Mark Han

Professor Kang

CS 432

13 April 2022

Project 3 Design Document

PL/SQL Procedures, Functions and Triggers:

Procedure/Function/Trigger name	Description
show_students	Function that returns cursor containing Students table
show_courses	Function that returns cursor containing Courses table
show_prerequisite	Function that returns cursor containing Prerequisites table
show_classes	Function that returns cursor containing Classes table
show_enrollments	Function that returns cursor containing Enrollments table
show_logs	Function that returns cursor containing logs Table
insert_student(v_SID in Students.SID%type, v_firstname in Students.firstname%type, v_lastname in Students.lastname%type, v_status in Students.status%type, v_GPA in Students.GPA%type, v_email in Students.email%type)	Procedure that inserts a new student from inputted parameters
student_info(v_SID in Students.SID%type, msg out varchar2, ref_cursor out sys_refcursor)	Procedure that either displays all classes of an inputted student or an error message
get_prereqs(v_dept_code in Prerequisites.dept_code%type, v_course_no in Prerequisites.course_no%type, ref_cursor out sys_refcursor)	Procedure that gets all direct and indirect prerequisites of an inputted course
get_class_students(v_classid in enrollments.classid%type, msg out varchar2, ref_cursor out sys_refcursor)	Function that displays all students who are taking or have taken an inputted class and returns 0. If the inputted class is invalid or a class is empty, returns 1

enroll_student(v_sid in Students.sid%type, v_classid in Enrollments.classid%type, msg out varchar2)	Procedure that enrolls a student of the inputted sid to the Class of the inputted classid. Displays an error message if the sid is invalid, the classid is invalid, the class is full, the student is taking 5 or more classes, or the prerequisites have not been met.
unenroll_student(v_sid in Students.sid%type, v_classid in Enrollments.classid%type, msg out varchar2)	Procedure that unenrolls a student of the inputted sid from the class of the inputted classid. Displays an error message if the sid is invalid, the class is invalid, the student is not enrolled, or the student is enrolled in only 1 class.
delete_student (v_sid in Students.sid%type, msg out varchar2)	Function that deletes a student from the Students table and returns 1. Returns 0 if the sid is not found.
enroll_student_log	Trigger that triggers when a student is enrolled into enrollments. Inserts a new log into the logs table and updates the class size of the enrolled class.

Java Methods:

showTable()	Executes functions to display tables based inputted selection
insertStudent()	Executes procedure insert_student
studentInfo()	Executes procedure student_info
getPrereqs()	Executes procedure get_prereqs
getClassStudents()	Executes function get_class_students
enrollStudents()	Executes function enroll_student
unenrollStudents()	Executes function unenroll_student
deleteStudent()	Executes function delete_student

