CSE 414 Section 2

1. Joins Examples

a|b

313

4 | 4

a|b

1|3

1 | 4

1|5

1 | 6

2|3

2 | 4

2|5

2|6

3|3

3 | 4

3|5

3|6

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 INNER JOIN B;
```

```
4 | 3
4 | 4
4 | 5
4 | 6
SELECT * FROM A LEFT OUTER JOIN B ON
A.a=B.b;
a|b
1|
2|
3|3
4 | 4
SELECT * FROM A RIGHT OUTER JOIN B ON
A.a=B.b;
a|b
3|3
4 | 4
|5
16
SELECT * FROM A FULL OUTER JOIN B ON
A.a=B.b;
a|b
1|
2|
3|3
4 | 4
|5
16
SELECT a AS c FROM A UNION SELECT b AS c FROM B;
С
1
```

```
2
3
4
5
SELECT a AS c FROM A UNION ALL SELECT b AS c FROM B;
1
2
3
4
3
4
5
6
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
Union selects distinct values
Union all select even duplicate values
```

2. 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 number of movies, and the average rating of all movie that the actor "Patrick Stewart" has appeared in?

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

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 M.gross > 1000000000;
```

What is the total budget of all movies released in year 2017, where the oldest actor is less than 30?

```
SELECT sum (M.budget)

FROM Movies as M, ActsIn as AI, Actors as A
```

```
WHERE M.id = AI.mid AND AI.aid = A.id AND M.year =
2017
GROUP BY
M.id
HAVING max(A.age) <
30;</pre>
```

3. Self Join

Consider the following over simplified Employee table:

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

Suppose all employees have an id which is not null. How would we find all distinct pairs of employees with the same boss?

```
SELECT E1.bossOf,
E2.bossOf

FROM Employees AS E1, Employees AS E2

WHERE E1.id = E2.id AND E1.bossOf > E2.bossOf;
```

Sidenote: The predicate "E1.bossOf > E2.bossOf" could also be written as "E1.bossOf < E2.bossOf". We cannot use plain inequality as the predicate condition because this would lead to duplicate pairs.