MONGO

Import the given data (mongofile1.json) into a collection named “City_Details”
2. List all the states with total population greater than 5 millions
db.city_details.find({ "pop": { $gt: 500000 } }, { "_id": 0, "state": 1, "population": 1 })
3. list the smallest and largest cities by population for each state
db.city_details.aggregate([{
$group: {_id: "$state",smallestCity: { $min: { city: "$city", pop: "$population" } },largestCity: {
$max: { city: "$city", pop: "$population" } }}},
{
 $project: {
_id: 0,
state: "$_id",
smallestCity: 1,
largestCity: 1
}
}])

4. Update the population as 9500 of the city “MONSON” present in “MA” state
db.city_details.updateMany({ state: "MA", city: "MONSOON" }, { $set:{ pop: "9500"}})

5. Remove the details of the states with population less than 1000.

db.city_details.deleteMany({ "pop": { $lt: 1000 } })

Neo

Import the given date (graphfile1.json) into the database “sportspersons”
2. Using Cypher Query Language, Write the query list the name of the players whose height is >=3 and age is >=26

- MATCH (p:PLAYER)

WHERE p.height >= 3 AND p.age >= 26
RETURN p.name

3. Using Cypher Query Language, Write the query list the name of the players who are playing for any teams with a salary >=35000000.

MATCH (p:PLAYER)-[plays:PLAYS_FOR]->(t:TEAM)

```
WHERE plays.salary >= 35000000
RETURN p.name
```

4. Using Cypher Query Language, Write the query to update the salary of the players who are playing for “Dallas Mavericks” to 9000000.

```
MATCH (p:PLAYER)-[plays:PLAYS_FOR]->(t:TEAM {name: "Dallas Mavericks"})

SET plays.salary = 9000000

RETURN p.name, t.name, plays.salary
```

5. Using Cypher Query Language, Write the query to list the statistics of player “Kristaps Porzingis” when he played against the team “Philadelphia 76ers”.

pl/sql

```
create table E_Bill (S_No number(10), C_Name varchar(10), P_Reading number(10),
N_Reading number(10), C_Charges number(10))
create table Payment_Status (S_No number(10), P_Status varchar(6));

insert into E_Bill(S_No,C_Name,P_Reading) values (101,'senthil',200);
insert into E_Bill(S_No,C_Name,P_Reading) values (102,'nathan',400);
insert into E_Bill(S_No,C_Name,P_Reading) values (103,'kavin',800);
insert into E_Bill(S_No,C_Name,P_Reading) values (104,'mithun',900);
insert into E_Bill(S_No,C_Name,P_Reading) values (105,'vikas',1000);
insert into E_Bill(S_No,C_Name,P_Reading) values (106,'shan',50);

insert into Payment_Status(S_No) values (101);
insert into Payment_Status(S_No) values (102);
insert into Payment_Status(S_No) values (103);
insert into Payment_Status(S_No) values (104);
insert into Payment_Status(S_No) values (105);
insert into Payment_Status(S_No) values (106);

select * from E_Bill;
select * from Payment_Status;
```

```
CREATE OR REPLACE PROCEDURE proc1(ser_no NUMBER) AS
  new_read E_Bill.N_Reading%TYPE;
  prev_read E_Bill.P_Reading%TYPE;
BEGIN

  SELECT P_Reading INTO prev_read FROM E_Bill WHERE S_no = ser_no;


  IF prev_read > 200 THEN
    new_read := 200;
  ELSE
    new_read := 100;
  END IF;


  UPDATE E_Bill SET N_Reading = new_read WHERE S_no = ser_no;

  COMMIT;
END;
/
```