CSCI 330 Database Systems
Fall 2016
Assignment 3
Total Point: 50
Due: Tuesday, Nov 29 (11:59 pm)

Introduction

The purpose of this assignment is to do some simple data mining of the johnson330 database.

Problem Statement

Compare the stocks in the johnson330 database against other stocks in the same industry. Store the comparison data ***in your own database (*****database name: *your username)***.

Detailed Description

Industry Groups and Trading Dates

An examination of the database reveals that there are ten industry groups, each containing a number of stocks (tickers):

```
select Industry, count(distinct Ticker) as TickerCnt
  from Company natural join PriceVolume
  group by Industry
  order by TickerCnt DESC, Industry;
```

```
+----------------------------+-----------+
| Industry                   | TickerCnt |
+----------------------------+-----------+
| Financials                 |        80 |
| Consumer Discretionary     |        78 |
| Information Technology     |        66 |
| Industrials                |        60 |
| Health Care                |        48 |
| Energy                     |        39 |
| Consumer Staples           |        38 |
| Utilities                  |        34 |
| Materials                  |        29 |
| Telecommunications Services |         7 |
+----------------------------+-----------+
```

By doing a further examination of the Telecommunications Services group (chosen mostly for its small size), we can determine the constituent stocks and the range of dates for which data is available for that stock:

```
select Ticker, min(TransDate), max(TransDate), count(distinct TransDate)
  from Company natural left outer join PriceVolume
  where Industry = 'Telecommunications Services'
  group by Ticker
  order by Ticker;
```

```
+--------+----------------+----------------+---------------------------+
| Ticker | min(TransDate) | max(TransDate) | count(distinct TransDate) |
+--------+----------------+----------------+---------------------------+
| AMT    | 1998.02.27     | 2014.08.18     |                      4145 |
| CTL    | 1987.11.05     | 2014.08.18     |                      6751 |
| FTR    | 1990.03.26     | 2014.08.18     |                      6149 |
| PCS    | NULL           | NULL           |                         0 |
| S      | 1985.01.02     | 2014.08.18     |                      7470 |
| T      | 1985.01.02     | 2014.08.18     |                      7470 |
| VZ     | 1985.01.02     | 2014.08.18     |                      7470 |
| WIN    | 2005.02.09     | 2014.08.18     |                      2397 |
+--------+----------------+----------------+---------------------------+
```

We want to compare each stock in an industry group with the group as a whole. Each stock will be compared to a "basket" of stocks which consists of other stocks in the industry group.

For the Telecommunication Services group, we eliminate PCS since it has no data. Using the remaining stocks, we first compare the performance of AMT with the basket [CTL, FTR, S, T, VZ, WIN]. Then we compare CTL with [AMT, FTR, S, T, VZ, WIN], and so forth.

Trading Intervals

Each comparison is to be evaluated for a number of disjoint time intervals, called trading intervals. For this exercise, the length of a trading interval will be approximately 60 consecutive trading days, that is, 60 consecutive rows from the PriceVolume table, not 60 calendar days.

We want to have at least two trading intervals for each stock. To allow for some missing data, we will only consider those stocks that have at least 2.5 trading intervals, 150 trading days, of data.

From the above data for the Telecommunications Services group, we would omit only PCS, which has no data. The remaining stocks all have the required 150 trading days of data.

We can modify our query to return just the stocks that we will include in the analysis of this group:

```
select Ticker, min(TransDate), max(TransDate),
       count(distinct TransDate) as TradingDays
  from Company natural join PriceVolume
  where Industry = 'Telecommunications Services'
  group by Ticker
  having TradingDays >= 150
  order by Ticker;
```

```
+--------+----------------+----------------+-------------+
| Ticker | min(TransDate) | max(TransDate) | TradingDays |
+--------+----------------+----------------+-------------+
| AMT    | 1998.02.27     | 2014.08.18     |        4145 |
| CTL    | 1987.11.05     | 2014.08.18     |        6751 |
| FTR    | 1990.03.26     | 2014.08.18     |        6149 |
| S      | 1985.01.02     | 2014.08.18     |        7470 |
| T      | 1985.01.02     | 2014.08.18     |        7470 |
| VZ     | 1985.01.02     | 2014.08.18     |        7470 |
| WIN    | 2005.02.09     | 2014.08.18     |        2397 |
+--------+----------------+----------------+-------------+
```

