1. Suppose a movie_studio has several film crews. The crews might be designated by a given studio as crew1, crew2, and so on. However, other studios might use the same designations for crews, so the attribute crew_number is not a key for crews. Movie_studio holds the information like name, branch and several locations. Each crew holds information like sector, and strength.

    a. Establish the database by normalising up to 3NF and considering all schema level constraints.
    b. Write SQL insertion query to insert few tuples to all the relations.
    c. List all movies studios which are not used a single crews.
    d. Retrieve the movie_studio which uses highest strength crew.
    e. Write a before insert trigger to check maximum number of crews to any studio is limited to 5.
    f. Write a procedure retrieve all crews used by specific studio.

a)

```
create table movie
(
name      varchar(10),
branch    varchar(10),
constraint pki1 primary key(name)
);


create table crew
(
crew_no    number(10),
name       varchar(10),
strength   number(10),
sector     varchar(20),
constraint pki2 primary key(name,crew_no),
constraint fki1 foreign key(name) references movie(name)
);




create table locations
(
```

```
name       varchar(10),
location   varchar(20),
constraint pki3 primary key(name,location)
);

b) insert into movie values('&name','&branch');
insert into movie values('pixar','blore');
insert into movie values('warnerbro','pune');
insert into movie values('redchill','goa');
insert into movie values('legendary','usa');

insert into crew values(&crew_no,'&name',&strength,'&sector');
insert into crew values(11111,'pixar',14,'thrill');
insert into crew values(11112,'pixar',11,'sijd');
insert into crew values(11113,'pixar',6,'sakjs');
insert into crew values(11114,'pixar',2,'sawe');
insert into crew values(11115,'pixar',12,'ssesd');
insert into crew values(11115,'warnerbro',4,'ssesd');
insert into crew values(11112,'legendary',2,'thrill');
insert into crew values(11116,'pixar',11,'ssessd');
insert into crew values(11117,'pixar',11,'ssessd');

insert into locations values('&name','&location');
insert into locations values('pixar','pune');
insert into locations values('pixar','goa');
insert into locations values('legendary','blore');

c)
select name
from movie
where name not in (select name
                   from crew);
```

d)

```
select name
from crew
where strength >=all(select strength
                  from crew);


select name
from crew c
where c.strength in(select max(strength) from crew);
```

e)

```
create or replace trigger max_crew
before insert on crew
for each row
declare
  cnt number;
begin
select count(*) into cnt from crew
                           where name=:NEW.name;
if(cnt>5) then
raise_application_error(-20009,'MAX CREW LIMIT REACHED');
end if;
end;
/
```

f)

```
create or replace procedure display_crew(sname varchar)
is
X crew%rowtype;
cursor c is select c1.* from crew c1,movie m
                         where m.name=sname and c1.name=m.name;
begin
```

```
for X in c loop
sys.dbms_output.put_line(X.crew_no||'   '||X.strength||'    '||X.sector);
end loop;
end;
/
```

2. The production company is organised into different studios. We store each studio's name, branch and location; every studio must own at least one movie. We store each movie's title, censor_number and year of production. star may act in any number of movies and we store each actors name and address.

    a. Establish the database by normalising up to 3NF and considering all schema level constraints.
    b. Write SQL insertion query to insert few tuples to all the relations.
    c. List all the studios of the movie "xyz".
    d. List all the actors, acted in a movie "xyz".
    e. Write a procedure to list all movies produced during the specific year.
    f. Write a deletion trigger, does not allow to deleting the current year movies.

