Solutions to practice problems on relational calculus and SQL

Problem 1

(a) List the bars that serve a beer that Joe likes.

(i) relational calculus:

\[
\{ b : \text{bar} \mid \exists s \in \text{serves} \exists l \in \text{likes} \\
(\text{s(bar) = b(bar)} \land s(\text{beer}) = \text{l(beer)} \land \text{l(drinker) = Joe}) \}
\]

(ii) SQL:

```
select s.bar
from serves s, likes l
where s.beer = l.beer
AND l.drinker = “Joe”
```

(b) List the drinkers that frequent at least one bar that serves a beer they like.

(i) relational calculus:

\[
\{ d : \text{drinker} \mid \exists f \in \text{frequents} \exists s \in \text{serves} \exists l \in \text{likes} \\
(d(\text{drinker}) = f(\text{drinker}) \land f(\text{bar}) = s(\text{bar}) \\
\land s(\text{beer}) = l(\text{beer}) \land \text{l(drinker) = f(drinker)}) \}
\]

(ii) SQL:

```
select f.drinker
from frequents f, serves s, likes l
where f.bar = s.bar
and s.beer = l.beer
and l.drinker = f.drinker
```
(c) List the drinkers that frequent only bars that serve some beer that they like.
(Assume each drinker likes at least one beer and frequents at least one bar.)

(i) relational calculus:

\[
\{ d : \text{drinker} \mid \exists f \in \text{frequents} \ (f(\text{drinker}) = d(\text{drinker}) \land \\
\forall y \in \text{frequents} \ [y(\text{drinker}) = f(\text{drinker}) \rightarrow \\
\exists s \in \text{serves} \exists l \in \text{likes} \ (s(\text{bar}) = y(\text{bar}) \land \\
s(\text{beer}) = l(\text{beer}) \land l(\text{drinker}) = y(\text{drinker})) \} \}
\]

Existential form:

\[
\{ d : \text{drinker} \mid \exists f \in \text{frequents} \ (f(\text{drinker}) = d(\text{drinker}) \land \\
\neg \exists y \in \text{frequents} \ [y(\text{drinker}) = f(\text{drinker}) \land \\
\neg \exists s \in \text{serves} \exists l \in \text{likes} \ (s(\text{bar}) = y(\text{bar}) \land \\
s(\text{beer}) = l(\text{beer}) \land l(\text{drinker}) = y(\text{drinker})) \} \}
\]

(ii) SQL:

Using NOT EXISTS (see relational calculus query above):

```sql
select f.drinker
from frequents f
where not exists
  (select *
   from frequents y
   where y.drinker = f.drinker and not exists
      (select *
       from serves s, likes l
       where s.bar = y.bar
       and s.beer = l.beer
       and l.drinker = y.drinker))
```

Another version using NOT IN:
select drinker
from frequents where drinker not in
(select f.drinker
from frequents f
where f.bar not in
(select bar
from serves, likes
where serves.beer = likes.beer
and likes.drinker = f.drinker))

(d) List the drinkers who frequent no bar that serves a beer that they like.
This is just the complement of (b).

Problem 2
(c) List the actors cast only in movies by Berto.

(i) relational calculus:

\[ \{ a: actor \mid \exists m \in movie[a(actor) = m(actor) \land
\forall t \in movie (t(actor) = m(actor) \rightarrow \exists s \in movie (s(title) = t(title)
\land s(director) = Berto))\} \]

EXISTENTIAL form:

\[ \{ a: actor \mid \exists m \in movie[a(actor) = m(actor) \land
\neg \exists t \in movie (t(actor) = m(actor) \land \neg \exists s \in movie (s(title) = t(title)
\land s(director) = Berto))\} \]

(ii) SQL (direct translation of the above calculus query, using NOT EXISTS):

select m.actor
from movie m
where not exists
(select * from movie t
where t.actor = m.actor and not exists
(select * from movie s
where s.title = t.title and s.director = 'Berto'))
Another possibility, making the unique director assumption:

select actor 
from movie 
where actor not in 
(select actor 
from movie 
where director \neq Berto )

(b) List all pairs of distinct actors who act together in at least one movie (avoid listing both \((a,b)\) and \((b,a)\)).

(i) relational calculus:

\{ a : actor1, actor2 | \exists m1 \in movie \exists m2 \in movie ( a(actor1) = m1(actor) 
\wedge a(actor2) = m2(actor) \wedge m1(title) = m2(title) \wedge m1(actor) < m2(actor)) \}

(ii) SQL:

select m1.actor as actor1, m2.actor as actor2 
from movie m1, movie m2 
where m1.title = m2.title \text{ and } m1.actor < m2.actor

(c) List the directors such that every actor is cast in one of his/her movies.

(i) relational calculus (no unique director assumption):

\{ d : director | \exists m \in movie [d(director) = m(director) \wedge 
\forall a \in movie \exists x \in movie \exists y \in movie (x(director) = m(director) \wedge x(title) = y(title) 
\wedge y(actor) = a(actor)) ] \}

EXISTENTIAL form:

\{ d : director | \exists m \in movie [d(director) = m(director) \wedge 
\neg \exists a \in movie \neg \exists x \in movie \exists y \in movie (x(director) = m(director) \wedge x(title) = y(title) 
\wedge y(actor) = a(actor)) ] \}
(ii) SQL (direct translation of the above calculus query):

```sql
select m.director from movie m
where not exists
    (select * from movie a
     where not exists
         (select * from movie x, movie y
          where x.director = m.director and x.title = y.title and y.actor = a.actor ))
```

Another possibility:

```sql
select director
from movie
where director not in
    (select d.director
     from movie d, movie a
     where d.director not in
         (select x.director
          from movie x, movie y
          where x.title = y.title and y.actor = a.actor ))
```