We see that there are varying time ranges of data available for the different stocks. We want to compare data for the same time periods. In order to do this, we want to examine stock data in the range `max(min(TransDate))` to `min(max(TransDate))`. When examining the data above, we see that the desired date range is 2005.02.09 to 2014.08.18. (Even though these are the available dates for WIN, there is no guarantee that the dates will be the dates for a single stock.)

We can revise our previous query to filter for dates in the desired range:

```
select Ticker, min(TransDate), max(TransDate),
       count(distinct TransDate) as TradingDays
  from Company natural join PriceVolume
  where Industry = 'Telecommunications Services'
    and TransDate >= '2005.02.09' and TransDate <= '2014.08.18'
  group by Ticker
  having TradingDays >= 150
  order by Ticker;
```

```
+--------+---------------+---------------+-------------+
| Ticker | min(TransDate) | max(TransDate) | TradingDays |
+--------+---------------+---------------+-------------+
| AMT    | 2005.02.09     | 2014.08.18     |        2397 |
| CTL    | 2005.02.09     | 2014.08.18     |        2397 |
| FTR    | 2005.02.09     | 2014.08.18     |        2397 |
| S      | 2005.02.09     | 2014.08.18     |        2397 |
| T      | 2005.02.09     | 2014.08.18     |        2397 |
| VZ     | 2005.02.09     | 2014.08.18     |        2397 |
| WIN    | 2005.02.09     | 2014.08.18     |        2397 |
+--------+---------------+---------------+-------------+
```

In this case, there are the same number of trading days for all stocks in the group. We have 39 (= 2397/60) complete trading intervals with 57 days left over at the end. In general, the number of trading days may differ from stock to stock. This occurs due to missing or additional trading days for individual stocks. In that case, we use the minimum of TradingDays over all the Tickers to compute the number of trading intervals. While there may be some misalignment of data points due to missing data, for simplicity, we will use the calendar dates in the first (alphabetically) ticker to determine the first and last days of each trading interval.

In this example, we can obtain the AMT data via the query:

```
select P.TransDate, P.openPrice, P.closePrice
  from PriceVolume P
  where Ticker = 'AMT' and TransDate >= '2005.02.09'
    and TransDate <= '2014.08.18';
```

Alternatively, we can obtain the data for all Tickers in the Industry by:

```
select P.TransDate, P.openPrice, P.closePrice
  from PriceVolume P natural join Company
  where Industry = 'Telecommunications Services'
    and TransDate >= '2005.02.09' and TransDate <= '2014.08.18'
  order by TransDate, Ticker;
```

This last query will give you the data for all tickers in the given industry with all PriceVolume data for a given date returned together.

Since we are using AMT to determine the dates for our trading intervals, the intervals for the Telecommunications Services industry will be:

Interval	Day Number	Start Date	Day Number	End Date
1	1	2005.02.09	60	2005.05.05
2	61	2005.05.06	120	2005.08.01
...				
6	301	2006.04.20	360	2006.07.14
7	361	2006.07.17	420	2006.10.09
...				
39	2281	2014.03.04	2340	2014.05.28

These dates are determined by counting the trading days for AMT. The first interval begins on 2005.02.09 and ends on 2005.05.05, which are the first and 60th days of trading for AMT.

Note that there is a gap between the end date for the 6th interval and the start date for the 7th interval. These dates are the 360th and 361st trading days for AMT. However, for some other stocks, there might be additional trading occurring during the skipped days. To address this problem, we will use the following rules:

1. For a given ticker and a given interval, the first day of the interval is the first trading day occurring on or after the first day as determined by the first ticker.
2. For a given ticker and a given interval other than the last interval, the last day of the interval is the last trading day occurring before the first day of the next interval as determined by the first ticker.
3. For a given ticker, the last day of the last interval is the last trading day occurring on or before the last of the interval as determined by the first ticker.

