-- ASSIGNMENT --

-- --
-- DATABASE SCHEMA :
-- --
-- depts([deptcode], deptname)
-- students([rollno], name, bdate, {deptcode}, hostel, parent_inc)
-- faculty([fac_code], fac_name, {fac_dept})
-- crs_offrd([crs_code], crs_name, crs_credits, {crs_fac_cd})
-- crs_regd([{crs_rollno}, {crs_cd}], marks)
-- --

-- SET 1 --

-- Q1> Delete records from dept where deptcode='CSE'.
--   (This deletes records from students whose deptcode='CSE')

```
delete from students where deptcode='CSE';
```

-- Q2> Find out the courses offered by the faculty dbp and nls.

```
select crs_name
from crs_offrd
where crs_fac_cd = (select fac_code from faculty where fac_name = 'DBP')
   or crs_fac_cd = (select fac_code from faculty where fac_name = 'NLS');
```

-- Q3> Find out the courses with full details offered by dbp.

```
select * from crs_offrd
where crs_fac_cd = (select fac_code from faculty where fac_name = 'DBP');
```

-- Q4> Get the courses the credits of which lies between 4.0 and 6.0.

```
select crs_name from crs_offrd
where crs_credits between 4.0 and 6.0;
```

-- Q5> Get the courses the credits of which are > 6.5.

```
select crs_name from crs_offrd
where crs_credits > 6.5;
```

-- SET 2 --

-- Q1> Count the number of students in CSE dept.

```
select count(rollno)
from students
where deptcode ='CSE';
```

-- Q2> Determine the minimum, maximum and average marks of each courses.

```
select crs_cd, min(marks), max(marks), avg(marks)
from crs_regd
group by crs_cd;
```

-- Q3> Determine the total credits of the courses registered by a student.

```
select crs_rollno, sum(crs_credits)
from crs_regd, crs_offrd
where crs_regd.crs_cd = crs_offrd.crs_code
group by crs_rollno;
```

-- Q4> Count the number of students in each hostel whose department is CSE.

```
select hostel, count(rollno)
from  students
where deptcode = 'CSE'
group by hostel;
```

-- Q5> Display the hostel, rollno, parent_inc of the student who has the max(parent_inc) in a hostel.

```
select hostel, rollno, parent_inc
from students
where parent_inc
   in (select max(parent_inc)
      from students
      group by hostel);
```

