Seminar in Business Analytics: Webscraper Design and Data Analysis using R and Python

– Winter Term 2021-22 –

Scraping Insider Trading Information from finanzen.net

Submitted by:
Mr. Stanley George
Student-ID:
4974121
Advisor:
Mr. Jannik Schäfer

Content

1. Introduction

Stock Markets are today an important entity in the global economic ecosystem. The 20th and the 21st centuries are filled with many events where a major stock market collapse led to a countries' economic downfall even at times leading to a global financial crisis.

A stock is a type of security that represents an individual's ownership in a company and a stock market is a place where such assets can be brought or sold by investors. Trading of stocks on such a public platform is beneficial to the economic growth of a company and such public trading brings in more investors and thereby more capital for the companies' growth.

Many companies find it beneficial to trade on such public platforms as it enables them to make capital for expansion without need to apply for a bank loan. A bank loan would simply bring in the need to pay added interest rates and the need to keep a collateral. On the other hand, if a company decides to trade one million shares on the public market worth 10 euros each, they can raise a capital of 10 million euros.

The stock market is also beneficial to the buyer (also known as the investor) wherein the person receives regular dividends – i.e., a share of the company's profits on a regular basis.

However, where there is a system that regulates a flow of money, there are often loopholes that can be exploited by the company or the investor to make additional fraudulent gain. Some such techniques include embezzlement by stockbrokers, incorrect data on company's financial statements released to public, lying to corporate auditors, insider trading and many more. Countries have hence set up regulatory agencies (e.g., BaFin in Germany, SEBI in India) which regulate the trades happening at the stock markets.

In this Seminar, we would be focusing on a stock-market activity called 'Insider Trading' which are strictly monitored by the government agencies. An Insider is any person with sufficient information that can impact the stock price of the company when the information gets revealed in the Public. However, not all insider trading is fraudulent and sometimes can be useful to a stock-trader to predict a stock or company's future by analysing such Insider Trades.

As part of this project, we have created a real-time database that would capture the insider trades that have happened on the stocks that are part of the DAX40 index. The data is extracted using a Webscraper from finanzen.net, a popular German Finance portal which alongside many other data also contains such insider trades happening on a listed stock. DAX40 is a collection of 40 major German blue-chip companies that are traded at the Frankfurt Stock Exchange.

2. Insider Trading

In September 2021, a Frankfurt court sentenced a former employee of Union Investment, Germany's third-largest asset manager, to a three-year jail term and a fine of €45M from him for a large-scale insider trading. His friend to whom was also occasionally tipped off also was sentenced to a two-year imprisonment along with a heavy fine. (Financial Times, 2022)

As per the German Securities Watchdog BaFin, Insiders are persons who have knowledge of facts relating to listed companies that are not in the public domain and that could potentially have a significant impact on the share price of these companies, for instance because these persons have come into possession of this inside information in a professional capacity (Insider surveillance, 2022).

However, these trades can also be declared legally through BaFin and gets published on public websites. A potential investor can use tracking tools such as the one being developed in this project to keep himself informed of the market happenings. There can be scenarios where a top management executive decides to sell off some assets citing some legal backdoors but in reality might be selling it off as he/she is less confident about the company's future prospects.

3. Methods

In this section we would explore whole project ecosystem (Fig. 1) from a technical viewpoint.

