CSCI X370 - Term Project Report

Major ETF Back Testing Application Using SQL, Java, and Spring

Team: DataDawgs

Original Contributions:

This project is a unique take on something that has been done before many times. The idea was to emulate a simple stock back tester. The back tester application would allow a user to test the gains or losses when investing in set of 4 exchange traded funds (ETFs) over a particular time frame. This can be particularly useful to analyze trends in different sectors of the economy, correlate economic performance with specific events in history, and to try to gain an understanding of future market performance based on past trends.

In the case of our application, the goal was to choose and work with 4 major ETFs that track 4 of the major stock market indexes, namely, the S&P 500, the NASDAQ, the Overall Market, and the Russell 2000. To add some background, and ETF is much like a stock. It trades and behaves very much in the same manner. The only difference is that an ETF is like buying a collection or a basket of stocks rather than individual ones. This allows a person to achieve broader market and portfolio diversification with a single "stock purchase."

Our purpose and what makes this project unique, is in strategically picking ETFs that track the indices listed above. With those 4 ETFs it allows us to easily identify the differences in performance in various parts of the market, in varying size market capitalizations, and to compare growth versus value stocks all with just 4 ETFs. The ETFs we have chosen are the following: iShares Russell 2000 ETF (IWM), Invesco QQQ Trust Series 1 ETF (QQQ), SPDR S&P 500 Trust ETF (SPY), and Vanguard Total Stock Market Index Fund ETF (VTI). These ETFs are very solid. Each has a long history, high liquidity, high volume traded per day, and consistently available public data.

The project is not the first stock back test ever, but what does make it special is its ability to quickly and simply allow the user to make important decisions about how various part of the market are preforming. For example, this year there has been a large and well documented shift in performance between the value side of the stock market and the growth side (value side would represent slow growing, dividend paying, blue-chip companies and growth would represent fast growing, no dividend paying, smaller companies). For roughly the last 10 years, growth has significantly out preformed the value side of the market by approximately 7% according to Forbes (Berger).

However, this year there seems to be a rotation to value. How can we quantify this? With this application the user can quickly compare the Russel 2000 as IWM (growth) to the S&P 500 as SPY (value) to determine if there is a historical performance difference between value and growth side of the market and lend clarity to the recent shift that is being talked about.

Broader Impact:

This application and others like it have had a significant impact on trend analysis and trading strategies since their inception. There are used by many investors and economist all over the word to develop trading strategies, quantify opportunity costs associated with investing, and analyze myriad of different market conditions and their effects on portfolio performance.

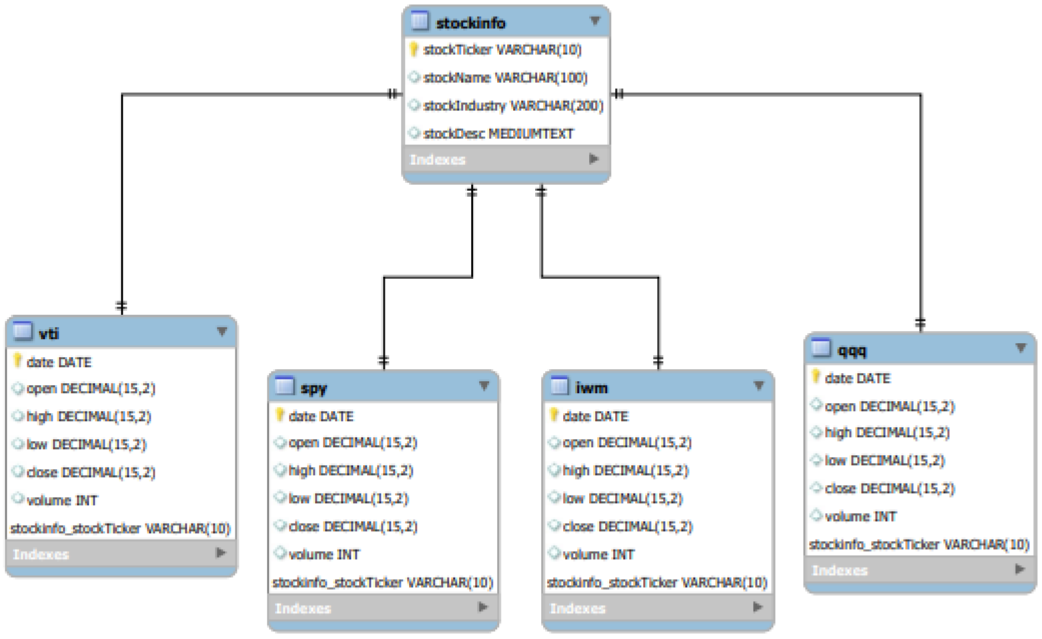
In recent decades with the sharp rise in popularity of mutual funds and now ETFs, there has been a paradigm shift in the way that most retail investors allocate their portfolios. In decades past the ideology was to buy a well-diversified portfolio of individual stocks, now however, the focus has shift for most people almost entirely to ETFs. After all, ETFs require much less thought, they are arguably safer, less volatile, and give immediate portfolio diversification even with a relatively small amount of money.