For example, if there were trades on 2006.07.15 and 2006.07.16 for some Telecommunications Services ticker, then 2006.07.16 would be considered the last day of the interval for that ticker.

Ticker and Industry Returns

For a given industry group, we will find some number, p, of time intervals in a similar manner to that illustrated above for the Telecommunications Services group, where $p = 39$. For each ticker in an industry group, we will generate p comparisons, one for each of the p time divisions.

We compute the comparison for ticker X in industry group Y in a given interval as follows: On the first trading day for the given stock in the given interval (see rules above for definition of "first trading day"), we invest D dollars in stock X, buying our shares at the opening price on that day. We will call the opening price *openPrice*. We assume that we can purchase fractional shares, since the number of shares bought will be *D/openPrice* and is unlikely to be an integer.

On the last trading day of the given interval (again, see rules) we sell all our shares at the closing price for that day, which we will call *closePrice*. Our percent gain, as a fraction, is

$$tickerReturn = \frac{closePrice * (D/openPrice) - D}{D} = \frac{closePrice}{openPrice} - 1.$$

Note that the *tickerReturn*, which can be negative, does not depend on the *D*, the amount invested.

Now, assuming there are *m* stocks in industry group Y with $m \geq 2$, we buy a "basket" of stocks by invest $D/(m-1)$ dollars in each stock in the industry, except for ticker X. We buy the stocks at the opening price on the first trading day for each ticker, same as we did for the ticker X. We sell all of these stocks at the closing price for the last trading day for each ticker. If D' is the dollar amount obtained from the sales, then the industry gain, as a fraction, is $(D' - D)/D$. You should work out the algebra for yourself to verify that this fraction is also independent of the investment D. You should find that

$$industryReturn = [\frac{1}{m-1} \cdot \sum_{k=1}^{m-1} \frac{closePrice_k}{openPrice_k}] - 1$$

where $openPrice_k$ and $closePrice_k$ are the opening price on the first trading day of the interval and the closing price on the last trading day of the interval for the k^{th} ticker in the comparison group.

Before computing these ratios, stock prices must be adjusted for splits. Use the same rules for detecting and adjusting splits that were used in the previous assignment.

Note that these purchases and sales for computing gains ignore transaction costs.

Output

In addition to reading data from the `johnson330` you used in the prior assignment, you will need to write your output to your `username` database. In order to connect to that database you will need another connection parameters file that looks like:

```
dburl=jdbc:mysql://mysql.cs.wwu.edu/username
user=username
password=password
```

You should be able to read and write this database.

Your program **must** use two files named "`readerparams.txt`" and "`writerparams.txt`" as the default files for the connection parameters for the `johnson330` and `username` database, respectively.

In the database `username`, create a table called Performance (note the capital P) with the following six attributes:

```
Industry CHAR(30)
```

```
Ticker CHAR(6)
StartDate CHAR(10)
EndDate CHAR(10)
TickerReturn CHAR(12)
IndustryReturn CHAR(12)
```

For each comparison, you will generate one row in the Performance table:

- Industry is the name of the industry group under consideration.
- Ticker is the ticker symbol which is being compared to other stocks in its industry.
- The StartDate and EndDate are the calendar dates for the alphabetically first ticker in the group. These dates will be the same for all tickers in the group, even if the first and last trading day for the given stock are not the same as the first and last trading days for the alphabetically first stock. (This condition can occur.)
- The TickerReturn is the *tickerReturn* as described above.
- The IndustryReturn is the *industryReturn* as described above.

When computing the returns you should use **Java doubles** for the computations.

Since the TickerReturn and IndustryReturn attributes are character strings in the database and the returns are computed using Java doubles, you will need to convert the doubles to Strings in order to store them in the database. If `r` is the variable containing the return you should use `String.format("%10.7f", r)` to do the conversion.