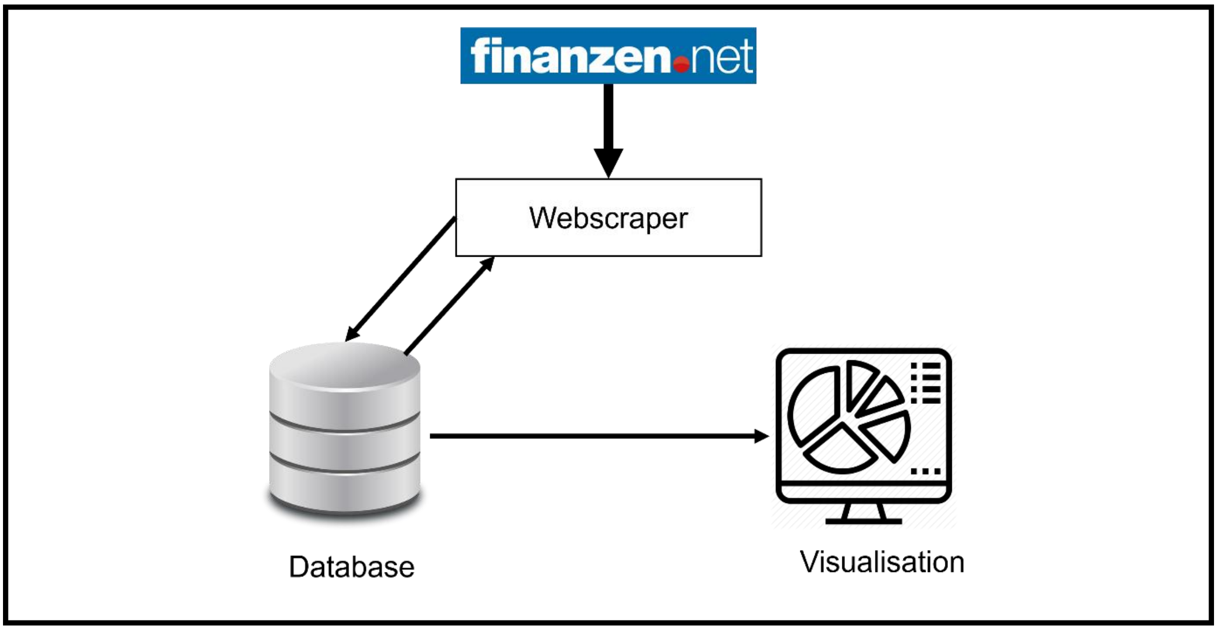


Figure 1: Project Technical Ecosystem.

3.1 Webscraper

A Webscraper is a software program (also referred to as web-crawler or bot) that’s designed specifically to extract (or ‘scrape’) relevant information from websites. (Kenny, 2022) Today, many python-based scraping libraries are available. For this project, we would be using a library called Scrapy.

We would be using three crawlers, each serving a purpose of its own as listed below (Table 1):

Crawler name	Tasks	Frequency
companies_spider	• Extracts the list of all DAX40 companies along with the weblink to obtain their insider trade data. • Store these data in the database table **companies**	One-time
insider_spider	• Fetches the insider trades for the weblinks present in the database table **companies**. • Store these data in the database table **trades**.	One-time
update_spider	• Updates the database for the insider trades reported since the last script execution.	Daily once

Table 1 : List of Scrapy spiders along with their tasks and frequency.

In the next sub-section, we would be discussing briefly about the Scrapy package in brief covering some important classes, its overall architecture and finally some advantages and disadvantages.

3.1.1 Scrapy – Overview

In this section, we would be discussing about Scrapy in depth. Scrapy is an application framework for crawling web sites and extracting structured data which can be used for a wide range of useful applications, like data mining, information processing or historical archival. Scrapy is available as an open-source package and is being actively maintained by some companies working in the web-scraping domain.

3.1.2 Scrapy – the Spider class

At the heart of a Webscraper built using the Scrapy library lies a Spider. A Spider is a custom class written by the Scrapy user to parse HTML or API responses and extract items from them or the subsequent actions to be performed. It is here one defines what to scrape (i.e. scraping items) in a webpage and how to crawl though the website (i.e. follow the links). To generate a custom Spider class, you need to inherit the parent Spider class *scrapy.Spider*.

You can configure multiple spiders to run within your project – each with a distinct purpose. The spiders built as part of this project are listed in Table 1.