PL/SQL CODE:

```
set serveroutput on


Drop trigger enroll_student_log;

Drop table prereq_helper_table;

Create table prereq_helper_table(dept_code varchar2(4) not null, course_no number(3) not
null);


drop sequence log_seq;

create sequence log_seq

increment by 1

start with 1000;


-- SPECIFICATIONS

create or replace package registrations as

type ref_cursor is ref cursor;


-- Q2: Show Tables

function show_students

  return ref_cursor;


function show_courses

  return ref_cursor;


function show_prerequisites

  return ref_cursor;
```

```
function show_classes
  return ref_cursor;

function show_enrollments
  return ref_cursor;

function show_logs
  return ref_cursor;

-- Q3: Insert Student
procedure insert_student(v_SID in Students.SID%type, v_firstname in Students.firstname%type,
v_lastname in Students.lastname%type,
                    v_status in Students.status%type, v_GPA in Students.GPA%type, v_email in
Students.email%type);

-- Q4: Student Info
procedure student_info(v_SID in Students.SID%type, msg out varchar2, ref_cursor out
sys_refcursor);

-- Q5: Prerequisites
procedure get_prereqs(v_dept_code in Prerequisites.dept_code%type, v_course_no in
Prerequisites.course_no%type, ref_cursor out sys_refcursor);

-- Q6: Get Class Students
function get_class_students(v_classid in enrollments.classid%type, msg out varchar2,
ref_cursor out sys_refcursor)
return number;

-- Q7: Enroll Students
```

```
procedure enroll_student(v_sid in Students.sid%type, v_classid in Enrollments.classid%type,
msg out varchar2);

-- Q8: Unenroll Students

procedure unenroll_student(v_sid in Students.sid%type, v_classid in Enrollments.classid%type,
msg out varchar2);

-- Q9: Delete Students

function delete_student(v_sid in Students.sid%type, msg out varchar2)

return number;

end;

/

show errors

-- BODY

create or replace package body registrations as

-- Q2: Show Tables

function show_students

return ref_cursor as

rc ref_cursor;

begin

  open rc for

  select * from students;

  return rc;

end;
```

```
function show_courses
return ref_cursor as
rc ref_cursor;
begin
  open rc for
  select * from courses;
  return rc;
end;

function show_prerequisites
return ref_cursor as
rc ref_cursor;
begin
  open rc for
  select * from prerequisites;
  return rc;
end;

function show_classes
return ref_cursor as
rc ref_cursor;
begin
  open rc for
  select * from classes;
  return rc;
end;
```

```
function show_enrollments
return ref_cursor as
rc ref_cursor;
begin
  open rc for
  select * from enrollments;
  return rc;
end;

function show_logs
return ref_cursor as
rc ref_cursor;
begin
  open rc for
  select * from logs;
  return rc;
end;

-- Q3: Insert Student
procedure insert_student(v_SID in Students.SID%type, v_firstname in Students.firstname%type,
v_lastname in Students.lastname%type,
                    v_status in Students.status%type, v_GPA in Students.GPA%type, v_email in
Students.email%type) as
begin
  insert into students(SID, firstname, lastname, status, GPA, email) values (v_SID, v_firstname,
v_lastname, v_status, v_GPA, v_email);
end;
```

```
-- Q4: Student Info

procedure student_info(v_SID in Students.SID%type, msg out varchar2, ref_cursor out
sys_refcursor) is

  v_SID_count Number;

  v_class_count Number;

begin

  select count(*) into v_SID_count from Students where v_SID = Students.SID;

  select count(*) into v_class_count from Students, Enrollments where Students.SID =
Enrollments.SID AND v_SID = Enrollments.SID;

  if v_SID_count = 0 then

     msg := 'invalid sid';

  else

    if v_class_count = 0 then

      msg := 'has not taken any course';

    else open ref_cursor for

      select students.sid, students.lastname, students.status, classes.classid,
concat(classes.dept_code, classes.course_no) as course_id from Students

      join Enrollments on students.sid = enrollments.sid

      join Classes on Enrollments.classid = Classes.classid

      where students.sid = v_sid;

    end if;

  end if;

end;


-- Q5: Prerequisites

/*

-make a prereq cursor that contains all the prereqs of the course we selected

            ex. CS and 442 are inputed and its prereqs are CS 240 and Math 314
```

the cursor now contains

CS 240

Math 314

-save these values from the cursor to a table prereq_helper_table(dept_code, course_no) for output

