Quiz

CSC230 Database Technologies for Analytics

06 November 2019

1 Getting started

1. Log into MySQL. Execute the *waterfalls.sql* script. (Type your username in place of Michael Stonebraker's.)

```
mysql -u mstonebraker14 -p
mysql> source waterfalls.sql;
```

2. Examine the tables in the database.

```
SHOW TABLES;

SELECT * FROM county;

/* The fall_description table is empty */
/* The fall_location table is empty */

SELECT * FROM gov_unit;

SELECT * FROM new_falls;

/* The other_unit table is empty */

SELECT * FROM owner;

/* The parent_example table is empty */

SELECT * FROM small_township;

SELECT * FROM state;

SELECT * FROM township;

SELECT * FROM trip;
```

```
SELECT * FROM upfall;
```

2 Exercises

1. Compose a query that produces in each row the name of a waterfall and the name of the county in which that waterfall is located. Ordered the results by county in ascending alphabetical order.

```
/*
 * name of waterfall, name of county,
 * ordered by county
*/
SELECT f.name AS 'Name of waterfall',
       c.name AS 'Name of county'
  FROM upfall f INNER JOIN county c ON f.county_id = c.id
  ORDER BY c.name, f.name;
```

2. Compose a query that produces in each row the name of a county and the number of waterfalls in that county. Order the results by county in ascending alphabetical order.

```
/*
 * name of county, number of waterfalls in county,
 * ordered by county
*/
SELECT c.name AS 'Name of county',
       COUNT(*) AS 'Number of waterfalls in county'
  FROM upfall f INNER JOIN county c ON f.county_id= c.id
  GROUP BY c.name
  ORDER BY c.name;
```

3. Compose a query that produces the names of counties that have only one waterfall.

```
/* names of counties that have only one waterfall */
SELECT name AS 'Name of county' FROM
   (SELECT c.name AS name, COUNT(*) AS numberOfFalls
       FROM upfall f INNER JOIN county c ON f.county_id = c.id
       GROUP BY c.name) AS t
  WHERE numberOfFalls = 1
  ORDER BY name;
```

4. Compose a query that produces the names of counties that have more than one waterfall.

```
SELECT name AS 'Name of county' FROM
   (SELECT c.name AS name, COUNT(*) AS numberOfFalls
       FROM upfall f INNER JOIN county c ON f.county_id = c.id
       GROUP BY c.name) AS t
  WHERE numberOfFalls > 1
  ORDER BY name;
```

5. Compose a query that produces in each row the name of a city or township and the name of the county in which that city/township is located. Order the results first by county and then by city/township in ascending alphabetic order.

```
/*
 * name of city or township, name of county ordered by county,
 * city/township
*/
```

```
SELECT a.name AS 'City/Township', b.name AS 'County'
  FROM gov_unit a INNER JOIN gov_unit b ON a.parent_id = b.id
  WHERE a.type IN ('City', 'Township') AND b.type = 'County'
  ORDER BY b.name, a.name;
```

6. Compose a query that produces in each row the name of a city or township, the name of the county in which that city/township is located, and the name of the state in which that county is located. Order the results by state, county, and city in ascending alphabetic order.

```
/*
 * name of city or township, name of county, name of state
 * ordered by state, county, city
*/
SELECT a.name AS 'City/Township',
       b.name AS 'County',
       c.name AS 'State'
  FROM gov_unit a INNER JOIN gov_unit b
    ON a.parent_id = b.id INNER JOIN gov_unit c
    ON b.parent_id = c.id
  WHERE a.type IN ('City', 'Township') AND b.type = 'County'
  ORDER BY c.name, b.name, a.name;
```

7. Compose a query that produces in each row the name of a waterfall together with the length of that waterfall's description. Order the results by the length of the description in descending order, and then by the name in ascending alphabetic order.

```
/*
 * name of waterfall, length of waterfall's description
 * ordered by length of description in descending order, name
*/
SELECT name, LENGTH(description) AS 'Length_of_description'
  FROM upfall ORDER BY LENGTH(description) DESC, name;
```

8. Compose a query that produces in each row the name of a trip and the name of a waterfall that we will visit on that trip.

```
/*
 * name of trip, name of waterfall
*/
SELECT t.name, f.name FROM
  upfall f INNER JOIN trip t ON t.stop = f.id
  ORDER BY t.name, f.name;
```

9. Compose a query that produces in each row the name of a trip together with the name of the first waterfall that we will visit on that on trip.

```
/*
 * name of trip, name of first waterfall on trip
*/
SELECT t.name, f.name FROM
  upfall f INNER JOIN trip t ON t.stop = f.id
  WHERE parent_stop IS NULL;
```