1. a)
Consider the Sailor database given below. The primary keys are underlined.
Assume relevant data types for attributes.

SAILORS(<u>Sid</u>, Sname, Rating, Age)
BOATS(<u>Bid</u>, Bname, Colour)
RESERVES(<u>Sid, Bid, day</u>)

Create the above tables in SQL. Specify primary and foreign keys properly. Enter at least 5 tuples in each table with relevant data. Solve the following queries.

i. Find the names of sailors who have reserved at least one boat.
ANS:select distinct(sname) from soilr,res where rsid=sid

ii. Find the Sid's of sailors who have reserved a red (or) a green boat.
ANS:select rsid from boat,res where rbid=bid and clr="block" or clr="blue";

iii. Find the Sid's of sailors who have not reserved a boat.
ANS:select sid from soilr,res where sid not in(select rsid from res);

ANS:

```
create table soilr(sid int primary key,
sname varchar(10),
rate int, age int);
create table boat( bid int primary key,
bname varchar(19),
clr varchar(10));

create table res(rsid int,
rbid int,
day int,
foreign key(rsid) references soilr(sid),
foreign key(rbid) references boat(bid));

insert into soilr values(111,"aa",4,23);
insert into soilr values(112,"aa",4,23);
insert into soilr values(113,"aa",4,23);
insert into boat values(121,"baa","block");
insert into res values(111,121,45);

select distinct(sname) from soilr,res where rsid=sid
select rsid from boat,res where rbid=bid and clr="block" or clr="blue";
select sid from soilr,res where sid not in(select rsid from res);
```

1B.//Consider the following restaurant database with the following attributes -
Name, address –(building, street, area, pincode),id, cuisine, nearby landmarks, online delivery- yes/no, famous for(name of the dish)

Create 10 collections with data relevant to the following questions. Write and execute MongoDB queries:
i. List the name and address of all restaurants in Bangalore with Italian cuisine
ii. List the name, address and nearby landmarks of all restaurants in Bangalore where north Indianthali is available

ANS:

```
use restaurant
switched to db restaurant
db.createCollection(“rest")
{ "ok" : 1 }
db.rest.insert({name:"McD",building:12,street:"KR",area:"JN",pin:072,ID:1,cuisine:"fast
food",landmark:"post office",od:"yes",famousfor:"cheese burger"});

WriteResult({ "nInserted" : 1 })

db.rest.insert({name:"Tailo",building:14, street:"MG", area:"Bangalore",pin:45,ID:2,
cuisine:"Italian",landmark:"bank",od:"yes",famousfor:"pasta"})

WriteResult({ "nInserted" : 1 })

db.rest.insert({name:"Kesar",building:15, street:"SS",area:"Bangalore",pin:55,ID:3, cuisine:"north
indian",landmark:"milk factory",od:"no",famousfor:"north indian thali"})

WriteResult({ "nInserted" : 1 })

//List the name and address of all restaurants in Bangalore with Italian cuisine.
db.rest.find({cuisine:"Italian",area:"Bangalore"},
{name:1,building:1,street:1,area:1,pin:1,_id:0}).pretty()

{ "name" : "Tailo", "building" : 14 , “street” : “MG” , “area” : “Bangalore” , “pin” : 45 }

//List the name, address and nearby landmarks of all restaurants in Bangalore where north
Indianthali is available.
db.rest.find({famousfor:"north indian thali",area:"Bangalore"},
{name:1,building:1,street:1,area:1,pin:1,landmark:1,_id:0})

{ "name" : "Kesar", "building" : 15 , “street” : “SS” , “area” : “Bangalore” , “pin” : 55 ,
"landmark" : "milk factory" }
```

//2A)Consider the Employee database given below. The primary keys are underlined. Assume relevant data types for attributes.
EMPLOYEE (Fname, Lname, SSN, Addrs, Sex, Salary, SuperSSN, Dno)
DEPARTMENT (Dname, Dnumber, MgrSSN, MgrStartDate)
PROJECT(Pno, Pname, Dnum)
WORKS_ON (ESSN, Pno, Hours)
Create the above tables in SQL. Specify primary and foreign keys properly. Enter at least 5 tuples in each table with relevant data. Solve the following queries.

i. Retrieve the name of all employees whose salary is greater than the salary of all employees in dept 5.
ii. Retrieve the ssn of all employees who work on project numbers 1,2 or 3
iii. Display the total Number of hours put in by all employees on every project.

```
CREATE TABLE EMPLOYEE (
Fname VARCHAR(10) NOT NULL,
Lname VARCHAR(15) NOT NULL,
Ssn VARCHAR(9) PRIMARY KEY,
Bdate DATE,
Address VARCHAR(40),
Sex CHAR,
Salary INT,
Super_ssn VARCHAR(9),
Dno INT
);

CREATE TABLE DEPARTMENT(
Dname VARCHAR(15) NOT NULL,
Dnumber INT PRIMARY KEY,
Mgr_ssn VARCHAR(9) NOT NULL,
Mgr_start_date DATE,
FOREIGN KEY (Mgr_ssn) REFERENCES EMPLOYEE(Ssn),
);

CREATE TABLE PROJECT
(
Pnumber INT PRIMARY KEY,
Pname VARCHAR(20) NOT NULL,
Dnum INT,
FOREIGN KEY (Dnum) REFERENCES DEPARTMENT(Dnumber),
);

CREATE TABLE WORKS_ON
(
Essn VARCHAR(9),
Pno INT,
Hours INT,
PRIMARY KEY(Essn, Pno),
FOREIGN KEY (Pno) REFERENCES PROJECT(Pnumber),
FOREIGN KEY (Essn) REFERENCES EMPLOYEE(Ssn)
);

INSERT INTO EMPLOYEE Values('John','Smith', 123456789, '09-JAN-65', '731 Fondren,
Houston, TX', 'M', 30000, NULL, NULL);
INSERT INTO EMPLOYEE Values('Franklin','Wong', 333445555, '08-DEC-55', '638 Voss,
Houston, TX', 'M', 40000, NULL, NULL);
INSERT INTO EMPLOYEE Values('Alicia','Zelaya', 999887777, '19-JAN-68', '3321 Castle,
Spring, TX', 'F', 25000, NULL, NULL);
INSERT INTO EMPLOYEE Values('Jennifer','Wallace', 987654321, '01-JAN-65', '731 Fondren,
Houston, TX', 'F', 43000, NULL, NULL);
INSERT INTO EMPLOYEE Values('Ramesh','Narayan', 666884444, '01-JAN-65', '731 Fondren,
Houston, TX', 'M', 38000, NULL, NULL);
INSERT INTO EMPLOYEE Values('Joyce','English', 453453453, '01-JAN-65', '731 Fondren,
Houston, TX', 'F', 25000, NULL, NULL);
INSERT INTO EMPLOYEE Values('Ahmad','Jabbar', 987987987, '01-JAN-65', '731 Fondren,
Houston, TX', 'M', 25000, NULL, NULL);
INSERT INTO EMPLOYEE Values('James','Borg', 888665555, '01-JAN-65', '731 Fondren,
Houston, TX', 'M', 55000, NULL, NULL);

INSERT INTO DEPARTMENT Values('Research',5,'333445555','22-MAY-1988');
INSERT INTO DEPARTMENT Values('Administration',4,'987654321','01-JAN-1995');
INSERT INTO DEPARTMENT Values('Headquarters',1,'888665555','19-JUN-1981');
INSERT INTO DEPARTMENT Values('Development',2,'666884444','22-MAY-1988');
INSERT INTO DEPARTMENT Values('Management',3,'999887777','01-JAN-1995');

INSERT INTO PROJECT Values(1,‘ProductX’,5);
INSERT INTO PROJECT Values(2,'ProductY',5);
INSERT INTO PROJECT Values(3,'ProductZ',2);
INSERT INTO PROJECT Values(10,'Computerization',1);
INSERT INTO PROJECT Values(20,'Reorganization',3);
INSERT INTO PROJECT Values(30,'Newbenefits',4);

INSERT INTO WORKS_ON Values('123456789',1,32);
INSERT INTO WORKS_ON Values('123456789',2,7);
INSERT INTO WORKS_ON Values('666884444',3,40);
INSERT INTO WORKS_ON Values('453453453',1,20);
INSERT INTO WORKS_ON Values('453453453',2,20);
INSERT INTO WORKS_ON Values('333445555',2,10);
INSERT INTO WORKS_ON Values('333445555',3,10);
INSERT INTO WORKS_ON Values('333445555',10,10);
INSERT INTO WORKS_ON Values('333445555',20,10);
INSERT INTO WORKS_ON Values('999887777',30,30);
INSERT INTO WORKS_ON Values('999887777',10,10);
INSERT INTO WORKS_ON Values('987987987',10,35);
INSERT INTO WORKS_ON Values('987987987',30,5);
INSERT INTO WORKS_ON Values('987654321',30,20);
INSERT INTO WORKS_ON Values('987654321',20,15);
INSERT INTO WORKS_ON Values('888665555',20,NULL);
```

