Scientific Research Publishing

Theoretical Economics Letters, 2022, 12, 195-215
https://www.scirp.org/journal/tel
ISSN Online: 2162-2086
ISSN Print: 2162-2078

Business Data Extraction Using a Programming Language

Fabio Bragato Do Carmo, Vinícius Medeiros Magnani, Rafael Confetti Gatsios, Fabiano Guasti Lima

School of Economics, Business Administration and Accounting at Ribeirão Preto, University of São Paulo, São Paulo, Brazil
Email: fabio_bragato@hotmail.com, viniciusmagnani@usp.br, rafaelgatsios@fearp.usp.br, fgl@usp.br

How to cite this paper: Carmo, F. B. D., Magnani, V. M., Gatsios, R. C., & Lima, F. G. (2022). Business Data Extraction Using a Programming Language. *Theoretical Economics Letters, 12,* 195-215.
https://doi.org/10.4236/tel.2022.121011

Received: November 10, 2021
Accepted: February 11, 2022
Published: February 14, 2022

Abstract

In the era of great informational quantity, the presence of technologies that assist in the extraction, transformation, and loading of data has become increasingly necessary. The term Big Data, usually used to describe this volume of information, requires the user to have knowledge of multiple tools such as Excel, VBA, SQL, Tableau, Python, Spark, AWS, and so on. In this context, the present work aims to study data extraction techniques using different methodologies. At the end of the work, a library of functions in the Python language will be made available that will deliver a compilation of stock price information available on the Yahoo Finance website as well as balance sheets from financial institutions released by Bacen. The main resource used will be Web Scraping, which is a method that aims to automate data collection via the web. Once the collection of functions has been structured, it will be made available for public enjoyment through the GitHub platform.

Keywords

Data Extraction, Big Data, Python, Web Scraping

1. Introduction

In recent years, the impact caused by the advancement of technology in the financial market has become an essential phenomenon for us to understand the new behaviors and socioeconomic dynamics that accompany the business world. In this context, companies process a considerable volume of information that requires new data management tools to capture, store, and analyze (Laudon & Laudon, 2014).

The large set of stored data has been entitled "Big Data" by academics and market professionals. According to Chen, Mao, and Liu (2014), Big Data refers to a set of data that cannot be read, managed, and processed by traditional IT tools and software/hardware within a tolerable time.

In consensus in the literature, Big Data is defined by five "Vs": velocity, volume, variety, veracity, and value (Naeem et al., 2022; Ferrara et al., 2014; Laender et al., 2002). Speed because companies need information to flow quickly, as close to real time as possible. Volume because of the large amount of data generated that needs to be stored. Variety due to the gathering of information from different sources. Veracity is related to the quality aspect of the information since it needs to be reliable. And finally, value, since it supports the decision making process of businesses, leading to the creation of new oportunities for companies (Chen, Mao, & Liu, 2014; Laudon & Laudon, 2014).

As a result of this technological expansion and the amount of interpreted information, then, programming languages, database managers, and cloud computing services have emerged as a foundation for the processing of incisive elements for decision making. One of the most important computational languages in the study of data is Python, created in 1991 by mathematician Guido Van Rossum. This language is object-oriented, interpreted, and written with the intention of easy learning and has several convenient modules to increase the productivity of the development of applications to users (Buriol, Marco, & Argenta, 2009).

This programming language includes several high-level structures such as lists, dictionaries, date/time, complexes, and others, in addition to a vast collection of ready-to-use modules and third-party frameworks that can be added. These circumstances provide sufficient resources to include it as one of the main components for analysis and decision making in data generation (Borges, 2014).

The Database Management System (DBMS), in turn, is responsible for the interface of the manipulation and organization of the collected information. Among such systems, a few common ones are MySQL, SQL Server, and Oracle. To use them, it is necessary to have knowledge of Structured Query Language (SQL). According to Miyagusuku (2008), the SQL language involves commands not only of data query but also of manipulation and definition of rules and operators that preserve the integrity and consistency of data in addition to allowing the implementation of procedures (stores procedures), functions (functions) and triggers (triggers), and applications developed in other languages. DBMSs are excellent informational processors and structures, but when it comes to massive amounts of data, activities need to be migrated to cloud computing; AWS (Amazon Web Services) is one example of a company that includes this service.

The term *Big Data*, commonly used to describe a vast mass of processed information, has also supported the emergence of new programming languages. Spark, for example, was built with a focus on the dimensionality of use, allowing the user, in the same application, to integrate different languages such as SQL. Because of these factors, today, there has been a growth in the number of professionals using Python, DBMSs, and cloud computing services to analyze, for example, the earnings of companies listed on the São Paulo Stock Exchange (B3).

Investors use many strategies to study the performance of a company, but it is essential that the first step before starting assessments is to build a database with all the information of the organization. Thus, the main objective of the present work is to structure the use of these new technologies for obtaining bulk data. Three distinct methods will be used to demonstrate the daily routine of a data scientist in the job market, the first being the search for prices of stock market negotiations and data analysis, the second the search for ".zip" format files extracted directly from the HyperText Markup Language (HTML) code of the site server page, and the third, the creation of "Comma-Separated Values" (CSV) format files with the objective of manipulating and organizing the data for further analysis. The codes are applied to information from DBMSs in Brazil; however the method and coding can be adapted to other DBMSs in other countries. Therefore, the novelty of this work is to highlight and disseminate routines for collecting and building a database applied to internet data in html, data in .zip format and in csv format.

2. Research Methodology

The work has a descriptive approach, with the main objective of providing evidence for the methodology used by a data scientist, from the creation of an account in a cloud platform for data storage to the construction and application of the code for data interpretation.

The objects of study are the historical prices of financial institutions traded on B3, as well as their income statements, during the periods of 2017, 2018, and 2019. The price information will be extracted directly from the *Application Programming Interface* (API) provided by the *Yahoo Finance* website, and the income statements will be collected directly from the Brazilian Central Bank website. To structure the information source, Amazon S3 (*Simple Storage Service*) will be used, which will store the files of the data that will be architected in CSV format. The programming language used will be *Python* in the *Jupyter* development environment, since there are function libraries that facilitate the extraction of the information to be collected.

Among the libraries available in the *Jupyter* development environment, the following will be used:

- *Pandas_datareader*: This includes functions that allow the user to extract historical stock price information from *Yahoo Finance*, one of the largest financial news sites in the world. A plausible alternative to obtaining the monetary reports of traded stocks is the *Quandl* library, a leading source of financial, economic, and alternative data sets for investment professionals.
- *NumPy*: This is mainly used to perform calculations on multidimensional arrangements. In addition, it provides a large set of functions and operations that help programmers easily perform numerical calculations.

- *Pandas*: This is the main tool for organizing data. It provides the simplicity of being able to clean, delete, or filter information according to a criterion as well as store tables and graphs in different file formats.
- *Requests*: This allows HTTP (*Hypertext Transfer Protocol*) *requests* to be made to the specified server of the webpage. It is the bridge for performing the extraction of data.
- *Zipfile*: This provides tools to create, read, write, add, and list files in the ".zip" format.
- *Io*: This is commonly used to speed up queries made via the web.

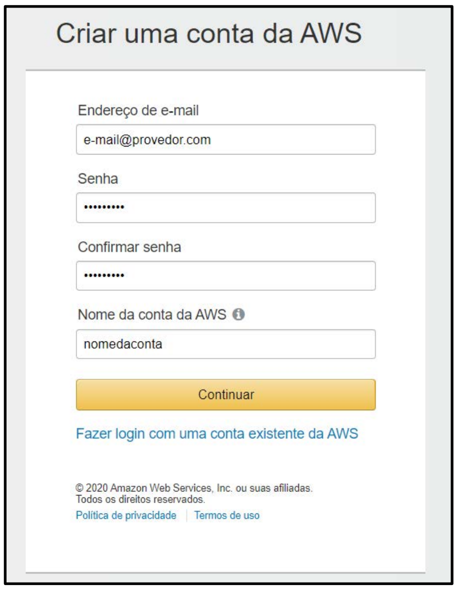
In the development of the methodology, all the cited collections will be used with the intention of contemplating the largest amount of possible resources allowed by Python in addition to having enough subisidies for the reader to learn and apply the obtained knowledge on his/her own having as focus the supply of enough subsidies to the reader to learn and apply by himself the obtained knowledge.

3. Development of Working Model

3.1. Creating an AWS Account

The AWS platform is the world's most widely adopted and comprehensive cloud platform, offering more than 175 comprehensive data center services worldwide. There are different uses for AWS. One of its main services is EMR, which facilitates the execution of Big Data frameworks such as Hadoop and Apache Spark. In the third data extraction method, only the function of S3 for storing CSV files in the cloud will be used. Therefore, to execute the codes, it is necessary to create an account on AWS.

Figure 1 shows the AWS platform account creation table.


Criar uma conta da AWS

Endereço de e-mail
e-mail@provedor.com

Senha
•••••••••

Confirmar senha
•••••••••

Nome da conta da AWS
nomedaconta

Continuar

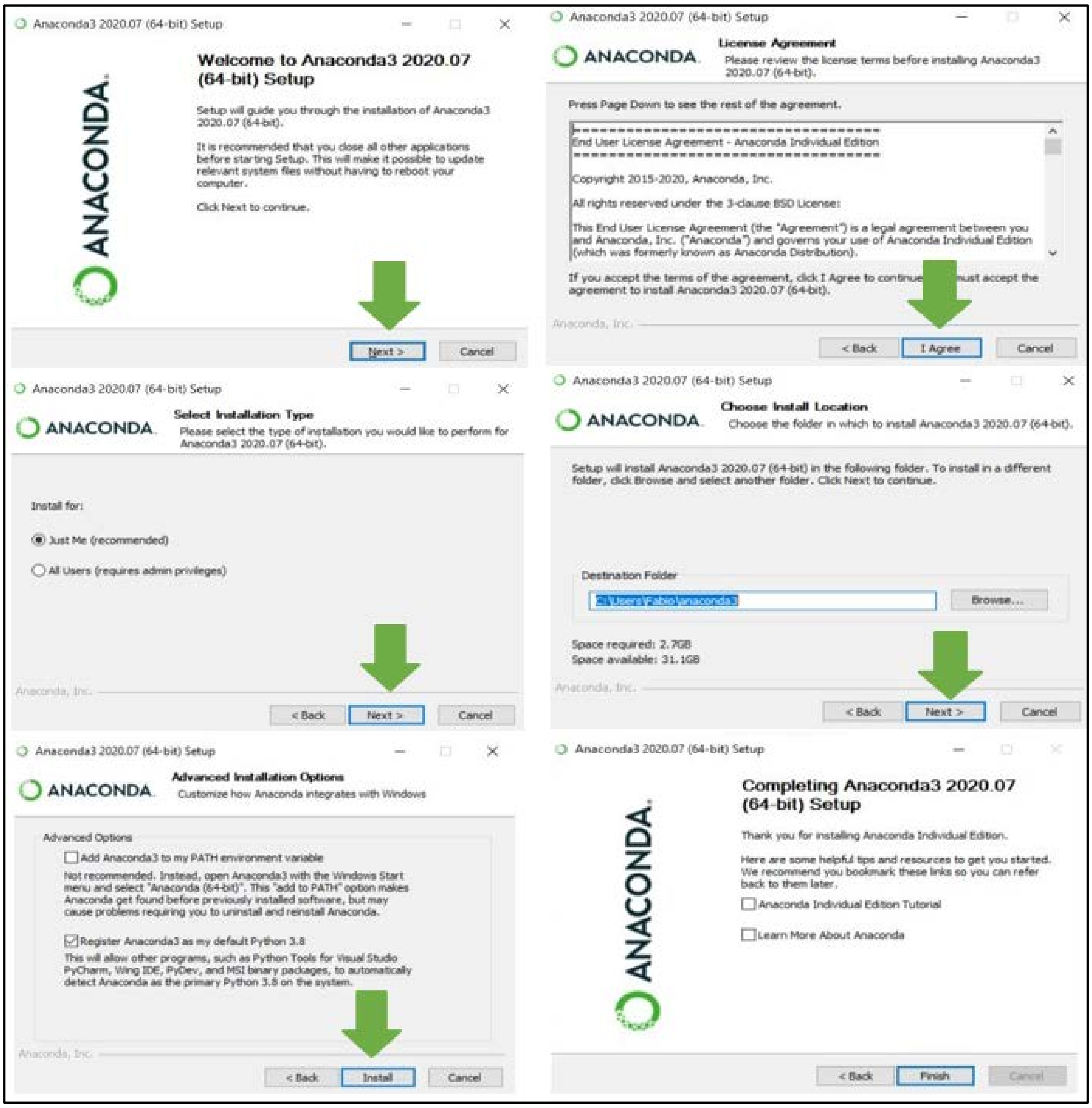
Fazer login com uma conta existente da AWS

© 2020 Amazon Web Services, Inc. ou suas afiliadas.
Todos os direitos reservados.
Política de privacidade | Termos de uso

Figure 1. AWS account creation screen. Source: Own authorship.