-- Q6> Display the name and parental income of each student greater than the parental income of some rollno 92005010.

```
select name, parent_inc
from students
```

```
where parent_inc > (select parent_inc
                    from students
                    where rollno = 92005010);

-- Q7> Find out marks of students who have marks more than rollno 92005102 for course
CH103 and PH106.

select crs_rollno,marks
from crs_regd
where ((marks > (select marks
                 from crs_regd
                 where crs_rollno = 92005102 and crs_cd = 'CH103')) and crs_cd = 'CH103')
      or
      ((marks > (select marks
                   from crs_regd
                   where crs_rollno = 92005102 and crs_cd = 'PH106')) and crs_cd = 'PH106');

-- SET 3 --

-- Q1> List students (rollno,name,deptcode) registered for course EE101.

select rollno, name, deptcode
from students,crs_regd
where crs_cd= 'EE101' and rollno = crs_rollno;

-- Q2> List students (rollno,name) in ELE dept registered for course EE101.

select rollno, name
from students, crs_regd
where rollno = crs_rollno
    and deptcode = 'ELE'
    and crs_cd = 'EE101';

-- Q3> List students (rollno,name) in ELE dept not registered for course EE101.

select rollno, name
from students,crs_regd
where deptcode = 'ELE'
minus
select rollno, name
from students, crs_regd
where rollno = crs_rollno
      and deptcode = 'ELE'
```

```
        and crs_cd = 'EE101';

-- Q4> List the names of the students who have registered for both the courses 'DBMS'and 'OS'.

select name
from students, crs_regd
where rollno = crs_rollno
     and crs_cd = 'DBMS'
intersect
select name
from students, crs_regd
where rollno = crs_rollno
     and crs_cd = 'OS';

-- Q5> Find the names of the faculty members who have offered either 'MIS' or 'Software Engg.'

select fac_name from faculty,crs_offrd
where fac_code = crs_fac_cd
and (crs_code = (select crs_code from crs_offrd where crs_name = 'Information Systems')
   or crs_code = (select crs_code from crs_offrd where crs_name = 'Software'));

-- Q6> Find the names of the faculty members who have offered 'MIS' but not offered 'Software
Engg.'

select fac_name from faculty,crs_offrd
where fac_code = crs_fac_cd
and (crs_code = (select crs_code from crs_offrd where crs_name = 'Information Systems')
   or crs_code = (select crs_code from crs_offrd where crs_name = 'Software'))
minus
select fac_name from faculty,crs_offrd
where fac_code = crs_fac_cd
and (crs_code = (select crs_code from crs_offrd where crs_name = 'Software'));

-- Q7> Find out the students in each hostel who are not registered for any course

select name, hostel
from students
minus
select name, hostel
from students, crs_regd
where rollno = crs_rollno;

-- Q8> Select the students who are in ELE dept or who have registered for course CS101.
```

```
select name
from students
where deptcode = 'ELE'
union
select name
from students, crs_regd
where rollno = crs_rollno
      and crs_cd = 'CS101';

-- Q9> Display the students who have registered to all the courses.

select crs_rollno
from students, crs_regd
where rollno = crs_rollno
group by crs_rollno
having count(crs_cd) = (select count(crs_cd)
                        from crs_offrd);

-- Q10> Give Grace Marks 5 in subject DBMSto the students who have scored less than 50 in
that subject.

update crs_regd
set marks = marks + 5
where crs_cd = 'DBMS' and marks < 50;

-- SET 4 --

-- Q1> Retrieve the name of the students whose name starts with ‘S’ and contains ‘r’
--      as the second last character.

select name
from students
where name like 'S%r_';

-- Q2> Retrieve the name of the youngest student(s) from the ‘CST’ department
--      along with the total marks obtained by him (them).

select s.name, sum(c.marks) as marks
from students s, crs_regd c,
    (select min(s1.bdate) as bdate
     from students s1, depts d
     where s1.deptcode = d.deptcode and d.deptname = 'Computer Science'
```

```
   ) x
where s.rollno = c.crs_rollno
     and x.bdate = s.bdate
group by s.name, s.bdate;

-- Q3> Find the age of all the students.

select name, floor(months_between(sysdate, bdate)/12) as Age
from students;

-- SET 5 --

-- Q1> Retrieve the name of the student(s) who obtained second highest marks in ‘DBMS’.

select distinct (name), rollno, marks
from students, crs_regd
where rollno = crs_rollno
     and marks = (select max(marks)
                  from crs_regd
                  where crs_cd = 'DBMS'
                       and marks < (select max(marks)
                                    from crs_regd
                                    where crs_cd = 'DBMS'));

-- Q2> Find out the differences between highest and lowest marks obtained in each subject.

select offr.crs_name, max(reg.marks) - min(reg.marks) difference
from crs_regd reg, crs_offrd offr
where reg.crs_cd = offr.crs_code
group by offr.crs_name;

-- Q3> Assuming the existance of several interdepartmental courses, retrieve the
