PART II

Consider the following database schema:

Frequents(kid, store)
Sells(store, candy)
Likes(kid, candy)

Table **Frequents** indicates what candy stores a kid likes to visit. Table **Sells** shows which candy each store sells. Table **Likes** tells which candy a kid likes.

Express the following queries. When you use relational algebra, you can either write a single expression, or write a sequential relational algebra program with a series of assignments, or draw a tree. If you think it is impossible to express a particular query in relational algebra, say so and justify your answer.

(a) (4 *pts*) Use relational algebra to find the kids who like "Hersheys" or "Mars" candy.

$H \leftarrow \Pi_{KID}(\sigma_{CANDY = \text{“HERSHEYS”}}(LIKES))$
$M \leftarrow \Pi_{KID}(\sigma_{CANDY = \text{“MARS”}}(LIKES))$
$BOTH \leftarrow H \text{ UNION } M$
$\Pi_{KID}(BOTH)$

(b) (6 *pts*) Use relational algebra to find the kids that frequent at least one candy store that sells a candy they like.

$R \leftarrow \Pi_{KID,\ STORE}(LIKES \text{ NATURAL-JOIN } SELLS)$
$Sol \leftarrow \Pi_{KID}(\sigma_{F.STORE=R.STORE \wedge F.KID = R.KID}(F \times R))$

(c) (6 *pts*) Assume each kid likes at least one candy and frequents at least one store. Use relational algebra to find the kids that frequent *only* stores that serve *some* candy that they like.

$R \leftarrow \Pi_{KID,\ STORE}\,(LIKES\ _{NATURAL\text{-}JOIN}\ SELLS)$
$S \leftarrow \Pi_{KID}\,(FREQUENTS - R)$
$Sol \leftarrow \Pi_{KID}\,(FREQUENTS)\ -\ S$

(d) (4 *pts*) Use **SQL** to list the stores (in alphabetical order) that sell more than 10 different candies.

```
Select store
From sells
Group by stores
Having count(distinct candy) > 10;
```

QUESTION 3: SQL tables. Write the SQL commands needed to create the *Dancer* and *Role* tables described on page 2. Be sure to include the correct names and types for all attributes, and any key or foreign key constraints. (You do *not* need to give SQL commands to create the other tables – just the ones asked for.)

```
CREATE TABLE Dancer (
  did       int PRIMARY KEY,
  name      varchar(20),
  birthyear int,
  country   varchar(20)
);


CREATE TABLE Role (
  did int references Dancer,
  sid int references Show,
  role varchar(20),
  company varchar(20) references Company,
  PRIMARY KEY(did, sid, role, company)
);
```

Write SQL queries to retrieve the requested information from the dance database tables described previously. The queries you write must be proper SQL that would be accepted by SQL Server or any other SQL implementation. You should not use incorrect SQL, even if sqlite might produce some sort of answer from the buggy SQL.

(a) (10 points) For every dancer who has performed the role 'Black Swan' in the show 'Swan Lake' for one or more companies, list the name of the dancer and the company name(s), sorted by dancer name. If the dancer has performed that role for more than one company, there should be one line of output for each dancer, company pair. The companies can be listed in any order.

```
SELECT distinct dancer.name, role.company
FROM role, show, dancer
WHERE role.sid = show.sid
  AND role.did = dancer.did
  AND role.role = "Black Swan"
  AND show.title = "Swan Lake"
ORDER BY dancer.name;
```

(b) (10 points) List the dancer ids (did) and names of all dancers who have danced in a show choreographed by 'Fosse' but have not danced in a show choreographed by 'Robbins'. Each did/name pair should only appear once in the output.

```
SELECT distinct dancer.did, dancer.name
FROM role, show, dancer
WHERE role.sid = show.sid
  AND role.did = dancer.did
  AND show.choreographer = "Fosse"
  AND role.did NOT IN (
                SELECT role.did
                FROM role, show
                WHERE role.sid = show.sid
                AND show.choreographer = "Robbins"
  );
```

Question 2. (cont.) (c) (10 points) List the dancer ids (did) and names of all dancers born on or before 1950 and who have danced in at least three different shows. If a dancer has danced different roles in the same show, it still only counts once in the total number of shows. Each dancer/did pair should only be listed once.

```
SELECT dancer.did, dancer.name
FROM dancer, role
WHERE dancer.birthyear <= 1950
  AND dancer.did = role.did
GROUP BY dancer.did, dancer.name
HAVING count(DISTINCT role.sid) >= 3;
```

(d) (10 points) For every dancer who has danced for one or more companies in a different country than where they were born, list the name of the dancer and the names of those companies.

```
SELECT distinct dancer.name, company.name
FROM dancer, role, company
WHERE dancer.did = role.did
  AND role.company = company.name
  AND company.country <> dancer.country;
```