a)

```
create table studio
(
st_name        varchar(20),
branch         varchar(20),
location       varchar(20),
constraint  pk22 primary key(st_name));



create table movie1
(
sensor_no      varchar(20),
title         varchar(20),
year           number(5),
constraint  pk23 primary key(sensor_no)
);

create table star
```

```
(
star_name      varchar(20),
address        varchar(20),
constraint  pk24 primary key(star_name)
);

create table owns
(
st_name        varchar(20),
sensor_no      varchar(20),
constraint  pk25 primary key(st_name,sensor_no),
constraint  fk21 foreign key(st_name) references studio(st_name),
constraint  fk22 foreign key(sensor_no) references movie1(sensor_no)
);

create table acted_by
(
star_name      varchar(20),
sensor_no      varchar(20),
constraint  pk26 primary key(star_name,sensor_no),
constraint  fk23 foreign key(star_name) references star(star_name),
constraint  fk24 foreign key(sensor_no) references movie1(sensor_no)
);



b)
insert into studio values('&st_name','&branch','&location');
insert into studio values('pixar','blore','dsds');
insert into studio values('warnerbro','pune','wewew');
insert into studio values('legendary','new york','szxzx');

insert into movie1 values('&sensor_no','&title','&date');
insert into movie1 values('s1111','xyz',2011);
insert into movie1 values('s1112','dark knight',2009);
```

```
insert into movie1 values('s1113','hurt locker',2009);

insert into star values('&star_name','&address');
insert into star values('heath ledger','california');
insert into star values('caprio','LA');
insert into star values('clooney','NY');

insert into owns values('&st_name','&sensor_no');
insert into owns values('legendary','s1112');
insert into owns values('warnerbro','s1111');
insert into owns values('pixar','s1113');

insert into acted_by values('&star_name','&sensor_no');
insert into acted_by values('heath ledger','s1112');
insert into acted_by values('caprio','s1111');
insert into acted_by values('clooney','s1113');

c)
select s.st_name
from studio s,movie1 m,owns o
where m.title='xyz' and s.st_name=o.st_name and m.sensor_no=o.sensor_no;

d)
select s.star_name
from star s,acted_by a,movie1 m
where m.title='xyz' and s.star_name=a.star_name and m.sensor_no=a.sensor_no;

e)
create or replace procedure pr3(s number)
is
x movie1.title%type;
cursor c is select title
          from movie1 c
          where c.year=s;
```

```
begin
for x in c loop
sys.dbms_output.put_line(x.title);
end loop;
end;
/
```

```
f)
create or replace trigger tr2
before delete on movie1
for each row
declare
cur number;
begin
select to_char(sysdate,'YYYY') into cur
 from dual;
if(:OLD.year=cur) then
 raise_application_error(-20009,'Cannot delete');
end if;
end;
/
```

3. The production company is organised into different studios. We store each studio's name, branch and location; a studio own any number of cartoon-serials. We store each cartoon-serials's title, censor_number and year of production. star may do voices in any number of cartoon-serials and we store each actors name and address.