-make a record cursor to store each line for recursion

-start a loop

-fetch first line of prereq_cursor into prereq_rec

-exit loop if nothing was fetched

-recursively call the function on the prereqs to get its prereqs

-store values into the table

-store table values onto ref_cursor

*/

```
procedure get_prereqs(v_dept_code in Prerequisites.dept_code%type, v_course_no in Prerequisites.course_no%type, ref_cursor out sys_refcursor) is

cursor prereq_cursor is

select pre_dept_code, pre_course_no from prerequisites

where dept_code = v_dept_code and course_no = v_course_no;

prereq_rec prereq_cursor%rowtype;



begin

 insert into prereq_helper_table select pre_dept_code, pre_course_no

 from prerequisites where v_dept_code = dept_code and v_course_no = course_no;

 open prereq_cursor;

 loop

  fetch prereq_cursor into prereq_rec;

  exit when prereq_cursor%notfound;
```

```
    get_prereqs(prereq_rec.pre_dept_code, prereq_rec.pre_course_no, ref_cursor);
  end loop;
  open ref_cursor for select * from prereq_helper_table;
  close prereq_cursor;
end;



-- Q6: Function that takes classid and prints classid, course title, sid, lastname, and email of all students taking/taken that class
function get_class_students(v_classid in enrollments.classid%type, msg out varchar2,
ref_cursor out sys_refcursor)
return number is
v_classid_count Number;
v_student_count Number;
begin
  select count(*) into v_classid_count from Classes where v_classid = Classes.classid;
  select count(*) into v_student_count from Enrollments where v_classid = Enrollments.classid;
  if v_classid_count = 0 then
    msg := 'invalid cid';
    return 1;
  elsif v_student_count = 0 then
    msg := 'empty class';
    return 1;
  else
    open ref_cursor for
    select enrollments.classid, courses.title, students.sid, students.lastname, students.email from enrollments
      join Classes on Enrollments.classid = Classes.classid
```

```
    join Courses on Classes.course_no = Courses.course_no
    join Students on students.sid = enrollments.sid
    where enrollments.classid = v_classid;
  end if;
  return 0;
end;


-- Q7: Procedure to enroll student into a class. Takes in student.sid and enrollment.classid.
procedure enroll_student(v_sid in Students.sid%type, v_classid in Enrollments.classid%type,
msg out varchar2) is
  v_sid_count Number;
  v_classid_count Number;
  v_open_seats Number;
  v_already_in Number;
  v_class_count Number;
  v_prereq_count Number;

begin
  begin
  -- Use counts to validate sid/cid
  select count(*) into v_sid_count from Students where v_sid = sid;

  select count(*) into v_classid_count from Classes where classid = v_classid;

  -- Limit - class_size = Number of available seats
  select limit-class_size into v_open_seats from Classes where v_classid = classid;
```

```
 select count(*) into v_already_in from Enrollments where v_sid = sid and v_classid = classid;


 -- Keep a count of the enrolled classes in the current semester and year of inputted SID
 select count(*) into v_class_count from enrollments, classes
 where enrollments.sid = v_sid and enrollments.classid = classes.classid and classes.semester =
'Spring' and classes.year = 2022 ;


 -- Select (prereqs of inputted SID and CID) minus (all classes taken by inputted SID with lgrade
of at least C+)
  -- If count > 0, then prerequisite requirements are not met
 select count(*) into v_prereq_count from
  (select prerequisites.pre_dept_code, prerequisites.pre_course_no from Prerequisites
        join Classes on Prerequisites.dept_code = Classes.dept_code and
Prerequisites.course_no = Classes.course_no
        join Enrollments on Enrollments.classid = Classes.classid
        where Enrollments.sid = v_sid and Enrollments.classid = v_classid
 minus
  select classes.dept_code, classes.course_no from Classes
        join Enrollments on Enrollments.classid = Classes.classid
        where Enrollments.sid = v_sid and Enrollments.lgrade < 'C') count;
 exception
  when no_data_found then
   msg := 'invalid classid';
 end;


 if v_sid_count = 0 then
```

```
    msg := 'invalid sid';
  elsif v_classid_count = 0 then
    msg := 'invalid classid';
  elsif v_open_seats = 0 then
    msg := 'class full';
  elsif v_already_in > 0 then
    msg := 'already in this class';
  elsif v_class_count > 4 then
    msg := 'overloaded!';
  elsif v_prereq_count > 0 then
    msg := 'prerequisite courses have not been completed';
  else
    insert into Enrollments(sid, classid) values (v_sid, v_classid);
  end if;
end;



-- Q8: Procedure to unenroll student from class
procedure unenroll_student(v_sid in Students.sid%type, v_classid in Enrollments.classid%type,
msg out varchar2) is
  v_sid_count Number;
  v_classid_count Number;
  v_already_in Number;
  v_class_count Number;
  v_class_size Number;

begin
```

```
  select count(*) into v_sid_count from Students where v_sid = sid;
  select count(*) into v_classid_count from Classes where classid = v_classid;
  select count(*) into v_already_in from Enrollments where v_sid = sid and v_classid = classid;
  select count(*) into v_class_count from Enrollments where v_sid = sid;
  select count(*) into v_class_size from Classes where classid = v_classid;

  if v_sid_count = 0 then
    msg := 'invalid sid';
  elsif v_classid_count = 0 then
    msg := 'invalid classid';
  elsif v_already_in = 0 then
    msg := 'student not enrolled';
  elsif v_class_count = 1 then
    msg := 'drop request rejected; must be enrolled in at least one class.';
  elsif v_class_size = 1 then
    msg := 'no student in this class';
    delete from enrollments where v_sid = sid and v_classid = classid;
  else
    delete from enrollments where v_sid = sid and v_classid = classid;
  end if;
end;

-- Q9: Procedure to delete student
function delete_student(v_sid in Students.sid%type, msg out varchar2)
  return number is
  v_sid_count Number;
```

```
begin
  select count(*) into v_sid_count from Students where v_sid = sid;
  if v_sid_count = 0 then
    msg := 'sid not found';
    return 1;
  end if;
  delete from students where v_sid = sid;
  return 0;
end;

end;
/
show errors

-- Triggers
create or replace trigger enroll_student_log
after insert on enrollments
for each row
declare
  v_logid Number;
  v_who varchar2(10);
  v_table_name varchar2(20);
  v_operation_name varchar2(6);
  v_sid enrollments.sid%type;
  v_classid enrollments.classid%type;
  v_key_value varchar2(14);
begin
```

```
  v_logid := log_seq.nextVal;
  select user into v_who from dual;
  v_table_name := 'Enrollments';
  v_operation_name := 'Insert';
  v_sid := :new.sid;
  v_classid := :new.classid;
  v_key_value := (v_sid || ', ' || v_classid);
  insert into logs values(v_logid, v_who, sysdate, v_table_name, v_operation_name,
v_key_value);
  update classes
  set class_size = class_size + 1
  where classid = v_classid;
end;
/
/*
create or replace trigger delete_student
after delete on students
begin
  delete from enrollments where sid = students.sid
end;
/
*/
show errors;
```

