**FULL INSTRUCTIONS**

**5 Files that need to complete the project. I have been sent to you, and mentioned in this instruction list are:**

1. **This instruction you are reading now.**
2. **qdb3.zip AFTER unzipping called** nqdb3.db (I am sending now).
3. gettingStarted.R pdf file (sent earlier)
4. **DaySummary\_v01.csv** (sent earlier)
5. **FiveMins\_v01.csv (sent earlier)**
6. TradesANDQuotes\_v01.cv (sent earlier)

To complete this project you will need (everything is open-source):

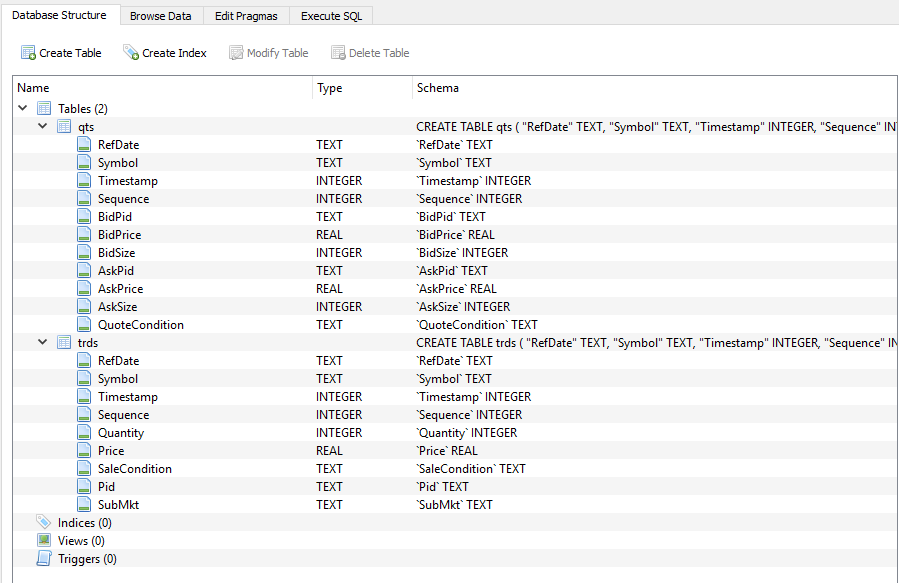
* A text editor (Notepad will work, or Notepad++ <https://notepad-plus-plus.org/>), or an IDE like R-Studio (<https://www.rstudio.com/>)
* R software (<https://www.r-project.org/>)
* Some R packages, particularly RSQLite (install these from within R) (<http://sqlitebrowser.org/)>

==================================================================================

START:

This project begins with the attached SQLite database that has two tables.

(ATTACHED ZIP FILE HAS THIS)

* The file is zipped: but called nqdb3.db
* one of trades (“trds”) and one of quotes (“qts”).
* This will OPEN after downloading (http://sqlitebrowser.org/) and opening the file.
* Sample of what the file looks like after you unzip and open in SQLITE is below:
* 

1. You will need to send SQL commands from your R code to the SQLite database to extract trades and quotes.
2. You will also do some filtering and calculations.
3. Those may be done in either SQL or R, as you see fit. Note that SQLite does not have all the SQL functionality of other databases.

* The attached file “gettingStarted.R pdf” gives you a few hints on how to begin using the SQLite database from R. (I already sent you)
* Use notepad or WORD to send an OUTPUT syntax summary file. This would be the main Syntax you used to complete a step or calculation.
* For example:

#mean(x) to calculate the mean of stocks traded on day 9

**Section 1: Summarize trades by stock**

The trades table includes all trades that occurred in 4 stocks on August 03, 2015. Each trade record has additional information such as the time that it occurred and the exchange where it occurred (“pid”). (TradesAndQuotes\_v01.csv)

* **Nasdaq’s pid is ‘Q’.**
* **An important field is the Sale Condition. (database in zip file)**
* **Not all trades are counted in volume or price updates (price is often called “last sale”).**

1. Use the information in the sale condition field (database in zipped file) and the table on the last page of this document to determine which trades to include/exclude in your calculations.



This is the row from zipped database

1. Trades may have multiple sale condition codes.
2. If any one of the codes indicates exclusion, then the trade should be ignored.

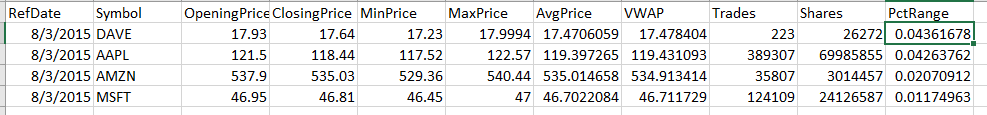
For example, an opening trade would have “O” in the sale condition. If it was determined by a crossing auction, then it would also have an “X”.

An “odd lot” trade (meaning a trade of less than 100 shares) would have an “I” in the Sale Condition field.

Odd-lot trades cannot set the last-sale price of a stock and are therefore excluded from calculations that involve last sale price.

