```
//////////////////////////////sequence.sql////////////////////////////////////
create sequence log_number minvalue 1000000 maxvalue 9999999 increment by 1 start with
1000000;
alter table logs modify logid number(7);


/////////////////////triggers.sql//////////////////////////////////////////////////
create or replace trigger t_after_enroll
after insert on enrollments
for each row
begin
    update classes set class_size = class_size+1 where classid=:new.classid;
    insert into logs values(log_number.nextval,user,sysdate,'enrollments','insert', :new.sid || ',' ||
:new.classid);
    exception
        when no_data_found then
            null;
end;
/

create or replace trigger t_after_add_student
after insert on students
for each row
begin
insert into logs values(log_number.nextval, user, sysdate, 'students', 'insert', :new.sid);
end;
/

create or replace trigger t_before_delete_student
before delete on students
for each row
begin
delete from enrollments where enrollments.sid = :old.sid;
```

```
exception
        when no_data_found then
                null;
end;
/

create or replace trigger log_entry_after_delete_stud
after delete on students
for each row
begin
insert into logs values(log_number.nextval,user,sysdate,'students','delete', :old.sid);
end;
/

create or replace trigger t_after_delete_enroll
after delete on enrollments
for each row
begin
        update classes set class_size = class_size-1 where classid=:old.classid;
        insert into logs values(log_number.nextval,user,sysdate,'enrollments','delete', :old.sid);
        exception
                when no_data_found then
                        null;
end;
/

create or replace trigger t_before_delete_class
before delete on classes
for each row
begin
delete from enrollments where enrollments.classid = :old.classid;
```

```
exception
	when no_data_found then
		null;
end;
/

show errors
////////////////////// packagespecification.sql////////////////////////////////////////
create or replace package project2procedurefunction as
	stud_enroll_id students.sid%type;
	class_enroll_id classes.classid%type;
	del_stud_sid students.sid%type;
	add_stud_sid students.sid%type;
	type  refcur is ref cursor;

	procedure show_students;
	procedure show_courses;
	procedure show_prerequisites;
	procedure show_classes;
	procedure show_enrollments;
	procedure show_logs;
	procedure add_student(student_id in char,first_name in varchar2,last_name in varchar2,student_status in varchar2,student_gpa in number, student_email in varchar2);
	procedure studentclass_info(student_id in char);
	procedure prerequisite_info(departmentcode in varchar2,coursenumber in number);
	procedure class_info(s_classid in char);
	procedure enrollment(student_id in char,c_classid in char);
	procedure drop_student_from_class(student_id in char,c_classid in char);
	procedure delete_student(student_id in char);

	/* Procedures/functions to use in java/jdbc*/
```

```
        function getstudents return refcur;

        function getcourses return refcur;

    function getenrollment return refcur;

    function getclasses return refcur;

    function getprereq return refcur;

    function getlogs return refcur;


        procedure j_studentclass_info(student_id in char, show_message OUT varchar2, sc_recordset
OUT SYS_REFCURSOR);

        procedure j_prerequisite_info(departmentcode in varchar2,coursenumber in number,
p_recordset OUT SYS_REFCURSOR,show_message OUT varchar2);

        procedure j_class_info(s_classid in char, show_message OUT varchar2, c_recordset OUT
SYS_REFCURSOR);

        procedure j_enrollment(student_id in char,c_classid in char, show_message OUT varchar2);

        procedure j_drop_student_from_class(student_id in char,c_classid in char, show_message1
OUT varchar2, show_message2 OUT varchar2, show_message3 OUT varchar2);

        procedure j_delete_student(student_id in char, show_message OUT varchar2);


end;

/



///////////////////////////////////packagebody.sql/////////////////////////////

set serveroutput on

create or replace package body project2procedurefunction as


procedure show_students is

cursor ss is select * from students;

ss_rec ss%rowtype;

begin

open ss;
```

```
fetch ss into ss_rec;

while ss%found loop

dbms_output.put_line(ss_rec.sid || ',' || ss_rec.firstname || ',' || ss_rec.lastname || ',' ||
ss_rec.status || ',' || ss_rec.gpa || ',' || ss_rec.email);

fetch ss into ss_rec;

end loop;

close ss;

end;


procedure show_courses is

cursor sc is select * from courses;

sc_rec sc%rowtype;

begin

open sc;

fetch sc into sc_rec;

while sc%found loop

dbms_output.put_line(sc_rec.dept_code || ',' || sc_rec.course_no || ',' || sc_rec.title);

fetch sc into sc_rec;

end loop;

close sc;

end;


procedure show_prerequisites is

cursor sp is select * from prerequisites;

sp_rec sp%rowtype;

begin

open sp;

fetch sp into sp_rec;

while sp%found loop

dbms_output.put_line(sp_rec.dept_code || ',' || sp_rec.course_no || ',' || sp_rec.pre_dept_code
|| ',' || sp_rec.pre_course_no);

fetch sp into sp_rec;
```

```
end loop;

close sp;

end;


procedure show_classes is

cursor scs is select * from classes;

scs_rec scs%rowtype;

begin

open scs;

fetch scs into scs_rec;

while scs%found loop

dbms_output.put_line(scs_rec.classid || ',' || scs_rec.dept_code || ',' || scs_rec.course_no || ',' ||
scs_rec.sect_no || ',' || scs_rec.year || ',' || scs_rec.semester || ',' || scs_rec.limit || ',' ||
scs_rec.class_size);

fetch scs into scs_rec;

end loop;

close scs;

end;


procedure show_enrollments is

cursor se is select * from enrollments;

se_rec se%rowtype;

begin

open se;

fetch se into se_rec;

while se%found loop

dbms_output.put_line(se_rec.sid || ',' || se_rec.classid || ',' || se_rec.lgrade);

fetch se into se_rec;

end loop;

close se;

end;
```

```
procedure show_logs is

cursor sl is select * from logs;

sl_rec sl%rowtype;

begin

open sl;

fetch sl into sl_rec;

while sl%found loop

dbms_output.put_line(sl_rec.logid || ',' || sl_rec.who || ',' || sl_rec.time || ',' || sl_rec.table_name
|| ',' || sl_rec.operation || ',' || sl_rec.key_value);

fetch sl into sl_rec;

end loop;

close sl;

end;


procedure add_student

(student_id in char,first_name in varchar2,last_name in varchar2,student_status in
varchar2,student_gpa in number, student_email in varchar2) is

begin

add_stud_sid:=student_id;

insert into students
values(student_id,first_name,last_name,student_status,student_gpa,student_email);

commit;

end;


procedure studentclass_info(student_id in char) as


s_sid students.sid%type;

s_fname students.firstname%type;

s_lname students.lastname%type;

s_status students.status%type;

c_classid classes.classid%type;

c_courseid varchar2(7);
```

```
c_title courses.title%type;
flag1 number;
flag2 number;
p1_classid classes.classid%type;
a1 sys_refcursor;

begin
    select count(*) into flag1 from students where sid = student_id;
    if(flag1=0) then
        raise_application_error(-20001,'The sid is invalid.');
    end if;

    select count(classid) into flag2 from enrollments where sid = student_id;
    if(flag2=0) then
        raise_application_error(-20002,'The student has not taken any course.');
    end if;

        select sid,firstname,lastname,status into s_sid,s_fname,s_lname,s_status from students
where sid = student_id;
    dbms_output.put_line(s_sid || ',' || s_fname ||',' || s_lname || ',' || s_status);

        open a1 for SELECT cl.classid, concat(cl.dept_code, cl.course_no) as courseid, c2.title
        FROM classes cl, enrollments e, courses c2, students s
        WHERE s.sid=student_id AND s.sid=e.sid AND e.classid = cl.classid AND cl.course_no =
c2.course_no;
        loop
            fetch a1 into c_classid,c_courseid,c_title;
            exit when a1%notfound;
            dbms_output.put_line(c_classid || ',' || c_courseid || ',' || c_title);
        end loop;
        close a1;
```

```
end;



procedure prerequisite_info(departmentcode in varchar2,coursenumber in number) as


no_prereq exception;
flag number:=1;
cursor c1 is select pre_dept_code,pre_course_no from prerequisites where dept_code =
departmentcode and course_no=coursenumber;
c1_rec c1%rowtype;


begin
    select count(*) into flag from prerequisites where dept_code=departmentcode and
course_no=coursenumber;
    if(flag=0) then
    raise no_prereq;
    end if;


    open c1;
    fetch c1 into c1_rec;
    while c1%found loop
    dbms_output.put_line(c1_rec.pre_dept_code || c1_rec.pre_course_no);
    fetch c1 into c1_rec;
    end loop;
    close c1;
