

# Week 4 - Project Update

## Independent Study

Michael Rechenthin  
Loyola University Chicago, Chicago, IL 60640  
mrechenthin@gmail.com

### Abstract

My intention with this short paper is to report my progress on my summer project.

### I. INTRODUCTION

OVERALL I am happy with the overall progress and direction of my summer project. The project has changed slightly from the original proposal that I sent you on May 16. However, I feel that the changes are appropriate and will improve the overall project. In this report, I will explain what I have achieved so far, and what there is left to do. First I will give a general overview of where I am at in my project, and then I will give you a more focused breakdown.

As I mentioned in my proposal the whole reason why I am interested in this project, is the potential to find opportunity in investigating strongly correlated stocks. If an individual (arbitrageur) possesses information not reflected in the stock price, then a profit can be made. Although correlations do not necessarily imply causality, correlations can help supply a skilled trader with ideas.

### II. PROJECT MANAGEMENT

By looking at the first two pages in the appendix, it can be seen that I am roughly 55% complete with my project. I have completed the warehousing of quotes into MySQL and figuring out correlations between different symbols. Yet to do, is how to host the information via web-pages. Furthermore, I intend to keep you informed of my progress weekly.

### III. DATABASE

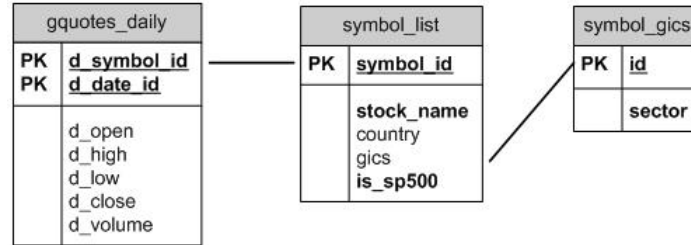
I decided to use MySQL with a relational database design, which is consistent with my proposal. Figure1 shows the layout of the database.

#### A. Table Design

A description of the tables is shown below:

- Gquotes\_daily, contains standard historical daily data beginning on Jan 3, 2000 up through the current day.
- Symbol\_list, contains the stock symbol, along with the full name. It also contains the individual companies industry classification code, known as the *Global Industry Classification Standard (GICS)*. A fifth field (is\_sp500) is a SmallInt that is a '1' if the symbol is part of the *Standard & Poors 500*, and a '0' if it is not.
- Symbol\_gics, contains the id code and full *GICS* name.

Figure 1. Database Layout



### B. Relational Design

There are several reasons why I decided to use a relational design database, over a object-oriented design database. My first and main reason is compatibility with existing programs. Two statistical packages, namely SAS and SPSS's Clementine can import data via a relational database. These are two programs that I will like to use to explore the data further.

A second reason is the transparency of the stored data. With the object-oriented database design the data is in objects which would require me to employ a program for every query that I wish to employ. With the MySQL Query Browser<sup>1</sup>, and the relation design, I can easily write queries with SQL. This allows me to easy test and explore my data.

## IV. QUOTE DOWNLOADING

### A. Main Screen

To download quotes from the quote server I designed a graphical user interface. This can be seen in figure 2. A description of buttons is shown below:

- *Static Update* retrieves a list of symbols from the *symbol\_list* table and request the historical data from the quote server for those particular symbols.
- *Update S&P 500 List*, retrieves the current days list of symbols contained in the Standard & Poors 500 list and updates the *symbol\_list* table in the database. This information is retrieved directly from the Standard & Poors website. This is particularly important since the list changes occasionally (especially with all the turmoil in the economy at the present time).
- *Add Miscellaneous Symbols*, allows the user to add and delete stocks that are not part of the Standard & Poors 500. See figure 3.

### B. Symbol Maintenance

To facilitate the adding of stocks not already included in the S&P 500, a user interface was needed. This can be seen in figure 3. This screen can be reached by clicking on *Add Miscellaneous Symbols* on the main interface (see figure 2). A nice feature is that the user can copy and paste large groups of stocks directly into the program for batch adding. To delete a symbol the user selects the symbols to delete and then clicks *Delete Symbol*.

## V. COMPUTING CORRELATIONS

As noted in my proposal I wanted to allow a user to request the top correlated stocks in the S&P when given a symbol. For this problem, I decided to place the logic within MySQL procedures rather than Java client side programming. Stored procedures computation within MySQL is slower then computation within Java [1]. However, the difference is greatly narrowed when taking into consideration the reduced network traffic that is required by not having to transfer the quote data across the server<sup>2</sup>. Furthermore, having the logic within the MySQL database increases the portability of the application[2]. The stored procedure code can be found in the appendix.