3.1.3 Scrapy - Architecture

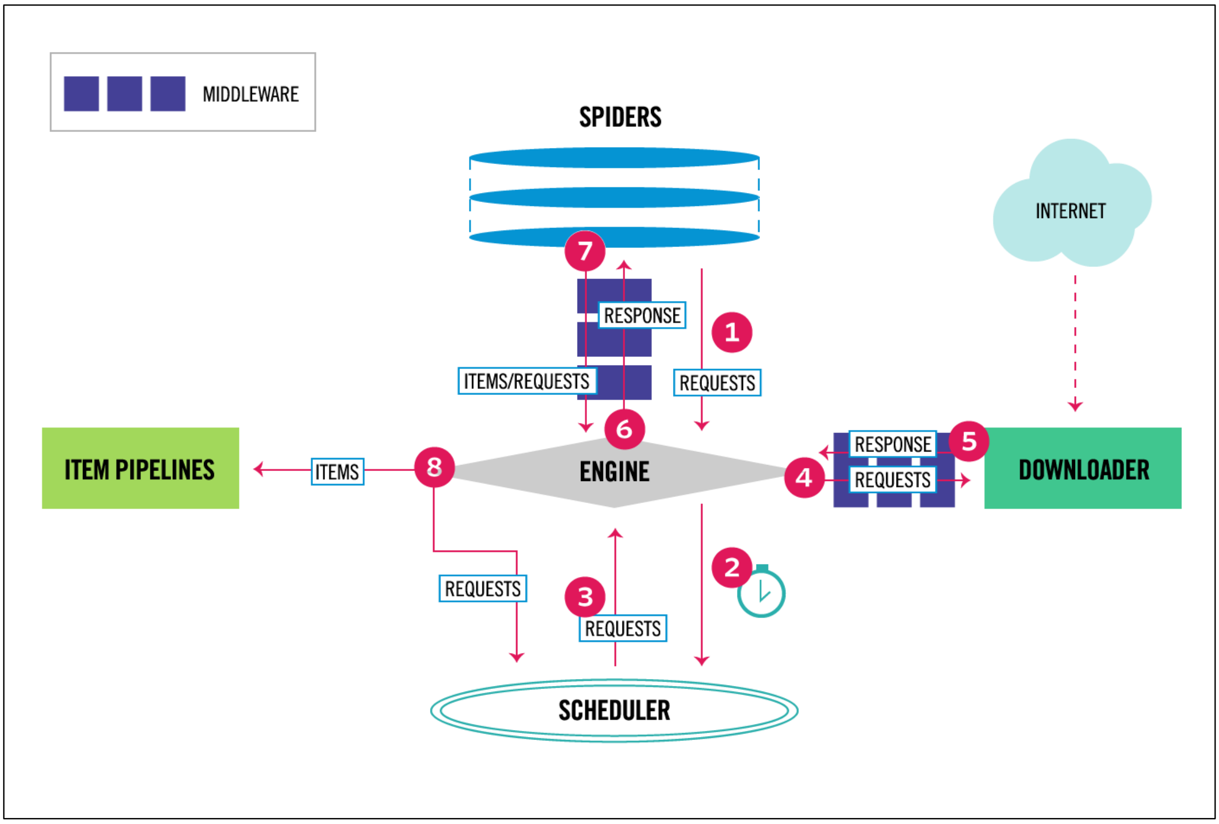


Figure 2: Scrapy Architecture (Architecture overview, 2022)

Fig. 2 shows a concise architecture and the data-flow pattern of a typical Scrapy program from Scrapy's documentation webpage. The table below is a brief overview of each of the entities listed in the architecture diagram followed by the dataflow happening between different entities.

Entity	Description
Engine	Responsible for controlling the data flow. Also handles the job of triggering events when certain actions occur.
Spider	Custom classes written by the user to defining what to scrape, how to scrape and what to do next after scraping (i.e. link following)
Scheduler	Maintains a queue of Requests. These Requests are sent by the engine and are also fed back to the engine when asked.
Downloader	Responsible for fetching webpages and feeding them to the engine which in turn feeds them to the spiders.

Spider Middlewares	These are special hooks that exist between the Engine and the Spiders which can be useful to process the spider input (Responses) and spider output (items and requests). Can be configured in the file ***middlewares.py.***
Downloader Middlewares	These are special hooks that exist between the Engine and the Downloader that can be used to perform additional actions on the data packets that flow in between those two entities. Can be configured in the file ***middlewares.py***.
Item Pipeline	It is responsible for processing the items once they have been extracted by the spiders for e.g. cleansing, database storage. Can be configured in the file ***pipelines.py***
Items	They are outputted by a Spider after it has scraped a webpage. They are a structured collection of the scraped data usually in key-value (Dictionary) format. The items can be defined in the ***items.py*** file.
Request	They are generated by spiders and is passed across the system until it reaches the Downloader to create a Response object.
Response	A Response object represents an HTTP response, which is usually downloaded (by the Downloader) and fed to the Spiders for processing.

The dataflow of a Webscraper based on Scrapy is as follows (Refer Figure 2).