Sample Output

You can query your final Performance table using the command-line interface or the MySQL Workbench. Here are some sample queries:

```
select Industry, count(*)
  from Performance
  group by Industry;
```

```
+----------------------------+----------+
| Industry                   | count(*) |
+----------------------------+----------+
| Consumer Discretionary     |      312 |
| Consumer Staples           |      684 |
| Energy                     |      429 |
| Financials                 |     2400 |
| Health Care                |      960 |
| Industrials                |      660 |
| Materials                  |     1073 |
| Telecommunications Services |      273 |
| Utilities                  |      561 |
+----------------------------+----------+
```

```
select *
  from Performance
  where Industry = 'Telecommunications Services'
  order by StartDate, Ticker
  limit 10;
```

```
+----------------------------+--------+------------+------------+--------------+----------------+
| Industry                   | Ticker | StartDate  | EndDate    | TickerReturn | IndustryReturn |
+----------------------------+--------+------------+------------+--------------+----------------+
| Telecommunications Services | AMT    | 2005.02.09 | 2005.05.05 | -0.0381005   | -0.0822529     |
| Telecommunications Services | CTL    | 2005.02.09 | 2005.05.05 | -0.0804185   | -0.0751999     |
| Telecommunications Services | FTR    | 2005.02.09 | 2005.05.05 | -0.0764706   | -0.0758579     |
| Telecommunications Services | S      | 2005.02.09 | 2005.05.05 | -0.0870286   | -0.0740982     |
| Telecommunications Services | T      | 2005.02.09 | 2005.05.05 | -0.0401297   | -0.0819147     |
| Telecommunications Services | VZ     | 2005.02.09 | 2005.05.05 | -0.0602740   | -0.0785573     |
| Telecommunications Services | WIN    | 2005.02.09 | 2005.05.05 | -0.1491961   | -0.0637370     |
| Telecommunications Services | AMT    | 2005.05.06 | 2005.08.01 |  0.2985755   |  0.0744118     |
| Telecommunications Services | CTL    | 2005.05.06 | 2005.08.01 |  0.1301948   |  0.1024753     |
| Telecommunications Services | FTR    | 2005.05.06 | 2005.08.01 |  0.0403162   |  0.1174551     |
+----------------------------+--------+------------+------------+--------------+----------------+
```

```
select *
  from Performance
  order by TickerReturn - IndustryReturn DESC
  limit 6;
```

```
+-------------+--------+------------+------------+--------------+----------------+
| Industry    | Ticker | StartDate  | EndDate    | TickerReturn | IndustryReturn |
+-------------+--------+------------+------------+--------------+----------------+
| Materials   | TIE    | 2013.01.02 | 2013.03.28 | 738.3939394  |  0.0186217     |
| Financials  | C      | 2011.04.05 | 2011.06.29 |  8.3049327   | -0.0501802     |
| Financials  | ETFC   | 2010.04.23 | 2010.07.19 |  6.1256831   | -0.1447643     |
| Financials  | AIG    | 2009.05.11 | 2009.08.04 |  5.8979592   |  0.0961546     |
| Health Care | THC    | 2012.09.25 | 2012.12.20 |  4.2839117   |  0.0186531     |
| Financials  | FITB   | 2009.02.12 | 2009.05.08 |  3.1014493   |  0.4667796     |
+-------------+--------+------------+------------+--------------+----------------+
```

```
select Industry, Ticker, count(*) as CompareCount from Performance group by
Industry, Ticker order by CompareCount, Ticker desc limit 10;
```

```
+----------------------------+--------+--------------+
| Industry                   | Ticker | CompareCount |
+----------------------------+--------+--------------+
| Telecommunications Services | AMT    |           39 |
| Telecommunications Services | CTL    |           39 |
| Telecommunications Services | FTR    |           39 |
| Telecommunications Services | S      |           39 |
| Telecommunications Services | T      |           39 |
| Telecommunications Services | VZ     |           39 |
| Telecommunications Services | WIN    |           39 |
| Materials                  | AA     |           37 |
| Materials                  | APD    |           37 |
| Materials                  | ARG    |           37 |
+----------------------------+--------+--------------+
```