    prerequisite_info(c1_rec.pre_dept_code,c1_rec.pre_course_no);


    exception
    when no_prereq then
        dbms_output.put_line('');
end;
```

```
procedure class_info(s_classid in char) as


flag number:=0;

flag1 number:=0;

c_title courses.title%type;

c_id classes.classid%type;

c_semester classes.semester%type;

c_year classes.year%type;

s_sid students.sid%type;

s_fname students.firstname%type;

s_lname students.lastname%type;

a1 sys_refcursor;


begin

    select count(*) into flag from classes where classid = s_classid;

    if(flag=0) then

        raise_application_error(-20003,'The cid is invalid');

    end if;


    select count(*) into flag1 from enrollments where classid = s_classid;

    if(flag1=0) then

        raise_application_error(-20004,'No student is enrolled in the class.');

    end if;


    select c1.classid,c1.semester,c1.year,c2.title into c_id,c_semester,c_year,c_title from classes
c1,courses c2 where classid = s_classid and c1.dept_code = c2.dept_code and c1.course_no =
c2.course_no;

    dbms_output.put_line(c_id || ',' || c_title ||',' || c_semester || ',' || c_year);


    open a1 for SELECT s.sid,firstname,lastname

    FROM classes cl, enrollments e, students s

    WHERE cl.classid = s_classid AND cl.classid = e.classid AND e.sid=s.sid;
```

```
        loop
                fetch a1 into s_sid,s_fname,s_lname;
                exit when a1%notfound;
                dbms_output.put_line(s_sid || ',' || s_fname || ',' || s_lname);
        end loop;
        close a1;

end;




procedure enrollment(student_id in char,c_classid in char) as


flag1 NUMBER:=1;
flag2 NUMBER:=1;
isavailable NUMBER:=1;
flag3 NUMBER:=0;
sem classes.semester%type;
yr classes.year%type;
flag4 NUMBER:=0;
count_var number:=0;
d_deptcode classes.dept_code%type;
c_courseno classes.course_no%type;
sc1_deptcode classes.dept_code%type;
sc1_courseno classes.course_no%type;
sc2_cid classes.classid%type;
e_lgrade enrollments.lgrade%type;
pre_count NUMBER:=1;
sc1 sys_refcursor;
sc2 sys_refcursor;
cursor c1 is select classid from enrollments where sid=student_id;
```

```
c1_rec c1%rowtype;

BEGIN
select count(*) into flag1 from students where sid=student_id;
if(flag1=0)then
	RAISE_APPLICATION_ERROR(-20004,'sid not found');
end if;

select count(*) into flag2 from classes where classid=c_classid;
if(flag2=0)then
	RAISE_APPLICATION_ERROR(-20005,'The classid is invalid.');
end if;

select (limit-class_size) into isavailable from classes where classid=c_classid;
if(isavailable = 0) then
	RAISE_APPLICATION_ERROR(-20006,'The class is full.');
end if;

select count(*) into flag3 from enrollments where sid=student_id and classid=c_classid;
if(flag3=1) then
	RAISE_APPLICATION_ERROR(-20007,'The student is already in this class');
end if;

open c1;
fetch c1 into c1_rec;
while c1%found loop
select count(*) into flag4 from classes where classid=c1_rec.classid and semester='Fall' and
year=2020;
if(flag4!=0) then
count_var:=count_var+1;
end if;
```

```
fetch c1 into c1_rec;

end loop;

close c1;


if(count_var=3)then

RAISE_APPLICATION_ERROR(-20008,'You are overloaded.');

end if;


select dept_code, course_no into d_deptcode, c_courseno from classes where classid=c_classid;


select count(*) into pre_count from prerequisites where dept_code=d_deptcode and
course_no=c_courseno;


if(pre_count=0) then

        stud_enroll_id:=student_id;

        class_enroll_id:=c_classid;

    insert into enrollments values(student_id,c_classid,null);

    commit;

    return;

end if;


open sc1 for select pre_dept_code, pre_course_no from prerequisites

where dept_code=d_deptcode and course_no=c_courseno;

fetch sc1 into sc1_deptcode, sc1_courseno;

  open sc2 for select classid from classes where dept_code=sc1_deptcode and
course_no=sc1_courseno;

  fetch sc2 into sc2_cid;

    select lgrade into e_lgrade from enrollments where classid=sc2_cid and sid=student_id;

    if(e_lgrade > 'C') then

        RAISE_APPLICATION_ERROR(-20009,'Prerequisite courses have not been completed.');

    elsif(e_lgrade <= 'C') then

                stud_enroll_id:=student_id;
                class_enroll_id:=c_classid;
        insert into enrollments values(student_id,c_classid,null);
        commit;
        return;
    end if;
  close sc2;
close sc1;

exception
    when NO_DATA_FOUND then
        RAISE_APPLICATION_ERROR(-20010,'Prerequisite courses have not been completed.');

end;



procedure drop_student_from_class(student_id in char,c_classid in char) as
flag1 NUMBER:=1;
flag2 NUMBER:=1;
flag3 NUMBER:=1;
flag4 NUMBER:=0;
class_count number:=0;
isavailable NUMBER:=1;
deptcode classes.dept_code%type;
courseno classes.course_no%type;
p_deptcode classes.dept_code%type;
p_courseno classes.course_no%type;
cid classes.classid%type;
e_lgrade number;
sc1 sys_refcursor;
sc2 sys_refcursor;
```

```
cursor c1 is select classid from enrollments where sid=student_id;
c1_rec c1%rowtype;

begin
select count(*) into flag1 from students where sid=student_id;
if(flag1=0)then
RAISE_APPLICATION_ERROR(-20011,'The sid is invalid.');
end if;

select count(*) into flag2 from classes where classid=c_classid;
if(flag2=0)then
RAISE_APPLICATION_ERROR(-20012,'The classid is invalid.');
end if;

select count(*) into flag3 from enrollments where sid=student_id and classid=c_classid;
if(flag3=0) then
RAISE_APPLICATION_ERROR(-20013,'The student is not enrolled in the class.');
end if;

select dept_code, course_no into p_deptcode, p_courseno from classes where classid=c_classid;
open sc1 for select dept_code, course_no from prerequisites
where pre_dept_code=p_deptcode and pre_course_no=p_courseno;
fetch sc1 into deptcode, courseno;
   open sc2 for select classid from classes where dept_code=deptcode and course_no=courseno;
   fetch sc2 into cid;
   select count(*) into e_lgrade from enrollments where classid=cid and sid=student_id;
   if(e_lgrade!=0) then
             RAISE_APPLICATION_ERROR(-20014,'The drop is not permitted because another class the
student registered uses it as a prerequisite.');
   end if;
   close sc2;
close sc1;

stud_enroll_id:=student_id;
class_enroll_id:=c_classid;

delete from enrollments where sid=student_id and classid=c_classid;
commit;
dbms_output.put_line('Student dropped from the class.');

open c1;
fetch c1 into c1_rec;
while c1%found loop
select count(*) into flag4 from classes where classid=c1_rec.classid and semester='Fall' and
year=2020;
if(flag4!=0) then
class_count:=class_count+1;
end if;
fetch c1 into c1_rec;
end loop;
close c1;

if(class_count=0) then
dbms_output.put_line('This student is enrolled in no class.');
end if;

select class_size into isavailable from classes where classid=c_classid;
if(isavailable=0) then
dbms_output.put_line('The class now has no students.');
end if;
end;
```

```
PROCEDURE delete_student(student_id in char) AS
flag NUMBER:=1;
BEGIN

select count(*) into flag from students where sid=student_id;
if(flag=0)then
RAISE_APPLICATION_ERROR(-20019,'The sid is invalid.');
end if;
del_stud_sid:=student_id;
delete from students where sid=student_id ;
commit;
dbms_output.put_line('Student deleted.');
END;


/*********************************************************************************
*****************************
Procedures/functions to use in java/jdbc
**********************************************************************************
*****************************/



function getstudents return refcur is
gs refcur;
begin
open gs for select * from students;
return gs;
end;

function getcourses return refcur is
```

```
gc refcur;

begin

open gc for select * from courses;

return gc;

end;


function getenrollment return refcur is

gs refcur;

begin

open gs for select * from enrollments;

return gs;

end;


function getclasses return refcur is

gs refcur;

begin

open gs for select * from classes;

return gs;

end;


function getprereq return refcur is

gs refcur;

begin

open gs for select * from prerequisites;

return gs;

end;


function getlogs return refcur is

gs refcur;

begin

open gs for select * from logs;
```

```
return gs;

end;




procedure j_studentclass_info(student_id in char, show_message OUT varchar2, sc_recordset OUT
SYS_REFCURSOR) as


flag1 number := 1;

flag2 number := 1;

msg varchar2(500) := '';


begin

    select count(*) into flag1 from students where sid = student_id;

    select count(classid) into flag2 from enrollments where sid = student_id;


        if(flag1=0) then

        msg := 'The sid is invalid.';

                show_message := msg;

