ETL Project Report

Building Up A Financial Market Database

Group 01

Dao Tran

Kirsten Tallner

Martin Biber

Date 09.12.2021

Table of Contents

- 1. Project Topic
- 2. Research Questions
- 3. Project Procedure
  - 3.1 Overview
  - 3.2 Data Sources and Data
  - 3.2.1 Data Sources
  - 3.2.2 Legal Basis: Check robots.txt
  - 3.3 Tools used
  - 3.4 Solution Steps
  - 3.4.1 Student A
    - 3.4.1.1 Data Extraction
    - 3.4.1.2 Data Transformation
  - 3.4.2 Student C
    - 3.4.2.1 Data Extraction
    - 3.4.2.2 Data Transformation (Phase 1)
  - 3.4.3 Student B
    - 3.4.3.1 Data Extraction
    - 3.4.3.2 Data Transformation
  - 3.4.4 Student C: Data Transformation (Phase 2)
  - 3.5 Answers to our Questions
  - 3.6 Loading Data into MariaDB
  - 3.6.1 Student A
  - 3.6.2 Student B
  - 3.6.3 Student C
- 4. Lessons Learned and Self-Reflection
- 5. Consent

Task and Questions

1. Project Topic

A lot of people have an interest in the financial market, be it for fun, trading or investing. One potential influence on stock market prices might be given by insider trading. In order to analyze such a possible connection and its impact, it is mandatory to bring data on stock market prices, company details and insider trading together.

Because there is a lot of free data available on the internet, we would like to build a financial market database that has various information bundled in one place. It should be designed in a way that it could be enhanced at a later stage of interest. Please note that we mainly focuses on the asset class equity.

This project is created using Python and MariaDB. It is furthermore set up as an ETL-job, where the data needs to be extracted from the internet (by webscraping / usage of API), cleaned and modified (transformation) as well as connected to and loaded into a database.

2. Research Questions

Our Group is interested in answering the following questions:

1. Who are the three largest buyers in the five best ranking ESG Total Score of the above sector average performing S&P500 Technology stocks in 2021?
   This is our most challenging question as we need to integrate the work of all three students. From the whole data set we select the technology sector and evaluate which stocks outperformed the average technology stocks. After that we select the five best ranked ESG Stocks of the former selection. And finally we want to show which insiders have bought the most stocks of those companies.
2. Who are the three largest buyers and sellers in the five best performing S&P stocks in 2021?
3. Do the S&P 500 members or the 500 most active non-members have the higher volume on average? (Activity is measured by the trading volume as of October 31st.)
4. How many of the 10 best ranking ESG stocks have a better Performance than the S&P500? (SPY is used as benchmark.)

3. Project Procedure

3.1 Overview

The project was successfully processed in close collaboration of the group members. The tasks were split as follows (see Figure 1):

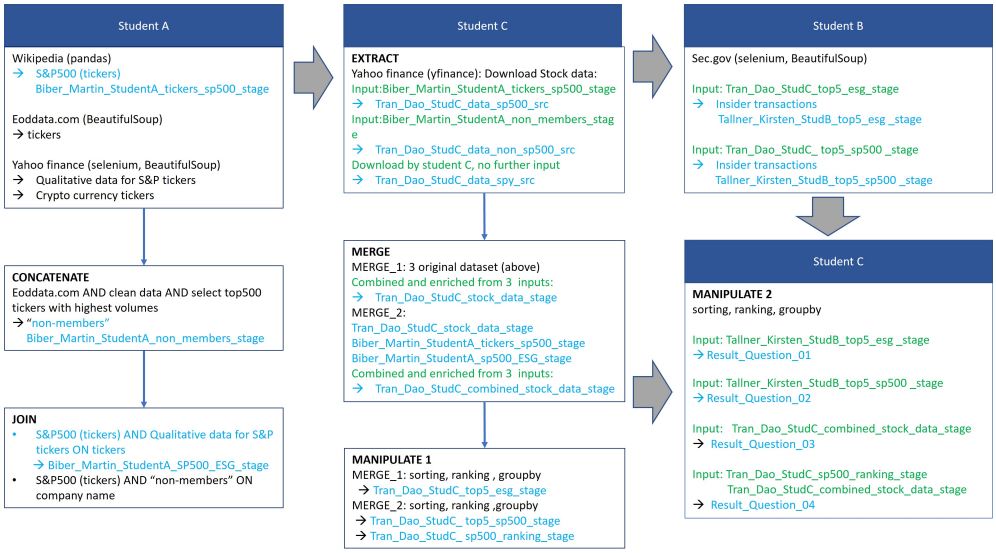


Figure 1: Overview of the project workflow. Input files are depicted in green, self-generated or output files in blue.

3.2 Data Sources and Data

3.2.1 Data Sources

Yahoo Finance: Stock quotes, news, international market data
Eoddata: Data on NYSE, NASDAQ, LSE, GLOBALINDEX, FOREX
Wikipedia: Members of the S&P 500
U.S. Securities and Exchange Comission: Company filings (trading documents)

3.2.2 Legal Basis: Check robots.txt

https://eoddata.com.siteindices.com/ doesn't mention any restrictions. It even allows: GoogleBot, Bingbot, BaiduSpider and YandexBot. On a second source: https://eoddata.com/robots.txt it mentions that there is a crawl-delay of 10. It doesn't give any units to the number.

https://de.wikipedia.org/robots.txt
Wikipedia has a large list. It does allow scraping just not for wikipedia related sensitive data.

https://www.sec.gov/robots.txt
U.S. Securities and Exchange Commission (SEC) explicitly allows for crawling */Archives/edgar/data/*. No info/limitations regarding crawl-delay is declared.

https://finance.yahoo.com/robots.txt
yahoo finance allows the scraping of data. The page we use "https://finance.yahoo.com/quote/" is nowhere mentioned.