<sup>1</sup>MySQL Query Browser is a graphical tool provided by the makers of MySQL for creating, executing and optimizing queries in a graphical environment. More information can be found at: <http://dev.mysql.com/doc/query-browser/en/>

<sup>2</sup>As of June 11, the gquotes\_daily table has 1.3 million rows of data, or around 300 megabytes of data. This number will probably be doubled within the next couple weeks.

Figure 2. GUI for Price Uploader

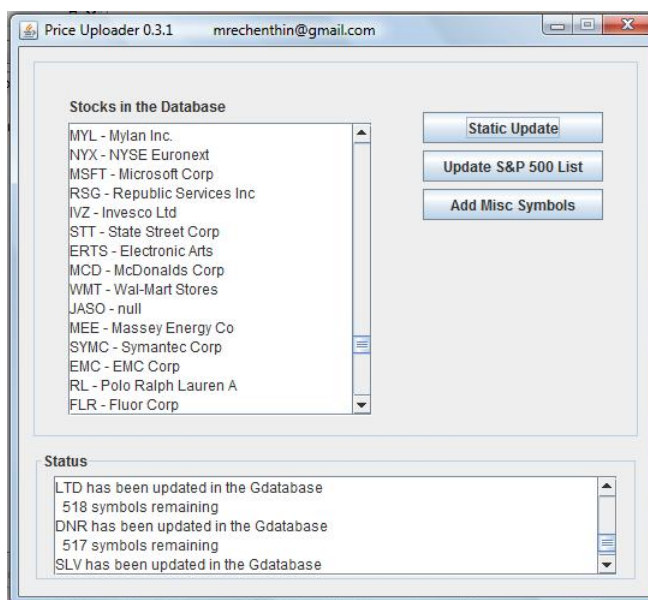
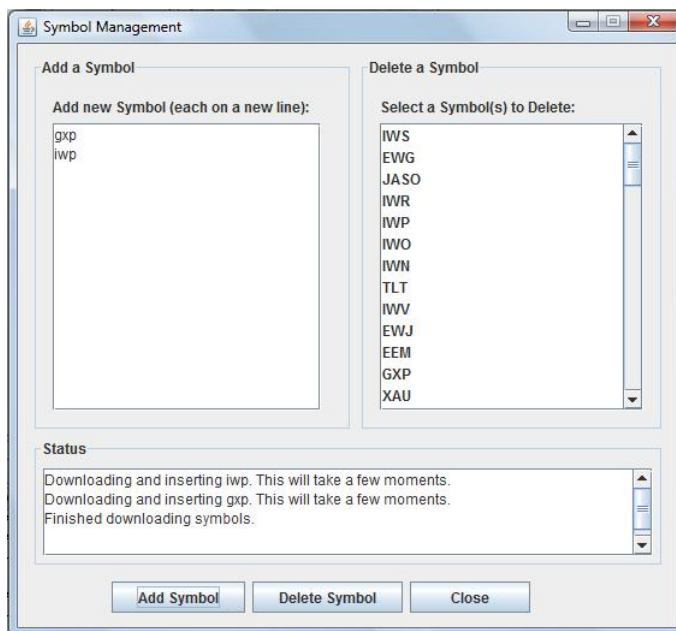


Figure 3. Symbol Management



## VI. WEB INTERFACE

Mock-ups of the web interface can be seen in figures 4 and 5. These pages are non-functioning, and are for display only at the moment. On May 16th you recommended that I use Google Web Toolkit. This is something I intend to explore further this week.

## VII. PROBLEMS

### A. Quote Provider

I decided to scrap my original idea of using Interactive Brokers. I was able to connect and retrieve quotes, but IB historical quote downloading were onerous. First the daily data was limited to only a couple months. Second I could only download 60 stocks every ten minutes before the server timed out. For these two reasons I decided to look for something else.