3.2. Jupyter and Python Installation

Jupyter Notebook is an Integrated Development Environment (IDE). Briefly, it is the development environment for the applications made by a programmer. It is also one of the most used tools in the elaboration of Python files because it allows users to unite code and text in a single document. The installer provided by the Anaconda project on its official website (Anaconda, 2020) should be used to install the software. Figure 2 shows the images demonstrating the steps to follow after downloading the file:


Anaconda3 2020.07 (64-bit) Setup

Welcome to Anaconda3 2020.07 (64-bit) Setup

Setup will guide you through the installation of Anaconda3 2020.07 (64-bit).

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

Anaconda3 2020.07 (64-bit) Setup

License Agreement

Please review the license terms before installing Anaconda3 2020.07 (64-bit).

Press Page Down to see the rest of the agreement.

```
===================================
End User License Agreement - Anaconda Individual Edition
===================================

Copyright 2015-2020, Anaconda, Inc.

All rights reserved under the 3-clause BSD License:

This End User License Agreement (the "Agreement") is a legal agreement between you
and Anaconda, Inc. ("Anaconda") and governs your use of Anaconda Individual Edition
(which was formerly known as Anaconda Distribution).
```

If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install Anaconda3 2020.07 (64-bit).

Anaconda3 2020.07 (64-bit) Setup

Select Installation Type

Please select the type of installation you would like to perform for Anaconda3 2020.07 (64-bit).

Install for:

● Just Me (recommended)

○ All Users (requires admin privileges)

Anaconda3 2020.07 (64-bit) Setup

Choose Install Location

Choose the folder in which to install Anaconda3 2020.07 (64-bit).

Setup will install Anaconda3 2020.07 (64-bit) in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.

Destination Folder: C:\Users\Fabio\anaconda3

Space required: 2.7GB
Space available: 31.1GB

Anaconda3 2020.07 (64-bit) Setup

Advanced Installation Options

Customize how Anaconda integrates with Windows

Advanced Options

☐ Add Anaconda3 to my PATH environment variable

Not recommended. Instead, open Anaconda3 with the Windows Start menu and select "Anaconda (64-bit)". This "add to PATH" option makes Anaconda get found before previously installed software, but may cause problems requiring you to uninstall and reinstall Anaconda.

☑ Register Anaconda3 as my default Python 3.8

This will allow other programs, such as Python Tools for Visual Studio PyCharm, Wing IDE, PyDev, and MSI binary packages, to automatically detect Anaconda as the primary Python 3.8 on the system.

Anaconda3 2020.07 (64-bit) Setup

Completing Anaconda3 2020.07 (64-bit) Setup

Thank you for installing Anaconda Individual Edition.

Here are some helpful tips and resources to get you started. We recommend you bookmark these links so you can refer back to them later.

☐ Anaconda Individual Edition Tutorial

☐ Learn More About Anaconda

Figure 2. Steps to download file. Source: Own authorship.

3.3. Creating the First Python File

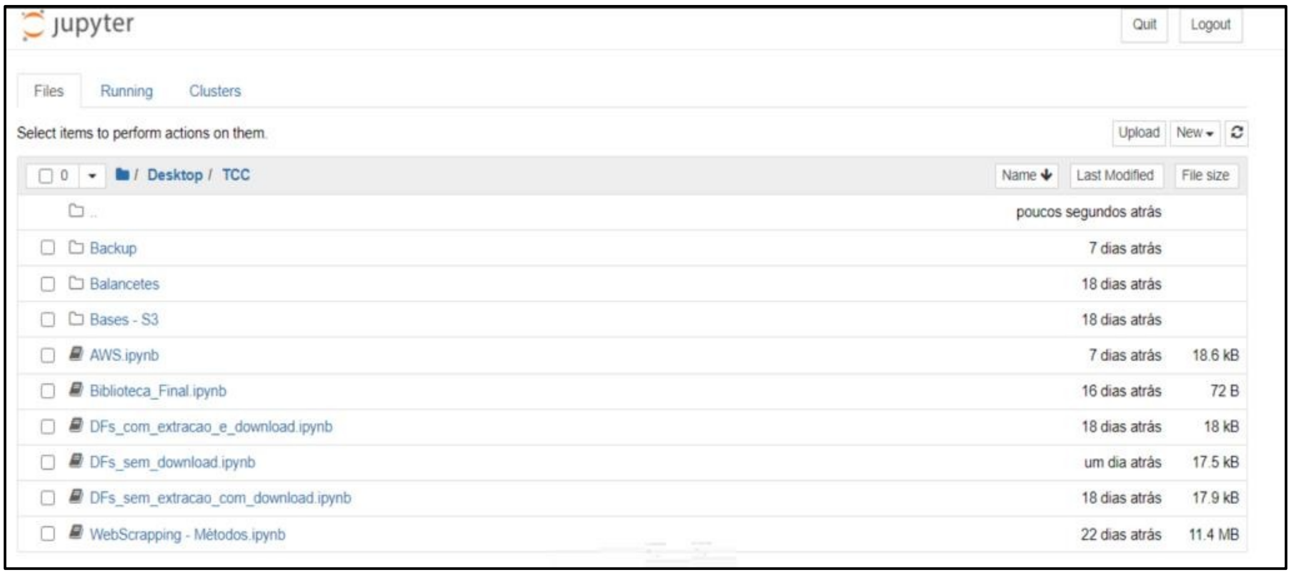
After installation, Jupyter Notebook should be started. A local server will run (by default, local output 8888), and, once loaded, a new tab in the browser will open, making it possible to see the computer Dashboard, Figure 3.

A list of files and subdirectories present on the computer is displayed. The user should create or select a folder to receive the new files. Afterward, the user should select the "New" option and choose "Python 3, Figure 4".

A new screen will open. This is where all the applications will be developed, Figure 5.

Figure 5 shows that the Jupyter Notebook is composed of the following elements:

1) File name: name of the file, which will also be displayed at the top of the screen.

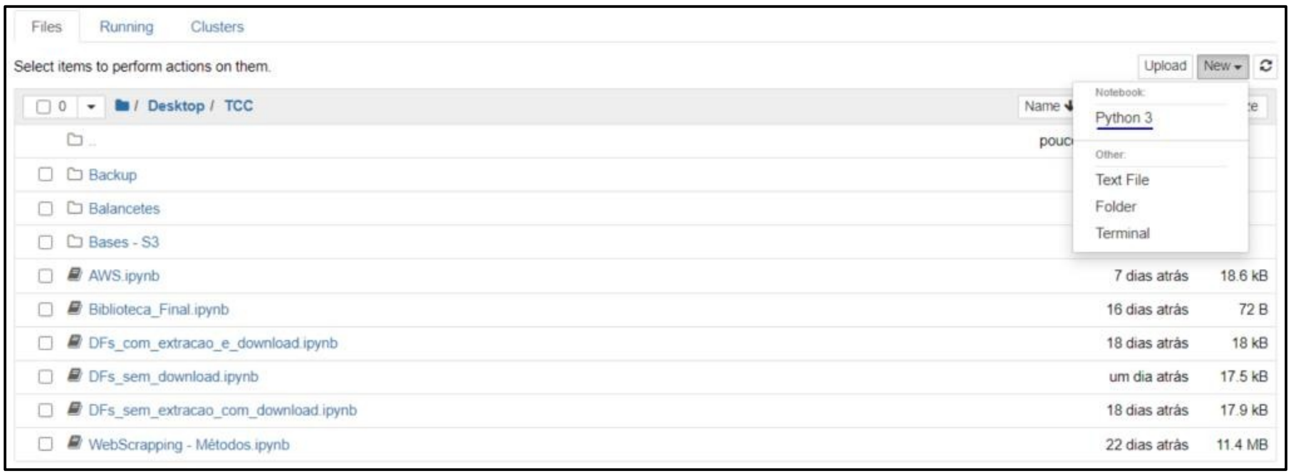


Select items to perform actions on them.

0 / Desktop / TCC

..	poucos segundos atrás	
Backup	7 dias atrás	
Balancetes	18 dias atrás	
Bases - S3	18 dias atrás	
AWS.ipynb	7 dias atrás	18.6 kB
Biblioteca_Final.ipynb	16 dias atrás	72 B
DFs_com_extracao_e_download.ipynb	18 dias atrás	18 kB
DFs_sem_download.ipynb	um dia atrás	17.5 kB
DFs_sem_extracao_com_download.ipynb	18 dias atrás	17.9 kB
WebScrapping - Métodos.ipynb	22 dias atrás	11.4 MB

Figure 3. Jupyter dashboard. Source: Own authorship.



Select items to perform actions on them.

0 / Desktop / TCC

..	pouc	
Backup		
Balancetes		
Bases - S3		
AWS.ipynb	7 dias atrás	18.6 kB
Biblioteca_Final.ipynb	16 dias atrás	72 B
DFs_com_extracao_e_download.ipynb	18 dias atrás	18 kB
DFs_sem_download.ipynb	um dia atrás	17.5 kB
DFs_sem_extracao_com_download.ipynb	18 dias atrás	17.9 kB
WebScrapping - Métodos.ipynb	22 dias atrás	11.4 MB

Figure 4. Jupyter directories. Source: Own authorship.

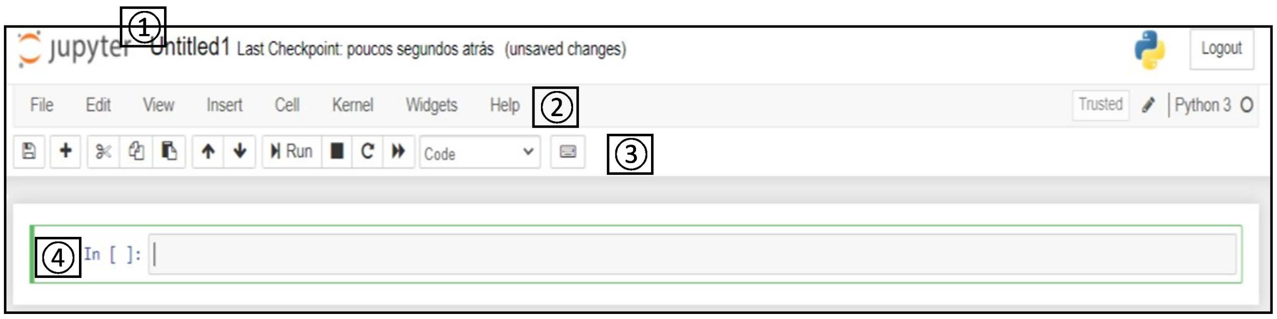


Figure 5. Application development screen. Source: Own authorship.