3.3 Tools used

Python (version 3.8)
Pycharm (version 2021.3)
Jupyter Notebook (version 6.0.3)
MariaDB version(Server version: 10.3.31-MariaDB-0ubuntu0.20.04.1 Ubuntu 20.04)

3.4 Solution Steps

3.4.1 Student A

3.4.1.1 Data Extraction

Each listed company has a ticker, sometimes also called symbol, which is a unique identifier like a primary key in a database. There is unfortunately no source where the stock exchanges list all available tickers. The webpage www.eoddata.com has the most comprehensive list that is available for scraping. A company is listed on one or more stock exchanges and a lot of countries have several stock exchanges. In order to gather all possible stocks in the US, it was necessary to scrape several different exchanges. One of the criterias of our project was that the scripts should be set up in a way that allows the users to further develop the chosen approach at a later stage. Due to that five exchanges from www.eoddata.com were scraped. The wikipedia has a site that lists the constituents of the S&P 500, the probably most widely followed equity index in the world. This list was scraped in order to answer some of our questions. As crypto currencies become more popular a list of the cryptos with the highest market value was scraped from yahoo.finance.com.

The first step uses the script "StudentA_E1_TickerAllinOne.py" to scrape the wikipedia list. This is done with python pandas. Every time a list is complete it is saved in a csv file as well as in a python dictionary. The csv files act as a safety option in case there is a problem with the dictionary that will hold all tickers. It is organized as a nested dictionary where each completed list is saved as a dictionary value and the dictionary key is the related exchange. The second step is to iterate over the defined exchanges. All Exchanges have a subpage for each letter of the alphabet. Therefore five markets times 26 letters result in 130 pages that are scraped with BeautifulSoup. After an exchange has been scraped the data frame is added to the dictionary and is saved as a csv for safety reasons. This was a worthwhile step as it turned out. Only about 10% of the whole data frame was saved in the json file. The following tasks had to be done with the csv files. In a last step the crypto currencies with the highest market values were scraped with BeautifulSoup on yahoo.finance.com

The ticker data for the S&P 500 should be enriched with some additional data like sector, industry and environmental-social-government (ESG) rating. This information is scraped with Selenium and BeautifulSoup on yahoo.finance.com. The downloaded information is saved in a dictionary. This dictionary is saved as a json file after each ticker. This ensures that in case of an interruption, we don't lose the successfully downloaded data. The script "StudentA_E2_scrapeYahoo2.py" first checks if the json file already exists which would be the case if the process was interrupted. If it doesn't exist, a new dictionary is created. After that a loop is going through all the S&P 500 tickers and checks if the ticker is already in the dictionary. If yes, it is skipped otherwise the script loads the profile page on yahoo finance and saves the sector, the industry and the homepage. It then changes the sub page to "Sustainability". If the company has an ESG Rating, the total ESG Rating, the environmental rating, the social rating and the goverance rating is scraped.

3.4.1.2 Data Transformation

The script "StudentA_T2_Joining.py" executes the joining tasks. In the first join the tickers from Wikipedia (S&P 500) are joined with the qualitative data (like sector and ESG Rating) that was scraped on yahoo finance. The json file with the yahoo data is loaded into a pandas data frame and the csv with the S&P 500 memebers is imported into a pandas data frame as well. The two data frames are joined on the column Symbol and ticker. These two columns have the same unique identifier (ticker) but wikipedia calls them symbols (some other sources call them Code). As it is a unique identifier, this join is very simple and save. The resulting csv is then used by student C to answer our questions.

The second join is used to add the volume to the S&P 500 ticker list. We were asked to use a different attribute for this join. This is done specifically for this task. In a real life situation you would always choose the unique identifier over an attribute which is not necessarily unique. The data from the eoddata is imported into a pandas data frame and then merged on the name of the company. This join only got 87 out of 505 right. A second approach was to use only lower case. This yielded in a slightly better result: 97 out of 505. The final solution was a module called fuzzywuzzy. It created a new column in one of the data frames with

the name of the most similar name. This yielded in over 91 percent correct results. There are other modules available. One of them seemed to be very promissing (fuzzy_pandas). It unfortunately could not be installed. It is well know that this module often doesn't work. But when it works it gives very interesting results. As explained above, we did it this way because that was the task. The easy, save solution would be a join with the ticker and symbol again.

3.4.2 Student C

3.4.2.1 Data Extraction

To serve the purpose of the project, financial data of stocks (including members of S&P 500 and non-members) are downloaded by using the **Yahoo Finance API**. The Python module called **yfinance** is used download the financial time series with the attributes of stock prices for the S&P 500 members and non-members over time. These attributes include in order to extract stock data per each ticker such as:

- Date: The given date
- Open:The prices at which a stock began trading in the same period
- High: The maximum prices in a given time period
- Low: The minimum prices in a given time period
- Close: The prices at which a stock ended trading in the same period
- Adj Close: The overall value of the stock
- Volume: The total amount of trading activity

For the purposes of the project, the data offered by student C will contribute to understand firstly the stock's performance, trading volume. It would help later to find out who the best buyers and sellers per each significant stock. With that said, we will only focus on analyzing the **Close** price, **Volume** in a given time period. The **Close** price is considered the price traders agreed on after all the action throughout the day. A stock's performance, such as one-year, is calculated by using the closing price from a year ago and compare it to the closing price from today to get the annual return. In the scope of our project, the stock's performance is calculated based on **Close** price on 31st of December, 2020 on 31st of October, 2021. Secondly, trading volume is the product of **Volume** and **Close** price per day. Volume is the total number of shares traded in a security over a period. It is important to understand the difference between trading volume of S&P 500 member and non-member groups.

