CS532 – DATABASE SYSTEMS

Project: 2

Implement Student Registration System using PL/SQL and JDBC

SUBMITTED BY:

NAME	EMAIL ID	SIGNATURE
VARUN SAXENA	vsaxena1@binghamton.edu	
KUNDAN SHRIVASTAV	kshriva1@binghamton.edu	
SEAN GALLAGHER	sgallag3@binghamton.edu	

CONTENTS:

1. **Introduction – Project Description**
2. **Project Implementation**
3. **Project Overflow**
    a. **PL/SQL code – Triggers**
    a. **Stored Procedures**
4. **Java Methods**
5. **PL/SQL code**
    a. **Procedure to show all table information**
    b. **PL/SQL code – Procedure to show various tables details**
    c. **PL/SQL code - Procedure to show the pre-requisites courses**
    d. **PL/SQL code - Procedure to show the TAs details**
    e. **PL/SQL code – Procedure to enroll a student**
    f. **PL/SQL code – Procedure to drop a student from a class**
    g. **PL/SQL code – Procedure to delete a student**
6. **Java Code**
7. **Conclusion**

1. Introduction – Project Description:

The project uses Oracle's PL/SQL and JDBC to create an application to for student registration system at a university using an interface.

Different data of students is manipulated as per the given requirements using procedures in PL/SQL.

The student data can be added and deleted from the tables.

Triggers and sequences are created to track the changes in tables.

---

2. Project Implementation:

The sequence starts generating **log_seq** from 100 and increments by 1 when new log records are inserted into the logs table. It uses sequence to generate the output with starting point as 100 and incrementing the value by 1.

---

3. Project Overflow created in this project:

SYS_REFCURSOR is used to return from the stored procedures and functions.

PL/SQL code – Triggers

Sr. No.	TRIGGER NAME	DESCRIPTION
1.	**ENROLLMENTS_INSERT**	Insert or update the log table using triggers. The classes table is updated by incrementing the value by 1 when a new entry is entered in the enrollments table.
2.	**ENROLLMENTS_DELETE**	Update the log table by firing the trigger by decrementing the class size by 1 while deleting the student from enrollments.
3.	**STUDENT_DELETE**	Update the log table while deleting the student from the student table. If the student is enrolled in the enrollments table then the student should be deleted form enrollments table too. Similar case if the student is a TA.

Stored Procedures:

Sr. No.	Procedure	Description
1.	show_student	Display student details and records
2.	show_courses	Display courses details and records
3.	Show_TAs	Display TA Table info.
4.	show_classes	Display classes information
5.	show_enrollments	Display student enrollments information
6.	show_prerequisites	Display required prerequisites for a required course
7.	show_logs	Display the table log
8.	ta_info	Parameter passed classid: Display the B#, Fist and last name of the TA
9.	get_prerequisites	Parameters passed are Dept_code and course#. Get the direct or indirect prerequisites for a particular course.
10.	enroll_student	Parameters passed are B# and classid. Student is enrolled by adding an entry in the enrollment table.
11.	drop_student	Parameters passed are B# and classid. Entry in the enrollments table is deleted.
12.	del_student	Parameter passed is B#. Student is deleted from student table along with TA table if the student is a TA.

4. JAVA METHODS:

METHODS	DESCRIPTION
enrollStudentClass()	To Enroll a student for a class
dropStudentClass()	Drop a student from Enrollment table
deleteStudent()	Delete a student from student table
infoPrerequisites()	Display information for the prerequisites
TAInfo()	Display TA information regarding B#, First name and last name
showTableInfo()	Select the table to be shown or displayed