2) Menu bar: different options that can be used to manipulate the operation of the notebook.

3) Toolbar: quick way to perform the most frequently used operations.

4) Code cell: allows the user you to edit or write new code.

3.4. Installing the Libraries

A library is a collection of modules accessible in a Python program that allows the user to simplify the programming process and remove the need to rewrite an already used function such as a calculation that extracts the square root of a number. We can call that function continuously just by modifying (or not modifying) the pre-existing parameters. Because of this, it is necessary to install open source libraries (that anyone has the right to study, modify, or distribute) that will help to create the data search methods. The execution is as follows, Figure 6.

3.5. Methods

3.5.1. Yahoo Finance

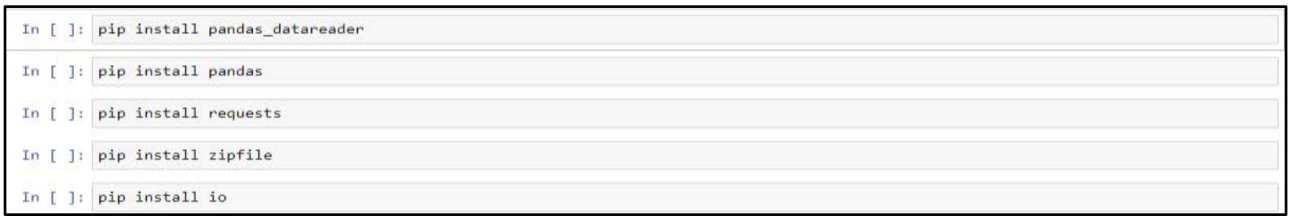
The first method used will be with the support of the library "pandas_datareader" to search the trading prices of shares on stock exchanges. The consulted data are extracted from the Yahoo Finance website and has a fifteen-minute lag time. The function created for the library called "data_actions_yahoo" was defined based on four parameters:

1st: Tickers, in which it is necessary to define the stock codes to be searched. By convention, all assets traded on B3 have the ending ".SA" (Joint Stock Company), as in "MGLU3.SA", "AZUL4.SA", "HBOR3.SA", and so on.

2nd: Start date, which will serve to reference the beginning of the consultation period. Written in a "year-month-day" pattern ("2020-01-01").

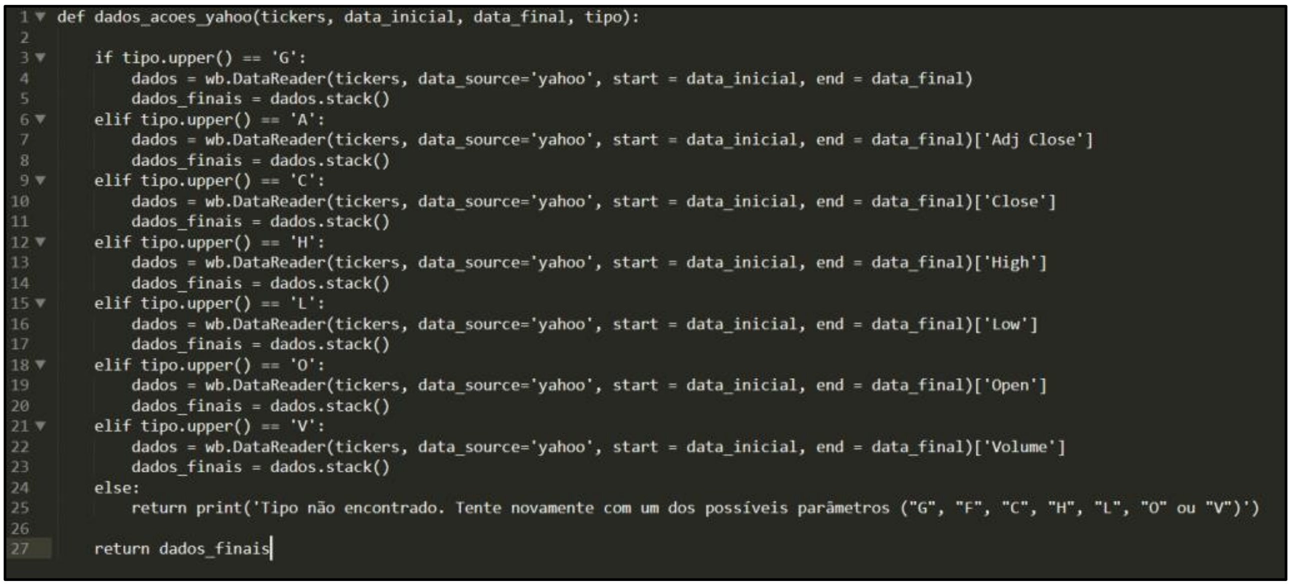
3rd: End date, which will serve to reference the end of the consultation period. Written in a "year-month-day" pattern ("2020-06-01").

4th: Type, which will reference the desired information among the possible types. "G"—All available information; "A"—Adjusted closing prices; "C"—Closing prices; "H"—Maximum prices; "L"—Minimum prices; "O"—Opening prices; "V"—Trading volume. Figure 7 shows the treatment of the function "dados_acoes_yahoo, Figure 7".



```
In [ ]: pip install pandas_datareader

In [ ]: pip install pandas

In [ ]: pip install requests

In [ ]: pip install zipfile

In [ ]: pip install io
```

Figure 6. Installing the libraries. Source: Own authorship.



```
def dados_acoes_yahoo(tickers, data_inicial, data_final, tipo):

    if tipo.upper() == 'G':
        dados = wb.DataReader(tickers, data_source='yahoo', start = data_inicial, end = data_final)
        dados_finais = dados.stack()
    elif tipo.upper() == 'A':
        dados = wb.DataReader(tickers, data_source='yahoo', start = data_inicial, end = data_final)['Adj Close']
        dados_finais = dados.stack()
    elif tipo.upper() == 'C':
        dados = wb.DataReader(tickers, data_source='yahoo', start = data_inicial, end = data_final)['Close']
        dados_finais = dados.stack()
    elif tipo.upper() == 'H':
        dados = wb.DataReader(tickers, data_source='yahoo', start = data_inicial, end = data_final)['High']
        dados_finais = dados.stack()
    elif tipo.upper() == 'L':
        dados = wb.DataReader(tickers, data_source='yahoo', start = data_inicial, end = data_final)['Low']
        dados_finais = dados.stack()
    elif tipo.upper() == 'O':
        dados = wb.DataReader(tickers, data_source='yahoo', start = data_inicial, end = data_final)['Open']
        dados_finais = dados.stack()
    elif tipo.upper() == 'V':
        dados = wb.DataReader(tickers, data_source='yahoo', start = data_inicial, end = data_final)['Volume']
        dados_finais = dados.stack()
    else:
        return print('Tipo não encontrado. Tente novamente com um dos possíveis parâmetros ("G", "F", "C", "H", "L", "O" ou "V")')

    return dados_finais
```

Figure 7. Handling of the "data_actions_yahoo" function. Source: Own authorship.

The second function, "plot_graphs_closing," aims to create a comparative chart of the stock prices evolution based on the parameters "tickers", "start_date", and "end_date". It is shown below Figure 8, demonstrating the function of graph plotting.

The third function, "simple_returns" serves to create a table with the simple stock returns in the specified period from the calculation and is evidenced in Figure 9. In Equation (1), the stock return calculation can be observed.

$$\left(\frac{\text{Share price in Tn}}{\text{Share Price in Tn}-1}\right)-1 \tag{1}$$

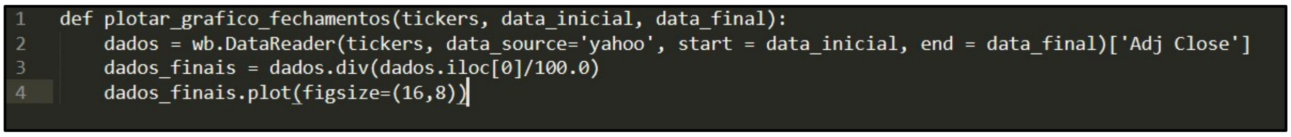
The parameters are "tickers", "start_date", and "end_date", Figure 9.

The last function, "retornos_log", has as objective of calculating the logarithmic returns of stocks in the specified period from the calculation:

$$\frac{\ln\left(\text{Share Price}_t\right)}{\ln\left(\text{Share Price}_{t-1}\right)}-1 \tag{2}$$

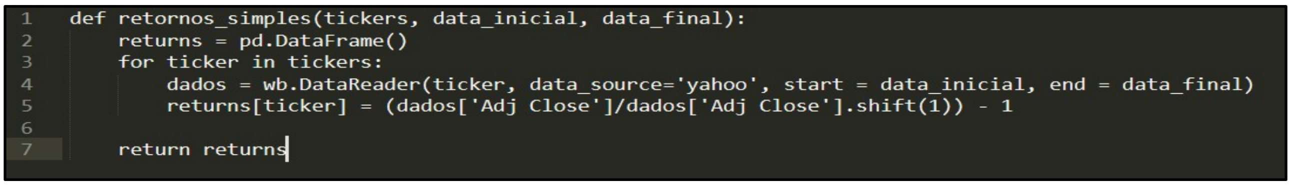
The parameters are "tickers", "start_date", and "end_date", Figure 10.

Finally, many other functions can be created based on each user's need. However, the main goal of programming is to avoid repeating work and make it easier to write code.



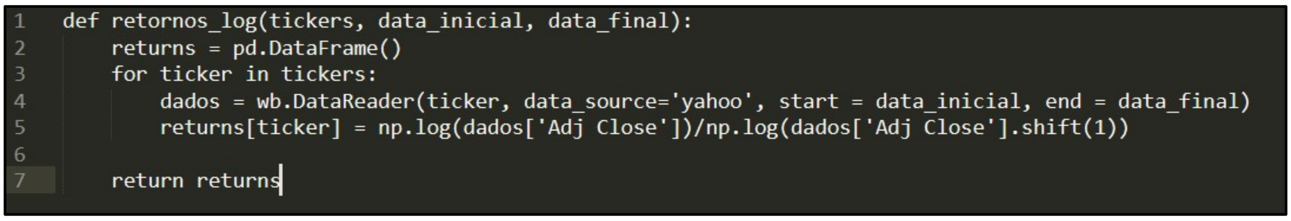
```python
def plotar_grafico_fechamentos(tickers, data_inicial, data_final):
    dados = wb.DataReader(tickers, data_source='yahoo', start = data_inicial, end = data_final)['Adj Close']
    dados_finais = dados.div(dados.iloc[0]/100.0)
    dados_finais.plot(figsize=(16,8))
```

Figure 8. Handling of the "plot_graphs_closure" function. Source: Own authorship.



```python
def retornos_simples(tickers, data_inicial, data_final):
    returns = pd.DataFrame()
    for ticker in tickers:
        dados = wb.DataReader(ticker, data_source='yahoo', start = data_inicial, end = data_final)
        returns[ticker] = (dados['Adj Close']/dados['Adj Close'].shift(1)) - 1

    return returns
```

Figure 9. Treatment of the "simple_returns" function. Source: Own authorship.



```python
def retornos_log(tickers, data_inicial, data_final):
    returns = pd.DataFrame()
    for ticker in tickers:
        dados = wb.DataReader(ticker, data_source='yahoo', start = data_inicial, end = data_final)
        returns[ticker] = np.log(dados['Adj Close'])/np.log(dados['Adj Close'].shift(1))

    return returns
```

Figure 10. Treatment of the function "retornos_log." Source: Own authorship.

