.setString(3,part[1]);
62.                     cs.registerOutParameter(4,Types.VARCHAR);
63.                     cs.registerOutParameter(5, OracleTypes.NUMBER);
64.                     cs.executeQuery();
65.                     String message1 = cs.getString(4);
66.                     String value1 = cs.getString(5);
67.                     if(Integer.valueOf(value1) == 0) {
68.                         System.out.println(message1);
69.                         return;
70.                     }
71.                     else {
72.                         continue;
73.                     }
74.
75.

```

```

76.
77.
78.
79.         }
80.     }
81.     cs = Driver.conn.prepareCall("begin project2.final_enroll(:1,:2); e
nd;");
82.     cs.setString(1,sid);
83.     cs.setString(2, classid);
84.     cs.executeQuery();
85.     System.out.println("Enrollment Successful");
86.
87.
88.
89.     }
90.
91.
92.
93.
94.     }
95.     catch (SQLException ex) { System.out.println ("\n*** SQLException caught ***\n"
+ ex.getMessage());}
96.     catch (Exception e) {System.out.println ("\n*** other Exception caught ***\n");
}
97. }
98.
99. }

```

GetClasses.java:

```

1. package project2;
2.
3. import java.io.BufferedReader;
4. import java.io.InputStreamReader;
5. import java.sql.CallableStatement;
6. import java.sql.ResultSet;
7. import java.sql.SQLException;
8. import java.sql.Types;
9.
10. import oracle.jdbc.OracleTypes;
11.
12. public class GetClasses {
13.
14.
15.
16.     public void getClasses() {
17.         try {
18.
19.             BufferedReader readKeyBoard;
20.             String sid;
21.             readKeyBoard = new BufferedReader(new InputStreamReader(System.in));
22.             System.out.print("Please Enter SID:");
23.             sid = readKeyBoard.readLine();
24.             CallableStatement cs = Driver.conn.prepareCall("begin project2.get_classes(
:1,:2,:3); end;");
25.             cs.setString(1, sid);
26.             cs.registerOutParameter(2,Types.VARCHAR);
27.             cs.registerOutParameter(3, OracleTypes.CURSOR);
28.             cs.executeQuery();
29.             String message = cs.getString(2);

```



```

30.         if(message != null){
31.             System.out.println(message);
32.         }
33.         ResultSet rs = (ResultSet)cs.getObject(3);
34.
35.         while (rs.next()) {
36.             System.out.println(rs.getString(1) + "\t" +
37.                 rs.getString(2) + "\t" + rs.getString(3) + "\t" +
38.                 rs.getString(4) +
39.                 "\t" +
40.                 rs.getString(5) + "\t" +
41.                 rs.getString(6) + "\t" +
42.                 rs.getString(7));
43.         }
44.
45.         cs.close();}
46.         catch (SQLException ex) { System.out.println ("\n*** SQLException caught ***\n"
+ ex.getMessage());}
47.         catch (Exception e) {System.out.println ("\n*** other Exception caught ***\n");
}
48.     }
49.
50.
51. }

```

GetPrerequisites:

```

1. package project2;
2.
3. import java.io.BufferedReader;
4. import java.io.InputStreamReader;
5. import java.sql.CallableStatement;
6. import java.sql.SQLException;
7. import java.sql.Types;
8.
9. import oracle.jdbc.OracleTypes;
10.
11. public class GetPrerequisites {
12.     String newvar;
13.     public void getPrerequisites() {
14.         try {
15.             String dept_code,course_no,checkvar;
16.
17.             BufferedReader readKeyBoard;
18.
19.             readKeyBoard = new BufferedReader(new InputStreamReader(System.in));
20.             System.out.print("Please Enter dept_code:");
21.             dept_code = readKeyBoard.readLine();
22.             readKeyBoard = new BufferedReader(new InputStreamReader(System.in));
23.             System.out.print("Please Enter course no:");
24.             course_no = readKeyBoard.readLine();
25.             checkvar = "1";
26.             CallableStatement cs = Driver.conn.prepareCall("begin project2.get_prerequi
sites(:1,:2,:3,:4,:5); end;");
27.             cs.setString(1,dept_code);
28.             cs.setString(2, course_no);
29.             cs.setString(4, checkvar);
30.
31.
32.             cs.registerOutParameter(3,Types.VARCHAR);

```