Here is a progress report from the java code which might be useful for you for debugging:

```
Reader connection established.
Writer connection established.
10 industries found
Consumer Discretionary
Consumer Staples
Energy
Financials
Health Care
Industrials
Information Technology
Materials
Telecommunications Services
Utilities

Processing Consumer Discretionary
78 accepted tickers for Consumer Discretionary(2013.06.19 - 2014.08.18), 294 common dates

Processing Consumer Staples
38 accepted tickers for Consumer Staples(2009.02.11 - 2013.06.07), 1088 common dates

Processing Energy
39 accepted tickers for Energy(2011.12.12 - 2014.08.18), 674 common dates

Processing Financials
80 accepted tickers for Financials(2007.06.14 - 2014.08.18), 1808 common dates

Processing Health Care
48 accepted tickers for Health Care(2009.08.21 - 2014.06.30), 1222 common dates

Processing Industrials
60 accepted tickers for Industrials(2011.10.13 - 2014.08.18), 715 common dates

Processing Information Technology
Insufficient data for Information Technology => no analysis

Processing Materials
29 accepted tickers for Materials(2005.08.11 - 2014.08.18), 2270 common dates

Processing Telecommunications Services
7 accepted tickers for Telecommunications Services(2005.02.09 - 2014.08.18), 2397 common dates

Processing Utilities
33 accepted tickers for Utilities(2010.06.16 - 2014.08.18), 1051 common dates

Database connections closed
```

Constraints

1. Similar to Assignment 2, you must use PreparedStatements when you are using SQL statements where values are filled in. As described in class, this is basic good practice for avoiding SQL injection attacks. (This is a HUGE security issue.)
2. As stated above, you must use doubles for all computations when working with prices and when computing returns. Do not round the numbers or convert the values to Strings until you are ready to store the values in the data base.

Hints and Comments

1. You will have to create the table in the `username` database. Your program will need to clear the Performance table before you store the new data. This will prevent you from having a combination of old and new data in your Performance table. There are two ways to do this:
   a) Create the Performance table once. When your program starts, delete all rows from the table.
   b) Drop and create the table anew each time you run your program. You can use this command to drop the table: `drop table if exists Performance;`. The "`if exists`" in the command prevents this command from raising an error if the table does not already exist.
2. Like Assignment 2 you do not need to decompose dates. You can handle dates as Strings.
3. Some of the data in the database is not "clean". In particular, the incredible return for TIE from 2013.01.02 to 2013.03.28 is based on bad data, not actual performance.

Design and Good Style

Similar to previous assignments, we will be grading your program for design and good style as well as correctness. In particular, for this assignment we will be looking to see that the work is divided into multiple functions and not all performed in a single large function.

Submission Instructions

- You only need to submit the source code.
- The file name which contains main method should be Assignment3.java.
- If you have more than one java files:
  - Please keep all java files in a single folder.
  - Folder name should be your last name.
  - Zip the folder and name it Assignment3.zip.
- Upload the Assignment3.java/ Assignment3.zip on canvas.

- In addition to submitting your source, you must leave the Performance table intact in your username database. We will read your data to check the correctness of your computations.

Grading (Total Points: 50)

Your program will be graded in the following way:

- 40 – The program correctly executes the analysis and inserts the correct answers into the Performance table.
- 4 – Program obeys constraints mentioned above.
- 5 – Program has good design and style.
- 1 – File name follows the rules mentioned above.

Late Policy

- Max 2 days after deadline are allowed
    - 1 day late: maximum 70% point
    - 2 days late: maximum 60% point

Academic Dishonesty

- Copying files from someone else and claiming they are your own is plagiarism.
- Providing files that you created to another student or being party to such actions also amounts to academic dishonesty.