3.5.2. ZIP Files

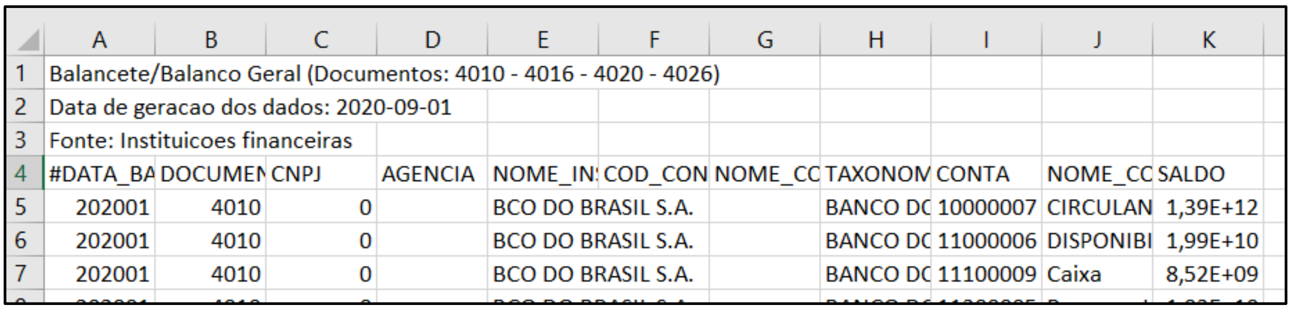
The method of using ".zip" files is one of the most frequently utilized in data search. The purpose is to search the information of a site without installing any files. The information is extracted from the HTML code of the page itself or from the HTTP connection with the site server. The following example consolidates information from the balance sheets of financial institutions published by Bacen.

First of all, it is necessary to check the repository format, which is the ".zip" extension with CSV files. In addition, when you open the file in Excel, you can check the following structure, Figure 11.

As shown above, the data starts on the fourth line. This information will be important in view of the need to structure the data when writing the function. The code is as follows, Figure 12.

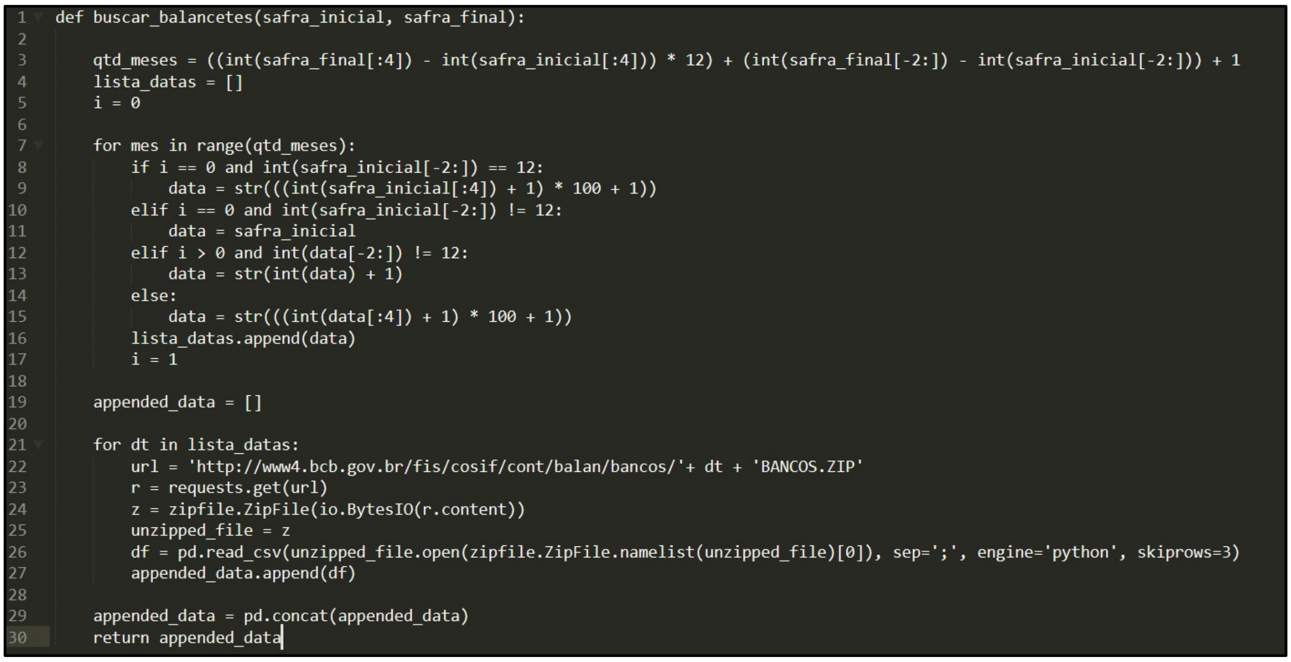
Analyzing the code line by line, we have the following steps:

- Line 1: Definition of the function that accepts two parameters, "start_harvest" and "end_harvest", both in the "year-month" format, for example, "202001".
- Line 3: Definition of the function to find the number of months between the initial and final harvests.
- Line 4: Creation of a list containing the crops to be searched.
- Line 5: Definition of a variable "i" with a value of zero that will serve as an auxiliary.
- Lines 7 to 17: A loop to add the crops chosen by the user.
- Line 19: Creation of a list that will contain the data from the CSV files to be extracted.



	A	B	C	D	E	F	G	H	I	J	K
1	Balancete/Balanco Geral (Documentos: 4010 - 4016 - 4020 - 4026)										
2	Data de geracao dos dados: 2020-09-01										
3	Fonte: Instituicoes financeiras										
4	#DATA_BA	DOCUMEN	CNPJ	AGENCIA	NOME_IN:	COD_CON	NOME_CC	TAXONOM	CONTA	NOME_CC	SALDO
5	202001	4010	0		BCO DO BRASIL S.A.			BANCO DC	10000007	CIRCULAN	1,39E+12
6	202001	4010	0		BCO DO BRASIL S.A.			BANCO DC	11000006	DISPONIBI	1,99E+10
7	202001	4010	0		BCO DO BRASIL S.A.			BANCO DC	11100009	Caixa	8,52E+09

Figure 11. Excel structure. Source: Own authorship.



```
1  def buscar_balancetes(safra_inicial, safra_final):
2  
3      qtd_meses = ((int(safra_final[:4]) - int(safra_inicial[:4])) * 12) + (int(safra_final[-2:]) - int(safra_inicial[-2:])) + 1
4      lista_datas = []
5      i = 0
6  
7      for mes in range(qtd_meses):
8          if i == 0 and int(safra_inicial[-2:]) == 12:
9              data = str(((int(safra_inicial[:4]) + 1) * 100 + 1))
10         elif i == 0 and int(safra_inicial[-2:]) != 12:
11             data = safra_inicial
12         elif i > 0 and int(data[-2:]) != 12:
13             data = str(int(data) + 1)
14         else:
15             data = str(((int(data[:4]) + 1) * 100 + 1))
16         lista_datas.append(data)
17         i = 1
18 
19     appended_data = []
20 
21     for dt in lista_datas:
22         url = 'http://www4.bcb.gov.br/fis/cosif/cont/balan/bancos/'+ dt + 'BANCOS.ZIP'
23         r = requests.get(url)
24         z = zipfile.ZipFile(io.BytesIO(r.content))
25         unzipped_file = z
26         df = pd.read_csv(unzipped_file.open(zipfile.ZipFile.namelist(unzipped_file)[0]), sep=';', engine='python', skiprows=3)
27         appended_data.append(df)
28 
29     appended_data = pd.concat(appended_data)
30     return appended_data
```

Figure 12. Handling the "fetch_balancets" function. Source: Own authorship.

- Lines 21 to 27: The loop to capture the file information is built.
- Line 22: Definition of the URL of the site where we will collect the information from the balance sheets.
- Line 23: Use of the method "requests.get" to make the connection with the site.
- Line 24: Use of a compilation of methods. Analyzing from the inner layer outward, we have the use of "r.content", which has the function of collecting all the information contained in the connection made in line 23. It also uses the "io.BytesIo" to increase the search speed and, lastly, the "zipfile.Zipfile" to handle the ".zip" file.
- Line 25: A variable is set to capture the unzipped ".zip" file.
- Line 26: A "DataFrame" is created and the method "pd.read_csv" is used to read the CSV file contained in the decompressed archive. It is important to observe the parameters used in the function.
- "sep = ';'": CSV file with separator ";".

- "skiprows = 3": Skips three rows as indicated in the Excel figure above to start reading the information from the row that has the data pertinent to the trial balances.
- "engine = 'python'": An optional parameter that defines the parsing engine. The C engine is faster, while the Python engine is currently more resourceful.
- Line 27: Addition of the extracted data to the list of data created.
- Line 29: Concatenation of all extracted data into a single table.
- Line 30: Return of the extracted data.

3.5.3. CSV in AWS S3

The CSV file method in AWS S3 is an alternative way to manipulate and organize data. As an example, we will use the financial statement data provided by the Central Bank on the IFDATA website (https://www3.bcb.gov.br/ifdata/). First, the user must download the desired files. In the institution type, the user must select the option "Financial Conglomerates and Independent Institutions" in addition to choosing one of the three possible reports (assets, liabilities, or income statement), Figure 13.

Once the files have been obtained, the user logs in to the AWS website and selects the "AWS Management Console" option to log into the account, Figure 14.

Upon logging in, the user must click on services in the top menu and select "S3" in the storage group, Figure 15.

This is the screen where the user can create several directories ("Buckets") that must have unique names. To do so, press "Create Bucket" and configure it the way you prefer, Figure 16.

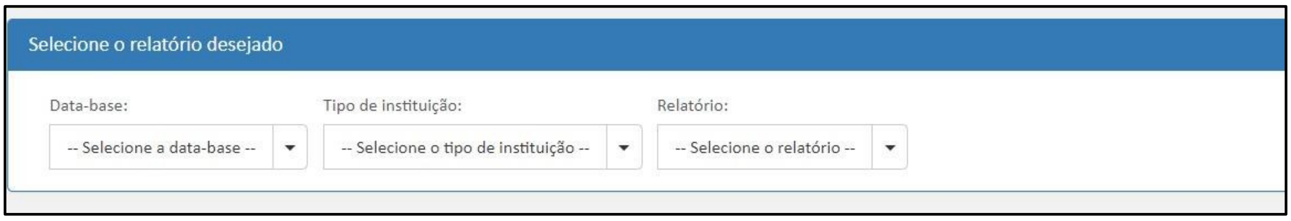

Selecione o relatório desejado
Data-base:
-- Selecione a data-base --
Tipo de instituição:
-- Selecione o tipo de instituição --
Relatório:
-- Selecione o relatório --

Figure 13. IFDATA data selection screen. Source: Own authorship.

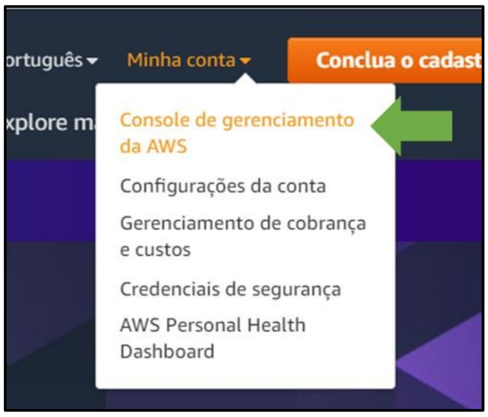

ortuguês
Minha conta
Conclua o cadast
xplore m
Console de gerenciamento da AWS
Configurações da conta
Gerenciamento de cobrança e custos
Credenciais de segurança
AWS Personal Health Dashboard

Figure 14. AWS login screen. Source: Own authorship.