//Retrieve the name of all employees whose salary is greater than the salary of all employees in dept 5.

```
SELECT Fname,Lname
FROM EMPLOYEE
WHERE Salary >all (SELECT Salary FROM EMPLOYEE WHERE Dno=5);
```

//Retrieve the social security numbers of all employees who work on project numbers 1,2, or 3.

```
SELECT DISTINCT Essn
FROM WORKS_ON
WHERE PNO IN (1, 2, 3);
```

//Display the total Number of hours put in by all employees on every project.

```
SELECT Pno,SUM(Hours)
FROM WORKS_ON
GROUP BY Pno;
```

Consider the following restaurant table with the following attributes -
Name, address –(building, street, area, pincode), id, cuisine, nearby landmarks, online delivery-(yes/no), famousfor(name of the dish)

2B)//Create 10 collections with data relevant to the following questions. Write and execute MongoDB queries:

i. List the name, address and nearby landmarks of all restaurants in Bangalore where north Indian thali is available

ii. List the name and address of restaurants and also the dish the restaurant is famous for, in Bangalore.

```
use restaurant
switched to db restaurant
db.createCollection(“rest")
{ "ok" : 1 }
db.rest.insert({name:"McD",building:12, street:"KR", area:"JN" pin:072,ID:1, cuisine:"fast food",landmark:"post office",od:"yes",famousfor:"cheese burger"})

WriteResult({ "nInserted" : 1 })
```

```
db.rest.insert({name:"Tailo",building:14, street:"MG", area:"Bangalore" pin:45,ID:2,
cuisine:"Italian",landmark:"bank",od:"yes",famousfor:"pasta"})

WriteResult({ "nInserted" : 1 })

db.rest.insert({name:"Kesar",building:15, street:”SS”,area:"Bangalore",pin:55,ID:3, cuisine:"north
indian",landmark:"milk factory",od:"no",famousfor:"north indian thali”})

WriteResult({ "nInserted" : 1 })

//List the name, address and nearby landmarks of all restaurants in Bangalore where north
Indianthali is available.
db.rest.find({famousfor:"north indian thali",area:"Bangalore"},
{name:1,building:1,street:1,area:1,pin:1,landmark:1,_id:0})

{ "name" : "Kesar", "building" : 15 , "street" : "SS" , "area" : "Bangalore" , "pin" : 55 , "landmark" :
"milk factory" }

//List the name and address of restaurants and also the dish the restaurant is famous for, in
Bangalore.
db.rest.find({area:"Bangalore"},{name:1, building:1, street:1, area:1, pin:1, famousfor:1})
```

3A)//Consider the Aircraft database given below. The primary keys are underlined. Assume relevant data types for attributes.

AIRCRAFT (Aircraft ID, Aircraft_name, Cruising_range)
CERTIFIED (Emp ID, Aircraft ID)
EMPLOYEE (Emp ID, Ename, Salary)

Create the above tables in SQL. Specify primary and foreign keys properly. Enter at least 5 tuples in each table with relevant data. Solve the following queries.

i. Find the employee ID’s of employee who make the highest salary.
ii. Find the name of aircrafts such that all pilots certified to operate them earn more than 50000
iii. Find the employees who are certified for the maximum number of aircrafts.

```
Create table aircraft(
aid varchar(9) primary key,
aname varchar(10),
crange int
);

Create table employees(
eid varchar(9) primary key,
ename varchar(10),
salary int
);

Create table certified(
eid varchar(9)references employees(eid),
aid varchar(9)references aircraft(aid)
);

insert into aircraft values('B001','Boeing',4000);
insert into aircraft values('B002','Boeing',2500);
insert into aircraft values('BB003','Blackbeard',6000);
insert into aircraft values('S004','Supermarine',8000);
insert into aircraft values('L005','Lockheed',2100);

insert into employees values(1,'Johnny',40000);
insert into employees values(2,'Timmy',60000);
insert into employees values(3,'Lawrence',70000);
insert into employees values(4,'Zuzu',90000);
insert into employees values(5,'Matt',80000);

insert into certified values(1,'B001');
insert into certified values(1,'B002');
insert into certified values(3,'S004');
insert into certified values(4,'S004');
insert into certified values(5,'L005');
insert into certified values(2,'B002');
insert into certified values(4,'BB003');
insert into certified values(3,'BB003');
insert into certified values(4,'L005');
```

//Find Emp ID of employee who makes highest salary

```
SELECT eid
FROM employees
WHERE salary=(SELECT MAX(salary) FROM employees);
```

//Find the name of aircrafts such that all pilots certified to operate them earn more than 50000

```
SELECT distinct aname
FROM aircraft a,certified c,employees e
WHERE a.aid=c.aid and c.eid=e.eid and e.salary>50000;
```

//Find the employees who are certified for the maximum number of aircrafts.

```
select ename,max(a.aid) from employees e,certified c,aircraft a where e.eid=c.eid and a.aid=c.aid
```

3B)//Consider the following restaurant table with the following attributes -
Name, address –(building, street, area, pincode), id, cuisine, nearby landmarks, online delivery- (yes/no), famous for(name of the dish)

Create 10 collections with data relevant to the following questions. Write and execute MongoDB queries:

i. List the name, address and nearby landmarks of all restaurants in Bangalore where north Indian thali is available.