JAVA CODE:

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

public class Driver {
	public static void main(String args[]) throws SQLException {
		try {
			OracleDataSource ds = new oracle.jdbc.pool.OracleDataSource();

	ds.setURL("jdbc:oracle:thin:@castor.cc.binghamton.edu:1521:ACAD111");
			Connection conn = ds.getConnection("mhan6", "dmonking");

			while (true) {
				System.out.println("\n----------MENU----------");
				System.out.println("1: View Table");
				System.out.println("2: Insert Student");
				System.out.println("3: Student's Classes");
				System.out.println("4: Class Prequisites");
				System.out.println("5: Show Class Students");
				System.out.println("6: Enroll Students");
				System.out.println("7: Unenroll Students");
				System.out.println("8: Delete Students");
				System.out.println("9: Exit");
```

```
                    BufferedReader readKeyBoard;
                    int i = 0;
                    readKeyBoard = new BufferedReader(new
InputStreamReader(System.in));
                    System.out.print("Select an option: ");
                    i = Integer.parseInt(readKeyBoard.readLine());

                    switch (i) {

                        // Q2 - Show Six Tables
                        case 1: {
                            System.out.println("\n-----SELECT A TABLE-----");
                            System.out.println("1: Students");
                            System.out.println("2: Courses");
                            System.out.println("3: Prerequisites");
                            System.out.println("4: Classes");
                            System.out.println("5: Enrollments");
                            System.out.println("6: Logs");

                            BufferedReader input;
                            int j = 0;
                            input = new BufferedReader(new
InputStreamReader(System.in));
                            System.out.print("Select an option: ");
                            j = Integer.parseInt(input.readLine());
                            System.out.println();
                            showTable(j, conn);
```

```
                    break;
                }

                // Q3 - Insert Student
                case 2: {
                    BufferedReader input;
                    input = new BufferedReader(new
InputStreamReader(System.in));
                    System.out.println("\nStudent SID: ");
                    String sid = input.readLine();
                    System.out.println("\nStudent First Name: ");
                    String firstname = input.readLine();
                    System.out.println("\nStudent Last Name: ");
                    String lastname = input.readLine();
                    System.out.println("\nStudent Status: ");
                    String status = input.readLine();
                    System.out.println("\nStudent GPA: ");
                    Double gpa =
Double.parseDouble(input.readLine());
                    System.out.println("\nStudent Email: ");
                    String email = input.readLine();
                    insertStudent(conn, sid, firstname, lastname,
status, gpa, email);
                    break;
                }

