HedgeOverflow

Sean Gaffney, Diarmuid McGonagle, Stephen Davis & Alice Doherty

Design

Homepage:

The Homepage is divided up into three sections. The first section, on the left, consists of basic text of our names. The second section consists of the Title of our project, HedgeOverflow. Below this are three buttons, the first two of which are links to the Losers List and then the Gainers List. The third button links to the Master List, or NYSE and NASDAQ exchanges. The third section, on the right, has five buttons. These include buttons to some sample stocks that we have personally chosen. They are hardcoded in and will not change. They are there for a quick demonstration of our Price Over Time page and for design purposes.

Master List:

The Master List is one page with two exchanges, NYSE and NASDAQ. There is a button on the left hand side to toggle between the two exchanges. The implementation of this includes passing in an ArrayList that includes the exchange data as well as having a variable that sets the name of the exchange. In the queries, the variable of the name of the exchange changes to the new exchange. The top portion of the page also includes buttons back to the homepage (on the right), a title of the exchange (in the middle), and buttons to the Gainers and Losers (on the left).

In the main portion of the screen, we include the data of the stocks. We use a Scroll bar on the right hand side to extend the screen to include the entire list of stocks from that exchange from the stocks.csv file. In the table, the furthermost left value is the name of the stock. Further, this value is also a button that links directly to that stocks Price Over Time page and shows the graph. The next three values are the open price, close price, and change in the price. We calculate the change in price by dividing the open price by the close price and using the Java Math Round function to get to two decimal places.

Losers:

This is a page which shows a list of the 10 stocks with the lowest percentage changes from the most recent opening price, to the most recent closing price. Each row on this page displays the ticker, open price, close price and percentage change of the stocks on the list.

Clicking any of the stock buttons on the list will bring you to that stock's price over time graph.

On the right hand side are 3 additional buttons, which provide the user with the functionality to: return to the home page, view the master list, or view the stocks with the highest percentage changes.

Gainers:

This page is the same as the loser page, except instead of showing a list of the 10 stocks with the lowest percentage changes (from the most recent opening price, to the most recent closing price), it shows a list of the 10 stocks with the highest percentage changes. Like the losers page, each row on this page displays the ticker, open price, close price and percentage change of the stocks on the list.

Again, clicking any of the stock buttons on the list will bring you to that stock's price over time graph.

This page applies the same functionality to the user whereby on the right hand side there are 3 additional buttons, which provide the user with the functionality to: return to the home page, view the master list, or view the stocks with the lowest percentage changes.

Price Over Time:

This page effectively displays the history of the desired stock with the use of an-axis labelled graph. The data points drawn on the graph represent the closing prices of the stock over time. As can be seen by the graph itself, the y-axis represents the price in dollars of the stock, and the x-axis represents the date from which the closing price value is found. The implemented closing price values, which are accessed by using a query, are passed in as a parameter to the method which draws the data points.

In the top-right hand corner, information about the specific stock is displayed, including the name of the stock, the ticker, the sector, industry, and the exchange in which the stock is from.

On the right hand side, outside of the graph, there are 2 buttons: a home page button, and a master list button. As one would expect, clicking the home page button allows the user to return to the home page, and clicking the master list button allows the user to view the master list.

Line and Stock objects:

The file daily_pricesX.csv is loaded in as a table and organized into objects with the readPriceData() function. This method of loading the data was more efficient when tested with the larger datasets, allowing the programing to handle the 100k file easily and the 2gb file given enough time and memory. Each line of data from “daily_pricesX.csv” is used to create a Line Object that has associated ticker, open_price, close_price, adjusted_close, low, high, volume and date variables.

Each Stock object represents each company listed in “stocks.csv” and has an associated ticker, exchange, name, sector and industry variable. Each Stock object also has an ArrayList called pricePoints which contains all the Line objects associated with that Stock company. Stock data is first loaded in as an array of strings using the loadStrings() function. LoadStrings() is used because it allows for simpler code, and it can handle the approximately 6500 stocks in the file.

The Stock objects are initialised using a method called createStocks , with all the Stock objects being created then stored in an ArrayList called stockData. The function first checks if the stock is present in both “stocks.csv” and “daily_pricesX.csv” before calling the initStock function which creates the Stock object with the associated company data and adds it to the ArrayList stockData. createStocks then initialises the pricePoints ArrayList for each Stock object in stockData.

