15 HW #7A: SQL Joins (II)

Please answer the following questions, making sure to only use the syntax from class.[9]

Before beginning the assignment make sure that you have indexes applied to the symbol and return date variables in both stock tables (otherwise the queries will take an eternity). The following commands will create the indexes necessary to complete the assignment.

```
create index s2010_symb_retdate on stocks.s2010 (symb, retdate);
create index s2011_symb_retdate on stocks.s2011 (symb, retdate);
```

The best approach to learning from these problems is to complete them using pen and paper, working by yourself and then using your group to double check your results. The First Five problems provide a short overview of the core concepts in the assignment, so make sure that you understand them. The Main Problems section contains questions which range from easy to very difficult. Remember to don't get stuck! If a problem is taking a long time or is too difficult, *use your group!*

First Five

1. Write a query which returns the symbol, date, volume and the total volume traded for that stock up to (but not including) that *day's* volume. Only consider 2010 data. Make sure that if there is no previous volume that the cumulative volume is set to *zero* and isn't null.
2. Write a query which returns the number of days that the stock has been traded, cumulatively, in 2010. Specifically, the first time that the stock appears it should be "1", the second date that it exists it should be "2". This should return a table with 3 columns (symb, retdate and cumulative trading days) and should have the same number of rows as the original s2010 dataset.
3. For each exchange in 2010 return the stock with the highest total traded volume in the year 2010. This should return two rows (one for each exchange) and 3 columns (exchange name, symbol and total volume traded for that year)
4. For each exchange in 2010 return the stock with the *second* highest total traded volume in the year 2010. This should return two rows (one for each exchange) and 3 columns (exchange name, symbol and total volume traded for that year)
5. Create a dataset which contains the following columns: symbol, date that the symbol first appears in 2010 and the total volume traded in the first 35 days that the stock is traded in 2010. If a stock has a first date late in the year, ignore the spill over into 2011.

Main Problems

1. Using the data from 2010, write a query which returns the seven day moving average for each stock's closing price. Only look at stocks whose symbols begin with the letter "A". Note that this should *not* be the last seven points, but instead the last seven *days*, not including the current day.
2. For each stock from 2010, write a query which returns the symbol, the closing price, the return date and the closing price on the previous day it was traded. Note that you just want to take the price from the previous row, if the rows are ordered by return date. Also, only do this for stocks that begin with the letter 'A'.
3. For each symbol in 2010, return the day(s) where it has its highest volume traded[10]

[9] Specifically if you decided to look ahead, analytic functions are not to be used to answer these questions.

[10] There could be multiple days for a symb.

4. Using only a single join, for each symbol, return the closing price on the first and last day that the stock is traded in 2010.
5. How many missing days are there in total? Make sure to only count missing days **after** a symbol has been in the data. So if a stock doesn't appear in the data until February, January does not count as missing. If a stock leaves the market before the end of the year, you can either count the days past their exit as missing or as not missing, just be consistent across all stocks.[11]
6. For each symbol that appears in 2011, calculate the number of missing trading days that it has in January 2010.
7. Write a query which returns the userid, trans_dt, amt, and the total amount the user has spent up to (but not including) that *day's* transaction. Make sure that if there is no previous transaction the amount is set to *zero* and isn't null.
8. Write a query which returns a purchase number for each order. In other words, for each row return the amount, userid, date and the number of the sale, incrementing from one for each order.
9. For each local in the table, return the most common month of an order.
10. For each local in the table, return the *second* most common month of an order.
11. Create a dataset which has the following information: (1) userid, (2) date of first transaction and (3) number of transactions within the first 35 days of their first transaction.

[11] In other words, if a stock leaves the data it maybe because the stock delisted, in which case the data is not missing.