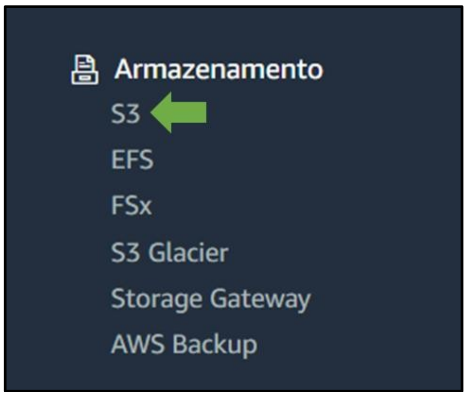


Figure 15. AWS selection menu. Source: Own authorship.

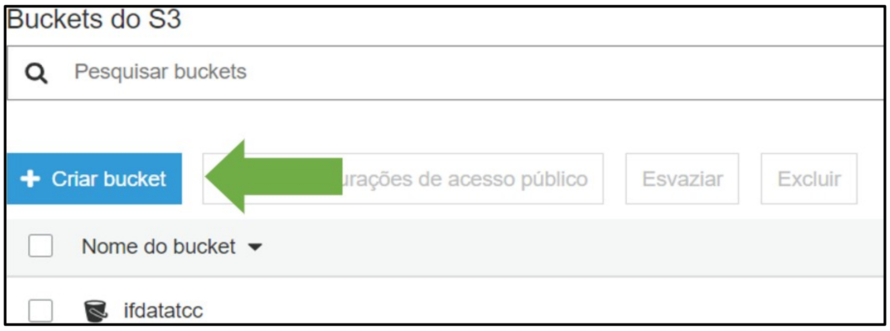


Figure 16. Creating the bucket. Source: Own authorship.

After creating the Bucket, creating a folder for each report in order to maintain the organization of the files is suggested, Figure 17.

The user must upload the files that he/she downloaded in the created folder, and it is essential that, for the functioning of the library, the file is named with the standard "year-quarter", for example, "202001" referring to the first quarter of 2020, Figure 18.

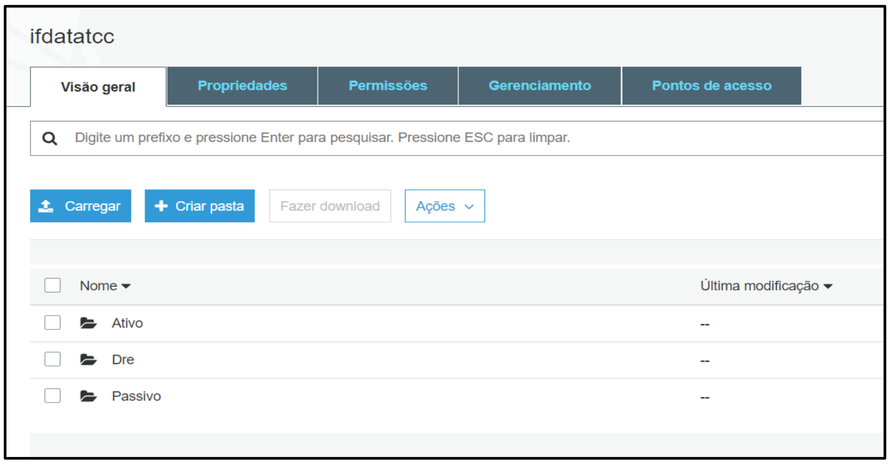
Finally, the user must select all the files that were uploaded to the folder and in the "Actions" option, select "Make Public", Figure 19.

After finishing the S3 step, the library is ready for operation. The image below shows its construction part by part and contains more than one function for use. In order to facilitate understanding, the code will be separated into four parts:

Function "buscar_relatorio": This function's aim is to structure a report of a specific period in a grid, Figure 20.

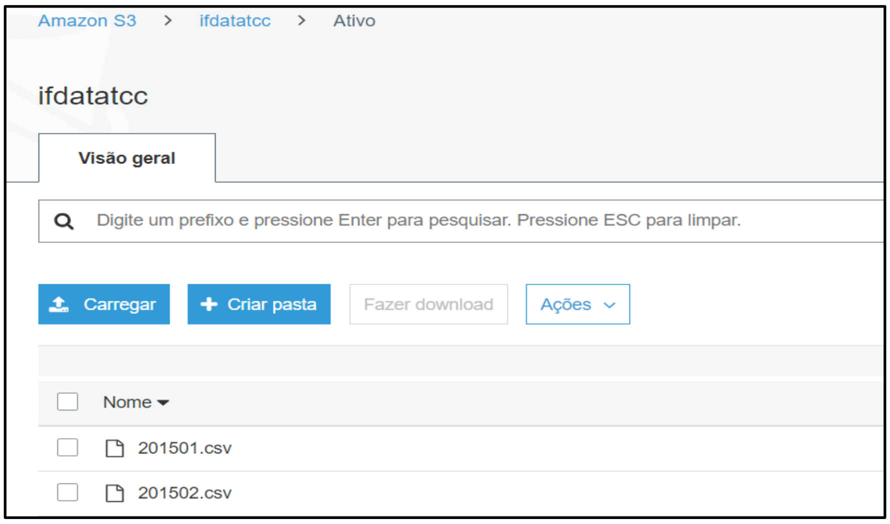
Part 1: construction of the asset report.

- Line 1: Define the function "buscar_relatorio", which has the parameters name_bucket (ex: "ifdatatcc"), account (ex: "assets", "liabilities", or "DRE"), year (ex: 2020) and quarter (ex: 1, 2, 3, or 4).
- Line 2: The "try" method is used for error handling, as will be shown at the end of the code.
- Line 4: A conditional is created to check if one of the reports was selected.
- Line 7: This shows a link where the CSV file is stored in S3.



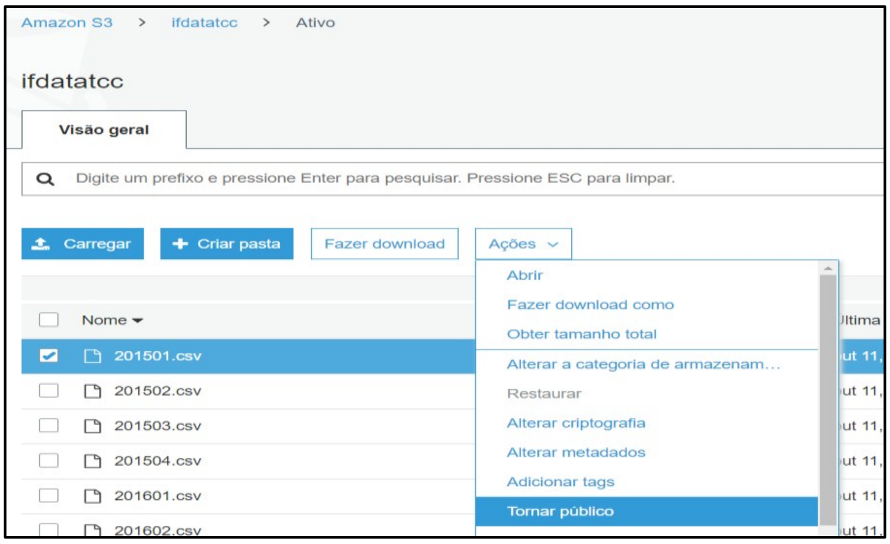
Nome	Última modificação
Ativo	--
Dre	--
Passivo	--

Figure 17. Bucket structure. Source: Own authorship.


Amazon S3 > ifdatatcc > Ativo
ifdatatcc
Visão geral
Digite um prefixo e pressione Enter para pesquisar. Pressione ESC para limpar.
Carregar
Criar pasta
Fazer download
Ações

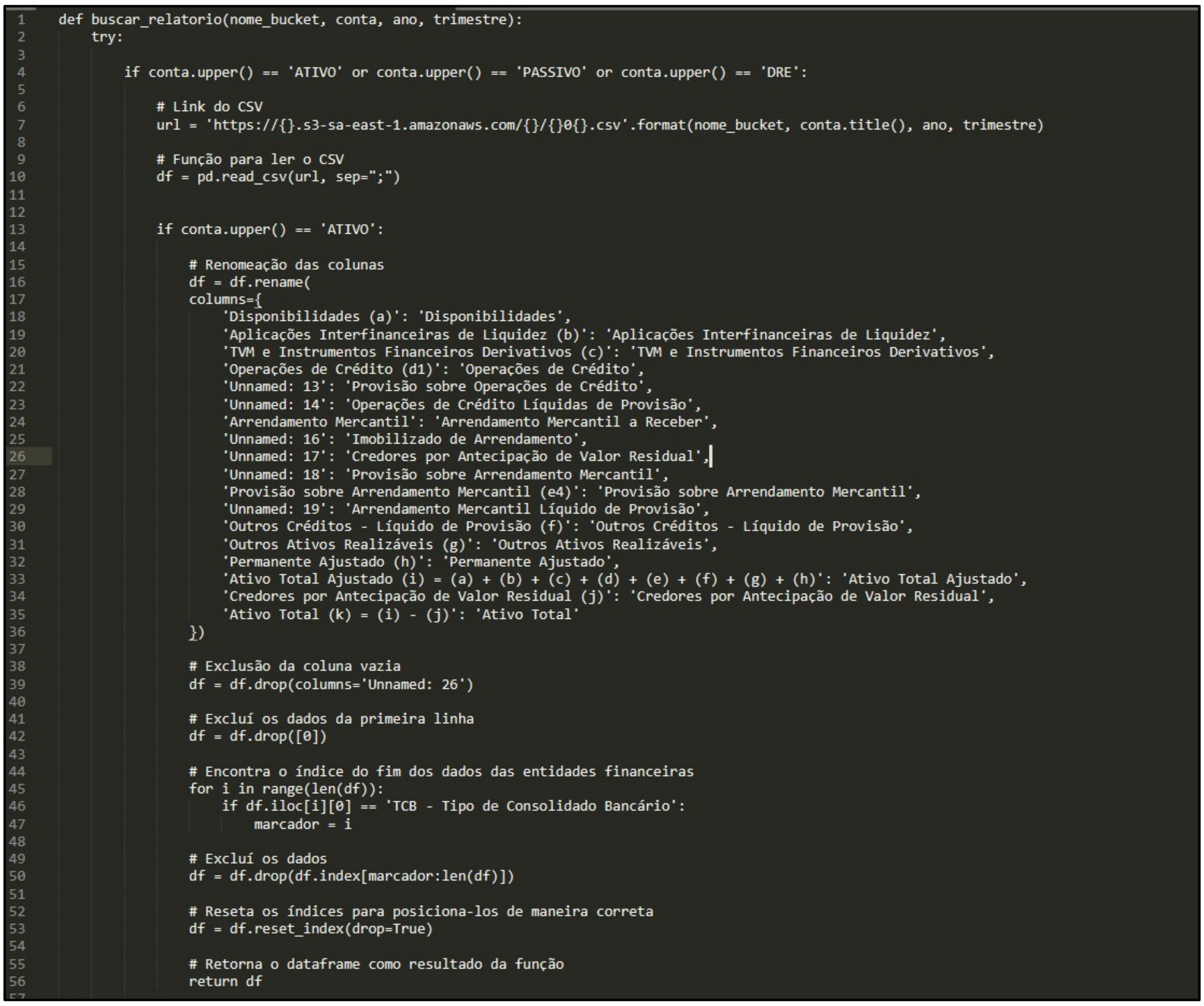
Nome
201501.csv
201502.csv

Figure 18. Structured files in AWS. Source: Own authorship.