To approach the stock data, we choose the finance module in Python and it is a very friendly easy module to work with. We need to load the following libraries: **pandas** and **yfinance** at a first glance. By using the function *yf.download()* to pull back stock prices from the 31st of December, 2020 to the 31st of October, 2021. The function helps to pull the ticker, date, open, high, low, close, adj close and volume of all tickers.

Because we would like to get the data of S&P 500 members and 500 most active non-members, we can download multiple stock prices of more than one asset at one time. Student A provides the 2 list of tickers (i.e., list of S&P 500 members and list (*Biber_Martin_StudentA_tickers_sp500_stage.csv*) of 500 most active non-members (*Biber_Martin_StudentA_non_members_stage.csv*) which helps to generate a dynamic list of tickers to pull from Yahoo Finance API based on the imported file with tickers.

All of codes that written to download and concatenate original files are shown the script "Tran_Dao_StudC_extract_code.ipynb". These following datasets are downloaded to serve the purpose of the project:

- Dataset 1, named as *Tran_Dao_StudC_data_sp500_src.csv* contains the stock data of S&P 500 members, with 505 tickers (companies).
- Dataset 2, named as *Tran_Dao_StudC_data_non_sp500_src.csv* contains the stock data of non-members of S&P 500, with 500 tickers.
- Last but not least, the SPY (SPDR S&P 500 ETF) is an important index that is used as benchmark for comparing the performance among stocks. The stock data of "SPY" is in the filename *Tran_Dao_StudC_data_spy_src.csv* as Dataset 3.

3.4.2.2 Data Transformation (Phase 1)

After achieving the 3 original datasets, a merged file which combines the historical stock data of S&P 500 members, non-members and "SPY" as a whole, is saved as *Tran_Dao_StudC_stock_data_src.csv*. In order to adapt the requirement of the project, the original data was made dirty manually and saved as *Tran_Dao_StudC_stock_data_src_dirty.csv*. The dirty data was cleaned and stored as *Tran_Dao_StudC_stock_data_stage.csv*.

To proceed for answering the research questions, a merged “master” file is created by combining the data of student C (*Tran_Dao_StudC_stock_data_stage.csv*) and student A (*Biber_Martin_StudentA_tickers_sp500_stage.csv* and *Biber_Martin_StudentA_sp500_ESG_stage*). The “master” file is named as *Tran_Dao_StudC_combined_stock_data_stage.csv* which are data of stock prices per ticker, their sectors and their ESG scores with other information. Cleaning and enrichments steps are written in the script “Tran_Dao_StudC_transform_code.ipynb”. From the combined data of student A and C, several enriched data would be created for student B to scrape more data and finally answer our research question. In order for the readers finding it easier to understand our project, we prefer to separate the codes for answering 4 questions in 4 corresponding files: "Tran_Dao_StudC_Question_01.ipynb", "Tran_Dao_StudC_Question_02.ipynb", ,"Tran_Dao_StudC_Question_03.ipynb", "Tran_Dao_StudC_Question_04.ipynb".

For question 1 and 2, there are several initial transformation steps for each question to limit the ticker lists that student B would use for scraping the data. The data of student C would be used to answer the completed questions.

Question 1:

- From the master file, we focus on 74 members of S&P 500 who belong to “Information Technology” sector.
- A stock’s performance is calculated by using the closing price on the 31st of December, 2020 and on the 31st of October, 2021, calculating how much it changed and save it as *Change* values.
- Average performing for S&P500 'Information Technology' is the mean of all values *Change* of all stocks.
- Use that result to categorize the technology stocks with "underperform" and "outperform"
- Rank the tickers based on their "ESG Score" and store the rank in a new column titled 'ESG_Rank' which means the lowest ESG score corresponds to the best rank.
- Sort values to find the 5 best ranked ESG stocks, then save it as *Tran_Dao_StudC_top5_esg_stage.csv* and send that file to student B for her work.

Question 2:

- Calculate the performance for all 505 members of S&P 500 by calculating how much “Close” price changed and save it in *Change* column.
- Average performing for S&P500 all members is the mean of all values *Change* of all stocks.
- Find the 5 best performing S&P 500 stocks in 2021, then save it as *Tran_Dao_StudC_top5_sp500_stage.csv* and send that file to student B for her work.

3.4.3 Student B

3.4.3.1 Data Extraction

Student B scraped data concerning insider transactions regarding the 5 best ranked ESG companies within the technology sector (*Tran_Dao_StudC_top5_esg_stage.csv*) and the top 5 companies with the highest price appreciation within all sectors (*Tran_Dao_StudC_top5_sp500_stage.csv*). Therefore, those files were provided by student C. The code was applied to both input files separately, resulting in two separate output files (*Tallner_Kirsten_StudB_top5_esg_src.csv* and *Tallner_Kirsten_StudB_top5_sp500_src.csv*). Note, that for setting up the source codes, dummy data (*Tallner_Kirsten_StudB_tickers_dummyData.csv*) was used, because the real data was just recently provided by my teammate. The column containing the tickers, that was addressed in the code was adjusted when the real data files were applied.
All data was scraped from sec.gov. This webpage required interaction such as filling up search fields or pressing buttons so that usage of the selenium package was involved (see Figure 2). Scraping of the webpage's source content itself was performed by usage of beautifulsoup4 package.