```

33.         cs.registerOutParameter(5, OracleTypes.NUMBER);
34.
35.         cs.executeQuery();
36.         String message = cs.getString(3);
37.
38.         if(message != null){
39.             System.out.println(message);
40.         }
41.         cs.close();
42.     }
43.     catch (SQLException ex) { System.out.println ("\n*** SQLException caught ***\n" + e
x.getMessage());}
44.     catch (Exception e) {System.out.println ("\n*** other Exception caught ***\n");}
45.
46.
47.
48.     }
49.
50.     public String getPrerequisites(String dept_code,String course_no) {
51.         String message = null;
52.         try {
53.
54.
55.             String checkvar = "1";
56.             CallableStatement cs = Driver.conn.prepareCall("begin project2.get_prerequi
sites(:1,:2,:3,:4,:5); end;");
57.             cs.setString(1,dept_code);
58.             cs.setString(2, course_no);
59.             cs.setString(4, checkvar);
60.
61.
62.             cs.registerOutParameter(3,Types.VARCHAR);
63.             cs.registerOutParameter(5, OracleTypes.NUMBER);
64.
65.             cs.executeQuery();
66.             message = cs.getString(3);
67.             newvar = cs.getString(5);
68.
69.             cs.close();
70.             return message;
71.         }
72.
73.         catch (SQLException ex) { System.out.println ("\n*** SQLException caught ***\n" + e
x.getMessage());}
74.         catch (Exception e) {System.out.println ("\n*** other Exception caught ***\n");}
75.
76.     }
77.     return message;
78.
79.
80. }
81.
82. public String getNewvar() {
83.     return newvar;
84. }
85.
86.
87.
88.

```

89. }

InsertIntoStudent:

```
1. package project2;
2.
3. import java.io.BufferedReader;
4. import java.io.InputStreamReader;
5. import java.sql.CallableStatement;
6. import java.sql.SQLException;
7. import java.sql.Types;
8.
9. public class InsertIntoStudent {
10.
11.     String status;
12.
13.     public void insert() {
14.
15.
16.         BufferedReader readKeyBoard;
17.         String sid,firstname,lastname, status_in, gpa,email ;
18.         readKeyBoard = new BufferedReader(new InputStreamReader(System.in));
19.         System.out.print("Please Enter SID:");
20.         try {
21.             sid = readKeyBoard.readLine();
22.             readKeyBoard = new BufferedReader(new InputStreamReader(System.in));
23.             System.out.print("Please Enter firstname:");
24.             firstname = readKeyBoard.readLine();
25.             readKeyBoard = new BufferedReader(new InputStreamReader(System.in));
26.             System.out.print("Please Enter lastname:");
27.             lastname = readKeyBoard.readLine();
28.             readKeyBoard = new BufferedReader(new InputStreamReader(System.in));
29.             System.out.print("Please Enter level of education :");
30.             status_in = readKeyBoard.readLine();
31.             readKeyBoard = new BufferedReader(new InputStreamReader(System.in));
32.             System.out.print("Please Enter GPA:");
33.             gpa = readKeyBoard.readLine();
34.             readKeyBoard = new BufferedReader(new InputStreamReader(System.in));
35.             System.out.print("Please Enter email:");
36.             email = readKeyBoard.readLine();
37.
38.             CallableStatement cs = Driver.conn.prepareCall("begin project2.insert_values(:1
, :2, :3, :4, :5, :6, :7); end;");
39.
40.             //set the in parameter (the first parameter)
41.             cs.setString(1, sid);
42.             cs.setString(2, firstname);
43.             cs.setString(3, lastname);
44.             cs.setString(4, status_in);
45.             cs.setString(5, gpa);
46.             cs.setString(6, email);
47.             cs.registerOutParameter(7, Types.VARCHAR);
48.
49.             cs.executeQuery();
50.             status = cs.getString(7);
51.             System.out.println(status);
52.             cs.close();
53.
54.
55.         }
```

```

56.         catch (SQLException ex) { System.out.println ("\n*** SQLException caught ***\n"
+ status);}
57.         catch (Exception e) {System.out.println ("\n*** other Exception caught ***\n");
}
58.
59.
60.
61.
62.
63.     }
64.
65. }

```

ShowQueries.java:

```

1. package project2;
2.
3. import java.sql.CallableStatement;
4. import java.sql.ResultSet;
5.
6. import oracle.jdbc.OracleTypes;
7.
8. public class ShowQueries {
9.
10.
11.
12.     public void show_students() {
13.         try {
14.             CallableStatement cs = Driver.conn.prepareCall("begin project2.show_student
s(:1); end;");
15.             cs.registerOutParameter(1, OracleTypes.CURSOR);
16.             cs.execute();
17.             ResultSet rs = (ResultSet)cs.getObject(1);
18.             System.out.println("SHOW STUDENTS QUERY :");
19.             while (rs.next()) {
20.                 System.out.println(rs.getString(1) + "\t" +
21.                     rs.getString(2) + "\t" + rs.getString(3) + "\t" +
22.                     rs.getString(4) + "\t" + rs.getString(5) + "\t" + rs.getString(6));
23.             }
24.             System.out.println("\n");
25.             cs.close();
26.
27.         }
28.         catch(Exception e1) {
29.             e1.printStackTrace();
30.         }
31.
32.
33.     }
34.
35.
36.     public void show_courses() {
37.         try {
38.             CallableStatement cs = Driver.conn.prepareCall("begin project2.show_courses
(:1); end;");
39.             cs.registerOutParameter(1, OracleTypes.CURSOR);
40.             cs.execute();
41.             ResultSet rs = (ResultSet)cs.getObject(1);

```