--    name of the student(s) who is(are) studying under at least one faculty from
--    each department.

select name from students
minus
select name from students
where (select count(*) from depts
      where depts.deptcode = students.deptcode) = (select count(distinct(fac_dept))
                                                   from (select fac_dept
                                                         from fac,crs_regd,crs_offrd
                                                         where rollno = crs_rollno
```

```
                                        and crs_reg_code = crs_cd
                                        and crs_fac_cd = fac_cd));

-- Q4> Assuming the existance of several interdepartmental courses, retrieve the
--    name of the student(s) who is(are) studying under the faculties only from
--    his(their) own department.

select name
from students
where (select count(*)
      from depts
      where depts.deptcode = students.deptcode)= (select count(distinct(fac_dept))
                                    from (select fac_dept
                                        from fac, crs_regd, crs_offrd
                                        where rollno = crs_rollno
                                            and crs_cd = crs_code
                                            and crs_fac_cd = fac_code));

-- SET 6 --

-- Q1> Display highest parent incomes in descending order, for each department,
--     excluding 'ARCH' such that only those highest parent incomes will appear
--     that are below 12,000

select x.deptname, x.p_inc
from (select d.deptname, max(s.parent_inc) as p_inc
     from students s, depts d
     where d.deptcode = s.deptcode
     group by d.deptname) x
where x.deptname <> 'Architecture'
     and x.p_inc < 12000
order by x.p_inc desc;

-- Q2> Retrieve the fifth highest parent income for hostel number 5.

select s1.parent_inc
from students s1
where (4) = (select count(distinct (parent_inc))
           from students s2
           where s2.parent_inc > s1.parent_inc
                and s2.hostel = s1.hostel
                and s1.hostel = 5);
```

```
-- Q3> Find the roll number of the students from each department who obtained
--     highest total marks in their own department.

select B.crs_rollno, A.deptcode, A.max_marks
from (select deptcode, max_marks
      from (select deptcode, max(total_marks) as max_marks
            from (select crs_rollno, sum(marks) as total_marks
                  from crs_regd
                  group by crs_rollno), students
            where rollno = crs_rollno
            group by deptcode)) A,
    (select crs_rollno, deptcode, max(total_marks) as max_marks
     from (select crs_rollno, sum(marks) as total_marks
           from crs_regd
           group by crs_rollno), students
     where rollno = crs_rollno
     group by crs_rollno, deptcode) B
where A.max_marks = B.max_marks
      and A.deptcode = B.deptcode;

-- SET 7 --

-- Creating departments table

    create table depts(
                deptcode varchar(3) PRIMARY KEY,
                deptname varchar(50) NOT NULL);

-- Creating students table

    create table students(
                rollno number(9) PRIMARY KEY,
                name varchar(30),
                deptcode varchar(3) REFERENCES DEPTS(deptcode) ON DELETE CASCADE,
                bdate date NOT NULL,
                hostel number CHECK(hostel < 17),
                parent_income number(5));

-- Inserting departments

    insert into depts
    values('CSE', 'Computer Science and Engineering');
```

```
insert into depts
values('ETC', 'Electronics and Telecommunication Engineering');

insert into depts
values('ME', 'Mechanical Engineering');

insert into depts
values('EE', 'Electrical Engineering');

-- Inserting students

insert into students
values('510517080', 'Mainak Basu', 'CSE', '14-JUL-1999', '1', '20000');

insert into students
values('510517086', 'Anjishnu Mukherjee', 'CSE', '2-NOV-1998', '9', '25000');

insert into students
values('510517087', 'Sagnik Acharya', 'CSE', '2-NOV-2000' ,'6', '18000');

insert into students
values('510517006', 'Anthony Rajiv', 'CSE', '2-MAR-1998', '4', '13000');

insert into students
values('510517004', 'Sourav Gaikwad', 'CSE', '1-JAN-2000', '8', '22000');

insert into students
values('510517001', 'Akash Singh', 'CSE', '2-NOV-1998', '11', '23000');

insert into students
values('510417021', 'Soham Das', 'ME', '2-FEB-1999', '7', '15000');

insert into students
values('510417018', 'Sanchari Saha', 'ME', '2-OCT-1997', '2', '26000');

insert into students
values('510417019', 'Kushal Paul', 'ME', '2-SEP-1998', '6', '17000');

insert into students
values('510417050', 'Anuradha Bhattacharya', 'ME', '2-DEC-1998', '10', '19000');

insert into students
values('510617001', 'Swastika Dutta', 'ME', '2-NOV-1998', '14', '10000');
```

```
    insert into students
