CSE 414 Section 2 Worksheet Solutions

1. SQL to Relational Algebra. Write an expression in the form of a logical query plan (i.e., draw a tree) that is equivalent to each of the SQL query below:

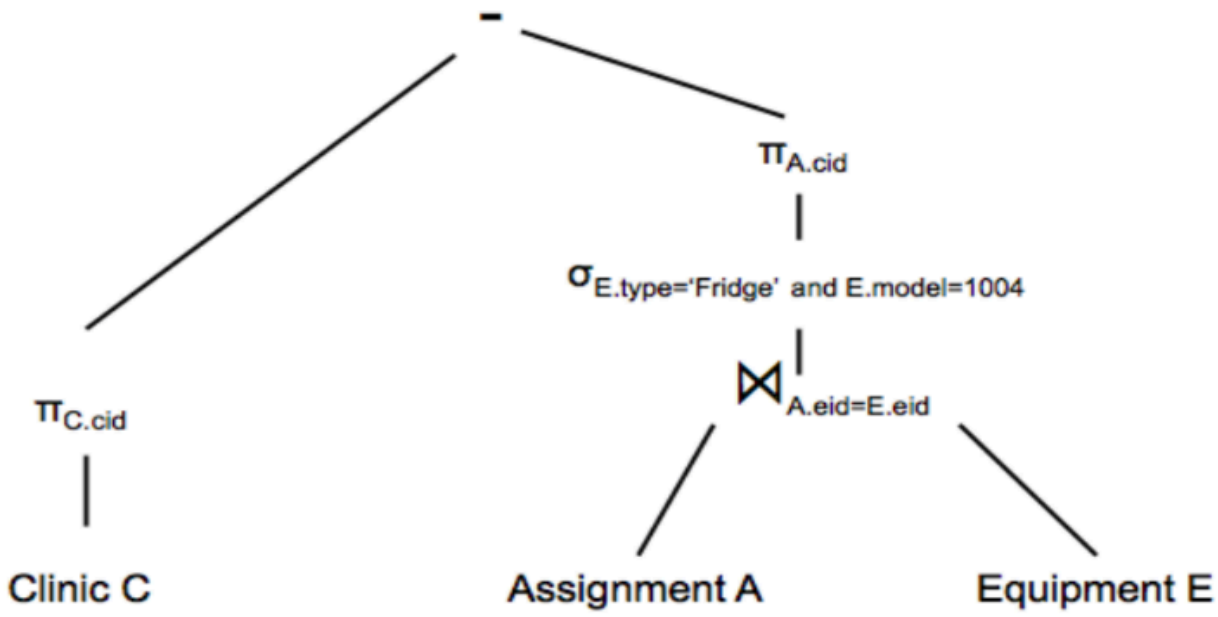
A. Select all clinics that do not have an assignment to a Model 1004 'Fridge'.

Schema: **Clinic(<u>cid</u>, name, street, state)** // cid is the Clinic ID

Equipment(<u>eid</u>, type, model) // eid is the Equipment ID

Assignment(cid, eid)

```
SELECT C.cid
  FROM Clinic C
 WHERE NOT EXISTS (SELECT * FROM Assignment A, Equipment E
                   WHERE C.cid = A.cid AND A.eid = E.eid
                         AND E.type = 'Fridge' AND E.model = 1004);
```



The selection could be pushed down into the join above E, as a query optimization.

B. Select the greatest difference in price between items exchanged between the same two people within the same category, for each category among all categories that have more than 5 such exchanges.

Schema: **Item(<u>oid</u>, category, price)**

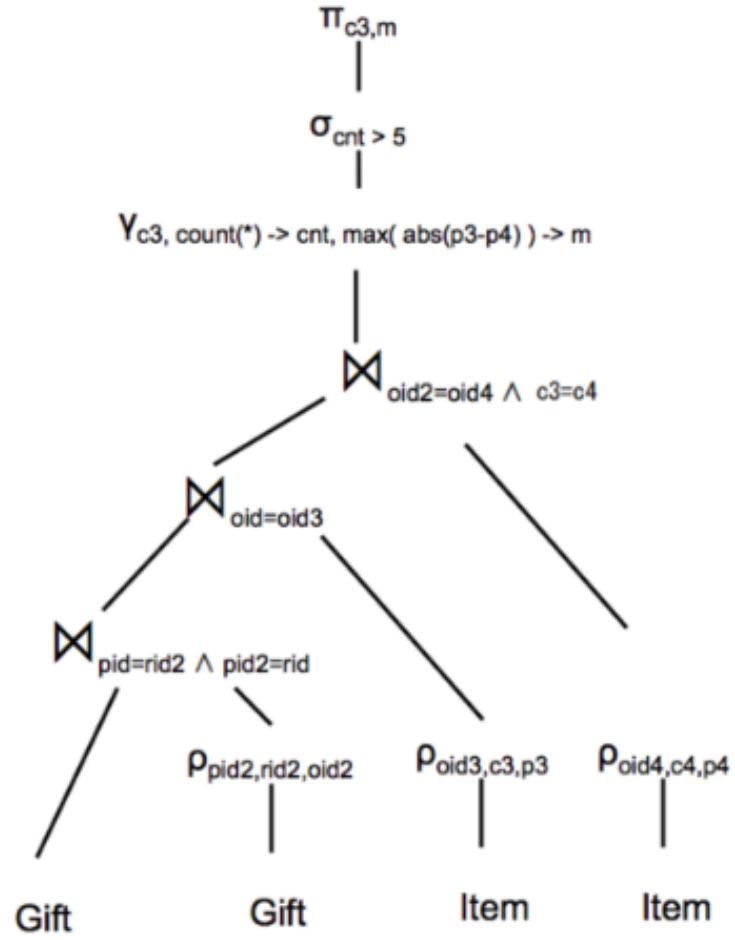
Gift(<u>pid</u>, rid, oid)

Gift.pid: presenter ID

Gift.rid: recipient ID

Gift.oid is a foreign key to Item.oid

```
SELECT O1.category, max(abs(O1.price - O2.price))
  FROM Gift G1, Gift G2, Item O1, Item O2
 WHERE G1.pid = G2.rid AND G2.pid = G1.rid
       AND O1.oid = G1.oid AND O2.oid = G2.oid
       AND O1.category = O2.category GROUP BY O1.category
HAVING count(*) > 5;
```



Solutions that use different join orders are possible.

2. 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:

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

```
a | b
3 | 3
4 | 4
```

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

```
a | b
3 | 3
4 | 4
1 |
2 |
```

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

```
a | b
  | 5
  | 6
3 | 3
4 | 4
```

(d) SELECT * FROM A FULL OUTER JOIN B ON A.a=B.b;

```
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);

3. 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));

(a) What is the number of movies, and the average rating of all movie that the actor ”Patrick Stewart” 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 = “Patric Stewart”;

(b) 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 > 1,000,000,000;

(c) 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 mac(A.age) < 30;

4. 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.id, E2.id FROM Employee AS E1, Employee AS E2
WHERE E1.bossOf = E2.bossOf AND E1.id > E2.id;
```

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