TestDome

태그

```
function countdown(seconds){
  var delay = 0;
  while (seconds >= 0){
    (function (){
      var time = seconds;
      setTimeout( function() {
        console.log(time);
        }, delay * 1000);
    delaty++;
    }());
    seconds--;
  }
}
```

```
TABLE employee
  id INTIGER NOT NULL PRIMARY KEY
  mgrId INTEGER
  name VARCHAR(30) NOT NULL
  FOREIGN KEY(mgrId) REFERENCES employees(id)
```

```
-- Suggested testing environment:
-- For MS SQL:
-- https://sqliteonline.com/ with language set as MS SQL
-- For MySQL:
-- https://www.db-fiddle.com/ with MySQL version set to 8
-- For SQLite:
-- http://sqlite.online/
-- Put the following without '--' at the top to enable foreign key support in SQLite.
-- PRAGMA foreign_keys = ON;

-- Example case create statement:
CREATE TABLE employees (
  id INTEGER NOT NULL PRIMARY KEY,
  mgrId INTEGER,
  name VARCHAR(30) NOT NULL,
  FOREIGN KEY (mgrId) REFERENCES employees(id)
);

INSERT INTO employees(id, mgrId, name) VALUES(1, NULL, 'Joey');
INSERT INTO employees(id, mgrId, name) VALUES(2, 1, 'Ross');
INSERT INTO employees(id, mgrId, name) VALUES(3, 1, 'Chandler');

-- Expected output (in any order):
```

```
-- name        manager
-- -----------------
-- Joey
-- Ross        Joey
-- Chandler    Joey

-- Explanation:
-- In this example.
-- Since managers are considered to be employees as well, Joey, Ross and Chandler are
 all employees.
-- Joey is manager for Ross and Chandler.
```

```
TABLE movies
  id INTEGER NOT NULL PRIMARY KEY
  name VARCHAR(30) NOT NULL

TABLE visitors
  id INTEGER NOT NULL PRIMARY KEY
  name VARCHAR(30) NOT NULL

TABLE movie_visitors
  movieId INTEGER NOT NULL REFERENCES movies(id)
  visitorId INTEGER NOT NULL REFERENCES visitors(id)
  PRIMARY KEY (movieId,visitorId)

  all qureis that return movies having at least the average number of visitors


select id, count(*) from movies join movies_visitors on moveId = id Group By id Having
COUNT(*)>=((SELECT COUNT(*) FROM movieS_visitors)*1.0/(SELECT COUNT(*) FROM movies));



select id, count(*) from movies join movies_visitors on moveId = id Group By id WHERE
 COUNT(*)>=((SELECT COUNT(*) FROM movieS_visitors)*1.0/(SELECT COUNT(*) FROM movies));

select id, count(*) from movies leeft join movies_visitors on movieId = id GROUP BY HA
VING COUNT(*)>=AVG(COUNT(*));

select movieid, count(*) from movies_visitors GROUP BY movieId HAVING COUNt(*)>+((SELE
CT COUNT(*) FROM movies_visitors)*1.0/(SELECT COUNT(*) FROM movies));
```

```
-- Suggested testing environment:
-- For MS SQL:
-- https://sqliteonline.com/ with language set as MS SQL
-- For MySQL:
-- https://www.db-fiddle.com/ with MySQL version set to 8
-- For SQLite:
-- http://sqlite.online/

-- Example case create statement:
CREATE TABLE roads (
  name VARCHAR(20) PRIMARY KEY NOT NULL,
  length INTEGER NOT NULL
);

INSERT INTO roads(name, length) VALUES('A417', 120);
INSERT INTO roads(name, length) VALUES('A40', 532);
INSERT INTO roads(name, length) VALUES('B41', 27);
INSERT INTO roads(name, length) VALUES('M25', 480);
INSERT INTO roads(name, length) VALUES('M1', 982);

-- Insert answer here

SELECT * FROM longRoads;

-- Expected output (in any order):
-- name    length
-- ----------------------------
-- A40     532
-- M25     480
-- M1      982

TABLE road
  name VARCHAR(2) NOT NULL PRIMARY KEy
  length INTEGER NOT NULL

create a view, named longRoads that selects the roads'name and length, which have a gr
eater than or equal to average length.
```