5. PL/SQL code:

```
/*To display the output on the console/terminal*/
set serveroutput on;

/*Drop triggers*/
Drop trigger enrollments_insert;
Drop trigger enrollments_delete;
```

```
Drop trigger student_delete;
Drop table temp_prerequisites;

Create table temp_prerequisites(dept_code varchar2(4) not null,course# number(3) not null);

--Q.1.
--Sequence
--Done by: Varun Saxena
/*Create sequence starting with 100*/
DROP SEQUENCE logs_seq;
CREATE SEQUENCE logs_seq
increment by 1
START WITH 100;

--Package Started
Create or replace package student_registration AS

/*Declaration of Procedures*/
  --Q.2.
  procedure show_students(student_cursor out sys_refcursor);
  procedure show_courses(student_cursor out sys_refcursor);
  procedure show_TAs(student_cursor out sys_refcursor);
  procedure show_classes(student_cursor out sys_refcursor);
  procedure show_enrollments(student_cursor out sys_refcursor);
  procedure show_prerequisites(student_cursor out sys_refcursor);
  procedure show_logs(student_cursor out sys_refcursor);

  --Q.3.
  procedure ta_info(v_classid in Classes.classid%type,error_message out varchar2,r_cursor out
sys_refcursor);

  --Q.4.
  procedure get_prerequisites(v_dept_code in courses.dept_code%type,v_course# in
courses.course#%type,error_message out varchar2,r_cursor out

sys_refcursor);

  --Q.5.
  procedure enroll_student(v_B# in students.B#%type,v_classid in
classes.classid%type,error_message out varchar2);

  --Q.6.
  procedure drop_student(dropB# in Students.B#%type,dropClassid in
Classes.Classid%type,error_message out varchar2);
```

```
 --Q.7.
 procedure del_student(delB# IN Students.B#%type,error_message out varchar2);

END;
/

create or replace package body student_registration AS

 --Q.2.
 --Done by: Varun Saxena
 /*students table */
 procedure show_students(student_cursor out
 sys_refcursor) AS
 BEGIN
     open student_cursor for
     select * from students;
 END;

 /* courses table */
 procedure show_courses(student_cursor out

 sys_refcursor) AS
 BEGIN
      open student_cursor for
      SELECT * FROM COURSES;
 END;

 /* TAs table */
 procedure show_TAs(student_cursor out sys_refcursor)
 AS
 BEGIN
      open student_cursor for
      select * from TAs;
 End;

 /* classes table */
 procedure show_classes(student_cursor out
 sys_refcursor) as
 BEGIN
     open student_cursor for
      select * from classes;
 END;
```

```
/* enrollments table */
  procedure show_enrollments(student_cursor out
  sys_refcursor)
  AS
  BEGIN
       open student_cursor for
       select * from enrollments;
  END;

/* prerequisites table */
  procedure show_prerequisites(student_cursor out
  sys_refcursor) AS
  BEGIN
       open student_cursor for
       select * from prerequisites;
  END;

/* logs table */
  procedure show_logs(student_cursor out sys_refcursor)
  AS
  BEGIN
      open student_cursor for
      select * from logs;
  END;

  --Q.3.
  --Done by: Varun Saxena
  /*procedure in the package to list the B#, the first name and last name of the TA of the class.
*/
  procedure ta_info (v_classid in Classes.classid%type,error_message out varchar2,r_cursor out
sys_refcursor)
  is v_data_found_classid Number;
  v_data_found_TA Number;

  Begin
       SELECT count(*) into v_data_found_classid from Classes WHERE classid =  v_classid;
       select count(*) into v_data_found_TA FROM TAs,Classes WHERE TAs.B# = Classes.TA_B#
AND Classes.classid = v_classid;
       if (v_data_found_classid = 0) THEN
           error_message := 'The classid is invalid';
       else
            if v_data_found_TA = 0 then
                  error_message := 'The class has no TA';
             else
```

```
            open r_cursor for
        select students.B#,students.first_name,students.last_name
        FROM Students
        JOIN TAs ON Students.B# = TAs.B#
        JOIN Classes ON TAs.B# = Classes.TA_B#
        WHERE Classes.classid = v_classid;
        end if;
    end if;
end;

--Q.4.
--Done by: Varun Saxena
/* procedure in the package that return all its prerequisites course */
 procedure get_prerequisites(v_dept_code in courses.dept_code%type,v_course# in
courses.course#%type,error_message out varchar2,r_cursor out
sys_refcursor) is
 v_found_dept_code_course# Number;
 cursor prereq_cursor is
 select pre_dept_code,pre_course# from prerequisites
 where dept_code = v_dept_code
 and course# = v_course#;
 v_found_prereq Number;

 prereq_record prereq_cursor%rowtype;
 BEGIN
   SELECT count(*) into v_found_dept_code_course# FROM Courses WHERE
