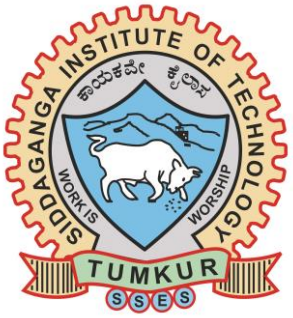
Database Management Systems Lab Report

Submitted for the partial fulfillment of Bachelor of Engineering

By

S Niveditha 1SI21CS088



Department of Computer Science and Engineering

(Program Accredited by NBA)

Siddaganga Institute of Technology, Tumakuru – 572103

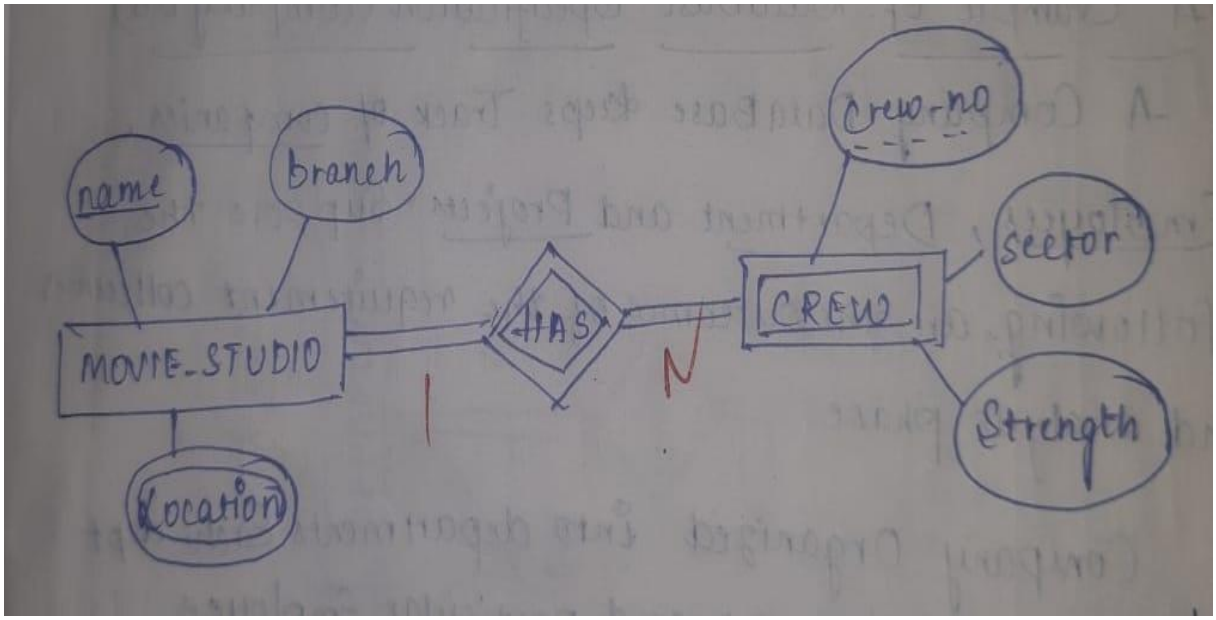
(An autonomous institution affiliated to VTU, Belagavi, Approved by AICTE, New Delhi, Accredited by NAAC with 'A++' grade & ISO 9001:2015 Certified)

2023-2024

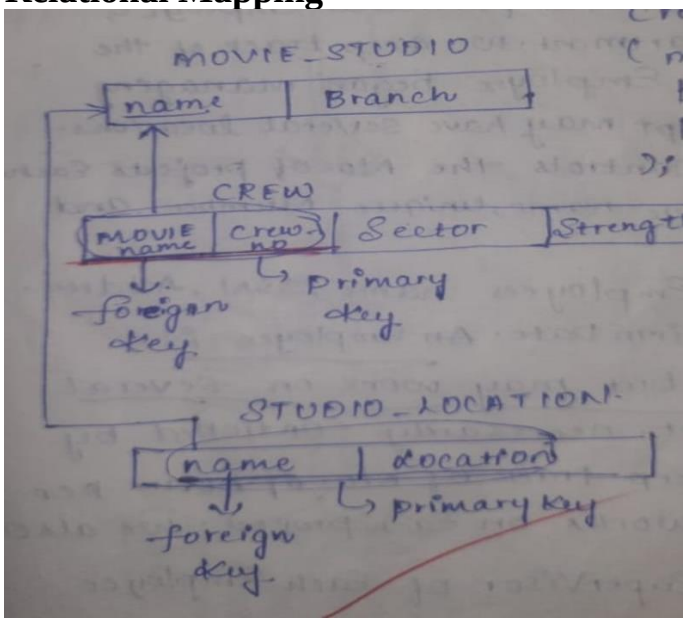
Problem Title

1.Suppose a movie_studio has several film crews. The crews might be designated by a given studio as crew1, crew 2, 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.

ER Model



Relational Mapping



DDL statements

CREATING TABLES:

movie_studio:

```
create table movie_studio(
movie_name varchar(25),
branch varchar(25),
primary key(movie_name));
```

```
mysql> desc movie_studio;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| movie_name | varchar(25) | NO   | PRI | NULL    |       |
| branch     | varchar(25) | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
2 rows in set (0.07 sec)
```

crew:

```
create table crew(
sector int,
strength int,
crew_no int,
movie_name varchar(25),
primary key(crew_no,movie_name),
foreign key(movie_name) references movie_studio(movie_name));
```

```
ysql> desc crew;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| sector     | int         | YES  |     | NULL    |       |
| strength   | int         | YES  |     | NULL    |       |
| crew_no    | int         | NO   | PRI | NULL    |       |
| movie_name | varchar(25) | NO   | PRI | NULL    |       |
+------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

studio_location:

create table studio_location(

movie_name varchar(25),

location varchar(25),

primary
key(movie_name,location),

foreign key(movie_name)
references
movie_studio(movie_name));

```
mysql> desc studio_location;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| movie_name | varchar(25) | NO   | PRI | NULL    |       |
| location   | varchar(25) | NO   | PRI | NULL    |       |
+------------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

Insert Statements

INSERTING VALUES INTO TABLES

movie_studio:

insert into movie_studio values("HiNanna","B1");

insert into movie_studio values("Guntur Karam","B2");

insert into movie_studio values("Hanuman","B4");
insert into movie_studio values("RRR","B3");

insert into movie_studio values("MEB","B5");

```
mysql> select * from movie_studio;
+--------------+--------+
| movie_name   | branch |
+--------------+--------+
| Guntur Karam | B2     |
| Hanuman      | B4     |
| HiNanna      | B1     |
| MEB          | B5     |
| RRR          | B3     |
+--------------+--------+
5 rows in set (0.00 sec)
```

crew:

insert into crew values(2,500,1,"HiNanna");

insert into crew values(3,1678,1,"Guntur Karam");

insert into crew values(4,789,3,"RRR");

insert into crew values(1,567,5,"RRR");

insert into crew values(6,678,1,"MEB");

```
mysql> select * from crew;
+--------+----------+---------+--------------+
| sector | strength | crew_no | movie_name   |
+--------+----------+---------+--------------+
|      3 |     1678 |       1 | Guntur Karam |
|      2 |      500 |       1 | HiNanna      |
|      6 |      678 |       1 | MEB          |
|      4 |      789 |       3 | RRR          |
|      1 |      567 |       5 | RRR          |
+--------+----------+---------+--------------+
5 rows in set (0.00 sec)
```

studio_location:

insert into studio_location values("Guntur Karam","Hindupur");

insert into studio_location values("Guntur Karam","Hyderabad");

insert into studio_location values("Hanuman","Hindupur");

insert into studio_location values("MEB","Anantapur");

insert into studio_location values("RRR","Vijaywada");

insert into studio_location values("HiNanna","Tumkur");

```
mysql> select * from studio_location;
+--------------+-----------+
| movie_name   | location  |
+--------------+-----------+
| Guntur Karam | Hindupur  |
| Guntur Karam | Hyderabad |
| Hanuman      | Hindupur  |
| HiNanna      | Tumkur    |
| MEB          | Anantapur |
| RRR          | Vijaywada |
+--------------+-----------+
6 rows in set (0.00 sec)
```

Queries:

1. **List all movie studios which are not used a single crews.**

select movie_name from movie_studio

where movie_name not in

(select movie_name from crew);

```
mysql> select movie_name from movie_studio
    -> where movie_name not in
    -> (select movie_name from crew);
+------------+
| movie_name |
+------------+
| Hanuman    |
+------------+
1 row in set (0.00 sec)
```

2. **Reterieve the movie studio which uses highest strength crew.**

select movie_name from crew

where strength>=ALL

(select strength from crew);

```
mysql> select movie_name from crew
    -> where strength>=ALL
    -> (select strength from crew);
+--------------+
| movie_name   |
+--------------+
| Guntur Karam |
+--------------+
1 row in set (0.01 sec)
```

Stored procedure:

Write a procedure to retrieve all crews used by specificstudio

```
DELIMITER //

CREATE PROCEDURE display_crew(IN sname VARCHAR(25))BEGIN

DECLARE done BOOLEAN DEFAULT FALSE;

DECLARE crew_no_var INT;

DECLARE strength_var INT;

DECLARE sector_var VARCHAR(25);

DECLARE c CURSOR FOR

SELECT c1.crew_no,c1.strength,c1.sector

FROM crew c1

JOIN movie_studio m ON

c1.movie_name=m.movie_nameWHERE

m.movie_name=sname;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done=TRUE;

OPEN c;

read_loop:LOOP

FETCH c INTO crew_no_var,strength_var,sector_var;

IF done THEN

LEAVE read_loop;
```

```
END IF;

SELECT CONCAT(crew_no_var,' ',strength_var,' ',sector_var) AS result;

END LOOP;

CLOSE c;

END //

DELIMITER;
```

```
mysql> call display_crew("RRR");
+---------+
| result  |
+---------+
| 3 789 4 |
+---------+
1 row in set (0.02 sec)

+---------+
| result  |
+---------+
| 5 567 1 |
+---------+
1 row in set (0.04 sec)
```

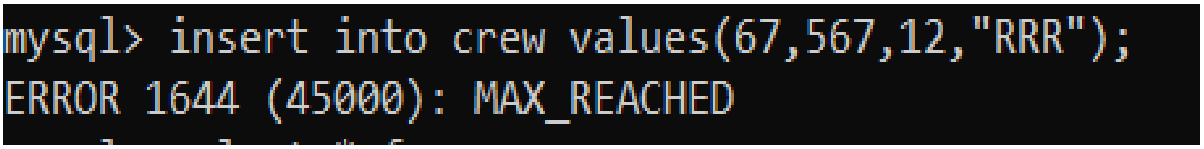
Trigger:

Write a before insert trigger to check maximum number of crews to any studio is limited to 5.