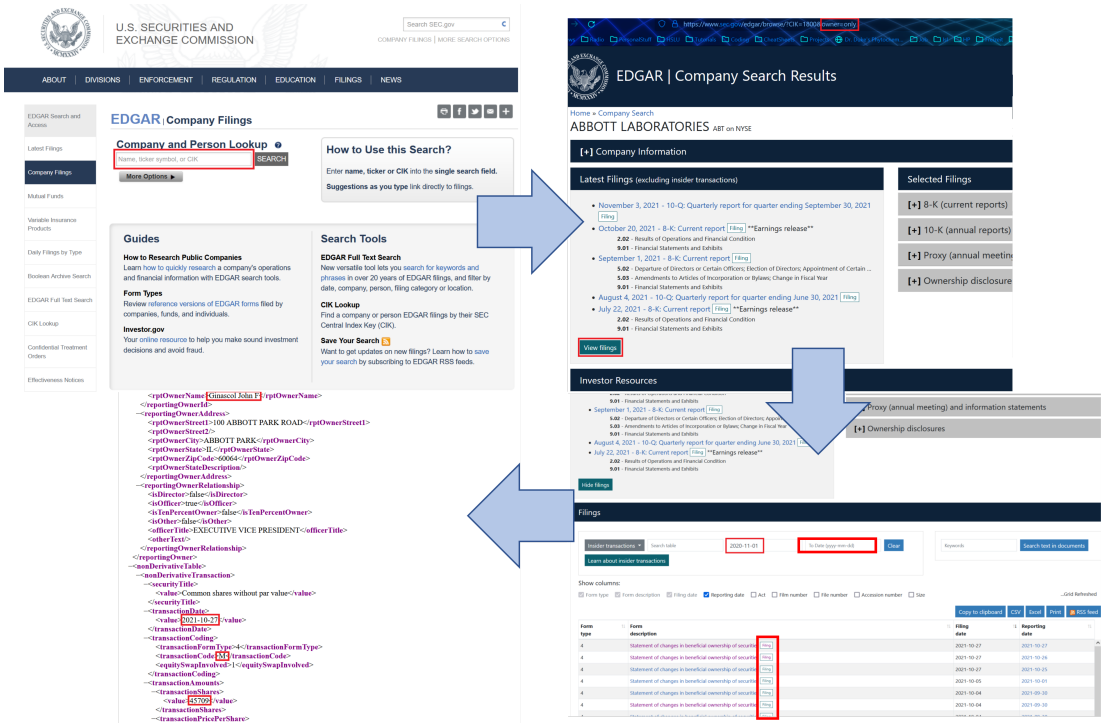


Figure 2: Workflow overview of the scraping process. The red boxes highlight the place of selenium interaction (top left and right, bottom right) or the data of interest to be scraped (bottom left). The step where the xml file is directly accesible to be scraped is not displayed here.

The following actions were taken to access the data of interest:

1. Filling up of the company search field with a ticker and press enter key (e.g. "KEYS") using selenium. Problems arising here and suitable work-arounds were the following:
   i. Sometimes, no overview containing exact or similar letter codes in companies appeared (see Figure 3). Pressing the enter key did not lead to the desired next page containing the filing information for the ticker of interest. When running the code again at a later point in time, it worked without any problems, indicating that the behavior of the website itself changed. This also could be reproduced manually, navigating on the webpage using other browsers (without running the script for web scraping) with cache being emptied before. The exact background for this behavior remains unknown.
   ii. The “more options” button was clicked and the radio button for “ownership forms 3, 4, and 5” was set from “Exclude” to “Only” because only form type 4 documents were of interest. However, when the search was initialized, you were led to the next page with “owner=exclude” as part of the url, indicating that the constraint via radio button did not work properly on the website. Instead, clicking the “more options” button and setting the radio button were skipped and the url was directly adapted to “owner=only” after pressing the enter key (see also action 2).
   iii. Different approaches were tried to start the search: Pressing the “search” button, pressing the return key and pressing the enter key. Only pressing the enter key allowed for the overview containing exact or similar letter codes in companies to show up. Only when this appeared, you were led to the next page of interest.
   iv. Pop-up window for feedback was popping up at various points. Several *try* and *except* statements were implemented to click this pop-up-window away. Further measures (to avoid running the whole web scraping procedure again) were taken as described in action 2i and action 4.