    values('510417010', 'Shreyan Ghosh', 'ME', '14-AUG-1999', '13', '10000');

    insert into students
    values ('510317011', 'Pramit Das', 'EE', '24-AUG-1999', '13', '25000');

    insert into students
    values ('510317012', 'Anindya Kundu', 'EE', '4-OCT-1999', '13', '30000');

    insert into students
    values ('510317015', 'Indranil Bit', 'EE', '17-DEC-1999', '13', '20000');

    insert into students
    values ('510217017', 'Soumyo Roy', 'ETC', '13-SEP-1999', '13', '10000');

-- Q1> Create a view of all students in dept CSE.

    create view all_students as
    select *
    from students
    where deptcode = 'CSE';

    select * from all_students;

-- Q2> Create a view named as cse_students for ‘CSE’ dept students
--     having attributes rollno, name, hostel

    create view cse_students as
    select rollno, name, hostel
    from students
    where deptcode = 'CSE';

    select * from cse_students;

-- Q3> Insert a new student of CSE. Analyse the result.

    -- Value inserted into students table but not in cst_stud view
    -- since deptcode, hostel, parent_income and bdate are null
    select * from all_students;
    select * from students where deptcode = 'CSE';

    insert into all_students
```

```
    values('510517090', 'Pragati Gupta', 'CSE', '26-DEC-1996', '16', '21000');

    select * from all_students;
    select * from students where deptcode = 'CSE';

    -- DOESN'T WORK. CAN'T INSERT NULL.
    insert into cse_students
    values('510517090', 'Arijit Singha', '16');

    select * from cse_students;
    select * from students where deptcode = 'CSE';

-- Q4> Increment parental income by Rs. 5000 (HRA).

    select * from all_students;
    select * from students where deptcode = 'CSE';

    update all_students
    set parent_income = parent_income + 5000;

    select * from all_students;
    select * from students where deptcode = 'CSE';

-- Q5> Delete the view.

    drop view cse_students;
    drop view all_students;

-- Q6> Create another view of all students in
--     dept Mechanical Engineering (department Name).
--     The view will contain attributes namely Roll-No, Name, Department Name, Age.

    select * from students;

    create view mech_students as
    select rollno, name, deptname, floor(months_between(sysdate, bdate)/12) as age
    from students, depts
    where students.deptcode = depts.deptcode and deptname = 'Mechanical Engineering';

    select * from mech_students;

-- Q7> Attempt : Insert a new student of Mechanical Engineering Department.
--     Analyse the result.
```

```
    -- --ERROR--
    -- ORA-01776: cannot modify more than one base table through a join view
    insert into mech_students
    values('511017100', 'Kaustav Sarkar', 'Mechanical Engineering', '20');

    -- ANALYSIS
    create view test as
      select rollno, name, students.deptcode as code, depts.deptcode, deptname, bdate, hostel,
parent_income
       from students, depts
       where students.deptcode = depts.deptcode and deptname = 'Mechanical Engineering';

    insert into test
    values('511017100', 'Kaustav Sarkar', 'ME', 'ME', 'Mechanical Engineering','2-NOV-1998',
'16', '17000');

-- Q8> Attempt : Delete a student (for a given Name) of the same department
--     Analyse the result.

    select * from mech_students;

    delete from mech_students where name = 'Soham Das';

    select * from mech_students;
    select * from students;

-- Q9> Attempt : Shift a student (for a given Name) from Mechanical to Computer Science.
--     Analyse the result.

    select * from mech_students;

    -- ERROR--
    -- ORA-01779: cannot modify a column which maps to a non key-preserved table
    update mech_students
    set deptname = 'Computer Science and Engineering'
    where name = 'Shreyan Ghosh';

    select * from mech_students;
    select * from students;

    -- Delete the view.
```

```
   drop view mech_students;

-- Drop the tables.

   drop table depts cascade constraints;
   drop table students cascade constraints;

-- PLSQL --

-- Q1. Write a pl/sql block for the following:
-- Insert data into a table containing two attributes namely radius & circumference of circles.
-- You may get different values of radius either from keyboard or you may generate different
-- values.

DROP TABLE circle;
CREATE TABLE circle(radius INTEGER,circumference NUMBER(8,2));

DECLARE
   i INTEGER;
   pi CONSTANT DOUBLE PRECISION:=3.14159;
   circumference circle.circumference%TYPE;

BEGIN
   FOR i in 10 .. 20 LOOP
        --radius:=i;
        circumference:=2*pi*i;
        INSERT INTO circle (radius,circumference) VALUES (i,circumference);
   END LOOP;
END;
/

SELECT * FROM circle;

-- Q2. Write a pl/sql block for the following:
-- Update the balance of each customer from a cust_acct table showing withdrawal of Rs.1000/-
-- as service charge provided that the customer balance shows at least Rs.1000/-.