1. The Engine gets the initial request to crawl from the Spider.
2. The Engine schedules this Request in the Scheduler and subsequently asks for the next Requests to crawl.
3. The Scheduler returns the next in queue Requests to the Engine.
4. The Engine sends the Requests to the Downloader, passing through the Downloader Middleware.
5. Once the page is downloaded, the Downloader generates a Response and sends it to the Engine, passing through the Downloader Middleware.
6. The Response is sent to the Spider, passing through the Spider Middleware.
7. The Spider processes the Response and returns Items and new Requests (i.e. new links to follow) to the Engine passing through the Spider Middleware.
8. The Engine sends the Items to the Item Pipelines for further handling. Simultaneously, it also sends the new Requests to the Scheduler.
9. The process repeats (from Step 1) until there are no more Requests left in the Scheduler.

3.1.4 Navigating through a webpage

A single webpage of a website is usually made up of many components apart from the relevant one is looking for (e.g., Hyperlinks to other webpages, company specific Headers and Footers, widgets etc.) Given a weblink, we would not like the Webscraper to extract all these unwanted information but only the specific content that we are looking for. Hence, we make use of the HTML and CSS structure components that defines the underlying structure of any webpage on the internet. A useful tool that was used to aid in faster recognition of such specific elements was the SelectorGadget software which is available as an extension on Google Chrome Web Browser.

Consider the webpage of Adidas' Insider Trading which contains many different data sections. The target data is lying at the middle of the page in a tabular format. Using the SelectorGadget tool, we can see that the XPath of this component is *//*[@class="col-sm-8"]//tr*. This XPath is given to the Webscraper program to selectively process only the given section of a webpage.

3.2 Database

In this project we have used MySQL database which is an open-source relational database management system (RDBMS). Since the data being scraped was well organised in tabular formats and no existing as free-text, the author decided to go for a Relational DBMS instead of a NoSQL based DBMS. Among the popular RDMS systems, MySQL and PostgreSQL are the most popular ones and are the ones with best community support. They both also have good integration with many free hosting services to deploy our crawler application. Since, both weighed almost equally, it was decided to carry out all experiments using MySQL.

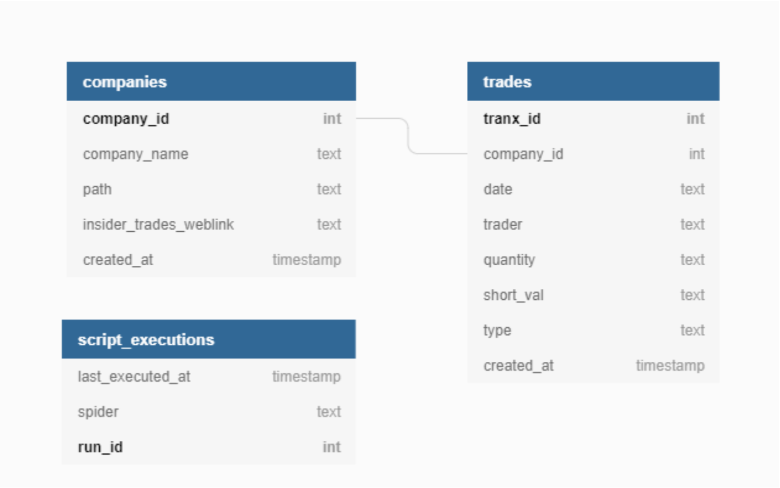


Figure 3: ER diagram of database

Fig.2 show the ER diagram for the database. The *companies* table houses the scraped list of all 40 companies listed on DAX40 index along with the weblink to access the insider trades information on the finanzen.net portal. The *trades* table contains the scraped list of all insider trading information. The field *company_id* is a foreign key that references the *company_id* field of the *companies* table. Finally, we have a special metadata table named *script_executions* which stores the last run instance of any spider (This is the reason for the bidirectional arrow in Figure 1).

3.3 Reporting

The main purpose of configuring a Webscraper and creating a database is to derive useful insights from the scraped data. All kind of data level analysis can be done any MySQL query tool by connecting to our database (like MySQL Workbench, Oracle SQL Developer etc.). We have also set up some real-time visualisation using Dash which is a popular python library for creating data visualisations.

The reporting dashboard connects in real-time to the MySQL database. It contains a bar-graph with companies on X-axis and the number of insider trades reported for the time-period selected using the dropdown given at the top. The user can also click on any individual bar. This would display a tabular detailed data of all the transactions that were accounted for under that bar. An 'Export' button enables the user to download the displayed tabular data in .xlsx format. Refer Appendix 7.3 for some images and a demo video

4. Observations

The data can be analysed and inspected from different angles.

