CIS4500 - Project Outline

1. Motivation for the idea/description of the problem the application solves

Our application aims to provide users with a comprehensive understanding of stocks' and indices' performances over time, with a specific focus on the 2013-2018 period. The 2013-2018 period holds several key similarities to the present, which makes it an interesting project for us to undertake. Namely, the 2013-2018 period was part of the global economic recovery from the 2008 Financial Crisis, and we similarly are in the recovery period from the COVID recession. There were also contrasting market dynamics and governmental policies in the 2013-2018 period with significant bull and bear runs and ever-shifting monetary/fiscal policies, which is very similar to the economic market we are currently in. As such, we believe the analysis that can be produced from this project has significant implications for our present.

With our two datasets, we plan to provide users with the ability to analyze not only individual stock performances and indices but also compare and contrast selected stock performances against chosen indices and against other stocks. The vast amount of data available to us allows us to produce very interesting queries and analysis, ranging from stock/index volatility analysis, visualizations of time-series performance data, best/worst performers, and much more. The ultimate goal is to provide an immensely flexible application that provides a user-friendly interface, allowing all users to explore/analyze the extent of the data available to them and draw their own conclusions for the looming recessionary period.

2. List of features you will definitely implement in the application

Selecting multiple stocks by ticker name, with specified dates, then overlaying them on top of a line graph for comparison. (Simple query)
Identify the top 5 stocks by percentage of the total S&P index. (Simple query)
Identify the top 10 stocks by positive percentage change in price in a selected period of time. (Simple query)
Identify the top 10 stocks by negative percentage change in price in a selected period of time. (Simple query)

Identify the top 10 stocks by volatility in a selected period of time. (Simple query)
Selecting multiple indexes by ticker name, with specified dates, then overlaying them on top of a line graph for comparison. (Simple query)
Compare and contrast the performance of selected S&P 500 stocks with selected indices. (Complex query)
Query to take the mean of selected stocks and compare with selected indices to see how a batch of stocks have performed in comparison to the overall market performance of a particular country. (Complex query)
Query to get the beta of a particular stock relative to an index in a given time period. (Complex query)
View to artificially produce the S&P500. We don't have this index, so we will be artificially creating it with a price-weighted index of the component stocks. (Complex query)
Find the top-performing stocks on a specific date, based on their percentage change in price relative to a market index. (Complex query)

3. List of features you might implement in the application, given enough time

Add a user login page with complex sign-in logic (following that of Google and Facebook) that allows users to create their own dashboards for stocks and index performances
Allow users to customize their dashboards with certain stocks and indices that pique their interests, by querying to these specific stocks and indices
Write unit tests for both the backend and frontend of the application
Rank each stock based on its performance index compared to the overall performance indices of all stocks
Rank stocks based on projected returns in the future using linear regression.
Give the user's dashboard an overall score based on the collective rankings of the stocks they have chosen to include in their dashboard

4. List of pages the application will have and a 1-2 sentence description of each page. We expect that the functionality of each page will be meaningfully different from the functionality of the other pages.

HomePage:

When a user navigates to our applications, they will arrive at HomePage.js, which will provide a written paragraph description of the application. Additionally, this page will have a logo for our site and an image of a trend line depicting a time series of stock returns over time.

Basic Stats:

A user can navigate to this page via a Navigation Bar at the top of the site that will be defined in NavBar.js. On this page, a user can select what S&P 500 statistics they are looking for by checking various boxes such as mean, standard deviation, min, max, and/or various percentiles, and upon selecting the 'Go' button, the user will see a specific output frame.

S&P 500 Stock Queries:

A user can also navigate to this page via the top NavBar. On this page, we will take in a user input of selected stock(s) via a search bar. The user will also be able to select specific information/statistics about the selected stocks that they want to visualize based on options in a dropdown menu. Based on these selected tickers and requested information, this page will display a visualization of the resulting table.

Stock Exchange Index Queries:

A user can also navigate to this page via the top NavBar. Here, users can select an index of their choice based on a display of options as well as particular stocks via a search bar, and upon pressing 'GO' a visualization of the resulting tables will appear on this page. This page will also have a start date and an end date field that users can select.

Stock/Index Comparisons Page:

When users navigate to this page via the NavBar, they can select an index from our stock exchange indices as well as features that we wish to compare to the S&P 500 over a given period of time. To do this, this page will have a start date and an end date field that users can select. The visualizations will be displayed upon selecting 'GO'.

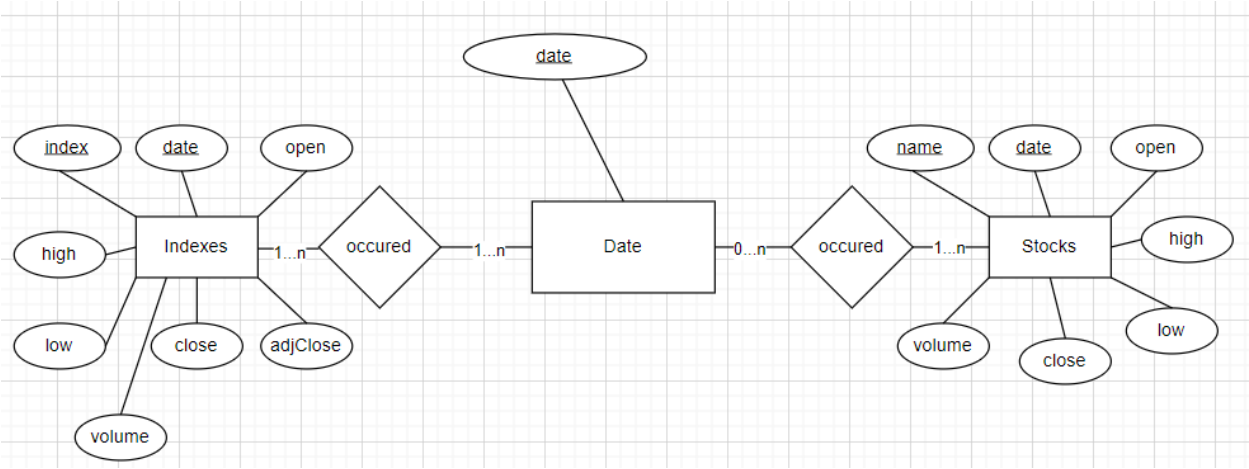
Login Page:

Login page includes Username and Password text fields that the users can enter their login information into if they already have an account, and an option at the bottom to create an account if the user does not already have one. The bottom of this page will have a ‘Login’ button that will send a user to the main page if information is verified as correct.

Register:

Register page includes the Username field (must be unique), email field, password, and confirm password, as well as a ‘Register’ button at the bottom that sends them to the Home Page.

5. Relational schema as an ER diagram



Explanation:

Underline indicates primary keys of each entity. Since our queries mainly involve joins based on common dates between indexes and stocks, we thought of creating a new entity Date that relates both Indexes and Stocks entities. This way, joins based on date can be easily done by merging Indexes and Stocks with Date table, where a subquery selects a specified date range in the Date table first. Regarding foreign keys, Indexes ‘date’ and Stocks ‘date’ are foreign keys referencing Date’s ‘date’ to ensure referential integrity. Note that Date has a 1 to n participation with Indexes while 0 to n participation with Stocks. The reason is because Indexes cover 1986 to 2021 while Stocks cover 2013 to 2018. Therefore, every Date must have occurred in an Index, but not every Date would have occurred in a Stock. Each Index and Stock has mandatory participation to Date,

since they must have a date with a particular price.

6. SQL DDL for creating the database

```
CREATE TABLE Date (
   date DATE PRIMARY KEY
);

CREATE TABLE Indexes (
   index VARCHAR(15),
   date DATE,
   open DECIMAL(15, 6),
   high DECIMAL(15, 6),
   low DECIMAL(15, 6),
   close DECIMAL(15, 6),
   adjClose DECIMAL(15, 6),
   volume INT,
   PRIMARY KEY (index, date),
   FOREIGN KEY (date) REFERENCES Date(date)
);

CREATE TABLE Stocks (
   name VARCHAR(10),
   date DATE,
   open DECIMAL(15, 6),
   high DECIMAL(15, 6),
   low DECIMAL(15, 6),
   close DECIMAL(15, 6),
   volume INT,
   PRIMARY KEY (name, date),
   FOREIGN KEY (date) REFERENCES Date(date)
);
```

7. Explanation of how you will clean and pre-process the data. This tutorial demonstrates how to do simple pre-processing in Python.

First convert all date columns in the two csv files indexedProcessed.csv and all_stocks_5yr.csv from string to datetime data type. Rename the column names such that they match the DDL specified column names above. Remove all null values from all_stocks_5yr, as it contains 11 null values. No null values for indexProcessed.

8. List of technologies you will use. You must use some kind of SQL database. We recommend using MySQL or Oracle specifically because you will use MySQL in HW2, and we will provide guidance for setting up a MySQL database. Some groups in the past have had issues with MySQL, but Oracle is another option.

MySQL is our selected option.

9. Description of what each group member will be responsible for. (ChatGPT used to provide specific details on what each role might entail)

Ashish Pothireddy (Writing SQL queries - Backend):

- Design and implement the database schema to support the project's data requirements.
- Write efficient SQL queries to retrieve, update, and manipulate data in the database.
- Optimize database performance to ensure the website runs smoothly and responds to user requests quickly.
- Collaborate with the frontend developers to establish the necessary APIs and endpoints for data retrieval and submission.
- Debug and troubleshoot any database-related issues and provide support to the team for database-related tasks.

Noah Erdogan (Website Development - Frontend):

- Design the user interface and user experience (UI/UX) of the website based on project requirements and user needs.
- Write HTML, CSS, and JavaScript code to create web pages and ensure they are responsive and visually appealing.
- Implement client-side functionality, such as user interactions, form handling, and dynamic content loading.
- Collaborate closely with the backend team to integrate the frontend with the backend through API calls..

Austin Wang (Writing SQL queries - Backend):

- Collaborate with Ashish to design and maintain the project's database schema.
- Write SQL queries and scripts to interact with the database, such as SELECT, INSERT,

UPDATE, and DELETE operations.
- Optimize database performance by implementing query optimization techniques.
- Provide support for data-related requests and troubleshoot database issues when they arise.

Alex Huang (Website Development - Frontend):

- Work with Noah to create the website's frontend components, layouts, and user interfaces.
- Write clean and well-structured HTML, CSS, and JavaScript code to ensure a seamless user experience.
- Collaborate with the backend team to integrate frontend components with backend services through API endpoints.
- Ensure that the website is user-friendly, intuitive, and visually engaging to maximize user engagement.

Overall, this division of responsibilities ensures a well-rounded approach to the group project, with Ashish and Austin focusing on the backend, and Noah and Alex concentrating on the frontend, which collectively leads to the development of a functional and user-friendly website with robust database support.