In response to this indexing craze, our team decided we would attempt to build a back testing application targeting the new age retail ETF investor. Our application gives the user a choice of the 4 most well-known ETFs. They can then choose an investible amount and test the performance of that amount over any of the 4 ETFs listed. This provides the new age investor with a quick, easy, and streamlined approach to test their investing strategy, compare past performance between the various ETFs, and preform basic long term trend analysis on the market.

System Architecture:

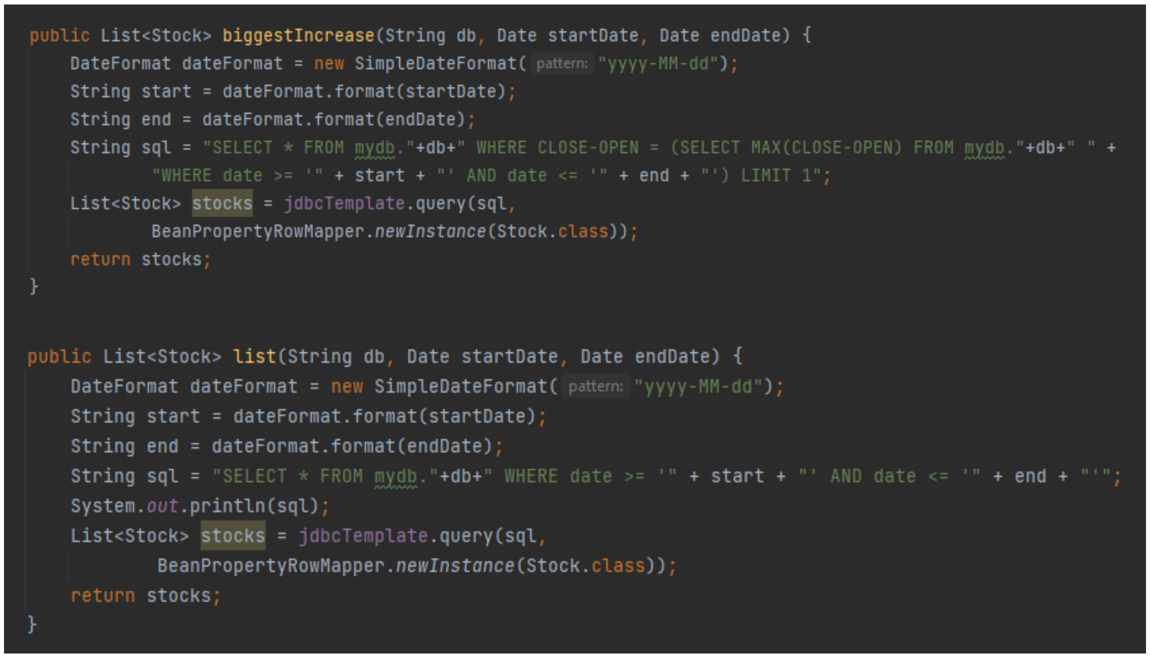
To develop this application, our team choose to use Java with Spring Boot framework and a mySQL database. In Spring Boot we choose to use the following dependencies: spring-boot-starter-jdbc to connect with our database, spring-boot-connector-java as our JDBC driver, spring-boot-thymeleaf to allow us to create premade html templates for our view, and also spring-boot-starter-web to help develop our UI.