   dept_code = v_dept_code and course# = v_course#;
   SELECT count(*) into v_found_prereq FROM Prerequisites WHERE dept_code = v_dept_code
and course# = v_course#;
   if (v_found_dept_code_course# = 0) THEN
      error_message := v_dept_code||v_course#||' does not exist';
   else
     if (v_found_prereq = 0) THEN
       error_message:= v_dept_code||v_course#||' does not exist';
     else
     insert into temp_prerequisites select pre_dept_code,pre_course# from
     prerequisites where dept_code = v_dept_code and course# =
     v_course#;
     open prereq_cursor;
     loop
      fetch prereq_cursor into prereq_record;
      exit when prereq_cursor%notfound;
```

```
get_prerequisites(prereq_record.pre_dept_code,prereq_record.pre_course#,error_message,r_c
ursor);
       end loop;
     open r_cursor for select * from temp_prerequisites;
     close prereq_cursor;
   end if;
  end if;
 end;

 --Q.5.
 --Done by: Kundan Shrivastav
/*Procedure in the package to enroll a student in the class */
 procedure enroll_student(v_B# in students.B#%type,
 v_classid in classes.classid%type,error_message out varchar2) is
 v_student_B# Number;
 v_student_classid Number;
 v_class_sem Number;
 v_student_in_sem Number;
 v_capacity Number;
 v_student_overloaded Number;
 v_count_prereqs Number;
 v_count_classid_prereqs Number;
 Begin
   Select count(*) into v_student_B# from Students where B# = v_B#;
   Select count(*) into v_student_classid from Classes where classid = v_classid;
   if (v_student_classid > 0) then
     select count(*) into v_class_sem from classes where classid = v_classid
     and year = 2018 and semester = 'Fall';
     select LIMIT-class_size into v_capacity from classes where classid =
     v_classid;
   end if;
   Select count(*) into v_student_in_sem from enrollments where B# = v_B#
   and classid = v_classid;
   Select count(*) into v_student_overloaded from enrollments e,classes c
   where e.B# = v_B# and e.classid = c.classid and c.year = 2018 and c.semester = 'Fall';
   Select count(*) into v_count_prereqs from prerequisites where
   (dept_code,course#) in (Select dept_code,course# from classes where classid = v_classid);
   Select count(classid) into v_count_classid_prereqs from enrollments where lgrade <= 'C'
   and B# = v_B# and classid in (Select classid from classes where (dept_code,course#) in
   (Select pre_dept_code,pre_course# from prerequisites where (dept_code,course#) in (Select
   dept_code,course# from classes where classid = v_classid)));
   if (v_student_B# = 0) then
     error_message := 'The B# is invalid';
```

```
  elsif (v_student_classid = 0) then
    error_message := 'The classid is invalid';
  elsif (v_class_sem = 0) then
    error_message := 'Cannot enroll into a class from a previous semester';
  elsif (v_capacity = 0) then
    error_message := 'The class is already full';
  elsif (v_student_in_sem <> 0) then
    error_message := 'The student is already in the class';
  elsif (v_count_prereqs <> v_count_classid_prereqs) then
    error_message := 'Prerequisite not satisfied';
  elsif (v_student_overloaded = 4) then
    error_message := 'The student will be overloaded with the new
    enrollment';
    INSERT INTO Enrollments(B#,classid) VALUES (v_B#,v_classid);
  elsif (v_student_overloaded > 4) then
    error_message := 'Students cannot be enrolled in more than five classes
    in the same semester';
  else
    INSERT into Enrollments(B#,classid) VALUES (v_B#,v_classid);
  end if;
end;

--Q.6.
--Done by: Sean Gallagher
/* procedure in the package to drop a student from a class */
procedure drop_student(
        dropB# in Students.B#%type,
        dropClassid in Classes.Classid%type,error_message out varchar2) IS
        --Local declarations
        count_B# Students.B#%type;
        count_Classid Classes.Classid%type;
        count_Enrollment Enrollments.B#%type;
        tempSemester Classes.Semester%type;
        tempYear Classes.Year%type;
        dCode Classes.DEPT_CODE%type;
        c# Classes.Course#%type;
        countPre Number;
        newSize Classes.Class_size%type;
        numClasses Number;
BEGIN
  SELECT count(*)
  INTO count_B# FROM Students WHERE B# = dropB#;
  SELECT count(*)
  INTO count_Classid FROM Classes WHERE Classid = dropClassid;
```

```
  SELECT count(*)
  INTO count_Enrollment FROM Enrollments WHERE B# = dropB# and Classid = dropClassid;
  IF (count_B# = 0) THEN
    error_message := 'The B# is invalid';
