Schema for Lab-5

⇒ Strong Entities

1. Employee (fname, minit, lname, bdate, address, salary, sex, ssn, super_ssn, dno)
2. department (dname, dnumber, mgr_ssn, start_date)
3. dept_locations (dnumber, dlocation)
4. project (pname, pnumber, plocation, dno)

Weak Entities

5. dependent (ssn, dependent_name, sex, birthdate, relationship).

Relationship Sets

→ supervision (super_ssn, ssn)

(many-one)

→ works_for (ssn, dname, dno)

(many-one)

→ manages (mgr_ssn, dname, dno, start_date)

(one-one)

→ works_on (ssn, pname, pno, hours).
} (many - many)
Controls (pname, pno, dname, dno)
} (many - one)

⇒ After merging we would get it in the merged table format.

(1) supervision get merged into employee and adds Super_ssn.

(2) works_for merges into employee and 'dno' gets added

(3) manages is merged into department and 'mgr_ssn' and start_date' are added

(4) works_on is not merged because it is many many relation

(5) controls is merges into project where dno is added

LAB-5

Schema?

1)
```
SELECT Bdate, Address
FROM employee
WHERE fname = 'John' AND minit = 'B' AND lname = 'Smith';

SELECT fname, minit, lname, address, dname
FROM department d, employee e
WHERE d.mgr_ssn = e.ssn AND dname = 'Research';
```

2)
```
SELECT fname, minit, lname, address
FROM employee e, department d
WHERE e.dno = d.dnumber and dname = 'Research';
```

3)
```
SELECT distinct (Salary)
FROM employee;
```

4)
```
Select e1.fname, e1.lname, e2.fname managers
FROM employee e1, employee e2
WHERE e1.super_ssn = e2.ssn;
```

5)
```
SELECT essn, pno, fname, lname
FROM works_on w, employee e
WHERE w.essn = e.ssn AND lname = 'Smith';
```

6)
```
Select fname, lname, address
FROM employee
WHERE address LIKE '%Houston%';
```

7)
```
Select fname, Salary * 1.1 = Salary
FROM project, employee
WHERE pname = 'Product x' and dno = dnum;
```

8) SELECT fname, lname, salary
FROM employee
WHERE salary BETWEEN 30000 AND 40000;

9) SELECT pno, dno, fname, lname
FROM works_on, employee
WHERE essn = ssn ORDER BY dno, fname, lname;

10) SELECT ssn, fname, lname
FROM employee
WHERE super_ssn IS NULL;

11) SELECT fname, minit, lname
FROM employee e, dependents d
WHERE e.fname = d.dependent_name and
e.sex = d.sex;

12) SELECT fname, lname, ssn
FROM employee
WHERE ssn NOT IN (
SELECT essn
FROM dependent);

13) SELECT distinct fname
FROM employee, department, dependent
WHERE ssn = mgr_ss AND ssn = essn;

14) SELECT pnumber, ssn
FROM project, employee
WHERE pnumber IN (1,2,3) AND dnum = dno
ORDER BY pnumber;

15)

16)

17)

18)

19)

15) SELECT sum(salary) SUM, max(salary) MAX, min(salary) MIN, avg(salary) AVG. FROM employee;

16) SELECT sum(salary) SUM, max(salary) MAX, min(salary) MIN, avg(salary) AVG
FROM department, employee
WHERE dname = 'research' AND dnumber = dno;

17) CREATE VIEW nums AS
SELECT pno, count(essn) count
FROM works-on
GROUP BY pno;

SELECT pname, pno, count
FROM project, nums
WHERE pno = pnumber;

18) SELECT pname, pno, count
FROM project, nums
WHERE pno = pnumber AND count > 2;

19) CREATE VIEW nums2 AS
SELECT dno, count(ssn) no_emps
FROM employee
GROUP BY dno;

CREATE VIEW num3 AS
SELECT dno, count(ssn) no_emps
FROM employee
GROUP BY dno;
CREATE VIEW num3 AS
SELECT dno
FROM nums2
WHERE no_emps > 4;