        elsif(flag2=0 and flag1 != 0) then

                msg := 'The student has not taken any course.';

                show_message := msg;

        else

                open sc_recordset for SELECT s.sid,s.firstname,s.lastname,s.status,cl.classid,
concat(cl.dept_code, cl.course_no) as courseid, c2.title FROM classes cl, enrollments e, courses c2,
students s WHERE s.sid=student_id AND s.sid=e.sid AND e.classid = cl.classid AND cl.course_no =
c2.course_no;

        end if;


end;
```

```
procedure j_prerequisite_info(departmentcode in varchar2,coursenumber in number, p_recordset
OUT SYS_REFCURSOR,show_message OUT varchar2) as


flag1 number := 1;

msg varchar2(500) := '';


begin


        select count(*) into flag1 from prerequisites where dept_code = departmentcode AND
course_no = coursenumber;


        if(flag1=0) then

                msg := 'No prerequisite';

                show_message := msg;

        else

                OPEN p_recordset FOR SELECT concat(pre_dept_code, pre_course_no) FROM prerequisites
CONNECT BY PRIOR pre_dept_code = dept_code AND PRIOR pre_course_no = course_no START
WITH dept_code = departmentcode AND course_no = coursenumber;

        end if;

end;



procedure j_class_info(s_classid in char, show_message OUT varchar2, c_recordset OUT
SYS_REFCURSOR) as

flag number:=1;

flag1 number:=1;

msg varchar2(500) := '';


begin

        select count(*) into flag from classes where classid = s_classid;

                select count(*) into flag1 from enrollments where classid = s_classid;


        if(flag=0) then
```

```
            msg := 'The cid is invalid';
                        show_message := msg;
      elsif(flag1=0 and flag !=0) then
                        msg := 'No student is enrolled in the class.';
                        show_message := msg;
            else
                  open c_recordset for select c1.classid, c2.title, c1.semester, c1.year, e.sid, s.firstname,
s.lastname from classes c1 INNER JOIN courses c2 ON c1.dept_code = c2.dept_code AND
c1.course_no = c2.course_no INNER JOIN enrollments e ON e.classid = c1.classid INNER JOIN
students s ON s.sid = e.sid WHERE c1.classid = s_classid;


            end if;


end;




procedure j_enrollment(student_id in char,c_classid in char, show_message OUT varchar2) as


flag1 NUMBER:=1;
flag2 NUMBER:=1;
isavailable NUMBER:=1;
flag3 NUMBER:=0;
sem classes.semester%type;
yr classes.year%type;
flag4 NUMBER:=0;
count_var number:=0;
d_deptcode classes.dept_code%type;
c_courseno classes.course_no%type;
sc1_deptcode classes.dept_code%type;
sc1_courseno classes.course_no%type;
sc2_cid classes.classid%type;
```

```
e_lgrade enrollments.lgrade%type;
pre_count NUMBER:=1;
sc1 sys_refcursor;
sc2 sys_refcursor;
cursor c1 is select classid from enrollments where sid=student_id;
c1_rec c1%rowtype;
msg varchar2(500) := '';

BEGIN
select count(*) into flag1 from students where sid=student_id;
select count(*) into flag2 from classes where classid=c_classid;

if(flag1=0)then
    msg := 'sid not found';
        show_message := msg;
        return;
elsif(flag2=0 and flag1 !=0)then
    msg := 'The classid is invalid.';
        show_message := msg;
        return;
end if;

select (limit-class_size) into isavailable from classes where classid=c_classid;
if(isavailable = 0) then
    msg := 'The class is full.';
        show_message := msg;
        return;
end if;



select count(*) into flag3 from enrollments where sid=student_id and classid=c_classid;
```

```
if(flag3=1) then
        msg := 'The student is already in this class';
                show_message := msg;
                return;
end if;


open c1;
fetch c1 into c1_rec;
while c1%found loop
select count(*) into flag4 from classes where classid=c1_rec.classid and semester='Fall' and
year=2020;
if(flag4!=0) then
count_var:=count_var+1;
end if;
fetch c1 into c1_rec;
end loop;
close c1;


if(count_var=3)then
        msg := 'You are overloaded.';
                show_message := msg;
                return;
end if;


        select dept_code, course_no into d_deptcode, c_courseno from classes where classid=c_classid;
        select count(*) into pre_count from prerequisites where dept_code=d_deptcode and
course_no=c_courseno;


        if(pre_count=0) then
                        stud_enroll_id:=student_id;
                        class_enroll_id:=c_classid;
                insert into enrollments values(student_id,c_classid,null);

                commit;

                return;

        end if;


        open sc1 for select pre_dept_code, pre_course_no from prerequisites

        where dept_code=d_deptcode and course_no=c_courseno;

        fetch sc1 into sc1_deptcode, sc1_courseno;

            open sc2 for select classid from classes where dept_code=sc1_deptcode and
    course_no=sc1_courseno;

            fetch sc2 into sc2_cid;

                select lgrade into e_lgrade from enrollments where classid=sc2_cid and sid=student_id;

                if(e_lgrade > 'C' and flag2!=0 and flag1 !=0) then

                        msg := 'Prerequisite courses have not been completed.';

                        show_message := msg;

                        return;

                elsif(e_lgrade <= 'C') then

                        stud_enroll_id:=student_id;

                        class_enroll_id:=c_classid;

                        insert into enrollments values(student_id,c_classid,null);

                        commit;

                        return;

                end if;

            close sc2;

        close sc1;


        exception

            when no_data_found then

                msg := 'Prerequisite courses have not been completed.';

                show_message := msg;

    end;
```

```
procedure j_drop_student_from_class(student_id in char,c_classid in char, show_message1 OUT
varchar2, show_message2 OUT varchar2, show_message3 OUT varchar2) as

flag1 NUMBER:=1;

flag2 NUMBER:=1;

flag3 NUMBER:=1;

flag4 NUMBER:=0;

class_count number:=0;

isavailable NUMBER:=1;

deptcode classes.dept_code%type;

courseno classes.course_no%type;

p_deptcode classes.dept_code%type;

p_courseno classes.course_no%type;

cid classes.classid%type;

e_lgrade number;

sc1 sys_refcursor;

sc2 sys_refcursor;

msg1 varchar2(500) := '';

msg2 varchar2(500) := '';

msg3 varchar2(500) := '';


cursor c1 is select classid from enrollments where sid=student_id;

c1_rec c1%rowtype;


begin

select count(*) into flag1 from students where sid=student_id;


select count(*) into flag2 from classes where classid=c_classid;
```

```
if(flag1=0)then
    msg1 := 'The sid is invalid.';
        show_message1 := msg1;
elsif(flag2=0 and flag1 !=0)then
    msg1 := 'The classid is invalid.';
        show_message1 := msg1;
else
    select count(*) into flag3 from enrollments where sid=student_id and classid=c_classid;


    select dept_code, course_no into p_deptcode, p_courseno from classes where classid=c_classid;
    open sc1 for select dept_code, course_no from prerequisites
    where pre_dept_code=p_deptcode and pre_course_no=p_courseno;
    fetch sc1 into deptcode, courseno;
  open sc2 for select classid from classes where dept_code=deptcode and course_no=courseno;
  fetch sc2 into cid;
  select count(*) into e_lgrade from enrollments where classid=cid and sid=student_id;
  close sc2;
    close sc1;


    if(flag3=0 and flag2!=0 and flag1 !=0) then
        msg1 := 'The student is not enrolled in the class.';
                show_message1 := msg1;
    elsif(e_lgrade!=0 and flag2!=0 and flag1 !=0) then
        msg1 := 'The drop is not permitted because another class uses it as a prerequisite.';
                show_message1 := msg1;
    else
        stud_enroll_id:=student_id;
        class_enroll_id:=c_classid;


        delete from enrollments where sid=student_id and classid=c_classid;
        commit;
```

```
            open c1;

            fetch c1 into c1_rec;

            while c1%found loop

                  select count(*) into flag4 from classes where classid=c1_rec.classid and semester='Fall'
and year=2020;

                  if(flag4!=0) then

                  class_count:=class_count+1;

                  end if;

            fetch c1 into c1_rec;

            end loop;

            close c1;


            if(class_count=0) then

                        msg2 := 'This student is enrolled in no class.';

                        show_message2 := msg2;

            end if;


            select class_size into isavailable from classes where classid=c_classid;


