1 Schema and data

We are using the IMDB database. Note that the schema contains more relations than the ones you need to work on the queries below. We have provided the relevant relations in a separate handout.

2 Queries

1. Find the name of everyone who is both a Sci-Fi director and an actor/actress.

   Solution:

   
   (SELECT name 
    FROM roles NATURAL JOIN people 
   ) INTERSECT ( 
    SELECT name 
    FROM directors NATURAL JOIN people NATURAL JOIN movies 
      NATURAL JOIN movie_genres NATURAL JOIN genres 
      WHERE label = 'Sci-Fi' 
   );

   Output:

   name

   ----------------------
   Patrik, Damien
   Pryor, Bryn
   Kucan, Joseph D.
   Kubrick, Stanley (I)
   Spielberg, Steven
   Cross, Logan
   Marquand, Richard
   Gilliam, Terry
   Carpenter, John (I)
   (9 rows)

2. Find all countries which have produced at least one movie with a rating higher than 8.5.

   Solution:
SELECT name  
FROM countries  
WHERE country_id IN (  
    SELECT country_id  
    FROM movie_countries NATURAL JOIN movies  
    WHERE rating > 8.5  
);  

Output:  
name  
--------  
Australia  
Brazil  
France  
Germany  
Italy  
Japan  
New Zealand  
Spain  
UK  
USA  
West Germany  
(11 rows)  

3. Find the highest-rated movie, and report its title, year, rating, and country. There can be ties; if so, you should report for each of them.

Solution:

SELECT title, year, rating, name  
FROM movies NATURAL JOIN movie_countries NATURAL JOIN countries  
WHERE rating = (  
    SELECT MAX(rating)  
    FROM movies  
);  

Output:  
<table>
<thead>
<tr>
<th>title</th>
<th>year</th>
<th>rating</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Shawshank Redemption</td>
<td>1994</td>
<td>9.2</td>
<td>USA</td>
</tr>
<tr>
<td>The Godfather</td>
<td>1972</td>
<td>9.2</td>
<td>USA</td>
</tr>
</tbody>
</table>
(2 rows)

4. Find the highest-rated Horror movie, and report the title, year, and rating. There can be ties; if so, you should report for each of them.

Solution:
SELECT title, year, rating
FROM (movies NATURAL JOIN movie_genres NATURAL JOIN genres) AS R1
WHERE label = 'Horror' AND NOT EXISTS (
    SELECT *
    FROM (movies NATURAL JOIN movie_genres NATURAL JOIN genres) AS R2
    WHERE R2.label = 'Horror' AND R1.rating < R2.rating
);

Alternatively,

SELECT title, year, rating
FROM movies NATURAL JOIN movie_genres NATURAL JOIN genres
WHERE label = 'Horror' AND rating >= ALL (
    SELECT rating
    FROM movies NATURAL JOIN movie_genres NATURAL JOIN genres
    WHERE R2.label = 'Horror'
);

Output:

<table>
<thead>
<tr>
<th>title</th>
<th>year</th>
<th>rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Psycho</td>
<td>1960</td>
<td>8.6</td>
</tr>
</tbody>
</table>

(1 row)

5. Find the genre which has the highest average rating. Report the name of the genre, and its average rating. There can be ties; if so, you should report for each of them.

Solution:

SELECT label, AVG(rating)
FROM genres NATURAL JOIN movie_genres NATURAL JOIN movies
GROUP BY genre_id
HAVING AVG(rating) >= ALL (
    SELECT AVG(rating)
    FROM genres NATURAL JOIN movie_genres NATURAL JOIN movies
    GROUP BY genre_id
);

Note: since SQLite doesn’t support the ANY/ALL operator. The query needs to be rewritten as the following:

SELECT label, AVG(rating)
FROM genres NATURAL JOIN movie_genres NATURAL JOIN movies
GROUP BY genre_id
HAVING AVG(rating) >= (SELECT AVG(rating) AS avg_rating
FROM genres NATURAL JOIN movie_genres NATURAL JOIN movies
GROUP BY genre_id
);
GROUP BY genre_id
ORDER BY avg_rating DESC LIMIT 1
);

Output:

| label | avg        |
|-------+------------|
| Short | 8.333333333 |

(1 row)

6. Find the largest number of roles participating in one single movie.

Solutions:

SELECT MAX(role_num)
FROM (SELECT count(*) AS role_num FROM roles GROUP BY movie_id)
AS R;

Output:

<table>
<thead>
<tr>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>308</td>
</tr>
</tbody>
</table>

(1 row)

7. Find the most productive director. Report the name and the number of movies he or she has directed. There can be ties; if so, you should report for each of them.

Solution:

SELECT name, count(*)
FROM people NATURAL JOIN directors NATURAL JOIN movies
GROUP BY person_id
HAVING count(*) >= ALL (SELECT count(*)
FROM people NATURAL JOIN directors NATURAL JOIN movies
GROUP BY person_id
);

Note: since SQLite doesn’t support the ANY/ALL operator. The query needs to be rewritten as the following:

SELECT name, count(*)
FROM people NATURAL JOIN directors NATURAL JOIN movies
GROUP BY person_id
HAVING count(*) >= (
SELECT count(*) AS cnt
FROM people NATURAL JOIN directors NATURAL JOIN movies
GROUP BY person_id
ORDER BY cnt DESC LIMIT 1
);

Output:

<table>
<thead>
<tr>
<th>name</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hitchcock, Alfred (I)</td>
<td>10</td>
</tr>
</tbody>
</table>

(1 row)

8. Find the name of everyone who is both a director and an actor/actress, but never directed and starred in the same movie.

Solution:

SELECT name
FROM people NATURAL JOIN (SELECT directors.person_id
                          FROM directors INNER JOIN roles
                          ON directors.person_id = roles.person_id
                          ) EXCEPT
                          (SELECT person_id
                          FROM directors NATURAL JOIN roles
                          )
                          ) AS R
ORDER BY name;

Output:

<table>
<thead>
<tr>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attenborough, Richard</td>
</tr>
<tr>
<td>De Sica, Vittorio</td>
</tr>
<tr>
<td>Hale, Alan (I)</td>
</tr>
<tr>
<td>Laughton, Charles</td>
</tr>
<tr>
<td>Leone, Sergio (I)</td>
</tr>
<tr>
<td>Lumet, Sidney</td>
</tr>
<tr>
<td>Moran, Nick (I)</td>
</tr>
<tr>
<td>Penn, Sean (I)</td>
</tr>
<tr>
<td>Preminger, Otto</td>
</tr>
<tr>
<td>Roth, Vanessa</td>
</tr>
<tr>
<td>Sheridan, Jim (I)</td>
</tr>
<tr>
<td>Spears, Randy (I)</td>
</tr>
<tr>
<td>Stanton, Andrew (I)</td>
</tr>
<tr>
<td>Summerville, Slim</td>
</tr>
<tr>
<td>Zinnemann, Fred</td>
</tr>
</tbody>
</table>

(15 rows)