Nome	Última
✓ 201501.csv	ut 11,
201502.csv	ut 11,
201503.csv	ut 11,
201504.csv	ut 11,
201601.csv	ut 11,
201602.csv	ut 11,

Figure 19. Making the archives public. Source: Own authorship.



```
1   def buscar_relatorio(nome_bucket, conta, ano, trimestre):
2       try:
3   
4           if conta.upper() == 'ATIVO' or conta.upper() == 'PASSIVO' or conta.upper() == 'DRE':
5   
6               # Link do CSV
7               url = 'https://{}.s3-sa-east-1.amazonaws.com/{}/{}0{}.csv'.format(nome_bucket, conta.title(), ano, trimestre)
8   
9               # Função para ler o CSV
10              df = pd.read_csv(url, sep=";")
11  
12  
13              if conta.upper() == 'ATIVO':
14  
15                  # Renomeação das colunas
16                  df = df.rename(
17                  columns={
18                      'Disponibilidades (a)': 'Disponibilidades',
19                      'Aplicações Interfinanceiras de Liquidez (b)': 'Aplicações Interfinanceiras de Liquidez',
20                      'TVM e Instrumentos Financeiros Derivativos (c)': 'TVM e Instrumentos Financeiros Derivativos',
21                      'Operações de Crédito (d1)': 'Operações de Crédito',
22                      'Unnamed: 13': 'Provisão sobre Operações de Crédito',
23                      'Unnamed: 14': 'Operações de Crédito Líquidas de Provisão',
24                      'Arrendamento Mercantil': 'Arrendamento Mercantil a Receber',
25                      'Unnamed: 16': 'Imobilizado de Arrendamento',
26                      'Unnamed: 17': 'Credores por Antecipação de Valor Residual',
27                      'Unnamed: 18': 'Provisão sobre Arrendamento Mercantil',
28                      'Provisão sobre Arrendamento Mercantil (e4)': 'Provisão sobre Arrendamento Mercantil',
29                      'Unnamed: 19': 'Arrendamento Mercantil Líquido de Provisão',
30                      'Outros Créditos - Líquido de Provisão (f)': 'Outros Créditos - Líquido de Provisão',
31                      'Outros Ativos Realizáveis (g)': 'Outros Ativos Realizáveis',
32                      'Permanente Ajustado (h)': 'Permanente Ajustado',
33                      'Ativo Total Ajustado (i) = (a) + (b) + (c) + (d) + (e) + (f) + (g) + (h)': 'Ativo Total Ajustado',
34                      'Credores por Antecipação de Valor Residual (j)': 'Credores por Antecipação de Valor Residual',
35                      'Ativo Total (k) = (i) - (j)': 'Ativo Total'
36                  })
37  
38                  # Exclusão da coluna vazia
39                  df = df.drop(columns='Unnamed: 26')
40  
41                  # Excluí os dados da primeira linha
42                  df = df.drop([0])
43  
44                  # Encontra o índice do fim dos dados das entidades financeiras
45                  for i in range(len(df)):
46                      if df.iloc[i][0] == 'TCB - Tipo de Consolidado Bancário':
47                          marcador = i
48  
49                  # Excluí os dados
50                  df = df.drop(df.index[marcador:len(df)])
51  
52                  # Reseta os índices para posiciona-los de maneira correta
53                  df = df.reset_index(drop=True)
54  
55                  # Retorna o dataframe como resultado da função
56                  return df
```

Figure 20. Handling the assets of the "fetch_report" function. Source: Own authorship.

- Line 13: This creates a conditional for when the "Asset" report is selected.
- Lines 16 to 53: These lines contain data processing, since it is necessary to process the information in order to use it.
- Line 56: This line returns the processed data to the function user.

Part 2: construction of the liability report, Figure 21.

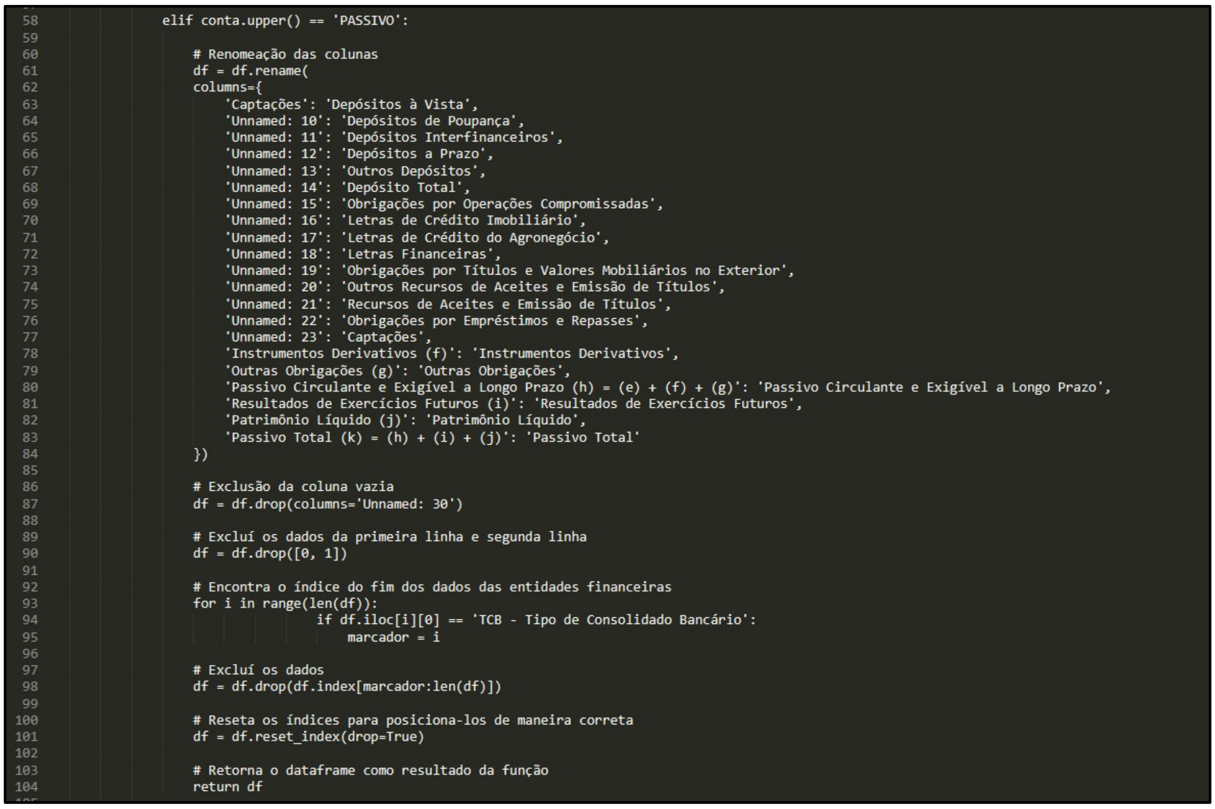
The same steps are performed as in the previous method, but the data are treated according to their peculiarities. The final return is given in line 104.

Part 3: construction of the DRE (Statement of Income for the Year) report, Figure 22.

This has the same logic, but is also treated according to its differences. The final return is on line 161.

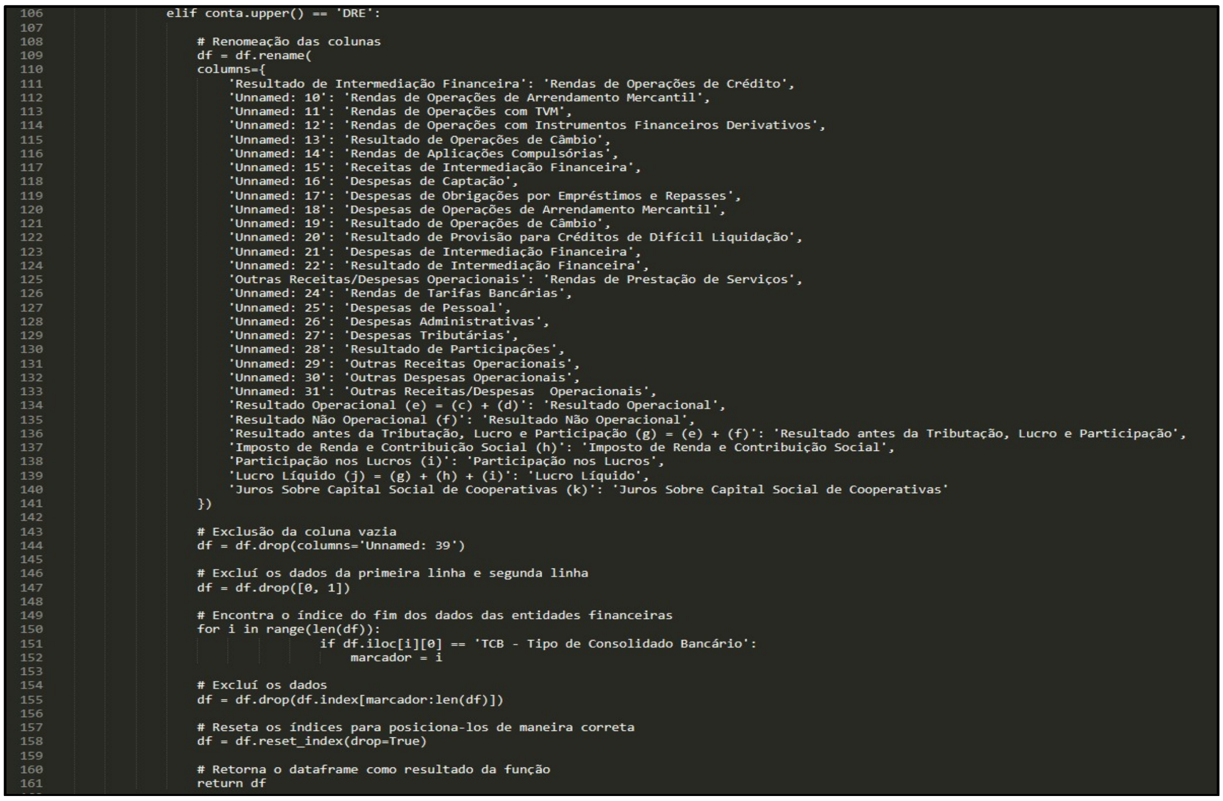
Part 4: final procedure of the code for handling exceptions, Figure 23.