Inserting 5 crews for same movie:

```
insert into crew values(4,789,7,"MEB");

insert into crew values(1,567,8,"MEB");

insert into crew values(34,789,1,"MEB");

insert into crew values(67,567,2,"MEB");

insert into crew values(67,567,12,"");
```

```
DELIMITER //
CREATE TRIGGER maxi
BEFORE INSERT ON crew
FOR EACH ROW
BEGIN
DECLARE cnt INT;
SELECT COUNT(*) INTO cnt FROM crew
WHERE movie_name=NEW.movie_name;
IF cnt>5 THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT='MAX_reached';
END IF;
END //
DELIMETER ;
```



MongoDB

Create “crew” collection and perform the following CRUD operations:
a) Create a document.
b) Create two or more documents at the same time.
c) Update a document with crew number 111111 to 20.
d) Delete all the crews with strength 11.
e) Retrieve the crews with strength greater than or equal to 10.

```
> db.createCollection("crew")
{ "ok" : 1 }
```

```
> db.crew.insertOne({crew_no:10,strength:15,sector:"Thriller"});
{
        "acknowledged" : true,
        "insertedId" : ObjectId("65c311ac78f5114e73bdaa0a")
}
```

```
db.crew.insertMany([
{crew_no:20,strength:20,sector:"Thriller"},
{crew_no:25,strength:22,sector:"action"},
{crew_no:20,strength:20,sector:"action"}]);
db.crew.insertMany([
 {crew_no:10,strength:25,sector:"Thriller"},
 {crew_no:10,strength:24,sector:"action"}]);
db.crew.insertMany([
{crew_no:25,strength:10,sector:"Thriller"},
{crew_no:24,strength:10,sector:"action"}]);
```

```
> db.crew.insertMany([
... {crew_no:20,strength:20,sector:"Thriller"},
... {crew_no:25,strength:22,sector:"action"},
... {crew_no:20,strength:20,sector:"action"}]);
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("65c3143f31edff1358b73493"),
                ObjectId("65c3143f31edff1358b73494"),
                ObjectId("65c3143f31edff1358b73495")
        ]
}
```

```
> db.crew.insertMany([
... {crew_no:10,strength:25,sector:"Thriller"},
... {crew_no:10,strength:24,sector:"action"}]);
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("65c3152d31edff1358b73496"),
                ObjectId("65c3152d31edff1358b73497")
        ]
}
```

```
> db.crew.insertMany([
... {crew_no:25,strength:10,sector:"Thriller"},
... {crew_no:24,strength:10,sector:"action"}]);
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("65c315ab31edff1358b73498"),
                ObjectId("65c315ab31edff1358b73499")
        ]
}
```

```
> db.crew.find()
{ "_id" : ObjectId("65c311ac78f5114e73bdaa0a"), "crew_no" : 20, "strength" : 15, "sector" : "Thriller" }
{ "_id" : ObjectId("65c3143f31edff1358b73493"), "crew_no" : 20, "strength" : 20, "sector" : "Thriller" }
{ "_id" : ObjectId("65c3143f31edff1358b73494"), "crew_no" : 25, "strength" : 22, "sector" : "action" }
{ "_id" : ObjectId("65c3143f31edff1358b73495"), "crew_no" : 20, "strength" : 20, "sector" : "action" }
{ "_id" : ObjectId("65c3152d31edff1358b73496"), "crew_no" : 10, "strength" : 25, "sector" : "Thriller" }
{ "_id" : ObjectId("65c3152d31edff1358b73497"), "crew_no" : 10, "strength" : 24, "sector" : "action" }
{ "_id" : ObjectId("65c315ab31edff1358b73498"), "crew_no" : 25, "strength" : 10, "sector" : "Thriller" }
{ "_id" : ObjectId("65c315ab31edff1358b73499"), "crew_no" : 24, "strength" : 10, "sector" : "action" }
```

db.crew.updateOne(
{crew_no:10},
{$set:{crew_no:20}});

```
> db.crew.find()
{ "_id" : ObjectId("65c311ac78f5114e73bdaa0a"), "crew_no" : 20, "strength" : 15, "sector" : "Thriller" }
{ "_id" : ObjectId("65c3143f31edff1358b73493"), "crew_no" : 20, "strength" : 20, "sector" : "Thriller" }
{ "_id" : ObjectId("65c3143f31edff1358b73494"), "crew_no" : 25, "strength" : 22, "sector" : "action" }
{ "_id" : ObjectId("65c3143f31edff1358b73495"), "crew_no" : 20, "strength" : 20, "sector" : "action" }
```

db.crew.deleteMany({strength:10});

```
> db.crew.find()
{ "_id" : ObjectId("65c311ac78f5114e73bdaa0a"), "crew_no" : 20, "strength" : 15, "sector" : "Thriller" }
{ "_id" : ObjectId("65c3143f31edff1358b73493"), "crew_no" : 20, "strength" : 20, "sector" : "Thriller" }
{ "_id" : ObjectId("65c3143f31edff1358b73494"), "crew_no" : 25, "strength" : 22, "sector" : "action" }
{ "_id" : ObjectId("65c3143f31edff1358b73495"), "crew_no" : 20, "strength" : 20, "sector" : "action" }
{ "_id" : ObjectId("65c3152d31edff1358b73496"), "crew_no" : 10, "strength" : 25, "sector" : "Thriller" }
{ "_id" : ObjectId("65c3152d31edff1358b73497"), "crew_no" : 10, "strength" : 24, "sector" : "action" }
```

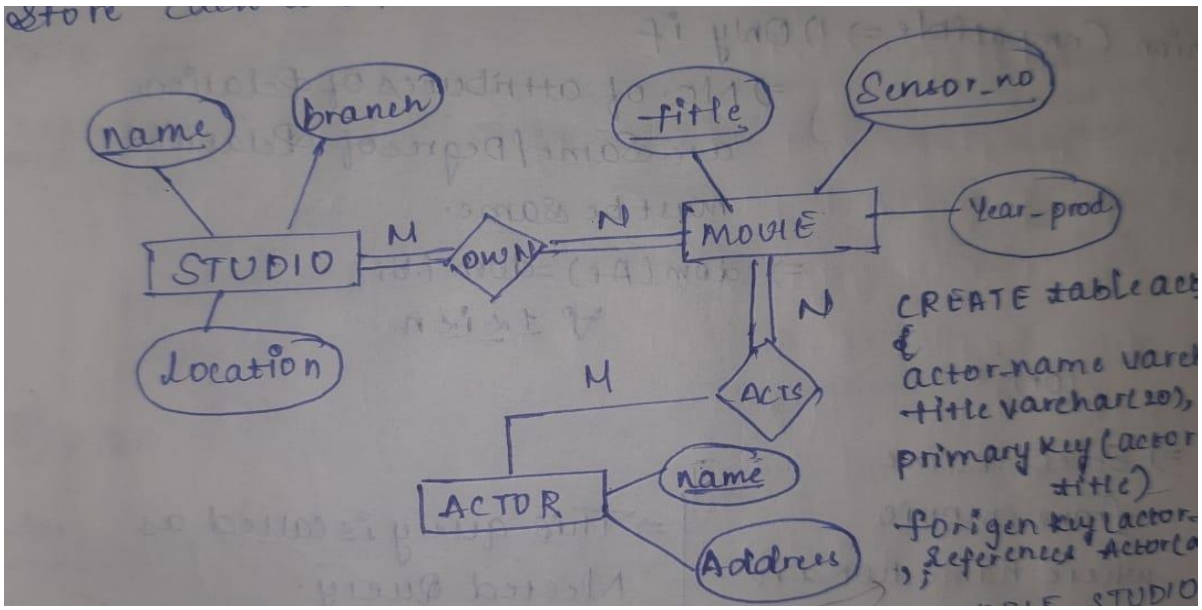
db.crew.find({strength:{$gte:20}});

```
> db.crew.find({strength:{$gte:20}});
{ "_id" : ObjectId("65c3143f31edff1358b73493"), "crew_no" : 20, "strength" : 20, "sector" : "Thriller" }
{ "_id" : ObjectId("65c3143f31edff1358b73494"), "crew_no" : 25, "strength" : 22, "sector" : "action" }
{ "_id" : ObjectId("65c3143f31edff1358b73495"), "crew_no" : 20, "strength" : 20, "sector" : "action" }
{ "_id" : ObjectId("65c3152d31edff1358b73496"), "crew_no" : 10, "strength" : 25, "sector" : "Thriller" }
{ "_id" : ObjectId("65c3152d31edff1358b73497"), "crew_no" : 10, "strength" : 24, "sector" : "action" }
```

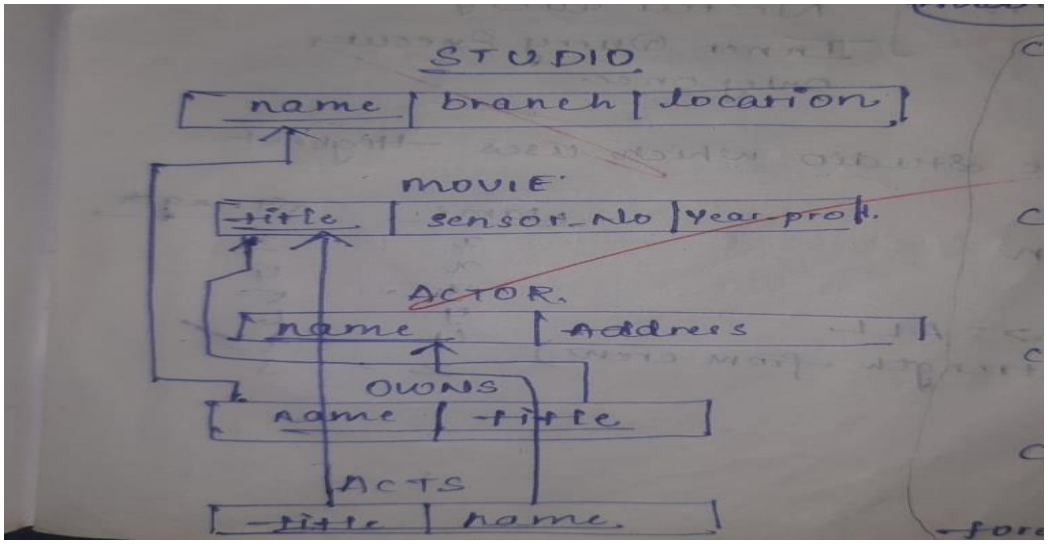
Problem Title:

2).The production company is organized 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, sensor number and year of production. Star may act in any number of movies and we store each actors name and address.