- A total of 2934 insider trades were captured by the Webscraper. (During the analysis, it was observed that for a certain company, the website stores only a maximum of 100 insider trades).
- During the last three months (Oct 2021 – Dec 2021), the most insider trading happened at Delivery Hero (Appendix 7.1) (Appendix 7.2). Upon inspecting the heavy volume of trades (both in quantity and value), these transactions have been performed by the CEO and the COO of Delivery Hero. The author assumes this probably has to do with their majority stake acquisition in their Spanish competitor 'Glovo'. (Reuters, 2022) (finanzen.net, 2022). Buying of shares probably indicate the confidence the leadership has of increased revenue in the coming months owing to the acquisition of a market-leader in another country
- The author also observed that many companies' share prices start increasing once some big value insider trade happens. Though both these occurrences might be

unrelated, it can also be interpreted as increased buying interest in market as many people in the market are also keeping track on insider trades to make their investment move

5. Conclusion and Future Work

This project demonstrated how insider trading information can be scraped from the online finance portal finanzen.net. The availability of well coded packages such as Scrapy coupled with visualisation libraries like Dash helps create such dashboards without need for some high-level python coding. However, the data obtained from stock market is often riddled with many uncertainties and the action of high-value purchase shouldn't trigger an investor to invest his hard-earned money too.

A possible limitation of the current method of declaring insider trades is the 2-3 days delay in publishing the purchase. These 2-3 days gap can be potentially useful for a person intended on committing malpractice as he/she can make his trade and impact the market even before the information is made available in public.

The current project is working only for DAX40 companies but can be easily extended to many more indexes with few more additional lines of code. For better stock analysis, the dashboard should also display the stock price variation before and after the trade

6. References

(2022, January 1). Retrieved from Financial Times: https://www.ft.com/content/dbd5ada5-b1f2-497c-81cb-d304b8ab5eb3

(2022, January 1). Retrieved from Reuters: https://www.reuters.com/markets/deals/delivery-hero-acquires-majority-stake-spanish-food-delivery-app-glovo-2021-12-31/

(2022, January 2). Retrieved from finanzen.net: https://www.finanzen.net/nachricht/aktien/zukauf-in-spanien-delivery-hero-uebernimmt-mehrheit-an-liefer-app-glovo-10887992

Architecture overview. (2022, March 04). Retrieved from https://scrapy.org/: https://docs.scrapy.org/en/latest/topics/architecture.html

Insider surveillance. (2022, January 01). Retrieved from bafin.de: https://www.bafin.de/EN/Aufsicht/BoersenMaerkte/Marktmissbrauch/Insiderueberwachung/insiderueberwachung_node_en.html

Kenny, C. (2022, March 03). *What is Web Scraping ?* Retrieved from Zyte.com: https://www.zyte.com/learn/what-is-web-scraping/#What-is-web-scraping?

7. Appendix

7.1 Graph of insider trades over different time periods

The below graphs have been produced by the Dash visualisation interface.