    a. Establish the database by normalising up to 3NF and considering all schema level constraints.
    b. Write SQL insertion query to insert few tuples to all the relations.
    c. Find total no. of actors, do voiced in a cartoon-serials "xyz".
    d. Retrieve name of studio, location and cartoon-serials title in which star "abc' is voiced.
    e. Write a procedure to list all cartoon-serials produced during the specific year.
    f. Write a deletion trigger, does not allow to deleting the current year cartoon-serials.

```
a)
create table studio1
```

```
(
st_name        varchar(20),
branch         varchar(20),
location       varchar(20),
constraint  pk51 primary key(st_name)
);


create table cartoon
(
sensor_no      varchar(20),
title         varchar(20),
year           number(5),
constraint  pk53 primary key(sensor_no)
);


create table star_cat
(
star_name      varchar(20),
address        varchar(20),
constraint  pk64 primary key(star_name)
);


create table owncat
(
st_name        varchar(20),
sensor_no      varchar(20),
constraint  pk55 primary key(st_name,sensor_no),
constraint  fk56 foreign key(st_name) references studio1(st_name),
constraint  fk57 foreign key(sensor_no) references cartoon(sensor_no)
);
```

```
create table voiced_by
(
star_name      varchar(20),
sensor_no      varchar(20),
constraint  pk58 primary key(star_name,sensor_no),
constraint  fk59 foreign key(star_name) references star_cat(star_name),
constraint  fk60 foreign key(sensor_no) references cartoon(sensor_no)
);

b)
insert into studio1 values('wb','pop','aa');
insert into studio1 values('gb','pio','dd');
insert into studio1 values('fg','uiy','kk');
insert into studio1 values('un','rus','pr');
insert into studio1 values('nb','rty','fg');

insert into cartoon values(201,'tj',1990);
insert into cartoon values(202,'ju',1991);
insert into cartoon values(203,'gt',1991);
insert into cartoon values(204,'tu',2012);
insert into cartoon values(206,'tr',1998);
insert into cartoon values(209,'tun',2012);

insert into star_cat values('james','dbn');
insert into star_cat values('tom','adjk');
insert into star_cat values('ross','jjk');
insert into star_cat values('peter','jkj');
insert into star_cat values('joe','ahfa');


insert into owncat values('wb',201);
insert into owncat values('gb',201);
insert into owncat values('fg',201);
insert into owncat values('nb',202);
```

```
insert into owncat values('nb',204);


insert into voiced_by values('james',201);

insert into voiced_by values('ross',201);

insert into voiced_by values('tom',203);

insert into voiced_by values('peter',206);

insert into voiced_by values('joe',201);


c)

select count(*)

from cartoon c,voiced_by v

where c.title='tj'and c.sensor_no=v.sensor_no;


d)

select s.st_name,s.location,c.title

from studio1 s,cartoon c,owncat o,voiced_by sr

where s.st_name=o.st_name and c.sensor_no=o.sensor_no and sr.star_name='james' and
sr.sensor_no=c.sensor_no;


e)

create or replace procedure pr3(s number)

is

x cartoon.title%type;

cursor c is select title

          from cartoon c

          where c.year=s;

begin

for x in c loop

sys.dbms_output.put_line(x.title);

end loop;

end;

/


f)
```

```
create or replace trigger tr3
before delete on cartoon
for each row
declare
cur number;
begin
select to_char(sysdate,'YYYY')into cur
 from dual;
if(:OLD.year=cur) then
 raise_application_error(-20019,'Cannot delete');
end if;
end;
/
```

4. Car marketing company wants keep track of marketed cars and their owner. Each car must be associated with a single owner and owner may have any number of cars. We store car's registration number, model & colour and owner's name, address & SSN. We also store date of purchase of each car.

    a. Establish the database by normalising up to 3NF and considering all schema level constraints.
    b. Write SQL insertion query to insert few tuples to all the relations.
    c. Find a person who owns highest number of cars.
    d. Retrieve persons and cars information purchased on day dd/mm/yyyy.
    e. Write a procedure to list all cars and owner information purchased during the specific year.
    f. Write a insertion trigger to check date of purchase must be less than current date (must use system date).

```
a)
create table owner2
(
ssn number(10),
name varchar(20),
address varchar(20),
constraint pk41 primary key(ssn)
);


create table car2
```

```
(
rgno number(10),
model varchar(10),
color varchar(10),
ssn number(10),
dop date,
constraint pk42 primary key(rgno),
constraint fk41 foreign key(ssn) references owner2(ssn)
);
```

b)

```
insert into owner2 values(&ssn,'&name','&address');
insert into owner2 values(4441,'josh','sfsdf');
insert into owner2 values(4442,'john','sfsdf');
insert into owner2 values(4443,'rose','sfsdf');
insert into owner2 values(4444,'robert','sfsdf');

insert into car2 values(&rgno,'&model','&color',&ssn,'&dop');
insert into car2 values(55551,'dfdf','red',4441,'10-jan-2011');
insert into car2 values(55552,'xcdf','black',4441,'11-nov-2011');
insert into car2 values(55553,'dfdfcx','blue',4441,'10-dec-2011');
insert into car2 values(55554,'asdfdf','white',4442,'10-jul-2011');
insert into car2 values(55555,'dfasdf','black',4443,'14-feb-2011');
```

c)

```
select o.name,c.ssn,count(c.ssn)
from owner2 o,car2 c
where o.ssn=c.ssn
group by o.name,c.ssn
having count(c.ssn) >=all(select count(m.ssn)
                          from car2 m
                          group by (m.ssn));
```

d)

```
select o.ssn,o.name,c.rgno,c.model
from owner2 o,car2 c
where c.ssn=o.ssn and dop='11-nov-2011';
```

