**Stock price prediction and analysis**

**Project Details:**

* Project: Stock price prediction and analysis using Yahoo finance data
* Team Name: NIK
* Members:
  + Ieshaan Sharmaa (ieshaans)
  + Kaustuv Majumder (kaustuvm)
  + Niraj Chetry (nirajche)

**Project Statement:**

Yahoo finance api is a popular data source for financial data related to stock market. This data can be used to get the live stock information; as well as historical stock performance.

The aim of this project is to extract all stock related information from Yahoo finance, and store it in relational data models. This project also aims to demonstrate the end-to-end data ecosystem. It will have the following components:

* Data extraction from api (yfinance api)
* Data transformation and enrichment
* Data modelling
* Data quality check
* Application of machine learning algorithm for prediction
* Graphical user interface for end users

1. *Which dataset are you going to use?*

We are extracting live and historic Yahoo Stock Price data via Yahoo Finance API. The data is then stored in PostgreSQL DB.

1. *What kind of queries do you want to ask?*

The DB will fulfill the following needs:

* It’ll help us to monitor the performance of the stock over a period of time.
* Depending upon the performance of the stock, should a person invest in the particular stock or not.
* Monitoring DB data as a form of real-world application in the frontend.
* Forecasting the stock price of Yahoo data based on historical data and comparing the accuracy with the test dataset.

1. *How is the data updated?*

In our PostgreSQL, we have created 5 tables and their updation policy are as under-

* **Market Table -** Static Table, manually updated if required
* **Company Table -** Static Table, manually updated if required
* **Company Detail -** Static Table, manually updated if required
* **Stock History Table -** Updated once every day
* **Stock Matrix Table -** Updated as per user request

*D. Why do you need a database instead of an excel file?*

* Databases store information more efficiently, databases can handle volumes of information that would be unmanageable in a spreadsheet. Spreadsheets have record limitations whereas databases do not.
* Compared to databases, spreadsheets can require a large amount of hard-drive space for data storage. When a spreadsheet has many fields or a large amount of data (1000s of rows), the spreadsheet can be hard to read.
* Finding specific data can be cumbersome. Relational databases use querying tools to overcome these issues.

**Target Users:**

The following user groups have been identified so far:

* Retail investors
* Stock market analysts

**Database Description:**

The data model has been designed to accommodate stock information. This consists the following:

* Historical data (Day wise trend)
* Live status from the market

The schema has been designed to maintain data integrity; as well as scope of future improvement.

Following are the tables:

**MARKET**

This table contains information about the stock markets

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Data Type | Constraint | Description |
| MARKET\_CODE | Varchar(5) | Primary Key | The Market code as per ISO standard |
| COUNTRY | Varchar(50) | Not Null | The country in which the market operates |
| CITY | Varchar(50) | Not Null | The city in which the market is physically located |
| REGION | Varchar(5) | Not Null | The geographical region of the market (eg. NA, EU, APAC, etc) |

**COMPANY**

This table contains basic information of the company

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Data Type | Constraint | Description |
| SYMBOL | Varchar(5) | Primary Key | The stock symbol as per the market convention |
| NAME | Varchar(50) | Not Null | The company name |
| SECTOR | Varchar(50) | Not Null | Company’s business (eg. Finance, Technology) |

**COMPANY\_DETAIL**

This table contains detail information of the company

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Name | Data Type | Constraint 1 | Constraint 2 | Description |
| SYMBOL | Varchar(5) | Primary Key | Foreign Key (COMPANY.SYMBOL) | The stock symbol |
| MARKET\_CODE | Varchar(5) | Foreign Key (MARKET.MARKET\_CODE) | The Market code |
| ADDRESS | Varchar(100) | Not Null |  | The registered address of the company as per stock market |
| PHONE | Varchar(12) | Not Null |  | The registered phone number of the company as per stock market |
| DESCRIPTION | Varchar(100) | Not Null |  | The company description as provided by Yahoo finance |

**STOCK\_HISTORY**

This table contains the historical information of the stocks

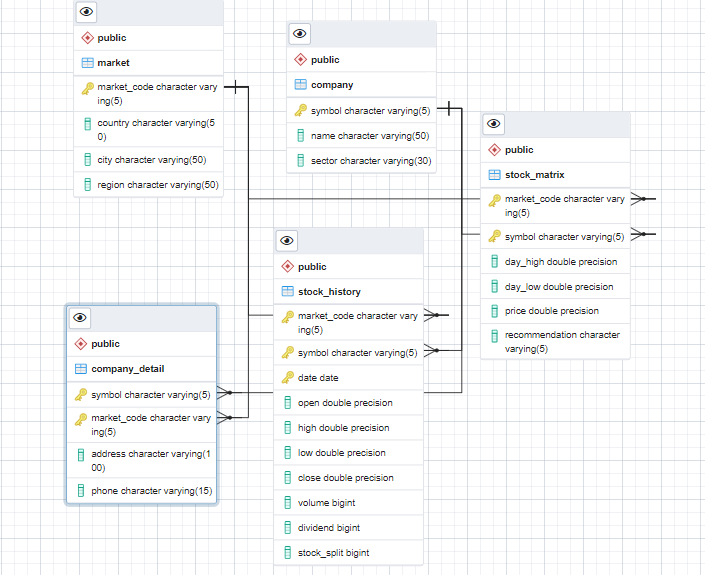
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Name | Data Type | Constraint 1 | Constraint 2 | Description |
| SYMBOL | Varchar(5) | Primary Key | Foreign Key (COMPANY.SYMBOL) | The stock symbol |
| MARKET\_CODE | Varchar(5) | Foreign Key (MARKET.MARKET\_CODE) | The Market code |
| DATE | Date |  | The record date |
| OPEN | Float8 | Not Null |  | Valuation at market open on that day |
| HIGH | Float8 | Not Null |  | Highest intraday valuation on that day |
| LOW | Float8 | Not Null |  | Lowest intraday valuation on that day |
| CLOSE | Float8 | Not Null |  | Valuation at market closure on that day |
| VOLUME | Int8 | Not Null |  | The volume of stocks in circulation |
| DIVIDEND | Int8 |  |  | Dividend announced (if any) |
| STOCK\_SPLIT | Int8 |  |  | Stock Split announced (if any) |

**STOCK\_MATRIX**

This table contains the live stock data as fetched from the api.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Name | Data Type | Constraint 1 | Constraint 2 | Description |
| SYMBOL | Varchar(5) | Primary Key | Foreign Key (COMPANY.SYMBOL) | The stock symbol |
| MARKET\_CODE | Varchar(5) | Foreign Key (MARKET.MARKET\_CODE) | The Market code |
| DAY\_HIGH | Float8 | Not Null |  | Highest intraday valuation |
| DAY\_LOW | Float8 | Not Null |  | Lowest intraday valuation |
| PRICE | Float8 | Not Null |  | Current price |
| RECOMMENDATION | Varchar(5) | Not Null |  | Buy/Sell recommendation |

**ERD Diagram**



Technical Details:

Data Usage :