ER Model



Relational Mapping



Creating tables:

studio:

create table studio(

name varchar(25),

branch varchar(25),

location varchar(25),

primary key (name));

```
mysql> desc studio;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| name     | varchar(25) | NO   | PRI | NULL    |       |
| branch   | varchar(25) | YES  |     | NULL    |       |
| location | varchar(25) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
3 rows in set (0.03 sec)
```

movie:

create table movie(

title varchar(25),

sensor_no int,

yop int,

primary key(sensor_no));

```
mysql> desc movie;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| title     | varchar(25) | YES  |     | NULL    |       |
| sensor_no | int         | NO   | PRI | NULL    |       |
| yop       | int         | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
3 rows in set (0.01 sec)
```

actor:

create table actor(

name varchar(25),

address varchar(25),

primary key(name));

```
mysql> desc actor;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| name    | varchar(25) | NO   | PRI | NULL    |       |
| address | varchar(25) | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

owns:

create table owns(

name varchar(25),

sensor_no int,

primary key(name,sensor_no),

foreign key(name) references studio(name),

foreign key(sensor_no) references movie(sensor_no));

```
mysql> desc owns;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| name      | varchar(25) | NO   | PRI | NULL    |       |
| sensor_no | int         | NO   | PRI | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

acts:

create table acts(

name varchar(25),

sensor_no int,

primary key(name,sensor_no),

foreign key(name) references actor(name),

foreign key(sensor_no) references movie(sensor_no));

```
mysql> desc acts;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| name      | varchar(25) | NO   | PRI | NULL    |       |
| sensor_no | int         | NO   | PRI | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

Inserting into tables:

studio:

insert into studio values("Mythri","1B","Hyderabad"); insert

into studio values("Annapurna","3B","Vijaywada");insert

into studio values("Geetha","2B","Vizag");

insert into studio values("Venkateshwara","4B","Banglore");

insert into studio values("Arka Media Works","5B","Tumkur");

```
mysql> select * from studio;
+------------------+--------+-----------+
| name             | branch | location  |
+------------------+--------+-----------+
| Annapurna        | 3B     | Vijaywada |
| Arka Media Works | 5B     | Tumkur    |
| Geetha           | 2B     | Vizag     |
| Mythri           | 1B     | Hyderabad |
| Venkateshwara    | 4B     | Banglore  |
+------------------+--------+-----------+
```

movie:

insert into movie values("Kushi",1,2023);

insert into movie values("Barath Ane Nenu",4,2018);

insert into movie values("RRR",3,2020);

insert into movie values("Druva",5,2021);

insert into movie values("Manam",2,2015);

```
mysql> select * from movie;
+-----------------+-----------+------+
| title           | sensor_no | yop  |
+-----------------+-----------+------+
| Kushi           |         1 | 2023 |
| Manam           |         2 | 2015 |
| RRR             |         3 | 2020 |
| Barath Ane Nenu |         4 | 2018 |
| Druva           |         5 | 2021 |
+-----------------+-----------+------+
5 rows in set (0.00 sec)
```

actor:

insert into actor values("Vijay","Hyderabad");

insert into actor values("Mahesh","Vijaywada");

insert into actor values("NTR","Secunderabad");

insert into actor values("RamCharan","Hyderabad");

insert into actor values("NagaChaitanya","Vizag");

```
mysql> select * from actor;
+---------------+-------------+
| name          | address     |
+---------------+-------------+
| Mahesh        | Vijaywada   |
| NagaChaitanya | Vizag       |
| NTR           | Secunderabad |
| RamCharan     | Hyderabad   |
| Vijay         | Hyderabad   |
+---------------+-------------+
5 rows in set (0.00 sec)
```

owns:

insert into owns values("Annapurna",3);

insert into owns values("Annapurna",2);

insert into owns values("Mythri",1);

insert into owns values("Venkateshwara",4);

insert into owns values("Arka Media Works",4);

insert into owns values("Geetha",5);

```
mysql> select * from owns;
+-----------------+-----------+
| name            | sensor_no |
+-----------------+-----------+
| Mythri          |         1 |
| Annapurna       |         2 |
| Annapurna       |         3 |
| Arka Media Works |         4 |
| Venkateshwara   |         4 |
| Geetha          |         5 |
+-----------------+-----------+
6 rows in set (0.00 sec)
```

acts:

insert into acts values("NTR",3);

insert into acts values("RamCharan",3);

insert into acts values("NagaChaitanya",2);

insert into acts values("Vijay",1);

insert into acts values("Mahesh",4); insert into acts values("RamCharan",5);

```
mysql> select * from acts;
+---------------+-----------+
| name          | sensor_no |
+---------------+-----------+
| Vijay         |         1 |
| NagaChaitanya |         2 |
| NTR           |         3 |
| RamCharan     |         3 |
| Mahesh        |         4 |
| RamCharan     |         5 |
+---------------+-----------+
6 rows in set (0.00 sec)
```

Queries:

1. **List all the studios of movie “Bharth Ane Nenu”**

   select s.name,branch,location
   from movie m,owns o,studio s

   where title="Barath Ane Nenu" and
   m.sensor_no=o.sensor_no and
   o.name=s.name

```
+------------------+--------+----------+
| name             | branch | location |
+------------------+--------+----------+
| Arka Media Works | 5B     | Tumkur   |
| Venkateshwara    | 4B     | Banglore |
+------------------+--------+----------+
2 rows in set (0.00 sec)
```

2.List all the actors acted in the movie “RRR”

```
select a.name
from acts a,movie m
where title="RRR" and
a.sensor_no=m.sensor_no;
```

```
+-----------+
| name      |
+-----------+
| NTR       |
| RamCharan |
+-----------+
2 rows in set (0.00 sec)
```

Procedure:

Write a procedure to list all movies produced during a specificyear(2023)

```
DELIMITER //

CREATE PROCEDURE pr2(IN s INT)

BEGIN

DECLARE v_title VARCHAR(25);

DECLARE cur_done INT DEFAULT FALSE;

DECLARE cur_title CURSOR FOR

SELECT title

FROM movie WHERE

yop = s;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET cur_done = TRUE;
```

OPEN cur_title;

read_loop: LOOP

FETCH cur_title INTO v_title;

IF cur_done THEN

LEAVE read_loop;

END IF;

SELECT v_title;

END LOOP;

CLOSE cur_title;

END //

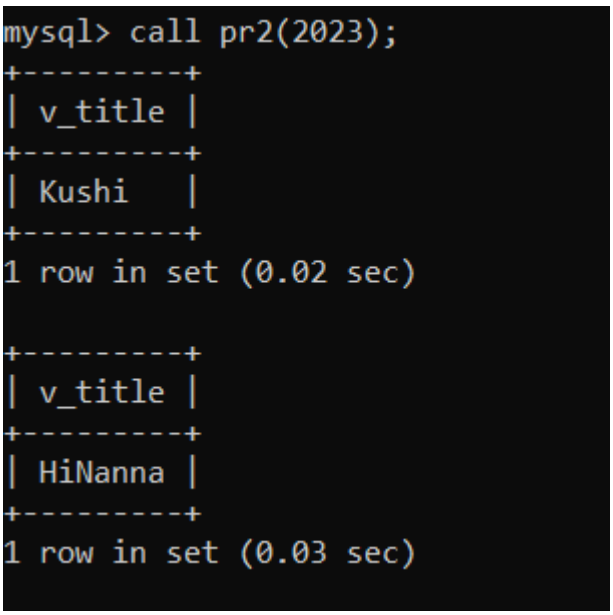
DELIMITER ;

Inserting other movie:

insert into movie values("HiNanna",6,2023);

```
mysql> select * from movie;
+-----------------+-----------+------+
| title           | sensor_no | yop  |
+-----------------+-----------+------+
| Kushi           |         1 | 2023 |
| Manam           |         2 | 2015 |
| RRR             |         3 | 2020 |
| Barath Ane Nenu |         4 | 2018 |
| Druva           |         5 | 2021 |
| HiNanna         |         6 | 2023 |
+-----------------+-----------+------+
6 rows in set (0.00 sec)
```

Procedure Call:



Trigger:

Write a deletion trigger,does not allow to deleting current year movies(2024)

```
DELIMITER //

CREATE TRIGGER tr2

BEFORE DELETE ON movie

FOR EACH ROW

BEGIN

DECLARE cur_year INT;

SET cur_year = YEAR(NOW());

IF OLD.yop = cur_year THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE_TEXT = 'Cannot delete';

END IF;

END //

DELIMITER ;
```

Inserting 2024 movies:

```
insert into movie values("GunturKaram",7,2024);

insert into movie values("Hanuman",8,2024);
```

```
mysql> select * from movie;
+-----------------+-----------+------+
| title           | sensor_no | yop  |
+-----------------+-----------+------+
| Kushi           |         1 | 2023 |
| Manam           |         2 | 2015 |
| RRR             |         3 | 2020 |
| Barath Ane Nenu |         4 | 2018 |
| Druva           |         5 | 2021 |
| HiNanna         |         6 | 2023 |
| GunturKaram     |         7 | 2024 |
| Hanuman         |         8 | 2024 |
+-----------------+-----------+------+
8 rows in set (0.00 sec)
```

Execution of Trigger:

```
mysql> delete from movie where yop=2024;
ERROR 1644 (45000): Cannot delete
```

Create studio collection and perform the following CRUD operations:

a) Create a document.

b) Create two or more documents at the same time.

c) Update a document with studio name ‘std1’ to ‘std2’.

d) Delete all the studios with location ‘xyz’.

e) Retrieve the studio with location equal to ‘xyz’.

