**Q 1. Consider the following three tables.**

**EMPLOYEE( empno, name, deptno,job,hiredate, sal , comission, dob, city, phone)**

**DEPARTMENT( deptno, dname, manager,loc)**

**SALARY( eno, basic,HR,DA,tax).**

**Write equivalent SQL for the following query. (Use foreign key to join**

**the tables.)**

**SOLUTION:-**

create database dbms\_project1221;

use dbms\_project1221;

create table DEPARTMENT1221(

deptno varchar(20) PRIMARY KEY,

dname varchar(20),

manager varchar(20)

);

create table EMPLOYEE1221(

empno varchar(20) PRIMARY KEY,

name varchar(20),

deptno varchar(20),

job varchar(20),

hiredate DATE,

sal int,

comission int,

dob DATE,

city varchar(20),

phone int,

FOREIGN KEY(deptno) references DEPARTMENT1221(deptno)

ON UPDATE CASCADE ON DELETE CASCADE

);

create table SALARY1221(

eno varchar(20),

basic int,

HR int,

DA int,

tax int,

FOREIGN KEY(eno) references EMPLOYEE1221(empno)

ON UPDATE CASCADE ON DELETE CASCADE

);

insert into DEPARTMENT1221 values('d1','engineering','Farheen'),('d2','Polytechnic','Sarib');

insert into EMPLOYEE1221 values('e1','Mario','d1','teacher','2010-02-12',20000,3000,'1993-02-16','jaddah',888888),('e2','Bushra','d2','teacher','2010-05-12',25000,3500,'1993-02-08','Bareily',666666),('e3','Sarah','d1','Assistant','2009-01-10',50000,7000,'1990-07-02','Delhi',000000);

insert into SALARY1221 values('e1',2000,3000,1000,500),('e2',5000,2000,1000,800),('e3',10000,2000,4000,900);

select sal+HR+DA-tax as pay from

insert into DEPARTMENT1221 values('d3','accounting','Sohrab');

insert into EMPLOYEE1221 values('e4','Sullu','d3','clerk','2011-02-10',15000,3000,'1993-09-21','Delhi',888777);

insert into SALARY1221 values('e4',200,300,100,50);

![A description...](data:None;base64,)

**QUERIES:**

**1) Get the name and city of the employee working for the accounting de**

**partment?**

**query:**select name,city from EMPLOYEE1221,DEPARTMENT1221 where EMPLOYEE1221.deptno=DEPARTMENT1221.deptno AND DEPARTMENT1221.dname='accounting';

![A description...](data:None;base64,)

**2) Get the name, department name of all the employees whose pay is gre**

**ater than 10000.**

**query:**select EMPLOYEE1221.name,DEPARTMENT1221.dname from EMPLOYEE1221,DEPARTMENT1221 where EMPLOYEE1221.sal>10000 AND

EMPLOYEE1221.deptno=DEPARTMENT1221.deptno;

![A description...](data:None;base64,)

**3) Get the name of the employee in ascending and descending order.**

**query:**select name from EMPLOYEE1221 order by name;

select name from EMPLOYEE1221 order by name DESC;

![A description...](data:None;base64,)

**4) Update the city of the employee no.2 from Mumbai to Delhi.**

**query:**update EMPLOYEE1221 set city='Delhi' where city='Mumbai';

![A description...](data:None;base64,)

**5) Get the sum of the basic salary of the employees belongs to Delhi c**

**ity.**

**query:**select ADD(sal) from EMPLOYEE1221,SALARY1221 where EMPLOYEE1221.empno=SALARY1221.eno AND EMPLOYEE1221.city='Delhi';

![A description...](data:None;base64,)

**6)Get the details of the highest income tax payee**

**query:**select \* from EMPLOYEE1221,SALARY1221 where SALARY1221.tax=(select MAX(SALARY1221.tax) from SALARY1221) AND

EMPLOYEE1221.empno=SALARY1221.eno;

![A description...](data:None;base64,)

**7) Which employee is the senior most?**

**qeury:**select name from EMPLOYEE1221 where dob=(select min(dob) from EMPLOYEE1221);

![A description...](data:None;base64,)