            if(isavailable=0) then

                        msg3 := 'The class now has no students.';

                        show_message3 := msg3;

            end if;


      end if;

end if;

end;



procedure j_delete_student(student_id in char, show_message OUT varchar2) as
```

```
flag NUMBER:=1;
msg varchar2(500);

begin
select count(*) into flag from students where sid=student_id;

if(flag=0)then
        msg := 'The sid is invalid.';
        show_message := msg;
else
        del_stud_sid:=student_id;
        delete from students where sid=student_id ;
        commit;
end if;
end;

end;
/
show errors

////////////////////UserInterface.java////////////////////////////////
import java.sql.*;
import oracle.jdbc.*;
import java.io.*;
import oracle.jdbc.pool.OracleDataSource;
import java.lang.Object.*;
import java.util.*;
import java.util.Scanner;
import java.util.Date;
import java.text.SimpleDateFormat;
import java.io.BufferedReader;
import java.io.FileInputStream;

import java.io.InputStream;

import java.io.InputStreamReader;


public class UserInterface1{



public static void main(String args[]) throws SQLException{


int option=0,subOption=0, courseno,pre_courseno,section,year,limit,class_size,logid;

String sid,fname,lname,dept,gpa,email,status,classid,title,pre_dept,sem;

boolean flag1=true;

boolean flag2=true;



Connection conn = null;


try{

        //Connection to Oracle server

    OracleDataSource ds = new oracle.jdbc.pool.OracleDataSource();

    ds.setURL("jdbc:oracle:thin:@castor.cc.binghamton.edu:1521:ACAD111");


    Scanner in = new Scanner(System.in);


    /*

    System.out.println("Enter username to connect to database");

    String username = in.next();

    System.out.println("Enter password to connect to database");

    String password = in.next();

    */
```

```
        try{
                conn = ds.getConnection("ptanpur1","dadsgift16");
        }
        catch(Exception e){

                System.out.println("wrong !!! username or password. Please check.");
                return;
        }



        while(flag1 == true){
        flag2=true;

System.out.println("###################################################################
################################");
                System.out.println("-----------------Main Menu---------------------------");
        System.out.println("\nPlease select required option to perform action on specific table");

System.out.println("\n1.Students\n2.Courses\n3.Prerequisites\n4.Classes\n5.Enrollments\n6.Logs\n
0.exit\n");

System.out.println("###################################################################
################################");
        option=in.nextInt();
        switch(option){
                case 1:
                        while(flag2 == true){

System.out.println("*****************************************************************
******************************");
                        System.out.println("\nWhich Action you want to perform?");
                        System.out.println(" 1.Display Students\n 2.Add Student\n 3.Delete Student\n 4.Show
student's class info \n 0. exit\n");
```

```
System.out.println("******************************************************************
*********************************");
                    subOption=in.nextInt();
                    switch(subOption){
                        case 1:
                            CallableStatement cs1 = conn.prepareCall("begin ? :=
project2procedurefunction.getstudents(); end;");
                            cs1.registerOutParameter(1, OracleTypes.CURSOR);


                            cs1.execute();
                            ResultSet rs1=(ResultSet)cs1.getObject(1);

System.out.println("***************************************************************");
                            System.out.printf("sid\tfirstname\tlastname\tstatus\tgpa\temail\n");

System.out.println("***************************************************************");
                            while(rs1.next()){
                            System.out.println(rs1.getString(1) + "\t" + rs1.getString(2) + "\t" +
rs1.getString(3) +"\t"+ rs1.getString(4) + "\t" + rs1.getDouble(5) + "\t" + rs1.getString(6));
                            System.out.println();
                            }
                            cs1.close();
                            break;
                        case 2:
                            System.out.println("Please enter student details to insert");
                                        System.out.println("Enter sid:");
                                        sid=in.next();
                                        System.out.println("Enter first name:");
                                        fname=in.next();
                                        System.out.println("Enter last name:");
                                        lname=in.next();
                                        System.out.println("Enter status('freshman',
'sophomore', 'junior', 'senior', 'graduate'):");
                                        status=in.next();
                                        System.out.println("Enter gpa between 0 and 4:");
                                        gpa=in.next();
                                        System.out.println("Enter email:");
                                        email=in.next();

                                        CallableStatement cs7 = conn.prepareCall("begin
project2procedurefunction.add_student(?,?,?,?,?,?); end;");
                                        cs7.setString(1, sid);
                                        cs7.setString(2, fname);
                                        cs7.setString(3, lname);
                                        cs7.setString(4, status);
                                        cs7.setString(5, gpa);
                                        cs7.setString(6, email);

                                        cs7.execute();
                                        System.out.println("\nStudent Added");
                                        cs7.close();
                                        break;
                case 3:
                    System.out.println("Please enter sid :");
                                        sid = in.next();
                                        CallableStatement cs13 = conn.prepareCall("begin
project2procedurefunction.j_delete_student(?,?); end;");
                                        cs13.setString(1,sid);
                                        cs13.registerOutParameter(2, java.sql.Types.VARCHAR);
                                        cs13.execute();

                                        String mmsg4 =
((OracleCallableStatement)cs13).getString(2);
```

```
                                        if(mmsg4 != null)
                                        {
                                                System.out.println(mmsg4);
                                        }
                                        else
                                        {
                                                System.out.println("Student deleted
successfully");
                                        }
                                        cs13.close();
                                        break;
                        case 4:
                                System.out.println("Please enter student sid :");
                                sid = in.next();
                                        CallableStatement cs8 = conn.prepareCall("begin
project2procedurefunction.j_studentclass_info(?,?,?); end;");
                                        cs8.setString(1, sid);
                                        cs8.registerOutParameter(2, java.sql.Types.VARCHAR);
                                        cs8.registerOutParameter(3, OracleTypes.CURSOR);
//REF CURSOR
                                        cs8.execute();
                                        ResultSet rs8 = null;
                                        String mmsg1 = null;

                                        rs8 = ((OracleCallableStatement)cs8).getCursor(3);
                                        mmsg1 = ((OracleCallableStatement)cs8).getString(2);

                                        if(mmsg1 != null)
                                        {
                                            System.out.println(mmsg1);
                                        }
                                        else

                                        {