Queries:

The queries sort the data so that only the information needed to be displayed by the page calling it is returned. There are seven different queries: queryTicker, queryPriceOverTime, queryLatestDataPoints, queryLowestPercentage, queryHighestPercentage, queryNASDAQ and queryNYSE.

queryTicker is passed in a ticker and searches the stockData ArrayList<Stock> for a Stock object with the corresponding ticker. Once a match is found the function returns the pricePoints (all the Line objects associated with that stock) for that Stock object.

queryPriceOverTime is called by the Price Over Time page and is passed in the relevant ticker. This function takes the pricePoints for a particular stock and sorts them by date (earliest to latest) and then returns all the closed prices for that stock so that they can be graphed by the Price Over Time page.

queryLatestDataPoints is called by the master list. It takes all the pricePoints for each Stock and sorts them by date (earliest to latest), it then takes the latest data point for each stock and adds it to an ArrayList<Line>. Essentially, this query allows the latest data for each stock to be displayed.

queryLowestPercentage is called by the Lowest Percentage Change page. This function takes the latest data points for each stock(by calling queryLatestDataPoints), calculates the percentage change for each, and sorts them lowest to highest. When calculating the percentage change the function also sets the percentageChange variable for the relevant Line object. This means that when the page is calling it, it's not necessary for the page itself to calculate the percentage change again, but rather it can just print out the percentageChange variable. queryHighestPercentage does the exact same but sorts the data from highest percentage change to lowest using Collections.reverse().

queryNASDAQ and queryNYSE return the all the latest price points for all the stocks on the NASDAQ or NYSE exchange. It does this first by searching all the stocks for the relevant exchange in stockData (from stocks.csv), once a stock matches the exchange needed, the ticker for that stock is taken and then searched for in the ArrayList of latest data points, that data point is then added to an ArrayList<Line> that gets returned once all the stocks are checked.

Two comparators have been written in order to sort the data efficiently. The comparator interface was chosen over the comparable interface in order to allow the data to be sorted in more than one way. The dateComparator sorts the data (Line objects) by date from the earliest date to latest and the percentageComparator sorts the data by the percentageChange variable from lowest to highest. Rather than storing the date associated with each Line as a String/integer a Date type is created using Java's SimpleDateFormat, as a result the data can be sorted much easier.

Functionality

Buttons:

To implement the buttons functionality, we created a Widgets class. This class was copied from our Week 6 assignments and then changed for our needs. We added extra attributes, such as current screen Number and added new methods. The most prominent of these new methods was the changeToNewPage() method which accepted three parameters, an X (always mouseX), a Y (always mouseY), and the new screen number to be switched to. If the mouseX and mouseY were within the coordinates of the buttons, the global variable screenNumber would switch to this number and the draw() function in the main would draw the new screen.

We called all these functions in one function, activateAllButtons() in the mousePressed() portion of the main tab. This function checked to see the screenNumber, and then only allowed the buttons on this screen to be able to be pressed so that buttons from other pages could not be activated by accident. We also made some changes to the changeToNewPage() method, making a special changeToPotPage() (price over time) that auto changed the screen to the Price Over Time page and also accepted a parameter to include which ticker to pass in. This way we could activate every POT button with one function and just change the global variable currentTicker to reflect which stock we wanted to inspect.

ScrollBar:

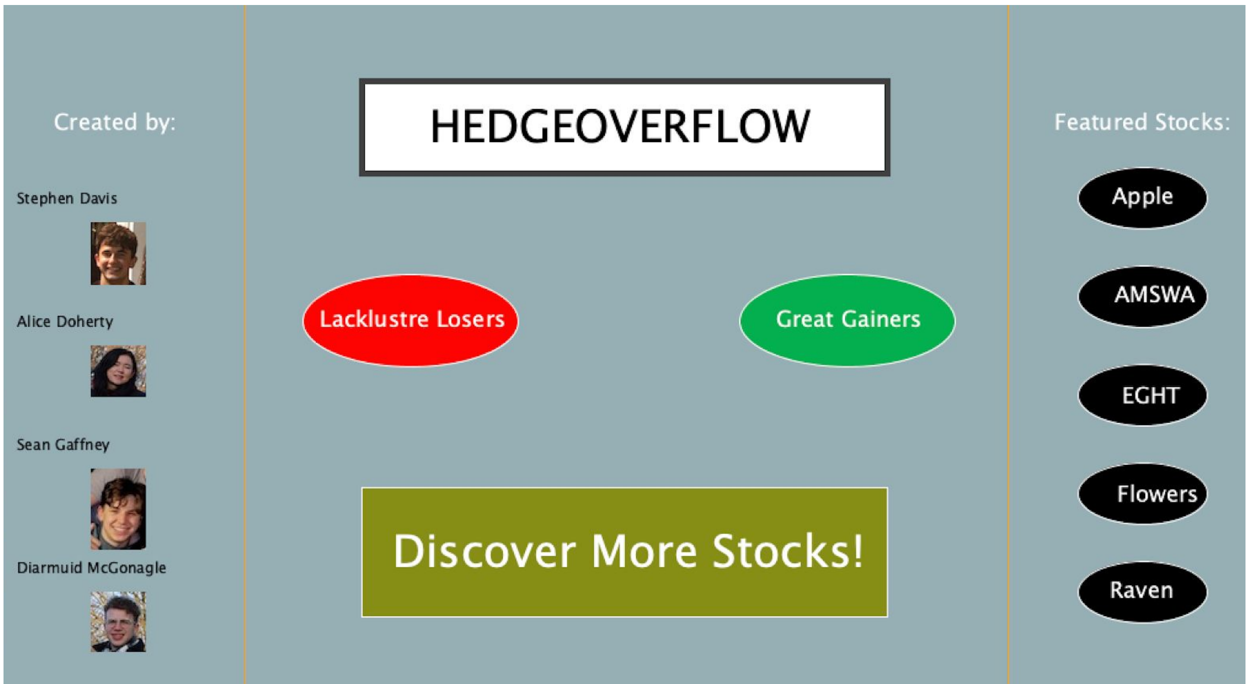
Our use of a scroll bar is one way in which our solution goes beyond the original project brief. We added a scroll bar to the master list because we felt it would be more user-friendly and that it added a bit of flavour to our project. Instead of us having screen after screen which simply displays information, we wanted to include something different which would add some flair to our project, hence the scroll bar.

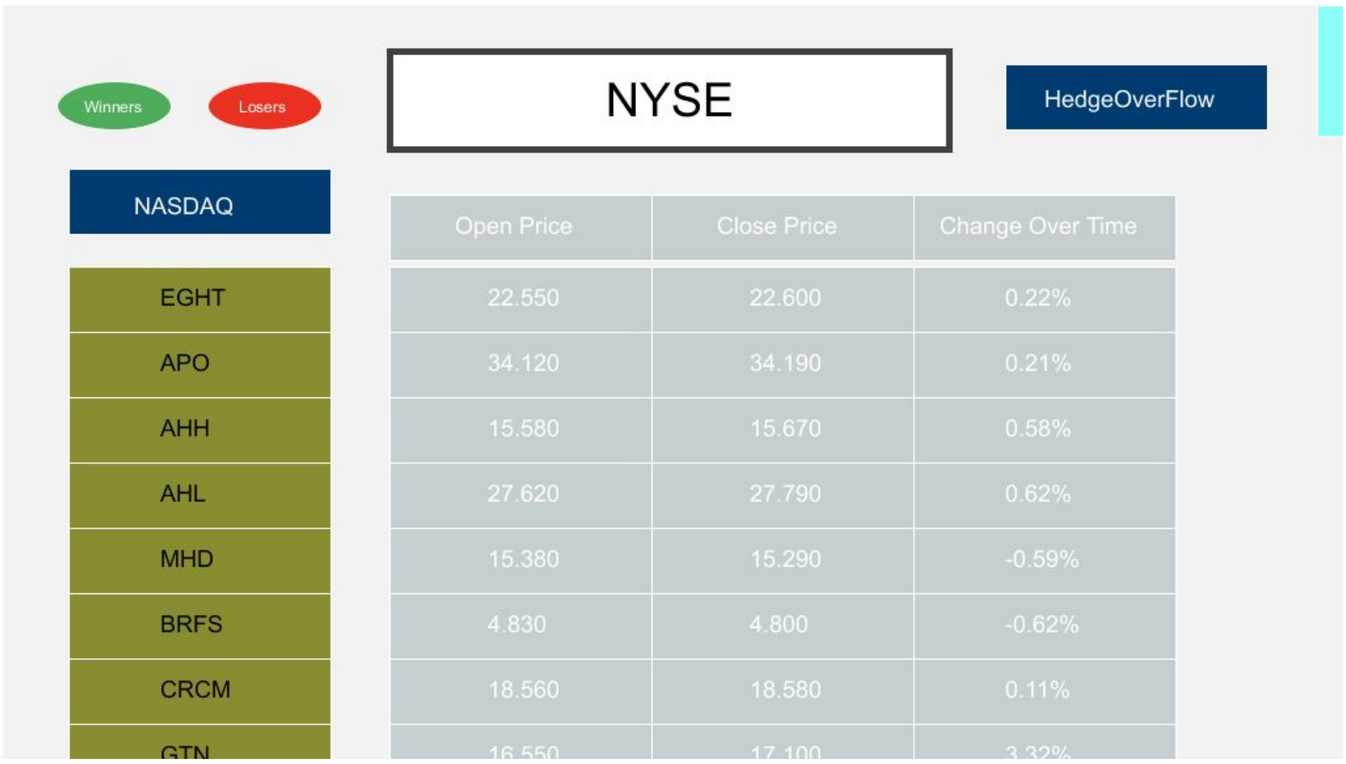
By allowing the user to scroll up and down through the master list, we felt that the user would feel like (s)he has more control over where they can go to next. The scroll bar essentially empowers the user and also allows him/her to view all of the stocks in the desired exchange without having to endlessly switch back and forth between different screens.