**8) Give the details of second highest salary employee (without use of**

**� < � operator).**

**query:**select \* from EMPLOYEE1221 order by sal DESC limit 1,1;

![A description...](data:None;base64,)

9)

10)

**11) Give the details of all employees of 5th highest salary ( or nth h**

**ighest salary).**

**query:s**elect \* from EMPLOYEE1221 order by sal DESC limit 4,1;

![A description...](data:None;base64,)

**12) How many clerks are there in the company?**

**query:**select count(job) as NO\_clerks from EMPLOYEE1221 where job='clerk';

![A description...](data:None;base64,)

**13) Which department has exactly one employee as clerk?**

**query:**select dname,EMPLOYEE1221.deptno,count(job) from (EMPLOYEE1221 inner join DEPARTMENT1221 on EMPLOYEE1221.deptno=DEPARTMENT1221.deptno) group by deptno,job having count(job)=1 and job='clerk';

![A description...](data:None;base64,)

**14) Which department has the highest number of clerks? Show the deptno**

**and count**

**query:s**elect dname,EMPLOYEE1221.deptno,count(job),job from (EMPLOYEE1221 inner join DEPARTMENT1221 on EMPLOYEE1221.deptno=DEPARTMENT1221.deptno) group by deptno,job having job='clerk' order by count(job) DESC limit 0,1;

![A description...](data:None;base64,)

**15) How many employees are there in each department?**

**query:**select deptno,count(empno) from EMPLOYEE1221 group by deptno;

![A description...](data:None;base64,)

**16) List the lowest salary for different jobs used in a company and li**

**st them in descending order**

**query:**select job,min(sal) from EMPLOYEE1221 group by job order by min(sal) DESC;

![A description...](data:None;base64,)

**17) Which department average salary is the lowest among all? Show the**

**deptno,average salary.**

**query:**select deptno,avg(sal) from EMPLOYEE1221 group by deptno order by avg(sal) limit 0,1;

![A description...](data:None;base64,)

**18) List the minimum, maximum and average salary for each job.**

**query:**select job,min(sal) as MIN\_SAL,max(sal) as MAX\_SAL from EMPLOYEE1221 group by job;

![A description...](data:None;base64,)

**19) Compute the difference between maximum and minimum salary**

**query:**select max(sal)-min(sal) as DIFF\_SAL from EMPLOYEE1221;

![A description...](data:None;base64,)

**20) List the names of the employees whose name contains LA.**

**query:**select name from EMPLOYEE1221 where name like "%LA%";

![A description...](data:None;base64,)

**21) List the names of the employees whose joining date is between 2nd**

**April,1981 and 8th Sept,1981.**

**query:**select name from EMPLOYEE1221 where hiredate>'1981-04-02' and hiredate<'1981-09-08

![A description...](data:None;base64,)

**22) How many different job titles exist in the employee table?**

**query:** select count(DISTINCT job) from EMPLOYEE1221;

![A description...](data:None;base64,)

**23) Compute the sum of all salaries of employee working under deptno=3**

**0.**

**query:**select sum(sal) from EMPLOYEE1221 where deptno='d3';

![A description...](data:None;base64,)

**24) For each salesman in the emp table retrieve the deptno and departm**

**ent name.**

**query:**select dname,EMPLOYEE1221.deptno from (EMPLOYEE1221 inner join DEPARTMENT1221 on EMPLOYEE1221.deptno=DEPARTMENT1221.deptno) where job='salesman';

![A description...](data:None;base64,)

**25) List the names of all the employees with their name of the manager.**

**query:**select name,manager from (EMPLOYEE1221 inner join DEPARTMENT1221 on EMPLOYEE1221.deptno=DEPARTMENT1221.deptno);

![A description...](data:None;base64,)

**26) List all employees who are working in department located at CHICAG**

**O.**

**query:**select name from EMPLOYEE1221 where city='Chicago';

![A description...](data:None;base64,)