Ques 1. Write a PL/SQL block to display the GPA of given student.

```
Set serveroutput on
declare
S1_roll StudentTable.RollNo %. TYPE;
S1_gpa StudentTable.GPA %.TYPE;
begin
        S1_roll := '&RollNo';
SELECT GPA
INTO S1_gpa
FROM StudentTable
WHERE RollNo = S1_roll;

dbms_output.put_line ('RollNo:' ||S1_roll||'
                        GPA :' ||S1_gpa||'');
end;
/
```

Ques 2. Write a PL/SQL block to display the letter grade (0-4: F ; 4-5 : E, 5-6: D ; 6-7 : C ; 7-8 : B, 8-9 : A; 9-10: A+) of given student

```
Set serveroutput on
declare
S1_roll StudentTable.RollNo %.TYPE;
S1_gpa StudentTable.GPA %.TYPE;
begin
        S1_roll := '&RollNo';
SELECT GPA
INTO S1_gpa
FROM StudentTable
WHERE RollNo = S1_roll;
```

```
IF s1_gpa <= 4 THEN
DBMS_OUTPUT.PUT_LINE ('F');

ELSIF s1_gpa >4 and s2_gpa <=5 THEN
DBMS_OUTPUT.PUT_LINE ('E');

ELSIF s1_gpa >8 and s2_gpa <= 10 THEN
DBMS_OUTPUT.PUT_LINE ('A+');

ELSE
DBMS_OUTPUT.PUTLINE ('No such GPA');
END IF;

end;
/
```

Ques3. Input the date of issue and date of return for a book. Calculate and display the fine with appropriate message using PL/SQL block.

```
set serveroutput on
declare
 issue date;
 rdate date;
 fine number(5);
 late number(2);
begin
      issue := to_date('&issue', 'dd-mm-yy');
      rdate := to_date('&rdate', 'dd-mm-yy');

late := rdate - issue;

If late <= 7 THEN
DBMS_OUTPUT.PUTLine ('No fine');
```

```
ELSIF &to between 8 and 15 THEN
DBMS_OUTPUT.PUT_LINE ('The fine is: '||&to*2||");

ELSIF &to between 16 and 30 THEN
DBMS_OUTPUT.PUT_LINE ('The fine is: '||&to*2||'');

ELSIF &to >= 30 THEN
DBMS_OUTPUT.PUT_LINE ('The fine is: '||&to*5||'');

ELSE
DBMS_OUTPUT.PUT_LINE ('No such input');
END IF;

end;
/
```

Ques 4. Write a PL/SQL block to print the letter grade of all the students (Roll No: 1-5).

```
Set serveroutput on
declare
S1_roll number(5);
S1_gpa StudentTable.GPA%TYPE;
begin
S1_roll := 0;
LOOP
S1_roll := S1_roll + 1;

SELECT GPA
INTO S1_gpa
FROM StudentTab
WHERE RollNo = S1_roll;

If S1_gpa <= 4 THEN
```

```
DBMS_OUTPUT.PUT_LINE ('F');

ELSIF s1_gpa > 4 and s1_gpa <= 5 THEN
DBMS_OUTPUT.PUT_LINE ('E');

ELSIF s1_gpa > 5 and s1_gpa <= 6 THEN
 DBMS_OUTPUT.PUT_LINE ('D');

ELSIF s1_gpa > 6 and s1_gpa <= 7 THEN
DBMS_OUTPUT.PUT_LINE ('C');

ELSIF s1_gpa > 7 and s1_gpa <= 8 THEN
DBMS_OUTPUT.PUT_LINE ('B');

ELSIF s1_gpa > 8 and s1_gpa <= 9 THEN
DBMS_OUTPUT.PUT_LINE ('A');

ELSIF s1_gpa > 9 and s1_gpa <= 10 THEN
DBMS_OUTPUT.PUT_LINE ('A+');

ELSE DBMS_OUTPUT.PUT_LINE ('No such GPA');
END IF;

If s1_roll > 5 THEN EXIT;
END IF;
END LOOP;

end;
/
```

Ques 5.

