CSE 344 Section 2

0. Joins Examples

Given tables created with these commands:

```
CREATE TABLE A (a int);
CREATE TABLE B (b int);
INSERT INTO A VALUES (1), (2), (3), (4);
INSERT INTO B VALUES (3), (4), (5), (6);
```

What's the output for each of the following:

```
SELECT * FROM A INNER JOIN B ON A.a=B.b;

SELECT * FROM A LEFT OUTER JOIN B ON A.a=B.b;

SELECT * FROM A RIGHT OUTER JOIN B ON A.a=B.b;

SELECT * FROM A FULL OUTER JOIN B ON A.a=B.b;
```

Inner:
a|b
3|3
4|4

Left Outer:
a|b
3|3
4|4
1|
2|

Right Outer:
a|b
|5
|6
3|3
4|4

Full Outer:
a|b
|5
|6
1|
2|
3|3
4|4

Sidenote: sqlite3 supports neither RIGHT OUTER nor FULL OUTER.

Right outer can be implemented with
SELECT * FROM B LEFT OUTER JOIN A ON A.a=B.b;

Full outer can be implemented with
(SELECT * FROM A LEFT OUTER JOIN B ON A.a=B.b)
UNION
(SELECT * FROM B LEFT OUTER JOIN A ON A>A=B.b);

We haven't talked about UNION really, but it's the same as the set operation

1. SQL Practice

```
CREATE TABLE Movies (
    id int,
    name varchar(30),
    budget int,
    gross int,
    rating int,
    year int,
    PRIMARY KEY (id)
);
```

```
CREATE TABLE Actors (
    id int,
    name varchar(30),
    age int,
    PRIMARY KEY (id)
);

CREATE TABLE ActsIn (
    mid int,
    aid int,
    FOREIGN KEY (mid) REFERENCES Movies (id),
    FOREIGN KEY (aid) REFERENCES Actors (id)
);
```

What is the total budget of all movies released in the year 2017?

```
SELECT sum(budget)
FROM Movies
WHERE year=2017;
```

What is the number of movies, and the average rating of all movies that the actor "abcd" has appeared in?

```
SELECT count(*), avg(rating)
FROM Movies as M, ActsIn as AI, Actors as A
WHERE M.id=AI.mid AND A.id=AI.aid AND A.name="abcd";
```

What is the minimum age of an actor who has appeared in a movie where the gross of the movie has been over $1,000,000,000?

```
SELECT min(age)
FROM Movies as M, ActsIn as AI, Actors as A
WHERE M.id=AI.mid AND AI.aid=A.id AND gross>1000000000;
```

---

2. Self Join

Consider the following over simplified Employee table

```
CREATE TABLE Employees (
    id int,
    bossOf int
);
```

```
SELECT DISTINCT e2.id
FROM Employees as e1
INNER JOIN Employees as e2
ON e2.bossOf=e1.id;
```

We want distinct because we only want every boss to show up once in our output.

Some people brought up the possible solution

```
SELECT id
FROM Employees
WHERE bossOf IS NOT NULL;
```

This works assuming every bossof is valid.
Suppose we had an employee id=0, bossOf=9999.
If 9999 isn't a valid employee id, then id=0 is not necessarily a boss.
We apologize for the confusing question.

Suppose all employees have an id which is not null. How would we find the id of all employees who are the boss of at least one other employee?

What do we select? (select * vs select table_alias.col_name)

We want select table_alias.col_name (e2.id in this case) because otherwise our output would contain a row for each pair of id, bossOf. This means we would get each boss to employee pairing whereas we only want the boss ids.

~~Consider the case with employees (0, null), (1, null), (2, 1), (2, 2). How many times does 2 appear in the output?~~

This question has a typo, and is malformed.

---

3. Group By and Order By

```
GROUP BY [colname]
```

This was here as a general guideline in case your TAs had some extra time to cover these additional statements for the homework. They will be covered in lecture on Friday 10/06.

```
ORDER BY [colname]
```