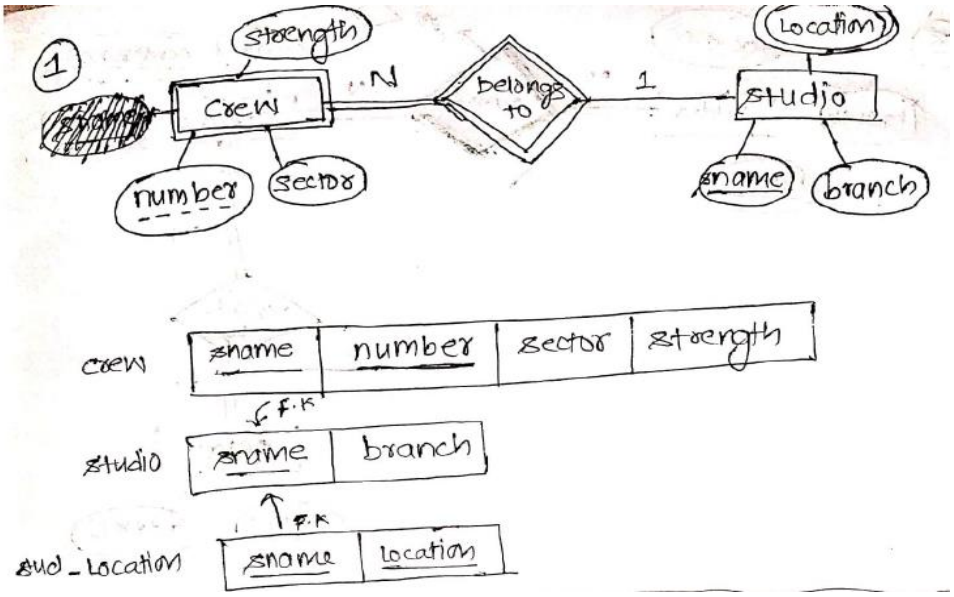
1. Suppose a movie_studio has several film crews. The crews might be designated by a given studio as crew1, crew2, and so on. However, other studios might use the same designations for crews, so the attribute crew_number is not a key for crews. Movie_studio holds the information like name, branch and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




```
 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. Write SQL insertion query to insert few tuples to all the relations.

```
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');
```

Table: movie

name	branch
pixar	blore
warnerbro	pune
redchill	goa
legendary	usa

Table: crew

crew_no	name	strength	sector
11111	pixar	14	thrill
11112	pixar	11	sijd
11113	pixar	6	sakjs
11114	pixar	2	sawe
11115	pixar	12	ssesd
11115	warnerbro	4	ssesd
11112	legendary	2	thrill
11116	pixar	11	ssessd
11117	pixar	11	ssessd

Table: locations

name	location
pixar	pune
pixar	goa
legendary	blore

c. List all movies studios which are not used a single crews.

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

Output:

name
pixar
warnerbro
legendary

name
redchill

d. Retrieve the movie_studio which uses highest strength crew.