  ELSIF (count_Classid = 0) THEN
    error_message := 'The classid is invalid';
  ELSIF (count_Enrollment = 0) THEN
    error_message := 'The student is not enrolled in the class';
  ELSE

          SELECT SEMESTER, YEAR
          INTO tempSemester, tempYear FROM CLASSES WHERE Classid = dropClassid;
          IF tempSemester != 'Fall' or tempYear != 2018 THEN
                  error_message := 'Only enrollment in the current semester can be dropped.';
                  RETURN;
          END IF;
         SELECT DEPT_CODE, COURSE#
          INTO dCode, c# FROM CLASSES WHERE Classid = dropClassid;
           SELECT count(DEPT_CODE) INTO countPre
            FROM PREREQUISITES WHERE DEPT_CODE in
    (SELECT DEPT_CODE FROM CLASSES WHERE Classid in
                          (SELECT Classid FROM ENROLLMENTS WHERE B# = dropB#)) and
               COURSE# in (SELECT COURSE# FROM Classes WHERE Classid in
                                  (SELECT Classid FROM Enrollments WHERE B# = dropB#))
                  and PRE_DEPT_CODE = dCode and PRE_COURSE# = c#;
                   IF countPre != 0 THEN
                          error_message := 'The drop is not permitted because another class
the student registered uses it as a
prerequisite.';
                           RETURN;
                   END IF;
          DELETE FROM Enrollments WHERE B# = dropB# and Classid = dropClassid;
          SELECT class_size INTO newSize
           FROM Classes WHERE Classid = dropClassid;
    IF newSize = 0 THEN
             error_message := 'The class now has no students';
           END IF;
          SELECT COUNT(Classid) into numClasses
          FROM Enrollments WHERE B# = dropB#;
          IF numClasses = 0 THEN
                  error_message := 'This student is not enrolled in any classes';
          END IF;
  END IF;
 END;
```

```
 --Q.7.
 --Done by: Sean Gallagher
/* procedure in the package to delete a student from the Students table */
 procedure del_student(
 delB# IN Students.B#%type,error_message out varchar2) IS

 --local
 count_B# Students.B#%type;
 BEGIN
  SELECT COUNT(*) into count_B# FROM Students Where B# = delB#;
  IF (count_B# = 0) THEN
      error_message := 'The B# is invalid';
  ELSE
      DELETE Students WHERE B# = delB#;
      Commit;
  END IF;
 END;
END;
/
show errors;

 --Triggers
 --Done by: Kundan Shrivastav
/* triggers to add tuples to the Logs table */
/*Enrollment Insert*/
 create or replace trigger enrollments_insert
 after insert on enrollments
 for each row
 Declare
  user_log varchar2(20);
  operation_log varchar2(20) default 'insert';
  key_value_log varchar2(50);
  B#_log enrollments.B#%type;
  classid_log enrollments.classid%type;
  table_name_log nvarchar2(20) default 'enrollments';
  id_log Number;
 Begin
  B#_log := :new.B#;
  classid_log := :new.classid;
  key_value_log := (B#_log||','||classid_log);
  id_log := logs_seq.nextval;
  select user into user_log from dual;
  Insert into logs
```

```
 values(id_log,user_log,sysdate,table_name_log,operation_log,key_value_log);
 Update classes
 set class_size = class_size+1
 where classid = classid_log;
End;
/

/*Enrollment Delete*/
 create or replace trigger ENROLLMENTS_DELETE
 AFTER DELETE ON Enrollments
 FOR EACH ROW
 DECLARE
  user_log varchar2(20);
  operation_log varchar2(20) default 'delete';
  key_value_log varchar2(50);
  B#_log enrollments.B#%type;
  classid_log enrollments.classid%type;
  table_name_log nvarchar2(20) default 'enrollments';
  id_log Number;
 BEGIN
  B#_log := :old.B#;
  classid_log := :old.classid;
  key_value_log := (B#_log||','||classid_log);
  id_log := logs_seq.nextval;
  select user into user_log from dual;
  Insert into logs
  values(id_log,user_log,sysdate,table_name_log,operation_log,key_value_log);
  Update classes
  set class_size = class_size-1
  where classid = classid_log;
 END;
 /

/*Student Delete*/
create or replace trigger STUDENT_DELETE
 AFTER DELETE ON Students
 FOR EACH ROW
 DECLARE
  user_log varchar2(20);
  operation_log varchar2(20) default 'delete';
  B#_log enrollments.B#%type;
  table_name_log nvarchar2(20) default 'students';
  id_log Number;
 BEGIN
```

```
 B#_log := :old.B#;
 id_log := logs_seq.nextval;
 select user into user_log from dual;
 Insert into logs
 values(id_log,user_log,sysdate,table_name_log,operation_log,B#_log);
 Delete From Enrollments Where B# = B#_log;
 UPDATE Classes SET TA_B# = NULL WHERE TA_B# = B#_log;
 DELETE FROM TAs WHERE B# = B#_log;
END;
/
show errors;
```