```

42.         System.out.println("SHOW COURSES QUERY :");
43.         while (rs.next()) {
44.             System.out.println(rs.getString(1) + "\t" +
45.                 rs.getString(2) + "\t" + rs.getString(3));
46.         }
47.         System.out.println("\n");
48.         cs.close();
49.     }
50.     catch(Exception e1) {
51.         e1.printStackTrace();
52.     }
53. }
54.
55. public void show_enrollments() {
56.     try {
57.         CallableStatement cs = Driver.conn.prepareCall("begin project2.show_enrollm
ents(:1); end;");
58.         cs.registerOutParameter(1, OracleTypes.CURSOR);
59.         cs.execute();
60.         ResultSet rs = (ResultSet)cs.getObject(1);
61.         System.out.println("SHOW Enrollments QUERY :");
62.         while (rs.next()) {
63.             System.out.println(rs.getString(1) + "\t" +
64.                 rs.getString(2) + "\t" + rs.getString(3));
65.         }
66.         System.out.println("\n");
67.         cs.close();
68.     }
69.     catch(Exception e1) {
70.         e1.printStackTrace();
71.     }
72.
73.
74. }
75.
76. public void show_prerequisites() {
77.     try {
78.         CallableStatement cs = Driver.conn.prepareCall("begin project2.show_prerequ
isites(:1); end;");
79.         cs.registerOutParameter(1, OracleTypes.CURSOR);
80.         cs.execute();
81.         ResultSet rs = (ResultSet)cs.getObject(1);
82.         System.out.println("SHOW Prerequisites QUERY :");
83.         while (rs.next()) {
84.             System.out.println(rs.getString(1) + "\t" +
85.                 rs.getString(2) + "\t" + rs.getString(3) + "\t" + rs.getString(4));
86.         }
87.         System.out.println("\n");
88.         cs.close();
89.     }
90.     catch(Exception e1) {
91.         e1.printStackTrace();
92.     }
93. }
94.
95. public void show_classes() {
96.     try {
97.         CallableStatement cs = Driver.conn.prepareCall("begin project2.show_classes
(:1); end;");
98.         cs.registerOutParameter(1, OracleTypes.CURSOR);
99.         cs.execute();

```

```

100.         ResultSet rs = (ResultSet)cs.getObject(1);
101.         System.out.println("SHOW Classes:");
102.         while (rs.next()) {
103.             System.out.println(rs.getString(1) + "\t" +
104.                 rs.getString(2) + "\t" + rs.getString(3) + "\t" +
105.                 rs.getString(4) +
106.                 "\t" + rs.getString(5) + "\t" +
107.                 rs.getString(6) +
108.                 "\t" + rs.getString(7) + "\t" +
109.                 rs.getString(8));
110.         }
111.         System.out.println("\n");
112.         cs.close();
113.     }
114.     catch(Exception e1) {
115.         e1.printStackTrace();
116.     }
117.
118.     }
119.
120.     public void show_logs() {
121.         try {
122.             CallableStatement cs = Driver.conn.prepareCall("begin project2.show_
logs(:1); end;");
123.             cs.registerOutParameter(1, OracleTypes.CURSOR);
124.             cs.execute();
125.             ResultSet rs = (ResultSet)cs.getObject(1);
126.             System.out.println("SHOW Logs:");
127.             while (rs.next()) {
128.                 System.out.println(rs.getString(1) + "\t" +
129.                     rs.getString(2) + "\t" + rs.getString(3) + "\t" +
130.                     rs.getString(4) +
131.                     "\t" + rs.getString(5) + "\t" +
132.                     rs.getString(6));
133.             }
134.             System.out.println("\n");
135.             cs.close();
136.         }
137.         catch(Exception e1) {
138.             e1.printStackTrace();
139.         }
140.
141.
142.     }
143. }

```

Start.java:

```

1. package project2;
2. import oracle.jdbc.*;
3. import java.math.*;
4. import java.io.*;
5. import java.awt.*;
6. import oracle.jdbc.pool.OracleDataSource;
7.
8. import java.awt.EventQueue;
9.
10. import javax.swing.JFrame;
11. import javax.swing.JButton;
12. import java.awt.event.ActionListener;

```

```

13. import java.sql.CallableStatement;
14. import java.sql.ResultSet;
15. import java.sql.Statement;
16. import java.awt.event.ActionEvent;
17. import javax.swing.SwingConstants;
18.
19. public class Start {
20.
21.     private JFrame frame;
22.     ShowQueries show;
23.
24.     /**
25.      * Launch the application.
26.      */
27.     public static void main(String[] args) {
28.         EventQueue.invokeLater(new Runnable() {
29.             public void run() {
30.                 try {
31.                     Start window = new Start();
32.                     window.frame.setVisible(true);
33.                 } catch (Exception e) {
34.                     e.printStackTrace();
35.                 }
36.             }
37.         });
38.     }
39.
40.     /**
41.      * Create the application.
42.      */
43.     public Start() {
44.         initialize();
45.         Driver.start_connection();
46.         show = new ShowQueries();
47.     }
48.
49.     /**
50.      * Initialize the contents of the frame.
51.      */
52.     private void initialize() {
53.         frame = new JFrame();
54.         frame.setBounds(100, 100, 700, 500);
55.         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
56.         frame.getContentPane().setLayout(null);
57.
58.         JButton btnNewButton = new JButton("Show all students");
59.         btnNewButton.addActionListener(new ActionListener() {
60.             public void actionPerformed(ActionEvent e) {
61.                 show.show_students();
62.             }
63.         });
64.         btnNewButton.setBounds(67, 58, 155, 40);
65.         frame.getContentPane().add(btnNewButton);
66.
67.         JButton btnNewButton_1 = new JButton("Show courses");
68.         btnNewButton_1.addActionListener(new ActionListener() {
69.             public void actionPerformed(ActionEvent e) {
70.                 show.show_courses();
71.             }
72.         });
73.

```