Our database is essentially the same as what was submitted in our preliminary assignment which was already normalized after creation. Our database was structured in the following way:



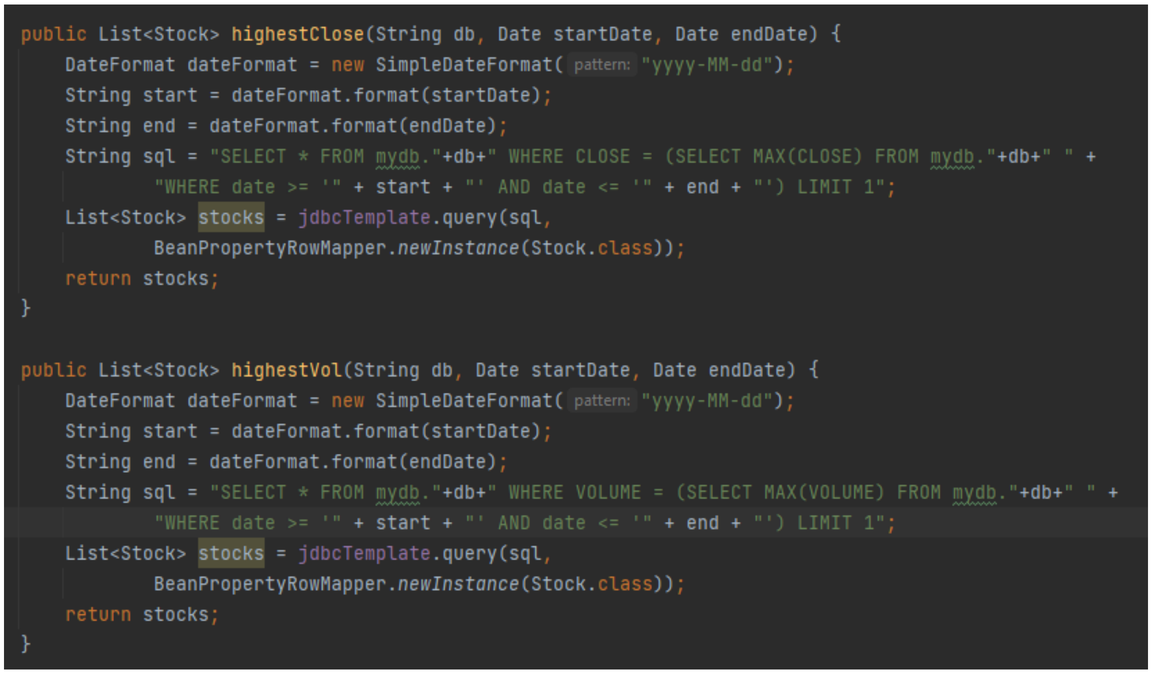
For this application, we tried to stick to a MVC style architecture. Our view was handled with the help of Thymeleaf and Spring-web. This allowed us to create static html and css pages to allow the user to view our data and interact with the application. The model or business layer was handled in several classes. The DOA class handled data access and two other classes Stock and StockInfo processed the data. Finally, we built a controller class to handle the flow of our application between the home page and stock page.

Sample Query Screenshots:



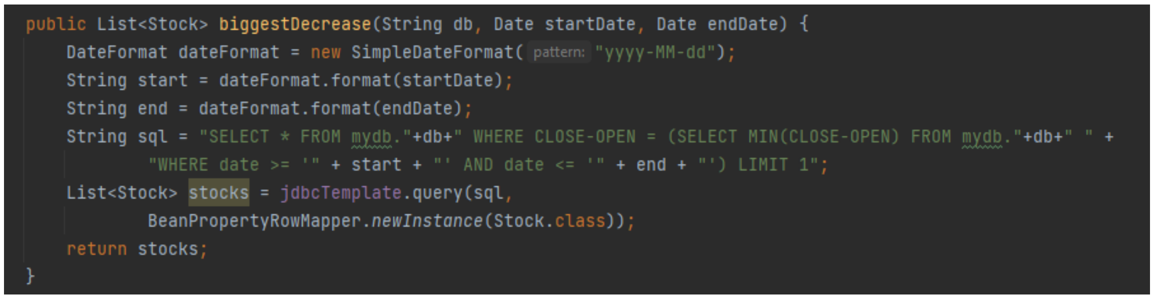
```java
public List<Stock> biggestIncrease(String db, Date startDate, Date endDate) {
    DateFormat dateFormat = new SimpleDateFormat( pattern: "yyyy-MM-dd");
    String start = dateFormat.format(startDate);
    String end = dateFormat.format(endDate);
    String sql = "SELECT * FROM mydb."+db+" WHERE CLOSE-OPEN = (SELECT MAX(CLOSE-OPEN) FROM mydb."+db+" " +
            "WHERE date >= '" + start + "' AND date <= '" + end + "') LIMIT 1";
    List<Stock> stocks = jdbcTemplate.query(sql,
            BeanPropertyRowMapper.newInstance(Stock.class));
    return stocks;
}

public List<Stock> list(String db, Date startDate, Date endDate) {
    DateFormat dateFormat = new SimpleDateFormat( pattern: "yyyy-MM-dd");
    String start = dateFormat.format(startDate);
    String end = dateFormat.format(endDate);
    String sql = "SELECT * FROM mydb."+db+" WHERE date >= '" + start + "' AND date <= '" + end + "'";
    System.out.println(sql);
    List<Stock> stocks = jdbcTemplate.query(sql,
            BeanPropertyRowMapper.newInstance(Stock.class));
    return stocks;
}
```

