

Michael Yates, m jy5xy
Justin Logan, jhl3mn
Assignment 3

1.1 (5 points) Find the names of all managers who have a salary between 2000 and 3000 (inclusive)

SELECT `ename` FROM `HW_emp` WHERE `job`='Manager' and sal >= 2000 and sal <= 3000

+ Options

ename

Clark

1.2 (5 points) Find the names of all salesmen who have an income (= salary plus commission) above 2000

SELECT distinct `ename` FROM `HW_emp` WHERE `job`='Salesman' and (sal+comm) > 2000

+ Options

ename

Allen

Ward

Martin

1.3 (5 points) Find the names of all employees who work in New York

SELECT distinct `ename` FROM `HW_emp` natural join HW_dept WHERE loc = 'New York'

+ Options

ename

Clark

King

Miller

1.4 (5 points) Find the average salary by town

SELECT loc, avg(sal) FROM `HW_emp` natural join HW_dept group by loc

+ Options

loc

avg(sal)

Chicago 1966.6667

Dallas 2615.0000

New York 3683.3333

1.5 (5 points) Find the names of all employees who earn more than their manager(s)
SELECT distinct ename FROM `HW_emp` as t1 where t1.sal > (select t2.sal from `HW_emp` as t2 where t1.mgr = t2.empno)

+ Options

ename

Scott

Ford

SQL to create tables and insert data to test queries:

```
-- Product (maker, model, type)
CREATE TABLE IF NOT EXISTS Product ( -- [Product Table]
    maker VARCHAR(10) NOT NULL,
    model VARCHAR(10) NOT NULL,
    type1 VARCHAR(10) NOT NULL,
    PRIMARY KEY (maker, model)
);

-- PC (model, speed, ram, hd, price)
CREATE TABLE IF NOT EXISTS PC (
    model VARCHAR(10) NOT NULL PRIMARY KEY,
    speed DECIMAL NOT NULL,
    ram VARCHAR(10) NOT NULL,
    hd VARCHAR(10) NOT NULL,
    price INT NOT NULL
);

-- Laptop (model, speed, ram, hd, screen, price)
CREATE TABLE IF NOT EXISTS Laptop (
    model VARCHAR(10) NOT NULL PRIMARY KEY,
    speed DECIMAL NOT NULL,
    ram VARCHAR(10) NOT NULL,
    hd INT NOT NULL,
    screen VARCHAR(10) NOT NULL,
    price INT NOT NULL
);

-- Printer (model, color, type, price)
CREATE TABLE IF NOT EXISTS Printer (
```

```

    model VARCHAR(10) NOT NULL PRIMARY KEY,
    color VARCHAR(10) NOT NULL,
    type1 VARCHAR(10) NOT NULL,
    price INT NOT NULL
);

INSERT INTO Product VALUES ('maker1', 'model11', 'PC');
INSERT INTO Product VALUES ('maker1', 'model12', 'Laptop');
INSERT INTO Product VALUES ('maker1', 'model13', 'Printer');
INSERT INTO Product VALUES ('maker2', 'model14', 'PC');
INSERT INTO Product VALUES ('maker2', 'model15', 'Laptop');
INSERT INTO Product VALUES ('maker2', 'model16', 'Printer');
INSERT INTO Product VALUES ('maker2', 'model17', 'PC');
INSERT INTO Product VALUES ('maker2', 'model18', 'PC');
INSERT INTO Product VALUES ('maker2', 'model19', 'Laptop');
INSERT INTO Product VALUES ('maker1', 'model110', 'Printer');

INSERT INTO PC VALUES ('model11', 1.0, 'ram1', 'hd1', 500);
INSERT INTO PC VALUES ('maker4', 2.0, 'ram2', 'hd1', 750);
INSERT INTO PC VALUES ('model17', 3.0, 'ram3', 'hd1', 1000);
INSERT INTO PC VALUES ('maker8', 4.0, 'ram4', 'hd1', 550);

INSERT INTO Laptop VALUES ('model12', 0, 'ram1', 480, 'screen1', 500);
INSERT INTO Laptop VALUES ('model15', 2.0, 'ram1', 720, 'screen1', 750);
INSERT INTO Laptop VALUES ('model19', 3.0, 'ram1', 1080, 'screen1', 1000);

INSERT INTO Printer VALUES ('model13', 'gray', 'type11', 750);
INSERT INTO Printer VALUES ('model16', 'white', 'type12', 1000);
INSERT INTO Printer VALUES ('model110', 'black', 'type13', 350);