- Line 163: In case the name of the selected report is not found, a phrase will be returned to the user in order to correct the value passed as parameter.



```python
        elif conta.upper() == 'PASSIVO':

            # Renomeação das colunas
            df = df.rename(
            columns={
                'Captações': 'Depósitos à Vista',
                'Unnamed: 10': 'Depósitos de Poupança',
                'Unnamed: 11': 'Depósitos Interfinanceiros',
                'Unnamed: 12': 'Depósitos a Prazo',
                'Unnamed: 13': 'Outros Depósitos',
                'Unnamed: 14': 'Depósito Total',
                'Unnamed: 15': 'Obrigações por Operações Compromissadas',
                'Unnamed: 16': 'Letras de Crédito Imobiliário',
                'Unnamed: 17': 'Letras de Crédito do Agronegócio',
                'Unnamed: 18': 'Letras Financeiras',
                'Unnamed: 19': 'Obrigações por Títulos e Valores Mobiliários no Exterior',
                'Unnamed: 20': 'Outros Recursos de Aceites e Emissão de Títulos',
                'Unnamed: 21': 'Recursos de Aceites e Emissão de Títulos',
                'Unnamed: 22': 'Obrigações por Empréstimos e Repasses',
                'Unnamed: 23': 'Captações',
                'Instrumentos Derivativos (f)': 'Instrumentos Derivativos',
                'Outras Obrigações (g)': 'Outras Obrigações',
                'Passivo Circulante e Exigível a Longo Prazo (h) = (e) + (f) + (g)': 'Passivo Circulante e Exigível a Longo Prazo',
                'Resultados de Exercícios Futuros (i)': 'Resultados de Exercícios Futuros',
                'Patrimônio Líquido (j)': 'Patrimônio Líquido',
                'Passivo Total (k) = (h) + (i) + (j)': 'Passivo Total'
            })

            # Exclusão da coluna vazia
            df = df.drop(columns='Unnamed: 30')

            # Excluí os dados da primeira linha e segunda linha
            df = df.drop([0, 1])

            # Encontra o índice do fim dos dados das entidades financeiras
            for i in range(len(df)):
                            if df.iloc[i][0] == 'TCB - Tipo de Consolidado Bancário':
                                marcador = i

            # Excluí os dados
            df = df.drop(df.index[marcador:len(df)])

            # Reseta os índices para posiciona-los de maneira correta
            df = df.reset_index(drop=True)

            # Retorna o dataframe como resultado da função
            return df
```

Figure 21. Handling the liabilities of the "fetch_report" function. Source: Own authorship.



```python
        elif conta.upper() == 'DRE':

            # Renomeação das colunas
            df = df.rename(
            columns={
                'Resultado de Intermediação Financeira': 'Rendas de Operações de Crédito',
                'Unnamed: 10': 'Rendas de Operações de Arrendamento Mercantil',
                'Unnamed: 11': 'Rendas de Operações com TVM',
                'Unnamed: 12': 'Rendas de Operações com Instrumentos Financeiros Derivativos',
                'Unnamed: 13': 'Resultado de Operações de Câmbio',
                'Unnamed: 14': 'Rendas de Aplicações Compulsórias',
                'Unnamed: 15': 'Receitas de Intermediação Financeira',
                'Unnamed: 16': 'Despesas de Captação',
                'Unnamed: 17': 'Despesas de Obrigações por Empréstimos e Repasses',
                'Unnamed: 18': 'Despesas de Operações de Arrendamento Mercantil',
                'Unnamed: 19': 'Resultado de Operações de Câmbio',
                'Unnamed: 20': 'Resultado de Provisão para Créditos de Difícil Liquidação',
                'Unnamed: 21': 'Despesas de Intermediação Financeira',
                'Unnamed: 22': 'Resultado de Intermediação Financeira',
                'Outras Receitas/Despesas Operacionais': 'Rendas de Prestação de Serviços',
                'Unnamed: 24': 'Rendas de Tarifas Bancárias',
                'Unnamed: 25': 'Despesas de Pessoal',
                'Unnamed: 26': 'Despesas Administrativas',
                'Unnamed: 27': 'Despesas Tributárias',
                'Unnamed: 28': 'Resultado de Participações',
                'Unnamed: 29': 'Outras Receitas Operacionais',
                'Unnamed: 30': 'Outras Despesas Operacionais',
                'Unnamed: 31': 'Outras Receitas/Despesas  Operacionais',
                'Resultado Operacional (e) = (c) + (d)': 'Resultado Operacional',
                'Resultado Não Operacional (f)': 'Resultado Não Operacional',
                'Resultado antes da Tributação, Lucro e Participação (g) = (e) + (f)': 'Resultado antes da Tributação, Lucro e Participação',
                'Imposto de Renda e Contribuição Social (h)': 'Imposto de Renda e Contribuição Social',
                'Participação nos Lucros (i)': 'Participação nos Lucros',
                'Lucro Líquido (j) = (g) + (h) + (i)': 'Lucro Líquido',
                'Juros Sobre Capital Social de Cooperativas (k)': 'Juros Sobre Capital Social de Cooperativas'
            })

            # Exclusão da coluna vazia
            df = df.drop(columns='Unnamed: 39')

            # Excluí os dados da primeira linha e segunda linha
            df = df.drop([0, 1])

            # Encontra o índice do fim dos dados das entidades financeiras
            for i in range(len(df)):
                            if df.iloc[i][0] == 'TCB - Tipo de Consolidado Bancário':
                                marcador = i

            # Excluí os dados
            df = df.drop(df.index[marcador:len(df)])

            # Reseta os índices para posiciona-los de maneira correta
            df = df.reset_index(drop=True)

            # Retorna o dataframe como resultado da função
            return df
```

Figure 22. Handling the DRE of the "fetch_report" function. Source: Own authorship.

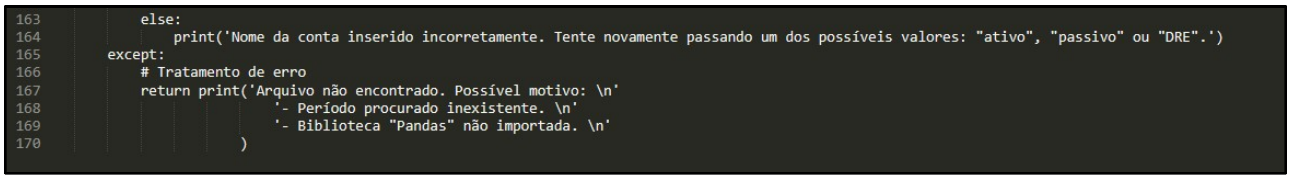
- Lines 165 to 170: Error handling related to searching for non-existent periods and when the required library for operation is not imported.

Function "buscar_relatorios": Its aim is to consolidate in a single table more than one period specified by the function "buscar_relatorio.", Figure 24.

To build this code, the function previously built was used to help in the consolidation of dates.

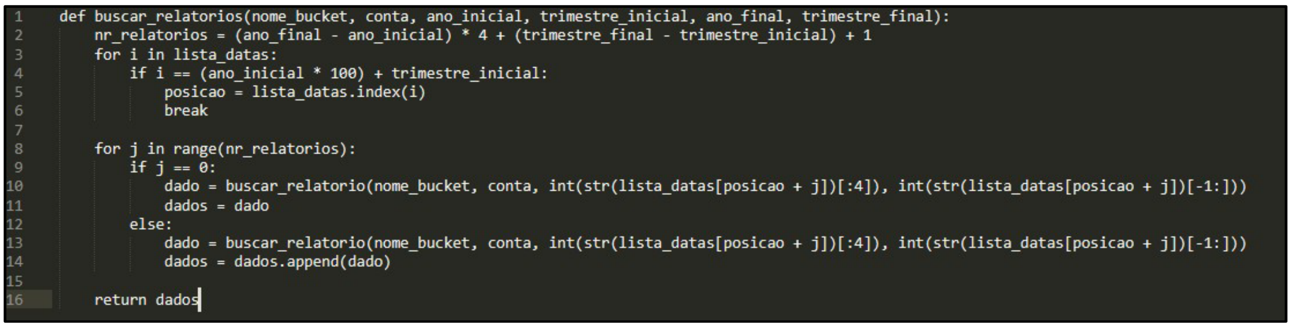
- Line 1: The function "buscar_relatorios" is defined; it the parameters "name_bucket" (ex: "ifdatatcc"), account (ex: "assets", "liabilities", or "DRE"), "initial_year" (ex: 2019), "initial_quarter" (ex: 1, 2, 3, or 4), "final_year" (ex: 2020), and "final_quarter" (ex: 1, 2, 3, or 4).
- Line 2: A calculation is created to find the number of reports to be extracted.
- Lines 3 to 6: A loop is needed to find the position of "start_date" in the date list.
- Lines 8 to 14: A loop runs that adds the number of reports calculated in line 2 from the start date. At the end of the procedure, the data are merged into a single table.
- Line 16: Returns the final information to the user.

Function "periodos_disponible": It has as objective to show the user which is the available report periods. It does not have parameters, Figure 25.



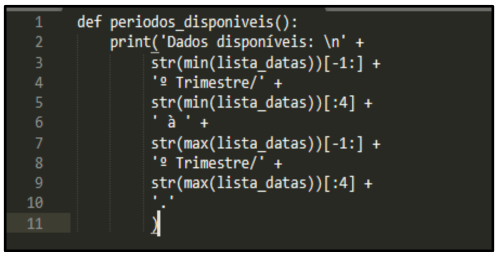
```
163             else:
164                 print('Nome da conta inserido incorretamente. Tente novamente passando um dos possíveis valores: "ativo", "passivo" ou "DRE".')
165         except:
166             # Tratamento de erro
167             return print('Arquivo não encontrado. Possível motivo: \n'
168                           '- Período procurado inexistente. \n'
169                           '- Biblioteca "Pandas" não importada. \n'
170                         )
```

Figure 23. Error handling of the "fetch_report" function. Source: Own authorship.



```
1   def buscar_relatorios(nome_bucket, conta, ano_inicial, trimestre_inicial, ano_final, trimestre_final):
2       nr_relatorios = (ano_final - ano_inicial) * 4 + (trimestre_final - trimestre_inicial) + 1
3       for i in lista_datas:
4           if i == (ano_inicial * 100) + trimestre_inicial:
5               posicao = lista_datas.index(i)
6               break
7
8       for j in range(nr_relatorios):
9           if j == 0:
10              dado = buscar_relatorio(nome_bucket, conta, int(str(lista_datas[posicao + j])[:4]), int(str(lista_datas[posicao + j])[-1:]))
11              dados = dado
12          else:
13              dado = buscar_relatorio(nome_bucket, conta, int(str(lista_datas[posicao + j])[:4]), int(str(lista_datas[posicao + j])[-1:]))
14              dados = dados.append(dado)
15
16      return dados
```

Figure 24. Handling the "fetch_reports" function. Source: Own authorship.



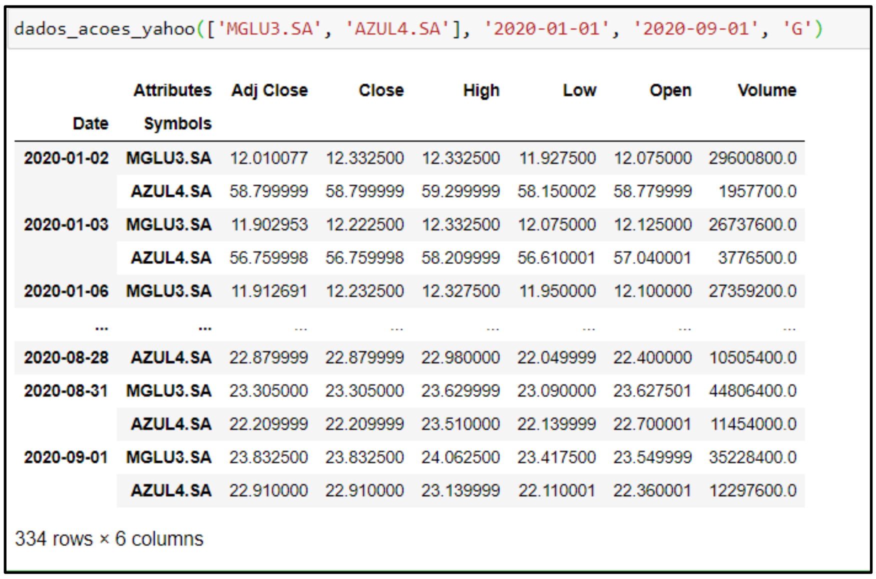
```
def periodos_disponiveis():
    print('Dados disponíveis: \n' +
          str(min(lista_datas))[-1:] +
          'º Trimestre/' +
          str(min(lista_datas))[:4] +
          ' à ' +
          str(max(lista_datas))[-1:] +
          'º Trimestre/' +
          str(max(lista_datas))[:4] +
          '.'
          )
```

Figure 25. Handling the "available_periods" function. Source: Own authorship.

4. Results and Findings

After the construction of the three methods, the information was consolidated into a single file from all the functions created in order to establish a vast repertoire of utilities to the users of the elaborated library. The result of each function is shown below, Figure 26.

- "data_actions_yahoo".
- "plot_graphic_closures", Figure 27.
- "returns_simple", Figure 28.
- "returns_log", Figure 29.
- "fetch_balancets", Figure 30.
- "fetch_report", Figure 31.
- "fetch_reports", Figure 32.
- "periods_available", Figure 33.