**27) List all the employees who are working in same department as their**

**managers.**

**query:**select manager,name from (EMPLOYEE1221 inner join DEPARTMENT1221 on EMPLOYEE1221.deptno=DEPARTMENT1221.deptno) group by manager,empno;

![A description...](data:None;base64,)

**28) Retrieve all the employees who are working in deptno=10 and who ea**

**rn salary atleast as much as any employee working in deptno=30.**

**query:**select name from EMPLOYEE1221 where sal >= (select min(sal) from EMPLOYEE1221 where deptno='d3') and deptno!= 'd3' ;

![A description...](data:None;base64,)

**29) List all the department who have no employees**

**query:**select dname from DEPARTMENT1221 where dname NOT IN (select dname from (EMPLOYEE1221 inner join DEPARTMENT1221 on EMPLOYEE1221.deptno=DEPARTMENT1221.deptno));

![A description...](data:None;base64,)

**30)Delete the EC department**

**query:**delete from DEPARTMENT1221 where dname='EC';

![A description...](data:None;base64,)

**Q.2)**

**1. Write a function and a stored procedure to print Hello ! How are**

**you?**

**->function**

delimiter /

create function myfunc()

returns text

begin

return "Hello ! How are you?";

end;

/

delimiter ;

![A description...](data:None;base64,)

**->procedure**

delimiter /

create procedure mypro()

begin

select 'Hello ! How are you?';

end;

/

delimiter ;

![A description...](data:None;base64,)

**2. Write a function and a stored procedure to count the number of em**

**ployees in the table employee?**

delimiter /

create function myfunc2()

returns int

begin

return (select count(\*) from EMPLOYEE1221);

end;

/

delimiter ;

select myfunc2();

![A description...](data:None;base64,)

delimiter /

create procedure mypro2()

begin

declare total int;

select count(EMPLOYEE1221.empno) from EMPLOYEE1221;

end;

/

delimiter ;

![A description...](data:None;base64,)

**3. Write a function and a stored procedure to calculate the factoria**

**l of the given number.**

**->function**

delimiter /

create function myfunc3(n int)

returns int

begin

declare i int default 1;

declare s int default 1;

myloop:loop

if i > n then

leave myloop;

else

set s = s\*i;

set i = i+1;

end if;

iterate myloop;

end loop;

return s;

end;

/

delimiter ;

![A description...](data:None;base64,)

**->procedure**

delimiter /

create procedure mypro3(in n int)

begin

declare i int default 1;

declare s int default 1;

myloop:loop

if i > n then

leave myloop;

else

set s = s\*i;

set i = i+1;

end if;

iterate myloop;

end loop;

select s;

end;

/

delimiter ;

![A description...](data:None;base64,)

**4. Write a function and a stored procedure to calculate the average**

**of three numbers.**

**->function**

delimiter $$

create function myfunc4(a float,b float,c float)

returns float

begin

declare avg float default 0.0;

set avg = ( a + b + c ) / 3;

return avg;

end;

$$

delimiter ;

![A description...](data:None;base64,)

**->procedure**

delimiter $$

create procedure mypro4(in a float,in b float,in c float)

begin

declare avg float default 0.0;

set avg = ( a + b + c ) / 3;

select avg;

end;

$$

delimiter ;

![A description...](data:None;base64,)

**5. Write a function and stored procedure to find fibonacci series and**

**its sum?**

**->function**

delimiter $$

create function myfunc5(n int)

returns int

begin

declare s int default 1;

declare a int default 0;

declare b int default 1;

declare c int default 0;

declare con varchar(20);

declare i int;

set i = 3;

set con = concat("0"," ","1"," ");

myloop:loop

if i > n then

leave myloop;

else

select a + b into s;

set a = b;

set b = c;

set s = s + c;

set i = i + 1;

select concat(con," ",c) into con;

iterate myloop;

end if;

end loop;

select con;

return s;

end;

$$

**->procedure**

delimiter $$

create procedure mypro5(in n int)

begin

declare s int default 1;

declare a int default 0;

declare b int default 1;

declare c int default 0;

declare con varchar(20);

declare i int;

set i = 3;

set con = concat("0"," ","1"," ");

myloop:loop

if i > n then

leave myloop;

else

set c = a + b;

set a = b;

set b = c;

set s = s + c;

set i = i + 1;

set con = concat(con," ",c);

iterate myloop;

end if;

end loop;

select con;

select s;

end;

$$

![A description...](data:None;base64,)