```
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);
```

crew_no	name	strength	sector
11111	pixar	14	thrill
11112	pixar	11	sijd
11113	pixar	6	sakjs
11114	pixar	2	sawe
11115	pixar	12	ssesd
11115	warnerbro	4	ssesd
11112	legendary	2	thrill
11116	pixar	11	ssessd
11117	pixar	11	ssessd

name
pixar

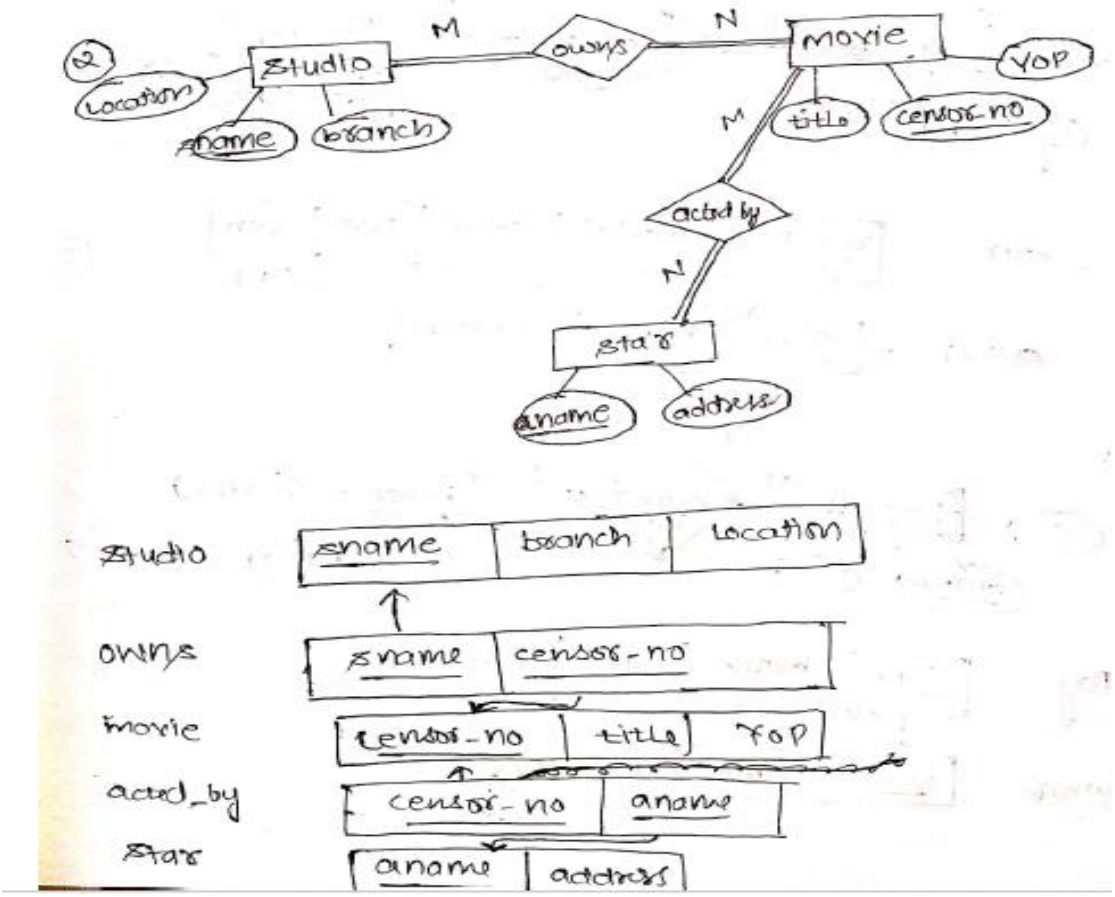
2. The production company is organised into different studios. We store each studio's name, branch and location; every studio must own at least one movie. We store each movie's title, censor_number and year of production. star may act in any number of movies and we store each actors name and address.

a. **Establish the database by normalising up to 3NF and considering all schema level constraints.**

```
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. Write SQL insertion query to insert few tuples to all the relations.

```
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');
```

Table: studio

st_name	branch	location
pixar	blore	dsds
warnerbro	pune	wewew
legendary	new york	szxzx

Table: movie1

sensor_no	title	year
s1111	xyz	2011
s1112	dark knight	2009
s1113	hurt locker	2009

Table: star

star_name	address
heath ledger	california
caprio	LA
clooney	NY

Table: owns

st_name	sensor_no
legendary	s1112
warnerbro	s1111
pixar	s1113

Table: acted_by

star_name	sensor_no
heath ledger	s1112
caprio	s1111
clooney	s1113

c. List all the studios of the movie "xyz".

```
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;
```

movie1 Table:

sensor_no	title	year
s1111	xyz	2011
s1112	dark knight	2009
s1113	hurt locker	2009

owns Table:

st_name	sensor_no
legendary	s1112
warnerbro	s1111
pixar	s1113

st_name
warnerbro

d. List all the actors, acted in a movie "xyz".

```
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;
```

movie1 Table:

sensor_no	title	year
s1111	xyz	2011
s1112	dark knight	2009
s1113	hurt locker	2009