                /* Q4:
                 *  Lists sid, lastname, status of a student
```

```
                                        *  + Lists classid, dept_code, and course_no (concatenate
dept_code and course_no)
                                        *
                                        *  Exceptions:
                                        *         If a student is not in the table, print 'invalid sid'
                                        *   If a student is in the table but has not taken any
courses, print 'has not taken any course'
                                        */
                                       case 3: {
                                                BufferedReader input;
                                                input = new BufferedReader(new
InputStreamReader(System.in));
                                                System.out.print("\nStudent SID: ");
                                                String sid = input.readLine();
                                                studentInfo(conn, sid);
                                                break;
                                       }

                                       /* Q5:
                                        *  Given dept_code and course_no as inputs, show all
prerequisites (including indirect)
                                        *  For each prerequisite show concatenated dept_code
and course_no
                                        * */
                                       case 4: {
                                                BufferedReader input;
                                                input = new BufferedReader(new
InputStreamReader(System.in));
                                                System.out.print("\nClass deptcode: ");
```

```
                String deptcode = input.readLine();
                System.out.print("\nClass course_no: ");
                String course_no = input.readLine();
                getPrereqs(conn, deptcode, course_no);
                break;
            }

            /* Q6: Prints all students taking a class
             *
             */
            case 5: {
                BufferedReader input;
                input = new BufferedReader(new
InputStreamReader(System.in));
                System.out.print("\nClassid: ");
                String classid = input.readLine();
                getClassStudents(conn, classid);
                break;
            }

            /* Q7: Enroll Student
             *
             * */
            case 6: {
                BufferedReader input;
                input = new BufferedReader(new
InputStreamReader(System.in));
                System.out.print("\nStudent Sid: ");
```

```
                String sid = input.readLine();
                System.out.print("\nClassid: ");
                String classid = input.readLine();
                enrollStudents(conn, sid, classid);
                break;
            }

            /* Q8: Unenroll Student
             *
             */
            case 7: {
                BufferedReader input;
                input = new BufferedReader(new
InputStreamReader(System.in));
                System.out.print("\nStudent Sid: ");
                String sid = input.readLine();
                System.out.print("\nClassid: ");
                String classid = input.readLine();
                unenrollStudents(conn, sid, classid);
                break;
            }

            /* Q9: Delete Student
             *
             */
            case 8: {
                BufferedReader input;
```

```
                                        input = new BufferedReader(new
InputStreamReader(System.in));
                                        System.out.print("\nStudent Sid: ");
                                        String sid = input.readLine();
                                        deleteStudent(conn, sid);
                                        break;
                                }

                                // Exit Program
                                case 9: {
                                        System.exit(1);
                                }
                        }

                }

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

    // Q2 - Show 6 Tables
    public static void showTable(int selection, Connection conn) {
        switch (selection) {
            // 1: Show Students
```

```
                case 1: {
                        try {
                                CallableStatement cs = conn.prepareCall("begin ? :=
registrations.show_students(); end;");
                                cs.registerOutParameter(1, OracleTypes.CURSOR);
                                cs.execute();
                                ResultSet rs = (ResultSet) cs.getObject(1);
                                while (rs.next()) {
                                        System.out.println(rs.getString(1) + "\t" +
rs.getString(2) + "\t" + rs.getString(3) + "\t"
                                                        + rs.getString(4) + "\t" +
rs.getDouble(5) + "\t" + rs.getString(6));
                                }
                                rs.close();
                                cs.close();
                        } catch (SQLException ex) {
                                System.out.println("\n*** SQLException caught ***\n" +
ex.getMessage());
                        }
                        break;
                }

                // 2: Show Courses
                case 2: {
                        try {
                                CallableStatement cs = conn.prepareCall("begin ? :=
registrations.show_courses(); end;");
                                cs.registerOutParameter(1, OracleTypes.CURSOR);
                                cs.execute();
```

```
                                        ResultSet rs = (ResultSet) cs.getObject(1);
                                        while (rs.next()) {
                                                System.out.println(rs.getString(1) + "\t" +
rs.getInt(2) + "\t" + rs.getString(3));
                                        }
                                        rs.close();
                                        cs.close();
                                } catch (SQLException ex) {
                                        System.out.println("\n*** SQLException caught ***\n" +
ex.getMessage());
                                }
                                break;
                        }