```

74.         }
75.     });
76.     btnNewButton_1.setBounds(67, 108, 155, 40);
77.     frame.getContentPane().add(btnNewButton_1);
78.
79.     JButton button = new JButton("Show Enrollments");
80.     button.addActionListener(new ActionListener() {
81.         public void actionPerformed(ActionEvent e) {
82.
83.             show.show_enrollments();
84.         }
85.     });
86.     button.setBounds(67, 158, 155, 40);
87.     frame.getContentPane().add(button);
88.
89.     JButton button_1 = new JButton("Show Prerequisites");
90.     button_1.addActionListener(new ActionListener() {
91.         public void actionPerformed(ActionEvent e) {
92.             show.show_prerequisites();
93.         }
94.     });
95.     button_1.setBounds(67, 208, 155, 40);
96.     frame.getContentPane().add(button_1);
97.
98.     JButton button_2 = new JButton("Show Classes");
99.     button_2.addActionListener(new ActionListener() {
100.         public void actionPerformed(ActionEvent e) {
101.             show.show_classes();
102.
103.
104.
105.
106.         }
107.     });
108.     button_2.setBounds(67, 258, 155, 40);
109.     frame.getContentPane().add(button_2);
110.
111.     JButton button_3 = new JButton("Show Logs");
112.     button_3.addActionListener(new ActionListener() {
113.         public void actionPerformed(ActionEvent e) {
114.             show.show_logs();
115.
116.         }
117.     });
118.     button_3.setBounds(67, 308, 155, 40);
119.     frame.getContentPane().add(button_3);
120.
121.     JButton button_4 = new JButton("Insert Student");
122.     button_4.addActionListener(new ActionListener() {
123.         public void actionPerformed(ActionEvent e) {
124.             InsertIntoStudent insert = new InsertIntoStudent();
125.             insert.insert();
126.
127.
128.
129.         }
130.     });
131.     button_4.setBounds(362, 58, 155, 40);
132.     frame.getContentPane().add(button_4);
133.
134.     JButton btnGetClassesOf = new JButton("Get classes of student ");

```



```

135.         btnGetClassesOf.setHorizontalAlignment(SwingConstants.LEFT);
136.         btnGetClassesOf.addActionListener(new ActionListener() {
137.             public void actionPerformed(ActionEvent e) {
138.                 GetClasses getClass = new GetClasses();
139.                 getClass.getClasses();
140.
141.
142.
143.             }
144.         });
145.         btnGetClassesOf.setBounds(362, 108, 155, 40);
146.         frame.getContentPane().add(btnGetClassesOf);
147.
148.         JButton btnGetPrere = new JButton("Get Prerequisites");
149.         btnGetPrere.addActionListener(new ActionListener() {
150.             public void actionPerformed(ActionEvent e) {
151.
152.                 GetPrerequisites getPrereq = new GetPrerequisites();
153.                 getPrereq.getPrerequisites();
154.
155.             }
156.         });
157.         btnGetPrere.setBounds(362, 158, 155, 40);
158.         frame.getContentPane().add(btnGetPrere);
159.
160.         JButton button_5 = new JButton("Get Class Details");
161.         button_5.addActionListener(new ActionListener() {
162.             public void actionPerformed(ActionEvent e) {
163.                 ClassDetails getClassDetails = new ClassDetails();
164.                 getClassDetails.getClassDetails();
165.
166.             }
167.         });
168.         button_5.setBounds(362, 208, 155, 40);
169.         frame.getContentPane().add(button_5);
170.
171.         JButton btnEnrollAStudent = new JButton("Student enrollment in class");
172.         btnEnrollAStudent.addActionListener(new ActionListener() {
173.             public void actionPerformed(ActionEvent e) {
174.                 Enroll enrollStudent = new Enroll();
175.                 enrollStudent.enrollAStudent();
176.
177.             }
178.         });
179.         btnEnrollAStudent.setBounds(362, 258, 202, 40);
180.         frame.getContentPane().add(btnEnrollAStudent);
181.
182.         JButton button_6 = new JButton("Drop a student from class");
183.         button_6.addActionListener(new ActionListener() {
184.             public void actionPerformed(ActionEvent e) {
185.                 Drop dropper = new Drop();
186.                 dropper.dropAStudent();
187.             }
188.         });
189.         button_6.setBounds(362, 308, 202, 40);
190.         frame.getContentPane().add(button_6);
191.
192.         JButton btnDeleteAStudent = new JButton("Delete a student");
193.         btnDeleteAStudent.addActionListener(new ActionListener() {
194.             public void actionPerformed(ActionEvent e) {

```