---

6. JAVA CODE:

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.Types;
import java.util.*;
import oracle.jdbc.OracleCallableStatement;
import oracle.jdbc.OracleTypes;
import oracle.jdbc.pool.OracleDataSource;

class DeleteStudent{
	public static void deleteStudent(Connection conn) {
		try
		{
			BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
			System.out.println("Student Bno: ");
			String Bno = br.readLine();

			CallableStatement stmt = conn.prepareCall("BEGIN
student_registration.del_student(?,?); END;");
			stmt.setString(1,Bno);
			stmt.registerOutParameter(2, java.sql.Types.VARCHAR);
			stmt.execute();

			String err_msg = ((OracleCallableStatement)stmt).getString(2);
```

```
                        if(err_msg == null){
                          System.out.println("\nStudent deleted successfully.");
                    }
                    else{
               System.out.println(err_msg);

                    }
          stmt.close();
                    }
                    catch (Exception e)
                    {
                              e.printStackTrace();
                              System.exit(1);
                    }
          }
}

class DropStudentClass{
          public static void dropStudentClass(Connection conn) {
                    try
                    {
                              BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
                              System.out.println("Student B#: ");
                              String Bno = br.readLine();
                              System.out.println("Enter Class ID: ");
                              String classid = br.readLine();

                              CallableStatement stmt = conn.prepareCall("BEGIN
student_registration.drop_student(?,?,?); END;");
                              stmt.setString(1,Bno);
                              stmt.setString(2,classid);
                              stmt.registerOutParameter(3, java.sql.Types.VARCHAR);
                              stmt.execute();

                              String err_msg = ((OracleCallableStatement)stmt).getString(3);
                         if(err_msg == null){
                              System.out.println("\nStudent dropped from the course successfully.");
                         }
                         else{
                              System.out.println(err_msg);
                         }

                              stmt.close();
```

```
                }
                catch (Exception e)
                {
                        e.printStackTrace();
                        System.exit(1);
                }
        }
}

class Enrollments{
        public static void enrollStudentClass(Connection conn) {
                try
                {
                        BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
                        System.out.println("Student B#: ");
                        String Bno = br.readLine();
                        System.out.println("Enter Class ID: ");
                        String classid = br.readLine();

                        CallableStatement stmt = conn.prepareCall("BEGIN
student_registration.enroll_student(?,?,?); END;");
                        stmt.setString(1,Bno);
                        stmt.setString(2,classid);
                    stmt.registerOutParameter(3, java.sql.Types.VARCHAR);
                    stmt.execute();

                    String err_msg = ((OracleCallableStatement)stmt).getString(3);
                    if(err_msg == null){
                         System.out.println("\nStudent enrolled into course successfully.");
                    }
                    else{
                         System.out.println(err_msg);
                }

                stmt.close();

                }
                catch (Exception e)
                {
                        e.printStackTrace();
                        System.exit(1);
                }
        }
```

```
}

class Prerequisites{
	public static void infoPrerequisites(Connection conn) {
		try
		{
			BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
			System.out.println("Enter Dept Code: ");
			String dept_code = br.readLine();
			System.out.println("Enter Course No: ");
			String course_no = br.readLine();

			CallableStatement stmt = conn.prepareCall("begin
student_registration.get_prerequisites(?,?,?,?); end;");
			stmt.setString(1, dept_code);
			stmt.setInt(2, Integer.parseInt(course_no));
		stmt.registerOutParameter(3,java.sql.Types.VARCHAR);
			stmt.registerOutParameter(4,OracleTypes.CURSOR);

			stmt.execute();