                        // 3: Show Prerequisites
                        case 3: {
                                try {
                                        CallableStatement cs = conn.prepareCall("begin ? :=
registrations.show_prerequisites(); end;");
                                        cs.registerOutParameter(1, OracleTypes.CURSOR);
                                        cs.execute();
                                        ResultSet rs = (ResultSet) cs.getObject(1);
                                        while (rs.next()) {
                                                System.out.println(
                                                        rs.getString(1) + "\t" + rs.getInt(2) +
"\t" + rs.getString(3) + "\t" + rs.getInt(4));
                                        }
                                        rs.close();
```

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

            // 4: Show Classes
            case 4: {
                try {
                    CallableStatement cs = conn.prepareCall("begin ? :=
registrations.show_classes(); end;");
                    cs.registerOutParameter(1, OracleTypes.CURSOR);
                    cs.execute();
                    ResultSet rs = (ResultSet) cs.getObject(1);
                    while (rs.next()) {
                        System.out.println(rs.getString(1) + "\t" +
rs.getString(2) + "\t" + rs.getInt(3) + "\t"
                                + rs.getInt(4) + "\t" + rs.getInt(5) +
"\t" + rs.getString(6) + "\t" + rs.getInt(7) + "\t"
                                + rs.getInt(8));
                    }
                    rs.close();
                    cs.close();
                } catch (SQLException ex) {
                    System.out.println("\n*** SQLException caught ***\n" +
ex.getMessage());
                }
```

```
                    break;
                }

                // 5: Show Enrollments
                case 5: {
                    try {
                        CallableStatement cs = conn.prepareCall("begin ? :=
registrations.show_enrollments(); end;");
                        cs.registerOutParameter(1, OracleTypes.CURSOR);
                        cs.execute();
                        ResultSet rs = (ResultSet) cs.getObject(1);
                        while (rs.next()) {
                            System.out.println(rs.getString(1) + "\t" +
rs.getString(2) + "\t" + rs.getString(3));
                        }
                        rs.close();
                        cs.close();
                    } catch (SQLException ex) {
                        System.out.println("\n*** SQLException caught ***\n" +
ex.getMessage());
                    }
                    break;
                }

                // 6: Show Logs
                case 6: {
                    try {
```

```
                                        CallableStatement cs = conn.prepareCall("begin ? :=
registrations.show_logs(); end;");
                                        cs.registerOutParameter(1, OracleTypes.CURSOR);
                                        cs.execute();
                                        ResultSet rs = (ResultSet) cs.getObject(1);
                                        while (rs.next()) {
                                                System.out.println(rs.getInt(1) + "\t" +
rs.getString(2) + "\t" + rs.getString(3) + "\t"
                                                                + rs.getString(4) + "\t" +
rs.getString(5) + "\t" + rs.getString(6));
                                        }
                                        rs.close();
                                        cs.close();
                                } catch (SQLException ex) {
                                        System.out.println("\n*** SQLException caught ***\n" +
ex.getMessage());
                                }
                                break;
                        }
                }
        }

        // Q3 - Insert a Student
        public static void insertStudent(Connection conn, String sid, String firstname, String
lastname, String status,
                        Double gpa, String email) {
                try {
                        CallableStatement cs = conn.prepareCall("begin
registrations.insert_student(?,?,?,?,?,?); end;");
```

```
                cs.setString(1, sid);
                cs.setString(2, firstname);
                cs.setString(3, lastname);
                cs.setString(4, status);
                cs.setDouble(5, gpa);
                cs.setString(6, email);
                cs.execute();
                cs.close();

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

        /* Q4 - Show Student's sid, lastname, status,
         *              classid, concatenate(dept_code, course_no)
         *     (Only prints sid if student is not taking any courses)
         */
        public static void studentInfo(Connection conn, String sid) {
            try {
                CallableStatement cs = conn.prepareCall("begin
registrations.student_info(?,?,?); end;");
                cs.setString(1, sid);
                cs.registerOutParameter(2, java.sql.Types.VARCHAR);
                cs.registerOutParameter(3, OracleTypes.CURSOR);
                cs.execute();
                String msg = cs.getString(2);
```

```
                        // If v_SID_count = 0, then 'invalid sid' is printed
