6.814 - Problem Set 1

SQL

Joseph Lin (joelin) and Long Nguyen (lpn)

September 17, 2017

Q1

Assumptions:
"people" is taken to mean a full row from the people table. A smaller selection (e.g. just id and name) can be made simply by replacing '*' in the query with the desired rows.

SQL query:

```
SELECT * FROM people WHERE name LIKE "John %";
```

Result (first 10 lines):

```
id          name          birth_year  death_year
----------  ------------  ----------  ----------
nm0000004   John Belushi  1949        1982
nm0000024   John Gielgud  1904        2000
nm0000078   John Wayne    1907        1979
nm0000092   John Cleese   1939
nm0000118   John Carpent  1948
nm0000131   John Cusack   1966
nm0000135   John Denver   1943        1997
nm0000237   John Travolt  1954
nm0000247   John Woo      1946
nm0000290   John Barry    1933        2011
```

Q2

SQL query:

```
SELECT title,
       year
FROM movies
WHERE year NOT NULL
ORDER BY year
LIMIT 5;
```

Result:

```
title       year
----------  ----------
Miss Jerry  1894
The Corbet  1897
Reproducti  1897
O Campo Gr  1898
O Carnaval  1898
```

Q3

SQL query:

```
SELECT AVG(runtime)/60
FROM movies
WHERE year = 1963;
```

Result:

```
AVG(runtime)/60
----------------
1.50278634587703
```

Q4

SQL query:

```
SELECT title
FROM movies,
     ratings
WHERE id = movie_id
ORDER BY rating DESC
LIMIT 1;
```

Result:

```
title
-------------------
The Maltese Phallus
```

Q5

Assumptions:
There are multiple people named Daniel Craig. As a result, this query returns all movie titles in which someone named Daniel Craig is a cast member.

SQL query:

```
SELECT DISTINCT movies.title
FROM movies,
     cast_members,
     people
WHERE movies.id = cast_members.movie_id
    AND cast_members.person_id = people.id
    AND people.name = "Daniel Craig";
```

Result (first 10 lines):

```
title
----------
Obsession
Love Is th
The Trench
Hotel Sple
Love & Rag
```

```
Some Voice
The Mother
Sylvia
The Jacket
Enduring L
```

Q6

SQL query:

```
SELECT AVG(runtime) FROM movies, ratings
WHERE id = movie_id AND rating > 9.0;
```

Result:

```
AVG(runtime)
----------------
82.7180722891566
```

Q7

SQL query:

```
SELECT COUNT(person_id)
FROM cast_members,
    (SELECT id
     FROM movies
     ORDER BY runtime DESC
     LIMIT 1)
WHERE id = movie_id;
```

Result:

```
COUNT(person_id)
----------------
4
```

Q8

SQL query:

```
SELECT name
FROM people,
    (SELECT person_id,
            COUNT(movie_id)
     FROM directors
     GROUP BY person_id
     ORDER BY 2 DESC
     LIMIT 1)
WHERE id = person_id;
```

Result:

```
name
-------------
Jirô Yoshino
```

Q9

Assumptions:
Nothing is assumed about the contents of directors or cast_members; as a result, a person could be both a director and a cast member of the same movie. Furthermore, it is assumed that the two people acting/directing together need not be distinct people.

SQL query:

```
SELECT p1.name AS director_name,
       p2.name AS cast_member_name
FROM people AS p1,
     people AS p2,
    (SELECT d.person_id AS did,
            c.person_id AS cid,
            COUNT(DISTINCT d.movie_id) AS nmovies
     FROM cast_members AS c,
          directors AS d
     WHERE c.movie_id = d.movie_id
     GROUP BY c.person_id,
              d.person_id
     ORDER BY nmovies DESC
     LIMIT 1)
WHERE p1.id = did
    AND p2.id = cid;
```

Result:

```
director_name  cast_member_name
-------------  ----------------
Jirô Yoshino  Jirô Yoshino
```

Q10

SQL query:

```
CREATE TEMP TABLE big_movies AS
SELECT movie_id,
       COUNT(person_id) AS ncast
FROM cast_members
GROUP BY movie_id
HAVING ncast >= 10;

CREATE TEMP TABLE big_cast AS
SELECT cm.movie_id,
       cm.person_id
FROM cast_members AS cm,
     big_movies AS bm
WHERE cm.movie_id = bm.movie_id;

CREATE TEMP TABLE max_shared_cast AS
SELECT mid1,
       MAX(nshared) AS max_nshared
FROM
    (SELECT bc1.movie_id AS mid1,
```

```
            bc2.movie_id AS mid2,
            COUNT(bc1.person_id) AS nshared
     FROM big_cast AS bc1,
          big_cast AS bc2
     WHERE bc1.person_id = bc2.person_id
         AND bc1.movie_id != bc2.movie_id
     GROUP BY bc1.movie_id,
              bc2.movie_id)
GROUP BY mid1;

SELECT title
FROM big_movies AS bm,
     max_shared_cast AS msc,
     movies
WHERE bm.movie_id = msc.mid1
    AND bm.ncast = msc.max_nshared
    AND bm.movie_id = id;
```

Result:

```
title
--------------
The Yankee Way
Henriette Jaco
Jettchen Geber
Die Jagd nach
Der Mann ohne
The Scrap of P
Fridericus Rex
Teru hi kumoru
The Lost City
The Lost City
```

Q11

SQL query:

```
SELECT name
FROM people,
     (SELECT person_id, MAX(year) - MIN(year)
     FROM cast_members, movies WHERE id = movie_id
     GROUP BY person_id ORDER BY 2 DESC LIMIT 1)
WHERE id = person_id;
```

Result:

```
name
--------------
John Malkovich
```

Q12

Assumptions: "Cast member" is taken to be uniquely determined by `person_id`; the name is given along with it for clarity. Additionally, it is assumed that Kevin Bacon is not considered in the list of cast members

who have acted with Kevin Bacon.