ii. List the name and address of restaurants and also the dish the restaurant is famous for, in Bangalore where online delivery is available.

```
use restaurant
switched to db restaurant
db.createCollection(“rest")
{ "ok" : 1 }
db.rest.insert({name:"McD",building:12, street:"KR", area:"JN" pin:072,ID:1, cuisine:"fast
food",landmark:"post office",od:"yes",famousfor:"cheese burger"})

WriteResult({ "nInserted" : 1 })
```

```
db.rest.insert({name:"Tailo",building:14, street:"MG", area:"Bangalore" pin:45,ID:2,
cuisine:"Italian",landmark:"bank",od:"yes",famousfor:"pasta"})

WriteResult({ "nInserted" : 1 })

db.rest.insert({name:"Kesar",building:15, street:"SS",area:"Bangalore",pin:55,ID:3, cuisine:"north
indian",landmark:"milk factory",od:"no",famousfor:"north indian thali"})

WriteResult({ "nInserted" : 1 })

//List the name, address and nearby landmarks of all restaurants in Bangalore where north
Indianthali is available.
db.rest.find({famousfor:"north indian thali",area:"Bangalore"},
{name:1,building:1,street:1,area:1,pin:1,landmark:1,_id:0})

{ "name" : "Kesar", "building" : 15 , “street” : “SS” , “area” : “Bangalore” , “pin” : 55 ,
"landmark" : "milk factory" }

//List the name and address of restaurants and also the dish the restaurant is famous for, in
Bangalore where online delivery is available.
db.rest.find({area:"Bangalore",od:"yes"},{name:1, building:1, street:1, arear:1, pin:1, famousfor:1})
```

4A)//Consider the Supply-Parts database given below. The primary keys are underlined. Assume relevant data types for attributes.

SUPPLIER (<u>Sid</u>, Sname, Address)
PART (<u>PID</u>, Pname, Color)
SHIPMENT (<u>Sid, PID</u>, Cost)

Create the above tables in SQL. Specify primary and foreign keys properly. Enter at least 5 tuples in each table with relevant data. Solve the following queries.

i. Find the Sid’s of suppliers who supply a green part
ii. For every supplier print the name of the supplier and the total number of parts that he/she supplies
iii. Update the part color supplied by supplier s3 to yellow

```
CREATE TABLE SUPPLIER
(
Sid int NOT NULL PRIMARY KEY,
Sname varchar(16) NOT NULL UNIQUE,
Address varchar(20) NOT NULL
);

CREATE TABLE PART
(
Pid int NOT NULL PRIMARY KEY,
Pname varchar(18) NOT NULL,
Color varchar(10) NOT NULL,
);

CREATE TABLE SHIPMENT
(
Sid int NOT NULL REFERENCES SUPPLIER,
Pid int NOT NULL REFERENCES PART,
Cost int NOT NULL,
PRIMARY KEY (Sid, Pid)
);

CREATE TABLE SHIPMENT
(
Sid int,
Pid int,
Cost int,
primary key(Sid,Pid),
foreign key(Sid) references SUPPLIER(Sid),
foreign key(Pid) references PART(Pid));


INSERT INTO SUPPLIER VALUES (1, 'Smith', 'London');
INSERT INTO SUPPLIER VALUES (2, 'Jones', 'Paris');
INSERT INTO SUPPLIER VALUES (3, 'Blake', 'Paris');
INSERT INTO SUPPLIER VALUES (4, 'Clark', 'London');
INSERT INTO SUPPLIER VALUES (5, 'Adams', 'Athens');

INSERT INTO PART VALUES (1, 'Nut', 'Red');
INSERT INTO PART VALUES (2, 'Bolt', 'Green');
INSERT INTO PART VALUES (3, 'Screw', 'Blue');
INSERT INTO PART VALUES (4, 'Screw', 'Red');
INSERT INTO PART VALUES (5, 'Cam', 'Blue');
INSERT INTO PART VALUES (6, 'Cog', 'Red');

INSERT INTO SHIPMENT VALUES (1, 1, 300);
INSERT INTO SHIPMENT VALUES (1, 2, 200);
INSERT INTO SHIPMENT VALUES (1, 3, 400);
INSERT INTO SHIPMENT VALUES (1, 4, 200);
INSERT INTO SHIPMENT VALUES (1, 5, 100);
INSERT INTO SHIPMENT VALUES (1, 6, 100);
INSERT INTO SHIPMENT VALUES (2, 1, 300);
INSERT INTO SHIPMENT VALUES (2, 2, 400);
INSERT INTO SHIPMENT VALUES (3, 2, 200);
INSERT INTO SHIPMENT VALUES (4, 2, 200);
INSERT INTO SHIPMENT VALUES (4, 4, 300);
INSERT INTO SHIPMENT VALUES (4, 5, 400);

//Find the Sids of suppliers who supply green part

SELECT Distinct S.Sid
FROM SUPPLIER S, PART P, SHIPMENT C
WHERE C.Sid=S.Sid AND P.PID=C.PID AND P.Color like 'Green';

//For every supplier print the name of the supplier and the total number of parts that he/she supplies.
SELECT S.Sname, COUNT(*) as PartCount
FROM SUPPLIER S, SHIPMENT C, PART P
WHERE C.Sid = S.Sid and P.PID = C.PID
GROUP BY S.Sname, S.Sid

//Update part color supplied by supplier S3 to Yellow
UPDATE PART
SET Color='Yellow'
WHERE PID IN (SELECT C.PID FROM  SUPPLIER S, SHIPMENT C WHERE C.Sid=S.Sid  and
C.Sid=3);
```

4B)//Consider the following Tourist places table with the following attributes -
Place, address – (state), id, tourist attractions,best time of the year to visit,modes of transport(include nearest airport, railway station etc), accommodation, food - what not to miss for sure

Create 10 collections with data relevant to the following questions. Write and execute MongoDB queries:

i. List all the tourist places of Karnataka
ii. List the tourist attractions of Kerala. Exclude accommodation and food

```
db.createCollection("tourist")
{ "ok" : 1 }
db.tourist.insert([{place:"bangalore",address:"karnataka",id:1,tour_att:"att1",time:"jan",mode:"train",acc:"acc1",food:"chicken"}])

db.tourist.insert([{place:"kochi",address:"kerala",id:2,tour_att:"att2",time:"feb",mode:"boat",acc:"acc2",food:"fish fry"}])

db.tourist.insert([{place:"agra",address:"delhi",id:3,tour_att:"taj mahal",time:"march",mode:"car",acc:"acc3",food:"petha"}])

//List all the tourist places of Karnataka
db.tourist.find({address:"karnataka"},{tour_att:true}).pretty()

{ "_id" : ObjectId("5c27375e7fd87cec5944fab7"), "tour_att" : "att1" }

//List the tourist attractions of Kerala. Exclude accommodation and food.
db.tourist.find({address:"kerala"},{acc:0,food:0}).pretty()
{
        "_id" : ObjectId("5c2737be7fd87cec5944fab8"),
        "place" : "kochi",
        "address" : "kerala",
        "id" : 2,
        "tour_att" : "att2",
        "time" : "feb",
        "mode" : "boat",

}
```

5A)//Consider the Aircraft database given below. The primary keys are underlined. Assume relevant data types for attributes.