e)

```
create or replace procedure pr4(pur_date date)
is
X owner2%rowtype;
X1 car2%rowtype;
cursor c is select o.* from owner2 o,car2 c1 where c1.ssn=o.ssn and dop=pur_date;
cursor f is select c1.* from owner2 o,car2 c1 where c1.ssn=o.ssn and dop=pur_date;
begin
sys.dbms_output.put_line('OWNER DETAILS');
for X in c loop
sys.dbms_output.put_line(X.name||'     '||X.ssn||'  '||X.address);
end loop;
sys.dbms_output.put_line('CAR DETAILS');
for X1 in f loop
sys.dbms_output.put_line(X1.rgno||'    '||X1.model||'      '||X1.color);
end loop;
end;
/
```

f)

```
create or replace trigger tr4
before insert on car2
for each row
declare
cur date;
begin
select sysdate into cur from dual;
if(cur<:NEW.dop) then
```

```
raise_application_error(-20009,'incorrect date');
end if;
end;
/
```

5. Puppy pet shop wants to keep track of dogs and their owners. The person can buy maximum three pet dogs. we store person's name, SSN and address and dog's name, date of purchase and sex. The owner of the pet dogs will be identified by SSN since the dog's names are not distinct.

    a. Establish the database by normalising up to 3NF and considering all schema level constraints.
    b. Write SQL insertion query to insert few tuples to all the relations.
    c. List all pets owned by a person "abhiman".
    d. List all persons who are not owned a single pet.
    e. Write a trigger to check the constraints that person can buy maximum three pet dogs.
    f. Write a procedure to list all dogs and owner details purchased on the specific date.

```
a)
create table ownerofdog
(
oname varchar(10),
ssn number(10),
address varchar(30),
constraint pk1 primary key(ssn));

create table doggg
(
ssn number(10),
dname varchar(10),
sex varchar(5),
dop date,
constraint pk2 primary key(ssn,dname),
constraint fk1 foreign key(ssn) references ownerofdog(ssn));

b)
insert into ownerofdog values('james',1234,'xyz');
```

```
insert into ownerofdog values('manoj',1235,'pqr');
insert into ownerofdog values('monika',1236,'yrs');
insert into ownerofdog values('gunda',1237,'rti ');
insert into ownerofdog values('harsh',1238,'rjs');
insert into ownerofdog values('Abhiman',1239,'stf');

insert into doggg values (1234,'ab','M','10-jan-2010');
insert into doggg values(1234,'bc','F','10-jan-2010');
insert into doggg values(1235,'de','M','20-feb-2009');
insert into doggg values(1236,'ef','F','21-mar-2010');
insert into doggg values(1237,'jp','F','22-mar-2010');
insert into doggg values(1239,'gh','M','09-jan-2010');
insert into doggg values(1239,'fu','F','10-jan-2011');

to check trigger

insert into doggg values(1239,'gr','F','12-jan-2011');
insert into doggg values(1239,'ks','M','14-jan-2012');

insert into doggg values (1234,'abd','M','10-jun-2011');
insert into doggg values(1234,'afg','F','09-jul-2012');

c)
select dname,sex,dop
from doggg d,ownerofdog o
where d.ssn=o.ssn and o.oname='Abhiman';

d)select o.oname
from ownerofdog o
where not exists( select *
                  from doggg d
                  where o.ssn=d.ssn);
e)
create or replace trigger tr5
```

```
before insert on doggg
for each row
declare
cnt number;
begin
select count(*) into cnt from ownerofdog o,doggg d
                      where (o.ssn=d.ssn and o.ssn=:NEW.ssn);
if(cnt>3) then
raise_application_error(-20010,'Capacity of the owner crossed');
end if;
end;


f)
create or replace procedure pr5(pur_date date)
is
X1 dog%rowtype;
X owner%rowtype;
cursor c is select o.* from ownerofdog o,doggg d where o.ssn=d.ssn and
dop=pur_date;
cursor f is select d.* from ownerofdog o,doggg d where o.ssn=d.ssn and
dop=pur_date;
begin
dbms_output.put_line('OWNER DETAILS');
for X in c loop
dbms_output.put_line(X.oname||'        '||X.ssn||'  '||X.address);
end loop;
dbms_output.put_line('DOG DETAILS');
for X1 in f loop
dbms_output.put_line(X1.dname||'       '||X1.ssn||' '||X1.sex);
end loop;
end;
/
```