System.out.printf("sid\tfirstname\tlastname\tstatus\tclassid\tcourseid\ttitle\n");

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

                                              while(rs8.next()){

                                              System.out.print(rs8.getString(1) + "\t" +
rs8.getString(2) + "\t" + rs8.getString(3) +"\t"+ rs8.getString(4) + "\t" + rs8.getString(5) + "\t" +
rs8.getString(6)+ "\t" + rs8.getString(7));

                                              System.out.println();

                                              }

                                        }


                                        cs8.close();

                                        break;

                      case 0:

                         flag2=false;

                         break;

                      default:

                         System.out.println("Invalid option!");

                         break;

                   }

                   }

                   break;

             case 2:

                   while(flag2 == true){

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

                   System.out.println("\nWhich Action you want to perform?");

                   System.out.println(" 1.Display Courses\n 2.Add course\n 3.Delete course\n 0. exit\n");

System.out.println("************************************************************");
```

```
                    subOption=in.nextInt();
                    switch(subOption){
                        case 1:
                            CallableStatement cs2 = conn.prepareCall("begin ? := 
project2procedurefunction.getcourses(); end;");
                            cs2.registerOutParameter(1, OracleTypes.CURSOR);


                            cs2.execute();
                            ResultSet rs2=(ResultSet)cs2.getObject(1);

System.out.println("****************************************************************");
                            System.out.printf("dept_code\tcourse_no\ttitle\n");

System.out.println("****************************************************************");
                            while(rs2.next()){
                            System.out.println(rs2.getString(1) + "\t" + rs2.getString(2) + "\t" + 
rs2.getString(3));
                            System.out.println();
                            }
                            cs2.close();
                            break;
                        case 2:
                                                        System.out.println("Please enter course details to 
insert");
                                                        System.out.println("Enter dept_code:");
                                                        dept=in.next();
                                                        System.out.println("Enter course_no:");
                                                        courseno=in.nextInt();
                                                        System.out.println("Enter title:");
                                                        title=in.next();


                                                        PreparedStatement insert = 
conn.prepareStatement("insert into courses values(?,?,?)");
```

```
                                        insert.setString(1, dept);
                                        insert.setInt(2, courseno);
                                        insert.setString(3, title);

                                        insert.executeUpdate();
                                        System.out.println("\nCourse Added");
                                        break;
                  case 3:
                                        System.out.println("Please enter course details to
Delete");
                                        System.out.println("Enter dept_code:");
                                        dept=in.next();
                                        System.out.println("Enter course_no:");
                                        courseno=in.nextInt();
                                        PreparedStatement delete =
conn.prepareStatement("delete from courses where dept_code=? and course_no=?");
                                        delete.setString(1, dept);
                                        delete.setInt(2, courseno);

                                        delete.executeUpdate();
                                        System.out.println("\nCourse deleted");
                                        break;
                  case 0:
                      flag2=false;
                      break;
                  default:
                      System.out.println("Invalid option!");
                      break;
              }
              }
              break;
        case 3:
              while(flag2 == true){

System.out.println("************************************************************");
              System.out.println("\nWhich Action you want to perform?");
              System.out.println(" 1.Display Prerequisites\n 2.Add Prerequisite\n 3.Delete
Prerequisites\n 4.Show Prerequisites info for a course\n 0.exit\n");

System.out.println("************************************************************");
              subOption=in.nextInt();
              switch(subOption){
                  case 1:
                      CallableStatement cs3 = conn.prepareCall("begin ? :=
project2procedurefunction.getprereq(); end;");
                      cs3.registerOutParameter(1, OracleTypes.CURSOR);


                      cs3.execute();
                      ResultSet rs3=(ResultSet)cs3.getObject(1);

System.out.println("************************************************************");
                      System.out.printf("dept_code\tcourse_no\t pre_dept_code\t
pre_course_no\n");

System.out.println("************************************************************");
                      while(rs3.next()){
                      System.out.println(rs3.getString(1) + "\t" + rs3.getString(2) + "\t" +
rs3.getString(3) + "\t" + rs3.getString(4));
                      System.out.println();
                      }
                      cs3.close();
                      break;
                  case 2:
                                        System.out.println("Please enter course details to
insert");
                                        System.out.println("Enter dept_code:");
```

```
                                        dept=in.next();
                                        System.out.println("Enter course_no:");
                                        courseno=in.nextInt();
                                        System.out.println("Enter pre_dept_code:");
                                        pre_dept=in.next();
                                        System.out.println("Enter pre_course_no:");
                                        pre_courseno=in.nextInt();

                                        PreparedStatement insert = 
conn.prepareStatement("insert into prerequisites values(?,?,?,?)");
                                        insert.setString(1, dept);
                                        insert.setInt(2, courseno);
                                        insert.setString(3, pre_dept);
                                        insert.setInt(4, pre_courseno);

                                        insert.executeUpdate();
                                        System.out.println("\nprerequisite Added");
                                        break;
            case 3:
                                        System.out.println("Please enter prerequisite details to 
Delete");
                                        System.out.println("Enter dept_code:");
                                        dept=in.next();
                                        System.out.println("Enter course_no:");
                                        courseno=in.nextInt();
                                        System.out.println("Enter pre_dept_code:");
                                        pre_dept=in.next();
                                        System.out.println("Enter pre_course_no:");
                                        pre_courseno=in.nextInt();
                                        PreparedStatement delete = 
conn.prepareStatement("delete from prerequisites where dept_code=? and course_no=? and 
pre_dept_code=? and pre_course_no=? ");
```

```
                                        delete.setString(1, dept);
                                        delete.setInt(2, courseno);