AIRCRAFT (<u>Aircraft ID</u>, Aircraft_name, Cruising_range)
CERTIFIED (<u>Emp ID, Aircraft ID</u>)
EMPLOYEE (<u>Emp ID</u>, Ename, Salary)

Create the above tables in SQL. Specify primary and foreign keys properly. Enter at least 5 tuples in each table with relevant data. Solve the following queries.

i. Find the names of pilots certified for Boeing aircraft
ii. Arrange the flight no with respect to the ascending order of distance.
iii. Find the name of pilots who can operate flights with a range greater than 3000 miles but are not certified on any Boeing aircraft.

```
Create table aircraft(
aid varchar(9) primary key,
aname varchar(10),
crange int
);

Create table employees(
eid varchar(9) primary key,
ename varchar(10),
salary int
);

Create table certified(
eid varchar(9)references employees(eid),
aid varchar(9)references aircraft(aid)
);

insert into aircraft values('B001','Boeing',4000);
insert into aircraft values('B002','Boeing',2500);
insert into aircraft values('BB003','Blackbeard',6000);
insert into aircraft values('S004','Supermarine',8000);
insert into aircraft values('L005','Lockheed',2100);

insert into employees values(1,'Johnny',40000);
insert into employees values(2,'Timmy',60000);
insert into employees values(3,'Lawrence',70000);
insert into employees values(4,'Zuzu',90000);
insert into employees values(5,'Matt',80000);

insert into certified values(1,'B001');
insert into certified values(1,'B002');
insert into certified values(3,'S004');
insert into certified values(4,'S004');
insert into certified values(5,'L005');
insert into certified values(2,'B002');
insert into certified values(4,'BB003');
insert into certified values(3,'BB003');
insert into certified values(4,'L005');

//Find names of pilots certified to fly Boeing

SELECT DISTINCT E.ename
FROM employees E, certified C, aircraft A
WHERE E.eid = C.eid AND C.aid = A.aid AND A.aname='Boeing';

//Arrange flight no with respect to ascending order of distance

SELECT aid
FROM aircraft
ORDER BY crange ASC;

//Find the name of pilots who can operate flights with a range greater than 3000 miles but are not
certified on any Boeing aircraft.

select distinct(ename) from employees E,certified C, aircraft A where A.crange > 3000 and C.aid
NOT in(select aid from aircraft A where A.aname='Boeing') and E.eid = C.eid
```

5B)//Consider the following Tourist places table with the following attributes -

Place, address –(state, id), tourist attractions,best time of the year to visit,modes of transport(include nearest airport, railway station etc), accommodation, food - what not to miss for sure

Create 10 collections with data relevant to the following questions. Write and execute MongoDB queries:

i. List the tourist attractions of Kerala. Exclude accommodation and food.
ii. List the places sorted state wise.

```
db.createCollection("tourist")
{ "ok" : 1 }
db.tourist.insert([{place:"bangalore",address:"karnataka",id:1,tour_att:"att1",time:"jan",mode:"train",acc:"acc1",food:"chicken"}])

db.tourist.insert([{place:"kochi",address:"kerala",id:2,tour_att:"att2",time:"feb",mode:"boat",acc:"acc2",food:"fish fry"}])

db.tourist.insert([{place:"agra",address:"delhi",id:3,tour_att:"taj mahal",time:"march",mode:"car",acc:"acc3",food:"petha"}])

//List the tourist attractions of Kerala. Exclude accommodation and food.
db.tourist.find({address:"kerala"},{acc:0,food:0}).pretty()
{
	"_id" : ObjectId("5c2737be7fd87cec5944fab8"),
	"place" : "kochi",
	"address" : "kerala",
	"id" : 2,
	"tour_att" : "att2",
	"time" : "feb",
	"mode" : "boat",

}

//List the places sorted state wise.
db.tourist.find({},{place:1}).sort({address:1})
{ "_id" : ObjectId("5c2738557fd87cec5944fab9"), "place" : "agra" }
{ "_id" : ObjectId("5c27375e7fd87cec5944fab7"), "place" : "bangalore" }
{ "_id" : ObjectId("5c2737be7fd87cec5944fab8"), "place" : "kochi" }
```

6A)//Consider the Employee database given below. The primary keys are underlined. Assume relevant data types for attributes.

EMPLOYEE (Fname, Lname, <u>SSN</u>, Addrs, Sex, Salary, SuperSSN, Dno)
DEPARTMENT (Dname, <u>Dnumber</u>, MgrSSN, MgrStartDate)
DEPENDENT(<u>Dname, ESSN</u>)

Create the above tables in SQL. Specify primary and foreign keys properly. Enter at least 5 tuples in each table with relevant data. Solve the following queries.

i. For each department, retrieve the department name and the average salary of all employees working in that department
ii. List the names of managers who have at least one dependent
iii. Display the details of all departments having ‘tech’ as their substring

```
CREATE TABLE EMPLOYEE15 (
Fname VARCHAR(10) NOT NULL,
Lname VARCHAR(15) NOT NULL,
Ssn VARCHAR(9) PRIMARY KEY,
Bdate DATE,
Address VARCHAR(40),
Sex CHAR,
Salary INT,
Super_ssn VARCHAR(9),
Dno INT
);

CREATE TABLE DEPARTMENT(
Dname VARCHAR(15) NOT NULL,
Dnumber INT PRIMARY KEY,
Mgr_ssn VARCHAR(9) NOT NULL,
Mgr_start_date DATE,
FOREIGN KEY (Mgr_ssn) REFERENCES EMPLOYEE15(Ssn),
);

CREATE TABLE DEPENDENT
(
Dname VARCHAR(15),
Essn VARCHAR(9),
PRIMARY KEY(Essn, Dname),
FOREIGN KEY (Essn) REFERENCES EMPLOYEE15(Ssn)
);

INSERT INTO EMPLOYEE15 Values('John','Smith', 123456789, '09-JAN-65', '731 Fondren,
Houston, TX', 'M', 30000, NULL, NULL);
INSERT INTO EMPLOYEE15 Values('Franklin','Wong', 333445555, '08-DEC-55', '638 Voss,
Houston, TX', 'M', 40000, NULL, NULL);
INSERT INTO EMPLOYEE15 Values('Alicia','Zelaya', 999887777, '19-JAN-68', '3321 Castle,
Spring, TX', 'F', 25000, NULL, NULL);
INSERT INTO EMPLOYEE15 Values('Jennifer','Wallace', 987654321, '01-JAN-65', '731 Fondren,
Houston, TX', 'F', 43000, NULL, NULL);
INSERT INTO EMPLOYEE15 Values('Ramesh','Narayan', 666884444, '01-JAN-65', '731 Fondren,
Houston, TX', 'M', 38000, NULL, NULL);
INSERT INTO EMPLOYEE15 Values('Joyce','English', 453453453, '01-JAN-65', '731 Fondren,
Houston, TX', 'F', 25000, NULL, NULL);
INSERT INTO EMPLOYEE15 Values('Ahmad','Jabbar', 987987987, '01-JAN-65', '731 Fondren,
Houston, TX', 'M', 25000, NULL, NULL);
INSERT INTO EMPLOYEE15 Values('James','Borg', 888665555, '01-JAN-65', '731 Fondren,
Houston, TX', 'M', 55000, NULL, NULL);

INSERT INTO DEPARTMENT Values('Research',5,'333445555','22-MAY-1988');
INSERT INTO DEPARTMENT Values('Administration',4,'987654321','01-JAN-1995');
INSERT INTO DEPARTMENT Values('Headquarters',1,'888665555','19-JUN-1981');
INSERT INTO DEPARTMENT Values('tech-Development',2,'666884444','22-MAY-1988');
INSERT INTO DEPARTMENT Values('tech-Management',3,'999887777','01-JAN-1995');

INSERT INTO DEPENDENT VALUES('Alice','333445555');
INSERT INTO DEPENDENT VALUES('Theodore','333445555');
INSERT INTO DEPENDENT VALUES('Joy','333445555');
INSERT INTO DEPENDENT VALUES('Abner','987654321');
INSERT INTO DEPENDENT VALUES('Michael','123456789');
INSERT INTO DEPENDENT VALUES('Alice','123456789');
INSERT INTO DEPENDENT VALUES('Elizabeth','123456789');

//For each dept, retreive the dept name and avg salary of all employees working in that department
```

```
SELECT Dname, AVG(Salary)
FROM DEPARTMENT, EMPLOYEE15;

//List names of managers who have at least one dependent

SELECT Fname, Lname
FROM EMPLOYEE15
WHERE EXISTS(SELECT * FROM DEPENDENT WHERE Ssn=Essn)
AND
EXISTS (SELECT * FROM DEPARTMENT WHERE Ssn=Mgr_ssn);

//Display details of all departments having 'tech' as their substring

/*SELECT *
FROM DEPARTMENT
WHERE EXISTS(SUBSTRING('tech',1) AS ExtractString);*/

select * from DEPARTMENT where Dname like'%sear%'
```