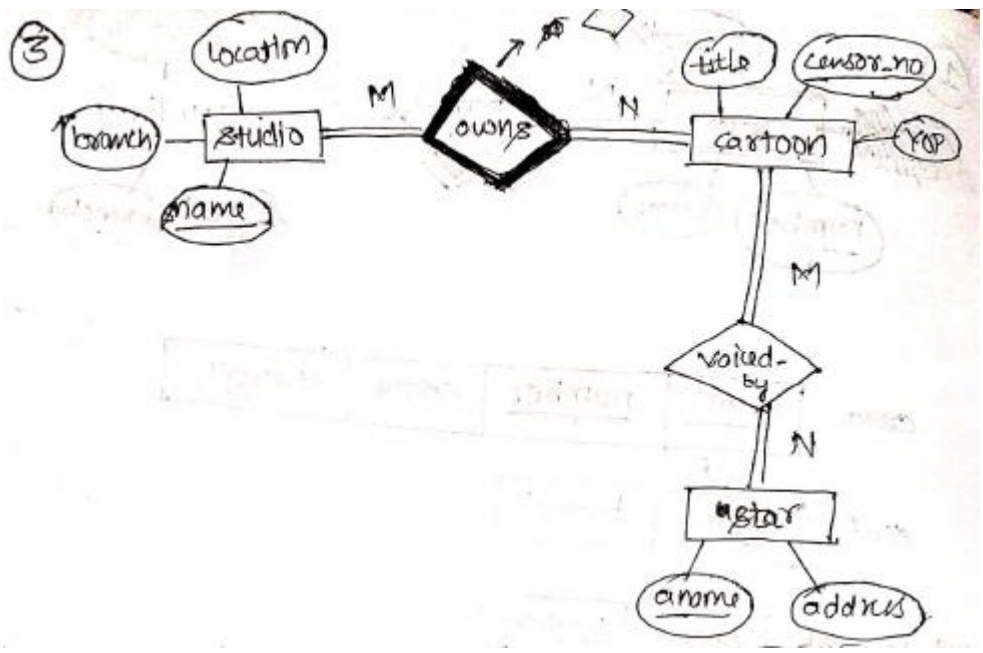
acted_by Table:

star_name	sensor_no
heath ledger	s1112
caprio	s1111
clooney	s1113

star_name
caprio

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.**




```
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. Write SQL insertion query to insert few tuples to all the relations.

```
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);
```

Table: studio1

st_name	branch	location
wb	pop	aa
gb	pio	dd
fg	uiy	kk
un	rus	pr
nb	rty	fg

Table: cartoon

sensor_no	title	year
201	tj	1990
202	ju	1991
203	gt	1991
204	tu	2012
206	tr	1998
209	tun	2012

Table: star_cat

star_name	address
james	dbn
tom	adjk
ross	jjk
peter	jkj
joe	ahfa

Table: owncat

st_name	sensor_no
wb	201
gb	201
fg	201
nb	202
nb	204

Table: voiced_by

star_name	sensor_no
james	201
ross	201
tom	203
peter	206
joe	201

c. Find total no. of actors, do voiced in a cartoon-serials “xyz”.