                                        delete.setString(3, pre_dept);
                                        delete.setInt(4, pre_courseno);

                                        delete.executeUpdate();
                                        System.out.println("\nprerequisite deleted");
                                        break;
                                case 4 :
                                        System.out.println("Please enter dept code :");
                                        dept = in.next();
                                        System.out.println("Please enter course number :");
                                        courseno = in.nextInt();
                                        CallableStatement cs9 = conn.prepareCall("begin
project2procedurefunction.j_prerequisite_info(?,?,?,?); end;");
                                        cs9.setString(1,dept);

                                        cs9.setInt(2,courseno);
                                        cs9.registerOutParameter(3, OracleTypes.CURSOR);
//REF CURSOR
                                        cs9.registerOutParameter(4, java.sql.Types.VARCHAR);
                                        cs9.execute();
                                        ResultSet rs9 = null;
                                        String mmsg9 = null;

                                        rs9 = ((OracleCallableStatement)cs9).getCursor(3);
                                        mmsg9 = ((OracleCallableStatement)cs9).getString(4);


                                        if(mmsg9 != null)
                                        {
                                                System.out.println("No Prerequisite");
                                        }
                                        else
                                        {
                                                System.out.printf("prerequisite courses\n");

System.out.println("*****************************************");
                                                while(rs9.next()){
                                                System.out.print(rs9.getString(1));
                                                System.out.println();
                                                }
                                        }
                                        cs9.close();
                                        break;
                        case 0:
                            flag2=false;
                            break;
                        default:
                            System.out.println("Invalid option!");
                            break;
                    }
                    }
                    break;
            case 4:
                    while(flag2 == true){

System.out.println("**************************************************************");
                    System.out.println("\nWhich Action you want to perform?");
                    System.out.println(" 1.Display Classes\n 2.Add Class\n 3.Delete Class\n 4.Show class
info\n 0.exit\n");

System.out.println("**************************************************************");
                    subOption=in.nextInt();
```

```
                switch(subOption){
                    case 1:
                        CallableStatement cs4 = conn.prepareCall("begin ? :=
project2procedurefunction.getclasses(); end;");
                        cs4.registerOutParameter(1, OracleTypes.CURSOR);


                        cs4.execute();
                        ResultSet rs4=(ResultSet)cs4.getObject(1);

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

System.out.printf("classid\tdept_code\tcourse_no\tsect_no\tyear\tsemester\tlimit\tclass_size\n");

System.out.println("************************************************************");
                        while(rs4.next()){
                        System.out.println(rs4.getString(1) + "\t" + rs4.getString(2) + "\t" +
rs4.getString(3) + "\t" + rs4.getString(4) + "\t" + rs4.getString(5) + "\t" + rs4.getString(6) + "\t" +
rs4.getString(7) + "\t" + rs4.getString(8));
                        System.out.println();
                        }
                        cs4.close();
                        break;
                    case 2:
                                    System.out.println("Please enter class details to
insert");
                                    System.out.println("Enter classid:");
                                    classid=in.next();
                                    System.out.println("Enter dept_code:");
                                    dept=in.next();
                                    System.out.println("Enter course_no:");
                                    courseno=in.nextInt();
                                    System.out.println("Enter sect_no:");
                                    section=in.nextInt();
```

```
                                                System.out.println("Enter year:");
                                                year=in.nextInt();
                                                System.out.println("Enter semester:");
                                                sem=in.next();
                                                System.out.println("Enter limit:");
                                                limit=in.nextInt();
                                                System.out.println("Enter class_size:");
                                                class_size=in.nextInt();

                                                PreparedStatement insert = 
conn.prepareStatement("insert into classes values(?,?,?,?,?,?,?,?)");
                                                insert.setString(1, classid);
                                                insert.setString(2, dept);
                                                insert.setInt(3, courseno);
                                                insert.setInt(4, section);
                                                insert.setInt(5, year);
                                                insert.setString(6, sem);
                                                insert.setInt(7, limit);
                                                insert.setInt(8, class_size);

                                                insert.executeUpdate();
                                                System.out.println("\nClass Added");
                                                break;
                    case 3:
                                                System.out.println("Please enter class details to 
Delete");
                                                System.out.println("Enter classid:");
                                                classid=in.next();

                                                PreparedStatement delete = 
conn.prepareStatement("delete from classes where classid=?");
                                                delete.setString(1, classid);
```

```
                                        delete.executeUpdate();
                                        System.out.println("\nclass deleted");
                                        break;
                                case 4 :
                                        System.out.println("Please enter classid :");
                                        classid = in.next();
                                        CallableStatement cs10 = conn.prepareCall("begin
project2procedurefunction.j_class_info(?,?,?); end;");
                                        cs10.setString(1, classid);
                                        cs10.registerOutParameter(2, java.sql.Types.VARCHAR);
                                        cs10.registerOutParameter(3, OracleTypes.CURSOR);
//REF CURSOR
                                        cs10.execute();
                                        ResultSet rs10 = null;
                                        String mmsg10 = null;

                                        rs10 = ((OracleCallableStatement)cs10).getCursor(3);
                                        mmsg10 = ((OracleCallableStatement)cs10).getString(2);

                                        if(mmsg10 != null)
                                        {
                                            System.out.println(mmsg10);
                                        }
                                        else if(rs10 !=null)
                                        {