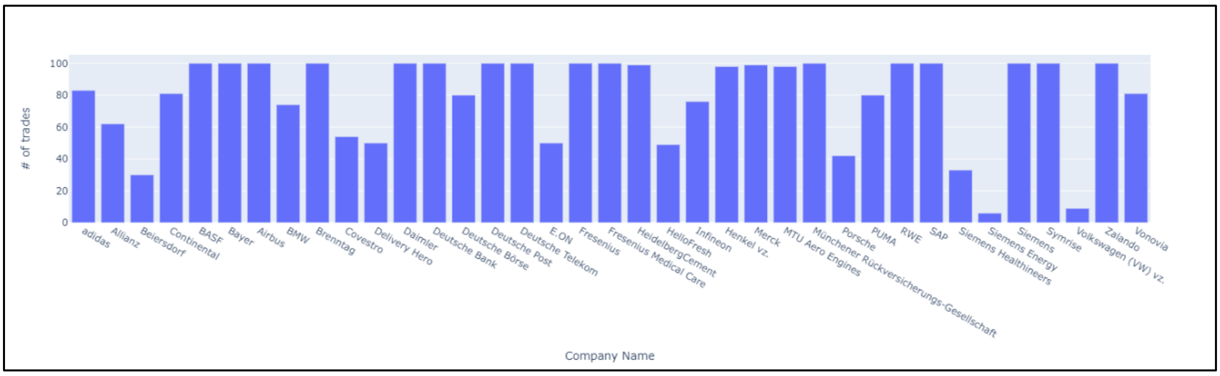


Figure 4 : Number of trades reported over all time

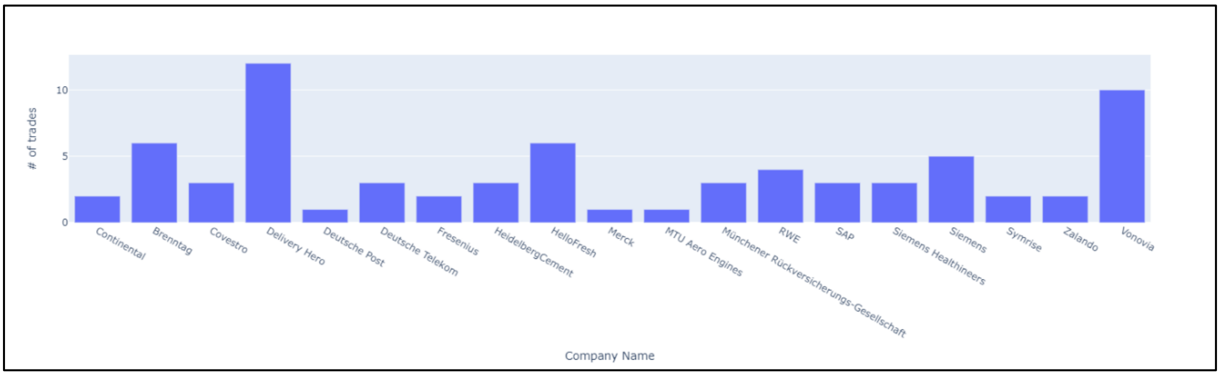


Figure 5 : Number of trades reported for last 3 months

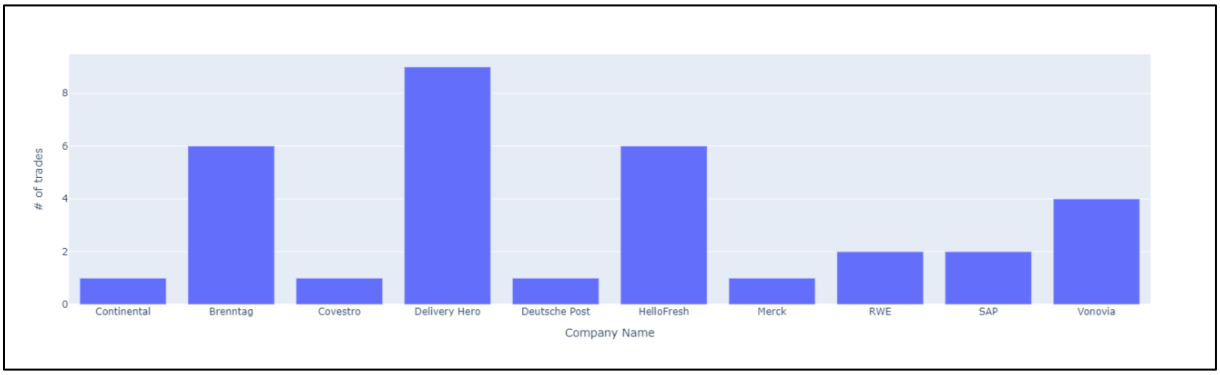


Figure 6 : Number of trades for last month.

