```
create table DEPT(
DEPTNO int(2),
DNAME varchar(15),
LOC      varchar(10));

Insert into DEPT(DEPT.DEPTNO, DEPT.DNAME, DEPT.LOC) values
(10, 'ACCOUNTING', 'NEW YORK'),
(20, 'RESEARCH', 'DALLAS'),
(30, 'SALES', 'CHICAGO'),
(40, 'OPERATIONS', 'BOSTON');

select * from DEPT;

create table EMP(
EMPNO int(4),
ENAME varchar(10),
JOB      varchar(9),
HIREDATE date,
SAL      float(7,2),
COMM float(7,2),
DEPTNO int(2)
);

Insert into EMP (EMP.EMPNO, EMP.ENAME, EMP.JOB, EMP.HIREDATE, EMP.SAL, EMP.COMM,
EMP.DEPTNO) values
(7839, 'KING', 'MANAGER', '1991-11-17', 5000, NULL, 10),
(7698, 'BLAKE', 'CLERK', '1981-05-01', 2850, NULL, 30),
(7782, 'CLAR', 'MANAGER', '1981-06-09', 2450, NULL, 10),
(7566, 'JONES', 'CLERK', '1981-04-02', 2975, NULL, 20),
(7654, 'MARTIN', 'SALESMAN', '1981-09-28', 1250, 1400, 30),
(7499, 'ALLEN', 'SALESMAN',      '1981-02-20', 1600, 300, 30);
```

```
select * from EMP;

-- 3.	Display all the employees where SAL between 2500 and 5000 (inclusive of both).
select * from EMP
	where EMP.SAL between 2500 and 5000;

-- 4.	Display all the ENAMEs in descending order of ENAME.
select EMP.ENAME from EMP
	order by ENAME desc;

-- 5.	Display all the JOBs in lowercase.
select lower(EMP.JOB) from EMP;


-- 6.	Display the ENAMEs and the lengths of the ENAMEs.
select ENAME, length(ENAME) from EMP;

-- 7.	Display the DEPTNO and the count of employees who belong to that DEPTNO .
select DEPTNO, count(DEPTNO) from EMP
	group by DEPTNO;

-- 8.	Display the DNAMEs and the ENAMEs who belong to that DNAME.
select DEPT.DNAME, EMP.ENAME from DEPT, EMP
	where EMP.DEPTNO = DEPT.DEPTNO
		order by DEPT.DEPTNO;

-- 9.	Display the position at which the string ‘AR’ occurs in the ename.
select ENAME from EMP
	where ENAME like '%AR%';
```

-- 10. Display the HRA for each employee given that HRA is 20% of SAL.

```
select EMP.EMPNO, EMP.ENAME, EMP.JOB, EMP.HIREDATE, EMP.SAL, EMP.COMM, EMP.DEPTNO,
EMP.SAL*0.20 as 'HRA' from EMP;
```

/* 1. Write a stored procedure by the name of PROC1 that accepts two varchar strings as parameters.

Your procedure should then determine if the first varchar string exists inside the varchar string. For example,

if string1 = ‘DAC’ and string2 = ‘CDAC, then string1 exists inside string2.

The stored procedure should insert the appropriate message into a suitable TEMPP output table.

Calling program for the stored procedure need not be written. */

```
create table temppp(
String1 varchar(10),
String2 varchar(20),
Result varchar(40)
);

delimiter //
create procedure PROC1(string1 varchar(10), string2 varchar(20))
begin
declare result int;
set result = (instr(string2, string1));
if result = 0 then
        insert into temppp values(string1, string2, 'String 1 does not exist in String 2');
else
        insert into temppp values(string1, string2, 'String 1 exists in String 2');
end if;
end; //
delimiter ;

call PROC1('DAC', 'CDAC');
call PROC1('SHRUTIKA', 'JANTRE');
```

```
select * from temppp;
```

/* 2. Create a stored function by the name of FUNC1 to take three parameters, the sides of a triangle.

The function should return a Boolean value:- TRUE if the triangle is valid, FALSE otherwise.

A triangle is valid if the length of each side is less than the sum of the lengths of the other two sides.

Check if the dimensions entered can form a valid triangle. Calling program for the stored function need not be written. */

```
create table tempp(
Side1 float,
Side2 float,
Side3 float,
TriangleType varchar(30)
);

delimiter //
create function FUNC1(a float, b float, c float)
returns boolean
deterministic
begin
if (a < (b + c)) and (b < (a + c)) and (c < (a + b)) then
        return true;
else
        return false;
end if;
```

```
end; //
delimiter ;

delimiter //
create procedure tri(x float, y float, z float)
begin
if FUNC1(x, y, z) then
        insert into tempp values(x, y, z, 'The triangle is Valid');
else
        insert into tempp values(x, y, z, 'The triangle is Invalid');
end if;
end; //
delimiter ;

call tri(2,3,5);
call tri(1,1,1);
select * from tempp;
```