			ResultSet rs = null;
		    try{
			rs = ((OracleCallableStatement)stmt).getCursor(4);
		    }
		    catch(Exception ex){
			String err_msg = ((OracleCallableStatement)stmt).getString(3);
			System.out.println(err_msg);
		    }

			if(rs != null){
				System.out.println("\n\nCOURSE");
			while (rs.next()) {
				System.out.println(rs.getString(1) + rs.getInt(2));
				}
				}

	    String TruncateTable = "Truncate table temp_prerequisites";
			Statement stmt1 = conn.createStatement();
			stmt1.executeQuery(TruncateTable);

		}
		catch(Exception e)
```

```
                {
                        e.printStackTrace();
                        System.exit(1);
                }
        }
}

class TAInfo{
public static void infoTA(Connection conn)
{
                try
                {
                BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
                System.out.println("Enter classid: ");
                String classid = br.readLine();

                CallableStatement stmt = conn.prepareCall("begin
student_registration.ta_info(?,?,?); end;");
                stmt.setString(1,classid);
                stmt.registerOutParameter(2,java.sql.Types.VARCHAR);
                stmt.registerOutParameter(3,OracleTypes.CURSOR);
                stmt.execute();
                ResultSet rs = null;
                try{
        rs = ((OracleCallableStatement)stmt).getCursor(3);
                }
                catch(Exception ex){
                String err_msg = ((OracleCallableStatement)stmt).getString(2);
                System.out.println(err_msg);
                }
                if(rs != null){
                while (rs.next()) {
                System.out.println(rs.getString(1) + "\t" + rs.getString(2) + "\t" + rs.getString(3));
                        }
                        }
                }

                catch(Exception e)
                {
                e.printStackTrace();
                System.exit(1);
                }
        }
}
```

```
class ShowTable{
	public static void showTableInfo(int choice, Connection conn)
	{
		switch(choice)
		{
			case 1:
			{
				try
				{
			CallableStatement stmt = conn.prepareCall("BEGIN
student_registration.show_students(?); END;");
				stmt.registerOutParameter(1, OracleTypes.CURSOR);
				stmt.execute();
				ResultSet rs = ((OracleCallableStatement)stmt).getCursor(1);

				while (rs.next())
				{

System.out.format("%-4s %-15s %-15s %-10s %.2f %-20s %-15s %-
6s\n",rs.getString(1),rs.getString(2),rs.getString(3),rs.getString(4),rs.getDouble(5),rs.getString(6)
,rs.getString(7).substring(0,11),rs.getString(8));

					}
				rs.close();
				}
				catch (Exception e)
				{
					e.printStackTrace();
					System.exit(1);
				}
				break;
			}