Ques:- Alter Student Table by appending an additional column Letter Grade Varchar2(2). Then write a PL/SQL block to update the table with letter grade of each student.

```
Set serveroutput on

DECLARE
S1_roll number := 0;
S2_gpa StudentTable.GPA %TYPE;
S1_grade varchar(2);
BEGIN

WHILE S2_roll <5
  LOOP
S1_roll := S1_roll + 1;

SELECT GPA
  INTO S1_gpa
FROM StudentTable
WHERE RollNo = S2_roll;

If S1_gpa <= 4 THEN
S1_grade := 'F';

ELSIF S1_gpa >4 and S1_gpa <= 5 THEN
S2_grade := 'E';

ELSIF S1_gpa > 5 and S2_gpa <= 6 THEN
S1_grade := 'D';

ELSIF S1_gpa >6 and S1_gpa <=7 THEN
S1_grade := 'C';
```

```
ELSE
DBMS_OUTPUT.PUT_LINE ('No Such GPA');
END IF;

Update Student_Table set LetterGrade = S2_grade
where RollNo = S2_Roll;
END LOOP;

DBMS_OUTPUT.PUT_LINE ('-----The table is
updated-----');
END;
/
```

Ques 6. Write a PL/SQL block to fins the Student with max. GPA without using aggregate function.

```
Set serveroutput on
declare
        gpa Studenttable.gpa%TYPE;
        maxval number;
begin
        Select gpa into maxval
        from Studentable
        where rollno = 2;

        for i in 2..5 loop
            select gpa into gpa
            from Student table
            where rollno = i;

            if gpa > maxval then maxval := gpa;
            end if;
end loop;
```

Ques 7.

dbms_output.put_line('-----');
dbms_output.put_line('Max GPA = '||max_gpa);
end;
/

Ques 7: Implement the exercise 4 using GOTO.

```
set serveroutput on
declare
    gpa student_table.gpa%TYPE;
    grade varchar(2);
    i number := 1;
begin
    dbms_output.put_line('-----');
    <<print_now>>
    loop
        select gpa into gpa
        from student_table
        where rollno = i;

        if gpa < 4 then grade := 'F';
        elsif gpa between 4 and 5 then grade := 'E';
        elsif gpa between 5 and 6 then grade := 'D';
        elsif gpa between 6 and 7 then grade := 'C';
        elsif gpa between 9 and 10 then grade := 'A+';
        end if;

        dbms_output.put_line('RollNo: '||i||'
                              Grade = '||grade);

        i := i+2;
        if i <= 5 then GOTO print_now;
        else exit;
        end if;
    end loop;
end;
/
```

Ques 8. Based on University database Schema, write a PL/SQL block to display the details of the instructor whose name is supplied by the user. Use exceptions to show appropriate error message for the following cases:
a) Multiple instructors with the same name
b) No instructor for the given name.

```
Set Serveroutput on
declare
    multiple_name exception;
    inst_name instructor.name%TYPE;
    inst_id instructor.id%TYPE;
    inst_sal instructor.salary%TYPE;
    inst_dept instructor.dept_name%TYPE;
    name_count number;
begin
    inst_name := '&name';

    select count(*)
    into name_count
    from instructor
    where name = inst_name;

    select id, salary, dept_name
    into inst_id, inst_sal, inst_dept
    from instructor
    where name = inst_name;

if name_count > 1 then RAISE multiple_name;
else
dbms_output.put_line('Name: '||inst_name);
dbms_output.put_line('ID: '||inst_id);
```

Ques 9.

dbms_output.put_line('Salary : '||inst_sal);
dbms_output.put_line('Department : '||inst_dept);
end if;

exception
WHEN multiple_rows then dbms_output.put_line
('Multiple instructors with same name');
WHEN others then dbms_output.put_line('No
instructor found');
end;
/

Ques 9. Extend lab exercise 5 to validate GPA value used to find letter grade. If it is out side the range, 0-10, display an error message, 'Out of range' via an exception handler.

set serveroutput on
declare
not_exist exception;
gpa student_table.gpa%type;
grade varchar(2);
i number:=1;
begin
while i<=5
loop
select gpa into gpa
from student_table
where rollno=i;

if gpa between 0 and 4 then grade:='F';
elsif gpa between 4 and 5 then grade:='E';
elsif gpa between 5 and 6 then grade:='D';
;

elsif gpa between 9 and 10 then grade := 'A+';
else raise not_exist;
end if;

update student1table set lettergrade = grade
where rollno = i;
j := i + 2;
end loop;
exception
when not_exist then dbms_output.put_line
('GPA Out of bounds');
end;
/