Section 4 Worksheet

Part 1: Interpreting SQL and Relational Data

For each SQL query, find

1) what the SQL statement is querying for (a short description) and
2) an equivalent relational algebra (RA) expression/tree

A. (Midterm 12AU)

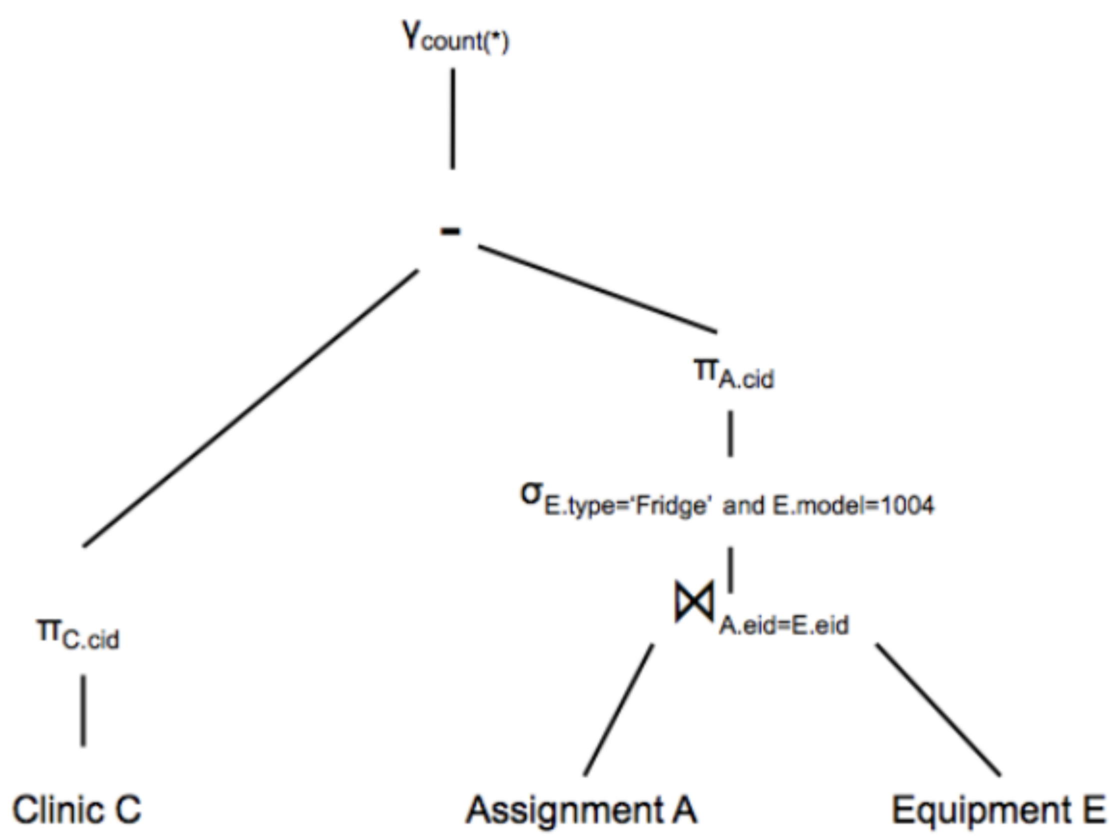
Clinic(cid, name, street, state)
Equipment(eid, type, model)
Assignment(cid, eid)

Finds the count of clinics that do not have a fridge (of model 1004) assigned to it.

1)

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

2)