**Q.3)Consider the following relations**

**Student (snum : integer ,sname:string,major :string,level : string,age**

**:integer).,**

**Class (name: string, meets\_at: time, room: string, fid: integer).**

**Enrolled (snum: integer, cname:string).Faculty (fid: intger, fname: st**

**ring, deptid: integer);**

**Enrolled has on record per studentclass pair such that the student is**

**enrolled in the class.**

**Write the SQL queries. No duplicates should be printed.(use foreign ke**

**y )**

**SOLUTION:**

create table Faculty1221

(

fid int PRIMARY KEY,

fname varchar(20),

deptid int

);

create table Class1221

(

name varchar(20) PRIMARY KEY,

meets\_at timestamp,

room varchar(20),

fid int,

FOREIGN KEY(fid) references Faculty1221(fid)

ON UPDATE CASCADE ON DELETE CASCADE

);

create table Student1221

(

snum int PRIMARY KEY,

sname varchar(20),

major varchar(20),

level varchar(20),

age int

);

create table Enrolled1221

(

snum int,

cname varchar(20),

PRIMARY KEY(snum,cname),

FOREIGN KEY(snum) references Student1221(snum)

ON UPDATE CASCADE ON DELETE CASCADE,

FOREIGN KEY(cname) references Class1221(name)

ON UPDATE CASCADE ON DELETE CASCADE

);

insert into Faculty1221 values(01,'Farheen','01'),(02,'Arif',02),(03,'Sohrab',03),(04,'Yasir',01),(05,'I.Teach',02);

insert into Class1221 values('Computer','10-00-00','R128',01),('Networking','02-30-00','R128',01),('Acting','10-00-00','BA1080',05),('Writing','12-30-00','05-00-00','R128',05),('Histoty','06-30-00','R678',02);

insert into Student1221 values(01,'Bushra','Computer','SR',20),(02,'Maria','Acting','JR',21),(03,'Sarib','SR','Writing','SR',22),(04,'Sarah','History','JR',21),(05,'Sultana','Networking','JR',23);

insert into Enrolled1221 values(01,'Computer'),(02,'Acting'),(03,'Writing'),(04,'History'),(05,'Networking'),(06,'Computer');

![A description...](data:None;base64,)

**QUERIES:**

**1)Find the names of all Juniors (level = JR) who are enrolled in**

**a class taught by I. Teach.**

**query:s**elect Student1221.sname from Student1221,Enrolled1221,Class1221,Faculty1221 where Student1221.snum=Enrolled1221.snum AND Enrolled1221.cname=Class1221.name and Class1221.fid=Faculty1221.fid and level='JR' and Faculty1221.fname='I.Teach';

![A description...](data:None;base64,)

**2.**

**Find the age of the oldest student who is either a History maj**

**or or enrolled in a course taught by I. Teach.**

**query:**select max(Student1221.age) from Student1221,Enrolled1221,Class1221,Faculty1221 where Student1221.snum=Enrolled1221.snum AND Enrolled1221.cname=Class1221.name and Class1221.fid=Faculty1221.fid and (Student1221.major='History' OR Faculty1221.fname='I.Teach');

![A description...](data:None;base64,)

**3.Find the names of all classes that either meet in room BA1080**

**or have 2 or more students enrolled.**

**query:**select Class1221.name from ((Enrolled1221 inner join Student1221 on Enrolled1221.snum=Student1221.snum) inner join Class1221 on Class1221.name=Enrolled1221.cname) group by Class1221.name having count(Class1221.name) >=2 UNION select Class1221.name from Class1221 where Class1221.room='BA1080';

![A description...](data:None;base64,)