Organisation

The group was split in half, with Stephen Davis and Diarmuid McGonagle working on the front end and Sean Gaffney and Alice Doherty working on the backend. Stephen was responsible for the Losers, Gainers and Price Over Time screens, and the scroll bar while Diarmuid worked on the Homepage, Master List and the button functionality. Sean was responsible for how the data was loaded in and stored. Alice worked on the queries and the sorting of the data.

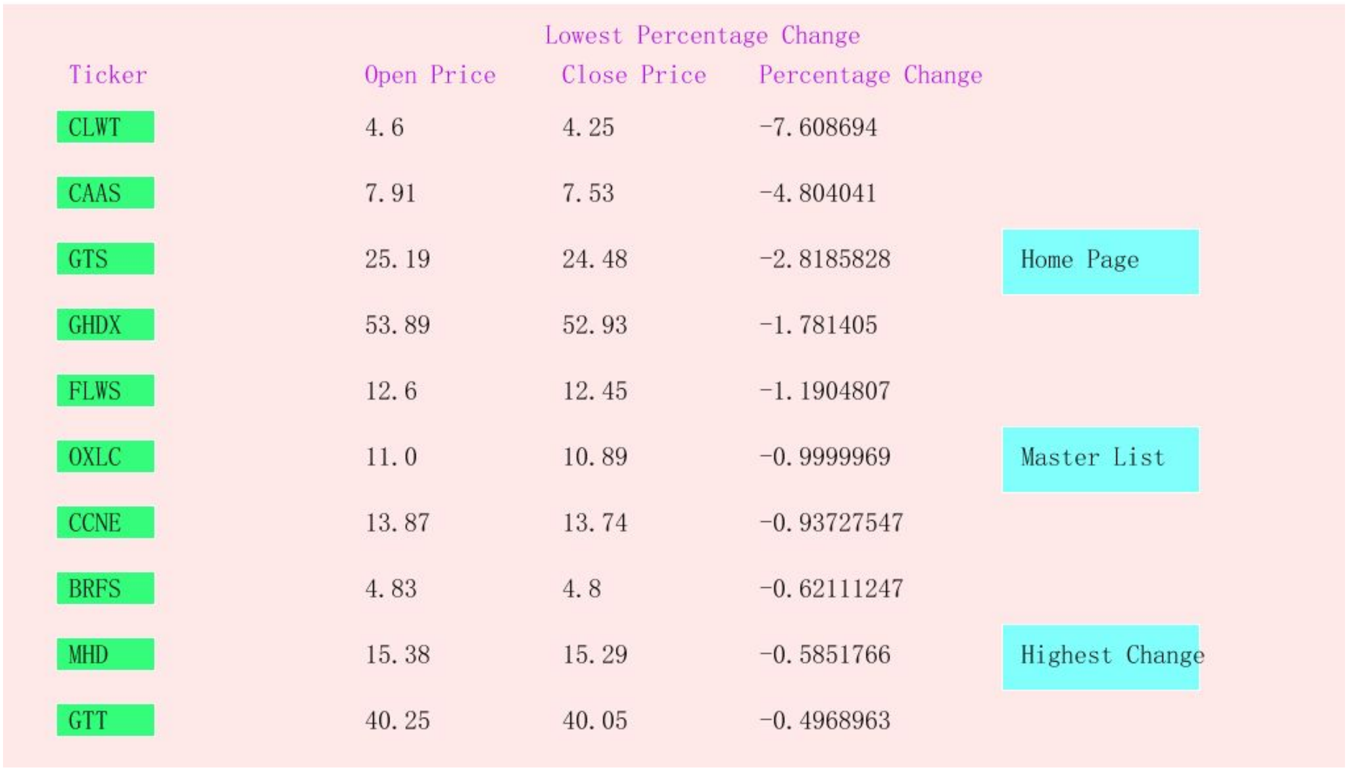
Screenshots





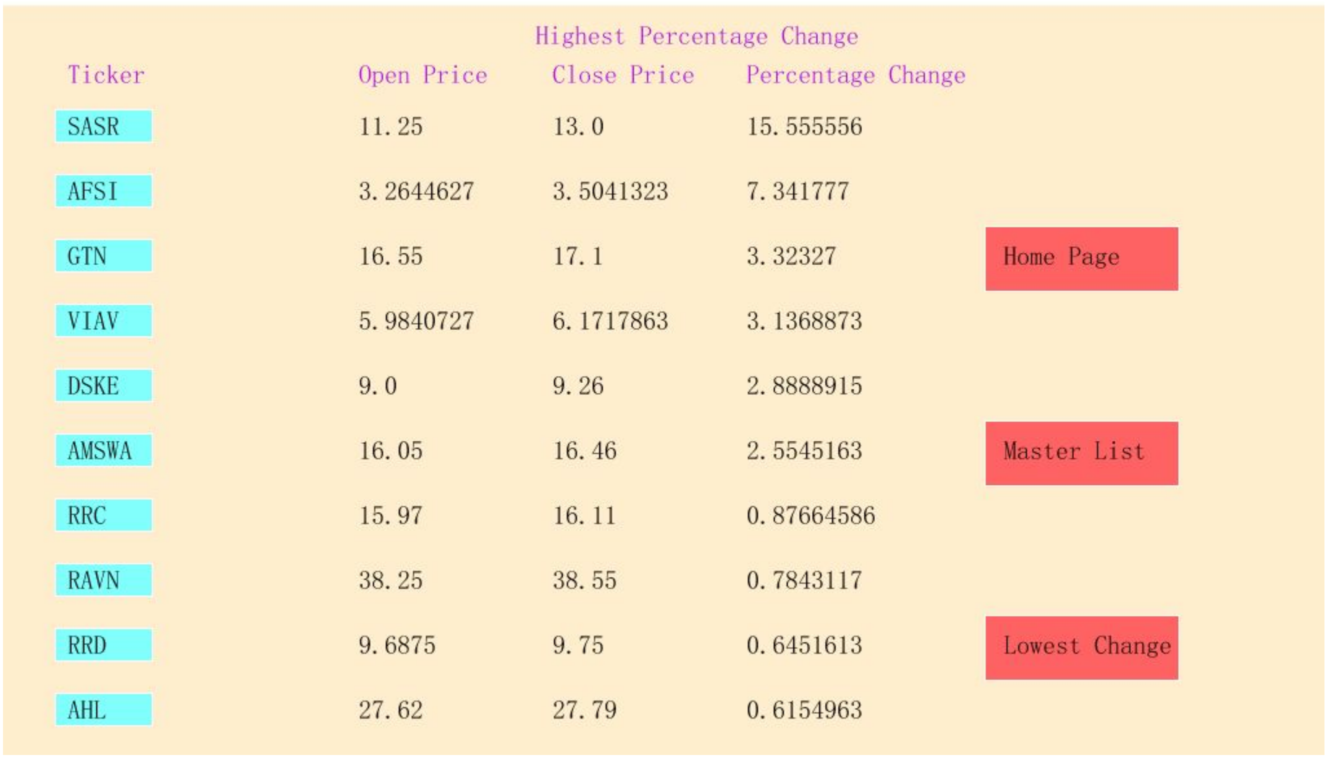
NYSE

	Open Price	Close Price	Change Over Time
EGHT	22.550	22.600	0.22%
APO	34.120	34.190	0.21%
AHH	15.580	15.670	0.58%
AHL	27.620	27.790	0.62%
MHD	15.380	15.290	-0.59%
BRFS	4.830	4.800	-0.62%
CRCM	18.560	18.580	0.11%
GTN	16.550	17.100	3.32%



