Project 2: SQL

Due on 10/21/2016

In this project, you will write SQL queries for the SQLite3 RDBMS for two different databases: the *Sales* database, and the *IMDB* database. You can use *only SQLite3* for your queries: no other tools or languages are allowed. It is okay to have multiple SQL statements as part of your answer to a single question. We might tweak the given instances to get different valid instances – your queries should still run without any issues and output the correct results on the modified instances as well.

PART A: THE SALES DATABASE **[50%]**

Download the *Sales* database as a zipped folder from this URL:

`http://pages.cs.wisc.edu/~paris/cs564-f16/data/salesdb.tar.gz`

To load the database, simply type ".read create-sales.sql" on the SQLite3 command prompt. The Sales database has the following schema:

- **Holidays** (WeekDate, IsHoliday)
- **Stores** (Store, Type, Size)
- **TemporalData** (Store, WeekDate, Temperature, FuelPrice, CPI, UnemploymentRate)
  Store is a foreign key referencing Stores (Store).
  WeekDate is a foreign key referencing Holidays (WeekDate).
- **Sales** (Store, Dept, WeekDate, WeeklySales)
  Store is a foreign key referencing Stores (Store).
  WeekDate is a foreign key referencing Holidays (WeekDate).
  (Store, WeekDate) is a foreign key referencing TemporalData (Store, WeekDate).

Write SQL queries that obtain the answers to the following questions:

1. **[5%]** Which stores had the largest and smallest overall sales during holiday weeks? Output the stores and the overall sales (2 tuples).
2. **[5%]** Get the top 20 departments overall ranked by total sales normalized by the size of the store where the sales were recorded. Output the department and the normalized total sales.

3. **[5%]** Get the stores at locations where the unemployment rate exceeded 10% at least once but the fuel price never exceeded 4.

4. **[5%]** How many non-holiday weeks had larger sales than the overall average sales during holiday weeks?

5. **[5%]** Get the total sales per month overall for each type of store. Since SQLite3 does not support native operations on the DATE datatype, use the LIKE predicate and the string concatenation operator ("||") of SQLite3 to create a workaround.

6. **[10%]** Which stores have had sales in every department in that store for every month of at least one calendar year among 2010, 2011, and 2012?

7. **[15%]** For each of the 4 numeric attributes in TemporalData, are they positively or negatively correlated with sales? For our purposes, the intuitive notion of "correlation" is defined using a standard statistical quantity known as the Pearson correlation coefficient. Given two numeric random variables X and Y, it is defined as follows:

$$\rho_{X,Y} = \frac{E[XY] - E[X]E[Y]}{\sqrt{E[X^2] - E[X]^2}\sqrt{E[Y^2] - E[Y]^2}}$$

On a given sample of data with n examples each for X and Y (label them x_i and y_i respectively for $i = 1 \ldots n$), it is estimated as follows:

$$\rho_{X,Y} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

In the above, $\bar{x}$ and $\bar{y}$ are the sample means, i.e., $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$, and $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} y_i$. Your SQL query should output an instance with the following schema: Output (AttributeName VARCHAR(20), CorrelationSign Integer) and 4 tuples, e.g.,{("CPI", 1), ("Temperature", -1), ...}. In your query, the values of AttributeName can be hard-coded string literals, but the values of CorrelationSign must be computed automatically using SQL queries over the given database instance.

PART B: THE IMDB DATABASE **[50%]**

Download the *IMDB* database as a zipped folder from this URL:

`http://pages.cs.wisc.edu/~paris/cs564-f16/data/imdb.tar.gz`

To load the database, simply type ".read create-imdb.sql" on the SQLite3 command prompt. The IMDB database has the following schema:

- **actor** (id, fname, lname, gender)
- **movie** (id, name, year)
- **directors** (id, fname, lname)
- **genre** (mid, genre)
- **casts** (pid, mid, role)
  pid is a foreign key referencing actor (id).
  mid is a foreign key referencing movie (id).
- **movie_directors** (did, mid)
  did is a foreign key referencing directors (id).
  mid is a foreign key referencing movie (id).

Before you attempt to write any SQL queries, you will need to create the necessary indexes to speed up the query execution. To create an index, we can use the following statement:

```
CREATE [UNIQUE] INDEX index_name ON table_name(col_1, col_2, ...);
```

For this exercise, we suggest that you use the following indexes:

```
CREATE UNIQUE INDEX movieid ON movie(id);
CREATE UNIQUE INDEX actorid ON actor(id);
CREATE UNIQUE INDEX directorsid ON directors(id);

CREATE INDEX castsmid ON casts(mid);
CREATE INDEX castspid ON casts(pid);

CREATE INDEX movieyear ON movie(year);

CREATE INDEX moviedirectorsmid ON movie_directors(mid);
CREATE INDEX moviedirectordid ON movie_directors(did);
```

You are welcome to use different indexes than the ones above when running your queries. However, you should make sure that all of the SQL queries run in reasonable time (< 5 minutes for each) using the given indexes.

Write SQL queries to answer the following questions:

1. **[8%]** List all actors (their first and last name) who have played in at least 10 different movies in 2010.
2. **[8%]** List all people (their first and last name) who have directed and played in the same movie in 2000.

3. **[8%]** List the top 100 directors who have directed the most movies from 1990 to 2010, in descending order of the number of movies they have directed. Output their first name, last name, and number of movies directed.

4. **[8%]** For each year, count the number of movies in that year that had only female actors. (A movie without any actors is a movie that has only female actors).

5. **[8%]** A decade is any sequence of 10 consecutive years (e.g., 1964, 1965, ..., 1973 is a decade). Find the decade with the largest number of films (output only the first year of the decade).

6. **[10%]** The Bacon number of an actor is the length of the shortest path between the actor and Kevin Bacon in the "co-acting" graph. Kevin Bacon has Bacon number 0; all actors who acted in the same film as Kevin Bacon have Bacon number 1; all actors who acted in the same film as some actor with Bacon number 1 (but not with Bacon himself) have Bacon number 2, and so on. Find the number of actors with Bacon number 2.

DELIVERABLES

You can download and install SQLite3 from the following link (the current version of SQLite3 is 3.14.2):

`https://www.sqlite.org/download.html`

You are required to submit a zipped folder which contains a ".sql" file per question. Upload the zipped folder using Canvas in `Learn@UW` (Project 2). You should name your SQL query files as "query<part><number>.sql", e.g., the file "queryA2.sql" is for question 2 of part A.