System.out.printf("classid\ttitle\tsemester\tyear\tsid\tfirstname\tlastname\n");

System.out.println("************************************************************");
                                                while(rs10.next()){
                                                System.out.print(rs10.getString(1) + "\t" +
rs10.getString(2) + "\t" + rs10.getString(3) +"\t"+ rs10.getString(4) + "\t" + rs10.getString(5) + "\t" +
rs10.getString(6)+ "\t" + rs10.getString(7));
                                                System.out.println();
                                                }
                                        }
                                        cs10.close();
                                        break;
                case 0:
                    flag2=false;
                    break;
                default:
                    System.out.println("Invalid option!");
                    break;
            }
            }
            break;
        case 5:
            while(flag2 == true){
System.out.println("*************************************************************");
            System.out.println("\nWhich Action you want to perform?");
            System.out.println(" 1.Display Enrollments\n 2.Enroll student\n 3.Drop student from
class\n 0.exit\n");
System.out.println("*************************************************************");
            subOption=in.nextInt();
            switch(subOption){
                case 1:
                    CallableStatement cs5 = conn.prepareCall("begin ? :=
project2procedurefunction.getenrollment(); end;");
                    cs5.registerOutParameter(1, OracleTypes.CURSOR);
                    cs5.execute();
                    ResultSet rs5=(ResultSet)cs5.getObject(1);

System.out.println("************************************************************");
                    System.out.printf("sid\tclassid\tlgrade\n");

System.out.println("************************************************************");
                    while(rs5.next()){
                    System.out.println(rs5.getString(1) + "\t" + rs5.getString(2) + "\t" +
rs5.getString(3));
                    System.out.println();
                    }
                    cs5.close();
                    break;
                case 2:
                                        System.out.println("Please enter sid :");
                                        sid = in.next();
                                        System.out.println("Please enter classid :");
                                        classid = in.next();
                                        CallableStatement cs11 = conn.prepareCall("begin
project2procedurefunction.j_enrollment(?,?,?); end;");
                                        cs11.setString(1,sid);
                                        cs11.setString(2,classid);
                                        cs11.registerOutParameter(3, java.sql.Types.VARCHAR);
                                        cs11.execute();

                                        String mmsg3 = null;

                                        mmsg3 = ((OracleCallableStatement)cs11).getString(3);

                                        if(mmsg3 != null)
                                        {
```

```
                                System.out.println(mmsg3);
                        }
                        else
                        {
                                System.out.println("Student Enroll
suceesfully!");
                        }
                        cs11.close();
                        break;
        case 3:
                        System.out.println("Please enter sid :");
                        sid = in.next();
                        System.out.println("Please enter classid :");
                        classid = in.next();
                        CallableStatement cs12 = conn.prepareCall("begin
project2procedurefunction.j_drop_student_from_class(?,?,?,?,?); end;");
                        cs12.setString(1,sid);
                        cs12.setString(2,classid);
                        cs12.registerOutParameter(3, java.sql.Types.VARCHAR);
                        cs12.registerOutParameter(4, java.sql.Types.VARCHAR);
                        cs12.registerOutParameter(5, java.sql.Types.VARCHAR);
                        cs12.execute();

                        String msg1 = null;
                        String msg2 = null;
                        String msg3 = null;

                        msg1 = ((OracleCallableStatement)cs12).getString(3);
                        msg2 = ((OracleCallableStatement)cs12).getString(4);
                        msg3 = ((OracleCallableStatement)cs12).getString(5);

                        if(msg1 != null)
```

```
                                        {
                                                System.out.println(msg1);
                                        }
                                        else
                                        {
                                                System.out.println("Student drop suceesfully
from class!");
                                                if(msg2 != null)
                                                {
                                                                System.out.println(msg2);
                                                }
                                                if(msg3 != null)
                                                {
                                                                System.out.println(msg3);
                                                }
                                        }
                                        cs12.close();
                                        break;
                case 0:
                    flag2=false;
                    break;
                default:
                    System.out.println("Invalid option!");
                    break;
            }
            }
            break;


        case 6:
            while(flag2 == true){

System.out.println("************************************************************");
```

```
                System.out.println("\nWhich Action you want to perform?");

                System.out.println(" 1.Display Logs\n 2.Delete log\n 0. exit\n");

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

                subOption=in.nextInt();

                switch(subOption){

                    case 1:

                        CallableStatement cs6 = conn.prepareCall("begin ? :=
project2procedurefunction.getlogs(); end;");

                        cs6.registerOutParameter(1, OracleTypes.CURSOR);


                        cs6.execute();

                        ResultSet rs6=(ResultSet)cs6.getObject(1);

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

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

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

                        while(rs6.next()){

                        System.out.println(rs6.getString(1) + "\t" + rs6.getString(2) + "\t" +
rs6.getString(3) + "\t" + rs6.getString(4) + "\t" + rs6.getString(5) + "\t" + rs6.getString(6));

                        System.out.println();

                        }

                        cs6.close();

                        break;

                    case 2:

                                    System.out.println("Please enter log details to Delete");

                                    System.out.println("Enter logid:");

                                    logid=in.nextInt();

                                    PreparedStatement delete =
conn.prepareStatement("delete from logs where logid=?");

                                    delete.setInt(1, logid);
```

```
                                                  delete.executeUpdate();
                                                  System.out.println("\nlog deleted");
                                                  break;
                        case 0:
                            flag2=false;
                            break;
                        default:
                            System.out.println("Invalid option!");
                            break;
                    }
                    }
                    break;
            case 0:
                flag1=false;
                break;
            default:
                        System.out.println("Invalid option!");
                        break;
            }
        }

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

