1. Use a sequence to generate the values for log# automatically when new log records are inserted into the logs table. Start the sequence with 100 with an increment of 1.

```
CREATE sequence log#_value
START WITH 100
INCREMENT BY 1;
```

2.
```
CREATE or REPLACE PROCEDURE show_students(pro OUT sys_refcursor) AS
BEGIN
OPEN pro for SELECT * FROM students;
END;
/

CREATE or REPLACE PROCEDURE show_courses(pro OUT sys_refcursor) AS
BEGIN
OPEN pro for SELECT * FROM courses;
END;
/

CREATE or REPLACE PROCEDURE show_classes(pro OUT sys_refcursor) AS
BEGIN
OPEN pro for SELECT * FROM classes;
END;
/

CREATE or REPLACE PROCEDURE show_enrollments(pro OUT sys_refcursor) AS
BEGIN
OPEN pro for SELECT * FROM enrollments;
END;
/

CREATE or REPLACE PROCEDURE show_logs(pro OUT sys_refcursor) AS
BEGIN
```

```
OPEN pro for SELECT * FROM logs;
END;
/
```

3. CREATE or REPLACE PROCEDURE enroll_into_class(B IN enrollments.B#%type,
class_num IN enrollments.cl$
constraint_classid number;
constraint_B# number;
constraint_classize number;
constraint_enrollment number;
constraint_overload number;
limitting number;
education_year classes.year%type;
education_semester classes.semester%type;

```
BEGIN
SELECT COUNT(B#) into constraint_B# FROM students WHERE B# in (B);
IF  constraint_B#=0 THEN
      DBMS_OUTPUT.PUT_LINE('THE B# IS INVALID!');
      result:=0;
      return;
END IF;

SELECT COUNT(classid) into constraint_classid FROM classes  WHERE classid in
(class_num);
IF constraint_classid=0 THEN
      DBMS_OUTPUT.PUT_LINE('THE CLASS ID IS INVALID!');
```

```
        result:=1;
        return;
END IF;

SELECT year into education_year FROM classes WHERE classid=class_num;
SELECT semester into education_semester FROM classes WHERE classid=class_num;
IF (education_year<>2020 OR education_semester<>'Spring') THEN
        DBMS_OUTPUT.PUT_LINE('CANNOT ENROLL INTO A CLASS FROM A PREVIOUS
SEMESTER!');
        result:=2;
        return;
END IF;

SELECT limit into limitting FROM classes WHERE classid=class_num;
SELECT class_size into constraint_classize FROM classes WHERE classid=class_num;
IF (limitting=constraint_classize)  THEN
        DBMS_OUTPUT.PUT_LINE('THE CLASS IS ALREADY FULL');
        result:=3;
        return;
END IF;

SELECT COUNT(*) into constraint_enrollment FROM enrollments WHERE B#=B AND
classid=class_num;
IF constraint_enrollment=1 THEN
        DBMS_OUTPUT.PUT_LINE('THE STUDENT IS ALREADY IN THE CLASS');
        result:=4;
        return;
END IF;
```

```
SELECT COUNT(B#) into constraint_overload FROM enrollments e, classes c WHERE B#=B
AND c.year=2020 $
c.semester='Spring' AND e.classid=c.classid;
IF constraint_overload=5 THEN
      DBMS_OUTPUT.PUT_LINE('STUDENTS CANNOT ENROLL INTO MORE THAN FOUR
COURSES IN THE SAME SEMEST$
      result:=5;
      return;
ELSE

            INSERT into enrollments VALUES (B, class_num, null);

END IF;
END enroll_into_class;
/
```

4.
```
CREATE or REPLACE PROCEDURE drop_from_class(B IN students.B#%type,
    class_num IN classes.classid%typ$
  result1 OUT number) IS
        constraint_classid number;
        constraint_B# number;
        constraint_enrollment number;
        constraint_offering number;
        not_enrolled number;
        no_students number;

BEGIN
```

```
SELECT COUNT(*) into constraint_B# FROM students s WHERE s.B#=B;
SELECT COUNT(*) into constraint_classid FROM classes c WHERE class_num=c.classid;
SELECT COUNT(*) into constraint_enrollment FROM enrollments e WHERE B=e.B#;
SELECT COUNT(*) into constraint_offering FROM classes c WHERE class_num=c.classid AND
year=2020 AND
semester='Spring';

IF (constraint_B#=0) THEN
DBMS_OUTPUT.PUT_LINE('THE B# IS INVALID.');
result1:=0;

ELSIF (constraint_classid=0) THEN
DBMS_OUTPUT.PUT_LINE('THE CLASS ID IS INVALID.');
result1:= 1;

ELSIF (constraint_enrollment=0 AND constraint_B#=1) THEN
DBMS_OUTPUT.PUT_LINE('THE STUDENT IS NOT ENROLLED IN THE CLASS.');
result1:= 2;

ELSIF (constraint_offering=0) THEN
DBMS_OUTPUT.PUT_LINE('ONLY ENROLLMENT IN THE CURRENT SEMESTER CAN BE
DROPPED.');
result1:= 3;

ELSE

DELETE FROM enrollments WHERE B# in (B) AND classid in (class_num);
SELECT COUNT(classid) into not_enrolled FROM enrollments WHERE B#=B;
SELECT COUNT(B#) into no_students FROM enrollments WHERE classid=class_num;
END IF;
```

```
IF (not_enrolled=0) THEN
DBMS_OUTPUT.PUT_LINE('THIS STUDENT IS NOT ENROLLED IN ANY CLASSES.');
result1:= 4;
END IF;

IF (no_students=0) THEN
DBMS_OUTPUT.PUT_LINE('THE CLASS NOW HAS NO STUDENTS.');
result1:= 5;
END IF;



END;
/
```

5.

```
/* Update the log table whenever a insert is done on student table*/

    create or replace trigger insert_student_logs
    after insert on Students
    for each row
    declare
    name varchar2(20);
    begin
    select user into name from dual;
    insert into Logs values
    (log#_seq.nextval,name,sysdate,'Students','insert',:new.B#);
```

```
end;
/
show errors



/* Update the log table whenever a delete is done on Student*/

create or replace trigger delete_student_logs
after delete on Students
for each row
declare
name varchar2(20);
begin
select user into name from dual;
insert into Logs values
(log#_seq.nextval,name,sysdate,'Students','delete',:old.B#);
end;
/
show errors


/* Update the log table whenever a delete is done on enrollment*/
create or replace trigger delete_enrollment_logs
after delete on Enrollments
for each row
declare
name varchar2(20);
begin
select user into name from dual;
insert into Logs values
```

```
(log#_seq.nextval,name,sysdate,'Enrollments','delete',:old.B#||','||:old.classid);
end;
/
show errors

/* Update the log table whenever a insert is done on enrollment*/

create or replace trigger insert_enrollment_logs
after insert on Enrollments
for each row
declare
name varchar2(20);
begin
select user into name from dual;
insert into Logs values
(log#_seq.nextval,name,sysdate,'Enrollments','insert',:new.B#||','||:new.classid);
end;
/
show errors
```