**4.**

**Find the names of all students who are enrolled in two classes**

**that meet at the same time.**

**query:s**elect Student1221.sname from ((Enrolled1221 inner join Student1221 on Enrolled1221.snum=Student1221.snum) inner join Class1221 on Class1221.name=Enrolled1221.cname) group by Class1221.meets\_at,Student1221.snum having count(Enrolled1221.cname) >= 2;

![A description...](data:None;base64,)

**5.**

**Find the names of faculty members who teach in every room in w**

**hich some class is taught.**

**query:**select Faculty1221.fname from (Class1221 inner join Faculty1221 on Class1221.fid=Faculty1221.fid) group by Faculty1221.fid having count(DISTINCT Class1221.room)=(select count(DISTINCT Class1221.room) from Class1221);

![A description...](data:None;base64,)

**6.**

**Find the names of faculty members for whom the combined enroll**

**ment of the courses that they teach is less than five.**

**query:**select Faculty1221.fname from(Class1221 right join Faculty1221 on Class1221.fid=Faculty.fid) group by Faculty1221.fname having count(DISTINCT Class1221.name) < 5;

![A description...](data:None;base64,)

**7.**

**For each level, print the level and the average age of student**

**s for that level.**

**query:s**elect Student1221.level ,avg(age) as AVG\_age from Student1221 group by Student1221.level;

![A description...](data:None;base64,)

**8.**

**For all levels except JR, print the level and the average age**

**of students for that level.**

**query:**select Student1221.level ,avg(Student1221.age) as AVG\_NOT\_JR from Student1221 where Student1221.level != 'JR' group by Student1221.level;

![A description...](data:None;base64,)

**9.**

**For each faculty member that has taught classes only in room R128, print the faculty members name and the total number of classes she or he has taught.**

**query:**select Faculty1221.fname,count(Class1221.name) as COUNT from (Class1221 inner join Faculty1221 on Class1221.fid=Faculty1221.fid) where Class1221.room='R128' group by Class1221.fid;

![A description...](data:None;base64,)

**10.**

**Find the names of students enrolled in the maximum number of c**

**lasses.**

**query:**select Student1221.sname,Class1221.name from ((Enrolled1221 inner join Student1221 on Enrolled1221.snum=Student1221.snum) inner join Class1221 on Class1221.name=Enrolled1221.cname) group by Student1221.sname having count(Class1221.name) >= ALL(select count(\*) from((Enrolled1221 inner join Student1221 on Enrolled1221.snum=Student1221.snum) inner join Class1221 on Class1221.name=Enrolled1221.cname) group by Student1221.sname);

![A description...](data:None;base64,)

![A description...](data:None;base64,)

**Write equivalent SQL for the following query.**

**SOLUTION:**

create table book1221(

bookid INT,

title varchar(20),

No\_of\_pages int,

copyright varchar(20),

PRIMARY KEY(bookid)

);

create table Author1221(

authid int,

authFirst varchar(20),

Authmid varchar(20),

Age int,

authCity varchar(20),

AuthlastName varchar(20)

PRIMARY KEY(authid)

);

create table WrittenBy1221(

Publisher\_Name varchar(20),

bookid int,

authid int,

PRIMARY KEY(bookid,authid),

FOREIGN KEY(bookid) references book1221(bookid)

ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY(authid) references Author1221(authid)

ON DELETE CASCADE ON UPDATE CASCADE

);

insert into book1221 values(1,'Introduction to Java',400,'source\_billion'),(2,'Mathematics at stake',599,'eye on'),(3,'Reading power',453,'check it');

insert into Author1221 values(1,'Jenny','D.,20,'London','Dave'),(2,'Sarib','Shaa',80,'Kanpur','Shahbaz'),(3,'Jenny','D.',20,'London','Dave'),(4,'Arheen','Pilo',90,'Mumbai','Pillai');

insert into WrittenBy1221 values ('Wiley',1,1),('digit',2,1),('Wiley',1,2),('Prinkson',3,4),('Outstanding',2,3);

**QUERIES:**

**Write equivalent SQL for the following query.**

**1. NOT DONE YET,WHAT IS a?**

Get the title,Author1221 name,publisher name for Author1221 whose city

contain total no of a=2?//WHAT IS a?