6B)//Consider the following Tourist places table with the following attributes -
Place, address – (state, id), tourist attractions,best time of the year to visit,modes of transport(include nearest airport, railway station etc), accommodation, food - what not to miss for sure Create 10 collections with data relevant to the following questions. Write and execute MongoDB queries:

i. List all the tourist places of Karnataka
ii. List the places sorted state wise

```
db.createCollection("tourist")
{ "ok" : 1 }
db.tourist.insert([{place:"bangalore",address:"karnataka",id:1,tour_att:"att1",time:"jan",mode:"train",acc:"acc1",food:"chicken"}])

db.tourist.insert([{place:"kochi",address:"kerala",id:2,tour_att:"att2",time:"feb",mode:"boat",acc:"acc2",food:"fish fry"}])

db.tourist.insert([{place:"agra",address:"delhi",id:3,tour_att:"taj mahal",time:"march",mode:"car",acc:"acc3",food:"petha"}])

//List all the tourist places of Karnataka.
db.tourist.find({address:"karnataka"},{tour_att:true}).pretty()
{ "_id" : ObjectId("5c27375e7fd87cec5944fab7"), "tour_att" : "att1" }

//List the places sorted state wise.
db.tourist.find({},{place:1}).sort({address:1})
{ "_id" : ObjectId("5c2738557fd87cec5944fab9"), "place" : "agra" }
{ "_id" : ObjectId("5c27375e7fd87cec5944fab7"), "place" : "bangalore" }
{ "_id" : ObjectId("5c2737be7fd87cec5944fab8"), "place" : "kochi" }
```

7A)//Consider the following Accident Tracker Schema. The primary keys are underlined.
PERSON (driver – id #: String, name: string, address: string)
CAR (Regno: string, model: string, year: int)
ACCIDENT (report-number: int, acc_date: date, location: string)
OWNS (driver-id #: string, Regno: string)
PARTICIPATED (driver-id: string, Regno: string, report-number: int, damageamount: int)
Create the above tables in SQL. Specify primary and foreign keys properly. Enter at least 5 tuples in each table with relevant data. Solve the following queries.

i. Display the unique Regno’s of the cars involved in accidents.
ii. Display the car Regno and model of the car which has the maximum damage amount.
iii. Displaythe number of cars owned by each driver.

```
//create tables

create table PERSON (
Did VARCHAR(10) PRIMARY KEY,
Pname VARCHAR(10),
Address VARCHAR(60));

create table CAR (
Regno VARCHAR(10) PRIMARY KEY,
Model VARCHAR(20),
Year INT);

create table ACCIDENT (
Repno INT PRIMARY KEY,
Date DATE,
Loc VARCHAR(20));

 create table OWNS (
Odid  VARCHAR(10),
Oregno VARCHAR(10),
Primary Key(Odid,Oregno),
Foreign key(Odid) references PERSON(Did) on delete cascade,
Foreign key(Oregno) references CAR(Regno) on delete cascade);

create table PARTICIPATED (
Pdid VARCHAR(10),
Pregno VARCHAR(10),
Prepno INT,
Damage INT,
Primary key(Pdid, Pregno, Prepno),
Foreign key (Pdid) references PERSON(Did) on delete cascade,
Foreign key (Pregno) references CAR(Regno) on delete cascade,
Foreign key(Prepno) references ACCIDENT(Repno) on delete cascade);


//insert values

insert into PERSON values('1','Steve','Frankfurt');
insert into PERSON values('2','Dustin','Perryridge');
insert into PERSON values('3','Mike','Brooklyn');
insert into PERSON values('4','Lucas','Perryridge ');
insert into PERSON values('5','John',' Brooklyn ');
insert into PERSON values('6','Antony','Hellington');

insert into CAR values('KA04','BMW',2000);
insert into CAR values('KA05','Ford',2002);
insert into CAR values('KA03','Maruthi',1999);
insert into CAR values('KA02','Tata',2002);
insert into CAR values('KA01', 'Audi',2003);
insert into CAR values('KA08', 'Maruthi',2003);
insert into CAR values('KA06', 'Maruthi',2003);
insert into CAR values('KA07', 'BMW',2003);

insert into ACCIDENT values(12,'01-Jun-2001','Frankfurt');
insert into ACCIDENT values(25,'02-Jul-2002','Brooklyn');
insert into ACCIDENT values(512,'08-Mar-2000',' Brooklyn');
insert into ACCIDENT values(1024,'25-Oct-2002','AvenueRoad');
insert into ACCIDENT values(1000,'23-Dec-2003','RichmondCircle');
insert into ACCIDENT values(1,'25-Dec-2004','ParkStreet');

insert into OWNS values('1', 'KA04');
insert into OWNS values('1', 'KA06');
insert into OWNS values('2', 'KA07');
insert into OWNS values('2', 'KA05');
insert into OWNS values('3', 'KA03');
insert into OWNS values('4', 'KA02');
insert into OWNS values('5', 'KA01');
insert into OWNS values('6', 'KA08');

insert  into PARTICIPATED values('1', 'KA04',12,1000);
insert  into PARTICIPATED values('1', 'KA06',25,1500);
insert  into PARTICIPATED values('2', 'KA05',512,1500);
insert  into PARTICIPATED values('2', 'KA05',1024,2500);
insert  into PARTICIPATED values('3', 'KA03',1000,1700);
insert  into PARTICIPATED values('4', 'KA02',1,100);

//Display unique car Regnos involved in accidents.

Select distinct (Pregno)
from PARTICIPATED;

//Display the car Regno and model which has the maximum damage amount.

Select  Pregno,Model
from CAR, PARTICIPATED
where Pregno=Regno and Damage in (select max(Damage) from PARTICIPATED);

//Display no. of cars owned by each driver.

Select  Odid,count(*) as no_of_cars
from OWNS
group by Odid;
```

7B)//Consider the following Movie table with the following attributes -
Actor_name,Actor_id, Actor_birthdate, Dirctor_name,Director_id, Director_birthdate, film_title, year of production ,type (thriller, comedy, etc.)