DROP TABLE customer_account;
CREATE TABLE customer_account(customer_id INTEGER PRIMARY KEY, balance
NUMBER(10,2));
INSERT INTO customer_account(customer_id,balance) VALUES (1,100);
INSERT INTO customer_account(customer_id,balance) VALUES (2,20000);
INSERT INTO customer_account(customer_id,balance) VALUES (3,700);
```

```
INSERT INTO customer_account(customer_id,balance) VALUES (4,3000);
INSERT INTO customer_account(customer_id,balance) VALUES (5,1000);
INSERT INTO customer_account(customer_id,balance) VALUES (6,1700);
INSERT INTO customer_account(customer_id,balance) VALUES (7,23000);
INSERT INTO customer_account(customer_id,balance) VALUES (8,100);

SELECT * FROM customer_account;

BEGIN
   UPDATE customer_account
   SET balance = balance-1000
   WHERE balance >= 1000;
   dbms_output.put_line('Performing transaction!');
END;
/

SELECT * FROM customer_account;

-- Q3. Write a pl/sql block for the following:
-- Update the salary of each employee from EMP table by 15% using cursor.
DROP TABLE employee;
CREATE TABLE employee (employee_id INTEGER PRIMARY KEY, employee_name
VARCHAR(20), salary NUMBER(10,2));
INSERT INTO employee (employee_id, employee_name, salary) VALUES (1, 'Souparno',
500000);
INSERT INTO employee (employee_id, employee_name, salary) VALUES (2, 'Anjishnu',
40000);
INSERT INTO employee (employee_id, employee_name, salary) VALUES (3, 'Anindya',
200000);
INSERT INTO employee (employee_id, employee_name, salary) VALUES (4, 'Arijit', 70000);
INSERT INTO employee (employee_id, employee_name, salary) VALUES (5, 'Kinjal', 30000);
INSERT INTO employee (employee_id, employee_name, salary) VALUES (6, 'Koustav',
30000);
INSERT INTO employee (employee_id, employee_name, salary) VALUES (7, 'Kishan', 30000);
INSERT INTO employee (employee_id, employee_name, salary) VALUES (8, 'Birbal', 30000);
INSERT INTO employee (employee_id, employee_name, salary) VALUES (9, 'Harshit', 30000);
INSERT INTO employee (employee_id, employee_name, salary) VALUES (10, 'Kushal',
30000);

SELECT * FROM employee;

-- Using implicit cursor (SQL)
DECLARE
```

```
total_rows NUMBER(2);
BEGIN
   UPDATE employee
   SET salary = salary + 0.15*salary;
   IF SQL%NOTFOUND THEN
      dbms_output.put_line('No employees selected');
   ELSIF SQL%FOUND THEN
      total_rows:=SQL%ROWCOUNT;
      dbms_output.put_line( total_rows || ' employees updated!');
   END IF;
END;
/

SELECT * FROM employee;

-- Q4. Write a pl/sql block for the following:
-- Update the balance in the ITEM_MSTR table each time a transaction takes place in the
-- ITEM_TR table. If this item_id is already present in the ITEM_MSTR table an update is
-- performed to decrease the balance by the quantity specified in the ITEM_TR table.
-- If the item_id is not present in the ITEM_MSTR table, the tuple is to be inserted.

DROP TABLE item_mstr;
DROP TABLE item_tr;
CREATE TABLE item_mstr(item_id integer primary key, balance number(10,2));
CREATE TABLE item_tr(item_id integer, deduction number(10,2));
INSERT INTO item_mstr(item_id, balance) VALUES (1,3000);
INSERT INTO item_mstr(item_id, balance) VALUES (2,20000);
INSERT INTO item_mstr(item_id, balance) VALUES (3,7000);

SELECT * FROM item_mstr;

DECLARE
   id item_tr.item_id%TYPE:=&item_id;
   ded item_tr.deduction%TYPE:=&deduction;
   cnt INTEGER;
BEGIN
   SELECT count(*) INTO cnt FROM item_mstr WHERE item_id=id;
   IF cnt=0 THEN
      INSERT INTO item_mstr (item_id,balance) VALUES (id,0);
   ELSE
      UPDATE item_mstr SET balance=balance-ded;
      INSERT INTO item_tr (item_id,deduction) VALUES (id,ded);
   END IF;
```

```
END;
/

SELECT * FROM item_mstr;

-- Q5. Write a pl/sql block for the following:
-- Write a PROCEDURE for raising salary of some employee by some amount. The
-- PROCEDURE to be written may carry two parameters emp_id and amt to be raised. Include
-- two exceptions which will be raised when either emp_id is not present or salary is NULL.

INSERT INTO employee (employee_id,employee_name) VALUES (11,'Rajdeep');

SELECT * FROM employee;

CREATE OR REPLACE PROCEDURE raise_salary
(eid IN employee.employee_id%TYPE,
raise IN employee.salary%TYPE)
IS
cnt INTEGER;
sal employee.salary%TYPE;
INVALID_ID EXCEPTION;
NULL_VALUE EXCEPTION;
BEGIN
   SELECT count(*) INTO cnt FROM employee WHERE employee_id=eid;
   IF cnt=0 THEN
      RAISE INVALID_ID;
   ELSE
      SELECT salary INTO sal FROM employee WHERE employee_id=eid;
      IF sal IS NULL THEN
         RAISE NULL_VALUE;
      ELSE
         UPDATE employee SET salary = salary + raise WHERE employee_id=eid;
         dbms_output.put_line('Salary raised!');
      END IF;
   END IF;
EXCEPTION
   WHEN INVALID_ID THEN
      dbms_output.put_line('User ID does not exist!');
   WHEN NULL_VALUE THEN
      dbms_output.put_line('Salary is null in table!');
   WHEN others THEN
      dbms_output.put_line('Error!');
END;
```

```
/

EXECUTE raise_salary('1','10000');

SELECT * FROM employee;
```