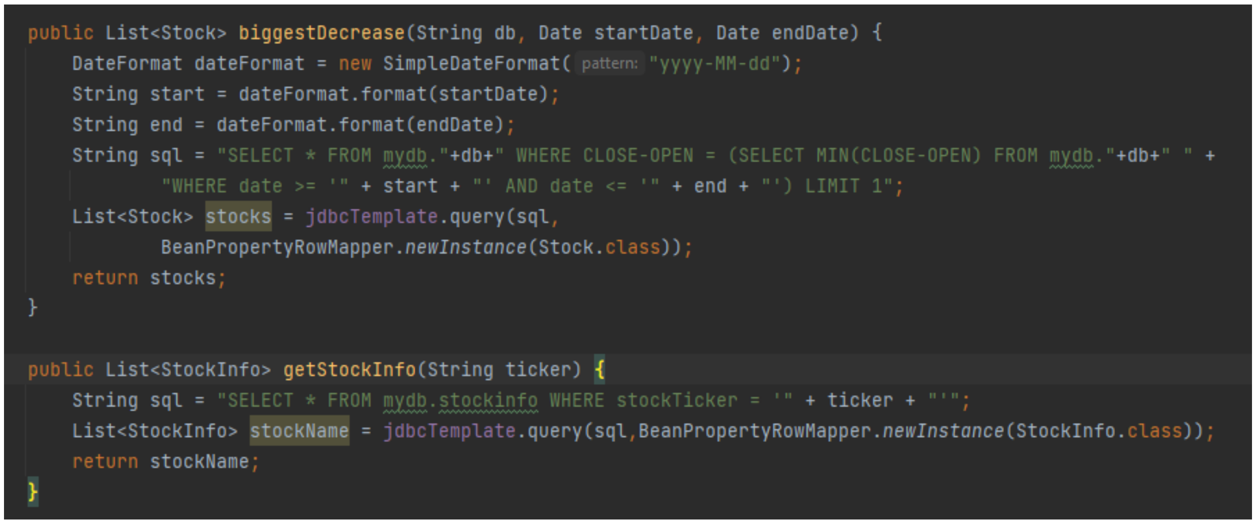


```java
public List<Stock> highestClose(String db, Date startDate, Date endDate) {
    DateFormat dateFormat = new SimpleDateFormat( pattern: "yyyy-MM-dd");
    String start = dateFormat.format(startDate);
    String end = dateFormat.format(endDate);
    String sql = "SELECT * FROM mydb."+db+" WHERE CLOSE = (SELECT MAX(CLOSE) FROM mydb."+db+" " +
            "WHERE date >= '" + start + "' AND date <= '" + end + "') LIMIT 1";
    List<Stock> stocks = jdbcTemplate.query(sql,
            BeanPropertyRowMapper.newInstance(Stock.class));
    return stocks;
}

public List<Stock> highestVol(String db, Date startDate, Date endDate) {
    DateFormat dateFormat = new SimpleDateFormat( pattern: "yyyy-MM-dd");
    String start = dateFormat.format(startDate);
    String end = dateFormat.format(endDate);
    String sql = "SELECT * FROM mydb."+db+" WHERE VOLUME = (SELECT MAX(VOLUME) FROM mydb."+db+" " +
            "WHERE date >= '" + start + "' AND date <= '" + end + "') LIMIT 1";
    List<Stock> stocks = jdbcTemplate.query(sql,
            BeanPropertyRowMapper.newInstance(Stock.class));
    return stocks;
}
```



```java
public List<Stock> biggestDecrease(String db, Date startDate, Date endDate) {
    DateFormat dateFormat = new SimpleDateFormat( pattern: "yyyy-MM-dd");
    String start = dateFormat.format(startDate);
    String end = dateFormat.format(endDate);
    String sql = "SELECT * FROM mydb."+db+" WHERE CLOSE-OPEN = (SELECT MIN(CLOSE-OPEN) FROM mydb."+db+" " +
            "WHERE date >= '" + start + "' AND date <= '" + end + "') LIMIT 1";
    List<Stock> stocks = jdbcTemplate.query(sql,
            BeanPropertyRowMapper.newInstance(Stock.class));
    return stocks;
}
```



```
public List<Stock> biggestDecrease(String db, Date startDate, Date endDate) {
    DateFormat dateFormat = new SimpleDateFormat( pattern: "yyyy-MM-dd");
    String start = dateFormat.format(startDate);
    String end = dateFormat.format(endDate);
    String sql = "SELECT * FROM mydb."+db+" WHERE CLOSE-OPEN = (SELECT MIN(CLOSE-OPEN) FROM mydb."+db+" " +
            "WHERE date >= '" + start + "' AND date <= '" + end + "') LIMIT 1";
    List<Stock> stocks = jdbcTemplate.query(sql,
            BeanPropertyRowMapper.newInstance(Stock.class));
    return stocks;
}

public List<StockInfo> getStockInfo(String ticker) {
    String sql = "SELECT * FROM mydb.stockinfo WHERE stockTicker = '" + ticker + "'";
    List<StockInfo> stockName = jdbcTemplate.query(sql,BeanPropertyRowMapper.newInstance(StockInfo.class));
    return stockName;
}
```

Team Member	Spencer King	Daniel Abramow	Shreenal Patel
Responsibilities	Responsible for creating and normalizing the database, setting up spring boot, creating the IWM webpage, and finalized the project documentation.	Responsible for setting up our .py script, created the SPY page and Home page, helped with the documentation, and helped create the Javadocs.	Responsible for finding and setting up our graphical representation of data, creating the QQQ and VTI pages, and created our readme file explaining our project.

Conclusion:

In this project, we used MySQL for our database, Java for the backend, JDBC to connect to the database, and Spring Boot for our Web Framework. The goal of this project was to create an application to allow users to do basic back testing going back at least 10 years for 4 of the largest ETFs (VTI, SPY, QQQ, and IWM), and that was successfully achieved.

We also achieved our goal of creating something not only usefully, but also original. We took something old like stock market back testing and recreated it in a streamlined and efficient way to target a new breed of investors that are more concerned with ETFs than individual stocks. We provided a way for these new investors to quickly gain real world quantitative data to facilitate analysis, strategy testing, and opportunity cost management.

Overall, this project was a successful, however it is only the beginning. There is much more to do, and our team plans on further developing this application with even more features in the future.

References:

Berger, Rob. Do Value Stocks Really Outperform Growth Stocks over the Long Run? 9 Mar. 2021, www.forbes.com/advisor/investing/value-vs-growth-stocks-performance/.