6. Education institute is managing the online course enrolment system. Students can enrol maximum of six courses of their choice and a maximum student to be enrolled to any course is 60. We store student details like name, USN, semester and several addresses, course details like unique title, unique id and credits.

   a. Establish the database by normalising up to 3NF and considering all schema level constraints.
   b. Write SQL insertion query to insert few tuples to all the relations.
   c. Find number of students enrolled for the course 'DBMS'.
   d. Retrieve student names that are enrolled for data structure course but not enrolled for logic design.
   e. Write a trigger to establish the constraint that the students can enrol maximum of six course of their choice.
   f. Write a procedure to list all the courses enrolled by the seventh semester students.

a)

```
create table stud1
(
	usn number(10),
	name varchar(20),
	sem varchar(10),
	constraint pkk primary key(usn)
);


create table course
(
	title varchar(20) unique,
	cid number(10),
	credits number(10),
	constraint pk44 primary key(cid)
);


create table enroll
(
	usn number(10),
	cid number(10),
```

```
	constraint pk54 primary key(usn,cid),
	constraint fk44 foreign key(cid) references course(cid),
	constraint fk54 foreign key(usn) references stud1(usn)
);
```

b)

```
insert into stud1 values(901,7,'josh');
insert into stud1 values(902,5,'guru');
insert into stud1 values(903,7,'srini');
insert into stud1 values(904,3,'turre');
insert into stud1 values(905,3,'rupa');
insert into stud1 values(906,5,'piggi');

insert into stud1 values(NULL,6,'jos');



insert into course values('dbms',301,4);
insert into course values('ld',302,5);
insert into course values('ds',303,4);
insert into course values('oops',304,4);
insert into course values('ada',305,3);
insert into course values('se',306,4);
insert into course values('cn',307,4);


insert into enroll values(901,301);
insert into enroll values(902,301);
insert into enroll values(903,303);
insert into enroll values(904,304);
insert into enroll values(901,302);
insert into enroll values(901,303);
insert into enroll values(901,304);
insert into enroll values(901,305);
```

```
insert into enroll values(903,305);

insert into enroll values(901,306);
insert into enroll values(901,307);

c)
select count(s.usn)
from stud1 s,course c,enroll e
where s.usn=e.usn and e.cid=c.cid and c.title='dbms';

d)
select s.name
from stud1 s,course c,enroll e
where s.usn=e.usn and e.cid=c.cid and c.title='ds'
minus
select s.name
from stud1 s,course c,enroll e
where s.usn=e.usn and e.cid=c.cid and c.title='ld';

e)
create or replace trigger tr6
before insert on enroll
for each row
declare
var1 number;
begin
select count(*) into var1 from enroll where usn=:new.usn;
if(var1>6)
then
raise_application_error(-20009,'limit reached');
end if;
end;
/
```

```
f)

create or replace procedure pr6

is

X course%rowtype;

cursor c is select c1.title from course c1, stud1 s,enroll e where s.usn=e.usn and
c1.cid=e,cid ans s.sem='7';

begin

for X in c loop

sys.dbms_output.put_line(c.name);

end loop;

end;

/
```

7. The commercial bank wants to keep track of the customer's account information. Each customer may have any number of accounts and account can be shared by any number of customers. The system will keep track of the date of last transaction. We store the following details:
   i. account: unique account number, type and balance.
   ii. customer: unique customer id, name and several addresses composed of street, city and state.