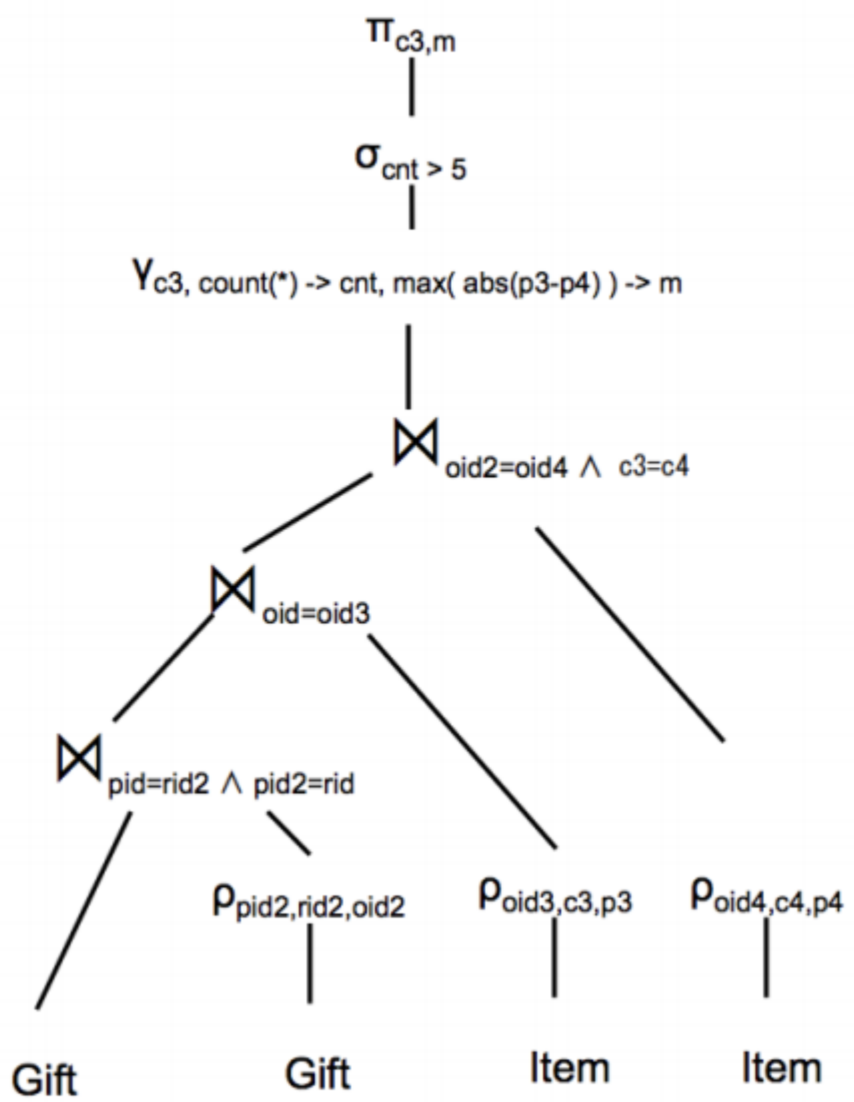
B. (Midterm 15AU)

`Item(oid, category, price)`
`Gift(pid, rid, oid) -- pid gifts oid to rid`

```
SELECT O1.category, max(abs(O1.price - O2.price))
  FROM Gift AS G1, Gift AS G2, Item AS O1, Item AS 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;
```

1) Finds item categories that have been mutually gifted over 5 times and the corresponding maximum price difference between mutually exchanged items (of said category).

2)



Section 5 Worksheet

Part 1. Datalog Practice

Consider a graph of colored vertices and undirected edges where the vertices can be red, green, blue. In particular, you have the relations

```
Vertex(x, color)
Edge(x, y)
```

The Edge relation is symmetric in that if (x, y) is in Edge, then (y, x) is in Edge.
Your goal is to write a datalog program to answer each of the following questions.

1. Find all green vertices.

```
GreenV(x) :- Vertex(x, 'green')
```

2. Find all pairs of blue vertices connected by one edge.

```
BluePairs(x, y) :- Vertex(x, 'blue'), Vertex(y , 'blue'), Edge(x, y)
```

3. Find all triangles where all the vertices are the same color. Output the three vertices and their color.

```
Triangle(x, y, z, c) :- Vertex(x, c), Vertex(y, c), Vertex(z, c),
                        Edge(x, y), Edge(y, z), Edge(z, x)
```

4. Find all vertices that don't have any neighbors.

WRONG ANSWER (UNSAFE)

```
LonelyV(x) :- !Edge(x, _)
```

WRONG ANSWER (UNSAFE)

```
LonelyV(x) :- Vertex(x, _), !Edge(x, _)
```

CORRECT ANSWER (SAFE)

```
OnlyX(x) :- Edge(x, _)
LonelyV(x) :- Vertex(x, _), !OnlyX(x)
```

5. Find all vertices such that they only have red neighbors.

```
BlueV(x) :- Vertex(x, _), Edge(x, y), Vertex(y, 'blue')
GreenV(x) :- Vertex(x,_), Edge(x, y), Vertex(y, 'green')
RedV(x) :- Vertex(x,_), !BlueV(x), !GreenV(x)
```

6. Find all vertices such that they only have neighbors with the same color. Return the vertex and color.

```
SameColor(x, y, a) :- Vertex(x, a), Vertex(y, a)
NotSameNeigh(x) :- Vertex(x, _), Edge(x, y), Edge(x, z), !SameColor(y, z)
OnlySameNeigh(x, a) :- Vertex(x, a), !NotSameNeigh(x)
```

OR

```
Neigh(x, y, a) :- Edge(x, y), Vertex(y, a)
DifferentNeigh(x) :- Neigh(x, y, a), Neigh(x, z, b), a != b
OnlySameNeigh(x, a) :- Vertex(x, a), !DifferentNeigh(x)
```

7. For some vertex v, find all vertexes connected to v by blue vertexes (this one requires recursion).

```
ConnectedTo(x) :- Vertex(x, 'blue'), Edge(x, 'v')
ConnectedTo(x) :- Vertex(x, 'blue'), Edge(x, y), ConnectedTo(y)
```