```
select count(*)
from cartoon c,voiced_by v
where c.title='tj'and c.sensor_no=v.sensor_no;
```

1. **Cartoon Table**: The relevant entry based on the previous insertions is:

sensor_no	title	year
201	tj	1990

2. **Voiced_by Table**: The entries in this table are:

star_name	sensor_no
james	201
ross	201
tom	203
peter	206
joe	201

count(*)
3

d. Retrieve name of studio, location and cartoon-serials title in which star "abc' is voiced.

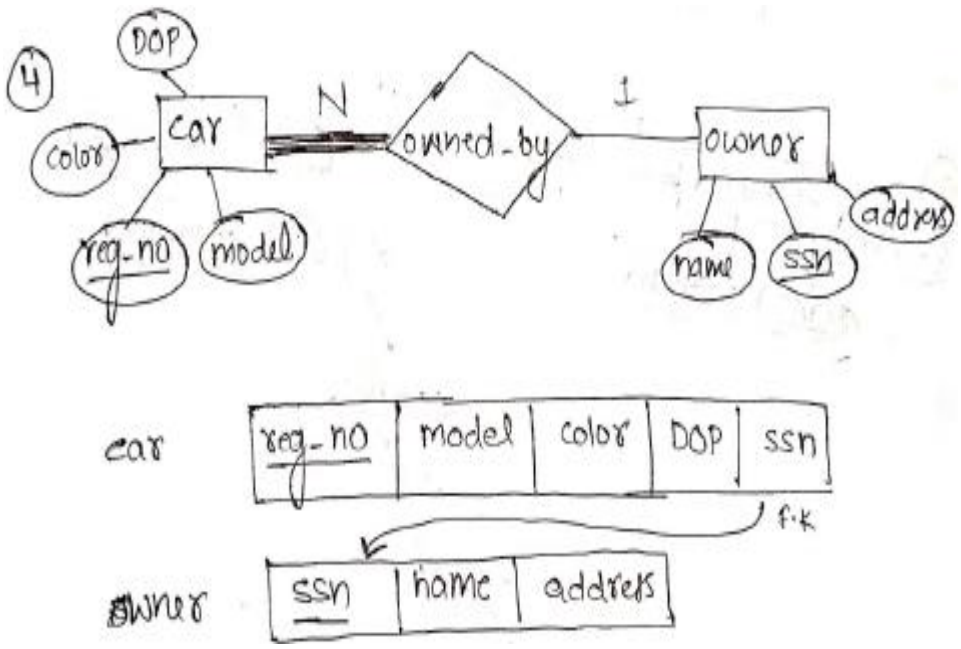
```
select s.st_name,s.location,c.title
from studio1 s,cartoon c,owncat o,voiced_by sr
where s.st_name=o.st_name and c.sensor_no=o.sensor_no and sr.star_name='james'
and sr.sensor_no=c.sensor_no;
```

- **voiced_by Table:**
  - `james` is associated with `sensor_no = 201`.
- **owncat Table:**
  - Cartoons owned with `sensor_no = 201`:
    - wb
    - gb
    - fg
- **studio1 Table**:
  - wb: location = pop
  - gb: location = pio
  - fg: location = uiy
- **cartoon Table:**
  - `sensor_no = 201` → title = tj

st_name	location	title
wb	pop	tj
gb	pio	tj
fg	uiy	tj

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.**




```
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. **Write SQL insertion query to insert few tuples to all the relations.**

```
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. Find a person who owns highest number of cars.

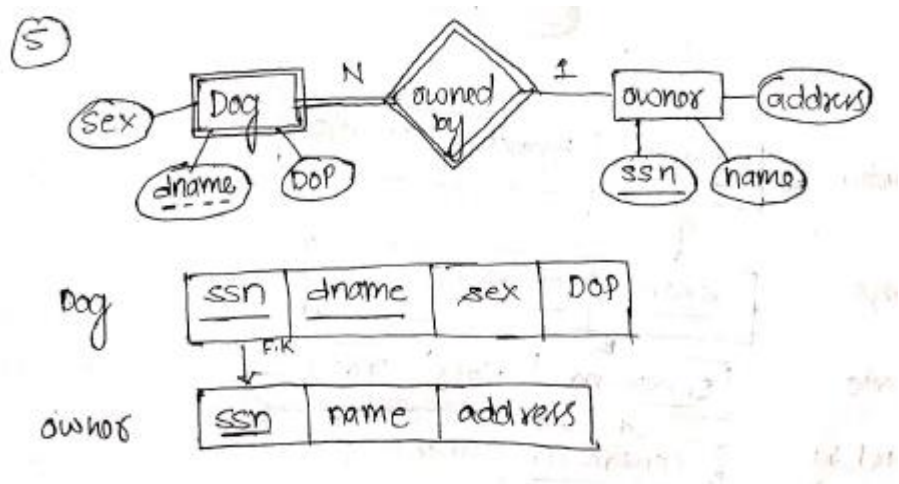
```
select o.name,c.ssn,count(c.ssn)
from owner2 o,car2 c
where o.ssn=c.ssn
group by o.name,c.ssn
having count(c.ssn) >=all(select count(m.ssn)
from car2 m
group by (m.ssn));
```

d. Retrieve persons and cars information purchased on day dd/mm/yyyy.

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

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.




```
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. Write SQL insertion query to insert few tuples to all the relations.

```
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');
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. List all pets owned by a person “abhiman”.

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

d. List all persons who are not owned a single pet.

```
select o.oname
from ownerofdog o
where not exists( select *
from doggg d
where o.ssn=d.ssn);
```