   a. Establish the database by normalising up to 3NF and considering all schema level constraints.
   b. Write SQL insertion query to insert few tuples to all the relations.
   c. Add 3% interest to the customer who have less than 10000 balances and 6% interest to remaining customers.
   d. List joint accounts involving more than three customers.
   e. Write a insertion trigger to allow only current date for date of last transaction field.
   f. Write a procedure to find the customer who has highest number of accounts, the customer who has lowest balance, the customer who involved in most of joint accounts.

```
   a) create table customer

(

c_id              varchar(10),

name            varchar(10),
```

```
constraint    pk71 primary key(c_id)
);

create table account
(
acc_no          number(10),
type               varchar(10),
balance           number(100000),
constraint   pk72 primary key(acc_no)
);

create table cust_acc
(
c_id               varchar(10),
acc_no          number(10),
last_tr           date,
constraint  pk73 primary key(c_id,acc_no),
constraint  fk71 foreign key(c_id) references customer(c_id),
constraint  fk72 foreign key(acc_no) references account(acc_no)
);

create table address
(
c_id               varchar(10),
street            varchar(20),
city        varchar(20),
state        varchar(20),
constraint pk74 primary key(c_id,street,city,state)
);

b)
insert into customer values('&c_id','&name');
insert into customer values('c1111','alex');
insert into customer values('c1112','john');
```

```
insert into customer values('c1113','steve');
insert into customer values('c1114','robert');


insert into account values(&acc_no,'&type','&balance');
insert into account values(2001,'savings','8000');
insert into account values(2002,'RD','9000');
insert into account values(2003,'joint','20000');

insert into cust_acc values('&c_id',&acc_no,'&last_tr');
insert into cust_acc values('c1111',2003,'20-jan-2011');
insert into cust_acc values('c1112',2003,'12-feb-2011');
insert into cust_acc values('c1113',2003,'13-jan-2011');
insert into cust_acc values('c1114',2003,'20-mar-2011');
insert into cust_acc values('c1111',2001,'20-dec-2011');

insert into address values('&c_id','&street','&city','&state');
insert into address values('c1111','fsfsf','ccxx','&hghg');
insert into address values('c1111','fsfcc','ccqwq','&hgcxhg');
```

c)

```
update account
set balance=case
when balance<=10000 then balance*1.05
else balance*1.06
end;
```

d)

```
select max(distinct(count(acc_no)))
from account
group by acc_no;
```

e)

```
create or replace trigger last_trans
before insert on cust_acc
for each row
declare
cur_date date;
begin
select sysdate into cur_date from dual;
if((:NEW.last_tr-cur_date)<=0) then
raise_application_error(-20006,'INVALID DATE');
end if;
end;
```

f)

```
create or replace procedure pr7()
is
X customer%rowtype;

cursor c is
select ca.c_id,c1.name,count(ca.acc_no)
from customer c1, cust_acc ca
where c1.c_id=ca.c_id
group by ca_cid,c1.name
having count(ca.acc_no) >= all(select count(ca.acc_no)
from cust_acc ca
group by ca.c_id;

cursor f is
select name from customer c, account a and cust_acc ca
where c.c_id=ca.c_id and  a.acc_no=ca.acc_no and a.balance in(select max(balance)
from account);
```

```
begin
sys.dbms_output.put_line('CUSTOMER DETAILS WITH MAX ACCOUNTS');
for X in c loop
sys.dbms_output.put_line(X.c_id||'     '||X.name);
end loop;
sys.dbms_output.put_line('CUSTOMER DETAILS WITH LOWEST BALANCE');
for X in f loop
sys.dbms_output.put_line(X.c_id||'     '||X.name);
end loop;
end;
/
```

8. The commercial bank wants to keep track of the customer's loan information. Each customer can take any number of loans from the bank and loan will not be shared. The system will keep track of the date of last transaction. We store the following details:
   i. loan: unique loan number, type and amount.
   ii. customer: unique customer id, name, annual income and several addresses composed of street, city and state.

   a. Establish the database by normalising up to 3NF and considering all schema level constraints.
   b. Write SQL insertion query to insert few tuples to all the relations.
   c. Add 12% interest to the customer who have less than 50000 amount and 14% interest to remaining customers.
   d. Retrieve the customers who have single loan in a bank.
   e. Write a insertion trigger to loan, that does not allow if the loan amount is more than two times of customers annual income.
   f. Write a procedure to retrieve all the loans of a specific customer.

```
Refer/get idea for A) and b) from previous question.
c)
update loans
set amount=case
when amount<=50000 then balance*1.12
else balance*1.14
end;
```