```

195.
196.             DeleteStudent del = new DeleteStudent();
197.             del.deleteStudent();
198.         }
199.     });
200.     btnDeleteAStudent.setBounds(362, 358, 202, 40);
201.     frame.getContentPane().add(btnDeleteAStudent);
202.
203.     JButton button_7 = new JButton("Show All Tables");
204.     button_7.addActionListener(new ActionListener() {
205.         public void actionPerformed(ActionEvent e) {
206.             show.show_students();
207.             show.show_courses();
208.             show.show_enrollments();
209.             show.show_prerequisites();
210.             show.show_classes();
211.             show.show_logs();
212.
213.         }
214.     });
215.     button_7.setBounds(67, 358, 155, 40);
216.     frame.getContentPane().add(button_7);
217. }
218. }

```

Proj2.sql:

```

1. drop table logs;
2. drop table prerequisites;
3. drop table enrollments;
4. drop table classes;
5. drop table courses;
6. drop table students;
7.
8. create table students (sid char(4) primary key check (sid like 'B%'),
9.  firstname varchar2(15) not null, lastname varchar2(15) not null, status varchar2(10)
10. check (status in ('freshman', 'sophomore', 'junior', 'senior', 'graduate')),
11.  gpa number(3,2) check (gpa between 0 and 4.0), email varchar2(20) unique);
12.
13. create table courses (dept_code varchar2(4) not null, course_no number(3) not null
14. check (course_no between 100 and 799), title varchar2(20) not null,
15.  primary key (dept_code, course_no));
16.
17. create table prerequisites (dept_code varchar2(4) not null,
18.  course_no number(3) not null, pre_dept_code varchar2(4) not null,
19.  pre_course_no number(3) not null,
20.  primary key (dept_code, course_no, pre_dept_code, pre_course_no),
21.  foreign key (dept_code, course_no) references courses on delete cascade,
22.  foreign key (pre_dept_code, pre_course_no) references courses
23.  on delete cascade);
24.
25. create table classes (classid char(5) primary key check (classid like 'c%'),
26.  dept_code varchar2(4) not null, course_no number(3) not null,
27.  sect_no number(2), year number(4), semester varchar2(6)
28.  check (semester in ('Spring', 'Fall', 'Summer')), limit number(3),
29.  class_size number(3), foreign key (dept_code, course_no) references courses
30.  on delete cascade, unique(dept_code, course_no, sect_no, year, semester),

```

```

31. check (class_size <= limit));
32.
33. create table enrollments (sid char(4) references students, classid char(5) references c
    lasses,
34. lgrade char check (lgrade in ('A', 'B', 'C', 'D', 'F', 'I', null)), primary key (sid, c
    lassid));
35.
36. create table logs (logid number(7) primary key, who varchar2(10) not null, time date no
    t null,
37. table_name varchar2(20) not null, operation varchar2(6) not null, key_value varchar2(14
    ));
38.
39.
40. insert into students values ('B001', 'Anne', 'Broder', 'junior', 3.6, 'broder@bu.edu');
41. insert into students values ('B002', 'Terry', 'Buttler', 'senior', 3.7, 'buttler@bu.edu
    ');
42. insert into students values ('B003', 'Tracy', 'Wang', 'senior', 4.0, 'wang@bu.edu');
43. insert into students values ('B004', 'Barbara', 'Callan', 'junior', 2.5, 'callan@bu.edu
    ');
44. insert into students values ('B005', 'Jack', 'Smith', 'graduate', 3.0, 'smith@bu.edu');
45. insert into students values ('B006', 'Terry', 'Zillman', 'graduate', 4.0, 'zillman@bu.e
    du');
46. insert into students values ('B007', 'Becky', 'Lee', 'senior', 4.0, 'lee@bu.edu');
47. insert into students values ('B008', 'Tom', 'Baker', 'freshman', null, 'baker@bu.edu');
48.
49.
50.
51.
52.
53.
54.
55.
56. insert into courses values ('CS', 432, 'database systems');
57. insert into courses values ('Math', 314, 'discrete math');
58. insert into courses values ('CS', 240, 'data structure');
59. insert into courses values ('Math', 221, 'calculus I');
60. insert into courses values ('CS', 532, 'database systems');
61. insert into courses values ('CS', 552, 'operating systems');
62. insert into courses values ('CS', 352, 'operating systems');
63.
64. insert into prerequisites values ('CS', 532, 'CS', 432);
65. insert into prerequisites values ('CS', 552, 'CS', 352);
66. insert into prerequisites values ('CS', 352, 'CS', 252);
67. insert into prerequisites values ('CS', 252, 'CS', 152);
68.
69.
70.
71. insert into classes values ('c0001', 'CS', 432, 1, 2020, 'Fall', 35, 34);
72. insert into classes values ('c0002', 'Math', 314, 1, 2020, 'Fall', 25, 24);
73. insert into classes values ('c0003', 'CS', 432, 1, 2019, 'Spring', 30, 30);
74. insert into classes values ('c0004', 'CS', 240, 1, 2019, 'Fall', 40, 39);
75. insert into classes values ('c0005', 'CS', 532, 1, 2019, 'Spring', 29, 28);
76. insert into classes values ('c0006', 'Math', 221, 1, 2020, 'Spring', 30, 30);
77. insert into classes values ('c0007', 'CS', 352, 1, 2019, 'Spring', 20, 19);
78.
79.
80.
81.

```