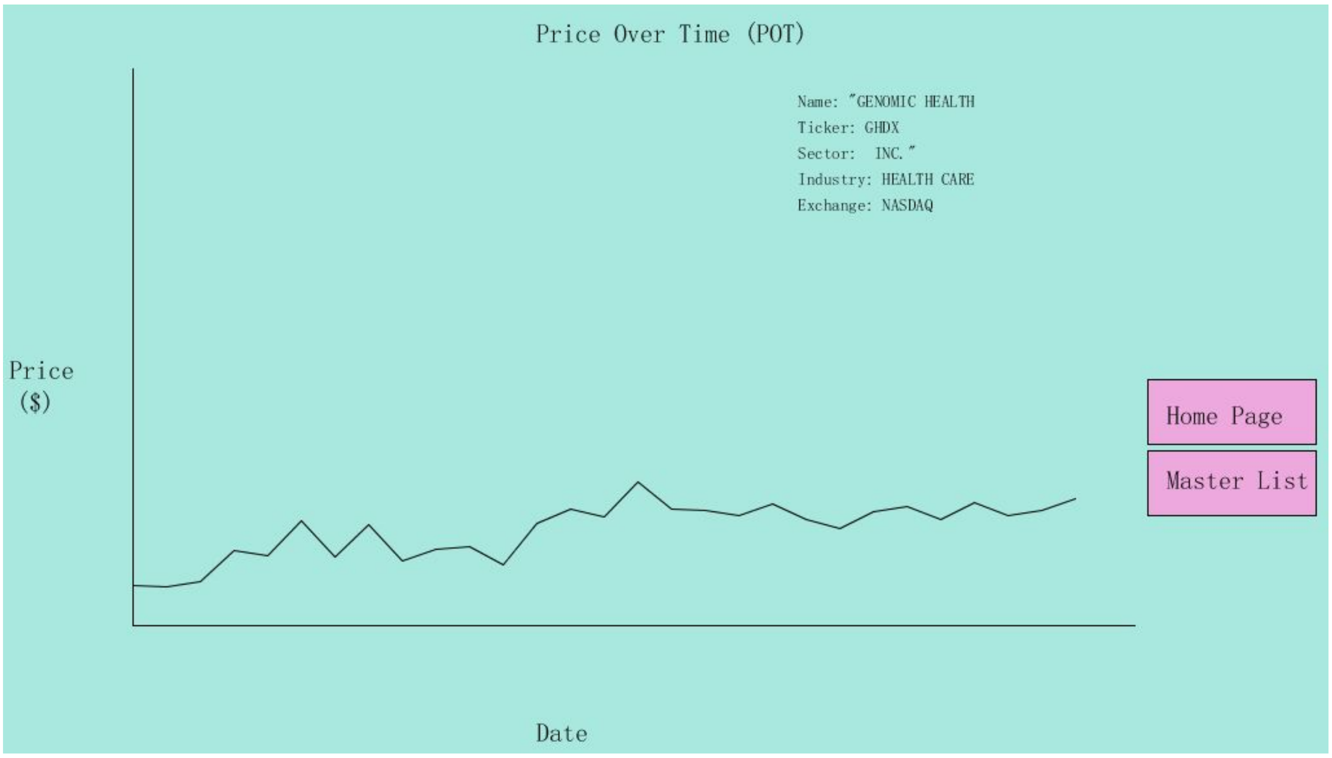
Lowest Percentage Change

Ticker	Open Price	Close Price	Percentage Change
CLWT	4.6	4.25	-7.608694
CAAS	7.91	7.53	-4.804041
GTS	25.19	24.48	-2.8185828
GHDX	53.89	52.93	-1.781405
FLWS	12.6	12.45	-1.1904807
OXLC	11.0	10.89	-0.9999969
CCNE	13.87	13.74	-0.93727547
BRFS	4.83	4.8	-0.62111247
MHD	15.38	15.29	-0.5851766
GTT	40.25	40.05	-0.4968963



Highest Percentage Change

Ticker	Open Price	Close Price	Percentage Change
SASR	11.25	13.0	15.555556
AFSI	3.2644627	3.5041323	7.341777
GTN	16.55	17.1	3.32327
VIAV	5.9840727	6.1717863	3.1368873
DSKE	9.0	9.26	2.8888915
AMSWA	16.05	16.46	2.5545163
RRC	15.97	16.11	0.87664586
RAVN	38.25	38.55	0.7843117
RRD	9.6875	9.75	0.6451613
AHL	27.62	27.79	0.6154963



Price Over Time (POT)

Name: "GENOMIC HEALTH
Ticker: GHDX
Sector: INC."
Industry: HEALTH CARE
Exchange: NASDAQ