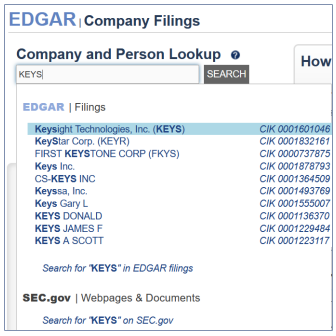
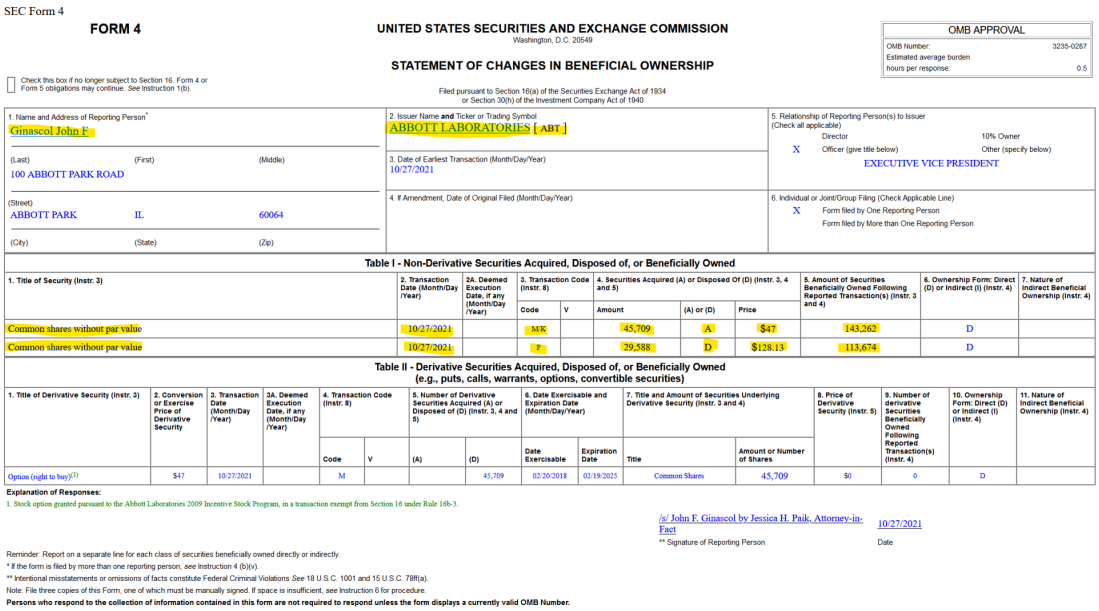


Figure 3: Overwiew containing exact or similar letter codes in companies as typed into field.

2. Refining the search by defining the time span and clicking a button to view filings. For all of those interactions, selenium package was used.
   i. Changing the search attributes within the url search attributes from “owner=exclude” to “owner=only” in order to get access to insider transaction documents (see action1). Those urls containing the “only” argument were written to the file “onlyURL_list.csv” so that you could skip the initial steps when running the script again (e.g. in case there was an interruption before).
   ii. Refining the search result to documents within a certain time span. This required at first the pressing of the “View Filings” button, which enabled further search fields to show up. The ones for “From date” and “To date” were filled accordingly to access documents from Jan 01 2021 until Oct 31 2021. An exemplary website can be found [here](https://www.sec.gov/edgar/browse/?CIK=1601046&owner=only) (please click the "view filings" button to follow).

3. Scraping of filings table (html format) by iterating through it. Therefore, beautifulsoup4 package was used. Only urls linked to type form type 4 documents were scraped. Those urls were appended to a list which was later iterated through to call those urls. When those urls were accessed, links to separate actual form type 4 documents (for exemplary html version see Figure 4) were available in different formats. As filing format of interest, xml was chosen so that only links containing the extension “xml” were pursued in the following.



SEC Form 4

FORM 4

UNITED STATES SECURITIES AND EXCHANGE COMMISSION
Washington, D.C. 20549

STATEMENT OF CHANGES IN BENEFICIAL OWNERSHIP

Filed pursuant to Section 16(a) of the Securities Exchange Act of 1934 or Section 30(h) of the Investment Company Act of 1940

OMB APPROVAL	
OMB Number:	3235-0287
Estimated average burden hours per response:	0.5

☐ Check this box if no longer subject to Section 16. Form 4 or Form 5 obligations may continue. *See* Instruction 1(b).

1. Name and Address of Reporting Person* Ginascol John F (Last) (First) (Middle) 100 ABBOTT PARK ROAD (Street) ABBOTT PARK IL 60064 (City) (State) (Zip)	2. Issuer Name and Ticker or Trading Symbol ABBOTT LABORATORIES [ABT] 3. Date of Earliest Transaction (Month/Day/Year) 10/27/2021 4. If Amendment, Date of Original Filed (Month/Day/Year)	5. Relationship of Reporting Person(s) to Issuer (Check all applicable) Director 10% Owner X Officer (give title below) Other (specify below) EXECUTIVE VICE PRESIDENT 6. Individual or Joint/Group Filing (Check Applicable Line) X Form filed by One Reporting Person Form filed by More than One Reporting Person

Table I - Non-Derivative Securities Acquired, Disposed of, or Beneficially Owned

1. Title of Security (Instr. 3)	2. Transaction Date (Month/Day /Year)	2A. Deemed Execution Date, if any (Month/Day /Year)	3. Transaction Code (Instr. 8) Code	V	4. Securities Acquired (A) or Disposed Of (D) (Instr. 3, 4 and 5) Amount	(A) or (D)	Price	5. Amount of Securities Beneficially Owned Following Reported Transaction(s) (Instr. 3 and 4)	6. Ownership Form: Direct (D) or Indirect (I) (Instr. 4)	7. Nature of Indirect Beneficial Ownership (Instr. 4)
Common shares without par value	10/27/2021		M/K		45,709	A	$47	143,262	D	
Common shares without par value	10/27/2021		F		29,588	D	$128.13	113,674	D	

Table II - Derivative Securities Acquired, Disposed of, or Beneficially Owned (e.g., puts, calls, warrants, options, convertible securities)