```
> db.createCollection("studios")
{ "ok" : 1 }
```

db.studios.insertOne({
name:"std1",
branch:"B1",
location:"Mandya"});

```
> db.studios.find()
{ "_id" : ObjectId("65c31c0b31edff1358b7349b"), "name" : "std1", "branch" : "B1", "location" : "Mandya" }
```

db.studios.insertMany([
{name:"std1",branch:"B1",location:"Hindupur"},
{name:"std2",branch:"B2",location:"Hindupur"},
{name:"std3",branch:"B3",location:"Mandya"},
{name:"std4",branch:"B4",location:"Hyderabad"}]);

```
> db.studios.insertMany([
... {name:"std1",branch:"B1",location:"Hindupur"},
... {name:"std2",branch:"B2",location:"Hindupur"},
... {name:"std3",branch:"B3",location:"Mandya"},
... {name:"std4",branch:"B4",location:"Hyderabad"}]);
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("65c31d9231edff1358b7349c"),
                ObjectId("65c31d9231edff1358b7349d"),
                ObjectId("65c31d9231edff1358b7349e"),
                ObjectId("65c31d9231edff1358b7349f")
        ]
}
> db.studios.find()
{ "_id" : ObjectId("65c31c0b31edff1358b7349b"), "name" : "std1", "branch" : "B1", "location" : "Mandya" }
{ "_id" : ObjectId("65c31d9231edff1358b7349c"), "name" : "std1", "branch" : "B1", "location" : "Hindupur" }
{ "_id" : ObjectId("65c31d9231edff1358b7349d"), "name" : "std2", "branch" : "B2", "location" : "Hindupur" }
{ "_id" : ObjectId("65c31d9231edff1358b7349e"), "name" : "std3", "branch" : "B3", "location" : "Mandya" }
{ "_id" : ObjectId("65c31d9231edff1358b7349f"), "name" : "std4", "branch" : "B4", "location" : "Hyderabad" }
```

db.studios.deleteMany({location:"Mandya"})

```
> db.studios.deleteMany({location:"Mandya"})
{ "acknowledged" : true, "deletedCount" : 2 }
> db.studios.find()
{ "_id" : ObjectId("65c31d9231edff1358b7349c"), "name" : "std1", "branch" : "B1", "location" : "Hindupur" }
{ "_id" : ObjectId("65c31d9231edff1358b7349d"), "name" : "std2", "branch" : "B2", "location" : "Hindupur" }
{ "_id" : ObjectId("65c31d9231edff1358b7349f"), "name" : "std4", "branch" : "B4", "location" : "Hyderabad" }
```

db.studios.find({location:"Hindupur"})

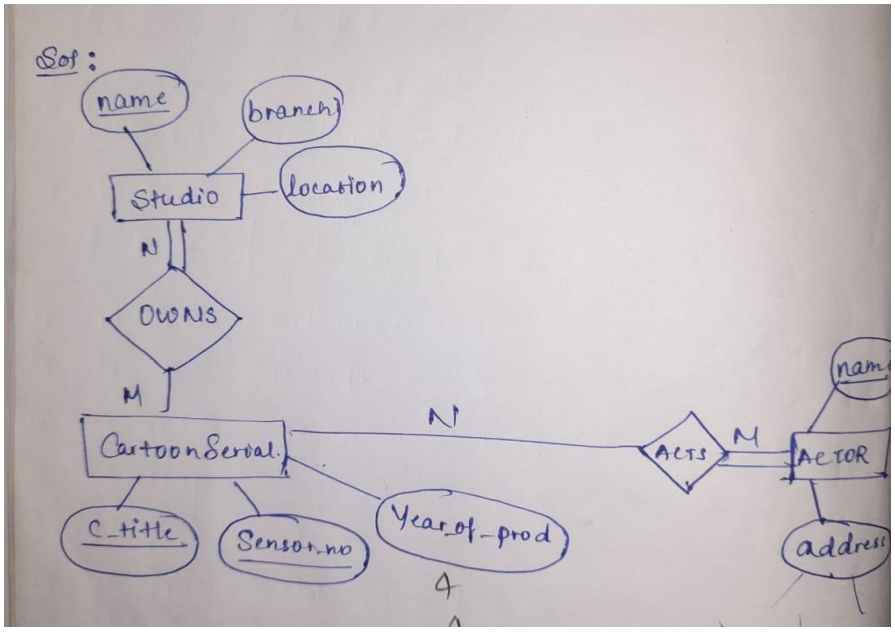
```
> db.studios.find({location:"Hindupur"})
{ "_id" : ObjectId("65c31d9231edff1358b7349c"), "name" : "std1", "branch" : "B1", "location" : "Hindupur" }
{ "_id" : ObjectId("65c31d9231edff1358b7349d"), "name" : "std2", "branch" : "B2", "location" : "Hindupur" }
```

db.studios.updateOne({name:"std1"},{$set:{name:"std2"}})

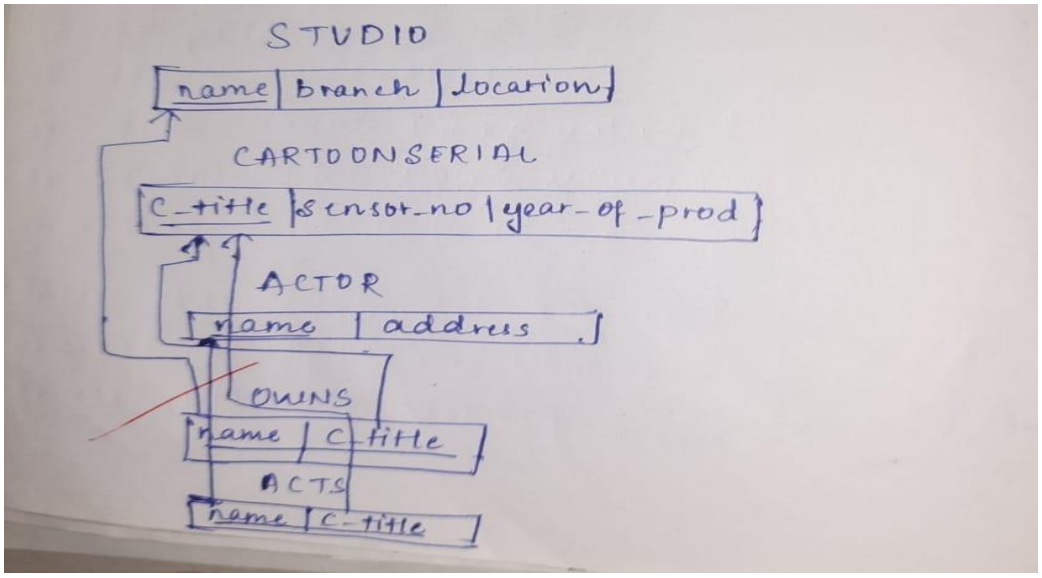
```
> db.studios.updateOne({name:"std1"},{$set:{name:"std2"}});
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.studios.find()
{ "_id" : ObjectId("65c31d9231edff1358b7349c"), "name" : "std2", "branch" : "B1", "location" : "Hindupur" }
{ "_id" : ObjectId("65c31d9231edff1358b7349d"), "name" : "std2", "branch" : "B2", "location" : "Hindupur" }
{ "_id" : ObjectId("65c31d9231edff1358b7349f"), "name" : "std4", "branch" : "B4", "location" : "Hyderabad" }
>
```

3)A production company is organized into different studios.We store each studios name, branch and location. A studio can own any number of cartoon serials . We store cartoon serial title,sensor no,yearofproduction.Star may voice in any number of cartoon serials and we store each actors name and address.

ERMODEL



RELATIONAL MAPPING



A production company is organized into different studios.We store each studios name, branch and location. A studio can own any number of cartoon serials . We store cartoon serial title,sensor no,yearofproduction.Star may voice in any number of cartoon serials and we store each actors name and address.

Creating Tables:

cartoonstudio:

create table cartoonstudio (name varchar(25),branch varchar(25),location varchar(25),primary key(name));

```
mysql> desc cartoonstudio
    -> ;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| name     | varchar(25) | NO   | PRI | NULL    |       |
| branch   | varchar(25) | YES  |     | NULL    |       |
| location | varchar(25) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
3 rows in set (0.22 sec)
```

cartoonserial:

create table cartoonserial(c_title varchar(25),sensor_no int,year_of_production int,primary key(c_title));

```
mysql> desc cartoonserial
    -> ;
+--------------------+-------------+------+-----+---------+-------+
| Field              | Type        | Null | Key | Default | Extra |
+--------------------+-------------+------+-----+---------+-------+
| c_title            | varchar(25) | NO   | PRI | NULL    |       |
| sensor_no          | int         | YES  |     | NULL    |       |
| year_of_production | int         | YES  |     | NULL    |       |
+--------------------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

cartoonactor:

create table cartoonactor(name varchar(25),address varchar(25),primary key(name));

```
mysql> desc cartoonactor;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| name    | varchar(25) | NO   | PRI | NULL    |       |
| address | varchar(25) | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

cartoonowns:
create table cartoonowns(name varchar(25),c_title varchar(25),foreign key(name) references cartoonstudio(name),foreign key(c_title) references cartoonserial(c_title),primary key(name,c_title));

```
mysql> desc cartoonowns
    -> ;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| name    | varchar(25) | NO   | PRI | NULL    |       |
| c_title | varchar(25) | NO   | PRI | NULL    |       |
+---------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

cartoonacts:
create table cartoonacts(c_title varchar(25),name varchar(25),foreign key(c_title) references cartoonserial(c_title),foreign key(name) references cartoonactor(name),primary key(c_title,name));

```
mysql> desc cartoonacts;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| c_title | varchar(25) | NO   | PRI | NULL    |       |
| name    | varchar(25) | NO   | PRI | NULL    |       |
+---------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

Inserting to cartoonstudio:
insert into cartoonstudio values("Disney","Disneyland","Florida");
insert into cartoonstudio values("CN","CNland","Europe");
insert into cartoonstudio values("Pogo","PogoLand","Russia");
insert into cartoonstudio values("Nick","NickLand","Australia");
insert into cartoonstudio values("kushi","kushiLand","India");

```
mysql> select * from cartoonstudio;
+--------+------------+-----------+
| name   | branch     | location  |
+--------+------------+-----------+
| CN     | CNland     | Europe    |
| Disney | Disneyland | Florida   |
| kushi  | kushiLand  | India     |
| Nick   | NickLand   | Australia |
| Pogo   | PogoLand   | Russia    |
+--------+------------+-----------+
5 rows in set (0.00 sec)
```

Inserting to cartoonserial:
insert into cartoonserial values("Doug",1,2005);
insert into cartoonserial values("Popeye The Sailor",2,2007);
insert into cartoonserial values("Pokemon",2,2007);
insert into cartoonserial values("Dexter's Laboratory ",3,2010);
insert into cartoonserial values("Alvin and the Chipmunks ",4,2003);