```

Product (maker, model, type) -- maker is foreign key to Manufacturer(maker)

PC (model, speed, ram, hd, price)

Laptop (model, speed, ram, hd, screen, price)

Printer (model, color, type, price)

2.1 (15 points) Find the makers of PC's with a speed of at least 3.0
SELECT maker FROM `PC` natural join Product WHERE speed >= 3.0







+ Options

maker

maker2




2.2 (15 points) Find the **printers** with the highest price
– I'm interpreting this question as find the two highest priced printers, because printers was plural in the question.

SELECT * FROM `Printer` ORDER BY price desc limit 2

<div><div>←</div><div>T</div><div>→</div></div>				model	color	type1	price	▼ 1		
<input type="checkbox"/>		Edit		Copy		Delete	model6	white	type12	1000
<input type="checkbox"/>		Edit		Copy		Delete	model3	gray	type11	750

2.3 (15 points) Find the laptops whose speed is slower than that of any PC
SELECT L1.model, L1.speed FROM `Laptop` as L1 where L1.speed < (Select min(pc1.speed)
from PC as pc1)

+ Options

<div>↔ T ↔</div>				model	speed
<input type="checkbox"/>	 Edit	 Copy	 Delete	model2	0

Actor (pid, fname, lname, gender)

Movie (mid, name, year, revenue) -- name represent the movie name

Directors (did, fname, lname)

Casts (pid, mid, role) -- pid is FK to Actor(pid), mid is FK to Movie(mid)

-- an actor can play multiple roles in one movie,

thus (pid, mid, role) is not a PK

Movie_directors (did, mid) -- pid is FK to Actor(pid), mid is FK to Movie(mid)

did is FK to Directors(did)

```
-- Actor (pid, fname, lname, gender)
CREATE TABLE IF NOT EXISTS Actor (
    pid INT NOT NULL PRIMARY KEY,
    fname VARCHAR(15) NOT NULL, -- first name
    lname VARCHAR(15) NOT NULL, -- last name
    gender VARCHAR(10) NOT NULL -- last name
);

INSERT INTO Actor VALUES (0, 'Mike', 'Yates', 'male');
INSERT INTO Actor VALUES (1, 'Justin', 'Logan', 'male');
INSERT INTO Actor VALUES (2, 'Kunal', 'Cha', 'male');
INSERT INTO Actor VALUES (3, 'Gabri', 'Hym', 'male');
INSERT INTO Actor VALUES (4, 'Lexi', 'Pass', 'female');
INSERT INTO Actor VALUES (5, 'Lizzy', 'Lynch', 'female');
INSERT INTO Actor VALUES (6, 'Jennifer', 'Lawrence', 'female');
INSERT INTO Actor VALUES (7, 'Daniel', 'Tosh', 'male');
INSERT INTO Actor VALUES (8, 'Emma', 'Watson', 'female');

-- Movie (mid, name year, revenue)
CREATE TABLE IF NOT EXISTS Movie (
    mid INT NOT NULL PRIMARY KEY,
    m_name VARCHAR(15) NOT NULL, -- name represent the movie name
    m_year int NOT NULL, -- year is a taken keyword
    revenue int NOT NULL -- in millions
);

INSERT INTO Movie VALUES(0, "The Wolf of Wallstreet", 2013, 392);
INSERT INTO Movie VALUES(1, "movie 2", 2013, 392);
INSERT INTO Movie VALUES(2, "movie 3", 2014, 392);
INSERT INTO Movie VALUES(3, "movie 4", 2013, 400);
INSERT INTO Movie VALUES(4, "movie 5", 2017, 200);
```

```

INSERT INTO Movie VALUES(5, "movie 6", 2016, 150);
INSERT INTO Movie VALUES(6, "movie 7", 2014, 225);
INSERT INTO Movie VALUES(7, "movie 8", 2020, 330);
INSERT INTO Movie VALUES(8, "movie 9", 2018, 275);
INSERT INTO Movie VALUES(9, "movie 10", 2014, 400);
INSERT INTO Movie VALUES(10, "No Cast", 2013, 100);

-- Directors (did, fname, lname)
CREATE TABLE IF NOT EXISTS Directors (
    did INT NOT NULL PRIMARY KEY,
    fname VARCHAR(15) NOT NULL, -- first name
    lname VARCHAR(15) NOT NULL -- last name
);

-- Martin Scorsese
INSERT INTO Directors VALUES(0, "Martin", "Scorsese");
INSERT INTO Directors VALUES(1, "name1", "name2");
INSERT INTO Directors VALUES(2, "name3", "name4");
INSERT INTO Directors VALUES(3, "name5", "name6");
INSERT INTO Directors VALUES(4, "name7", "name8");

-- Casts (pid, mid, role)
CREATE TABLE IF NOT EXISTS Casts (
    pid INT NOT NULL, -- foreign key to Actor
    mid INT NOT NULL, -- foreign key to Movie
    m_role VARCHAR(15) NOT NULL -- 'role' is a taken keyword
);

INSERT INTO Casts VALUES(0, 0, "actor"); -- mike yates was in wolf of
wallstreet
INSERT INTO Casts VALUES(1, 0, "actor");
INSERT INTO Casts VALUES(2, 1, "actor");
INSERT INTO Casts VALUES(3, 2, "actor");
INSERT INTO Casts VALUES(4, 1, "actor");
INSERT INTO Casts VALUES(5, 3, "actor");
INSERT INTO Casts VALUES(6, 5, "actor");
INSERT INTO Casts VALUES(1, 6, "actor");
INSERT INTO Casts VALUES(2, 4, "actor");
INSERT INTO Casts VALUES(3, 4, "actor");
INSERT INTO Casts VALUES(4, 7, "actor");
INSERT INTO Casts VALUES(7, 8, "actor");

