CSE 462: Project #1 (due 10/25/13)

You will be using the following relational schema:

Book(Title,Price,Year)
Author(Name,Booktitle,Position)

Keys are underlined. The attribute Booktitle in Author is a foreign key referencing Book(Title).

Submit your solutions as a single .zip file, containing THREE files: a text file with all SQL queries, a text file with all query results, and a jar file. Use `submit_cse462`.

Problem 1 (20 pts): SQL Queries

Write the following queries in SQL2, defining appropriate views if necessary:

Productivity: For each author, return the maximum number of books published in a single year, in descending order.

Slackers: Return every author who was in the second or third position in all of his/her books.

Moneymakers: Return all the authors whose average book price was at least twice the average book price of every other author.

Histogram: Assume you are given another table Ranges(Low,High), containing price ranges. For every such range, return the number of books in it. You can assume that the prices are positive integers in the range $[0, 100]$ and that the Low value of a range is equal to the High value of the previous range[1] plus 1. Also, the smallest Low value is 0 and the largest High value is 100. An example of price ranges:

$$[0, 10], [11, 15], [16, 50], [51, 100].$$

Crowd pleasers: Make the same assumptions as in **Histogram**. Return all the authors that had a book in every price range.

Run the queries against the sample database, and report the results.

Problem 2 (20 pts): Top K

Introduction. Given a relation R, an integer $K > 0$, and a scoring function $s : R \to \mathbb{R}$, the Top$-K$ rows of R are the K tuples in R with the highest scores according to s. The scoring function s has two important properties. First, it is deterministic, that is, it always computes the same score for a given tuple. And second, the distinct tuples $t_1, t_2 \in R$ may have the same score, that is, it may be the case that $s(t_1) = s(t_2)$. For the purpose of this assignment, you may consider the scoring function a black box.

Requirements. Your assignment is to write a class in Java that computes the Top$-K$ rows given R, K, and s. You will be given a library in the form of a jar file containing two interfaces: `TopK` and `ScoringFunction`. Your only task is to implement the `TopK` interface. A skeleton class `TopKImpl` implementing `TopK` will be provided as a starting point.

In order to help you test your implementation, a scoring function implementation is available in the library. Sample code showing how to use this scoring function and how to call the methods in

[1] If one exists.

`TopK` will also be provided. All data access will be done using JDBC, therefore, you must connect to Oracle as described in the recitation in order to test your code.

Observations. If R contains fewer than K tuples, then all tuples in R are Top$-K$ tuples. If two tuples have the same score, you may choose between them arbitrarily (breaking ties). Note that as a result of breaking ties, it is possible for some tuples outside the final Top$-K$ set to have the same score as the tuple with the lowest score in the Top$-K$ set.

Problem 4 (Extra credit: 10 pts)

Use the following relational schema (keys are underlined):

`STOCKS(Symbol,Date,Price)`

You can assume that dates have been converted to consecutive integers.

The *maximum growth period* for a stock is defined to be the longest period of time (in days) in which the price of the stock never declined. Write an SQL query that lists for every stock the length of its maximum growth period.