```
mysql> select * from cartoonserial;
+--------------------------+-----------+--------------------+
| c_title                  | sensor_no | year_of_production |
+--------------------------+-----------+--------------------+
| Alvin and the Chipmunks  |         4 |               2003 |
| Dexter's Laboratory      |         3 |               2010 |
| Doug                     |         1 |               2005 |
| Pokemon                  |         2 |               2007 |
| Popeye The Sailor        |         2 |               2007 |
+--------------------------+-----------+--------------------+
5 rows in set (0.01 sec)
```

Inserting to cartoonactor:

insert into cartoonactor values("Mickey Mouse","Hindupur");
insert into cartoonactor values("Tom And Jerry","Anantapur");
insert into cartoonactor values("Bugs Bunny","Vijaywada");
insert into cartoonactor values("SpongeBob Square Pants","Tumkur");
insert into cartoonactor values("Homer J. Simpson","Hyderabad");

```
mysql> select * from cartoonactor;
+------------------------+-----------+
| name                   | address   |
+------------------------+-----------+
| Bugs Bunny             | Vijaywada |
| Homer J. Simpson       | Hyderabad |
| Mickey Mouse           | Hindupur  |
| SpongeBob Square Pants | Tumkur    |
| Tom And Jerry          | Anantapur |
+------------------------+-----------+
```

Inserting to cartoonowns:

insert into cartoonowns values("Disney","Pokemon");
insert into cartoonowns values("Disney","Popeye The Sailor");
insert into cartoonowns values("Nick","Doug");
insert into cartoonowns values("Pogo","Popeye The Sailor");

```
mysql> select * from cartoonowns;
+--------+-------------------+
| name   | c_title           |
+--------+-------------------+
| Nick   | Doug              |
| Disney | Pokemon           |
| Disney | Popeye The Sailor |
| Pogo   | Popeye The Sailor |
+--------+-------------------+
4 rows in set (0.00 sec)
```

Inserting to cartoonacts:

```
insert into cartoonacts values("Popeye The Sailor","Tom And Jerry");
insert into cartoonacts values("Popeye The Sailor","Mickey Mouse");
insert into cartoonacts values("Pokemon","Homer J. Simpson");
insert into cartoonacts values("Doug","Homer J. Simpson");
```

```
mysql> select * from cartoonacts;
+-------------------+------------------+
| c_title           | name             |
+-------------------+------------------+
| Doug              | Homer J. Simpson |
| Pokemon           | Homer J. Simpson |
| Popeye The Sailor | Mickey Mouse     |
| Popeye The Sailor | Tom And Jerry    |
+-------------------+------------------+
```

Queries:

1.Find the total Number of actors who voied in a serial "Popeye The Sailor"

```
select count(*)
from cartoonacts
group by c_title
having c_title="Popeye The Sailor";
```

```
mysql> select count(*) from cartoonacts group by c_title having c_title="Popeye The Sailor"
+----------+
| count(*) |
+----------+
|        2 |
+----------+
```

2.Retrive name of studio and cartoon serials title in which star "Homer J. Simpson" voiced.

```
select s.name,s.location,c.c_title
from cartoonacts a,cartoonserial c,cartoonowns o,cartoonstudio s
where a.name="Homer J. Simpson" and
a.c_title=c.c_title and
c.c_title=o.c_title and
o.name=s.name;
```

```
mysql> select s.name,s.location,c.c_title from cartoonacts a,cartoonserial c,cartoonowns o,cartoonstudio s where a.name="Homer J. Simpson" and a.c_title=c.c_title an
d c.c_title=o.c_title and o.name=s.name;
```

```
+--------+-----------+---------+
| name   | location  | c_title |
+--------+-----------+---------+
| Nick   | Australia | Doug    |
| Disney | Florida   | Pokemon |
+--------+-----------+---------+
```

Procedure:
Write a procedure to list all cartoon serials produced during the specific year.

```
DELIMITER //
CREATE PROCEDURE pr003(IN s INT)
BEGIN
DECLARE v_title VARCHAR(25);
DECLARE cur_done BOOLEAN DEFAULT FALSE;
DECLARE cur_title CURSOR FOR
SELECT c_title
FROM cartoonserial
WHERE year_of_production = s;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET cur_done = TRUE;
OPEN cur_title;
read_loop: LOOP
FETCH cur_title INTO v_title;
IF cur_done THEN
LEAVE read_loop;
END IF;
SELECT v_title;
END LOOP;
CLOSE cur_title;
END//
DELIMITER ;
```

```
mysql> call pr003(2007);
+---------+
| v_title |
+---------+
| Pokemon |
+---------+
1 row in set (0.00 sec)

+-------------------+
| v_title           |
+-------------------+
| Popeye The Sailor |
+-------------------+
1 row in set (0.01 sec)

Query OK, 0 rows affected (0.01 sec)
```

Trigger:
Write a deletion trigger that doesn't allow to delete current year cartoon serial
Inserting current year cartoonserial:

```
insert into cartoonserial values('Mickey',1,2024);
```

carttonserial table:

```
mysql> select * from cartoonserial;
+--------------------------+----------+--------------------+
| c_title                  | sensor_no | year_of_production |
+--------------------------+----------+--------------------+
| Alvin and the Chipmunks  |         4 |               2003 |
| Dexter's Laboratory      |         3 |               2010 |
| Doug                     |         1 |               2005 |
| Mickey                   |         1 |               2024 |
| Pokemon                  |         2 |               2007 |
| Popeye The Sailor        |         2 |               2007 |
+--------------------------+----------+--------------------+
6 rows in set (0.00 sec)
```

Trigger:

```
DELIMITER //
CREATE TRIGGER  tr3
BEFORE DELETE ON cartoonserial
FOR EACH ROW
BEGIN
DECLARE cur_year INT;
SET cur_year=YEAR(NOW());
IF OLD.year_of_production=cur_year THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT='cannot delete';
END IF;
END//
DELIMITER ;
```

MongoDB

Create “cartoon serial” collection and perform the following CRUD operations:

- Create a document.
- Create two or more documents at the same time.
- Update a document with cartoon serial title ‘std1’ to ‘std2’.
- Delete all the cartoon serials with title ‘xyz’.
- Retrieve the cartoon serials with sensor numbers lesser than 10

db.createCollection("cartoonserial")

```
> db.createCollection("cartoonserial")
{ "ok" : 1 }
```

db.cartoonserial.insertOne({title:"std1",s_no:1,yop:"2003"});

```
> db.cartoonserial.insertOne({title:"std1",s_no:1,yop:"2003"});
{
        "acknowledged" : true,
        "insertedId" : ObjectId("65c3219431edff1358b734a0")
}
> db.cartoonserial.find()
{ "_id" : ObjectId("65c3219431edff1358b734a0"), "title" : "std1", "s_no" : 1, "yop" : "2003" }
```

db.cartoonserial.insertMany([
{title:"std1",s_no:5,yop:"2003"},
{title:"std2",s_no:2,yop:"2001"},
{title:"std3",s_no:3,yop:"2002"},
{title:"std3",s_no:4,yop:"2008"}]);

```
> db.cartoonserial.find()
{ "_id" : ObjectId("65c3219431edff1358b734a0"), "title" : "std1", "s_no" : 1, "yop" : "2003" }
> db.cartoonserial.insertMany([
... {title:"std1",s_no:5,yop:"2003"},
... {title:"std2",s_no:2,yop:"2001"},
... {title:"std3",s_no:3,yop:"2002"},
... {title:"std3",s_no:4,yop:"2008"}]);
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("65c322f931edff1358b734a1"),
                ObjectId("65c322f931edff1358b734a2"),
                ObjectId("65c322f931edff1358b734a3"),
                ObjectId("65c322f931edff1358b734a4")
        ]
}
> db.cartoonserial.find()
{ "_id" : ObjectId("65c3219431edff1358b734a0"), "title" : "std1", "s_no" : 1, "yop" : "2003" }
{ "_id" : ObjectId("65c322f931edff1358b734a1"), "title" : "std1", "s_no" : 5, "yop" : "2003" }
{ "_id" : ObjectId("65c322f931edff1358b734a2"), "title" : "std2", "s_no" : 2, "yop" : "2001" }
{ "_id" : ObjectId("65c322f931edff1358b734a3"), "title" : "std3", "s_no" : 3, "yop" : "2002" }
{ "_id" : ObjectId("65c322f931edff1358b734a4"), "title" : "std3", "s_no" : 4, "yop" : "2008" }
```

db.cartoonserial.updateOne({title:"std2"},{$set:{title:"std5"}})