SQL query:

```
SELECT c1.person_id, p1.name
FROM cast_members AS c1,
     cast_members AS c2,
     people AS p1,
     people AS p2
WHERE c1.movie_id = c2.movie_id
     AND c1.person_id = p1.id
     AND c2.person_id = p2.id
     AND p2.name = "Kevin Bacon"
     AND c1.person_id != c2.person_id;
```

Result (first 10 rows):

```
person_id   name
----------  ----------------
nm0000430   Steve Guttenberg
nm0000620   Mickey Rourke
nm0001469   Barry Levinson
nm0005545   Jerry Weintraub
nm0111021   Bruce Brody
nm0469380   Ivan Král
nm0511742   Stu Linder
nm0816292   Peter Sova
nm0827663   Daniel Stern
nm0004730   Orson Bean
```

Q13

Note: When trying to find the Bacon Number of a specific actor name, replace `<actor name here>` on line 71 with the desired actor. When finding Bacon Numbers for multiple cast members, one only needs to run the `CREATE TEMP TABLE` commands once at the beginning. All subsequent queries need to only run lines 66-100.

SQL queries:

```
1  CREATE TEMP TABLE temp1 AS
2  SELECT DISTINCT movie_id
3  FROM cast_members,
4       people
5  WHERE name = "Kevin Bacon"
6      AND id = person_id;
7
8
9  CREATE TEMP TABLE temp2 AS
10 SELECT DISTINCT c2.movie_id
11 FROM cast_members AS c1,
12      cast_members AS c2,
13      temp1 AS t
14 WHERE c1.movie_id = t.movie_id
15     AND c1.person_id = c2.person_id
```

```
    AND c2.movie_id NOT IN temp1;


CREATE TEMP TABLE temp3 AS
SELECT DISTINCT c2.movie_id
FROM cast_members AS c1,
     cast_members AS c2,
     temp2 AS t
WHERE c1.movie_id = t.movie_id
    AND c1.person_id = c2.person_id
    AND c2.movie_id NOT IN
        (SELECT *
         FROM temp1
         UNION SELECT *
         FROM temp2);


CREATE TEMP TABLE temp4 AS
SELECT DISTINCT c2.movie_id
FROM cast_members AS c1,
     cast_members AS c2,
     temp3 AS t
WHERE c1.movie_id = t.movie_id
    AND c1.person_id = c2.person_id
    AND c2.movie_id NOT IN
        (SELECT *
         FROM temp1
         UNION SELECT *
         FROM temp2
         UNION SELECT *
         FROM temp3);


CREATE TEMP TABLE temp5 AS
SELECT DISTINCT c2.movie_id
FROM cast_members AS c1,
     cast_members AS c2,
     temp4 AS t
WHERE c1.movie_id = t.movie_id
    AND c1.person_id = c2.person_id
    AND c2.movie_id NOT IN
        (SELECT *
         FROM temp1
         UNION SELECT *
         FROM temp2
         UNION SELECT *
         FROM temp3
         UNION SELECT *
         FROM temp4);

WITH actor_movies AS
    (SELECT movie_id
     FROM cast_members,
          people
```

```
      WHERE person_id = id
          AND name = "<actor name here>")
SELECT (CASE
             WHEN cnt1 > 0 THEN 1
             WHEN cnt2 > 0 THEN 2
             WHEN cnt3 > 0 THEN 3
             WHEN cnt4 > 0 THEN 4
             WHEN cnt5 > 0 THEN 5
             ELSE 6
        END) AS bacon_number
FROM
    (SELECT COUNT(*) AS cnt1
     FROM actor_movies,
          temp1
     WHERE actor_movies.movie_id = temp1.movie_id),
    (SELECT COUNT(*) AS cnt2
     FROM actor_movies,
          temp2
     WHERE actor_movies.movie_id = temp2.movie_id),
    (SELECT COUNT(*) AS cnt3
     FROM actor_movies,
          temp3
     WHERE actor_movies.movie_id = temp3.movie_id),
    (SELECT COUNT(*) AS cnt4
     FROM actor_movies,
          temp4
     WHERE actor_movies.movie_id = temp4.movie_id),
    (SELECT COUNT(*) AS cnt5
     FROM actor_movies,
          temp5
     WHERE actor_movies.movie_id = temp5.movie_id);
```

(a)

RESULT:

```
bacon_number
------------
2
```

(b)

RESULT:

```
bacon_number
------------
3
```

(c)

ASSUMPTION: Because there are two Spencer Tracy's, we chose the one born in 1900. This required adding

```
AND birth_year = 1900
```

after the

```
AND name = "<actor name here>"
```

on line 71.

Result:

```
bacon_number
------------
3
```

(d)

Result:

```
bacon_number
------------
3
```