			case 2:
			{
				try
				{
		CallableStatement stmt = conn.prepareCall("BEGIN
student_registration.show_courses(?); END;");
		stmt.registerOutParameter(1, OracleTypes.CURSOR); //REF CURSOR
				stmt.execute();
				ResultSet rs = ((OracleCallableStatement)stmt).getCursor(1);
				while (rs.next())
```

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

                                catch (Exception e)
                                {
                                        e.printStackTrace();
                                        System.exit(1);
                                }
                                break;
                        }

                    case 3:
                    {
                    try
                    {
                            CallableStatement stmt = conn.prepareCall("BEGIN
student_registration.show_TAs(?); END;");
                            stmt.registerOutParameter(1, OracleTypes.CURSOR);
                    stmt.execute();
                    ResultSet rs = ((OracleCallableStatement)stmt).getCursor(1);

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

                    catch (Exception e)
                    {
                            e.printStackTrace();
                            System.exit(1);
                    }
                    break;
                    }

                            case 4:
```

```
                        {
                                try
                                {
                            CallableStatement stmt = conn.prepareCall("BEGIN
student_registration.show_classes(?); END;");
                            stmt.registerOutParameter(1, OracleTypes.CURSOR); //REF CURSOR
                            stmt.execute();
                            ResultSet rs = ((OracleCallableStatement)stmt).getCursor(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)+"\t"
                                                  + rs.getString(9)+"\t"
                                                  + rs.getString(10));
                                  }
                                  rs.close();
                                }
                                catch (Exception e)
                                {
            e.printStackTrace();
                                        System.exit(1);
                                }
                                break;
                        }

                        case 5:
                {
                try
                {
                        CallableStatement stmt = conn.prepareCall("BEGIN
student_registration.show_enrollments(?); END;");
                        stmt.registerOutParameter(1, OracleTypes.CURSOR);
                stmt.execute();
                ResultSet rs = ((OracleCallableStatement)stmt).getCursor(1);
                while (rs.next())
                {
                        System.out.println(rs.getString(1)+"\t"
                                                  + rs.getString(2)+"\t"
```

```
                                                    + rs.getString(3));
                        }
                        rs.close();
                    }
                    catch (Exception e)
                    {
                e.printStackTrace();
                        System.exit(1);
                    }
                    break;
                    }

                        case 6:
                        {
                                try
                                {
                                   CallableStatement stmt = conn.prepareCall("BEGIN
student_registration.show_prerequisites(?); END;");
                                   stmt.registerOutParameter(1, OracleTypes.CURSOR); //REF
CURSOR
                                   stmt.execute();
                                   ResultSet rs = ((OracleCallableStatement)stmt).getCursor(1);
                                   while (rs.next())
                                   {
                                            System.out.println(rs.getString(1)+"\t"
                                                    + rs.getInt(2)+"\t"
                                                    + rs.getString(3)+"\t"
                                                    + rs.getInt(4));
                                   }
                                   rs.close();
                                }
                                catch (Exception e)
                                {
                e.printStackTrace();
                                        System.exit(1);
                                }
                                break;
                        }

                        case 7:
                        {
                                try
                                {
```

```
                                        CallableStatement stmt = conn.prepareCall("BEGIN
student_registration.show_logs(?); END;");
                                        stmt.registerOutParameter(1, OracleTypes.CURSOR);
//REF CURSOR
                                    stmt.execute();
                                    ResultSet rs = ((OracleCallableStatement)stmt).getCursor(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();
                                }
                                catch (Exception e)
                                {
                                        e.printStackTrace();
                                        System.exit(1);
                                }
                                break;
                        }
                }
        }
}

//Main Class
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("vsaxena1","varun");

                   while(true)
                {
                                System.out.println();
                                System.out.println("*****Main Menu*****");

                                System.out.println("1.View Table data");
                                System.out.println("2.View TA Information");
                                System.out.println("3.View Prerequisites Information");
```

```
                        System.out.println("4.Enroll a Student in Class");
                        System.out.println("5.Drop a Student from Class");
                        System.out.println("6.Delete a Student");
                        System.out.println("7.Exit");
                int n = 0;
                Scanner sc = new Scanner(System.in);
                System.out.println("Please select an option from the above : ");
                n = sc.nextInt();

                switch(n)
                {
                        case 1:
                        {
                                ShowTable showTable = new ShowTable();

                        System.out.println();
                                                System.out.println("***Select Table***");
                                                System.out.println("1.Students\n"
                                                        + "2.Courses\n"
                                                        + "3.TAs\n"
                                                        + "4.Classes\n"
                                                        + "5.Enrollments\n"
                                                        + "6.Prerequisites\n"
                                        + "7.Logs\n");
                        int m = 0;
                        try {
                                                BufferedReader inputReader = new
BufferedReader(new InputStreamReader(System.in));
                                                do
                                                {
                                                        System.out.println("Enter Choice From
Above Options");
                                                        m =
Integer.parseInt(inputReader.readLine());
                                                }
                                                while(m < 1 || m > 7);
                                }
                                catch (Exception e) {
                                                e.printStackTrace();
                                                System.exit(1);
                        }
                        showTable.showTableInfo(m,conn);
                                        break;
                                        }
```

```
                case 2:
                {
                        TAInfo ta = new TAInfo();
                        ta.infoTA(conn);
                        break;
                }

                case 3:
                {
                        Prerequisites prerequisite = new Prerequisites();
                        prerequisite.infoPrerequisites(conn);
                        break;
                }

                case 4:
                {
                        Enrollments enroll = new Enrollments();
                        enroll.enrollStudentClass(conn);
                        break;
                }

                case 5:
                {
                        DropStudentClass drop = new DropStudentClass();
                        drop.dropStudentClass(conn);
                        break;
                }
                case 6:
                {
                        DeleteStudent delStudent = new DeleteStudent();
                        delStudent.deleteStudent(conn);
                        break;
                }
                case 7:
                {
                        System.exit(1);;
                        break;
                }
        }
        }
}
catch (Exception e) {
        System.out.println("Connection not Established. Try Again");
```

```
                System.exit(1);
        }
        }
}
```

7. CONCLUSION

Thus, we have implemented the project successfully using PL-SQL and Java by connecting it using JDBC within the required time frame provided.