->

**2.**

**Give the details of the book1221 which is written by at least two**

**Author1221s.**

->select book1221.bookid,title,No\_of\_pages,copyright,Publisher\_Name from ((book1221 inner join WrittenBy1221 on book1221.bookid=WrittenBy1221.bookid) inner join Author1221 on WrittenBy1221.authid=Author1221.authid) group by Author1221.authid having count(\*) >= 2;

A description...

**3.**

**Write a stored procedure (SP Name : insertIntoAuth) to insert**

**the Author information.**

**->**

delimiter $$

create procedure insertIntoAuth(in aid int,in afn varchar(20),in amn varchar(20),in ag,in ac varchar(20),in aln varchar(20))

begin

insert into Author(authid,authFirst,Authmid,Age,authCity,AuthlastName) values(aid,afn,amn,ag,ac,aln);

end$$

delimiter ;

A description...

**4.**

**Write a stored procedure (SP Name : insertBookInfo) to insert**

**the book information such as bookid, title, no. of pages, copyright,**

**authorId, Publisher Name. (Use two stored procedure and call it from o**

**ne stored procedure i.e nested SP). (SP Name : insertBook, insertWBy).**

**-**

**->**

delimiter $$

create procedure insertWBy1(in pub varchar(20),in bid int,in aid int)

begin

insert into WrittenBy1221 values(pub,bid,aid);

end$$

create procedure insertBook1(in bk int,in tit varchar(20),in nop int,in cpr varchar(20))

begin

insert into book1221 values(bk,tit,nop,cpr);

end$$

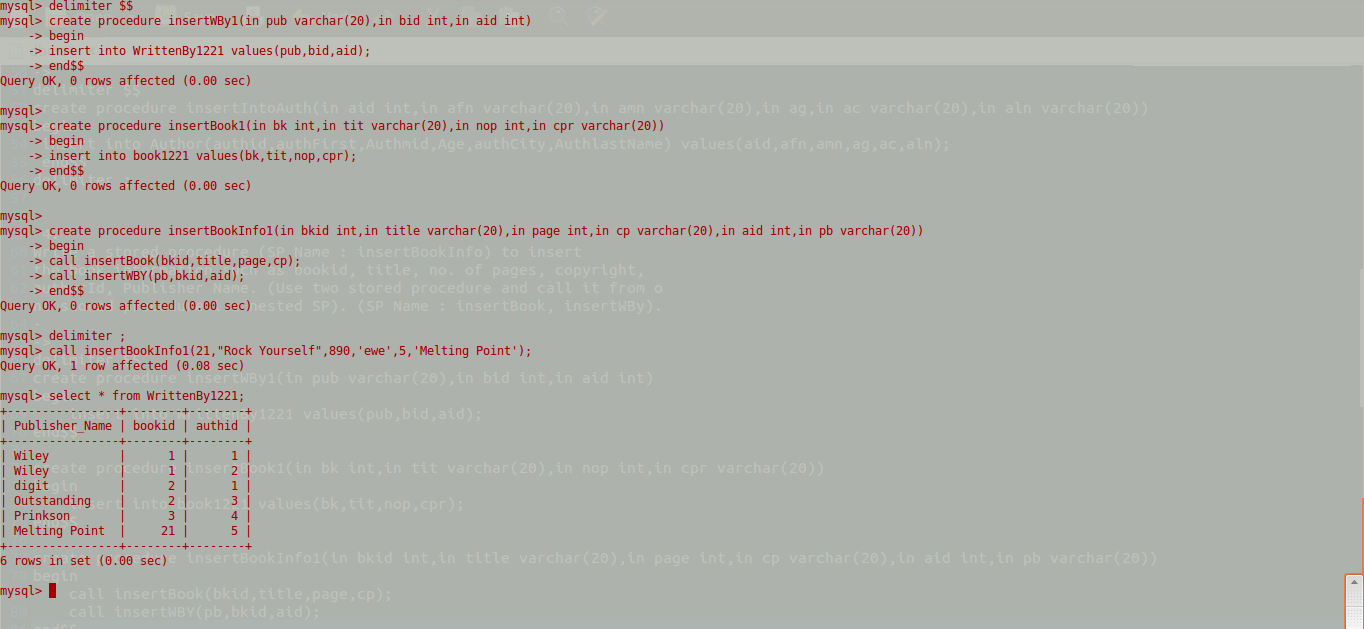
create procedure insertBookInfo1(in bkid int,in title varchar(20),in page int,in cp varchar(20),in aid int,in pb varchar(20))