                        // If v_class_count = 0, then 'has not taken any course' is printed
                        // Otherwise, Cursor results are printed
                        if (msg != null) {
                                System.out.println(msg);
                        } else {
                                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));
                                }
                                rs.close();
                                cs.close();
                        }
                } catch (SQLException ex) {
                        System.out.println("\n*** SQLException caught ***\n" +
ex.getMessage());
                }
        }

        // Q5
        public static void getPrereqs(Connection conn, String deptcode, String course_no) {
                try {
                        CallableStatement cs = conn.prepareCall("begin
registrations.get_prereqs(?,?,?); end;");
                        cs.setString(1, deptcode);
                        cs.setString(2, course_no);
```

```
                cs.registerOutParameter(3, OracleTypes.CURSOR);
                cs.execute();

                ResultSet rs = (ResultSet) cs.getObject(3);
                while (rs.next()) {
                        System.out.println(rs.getString(1) + rs.getString(2));
                }
                Statement truncate = conn.createStatement();
                truncate.executeQuery("truncate table prereq_helper_table");
                rs.close();
                cs.close();
                truncate.close();
        } catch (Exception ex) {
                ex.printStackTrace();
                System.out.println("Exception in getPrereqs()");
        }
    }

    /* Q6 - Given a classid, prints the classid, course title, sid, lastname, and email
     *          Function get_class_students(classid, msg, cursor) returns 0 if at least one student is printed
     *          Returns 1 and prints "invalid cid" if classid is not in class table
     *          Returns 1 and prints "empty class" if classid is not in enrollments table
     * */
    public static void getClassStudents(Connection conn, String classid) {
        try {
            CallableStatement cs = conn.prepareCall("begin ? := registrations.get_class_students(?, ?, ?); end;");
```

```
                        cs.registerOutParameter(1, java.sql.Types.NUMERIC);
                        cs.setString(2, classid);
                        cs.registerOutParameter(3, java.sql.Types.VARCHAR);
                        cs.registerOutParameter(4, OracleTypes.CURSOR);
                        cs.execute();
                        String msg = cs.getString(3);
                        if (msg != null) {
                                System.out.println(msg);
                        } else {
                                ResultSet rs = (ResultSet) cs.getObject(4);
                                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));
                                }
                                rs.close();
                                cs.close();
                        }

                } catch (Exception ex) {
                        ex.printStackTrace();
                        System.out.println("Exception in getClassStudents()");
                }
        }

        /* Q7: Enroll Student
         * */
```

```
	public static void enrollStudents(Connection conn, String sid, String classid) {
		try {
			CallableStatement cs = conn.prepareCall("begin
registrations.enroll_student(?,?,?); end;");
			cs.setString(1, sid);
			cs.setString(2, classid);
			cs.registerOutParameter(3, java.sql.Types.VARCHAR);
			cs.execute();
			String msg = cs.getString(3);
			if (msg != null) {
				System.out.println(msg);
			} else {
				System.out.println("student enrolled");
			}
		} catch (Exception ex) {
			ex.printStackTrace();
			System.out.println("Exception in enrollStudents()");
		}
	}

	/* Q8: Unenroll Student
	 * */
	public static void unenrollStudents(Connection conn, String sid, String classid) {
		try {
			CallableStatement cs = conn.prepareCall("begin
registrations.unenroll_student(?,?,?); end;");
			cs.setString(1, sid);
			cs.setString(2, classid);
```

```
                cs.registerOutParameter(3, java.sql.Types.VARCHAR);
                cs.execute();
                String msg = cs.getString(3);
                if (msg != null) {
                        System.out.println(msg);
                } else {
                        System.out.println("student enrolled");
                }
        } catch (Exception ex) {
                ex.printStackTrace();
                System.out.println("Exception in unenrollStudents()");
        }
}

/* Q9: Delete Student
 * */
public static void deleteStudent(Connection conn, String sid) {
        try {
                CallableStatement cs = conn.prepareCall("begin ? :=
registrations.delete_student(?, ?); end;");
                cs.registerOutParameter(1, java.sql.Types.NUMERIC);
                cs.setString(2, sid);
                cs.registerOutParameter(3, java.sql.Types.VARCHAR);
                cs.execute();
        } catch (Exception ex) {
                ex.printStackTrace();
                System.out.println("Exception in deleteStudent()");
```

```
                }
            }
        }
```