1. Title of Derivative Security (Instr. 3)	2. Conversion or Exercise Price of Derivative Security	3. Transaction Date (Month/Day /Year)	3A. Deemed Execution Date, if any (Month/Day /Year)	4. Transaction Code (Instr. 8) Code	V	5. Number of Derivative Securities Acquired (A) or Disposed of (D) (Instr. 3, 4 and 5) (A)	(D)	6. Date Exercisable and Expiration Date (Month/Day/Year) Date Exercisable	Expiration Date	7. Title and Amount of Securities Underlying Derivative Security (Instr. 3 and 4) Title	Amount or Number of Shares	8. Price of Derivative Security (Instr. 5)	9. Number of derivative Securities Beneficially Owned Following Reported Transaction(s) (Instr. 4)	10. Ownership Form: Direct (D) or Indirect (I) (Instr. 4)	11. Nature of Indirect Beneficial Ownership (Instr. 4)
Option (right to buy)(1)	$47	10/27/2021		M			45,709	02/20/2018	02/19/2025	Common Shares	45,709	$0	0	D	

Explanation of Responses:
1. Stock option granted pursuant to the Abbott Laboratories 2009 Incentive Stock Program, in a transaction exempt from Section 16 under Rule 16b-3.

/s/ John F. Ginascol by Jessica H. Paik, Attorney-in-Fact — 10/27/2021
** Signature of Reporting Person — Date

Reminder: Report on a separate line for each class of securities beneficially owned directly or indirectly.
* If the form is filed by more than one reporting person, *see* Instruction 4 (b)(v).
** Intentional misstatements or omissions of facts constitute Federal Criminal Violations *See* 18 U.S.C. 1001 and 15 U.S.C. 78ff(a).
Note: File three copies of this Form, one of which must be manually signed. If space is insufficient, *see* Instruction 6 for procedure.
Persons who respond to the collection of information contained in this form are not required to respond unless the form displays a currently valid OMB Number.

Figure 4: An exemplary form 4 document. The data of interest which was to be scraped is highlighted in yellow.

4. Accessing those xml related urls and scraping the data of interest:
   reporting person, issuer, issuer trading symbol, security title, transaction date, transaction code, transaction shares, transaction price per share, transaction acquired or disposed code, shares following transaction. The data was written into a data frame and saved as csv file. Moreover, the raw data was also transferred directly to the database *CIP* on MariaDB.
   Note: Because there appeared several interruptions like pop-up windows (see action 1i) or strange behavior of the website to be scraped from (see action 1iv), an additional step was implemented: The xml files were saved in the folder "xml_documents". So in case you had to restart running the code again, the program checked if the respective xml files were already located in the *xml_documents* folder and did not further process the whole scraping procedure again. Then the data was read into the program and scraped using beautifulsoup4 package.

3.4.3.2 Data Transformation

1. Dirty data: One of the two datasets was made dirty in order to apply data cleaning via python's pandas module. Instead of doing this manually, a jupyter notebook was created (*Tallner_Kirsten_StudB_src_makeDirty__top5_esg_src_dirty.ipynb*) in order to follow the procedure step by step. Running this notebook implemented various changes in several columns randomly. The resulting file (*Tallner_Kirsten_StudB_top5_esg_src_dirty.csv*) was saved before reading it into the next script for cleaning. Note, that the raw file itself was not totally clean before but was now expanded with further mistakes.

2. Cleaning of data: Both datasets were cleaned by running the code *Tallner_Kirsten_StudB_cleanUp_stage.py*. Amongst other things, it checked if numbers were given in the desired format using regex. It furthermore eliminated NaNs in both, the dirty and the raw datasets. The resulting cleaned datasets were saved (*Tallner_Kirsten_StudB_top5_esg_stage.csv* and *Tallner_Kirsten_StudB_top5_sp500_stage.csv*) and provided to student C for further processing to prepare for answering the project questions. Note, that there is also a jupyter notebook (*Tallner_Kirsten_StudB_cleanUp__top5_esg_stage.ipynb*) available to follow the clean-up of the dirty dataset. In principle, it contains the same code but without functions. Because the cleaning should later be applied to two files, the python file including functions was created.

3.4.4 Student C: Data Transformation (Phase 2)

After receiving the data of student B about insider transaction for each ticker on the SEC/EDGAR site. There are a few transformation steps should be done to finalize the answer for research questions 1 and 2:

Question 1:

- After receiving the output data scraped by student B (*Tallner_Kirsten_StudB_top5_esg_stage.csv*) about insider transaction data through scraping on the SEC/EDGAR site, the difference between acquired shares (column "A") and disposed shares (column "D") can be calculated in order to find out how much shares each person really exchanged in transaction.
- There are some missing values in column "A" means that this person has no acquired transactions. Similarly, some missing values in column "D" means that this person has no disposed transactions.
- If the value of "Difference" is positive, it means the corresponding person had acquired shares more than disposed shares. If the value of "Difference" is negative, it means the corresponding person had disposed shares more than acquired shares.
- To answer the research question, need to find out who has the highest positive 'Difference' between acquired and disposed.
- The answer question 1 is stored in *Result_Question_01.xlsx*.

Question 2:

- The input of student B (*Tallner_Kirsten_StudB_top5_sp500_stage.csv*) is used to calculated the difference between acquired shares (column "A") and disposed shares (column "D") to find out how much shares each person really exchanged in transaction.
- If the value of "Difference" is positve, it means the corresponding person had acquired shares more than disposed shares.
  => The largest buyer will be the person who has the highest value of 'Difference' between acquired and disposed.
- If the value of "Difference" is negative, it means the corresponding person had disposed shares more than acquired shares.
  => The largest seller will be the person who has the lowest value of 'Difference' between acquired and disposed.
- The answer question 2 is stored in *Result_Question_02.xlsx*.

