Project 2: SQL

due 3/8/19

DBMS

Use of postgres is highly recommended. Other DBMS's may be used for development but the answers must be able to run against postgres server in the lab.

Part A: Querying the Sales Database [50%]

The **Sales** database tables are available under "**hw2**" schema on the lab postgres system (under cs564instr database). You are encouraged to create views in your own schema to make it easier to access the hw2 tables.

[The tables can also be created using the script available in ~shatdal/lab/Sales.sql. If you are using postgres on your personal machine, to load the database, simply type "psql -f Sales.sql" on the command prompt or "\i Sales.sql" inside psql. It would create a schema **hw2** in your postgres DBMS.]

The hw2 schema has the following 4 tables. The key of each table is underlined and the foreign keys are also mentioned:

- **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 over the given schema that obtain the answers to the following questions:

1. Which stores had the largest and smallest overall sales during holiday weeks?
2. Get the stores at locations where the unemployment rate exceeded 10% at least once but the fuel price never exceeded 4.
3. How many non-holiday weeks had larger sales than the overall average sales during holiday weeks?
4. Get the total sales per month, and its contribution to total sales (across months) overall for each type of store.
5. 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?

6. For each of the 4 numeric attributes in TemporalData, are they positively or negatively correlated with sales and how strong is the correlation? Your SQL query should output an instance with the following schema with 4 rows:
   Output6 (AttributeName VARCHAR(20), CorrSign char(1), CorrValue (float))
   e.g., (Temperature, -, -0.5) In your query, the values of AttributeName can be hardcoded string literals, but the other values must be computed automatically using SQL queries over the given database instance.
7. Which departments contribute to at least 5% of store sales across for at least 3 stores? List the departments and their average contribution to sales across the stores.
8. Get the top 10 departments overall ranked by total sales normalized by the size of the store where the sales were recorded.
9. For the top 10 departments (in above query, #8), find the 3-monthly moving average and cumulative total of sales.
10. The accounting department has asked for the following report. Write a SQL Query that would most closely produce the needed report. Quarters are defined traditionally, with Jan,Feb,March being Q1, etc.

Year	Quarters	Store Type A Sales	Store Type B Sales	Store Type C Sales
2010	Q1	…	…	…
2010	Q2	…	…	…
2010	Q3	…	…	…
2010	Q4	…	…	…
2010	—	…	…	…
…				
2012	Q4	…	…	…
2012	—	…	…	…

Note: use of LIMIT N; feature in postgres is not allowed to constrain number of rows returned.

PART B: Sampling Application [50%]

Since postgres (and most DBMS's) don't allow sampling, the goal is to create a JDBC application that would let user create samples of data in the DBMS. The Sales data from Part A could be used for developing/testing the application. You would use sampling without replacement.

The application should:

1. The JDBC connection should be to the lab postgres server. Instructions are posted on the webpage as to how to connect to lab postgres server. You may optionally also connect to your own hosted postgres for development/testing. Note that lab postgres can only be connected to from lab linux machines because of security protocols.
2. Accept a table name or a query (prompts can distinguish the two if needed)
3. Ask for how many sample rows are desired
4. Ask if the user wants a table created for the sampled rows (instead of being returned). [These tables could be sampled in next iteration.]
5. Fetch/insert exactly that number of random samples from the table or query result
6. Allow the user to reset the seed of the random number generator
7. Errors may be given for any syntax error in queries or invalid table entries
8. If number of samples requested is greater than rows in the table/query, all rows should be returned (or inserted in the sample table) and a message should be given noting that fact.
9. Ask if user wants more samples (or quit)

The application MAY NOT fetch the entire table or query result and do the sampling work in the application itself. (That is, it is assumed that the original table is too large to fit in memory.)

You may create additional tables (and insert rows) in the database (under your schema).

You may execute any query you like in the database.

Hint: One way to number all rows in a table is to use the "row_number()" ordered-analytic function.

The algorithm for random sampling of N rows (from Knuth: Art of computer programming, volume 2: semi-numerical algorithms) is on the last page.

Deliverables

You are required to submit a zipped folder with the following contents:

1. A ".sql" file per question. Name your files as "query<number>.sql", e.g., the file "query2.sql" is for question 2 (of part A). These would contain the SQL and the result rows.
2. The JDBC app (both java file and executable). Show the result of sampling 10 rows from each of the tables in part A. Additional testing/validation would be done for grade.
3. Readme.txt with your group members' name, CS logins and Wisc ids. and describe any assumptions etc made in the application that are not in the documentation of the code.

3.4.2. Random Sampling and Shuffling

Many data processing applications call for an unbiased choice of n records at random from a file containing N records. This problem arises, for example, in quality control or other statistical calculations where sampling is needed. Usually N is very large, so that it is impossible to contain all the data in memory at once; and the individual records themselves are often very large, so that we can't even hold n records in memory. Therefore we seek an efficient procedure for selecting n records by deciding either to accept or to reject each record as it comes along, writing the accepted records onto an output file.

Several methods have been devised for this problem. The most obvious approach is to select each record with probability n/N; this may sometimes be appropriate, but it gives only an *average* of n records in the sample. The standard deviation is $\sqrt{n(1-n/N)}$, and the sample might turn out to be either too large for the desired application or too small to give the necessary results.

Fortunately, a simple modification of the "obvious" procedure gives us what we want: The $(t+1)$st record should be selected with probability $(n-m)/(N-t)$, if m items have already been selected. This is the appropriate probability, since of all the possible ways to choose n things from N such that m values occur in the first t, exactly

$$\binom{N-t-1}{n-m-1}\Big/\binom{N-t}{n-m} = \frac{n-m}{N-t} \qquad (1)$$

of them select the $(t+1)$st element.

The idea developed in the preceding paragraph leads immediately to the following algorithm:

Algorithm S (*Selection sampling technique*). To select n records at random from a set of N, where $0 < n \le N$.

S1. [Initialize.] Set $t \leftarrow 0$, $m \leftarrow 0$. (During this algorithm, m represents the number of records selected so far, and t is the total number of input records that we have dealt with.)

S2. [Generate U.] Generate a random number U, uniformly distributed between zero and one.

S3. [Test.] If $(N-t)U \ge n-m$, go to step S5.

S4. [Select.] Select the next record for the sample, and increase m and t by 1. If $m < n$, go to step S2; otherwise the sample is complete and the algorithm terminates.

S5. [Skip.] Skip the next record (do not include it in the sample), increase t by 1, and go back to step S2. ▮