```

82. insert into enrollments values ('B001', 'c0001', 'A');
83. insert into enrollments values ('B002', 'c0002', 'B');
84. insert into enrollments values ('B003', 'c0004', 'A');
85. insert into enrollments values ('B004', 'c0004', 'C');
86. insert into enrollments values ('B004', 'c0005', 'B');
87. insert into enrollments values ('B005', 'c0006', 'B');
88. insert into enrollments values ('B006', 'c0006', 'A');
89. insert into enrollments values ('B001', 'c0002', 'C');
90. insert into enrollments values ('B003', 'c0005', 'D');
91. insert into enrollments values ('B007', 'c0007', 'A');
92. insert into enrollments values ('B001', 'c0003', 'B');
93. insert into enrollments values ('B001', 'c0006', 'B');
94. insert into enrollments values ('B001', 'c0004', 'A');
95. insert into enrollments values ('B001', 'c0005', 'B');
96. insert into enrollments values ('B008', 'c0007', 'A');
97. insert into enrollments values ('B008', 'c0005', 'B');
98.
99. create sequence seq1
100.     increment by 1
101.     start with 1000001
102.     minvalue 000000
103.     maxvalue 9999999
104.     nocycle
105.     cache 20;
106.
107.
108.
109.     create or replace trigger student_insertion_trigger
110.     after insert on students
111.     for each row
112.     declare
113.         user_name varchar2(15);
114.     begin
115.         select user into user_name from dual;
116.         insert into logs values(seq1.nextval,user_name,sysdate,'students','insert',:new.
sid);
117.     end;
118.     /
119.     show errors
120.
121.     create or replace trigger delete_a_student_trigger
122.     before delete on students
123.     for each row
124.     declare
125.         sid_in char(4);
126.         user_name varchar2(15);
127.     begin
128.         sid_in := :old.sid;
129.         delete from enrollments where sid = sid_in;
130.         select user into user_name from dual;
131.         insert into logs values(seq1.nextval,user_name,sysdate,'students','delete',sid_i
n);
132.     end;
133.     /
134.     show errors
135.
136.     create or replace trigger enroll_a_student
137.     after insert on enrollments
138.     for each row
139.     declare
140.         user_name varchar2(15);

```

```

141.      classid_in Classes.classid%type;
142.      begin
143.      classid_in := :new.classid;
144.      update Classes set class_size = class_size +1 where classid = classid_in;
145.      select user into user_name from dual;
146.      insert into logs values(seq1.nextval,user_name,sysdate,'enrollments','insert',cl
      assid_in);
147.      end;
148.      /
149.      show errors
150.
151.      create or replace trigger remove_from_classes_trigger
152.      before delete on enrollments
153.      for each row
154.      declare
155.      user_name varchar2(15);
156.      classid_in Classes.classid%type;
157.      begin
158.      classid_in := :old.classid;
159.      update Classes set class_size = class_size -1 where classid = classid_in;
160.      select user into user_name from dual;
161.      insert into logs values(seq1.nextval,user_name,sysdate,'enrollments','insert',cl
      assid_in);
162.      end;
163.      /
164.      show errors
165.
166.
167.
168.
169.
170.      create or replace package project2 as
171.
172.      procedure
173.      show_students(list out sys_refcursor);
174.
175.      procedure
176.      show_courses(list out sys_refcursor);
177.
178.      procedure
179.      show_prerequisites(list out sys_refcursor);
180.
181.      procedure
182.      show_classes(list out sys_refcursor);
183.
184.      procedure
185.      show_enrollments(list out sys_refcursor);
186.
187.      procedure
188.      show_logs(list out sys_refcursor);
189.
190.      procedure
191.      insert_values(sid_in in students.sid%type,
192.      firstname_in in students.firstname%type,
193.      lastname_in in students.lastname%type,
194.      status_in in students.status%type,
195.      gpa_in in students.gpa%type,
196.      email_in in students.email%type,message out varchar2);
197.
198.      procedure

```

```

199.         get_classes(sid_in students.sid%type,message out varchar2,list out sys_refcursor
200.     );
201.     procedure
202.     get_prerequisites(dept_code_in in prerequisites.dept_code%type,course_no_in in p
rerequisites.course_no%type,result out varchar2,checkvar in number,prereq out number);

203.
204.     procedure
205.     query_6(class_id_in in Classes.classid%type,message out varchar2,list out sys_re
fcursor);
206.
207.     procedure
208.     query_7(sid_in in Students.sid%type,class_id_in in Classes.classid%type,message
out varchar2,truthvalue out number);
209.
210.     procedure
211.     final_enroll(sid_in in students.sid%type,
212.     class_id_in in Classes.classid%type);
213.
214.     procedure
215.     check_grade(sid_in in students.sid%type,
216.     dept_code_in in Classes.dept_code%type,
217.     course_no_in in Classes.course_no%type,message out varchar2,truthvalue out numbe
r);
218.
219.     procedure
220.     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,cours
e_no1 out varchar2);
221.
222.     procedure
223.     drop_student(sid_in in students.sid%type,class_id_in in Classes.classid%type,mes
sage out varchar2);
224.
225.     procedure
226.     delete_a_student(sid_in in students.sid%type,message out varchar2);
227.
228.
229.     end project2;
230.     /
231.     show errors
232.
233.     create or replace package body project2 as
234.
235.     procedure
236.     show_students(list out sys_refcursor)
237.     is
238.     begin
239.         open list for
240.         select * from students;
241.
242.     end show_students;
243.
244.     procedure
245.     show_courses(list out sys_refcursor)
246.     is
247.     begin
248.         open list for
249.         select * from courses;
250.

```