Create 10 collections with data relevant to the following questions. Write and execute MongoDB queries:

i. List all the movies acted by John in the year 2018
ii. List only the actors names and type of the movie directed by Ram

```
db.createCollection("movie")
{ "ok" : 1 }
db.movie.insert([{act_n:"ram",act_id:13,act_bdate:"2/3/1997",dir_n:"williams",dir_id:101,dir_bdate:"12/9/1987",film:"battleship",year:2015,type:"thriller"}])

db.movie.insert([{act_n:"john",act_id:11,act_bdate:"1/2/1998",dir_n:"ram",dir_id:100,dir_bdate:"2/3/1997",film:"john wick",year:2012,type:"killer"}])
db.movie.insert([{act_n:"elly",act_id:12,act_bdate:"4/12/1998",dir_n:"ram",dir_id:100,dir_bdate:"2
/3/1997",film:"aquaman",year:2012,type:"action"}])

db.movie.insert([{act_n:"ram",act_id:13,act_bdate:"2/3/1997",dir_n:"thomas",dir_id:103,dir_bdate:
"12/3/1999",film:"xxx",year:2018,type:"action"}])

db.movie.insert([{act_n:"john",act_id:11,act_bdate:"1/2/1998",dir_n:"ram",dir_id:100,dir_bdate:"2/
3/1997",film:"mr.bean",year:2018,type:"comedy"}])

//List all the movies acted by John in the year 2018.
db.movie.find({$and : [{act_n:"john"},{year:2018}]},{film:1}).pretty()

{ "_id" : ObjectId("5c273f627fd87cec5944fac3"), "film" : "mr.bean" }

//List only the actors names and type of the movie directed by Ram.
db.movie.find({dir_n:"ram"},{act_n:1,type:1}).pretty()
{
        "_id" : ObjectId("5c273e247fd87cec5944fabf"),
        "act_n" : "john",
        "type" : "killer"
}
{
        "_id" : ObjectId("5c273e2e7fd87cec5944fac0"),
        "act_n" : "elly",
        "type" : "action"
}
{
        "_id" : ObjectId("5c273f627fd87cec5944fac3"),
        "act_n" : "john",
        "type" : "comedy"
}
```

//8A)//Consider the Cricket database given below. The primary keys are underlined. Assume
PLAYER (PId, Lname, Fname, Country, Yborn, Bplace)
MATCH (MatchId, Team1,Team2, Ground, Date, Winner)
BATTING (MatchId, Pid, Nruns, Fours, Sixes)
BOWLING (MatchId, Pid, Novers, Maidens, Nruns, Nwickets)

Create the above tables in SQL. Specify primary and foreign keys properly. Enter at least 5 tuples in each table with relevant data. Solve the following queries.

i. Display the sorted list of ground names where Australia has played as team1
ii. Find the match information of all matches in which Dhoni did batting.
iii. Find the names of players who did batting in match 2689

```
create table PLAYER (
Pid int,
Lname varchar(10),
Fname varchar(10),
Country varchar(10),
Yborn date,
Bplace varchar(10),
primary key (Pid)
);

create table MATCHING (
MatchID int,
Team1 varchar(10),
Team2 varchar(10),
Ground varchar(10),
Date date,
Winner varchar(10),
primary key(MatchID)
);

create table BATTING (
MatchID int,
Pid int,
Nruns int,
Fours int,
Sixes int,
primary key(MatchID,Pid),foreign key(MatchID) references MATCHING(MatchID),foreign
key(Pid) references PLAYER(Pid)
);

create table BOWLING (
MatchID int,
Pid int,
Novers int,
Maidens varchar(10),
Nruns int,
Nwickets int,
primary key(MatchID,Pid),foreign key(MatchID) references MATCHING(MatchID),foreign
key(Pid) references PLAYER(Pid)
);

//insert values
insert into PLAYER values("1,"aa","Dhoni","india","01-jan-88","chennai");
insert into MATCHING values("12,"Australia","D2","china","01-jan-2012","india");
insert into BATTING values("12,1,20,4,4);
insert into BOWLING values("12,1,12,"cc",20,2);

//Display the sorted list of ground names where Australia has played as team1.
SELECT Ground
FROM MATCHING
WHERE Team1='Australia'
ORDER BY Ground;

//Find the match information of all matches in which Dhoni did batting.
SELECT *
FROM (PLAYER natural join MATCHING) natural join BATTING
WHERE Fname='Dhoni';

//Find the names of players who did batting in match 2689.
SELECT Fname,Lname
FROM (PLAYER natural join MATCHING) natural join BATTING
WHERE MatchID=2686;

//8B)Consider the following Movie table with the following attributes -
```

Actor_name,Actor_id, Actor_birthdate , Dirctor_name,Director_id, Director_birthdate, film_title, year of production ,type (thriller, comedy, etc.)

Create 10 collections with data relevant to the following questions. Write and execute MongoDB queries:

i. List all the movies acted by John and Elly in the year 2012.
ii. List only the name and type of the movie where Ram has acted sorted by movie names

```
db.createCollection("movie")
{ "ok" : 1 }
db.movie.insert([{act_n:"ram",act_id:13,act_bdate:"2/3/1997",dir_n:"williams",dir_id:101,dir_bdate:"12/9/1987",film:"battleship",year:2015,type:"thriller"}])

db.movie.insert([{act_n:"john",act_id:11,act_bdate:"1/2/1998",dir_n:"ram",dir_id:100,dir_bdate:"2/3/1997",film:"john wick",year:2012,type:"killer"}])

db.movie.insert([{act_n:"elly",act_id:12,act_bdate:"4/12/1998",dir_n:"ram",dir_id:100,dir_bdate:"2/3/1997",film:"aquaman",year:2012,type:"action"}])

db.movie.insert([{act_n:"ram",act_id:13,act_bdate:"2/3/1997",dir_n:"thomas",dir_id:103,dir_bdate:"12/3/1999",film:"xxx",year:2018,type:"action"}])

db.movie.insert([{act_n:"john",act_id:11,act_bdate:"1/2/1998",dir_n:"ram",dir_id:100,dir_bdate:"2/3/1997",film:"mr.bean",year:2018,type:"comedy"}])

//List all the movies acted by John and Elly in the year 2012.
db.movie.find({$and : [{act_n:{$in : ["john","elly"]}},{year:2012}]},{film:1}).pretty()

{ "_id" : ObjectId("5c273e247fd87cec5944fabf"), "film" : "john wick" }
{ "_id" : ObjectId("5c273e2e7fd87cec5944fac0"), "film" : "aquaman" }

//ii.      List only the name and type of the movie where Ram has acted sorted by movie names.
db.movie.find({act_n:"ram"},{film:1,type:1}).sort({film:1}).pretty()

{ "_id" : ObjectId("5c273ddb7fd87cec5944fabd"), "film" : "battleship", "type" : "thriller" }
{ "_id" : ObjectId("5c273ef97fd87cec5944fac1"), "film" : "xxx", "type" : "action" }
```

9A)//Consider the following shipment schema. The primary keys are underlined. Assume relevant data types for attributes.