Figure 4. Web Interface Mock-up

Figure 5. Web Interface Results Mock-up

_symbolK	_symbolY	_tempCorr	_days_back
ibm	WYN	0.741762	5000
ibm	NTRS	0.733171	5000
ibm	CBS	0.728019	5000
ibm	WMB	0.725205	5000
ibm	DTV	0.719479	5000
ibm	VIA.B	0.718786	5000
ibm	IVZ	0.670754	5000
ibm	WU	0.669653	5000
ibm	BK	0.658983	5000
ibm	T	0.655882	5000
ibm	SCHW	0.6535	5000
ibm	GT	0.651467	5000
ibm	ORCL	0.646734	5000
ibm	HPQ	0.639547	5000
ibm	PKI	0.636967	5000
ibm	GOOG	-0.624186	5000
ibm	DIS	0.615497	5000
ibm	HON	0.605815	5000
ibm	ADP	0.603646	5000
ibm	CPWR	0.593948	5000
ibm	WAT	0.593644	5000

Ultimately I decided to use Google Finance. Furthermore, I also wrote a class that allows me to use Yahoo Finance. Although neither of these allows for downloading of intra-day data, I feel there is enough to be explored for this project using daily data.

### B. Splits and Dividends

In the next couple weeks I plan on exploring a better way of dealing with stock splits and the issuance of dividends. For example, a company might issue a 2 for 1 stock split on its \$100 share price. This causes the shareholder to have 2 shares worth \$50 a piece. Therefore, historical prices need to be adjusted to account for this change. Otherwise, the stock would seem

to have a 50% drop overnight, which would be incorrect.

Secondly, a dividend being issued would have a similar effect on the price of a stock. If a shareholder receives a dividend representing an annualized yield of 5%, or \$1 a share per quarter, this dividend should be subtracted from historical prices to prevent a misleading drop in share price.

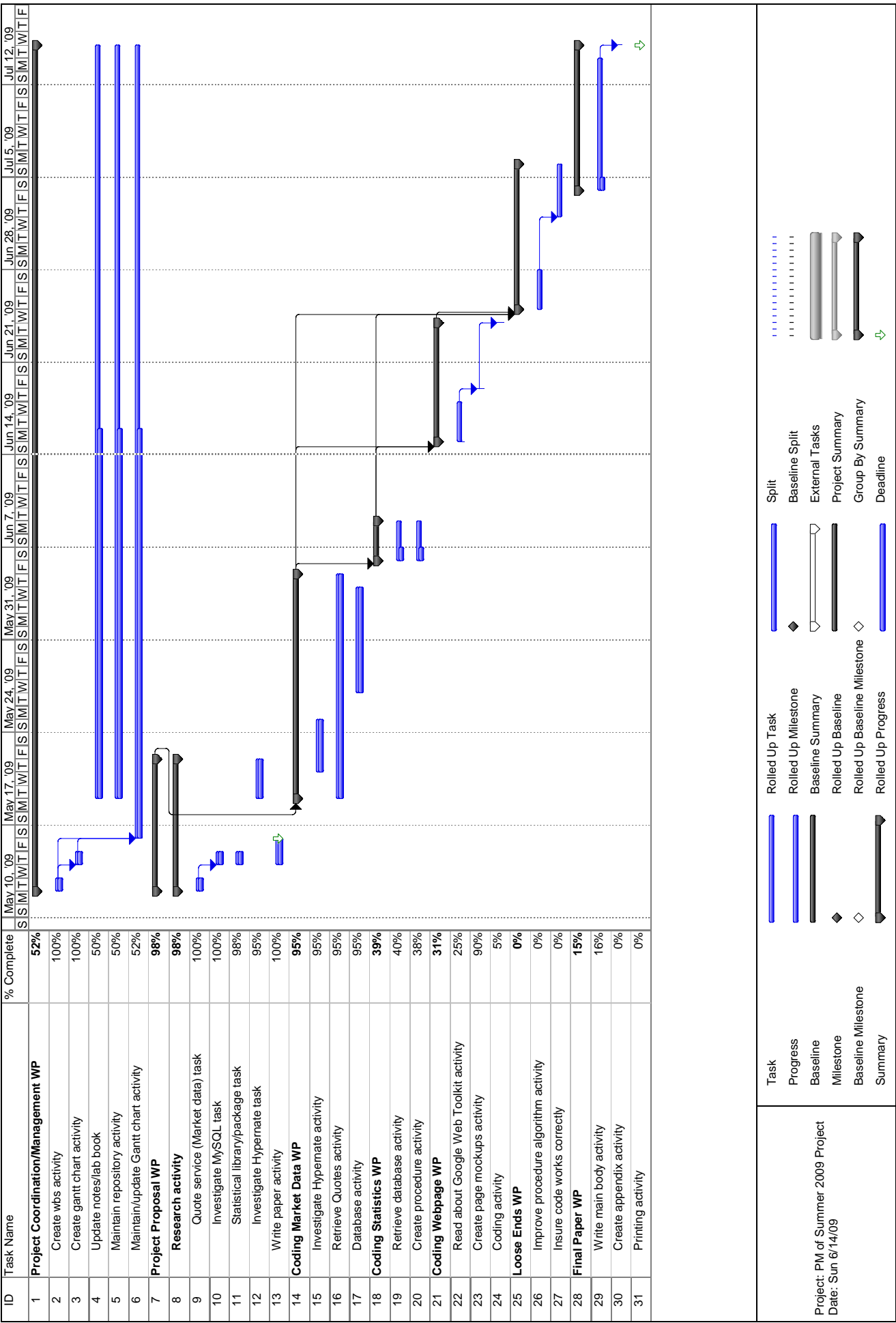
I'm currently working on an efficient solution to both of these problems. The most inefficient way is to completely download the historical information for each stock each time the user updates the database. A second more efficient way is to check prices against the quote server against a date several days back to check for changes. A third, and more efficient solution, is to retrieve a list of dividends and split adjustments from the Standard & Poors.