d)

```
select c.cname,l.cid,count(l.lno)
from customer c, loans l
where c.cid=l.cid
group by c.cname,l.cid
having count(lno=1);
```

e)

```
create or replace trigger tr8
before insert on loans
for each row
begin
select cincome from customer c where c.cid=:NEW.cid;
if(:NEW.amount > 2*cincome) then
raise_application_error(-20009,'LIMIT EXCEEDED');
end if;
end;
/
```

f)

```
create or replace procedure pr8(name varchar)
is
X loans%rowtype;
cursor c is select l.* from customer c1, loans l where c1.cname=name and
l.cid=c1.cid;
begin
sys.dbms_output.put_line('LOAN DETAILS');
for X in c loop
sys.dbms_output.put_line(X.lno||'      '||X.type||' '||X.amount);
end loop;
end;
/
```

9. The xyz book shop wants keep track of orders of the book. The book is composed of unique id, title, year of publication, single author and single publisher. Each order will be uniquely identified by order-id and may have any number of books. We keep track of quantity of each book ordered. We store the following details:
   i. author: unique author-id, name, city, country.
   ii. Publisher: unique publisher-id, name, city, country.

   a. Establish the database by normalising up to 3NF and considering all schema level constraints.
   b. Write SQL insertion query to insert few tuples to all the relations.
   c. Find the author who has published highest number of books.
   d. List the books published by specific publisher during the year 2011.
   e. Write a insertion trigger to book to check year of publication should allow current year only.
   f. Write a procedure to list all books published by a specific author during the specific year.

a)

```
create table order1(
orderid varchar(10),
constraint order_pk primary key(orderid)
);

create table book(
bid varchar(10),
btitle varchar(10),
byop number,
aid varchar(10),
pid varchar(10),
constraint book_pk primary key(bid),
constraint book_fk1 foreign key(aid) references author(aid),
constraint book_fk2 foreign key(pid) references publisher(pid)
);

create table author(
aid varchar(10),
aname varchar(10),
```

```
acity varchar(10),
acountry varchar(10),
constraint author_pk primary key(aid)
);

create table publisher(
pid varchar(10),
pname varchar(10),
pcity varchar(10),
pcountry varchar(10),
constraint publisher_pk primary key(pid)
);

create table order_book(
orderid varchar(10),
bid varchar(10),
constraint ob_pk primary key(orderid,bid),
constraint ob_fk1 foreign key(bid) references book(bid),
constraint ob_fk2 foreign key(orderid) references order1(orderid)
);



b)
insert into order1 values('1');
insert into order1 values('2');

insert into book values('1','abc','2001','1','1');
insert into book values('2','xyz','2001','1','1');
insert into book values('3','mno','2001','2','1');

insert into author values('1','a','b','c');
insert into author values('2','a','b','c');

insert into publisher values('1','a','b','c');
```

```
insert into order_book values('1','1');
```

c)

```
select a.aname,b.aid,count(b.bid)
from author a,book b
where a.aid=b.aid
group by a.aname,b.aid
having count(b.bid)>=all(select count(b.bid)
from book b
group by b.aid);
```

d)

```
select btitle
from book
where byop='2001' and pid='1';
```

e)

```
create or replace trigger tr9
before insert on book
for each row
declare
cur number;
begin
select to_char(sysdate,'YYYY')into cur
 from dual;
if(:NEW.byop!=cur) then
 raise_application_error(-20019,'Cannot insert');
end if;
end;
/
```

f)

```
create or replace procedure pr9(year number,name varchar)
is
X book%rowtype;
cursor c is select b.* from book b,author a where a.aname=name and b.byop=year and
a.aid=b.aid;
begin
sys.dbms_output.put_line('BOOK DETAILS');
for X in c loop
sys.dbms_output.put_line(X.bid||'      '||X.btitle||'      '||X.byop);
end loop;
end;
/
```

Disclaimer

```
I am not responsible for wrong answers, system errors,
thermonuclear war, or you getting fired because the database failed.
Please do some research if you have any concerns about answers included
in this document before reading it! YOU are choosing to read this
document and if you point the finger at me for messing up your marks,
 I will laugh at you. although I did my BEST.
```