CUSTOMER (<u>cust #</u> , cname, city)
ORDER (<u>order#</u>, odate, cust #, ord-Amt)
ORDER – ITEM (<u>order #, Item #,</u> qty)
ITEM (<u>item #</u>, unit price)
SHIPMENT (<u>order #,</u> ship-date)

Create the above tables in SQL. Specify primary and foreign keys properly. Enter at least 5 tuples in each table with relevant data. Solve the following queries.

i. List the customer names who have placed more than 2 orders.
ii. Find the total order amount for each day
iii. List the customer details who has the largest order amount.

```
//create tables

Create table CUSTOMER(
Cno INT,
Cname VARCHAR(50),
Ccity VARCHAR(50),
Primary key(Cno)
);
```

```
Create table ORDER1(
Ono INT,
Odate DATE,
Ocno INT,
Oamt INT,
Primary key(Ono),
Foreign Key(Ocno) references CUSTOMER(Cno) on delete cascade
);

Create table ITEM(
Ino INT ,
Uprice INT,
Primary key(Ino)
);

Create table ORDER_ITEM(
Iono INT,
Oino  INT,
Qty INT,
Primary key(Iono,Oino),
Foreign Key(Iono) references ORDER1(Ono) on delete cascade,
Foreign Key(Oino) references ITEM(Ino) on delete cascade
);

Create table WAREHOUSE(
Wno INT,
Wcity VARCHAR(50),
Primary Key(Wno)
);

Create table SHIPMENT(
Sono INT,
Swno INT,
Shipdate DATE,
Primary key(Sono,Swno),
Foreign Key(Sono) references ORDER1(Ono) on delete cascade,
Foreign Key(Swno) references WAREHOUSE(Wno) on delete cascade
);

//insert values

Insert into CUSTOMER values (1,'Archie','Bangalore');
Insert into CUSTOMER values (2,'Veronica','Bangalore');
Insert into CUSTOMER values (3,'Betty','Mysore');
Insert into CUSTOMER values (4,'Jughead','Bangalore');
Insert into CUSTOMER values (5,'Cheryl','Mysore');
Insert into CUSTOMER values (6,'Jason','Mangalore');

Insert into ORDER1 values (1, '01-Jun-2016',1,1000);
Insert into ORDER1 values (2, '03-Jun-2016',1,2000);
Insert into ORDER1 values (3, '03-Jun-2016',3,3000);
Insert into ORDER1 values (4, '05-Jun-2016',4,4000);
Insert into ORDER1 values (5, '06-Jun-2017',5,5000);

Insert into ITEM values(1,100);
Insert into ITEM values(2,200);
Insert into ITEM values(3,300);
Insert into ITEM values(4,400);
Insert into ITEM values(10,500);

Insert into ORDER_ITEM values(1,1,2);
Insert into ORDER_ITEM values(1,2,1);
Insert into ORDER_ITEM values(2,3,2);
Insert into ORDER_ITEM values(3,4,3);
Insert into ORDER_ITEM values(4,3,1);
Insert into ORDER_ITEM values(5,2,1);

Insert into WAREHOUSE values(1,'Bangalore');
Insert into WAREHOUSE values(2,'Bangalore');
Insert into WAREHOUSE values(3,'Mysore');
Insert into WAREHOUSE values(4,'Mangalore');
Insert into WAREHOUSE values(5,'Mangalore');

Insert into SHIPMENT values(1,1,'03-Jun-2016');
Insert into SHIPMENT values(2,1,'03-Jun-2016');
Insert into SHIPMENT values(3,2,'05-Jun-2016');
Insert into SHIPMENT values(4,3,'05-Jun-2016');
Insert into SHIPMENT values(5,4,'06-Jun-2016');

//List the customer names who have placed more than 2 orders

Select Cname FROM CUSTOMER
where  (Select count(Ono) from ORDER1 where Cno=Ocno)>=2;

//Find the total order amount for each day.

Select Odate,SUM(Oamt)
from ORDER1
group by Odate;

//List customer details who has the largest order amount.

Select Cno,Cname,Ccity
from CUSTOMER,ORDER1
where Cno=Ocno and Oamt in(select max(Oamt) from ORDER1);
```

//9B)Consider the following Movie table with the following attributes - Actor_name,Actor_id, Actor_birthdate , Dirctor_name,Director_id, Director_birthdate, film_title, year of production ,type (thriller, comedy, etc.)

Create 10 collections with data relevant to the following questions. Write and execute MongoDB queries:

i. List all the movies acted by John in the year 2018
ii. List only the actors names and type of the movie directed by Ram

```
db.createCollection("movie")
{ "ok" : 1 }
db.movie.insert([{act_n:"ram",act_id:13,act_bdate:"2/3/1997",dir_n:"williams",dir_id:101,dir_bdate:"12/9/1987",film:"battleship",year:2015,type:"thriller"}])

db.movie.insert([{act_n:"john",act_id:11,act_bdate:"1/2/1998",dir_n:"ram",dir_id:100,dir_bdate:"2/3/1997",film:"john wick",year:2012,type:"killer"}])
db.movie.insert([{act_n:"elly",act_id:12,act_bdate:"4/12/1998",dir_n:"ram",dir_id:100,dir_bdate:"2
/3/1997",film:"aquaman",year:2012,type:"action"}])

db.movie.insert([{act_n:"ram",act_id:13,act_bdate:"2/3/1997",dir_n:"thomas",dir_id:103,dir_bdate:
"12/3/1999",film:"xxx",year:2018,type:"action"}])

db.movie.insert([{act_n:"john",act_id:11,act_bdate:"1/2/1998",dir_n:"ram",dir_id:100,dir_bdate:"2/
3/1997",film:"mr.bean",year:2018,type:"comedy"}])

//List all the movies acted by John in the year 2018.
db.movie.find({$and : [{act_n:"john"},{year:2018}]},{film:1}).pretty()

{ "_id" : ObjectId("5c273f627fd87cec5944fac3"), "film" : "mr.bean" }

//List only the actors names and type of the movie directed by Ram.
db.movie.find({dir_n:"ram"},{act_n:1,type:1}).pretty()
{
        "_id" : ObjectId("5c273e247fd87cec5944fabf"),
        "act_n" : "john",
        "type" : "killer"
}
{
        "_id" : ObjectId("5c273e2e7fd87cec5944fac0"),
        "act_n" : "elly",
        "type" : "action"
}
{
        "_id" : ObjectId("5c273f627fd87cec5944fac3"),
        "act_n" : "john",
        "type" : "comedy"
}
```

//10A)Consider the following shipment schema. The primary keys are underlined. Assume relevant data types for attributes.