### *C. Web Interface*

I find web programming tedious and confusing, especially when using JSP. Perhaps I was not using the correct tools when I have done it before.

### REFERENCES

- [1] Guy Harrison and Steven Feuerstein. *MySQL Stored Procedure Programming*. O'Reilly Media Inc., 2006.
- [2] Peter Gulutzan. MySQL 5.0 Stored Procedures. <http://dev.mysql.com/tech-resources/articles/mysql-storedprocedures.pdf>.



**PM of Summer 2009 Project  
Loyola University Chicago**

as of Sun 6/14/09

**Dates**

Start:	Mon 5/11/09	Finish:	Wed 7/15/09
Baseline Start:	NA	Baseline Finish:	NA
Actual Start:	Mon 5/11/09	Actual Finish:	NA
Start Variance:	0 days	Finish Variance:	0 days

**Duration**

Scheduled:	97.5 days?	Remaining:	43.51 days?
Baseline:	0 days?	Actual:	53.99 days
Variance:	97.5 days?	Percent Complete:	55%

**Work**

Scheduled:	0 hrs	Remaining:	0 hrs
Baseline:	0 hrs	Actual:	0 hrs
Variance:	0 hrs	Percent Complete:	0%

**Costs**

Scheduled:	\$0.00	Remaining:	\$0.00
Baseline:	\$0.00	Actual:	\$0.00
Variance:	\$0.00		

**Task Status**

Tasks not yet started:	5
Tasks in progress:	21
Tasks completed:	5
Total Tasks:	31

**Resource Status**

Work Resources:	0
Overallocated Work Resources:	0
Material Resources:	0
Total Resources:	0

DELIMITER \$\$

```

DROP FUNCTION IF EXISTS `equity_data`.`getCorrelation`$$
CREATE DEFINER=`root`@`localhost` FUNCTION `equity_data`.`getCorrelation`(symbolX char(10),
                                                                    symbolY char(10),
                                                                    days_back int) RETURNS float
BEGIN
-- -----
-- Michael Rechenthin
-- June 2009
--
-- finds the correlation of two symbols,
-- symbolX & symbolY, going back
-- days_back number of days
-- Returns the pearsons correlation coefficient
-- -----
declare num          int      default 0; -- number of rows
declare ccloseX      float    default 0; -- the closing price
declare ccloseY      float    default 0; -- the closing price

declare std_devX     float    default 0; -- standard deviation
declare std_devY     float    default 0; -- standard deviation

declare meanX        float    default 0; -- mean
declare meanY        float    default 0; -- mean

declare popcovXY     float    default 0; -- population covariance

declare corr         float    default 0; -- pearsons correlation coefficient

-- cursor via X
declare cur_ccloseX cursor for
  select d_close from gquotes_daily
  where d_symbol_id=symbolX and
  d_date_id > date_sub(curdate(),interval days_back day);

-- cursor via Y
declare cur_ccloseY cursor for
  select d_close from gquotes_daily
  where d_symbol_id=symbolY and
  d_date_id > date_sub(curdate(),interval days_back day);

declare continue handler for not found set ccloseX=-1;

-- initializing
-- -----

-- get the mean & standard deviation for X going back some number of days
select AVG(d_close), STDDEV_POP(d_close) from gquotes_daily
  where d_symbol_id=symbolX and
  d_date_id > date_sub(curdate(),interval days_back day)
  into meanX, std_devX;

-- get the mean & standard deviation for Y going back some number of days
select AVG(d_close), STDDEV_POP(d_close) from gquotes_daily