* **Write code that creates an exact copy of the attached file “DaySummary\_v01.csv”.**

Sample of the **DaySummary\_v01.csv** is below:

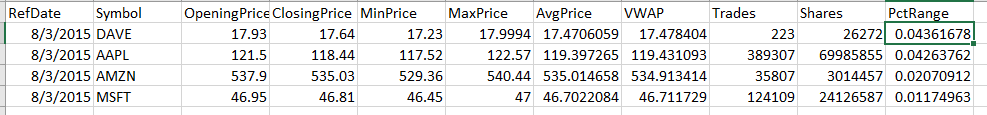


Hints:

* For each stock, get the prices from the opening and closing cross trades on Nasdaq (hint: use sale conditions and pid fields. With the correct filters there will only be one open and one close trade per stock).
* For each stock, get the min, max, average, and VWAP for consolidated last-sale eligible trades (hint: exclude trades that have a sale condition code with “No” under “Consolidated Processing Guidelines”, “Update Last” in the table below. Ignore the special cases in the footnotes.)
* **VWAP is volume-weighted average last sale price. It is calculated by weighting each trade price by its volume (price\*quantity). Sum that value** across all included trades. Then divide that value by the sum of all volume in the included trades.
* For **each stock, get the total share volume and number of trades for volume-eligible trades** (hint: exclude trades that have sale condition codes with “No” under “Update Volume” in the table below).
* **Combine all of the results above in a single table, with one row per stock.**
* **Calculate the percent price range for each stock [use (max-min)/close].**
* **Sort them from most to least volatile as measured by PctRange.**
* **Write this table to a .csv file called DaySummary\_v02.csv.**

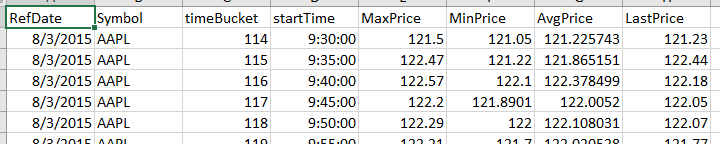
**It should be produced only by your code and be identical to the version attached except for the version number in the file name.**

**SHOULD LOOK LIKE THIS ONE**



**Part 2: Summarize prices in 5-minute time buckets**

Use the concepts you learned above (about what counts as a last-sale-eligible trade for example) to summarize trading within each 5 minute period.

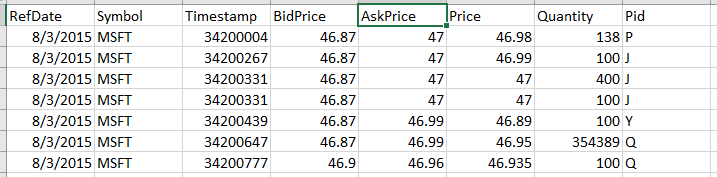
* **Re-create the attached file “FiveMins\_v01.csv”.**
* **Should look like this one: of FiveMins\_v01.csv**
* **Provide max, min, average and last price for each 5-minute time period (in the file these are called timeBuckets, which count each 5 minute period since midnight), for each stock (only include market hours 9:30-16:00).**

Hints:

* Timestamps are in milliseconds since midnight (for example 9:30am is 34200000 and 4:00pm is 57600000)
* **You can create a start time for each bucket using the floor function after dividing the timestamp by 300,000ms.**
* **SQLite doesn’t have a floor function but you can use cast as described in the next hint.**
* **SQL has commands to summarize within groups that you create in your query (for example, create a new field called timeBucket using something like “cast(Timestamp/300e3 as int)”, then summarize all the observations with the same timeBucket)**
* **Add a field for each time bucket that shows the start time in HH:MM:SS format.**
* **One way to do this is to create a function that transforms Timestamp (ms since midnight) into hours, minutes, and seconds and then returns them as a text in the correct format.**

**Part 3:**

* In this exercise we want to know the prices of the national best bid and offer when a trade occurred. This requires matching data in the trades table (trds) to data in the quotes table (qts).
* Note that to control the size of the file, quotes from only one stock are available. Include only trades that occurred between 9:30 and 16:00.
* Match each last-sale eligible trade with the national best bid and offer that was in effect for the stock at the time when the trade occurred (the quote in effect should be the most recent one with a timestamp that is less than or equal to the trade timestamp).
* After matching you should produce a file identical to the attached file “TradesAndQuotes\_v01.csv”.

Should look like TradesANDQuotes\_v01.cv below

* To limit the size of the final file, the output files should only include the first 100 trades and trades 100,000 through 100,050.

Then calculate the effective spread for the symbol that day. The effective spread is a measurement of the cost of getting trades done.

* For each trade the percent effective spread can be calculated as ABS(Price-BAM)/BAM, where BAM is the bid-ask midpoint.
* Find the volume-weighted average effective spread for all the trades you matched (9:30-16:00). It is normally presented in basis point (1/100 of a percent) such as 0.926bp.

Hints:

* There are many ways to match the trades to the appropriate quotes.

1. One is to merge the records and then fill down the quote information to subsequent trades that don’t match quote times exactly.
2. Another is to find the start and end time for each quote and then match the trade times into the appropriate quote. There is a function in the “zoo” R package that fills down missing values.