CUSTOMER (<u>cust #</u> , cname, city)
ORDER (<u>order#</u>, odate, cust #, ord-Amt)
ORDER – ITEM (<u>order #</u>, <u>Item</u> #, qty)
ITEM (<u>item #</u>, unit price)
SHIPMENT (<u>order #,</u> ship-date)

Create the above tables in SQL. Specify primary and foreign keys properly. Enter at least 5 tuples in each table with relevant data. Solve the following queries.

i. List name of the customer, no. of orders placed by each customer residing in Bangalore city.
ii. List the names of the customers who have ordered at least 10 items
iii. List the customer names who have not ordered for item no. 10.

```
//create tables

Create table CUSTOMER(
Cno INT,
Cname VARCHAR(50),
Ccity VARCHAR(50),
Primary key(Cno)
);

Create table ORDER1(
Ono INT,
Odate DATE,
Ocno INT,
Oamt INT,
Primary key(Ono),
Foreign Key(Ocno) references CUSTOMER(Cno) on delete cascade
);

Create table ITEM(
Ino INT ,
Uprice INT,
Primary key(Ino)
);

Create table ORDER_ITEM(
Iono INT,
Oino  INT,
Qty INT,
Primary key(Iono,Oino),
Foreign Key(Iono) references ORDER1(Ono) on delete cascade,
Foreign Key(Oino) references ITEM(Ino) on delete cascade
);

Create table WAREHOUSE(
Wno INT,
Wcity VARCHAR(50),
Primary Key(Wno)
);

Create table SHIPMENT(
Sono INT,
Swno INT,
Shipdate DATE,
Primary key(Sono,Swno),
Foreign Key(Sono) references ORDER1(Ono) on delete cascade,
Foreign Key(Swno) references WAREHOUSE(Wno) on delete cascade
);

//insert values

Insert into CUSTOMER values (1,'Archie','Bangalore');
Insert into CUSTOMER values (2,'Veronica','Bangalore');
Insert into CUSTOMER values (3,'Betty','Mysore');
Insert into CUSTOMER values (4,'Jughead','Bangalore');
Insert into CUSTOMER values (5,'Cheryl','Mysore');
Insert into CUSTOMER values (6,'Jason','Mangalore');

Insert into ORDER1 values (1, '01-Jun-2016',1,1000);
Insert into ORDER1 values (2, '03-Jun-2016',1,2000);
Insert into ORDER1 values (3, '03-Jun-2016',3,3000);
Insert into ORDER1 values (4, '05-Jun-2016',4,4000);
Insert into ORDER1 values (5, '06-Jun-2017',5,5000);

Insert into ITEM values(1,100);
Insert into ITEM values(2,200);
Insert into ITEM values(3,300);
Insert into ITEM values(4,400);
Insert into ITEM values(10,500);

Insert into ORDER_ITEM values(1,1,2);
Insert into ORDER_ITEM values(1,2,1);
Insert into ORDER_ITEM values(2,3,2);
Insert into ORDER_ITEM values(3,4,3);
Insert into ORDER_ITEM values(4,3,1);
Insert into ORDER_ITEM values(5,2,1);

Insert into WAREHOUSE values(1,'Bangalore');
Insert into WAREHOUSE values(2,'Bangalore');
Insert into WAREHOUSE values(3,'Mysore');
Insert into WAREHOUSE values(4,'Mangalore');
Insert into WAREHOUSE values(5,'Mangalore');

Insert into SHIPMENT values(1,1,'03-Jun-2016');
Insert into SHIPMENT values(2,1,'03-Jun-2016');
Insert into SHIPMENT values(3,2,'05-Jun-2016');
Insert into SHIPMENT values(4,3,'05-Jun-2016');
Insert into SHIPMENT values(5,4,'06-Jun-2016');

//List name of customer, no. of orders placed by each customer residing in Bangalore city.

Select Cname,count(Ono)
from CUSTOMER,ORDER1
where Ocno=Cno and Ccity=’Bangalore’
group by Cname;

//List the names of customers who have ordered at least 10 items.

Select Cname,count(*)
from CUSTOMER,ORDER1
where Ocno=Cno group by Cname having count(*)>=10;

//List the customer names who have not ordered for item no. 10

Select distinct(Cname)
from ORDER1,CUSTOMER,ORDER_ITEM
where  Ono=Iono and Ocno=Cno and Iono!=10;

//10B)Consider the following Movie table with the following attributes -
```

Actor_name, Actor_id, Actor_birthdate, Director_name, Director_id, Director_birthdate, film_title, year of production, type (thriller, comedy, etc.)

Create 10 collections with data relevant to the following questions. Write and execute MongoDB queries:

i. List all the movies acted by John and Elly in the year 2012.
ii. List only the name and type of the movie where Ram has acted, sorted by movie names.

```
db.createCollection("movie")
{ "ok" : 1 }
db.movie.insert([{act_n:"ram",act_id:13,act_bdate:"2/3/1997",dir_n:"williams",dir_id:101,dir_bdate:"12/9/1987",film:"battleship",year:2015,type:"thriller"}])

db.movie.insert([{act_n:"john",act_id:11,act_bdate:"1/2/1998",dir_n:"ram",dir_id:100,dir_bdate:"2/3/1997",film:"john wick",year:2012,type:"killer"}])
db.movie.insert([{act_n:"elly",act_id:12,act_bdate:"4/12/1998",dir_n:"ram",dir_id:100,dir_bdate:"2
/3/1997",film:"aquaman",year:2012,type:"action"}])

db.movie.insert([{act_n:"ram",act_id:13,act_bdate:"2/3/1997",dir_n:"thomas",dir_id:103,dir_bdate:
"12/3/1999",film:"xxx",year:2018,type:"action"}])

db.movie.insert([{act_n:"john",act_id:11,act_bdate:"1/2/1998",dir_n:"ram",dir_id:100,dir_bdate:"2/
3/1997",film:"mr.bean",year:2018,type:"comedy"}])

//List all the movies acted by John and Elly in the year 2012.
db.movie.find({$and : [{act_n:{$in : ["john","elly"]}},{year:2012}]},{film:1}).pretty()

{ "_id" : ObjectId("5c273e247fd87cec5944fabf"), "film" : "john wick" }
{ "_id" : ObjectId("5c273e2e7fd87cec5944fac0"), "film" : "aquaman" }

//ii.     List only the name and type of the movie where Ram has acted sorted by movie names.
db.movie.find({act_n:"ram"},{film:1,type:1}).sort({film:1}).pretty()

{ "_id" : ObjectId("5c273ddb7fd87cec5944fabd"), "film" : "battleship", "type" : "thriller" }
{ "_id" : ObjectId("5c273ef97fd87cec5944fac1"), "film" : "xxx", "type" : "action" }
```