```



```
where d_symbol_id=symbolY and
d_date_id > date_sub(curdate(),interval days_back day)
into meanY, std_devY;

OPEN cur_ccloseX;
OPEN cur_ccloseY;
  aloop:LOOP
    fetch cur_ccloseX into ccloseX;
    fetch cur_ccloseY into ccloseY;

    if ccloseX=-1 then leave aloop; end if;

    -- to get the true population covariance of x & y we need to divide it by num
    set popcovXY = popcovXY + (ccloseX - meanX)*(ccloseY - meanY);

    set num=num+1;  -- keep track of the total count/rows

  END LOOP aloop;
CLOSE cur_ccloseX;
CLOSE cur_ccloseY;

set popcovXY = popcovXY/num;  -- computing the population covariance

set corr = popcovXY / (std_devX*std_devY);  -- computing the correlation

return(corr);
END;

$$

DELIMITER ;
```

DELIMITER \$\$

DROP PROCEDURE IF EXISTS `equity\_data`.`getTopSP500Correlation`\$\$

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `equity_data`.`getTopSP500Correlation`(  
    IN symbolX char(10),  
    IN days_back int,  
    IN topNum int  
)
```

BEGIN

-- -----

-- Michael Rechenthin

-- June 2009

--

-- Finds the top number (topNum) of correlations between the stock (symbolX) and the  
-- rest of the stocks in the s&p 500 table (sp500\_list)...going back (days\_back)  
-- number of days.

--

-- To do this I am going to create a temporary table, and store the symbols along with  
-- the correlation coefficient.

--

-- \* notice that we are not including the stocks that are not in the s&p500.

-- We want to only include stocks that are actively traded and s&p 500

-- stocks have a strong likelihood of having

-- -----

```
declare symbolY      char(10);          -- symbol y...the temp to compare symbolX with  
declare tempCorr      float  default 0; -- temp correlation for the current symbol  
declare tempCorrAbs   float  default 0; -- absolute value of correlation for cur. symbol  
declare tempHigh      float  default 0; -- the temp high for the current symbol  
declare num           int default 0;     -- counter
```

-- create cursor...no need to include the stock we are requesting for...

-- and only include the stocks that are market as being in the s&p 500 (which is market  
-- with a '1')

declare cur\_symbolList cursor for

```
select symbol_id from symbol_list where symbol_id<>symbolX and is_sp500='1';
```

declare continue handler for not found set symbolY='quit';

-- create a temporary table...using this as storage for the top numbers

-- of symbols that correlate with symbolX

DROP TEMPORARY TABLE if exists tempTopCorrelation;

```
CREATE temporary TABLE tempTopCorrelation (  
    _symbolY      char(10) not null,    -- symbolY  
    _symbolX      char(10) not null,    -- symbolX  
    _tempCorr     float,                -- pearson correlation coefficient  
    _tempCorrAbs  float,                -- absolute value of the pearson corr. coeff.  
    _days_back   int,                 -- the days back to retrieve  
    primary key (_symbolY)  
);
```

open cur\_symbolList;

```
symbolListLoop:LOOP          -- START OF LOOP
```

```
fetch cur_symbolList into symbolY;
```

```
if symbolY='quit' then leave symbolListLoop; end if;
```

```
set tempCorr = getCorrelation(symbolX, symbolY, days_back);
```

```
    set tempCorrAbs = ABS(tempCorr);
    -- insert the appropriate information into the table
    INSERT INTO tempTopCorrelation VALUES (symbolY, symbolX,
                                           tempCorr, tempCorrAbs,
                                           days_back);

    set num = num+1; -- advance counter
END LOOP symbolListLoop;      -- END OF LOOP
close cur_symbolList;

SELECT _symbolX, _symbolY, _tempCorr, _days_back
      FROM tempTopCorrelation order by _tempcorrAbs desc LIMIT 30;
END $$

DELIMITER ;
```