```

251.         end show_courses;
252.
253.     procedure
254.     show_prerequisites(list out sys_refcursor)
255.     is
256.     begin
257.         open list for
258.         select * from prerequisites;
259.
260.     end show_prerequisites;
261.
262.     procedure
263.     show_classes(list out sys_refcursor)
264.     is
265.     begin
266.         open list for
267.         select * from classes;
268.
269.     end show_classes;
270.
271.     procedure
272.     show_enrollments(list out sys_refcursor)
273.     is
274.     begin
275.         open list for
276.         select * from enrollments;
277.
278.     end show_enrollments;
279.
280.     procedure
281.     show_logs(list out sys_refcursor)
282.     is
283.     begin
284.         open list for
285.         select * from logs;
286.
287.     end show_logs;
288.
289.
290.     procedure
291.     insert_values(sid_in in students.sid%type,
292.     firstname_in in students.firstname%type,
293.     lastname_in in students.lastname%type,
294.     status_in in students.status%type,
295.     gpa_in in students.gpa%type,
296.     email_in in students.email%type,message out varchar2) is
297.     counter number;
298.     begin
299.         select count(sid) into counter from students where sid = sid_in;
300.         if(counter>0) then
301.             message := 'Error Sid exists';
302.         elsif( status_in not in ('freshman', 'sophomore', 'junior', 'senior', 'graduate')) then
303.             message := 'Error . status wrongly defined.';
304.         else
305.             insert into students values (sid_in, firstname_in, lastname_in, status_in, gpa_in,email_in);
306.             message := 'Successful';
307.         end if;
308.     end insert_values;
309.

```

```

310.
311.     procedure
312.     get_classes(sid_in students.sid%type,message out varchar2,list out sys_refcursor
313.     )
314.     is
315.     counter number;
316.     begin
317.         select count(students.sid) into counter from students where students.sid = s
318.         id_in;
319.         if (counter > 0) then
320.             select count(enrollments.sid) into counter from enrollments where enroll
321.             ments.sid = sid_in;
322.             if (counter > 0) then
323.                 open list for
324.                 select s.sid,s.firstname,s.lastname,s.status,e.classid,concat (c.dept
325.                 _code,c.course_no) as Course_id, c1.title from Students s,Enrollments e, Classes c, Co
326.                 urses c1 where s.sid = e.sid and e.classid = c.classid and c.dept_code = c1.dept_code a
327.                 nd c.course_no = c1.course_no and e.sid = sid_in;
328.             else
329.                 message := 'Student has not enrolled in any classes';
330.             end if;
331.         else
332.             message:= 'Invalid sid for student';
333.         end if;
334.     end get_classes;
335.
336.     procedure
337.     get_prerequisites(dept_code_in in prerequisites.dept_code%type,course_no_in in p
338.     rerequisites.course_no%type,result out varchar2,checkvar in number,prereq out number)
339.     is
340.     CURSOR crefcur is
341.         select pre_dept_code,pre_course_no from prerequisites where dept_code = dept
342.         _code_in and course_no = course_no_in;
343.         course_row crefcur%rowtype;
344.         counter number;
345.     begin
346.         select count(*) into counter from Courses where dept_code = dept_code_in
347.         and course_no = course_no_in;
348.         open crefcur;
349.         fetch crefcur into course_row;
350.         if(crefcur%found) then
351.             while(crefcur%found) loop
352.                 get_prerequisites(course_row.pre_dept_code, course_row.pre_cours
353.                 e_no,result,0,prereq);
354.                 if(result is NULL) then
355.                     result := course_row.pre_dept_code || ' ' || course_row.pre_
356.                     course_no;
357.                     prereq := 1;
358.                 else
359.                     result := result || ',' || course_row.pre_dept_code || ' ' |
360.                     | course_row.pre_course_no;
361.                     prereq := prereq+1;
362.                 end if;
363.                 fetch crefcur into course_row;
364.             end loop;
365.         elsif (counter = 1 and checkvar = 1) then
366.             result := 'No prerequisite';
367.             prereq := 0;
368.         elsif (counter < 1 and checkvar = 1) then
369.             result := 'This course does not exists';

```