```
> db.cartoonserial.updateOne({title:"std2"},{$set:{title:"std5"}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.cartoonserial.find()
{ "_id" : ObjectId("65c3219431edff1358b734a0"), "title" : "std1", "s_no" : 1, "yop" : "2003" }
{ "_id" : ObjectId("65c322f931edff1358b734a1"), "title" : "std1", "s_no" : 5, "yop" : "2003" }
{ "_id" : ObjectId("65c322f931edff1358b734a2"), "title" : "std5", "s_no" : 2, "yop" : "2001" }
{ "_id" : ObjectId("65c322f931edff1358b734a3"), "title" : "std3", "s_no" : 3, "yop" : "2002" }
{ "_id" : ObjectId("65c322f931edff1358b734a4"), "title" : "std3", "s_no" : 4, "yop" : "2008" }
```

db.cartoonserial.deleteMany({title:"std1"})

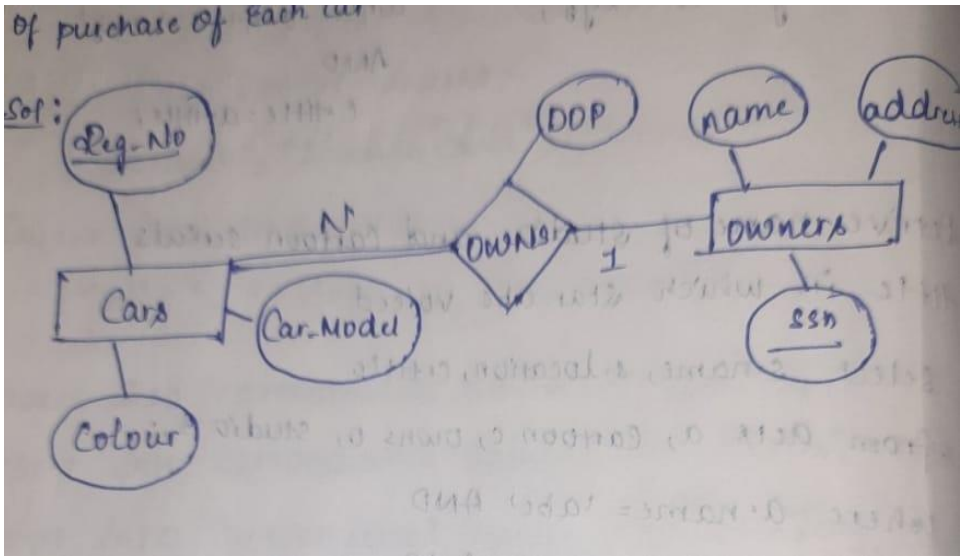
```
> db.cartoonserial.deleteMany({title:"std1"})
{ "acknowledged" : true, "deletedCount" : 2 }
> db.cartoonserial.find()
{ "_id" : ObjectId("65c322f931edff1358b734a2"), "title" : "std5", "s_no" : 2, "yop" : "2001" }
{ "_id" : ObjectId("65c322f931edff1358b734a3"), "title" : "std3", "s_no" : 3, "yop" : "2002" }
{ "_id" : ObjectId("65c322f931edff1358b734a4"), "title" : "std3", "s_no" : 4, "yop" : "2008" }
```

db.cartoonserial.find({s_no:{$lt:4}})

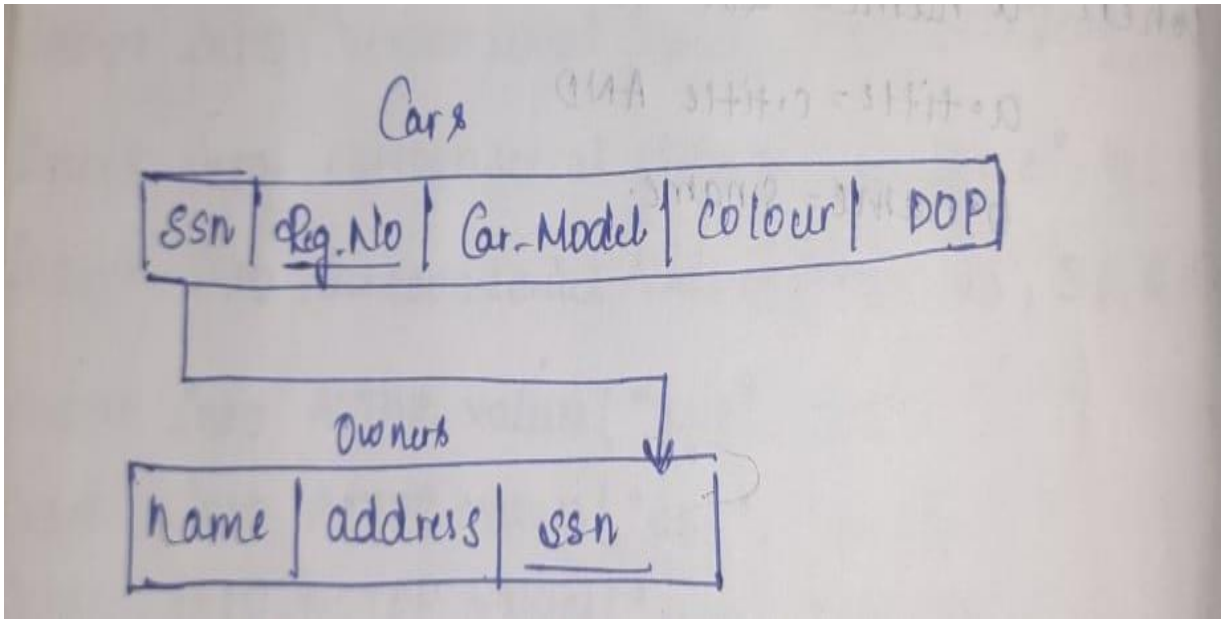
```
> db.cartoonserial.find({s_no:{$lt:4}})
{ "_id" : ObjectId("65c322f931edff1358b734a2"), "title" : "std5", "s_no" : 2, "yop" : "2001" }
{ "_id" : ObjectId("65c322f931edff1358b734a3"), "title" : "std3", "s_no" : 3, "yop" : "2002" }
```

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

ERMODEL:



RELATIONAL MAPPING:



CREATE TABLE STATEMENTS:

```
create table owners(
name varchar(25),
address varchar(25),
ssn int,
primary key(ssn));
```

```
mysql> select * from owners;
+-----------+-----------+-----+
| name      | address   | ssn |
+-----------+-----------+-----+
| Niveditha | Hindupur  |   1 |
| Pratiksha | Shivmogga |   2 |
| Nanditha  | Tumkur    |   3 |
| Sinchana  | chennai   |   4 |
| Unnati    | sira      |   5 |
| Sowmya    | bijapur   |   6 |
+-----------+-----------+-----+
6 rows in set (0.00 sec)
```

```
create table cars(
ssn int,
rg_no int,
car_model varchar(25),
color varchar(25),
dop date,
primary key(ssn,rg_no),
foreign key(ssn) references owners(ssn));
```

```
mysql> desc cars
    -> ;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| ssn       | int         | NO   | PRI | NULL    |       |
| rg_no     | int         | NO   | PRI | NULL    |       |
| car_model | varchar(25) | YES  |     | NULL    |       |
| color     | varchar(25) | YES  |     | NULL    |       |
| dop       | date        | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
```

Insert Statements:

```
insert into owners values("Niveditha","Hindupur",1);
insert into owners values("Pratiksha","Shivmogga",2);
insert into owners values("Nanditha","Tumkur",3);
insert into owners values("Sinchana","chennai",4);
insert into owners values("Unnati","sira",5);
insert into owners values("Sowmya","bijapur",6);
```

```
mysql> select * from owners;
+-----------+-----------+-----+
| name      | address   | ssn |
+-----------+-----------+-----+
| Niveditha | Hindupur  |   1 |
| Pratiksha | Shivmogga |   2 |
| Nanditha  | Tumkur    |   3 |
| Sinchana  | chennai   |   4 |
| Unnati    | sira      |   5 |
| Sowmya    | bijapur   |   6 |
+-----------+-----------+-----+
6 rows in set (0.00 sec)
```

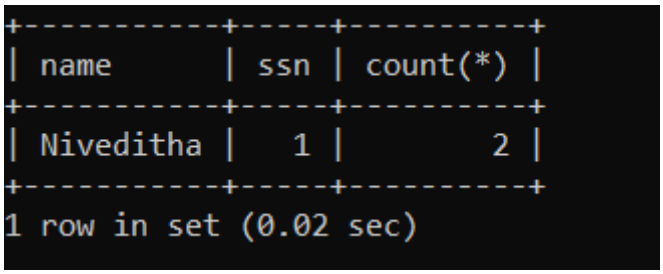
```
insert into cars values(1,88,"kia","blue","2011-11-11");
insert into cars values(1,98,"alto","blue","2011-12-11");
insert into cars values(2,56,"i3","blue","2011-11-11");
insert into cars values(3,45,"nia","blue","2011-11-30");
insert into cars values(4,90,"bmw","blue","2011-07-04");
insert into cars values(6,67,"swift","blue","2011-06-05");
```

```
mysql> select * from cars;
+-----+-------+-----------+-------+------------+
| ssn | rg_no | car_model | color | dop        |
+-----+-------+-----------+-------+------------+
|   1 |    88 | kia       | blue  | 2011-11-11 |
|   1 |    98 | alto      | blue  | 2011-12-11 |
|   2 |    56 | i3        | blue  | 2011-11-11 |
|   3 |    45 | nia       | blue  | 2011-11-30 |
|   4 |    90 | bmw       | blue  | 2011-07-04 |
|   6 |    67 | swift     | blue  | 2011-06-05 |
+-----+-------+-----------+-------+------------+
6 rows in set (0.01 sec)
```

Queries:

Find the person who owns highest number of cars

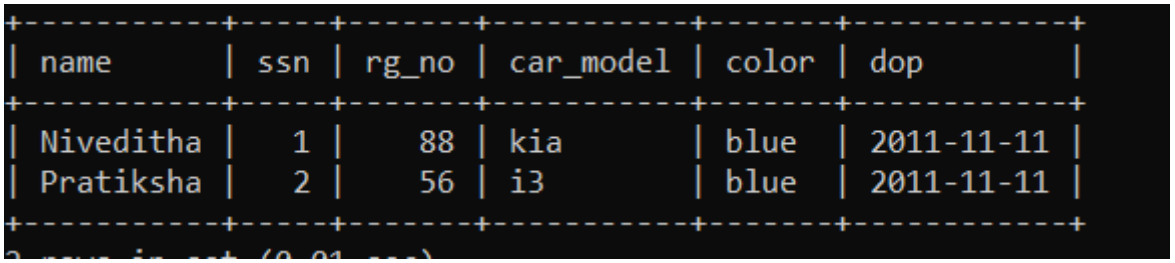
```
select o.name,c.ssn,count(*)
from cars c,owners o
where c.ssn=o.ssn
group by o.name,c.ssn
having count(c.ssn)>=ALL
(select count(c1.ssn)
from cars c1
group by (c1.ssn));
```



```
+-----------+-----+----------+
| name      | ssn | count(*) |
+-----------+-----+----------+
| Niveditha |   1 |        2 |
+-----------+-----+----------+
1 row in set (0.02 sec)
```

Retrive persons and cars information purchased on the day 11-11-11

```
select o.name,o.ssn,c.rg_no,c.car_model,c.color,c.dop
from cars c,owners o
where c.ssn=o.ssn and c.dop="2011-11-11";
```



```
+-----------+-----+-------+-----------+-------+------------+
| name      | ssn | rg_no | car_model | color | dop        |
+-----------+-----+-------+-----------+-------+------------+
| Niveditha |   1 |    88 | kia       | blue  | 2011-11-11 |
| Pratiksha |   2 |    56 | i3        | blue  | 2011-11-11 |
+-----------+-----+-------+-----------+-------+------------+
```

Write a procedure to list all cars and owner information purchased during a specific year

```
DELIMITER //
CREATE PROCEDURE sy1(IN sdate date)
BEGIN
DECLARE o_name varchar(25);
DECLARE o_address varchar(25);
DECLARE o_ssn int;
DECLARE crg_no int;
DECLARE c_colour varchar(25);
DECLARE c_model varchar(25);
DECLARE done INT DEFAULT FALSE;

DECLARE c1 CURSOR FOR
select o.name,o.address,o.ssn
from cars c,owners o
where c.ssn=o.ssn and c.dop=sdate;