7.2 Data Exports

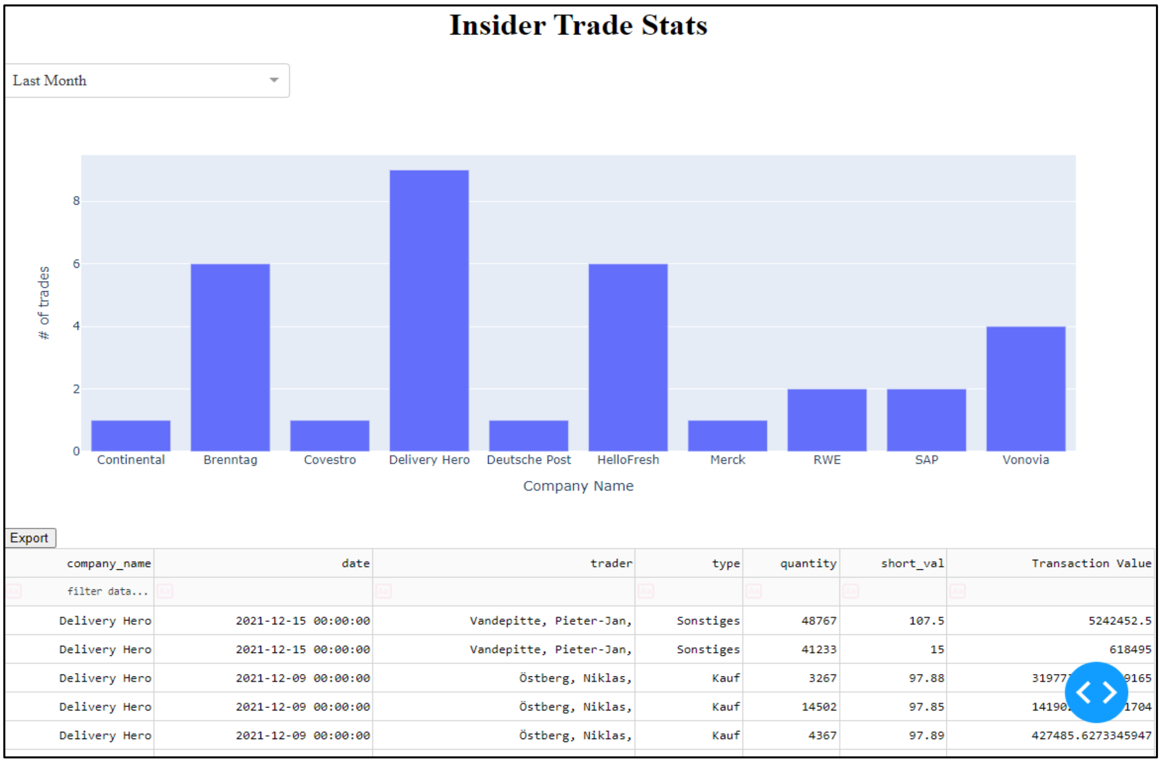
The below exports were taken using the ‘Export’ functionality of the visualisation software.

Company Name	Date	Trader	Type	Quantity	Short Value	Transaction Value
Delivery Hero	2021-12-15	Vandepitte, Pieter-Jan,	Sonstiges	48767	107,5	5242453
Delivery Hero	2021-12-15	Vandepitte, Pieter-Jan,	Sonstiges	41233	15	618495
Delivery Hero	2021-12-09	Östberg, Niklas,	Kauf	3267	97,88	319774
Delivery Hero	2021-12-09	Östberg, Niklas,	Kauf	14502	97,85	1419021
Delivery Hero	2021-12-09	Östberg, Niklas,	Kauf	4367	97,89	427485,6
Delivery Hero	2021-12-09	Östberg, Niklas,	Kauf	1226	97,95	120086,7
Delivery Hero	2021-12-09	Östberg, Niklas,	Kauf	7249	97,75	708589,8
Delivery Hero	2021-12-08	Thomassin, Emmanuel,	Sonstiges	27065	16,67	451173,6
Delivery Hero	2021-12-08	Thomassin, Emmanuel,	Sonstiges	32935	105,83	3485511

Table 2 : Transactions at Delivery Hero in Dec-2021

7.3 Dashboard demo

Below (Fig. 6) is a screenshots of the visualisation dashboard. A video demo has also been published on the following YouTube link : https://youtu.be/vyunvML4Zdg



company_name	date	trader	type	quantity	short_val	Transaction Value
filter data...						
Delivery Hero	2021-12-15 00:00:00	Vandepitte, Pieter-Jan,	Sonstiges	48767	107.5	5242452.5
Delivery Hero	2021-12-15 00:00:00	Vandepitte, Pieter-Jan,	Sonstiges	41233	15	618495
Delivery Hero	2021-12-09 00:00:00	Östberg, Niklas,	Kauf	3267	97.88	31977[illegible]9165
Delivery Hero	2021-12-09 00:00:00	Östberg, Niklas,	Kauf	14502	97.85	14190[illegible]1704
Delivery Hero	2021-12-09 00:00:00	Östberg, Niklas,	Kauf	4367	97.89	427485.6273345947

Figure 7 : Dashboard screenshot showing bar-graph for Dec-2021 and transaction details of Delivery Hero