begin

call insertBook(bkid,title,page,cp);

call insertWBY(pb,bkid,aid);

end$$



**5. //NOT DONE YET**

**Write a stored procedure to delete the Author information usin**

**g its AuthID. (Notr: If Author book(in Book Table) exists for AuthID,**

**then it should display message as You cant delete Author because total**

**no book exist in BookTable. First delete all the books written by him**

**).**

delimiter $$

create procedure del\_auth3(in aid int)

begin

Declare cnt int default 0;

Declare at int default 0;

set at=aid;

select at;

select count(authid) INTO cnt from WrittenBy1221 where authid=(select at);

if cnt=0 then

delete from Author1221 where authid=aid;

else

select "You cant delete Author because total";

delete from book1221 where exists(select WrittenBy1221.bookid from WrittenBy1221,book1221 where aid=WrittenBy1221.authid and WrittenBy1221.bookid=book1221.bookid);

delete from Author1221 where authid=aid;

end if;

end

$$

delimiter $$

create procedure del\_auth31(in aid int)

begin

declare ct int default 0;

set ct =`select count(authid) from WrittenBy1221 where authid = aid`;

if ct = 0 then

delete from Author1221 where authid = aid;

select 'deleted';

else

select 'You cant delete Author because total

no book exist in BookTable. First delete all the books written by him';

end if;

end$$

6. //DELETING THE TABLE CONTENT INSTEAD OF DELETING PARTICULAR ROWS

Write a stored procedure to delete the Book using AuthID.

NOTE: Book information should be deleted from both Book and Book\_writtenBy\_Author table.

delimiter $$

create procedure del\_Book5(in aid int)

begin

delete from book1221 where exists (select book1221.bookid from (book1221 inner join WrittenBy1221 on book1221.bookid=WrittenBy1221.bookid) where WrittenBy1221.authid=aid);

end

$$

**Q.5)**

**Create function that validate the age of employee. Function accept the**

**dob of employee and return 1 if age is lies between 18 and 60 else re**

**turn 0**

**sol->**

delimiter $$

create function age1(age date)

returns int

begin

declare diff int;

set diff = extract(year from now()) - extract(year from age);

if diff >=18 and diff <=60 then

return 1;

else

return 0;

end if;

end$$![A description...](data:None;base64,)

**Q.6)Consider a following table of a database :**

**Book(bid, bname,authrname)**

**1. Create triggers which create a log of every Insert ,Delete and Upda**

**te operation on the book table record.**

**It should also hold the username who was operating at that time and t**

**ime and type of operation.**

**Delimiter $$**

**SOLUTION:**

create table Book(

bid varchar(20),

bname varchar(20),

authrname varchar(20)

);

create table log(

user varchar(20),

opertion varchar(20),

pbid varchar(20),

pbname varchar(20),

pauthname varchar(20),

nbid varchar(20),

nbname varchar(20),

nauthname varchar(20),

timeofop date

);

create trigger mytrigger after insert on Book for each row

begin

insert into log value ('root','insert','','','',new.bid,new.bname,new.authrname,now());

end$$

create trigger mytrigger1 after update on Book for each row

begin

insert into log value ('root','update',old.bid,old.bname,old.authrname,new.bid,new.bname,new.authrname,now());

end$$

create trigger mytrigger3 after delete on Book for each row

begin

insert into log value ('root','delete',old.bid,old.bname,old.authrname,'','','',now());

end$$

![A description...](data:None;base64,)

![A description...](data:None;base64,)

![A description...](data:None;base64,)