DECLARE c2 CURSOR FOR
select c.rg_no,c.color,c.car_model
from cars c,owners o
where c.ssn=o.ssn and c.dop=sdate;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done=TRUE;

OPEN c1;
SELECT 'OWNER DETAILS';
read_loop1:LOOP
FETCH c1 INTO o_name,o_address,o_ssn;
IF done THEN
LEAVE read_loop1;
END IF;
SELECT CONCAT(o_name,' ',o_address,' ',o_ssn);
END LOOP;

CLOSE c1;
SET done=FALSE;
OPEN c2;
SELECT 'CAR DETAILS';
read_loop2:LOOP
FETCH c2 INTO crg_no,c_colour,c_model;
IF done THEN
LEAVE read_loop2;
END IF;
SELECT CONCAT(crg_no,' ',c_colour,' ',c_model);
END LOOP;
CLOSE c2;
END //
DELIMETER ;
```

```
mysql> call sy1("2011-11-11");
+---------------+
| OWNER DETAILS |
+---------------+
| OWNER DETAILS |
+---------------+
1 row in set (0.01 sec)

+----------------------------------------+
| CONCAT(o_name,' ',o_address,' ',o_ssn) |
+----------------------------------------+
| Niveditha Hindupur 1                   |
+----------------------------------------+
1 row in set (0.02 sec)

+----------------------------------------+
| CONCAT(o_name,' ',o_address,' ',o_ssn) |
+----------------------------------------+
| Pratiksha Shivmogga 2                  |
+----------------------------------------+
1 row in set (0.03 sec)

+-------------+
| CAR DETAILS |
+-------------+
| CAR DETAILS |
+-------------+
1 row in set (0.05 sec)

+---------------------------------------+
| CONCAT(crg_no,' ',c_colour,' ',c_model) |
+---------------------------------------+
| 88 blue kia                           |
+---------------------------------------+
1 row in set (0.06 sec)

+---------------------------------------+
| CONCAT(crg_no,' ',c_colour,' ',c_model) |
+---------------------------------------+
| 56 blue i3                            |
+---------------------------------------+
1 row in set (0.09 sec)

Query OK, 0 rows affected (0.11 sec)
```

Write a insertion trigger to check date of purchase must be less than current date

```
DELIMITER //
CREATE TRIGGER t4
BEFORE INSERT ON cars
FOR EACH ROW
BEGIN
DECLARE cur_date DATE;
SET cur_date=NOW();
IF NEW.dop>cur_date THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT='Incorrect data';
END IF;
END //
DELIMITER ;
```

insert into cars values(6,67,"swift","blue","2040-06-05");

```
mysql> insert into cars values(6,67,"swift","blue","2040-06-05");
ERROR 1644 (45000): Incorrect data
```

MONGODB:

Create “car” collection and perform the following CRUD operations:

- **Create a document.**
- **Create two or more documents at the same time.**
- **Update a document with car reg.no 10 to 20.**
- **Delete all the cars with model ‘xyz’.**
- **Retrieve the cars with colour green**

db.createCollection('cars')

db.cars.insertOne({r_gno:10,model:'kia',colour:'green'})

```
> db.cars.insertOne({r_gno:10,model:'kia',colour:'green'})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("65de3404d9d45f13bc09f36c")
}
```

db.cars.insertMany([{r_gno:24,model:'swift',colour:'yellow'},
{r_gno:21,model:'kia',colour:'black'},
{r_gno:22,model:'i2',colour:'white'}])

```
> db.cars.insertMany([{r_gno:24,model:'swift',colour:'yellow'},
... {r_gno:21,model:'kia',colour:'black'},
... {r_gno:22,model:'i2',colour:'white'}])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("65de340fd9d45f13bc09f36d"),
                ObjectId("65de340fd9d45f13bc09f36e"),
                ObjectId("65de340fd9d45f13bc09f36f")
        ]
}
```

db.cars.updateOne({r_gno:10},{$set:{r_gno:20}})

```
> db.cars.updateOne({r_gno:10},{$set:{r_gno:20}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.cars.delteMany({model:'kia'})
```

db.cars.deleteMany({model:'kia'})

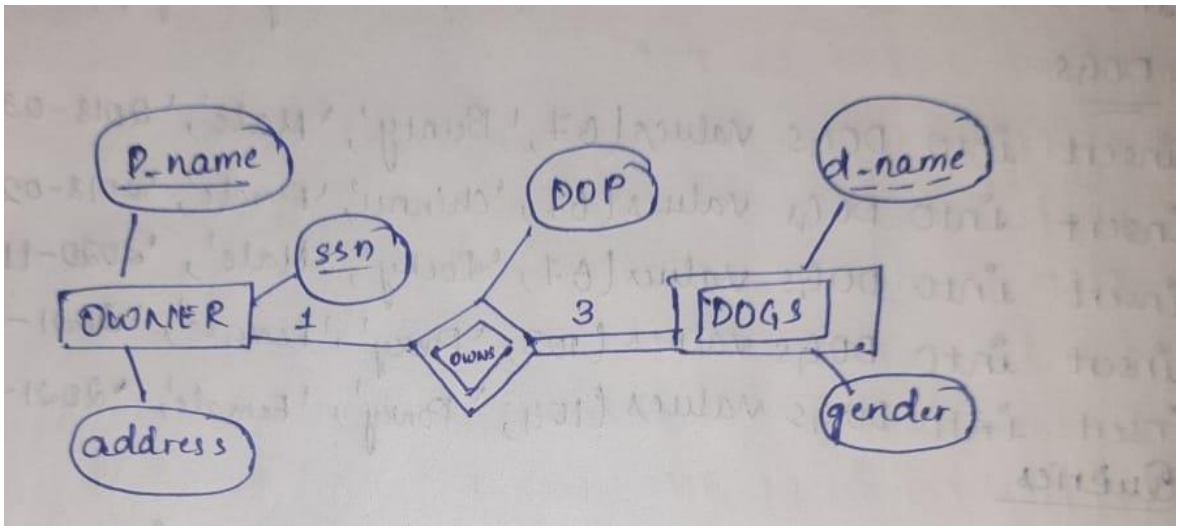
```
> db.cars.deleteMany({model:'kia'})
{ "acknowledged" : true, "deletedCount" : 2 }
> db.cars.find({colour:'green'})
```

db.cars.find({colour:'green'})

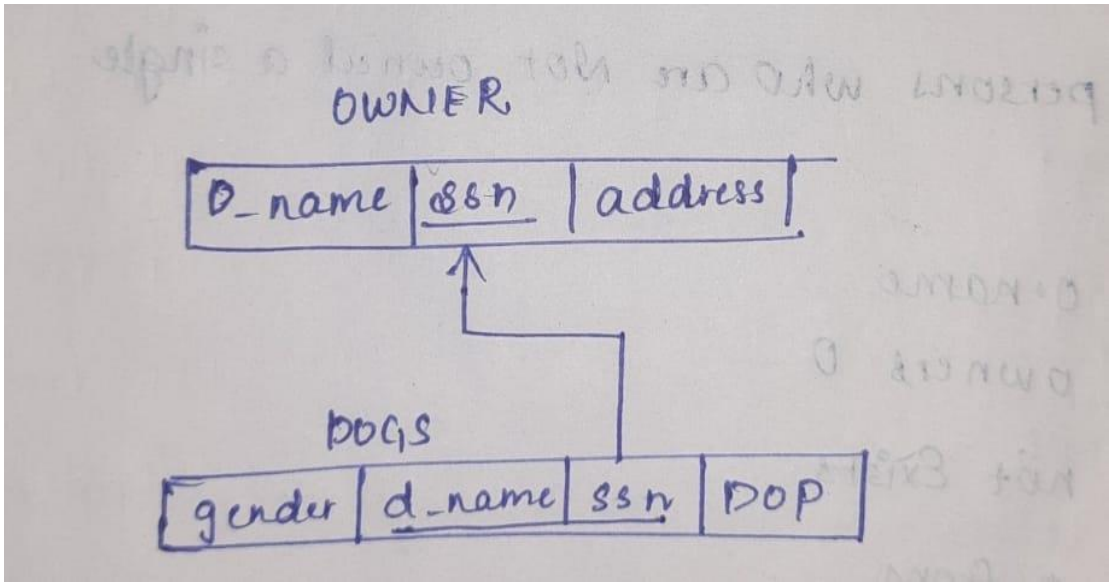
```
> db.cars.find({colour:'green'})
{ "_id" : ObjectId("65de346ad9d45f13bc09f370"), "r_gno" : 10, "model" : "kia", "colour" : "green" }
>
```

5)The puppy pet Shop wants to keep track of dogs and their owners. The person can buy maximum 3 pet dogs. We store person's name, SSN and address and dog's name, date of purchase and gender. The owner of the pets dogs will be identified by SSN, since the dog's names are not distinct.

ER DIAGRAM



RELATIONAL MAPPING:



Creating tables:
owner:
create table owner(
ssn int,
name varchar(25),
address varchar(25),
primary key(ssn));

```
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| ssn     | int         | NO   | PRI | NULL    |       |
| name    | varchar(25) | YES  |     | NULL    |       |
| address | varchar(25) | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
```

dog:

create table dog(
ssn int,
name varchar(25),
gender varchar(25),
dop date,
primary key(ssn,name),
foreign key(ssn) references owner(ssn));

```
ysql> desc dog;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| ssn    | int         | NO   | PRI | NULL    |       |
| name   | varchar(25) | NO   | PRI | NULL    |       |
| gender | varchar(25) | YES  |     | NULL    |       |
| dop    | date        | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
```

Inserting:

owner:

insert into owner values(88,"Niveditha","Hindupur");
insert into owner values(67,"Nanditha","Tumkur");
insert into owner values(75,"Pratiksha","Shivmogga");
insert into owner values(103,"Soumya","Bijapur");
insert into owner values(104,"SoumyaShetty","Shivmogga");