For question 3 and 4, those questions can be answered based on the data of student A and student C.

Question 3:

- The volume data for the S&P 500 members and the non-members are in *Tran_Dao_StudC_combined_stock_data_stage.csv*. The trading volume per each group is defined by the product of “Volume” and “Close”, and the result is stored in a new column “Trading_Volume”.
- Then, we use *groupby()* to aggregate the average value of “Trading_Volume” per each group.
- Comparing the 2 values of average “Trading_Volume” and concluding which group has the higher volume on average.
- The answer of question 3 is stored in *Result_Question_03.xlsx*

Question 4:

- The dataset (created in Question 2) named *Tran_Dao_StudC_sp500_ranking_stage.csv* includes the data about S&P 500 performance ranking and their ESG scores. Based on this dataset, the stocks are ranked based on “ESG Score” and top 20 best ranking ESG are selected.
- Similarly how to calculate the performance of stocks (mentioned in Question 1 and 2), the performance of “SPY” is calculated as well from 2020-12-31 to 2021-10-29. The used data of "SPY" is in *Tran_Dao_StudC_combined_stock_data_stage.csv*.Then, SPY is used as a benchmark to know which stocks have higher performance than S&P 500.
- Filter the list of 20 best stocks by using *.query()* function to check which one has a higher “Change” value than “SPY”.
- The answer of question 4 is stored in *Result_Question_04.xlsx*

3.5 Answers to our Questions

Question 1: Who are the three largest buyers in the five best ranking ESG Total Score of the above sector average performing S&P500 Technology stocks in 2021?

The 5 best ranked ESG stocks in Technology sector in 2021 are: *Keysight Technologies, Inc.* (KEYS), *CDW Corp* (CDW), *HP INC* (HPQ), *Accenture plc* (ACN), *Seagate Technology Holdings plc* (STX).

Answer: The 3 largest buyers in the five best ranking ESG Total Score of the above sector average performing S&P500 Technology stocks in 2021 are (see Figure 5): *Schell Christoph* (HPQ), *Myers Marie* (HPQ), *Nygaard Jeffrey D.* (STX)

Sector	Ticker	2020-12-31	2021-10-29	Change	Performance	ESG Score	ESG_Rank
Information Technology	KEYS	132.089996	180.020004	0.362859	Outperform	9.0	1.0
Information Technology	CDW	131.789993	186.649994	0.416268	Outperform	9.0	1.0
Information Technology	HPQ	24.590000	30.330000	0.233428	Outperform	10.0	2.0
Information Technology	ACN	261.209992	358.790008	0.373569	Outperform	11.0	3.0
Information Technology	STX	62.160000	89.070000	0.432915	Outperform	11.0	3.0

Person	Company	Ticker	A	D	Difference
Schell Christoph	HP INC	HPQ	84346	0	84346
Myers Marie	HP INC	HPQ	33592	8454	25138
Nygaard Jeffrey D.	Seagate Technology Holdings plc	STX	21344	6893	14451

Figure 5: Answer of Question 1

Question 2: Who are the three largest buyers and sellers in the five best performing S&P stocks in 2021?
The 5 best performing S&P stocks in 2021 are: *Moderna, Inc.* (MRNA), *DEVON ENERGY CORP/DE* (DVN), *MARATHON OIL CORP* (MRO), *Bath & Body Works, Inc.* (BBWI), *Fortinet, Inc.* (FTNT)

Answers:

- The 3 largest buyers in the 5 best performing S&P stocks in 2021 are (see Figure 6, middle): *Whitehead Dane E* (MRO); *Tillman Lee M* (MRO); *Harris David Gerard* (DVN).
- The 3 largest sellers in the 5 best performing S&P stocks in 2021 are (see Figure 6, bottom): *Felix Investments Holdings Ii, Llc* (DVN), *Afeyan Noubar* (MRNA), *Bancel Stephane* (MRNA).

Ticker	ESGtotal	2020-12-31 00:00:00	2021-10-29 00:00:00	Change	Performance	Perform_Rank
MRNA	NaN	104.470001	345.209992	2.304393	Outperform	1
DVN	32.0	15.810000	40.080002	1.535104	Outperform	2
MRO	43.0	6.670000	16.320000	1.446777	Outperform	3
BBWI	16.0	30.064672	69.089996	1.298046	Outperform	4
FTNT	20.0	148.529999	336.339996	1.264458	Outperform	5

Top 3 largest buyers

Person	Company	Ticker	A	D	Difference
Whitehead Dane E	MARATHON OIL CORP	MRO	83705	16902	66803
Tillman Lee M	MARATHON OIL CORP	MRO	479857	414075	65782
Harris David Gerard	DEVON ENERGY CORP/DE	DVN	67455	19528	47927

Top 3 largest sellers

Person	Company	Ticker	A	D	Difference
Felix Investments Holdings Ii, Llc	DEVON ENERGY CORP/DE	DVN	0	29579389	-29579389
Afeyan Noubar	Moderna, Inc.	MRNA	0	10290906	-10290906
Bancel Stephane	Moderna, Inc.	MRNA	0	1070028	-1070028

Figure 6: Answer of Question 2

Question 3: Do the S&P 500 members or the 500 most active non-members have the higher volume on average? (Activity is measured by the trading volume as of October 31st).