```

```

INSERT INTO Casts VALUES(8, 9, "actor");

CREATE TABLE IF NOT EXISTS Movie_directors (
    did INT NOT NULL,
    mid INT NOT NULL,
    PRIMARY KEY (did, mid)
);

-- dir 0 to 4 , movies 0 to 9
INSERT INTO Movie_directors VALUES(0, 0);
INSERT INTO Movie_directors VALUES(1, 1);
INSERT INTO Movie_directors VALUES(2, 4);
INSERT INTO Movie_directors VALUES(3, 2);
INSERT INTO Movie_directors VALUES(4, 3);
INSERT INTO Movie_directors VALUES(4, 5);
INSERT INTO Movie_directors VALUES(4, 6);
INSERT INTO Movie_directors VALUES(3, 7);
INSERT INTO Movie_directors VALUES(4, 8);
INSERT INTO Movie_directors VALUES(4, 9);

```

3.1 (15 pts.) List all directors who directed at least 200 movies, in descending order of the number of movies they directed. Display the directors' first and last names and the number of movies each of them directed.

Note: 'name', 'role', and 'year' are keywords in SQL, so I adjusted the names to 'm_name', 'm_role', and 'm_year', respectively.

```

Select fname, lname, count(Movie_directors.did) as total_movies_directed
from Movie_directors inner join Directors on Movie_directors.did = Directors.did
group by fname, lname
having total_movies_directed >= 200
order by total_movies_directed desc

```

For my data, when running the command but changing at least 200 movies to at least 2 movies produces result:

+ Options		
fname	lname	total_movies_directed ▾ 1
name7	name8	5
name5	name6	2

3.2 (15 pts.) For each year, count the number of movies in that year that had only female actors. Display the year and the number of movies in that year.

[hint] consider a universal quantifier: a movie without any actors is also a movie with only female actors (since there are no male actors in that movie)

Note: 'name', 'role', and 'year' are keywords in SQL, so I adjusted the names to 'm_name', 'm_role', and 'm_year', respectively.

```
SELECT M1.m_year, count(M1.m_year) as total
FROM `Movie` as M1
where M1.mid not in (SELECT mid FROM `Movie` natural join Actor natural join Casts where
gender = 'male')
group by M1.m_year
```

+ Options

m_year	total
2013	2
2014	1
2016	1
2020	1