```
+-----+-------------+-----------+
| ssn | name        | address   |
+-----+-------------+-----------+
|  67 | Nanditha    | Tumkur    |
|  75 | Pratiksha   | Shivmogga |
|  88 | Niveditha   | Hindupur  |
| 103 | Soumya      | Bijapur   |
| 104 | SoumyaShetty| Shivmogga |
+-----+-------------+-----------+
```

dog:

insert into dog values(67,"Bunty","male",'2018-03-27');
insert into dog values(67,"Chinnu","female",'2018-02-20');
insert into dog values(67,"Rocky","male",'2020-11-03');

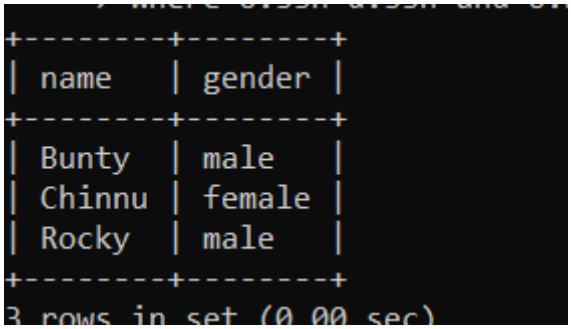
insert into dog values(103,"Pinky","female",'2021-10-07');
insert into dog values(104,"Ponky","female",'2021-10-08');

```
mysql> select * from dog;
+-----+--------+--------+------------+
| ssn | name   | gender | dop        |
+-----+--------+--------+------------+
|  67 | Bunty  | male   | 2018-03-27 |
|  67 | Chinnu | female | 2018-02-20 |
|  67 | Rocky  | male   | 2020-11-03 |
| 103 | Pinky  | female | 2021-10-07 |
| 103 | Sweety | female | 2018-03-27 |
| 104 | Juli   | male   | 2018-03-27 |
| 104 | Ponky  | female | 2021-10-08 |
+-----+--------+--------+------------+
```

Queries:
1.List all persons who are not owned a single pet
select o.name
from owner o
where not exists
(select * from dog d
where o.ssn=d.ssn);

```
    -> where d.ssn=o.ssn);
+-----------+
| name      |
+-----------+
| Pratiksha |
| Niveditha |
+-----------+
2 rows in set (0.01 sec)
```

2.List all pets owned by “Nanditha”
select d.name,d.gender,d.dop
from dog d,owner o
where o.name="Nanditha" and
d.ssn=o.ssn;

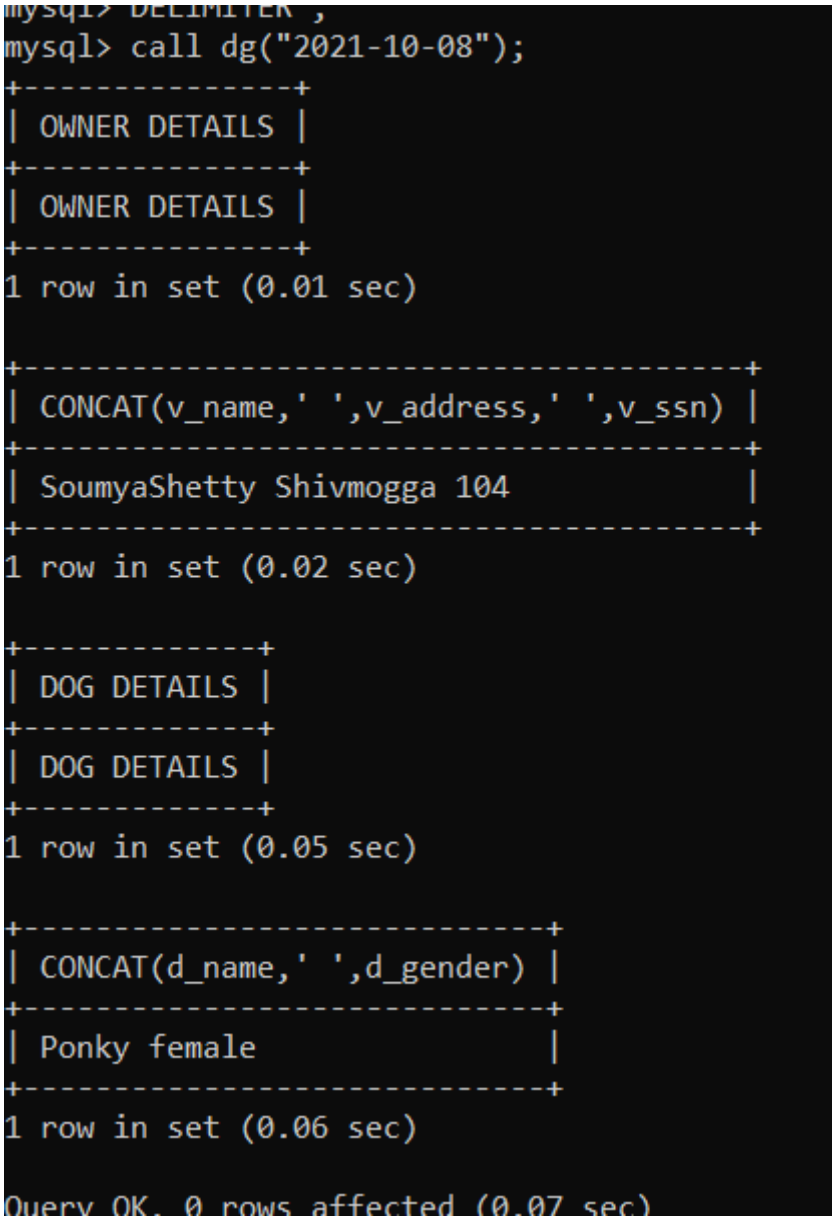


```
+--------+--------+
| name   | gender |
+--------+--------+
| Bunty  | male   |
| Chinnu | female |
| Rocky  | male   |
+--------+--------+
3 rows in set (0.00 sec)
```

Procedure:
Write a procedure to list all dogs and owner details purchased on a specific date

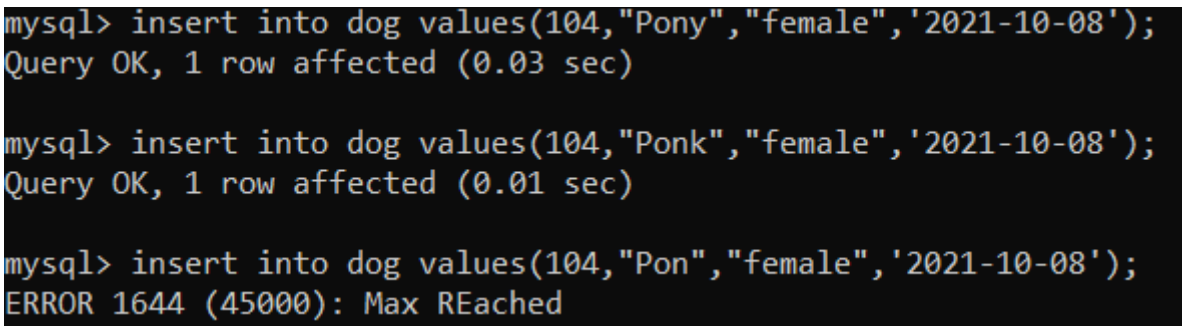
```
DELIMITER //
CREATE PROCEDURE dg(IN sdate DATE)
BEGIN
DECLARE v_name VARCHAR(25);
DECLARE v_address VARCHAR(25);
DECLARE v_ssn INT;
DECLARE d_name VARCHAR(25);
DECLARE d_gender VARCHAR(25);
DECLARE done INT DEFAULT FALSE;

DECLARE c1 CURSOR FOR
select o.name,o.address,o.ssn
from owner o,dog d
WHERE o.ssn=d.ssn AND d.dop=sdate;

DECLARE c2 CURSOR FOR
select d.name,d.gender
from owner o,dog d
WHERE o.ssn=d.ssn AND d.dop=sdate;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done=TRUE;

OPEN c1;
SELECT 'OWNER DETAILS';
read_loop1:LOOP
FETCH c1 INTO v_name,v_address,v_ssn;
IF done THEN
LEAVE read_loop1;
END IF;
SELECT CONCAT(v_name,' ',v_address,' ',v_ssn);
END LOOP;

CLOSE c1;
SET done=FALSE;
OPEN c2;
```

```
SELECT 'DOG DETAILS';
read_loop2:LOOP
FETCH c2 INTO d_name,d_gender;
IF DONE THEN
LEAVE read_loop2;
END IF;
SELECT CONCAT(d_name,' ',d_gender);
END LOOP;

CLOSE c2;
END //
DELIMITER ;
```



Trigger:
Write a trigger to check the constraint that aperson can buy maximum three pet dogs

```
DELIMITER //
CREATE TRIGGER dgg
BEFORE INSERT on dog
FOR EACH ROW
BEGIN
DECLARE cnt INT;
SELECT count(*) INTO cnt FROM dog
WHERE ssn=NEW.ssn;
IF cnt>3 THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT='Max REached';
END IF;
END //
DELIMITER ;
```



MONGODB:

Create “dog” collection and perform the following CRUD operations:

- **Create a document.**
- **Create two or more documents at the same time.**
- **Update a document with dog name ‘xyz’ to ‘abc’.**
- **Delete all the dogs with gender male.**
- **Retrieve the dogs with gender female**

db.createCollection('dog');

db.dog.insertOne({name:'Charlie',gender:'male'});

```
> db.dog.insertOne({name:'Charlie',gender:'male'});
{
        "acknowledged" : true,
        "insertedId" : ObjectId("65de3078d9d45f13bc09f368")
}
```

db.dog.insertMany([
{name:'Bunty',gender:'male'},
{name:'Rosy',gender:'female'},
{name:'Spicy',gender:'male'}]);

```
> db.dog.insertMany([
... {name:'Bunty',gender:'male'},
... {name:'Rosy',gender:'female'},
... {name:'Spicy',gender:'male'}]);
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("65de310cd9d45f13bc09f369"),
                ObjectId("65de310cd9d45f13bc09f36a"),
                ObjectId("65de310cd9d45f13bc09f36b")
        ]
}
```

```
> db.dog.find();
{ "_id" : ObjectId("65de3078d9d45f13bc09f368"), "name" : "Charlie", "gender" : "male" }
{ "_id" : ObjectId("65de310cd9d45f13bc09f369"), "name" : "Bunty", "gender" : "male" }
{ "_id" : ObjectId("65de310cd9d45f13bc09f36a"), "name" : "Rosy", "gender" : "female" }
{ "_id" : ObjectId("65de310cd9d45f13bc09f36b"), "name" : "Spicy", "gender" : "male" }
```

db.dog.updateOne({name:'Spicy'},{$set:{name:'Julie'}});

```
> db.dog.updateOne({name:'Spicy'},{$set:{name:'Julie'}});
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
```

db.dog.deleteMany({gender:'male'})

```
> db.dog.deleteMany({gender:'male'});
{ "acknowledged" : true, "deletedCount" : 3 }
```

db.dog.find(gender:'female'}