```
finally{

        if(conn != null)

        {

                try{

                        conn.close();

                }

                catch(Exception e){

                        e.printStackTrace();

                }

        }

}


}


}
```

Database Systems Project 2 - Student Registration System

This project has been done by:

Priyanka Prakash Tanpure.

B00821027

This report entails summary of all the tools used, PL SQL code and how to execute this code.

Tools:

Database used: Oracle 12c

Notepad++ : To write java code

CMD: To execute code

remote.cs: to execute code, for demo

PL SQL Objects:

Package:

1. project2procedurefunction: It contains all the functions and procedures created in this project

Procedures:

1. show_students: It displays all the tuples present in students table when you execute the procedure.
2. show_courses: It displays all the tuples present in courses table when you execute the procedure.
3. show_prerequisites: It simply displays all the tuples present in prerequisites table when you execute the procedure.
4. show_classes: It displays all the tuples present in classes table when you execute the procedure.
5. show_enrollments: It displays all the tuples present in enrollments table when you execute the procedure.

6. show_logs: It displays all the tuples present in logs table when you execute the procedure.
7. procedure add_student(student_id in char,first_name in varchar2,last_name in varchar2,student_status in varchar2,student_gpa in number, student_email in varchar2) : Procedure to add student into students table
8. procedure studentclass_info(student_id in char) : procedure to display the student details corresponding to student_id(i.e sid) entered and also student's class info.
9. procedure prerequisite_info(departmentcode in varchar2,coursenumber in number): It displays all the direct and indirect prerequisite courses of the course you provide.
10. procedure class_info(s_classid in char) : displays class info and also the details of students enrolled in this class.
11. procedure enrollment(student_id in char,c_classid in char): It is to enroll student in class where sid and classid are given as input and if it matches all the required criteria then student is successfully enrolled into the class otherwise the failing criteria i.e message is display. On successful enrollment, class size is increased, and entry is inserted in logs table.
12. procedure drop_student_from_class(student_id in char,c_classid in char): It is to remove student from a class where sid and classid are taken as input and if all the checks matches, student is successfully removed otherwise error message is displayed. On successful deletion, class size is decreased, and entry is inserted in logs table.
13. procedure delete_student(student_id in char): It is to delete a student from a system where sid is provided as input. Which in turn delete the entry from enrollments table and entry inserted in log table.

Functions and procedures created to use them in java/jdbc code. (Which send ref cursor and a show_message varchar as and when required to display output from java program)

1. function getstudents return refcur: It is a function is same as show_students procedure which will return students tuple and are displayed as output in java program.

2. function getcourses return refcur: It is a function is same as show_courses procedure which will return courses tuple and are displayed as output in java program.
3. function getenrollment return refcur: It is a function same as show_enrollments procedure which will return enrollments tuple and are displayed as output in java program.
4. function getclasses return refcur: It is a function same as show_classes procedure which will return classes tuple and are displayed as output in java program.
5. function getprereq return refcur: It is a function same as show_prerequisites procedure which will return prerequisites tuple and are displayed as output in java program.
6. function getlogs return refcur: It is a same as show_logs procedure which will return logs tuple and are displayed as output in java program.
7. procedure j_studentclass_info(student_id in char, show_message OUT varchar2, sc_recordset OUT SYS_REFCURSOR) : It is a same as studentclass_info procedure which will return logs tuple and are displayed as output in java program.
8. procedure j_prerequisite_info(departmentcode in varchar2,coursenumber in number, p_recordset OUT SYS_REFCURSOR,show_message OUT varchar2) : It is a same as prerequisite_info procedure which will return logs tuple and are displayed as output in java program.
9. procedure j_class_info(s_classid in char, show_message OUT varchar2, c_recordset OUT SYS_REFCURSOR) : It is a same as class_info procedure which will return logs tuple and are displayed as output in java program.
10. procedure j_enrollment(student_id in char,c_classid in char, show_message OUT varchar2) : It is a same as enrollment procedure which will return logs tuple and are displayed as output in java program.
11. procedure j_drop_student_from_class(student_id in char,c_classid in char, show_message1 OUT varchar2, show_message2 OUT varchar2, show_message3 OUT varchar2) : It is a same as drop_student_from_class procedure which will return logs tuple and are displayed as output in java program.

12. procedure j_delete_student(student_id in char, show_message OUT varchar2) : It is a same as delete_student procedure which will return logs tuple and are displayed as output in java program.

<u>Sequence:</u>

1. log_number: It is 7 digit id for auto-increment and auto – insert in logs table when any new entry is made to the table.

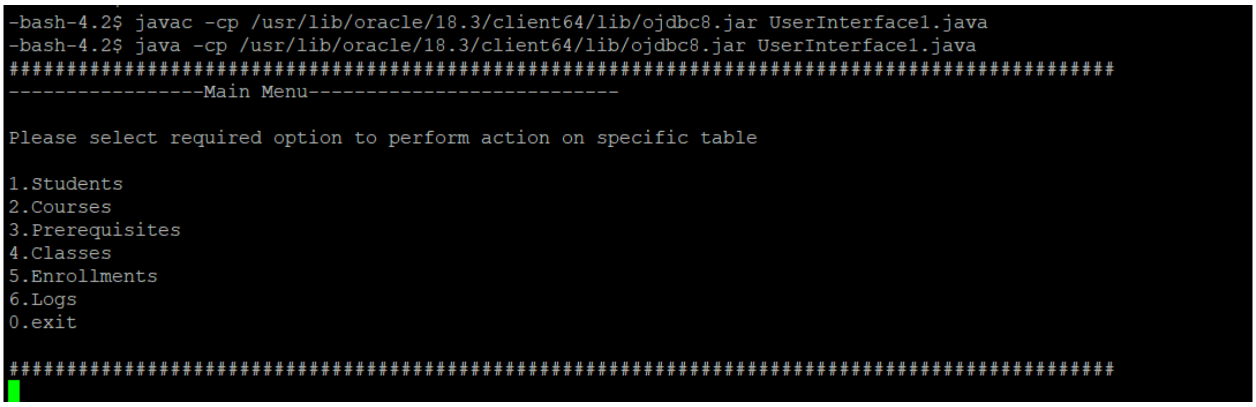
<u>Triggers:</u>

1. t_after_add_student : entry inserted into log table.
2. t_after_enroll : class_size increses by 1 and entry inserted into log table.
3. t_before_delete_student : entry deleted from enrollments tabe.
4. log_entry_after_delete_stud : entry inserted into log table.
5. t_after_delete_enroll: decreases class size by 1 and insert entry into log table.
6. t_before_delete_class : delete entry from enrollments table

<u>How to run code:</u>

1. Open remote.cs. Go to folder where file Try.java (source code) is kept.
2. Change unsername and password in UserInterface1.java file.
3. javac -cp /usr/lib/oracle/18.3/client64/lib/ojdbc8.jar UserInterface1.java -- compile java file
4. java -cp /usr/lib/oracle/18.3/client64/lib/ojdbc8.jar UserInterface1.java -- execute code

After above commands, on correct authentication, a text-based menu driven program will be returned which will take user input and return output accordingly. 1-6 option contains sub menu also. It is a continuous loop unless sql exception is thrown or user choose to exit.



```
-bash-4.2$ javac -cp /usr/lib/oracle/18.3/client64/lib/ojdbc8.jar UserInterface1.java
-bash-4.2$ java -cp /usr/lib/oracle/18.3/client64/lib/ojdbc8.jar UserInterface1.java
#################################################################################################
-----------------Main Menu---------------------------

Please select required option to perform action on specific table

1.Students
2.Courses
3.Prerequisites
4.Classes
5.Enrollments
6.Logs
0.exit

#################################################################################################
```