Answer: Average trading volume of S&P 500 members is around 607’388’227.8 and average trading volume of S&P 500 non-members is around 217’894’056.1 (see Figure 7). The returned result shows that the S&P 500 members have higher volume on average compared to the 500 most active non-members.

S&P 500	Trading_Volume mean
member	6.073882e+08
non-member	2.178941e+08

Figure 7: Answer of Question 3

Question 4: How many of the 10 best ranking ESG stocks have a better Performance than the S&P500 (SPY is used as benchmark)?

Answer: SPY is used as a benchmark and its performance rate is 0.228335 from 2020-12-31 to 2021-10-29. Compare the performance of 20 top best stocks to the performance of SPY to have the answer (see Figure 8). Finally, we have 16 of the 20 best ranking ESG stocks have a better Performance than the S&P500.

```
# Select the tickers in the list of best stocks that have higher performance rate of SPY:
result=top20.query('Change > 0.228335')
result
```

	Ticker	2020-12-31 00:00:00	2021-10-29 00:00:00	Change	Performance	Perform_Rank	ESG Score	ESG_Rank
0	CBRE	62.720001	104.080002	0.659439	Outperform	29	7.0	1.0
1	CDW	131.789993	186.649994	0.416268	Outperform	107	9.0	2.0
2	KEYS	132.089996	180.020004	0.362859	Outperform	134	9.0	2.0
3	RHI	62.480000	113.070000	0.809699	Outperform	17	9.0	2.0
4	PLD	99.660004	144.960007	0.454545	Outperform	86	10.0	3.0
5	AVB	160.429993	236.679993	0.475285	Outperform	74	10.0	3.0
7	HPQ	24.590000	30.330000	0.233428	Underperform	234	10.0	3.0
8	CRM	222.529999	299.690002	0.346740	Outperform	145	11.0	4.0
10	ACN	261.209992	358.790008	0.373569	Outperform	129	11.0	4.0
11	EQR	59.279999	86.400002	0.457490	Outperform	83	11.0	4.0
14	ADBE	500.119995	650.359985	0.300408	Outperform	178	11.0	4.0

```
len(result.Ticker.unique())
```

```
16
```

Figure 8: Answer of Question 4

3.6 Loading Data into MariaDB

3.6.1 Student A

The module *mysql.connector* was used in the script "StudentA_L1_createDB.py" to create the database *CIP*. The same module was used
in the script "StudentA_L2_createTables.py" to connect to the database
and create the eleven tables. The third script "StudentA_L3_loadData.py" used the LOAD DATA LOCAL INFILE to import the eleven csv's into
the database.

3.6.2 Student B

Within my virtual machine, the database *CIP* and two tables within it were created (see SQL statements in *setUpTables_MariaDB.txt*). Because not all definitions in your pdfs regarding the module deliverables were absolutely clear when setting up the database, also the dirty dataset was transferred to MariaDB. The respective table *top5_esg_src_dirty* was not unified with raw and cleaned data because of differences in column-specific data types. Instead of loading each dataset into separate tables, only one was created for all data except the dirty one, containing all raw and cleaned data. Therefore, also an additional column for the kind of data (called *category*) was introduced which later allows for addressing the data of interest (e.g. top5_esg_src, top5_esg_stage) within SQL's SELECT FROM WHERE statement. Null values of dirty (*top5_esg_src_dirty*) and raw data (*top5_esg_src.csv* and *top5_sp500_src.csv*) were set to 0 when data were inserted into the tables.

For the data import itself, a python script (*mariaDB_util.py*) was written, which was called from *Tallner_Kirsten_StudB_cleanUp_stage.py* for the raw data as well as from *Tallner_Kirsten_StudB_cleanUp__top5_esg_stage.py* for dirty and cleaned data. It uses the *pymysql* module and inserts the data via INSERT INTO statements. To avoid adding the same data again when running the code several times, a function for deleting the entries via DELETE FROM statements before inserting the data again was implemented.

3.6.3 Student C

The database named "CIP" was created in MariaDB by using python module *pymysql* and details of creating steps refer to the script "Tran_Dao_StudC_createDB_mariaDB.ipynb". Then, module *sqlalchemy* is used to write Pandas dataframes into tables in "CIP" database, 13 neccessary tables containing all data of student C and answers of group's questions were created and loaded successfully. Furthermore, there are some examples to read data from "CIP" database. The script "Tran_Dao_StudC_loadData_mariaDB.ipynb" would describe in more details how the loading phase is.

4. Lessons Learned and Self-Reflection

During this project, we enhanced our technical skills in data collection, integration and preprocessing. The material provided by the lecturers was helpful in the beginning. Additional and helpful information could also be obtained from the internet. We got familiar using new python libraries like e.g. yfinance, BeautifulSoup or Selenium to extract the data of interest. Because the web scraped or downloaded data was not always clean, we were diving deep into data manipulation techniques using python's pandas package, which provided a really useful tool with various functionality (in data processing and analysis) to answer our research questions.

We noticed that it was not easy to scrape data from dynamic websites, especially when several challenges like pop-up windows appear from time to time or website maintenance takes place. One of the most challenging parts using python's selenium package was to implement the waiting time for the website to be loaded.

It was also very interesting to create and use a database without the help of a graphical user interface.

For our collaboration, we used a shared repository on GitHub. This facilitated parallel workflows as well as project management and gave us another opportunity to learn a tool that is widely used in the data science community.

5. Consent

We would like to keep our project private and not to provide it to everybody as open source. However, when somebody has an interest in our project, we are happy to provide our project material directly to that person.