```

359.         prereq := 0;
360.     end if;
361. end get_prerequisites;
362.
363. procedure
364. query_6(class_id_in in Classes.classid%type,message out varchar2,list out sys_re
fcursor)
365. is
366.     counter number;
367.     begin
368.         select count(classes.classid) into counter from Classes where Classes.classi
d = class_id_in;
369.         if (counter > 0 ) then
370.             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;
371.             if (counter > 0) then
372.                 open list for
373.                 select s.sid,s.firstname,s.lastname,c.classid,c1.title,c.semester,c.c
year from Students s,Enrollments e, Classes c, Courses c1 where s.sid = e.sid and e.cla
ssid = c.classid and c.course_no = c1.course_no and c.dept_code = c1.dept_code and c.cl
assid = class_id_in;
374.             else
375.                 message := 'No students found';
376.             end if;
377.         else
378.             message := 'Invalid class id';
379.         end if;
380.     end query_6;
381.
382.
383. procedure
384. query_7(sid_in in Students.sid%type,class_id_in in Classes.classid%type,message
out varchar2,truthvalue out number)
385. is
386.     counter number;
387.     limits number;
388.     sizes number;
389.     begin
390.         truthvalue := 0;
391.         select count(students.sid) into counter from students where students.sid = s
id_in;
392.         if (counter > 0) then
393.             select count(classes.classid) into counter from Classes where classes.cl
assid = class_id_in;
394.             if (counter > 0) then
395.                 select limit into limits from Classes where classid = class_id_in;
396.                 select class_size into sizes from Classes where classid = class_id_i
n;
397.                 if (limits - sizes > 0) then
398.                     select count(e.sid) into counter from Enrollments e where e.clas
sid = class_id_in and e.sid = sid_in;
399.                     if (counter > 0) then
400.                         message := 'Student is already enrolled in this class.';
401.                     else
402.                         select count(*) into counter from (select count(e.classid) f
rom 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));
403.                         if (counter > 0) then
404.                             message := 'You are overloaded';
405.                         else

```

```

406.                message := 'All conditions satisfied.Proceed for prerequ
isite check';
407.                truthvalue := 1;
408.            end if;
409.        end if;
410.
411.
412.        else
413.            message := 'The class is full';
414.        end if;
415.
416.    else
417.        message := 'class id is invalid';
418.    end if;
419.
420.    else
421.        message := 'Sid not found';
422.    end if;
423. end query_7;
424.
425. procedure
426. final_enroll(sid_in in students.sid%type,
427. class_id_in in Classes.classid%type) is
428. counter number;
429. begin
430.     insert into enrollments values(sid_in, class_id_in, null);
431.     commit;
432. end;
433.
434. procedure
435. check_grade(sid_in in students.sid%type,
436. dept_code_in in Classes.dept_code%type,
437. course_no_in in Classes.course_no%type,message out varchar2,truthvalue out numbe
r) is
438. counter number;
439. begin
440.     truthvalue := 0;
441.     select count(e.lgrade) into counter from Classes c,Enrollments e where e.cla
ssid = c.classid and e.sid =sid_in and c.dept_code =dept_code_in and c.course_no=course
_no_in;
442.     if(counter >0 ) then
443.
444.         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=co
urse_no_in and e.lgrade not in ('A','A-','B+','B','B-','C+','C','C-');
445.         if(counter > 0) then
446.             message := 'Student has failed in prerequisites.';
447.             truthvalue := 0;
448.         else
449.             message := 'Case passed';
450.             truthvalue :=1;
451.         end if;
452.     else
453.         message:= 'Prerequisite courses have not been completed';
454.         truthvalue := 0;
455.     end if;
456. end check_grade;
457.
458. procedure

```

```

459.      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,cours
e_no1 out varchar2)
460.      is
461.      counter number;
462.      begin
463.          truthvalue := 0;
464.          select count(sid) into counter from students where sid = sid_in;
465.          if(counter > 0) then
466.              select count(classid) into counter from Classes where classid = class_id
_in;
467.              if (counter >0) then
468.                  select count(*) into counter from enrollments where sid = sid_in and
classid = class_id_in;
469.                  if(counter >0 ) then
470.                      truthvalue := 1;
471.                      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.clas
sid = class_id_in;
472.                      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.clas
sid = class_id_in;
473.                      open list for
474.                      select c.dept_code,c.course_no from Students s,Enrollments e, Cl
asses 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 an
d e.classid = c.classid and e.sid = sid_in and e.classid = class_id_in);
475.                      else
476.                          message := 'Student not enrolled in class';
477.                      end if;
478.                  else
479.                      message := 'Class id not found';
480.                  end if;
481.              else
482.                  message := 'sid not found';
483.              end if;
484.          end query_8;
485.
486.      procedure
487.      drop_student(sid_in in students.sid%type,class_id_in in Classes.classid%type,mes
sage out varchar2)
488.      is
489.      begin
490.          delete from enrollments where sid= sid_in and classid=class_id_in;
491.          commit;
492.          message := 'Sucessfully dropped';
493.      end drop_student;
494.
495.
496.      procedure
497.      delete_a_student(sid_in in students.sid%type,message out varchar2)
498.      is
499.      counter number;
500.      begin
501.          select count(*) into counter from students where sid = sid_in;
502.          if (counter = 0) then
503.              message := 'sid not found';
504.          else
505.              delete from students where students.sid = sid_in;
506.              commit;
507.              message := 'Delete successful';

```

```
508.         end if;  
509.  
510.  
511.     end delete_a_student;  
512.  
513. end project2;  
514. /  
515. show errors
```