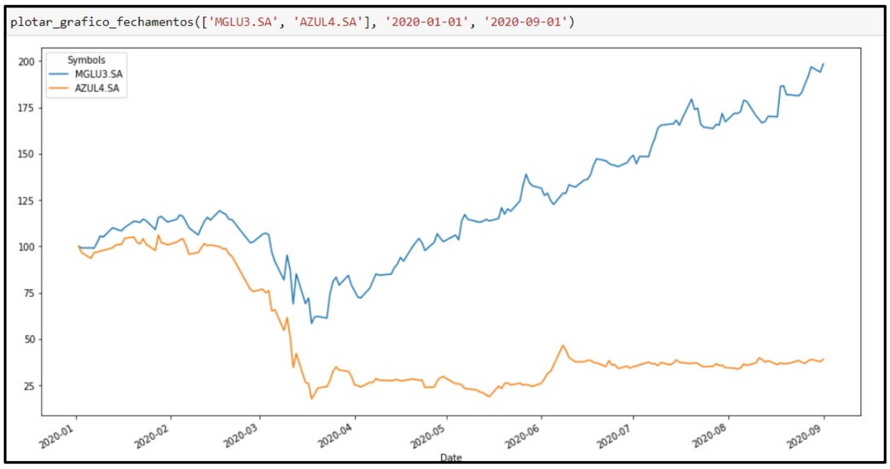


```
dados_acoes_yahoo(['MGLU3.SA', 'AZUL4.SA'], '2020-01-01', '2020-09-01', 'G')
```

	Attributes	Adj Close	Close	High	Low	Open	Volume
Date	Symbols						
2020-01-02	MGLU3.SA	12.010077	12.332500	12.332500	11.927500	12.075000	29600800.0
	AZUL4.SA	58.799999	58.799999	59.299999	58.150002	58.779999	1957700.0
2020-01-03	MGLU3.SA	11.902953	12.222500	12.332500	12.075000	12.125000	26737600.0
	AZUL4.SA	56.759998	56.759998	58.209999	56.610001	57.040001	3776500.0
2020-01-06	MGLU3.SA	11.912691	12.232500	12.327500	11.950000	12.100000	27359200.0
...	...	...	...	...	...	...	...
2020-08-28	AZUL4.SA	22.879999	22.879999	22.980000	22.049999	22.400000	10505400.0
2020-08-31	MGLU3.SA	23.305000	23.305000	23.629999	23.090000	23.627501	44806400.0
	AZUL4.SA	22.209999	22.209999	23.510000	22.139999	22.700001	11454000.0
2020-09-01	MGLU3.SA	23.832500	23.832500	24.062500	23.417500	23.549999	35228400.0
	AZUL4.SA	22.910000	22.910000	23.139999	22.110001	22.360001	12297600.0

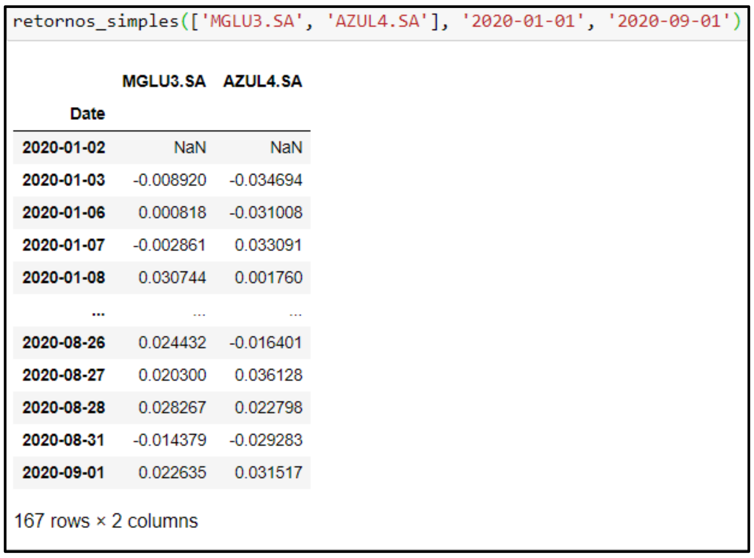
334 rows × 6 columns

Figure 26. Return of the function "dados_acoes_yahoo". Source: Own authorship.



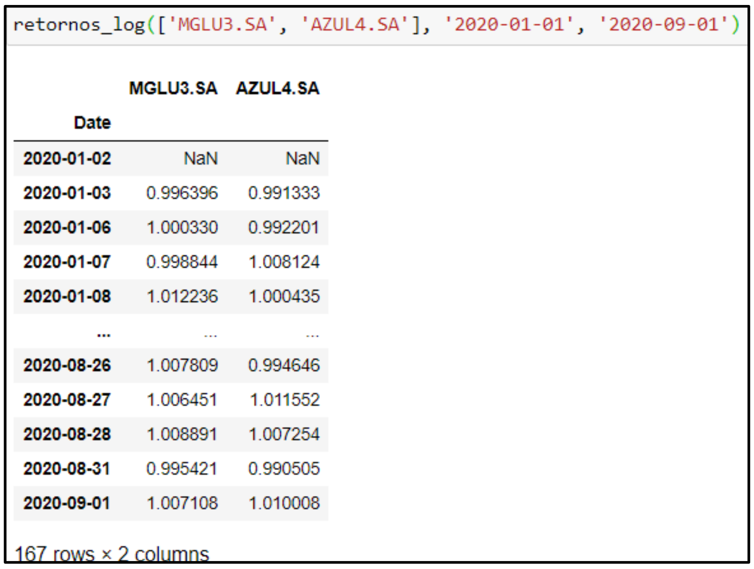
```
plotar_grafico_fechamentos(['MGLU3.SA', 'AZUL4.SA'], '2020-01-01', '2020-09-01')
```

Figure 27. Return of the function "plot_graphic_closures". Source: Own authorship.



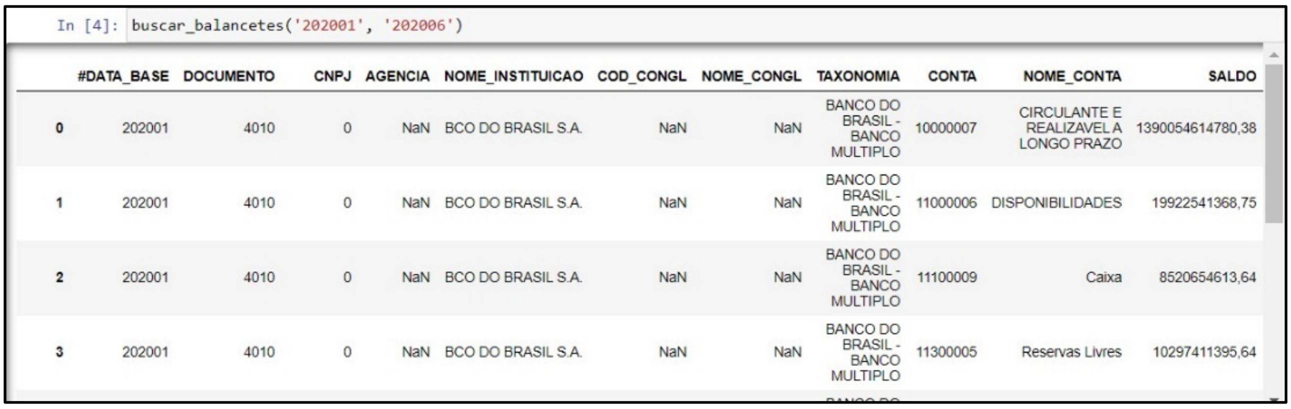
```
retornos_simples(['MGLU3.SA', 'AZUL4.SA'], '2020-01-01', '2020-09-01')

            MGLU3.SA   AZUL4.SA
      Date
2020-01-02       NaN        NaN
2020-01-03 -0.008920  -0.034694
2020-01-06  0.000818  -0.031008
2020-01-07 -0.002861   0.033091
2020-01-08  0.030744   0.001760
       ...       ...        ...
2020-08-26  0.024432  -0.016401
2020-08-27  0.020300   0.036128
2020-08-28  0.028267   0.022798
2020-08-31 -0.014379  -0.029283
2020-09-01  0.022635   0.031517

167 rows × 2 columns
```

Figure 28. Return from the "simple_returns" function. Source: Own authorship.



```
retornos_log(['MGLU3.SA', 'AZUL4.SA'], '2020-01-01', '2020-09-01')

            MGLU3.SA  AZUL4.SA
      Date
2020-01-02       NaN       NaN
2020-01-03  0.996396  0.991333
2020-01-06  1.000330  0.992201
2020-01-07  0.998844  1.008124
2020-01-08  1.012236  1.000435
       ...       ...       ...
2020-08-26  1.007809  0.994646
2020-08-27  1.006451  1.011552
2020-08-28  1.008891  1.007254
2020-08-31  0.995421  0.990505
2020-09-01  1.007108  1.010008

167 rows × 2 columns
```

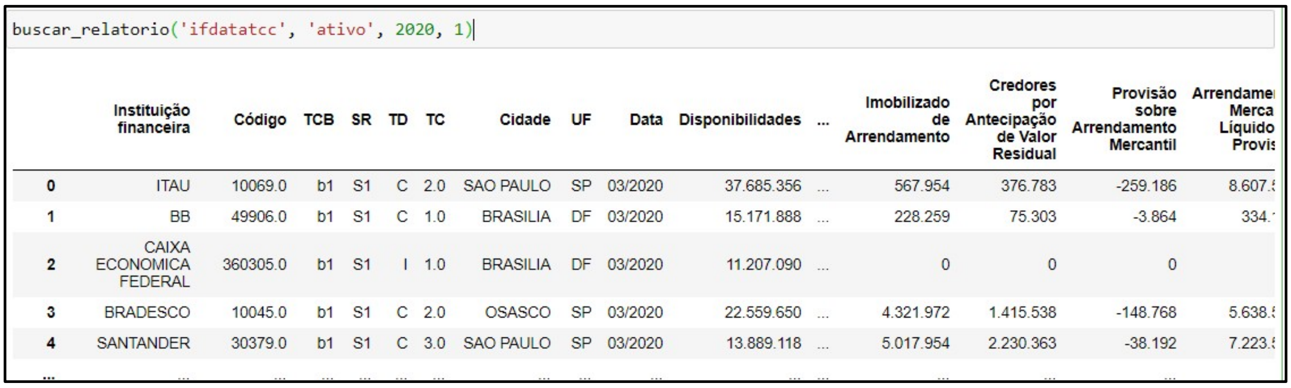
Figure 29. Return of the function "retornos_log". Source: Own authorship.



```
In [4]: buscar_balancetes('202001', '202006')
```

	#DATA_BASE	DOCUMENTO	CNPJ	AGENCIA	NOME_INSTITUICAO	COD_CONGL	NOME_CONGL	TAXONOMIA	CONTA	NOME_CONTA	SALDO
0	202001	4010	0	NaN	BCO DO BRASIL S.A.	NaN	NaN	BANCO DO BRASIL - BANCO MULTIPLO	10000007	CIRCULANTE E REALIZAVEL A LONGO PRAZO	1390054614780,38
1	202001	4010	0	NaN	BCO DO BRASIL S.A.	NaN	NaN	BANCO DO BRASIL - BANCO MULTIPLO	11000006	DISPONIBILIDADES	19922541368,75
2	202001	4010	0	NaN	BCO DO BRASIL S.A.	NaN	NaN	BANCO DO BRASIL - BANCO MULTIPLO	11100009	Caixa	8520654613,64
3	202001	4010	0	NaN	BCO DO BRASIL S.A.	NaN	NaN	BANCO DO BRASIL - BANCO MULTIPLO	11300005	Reservas Livres	10297411395,64

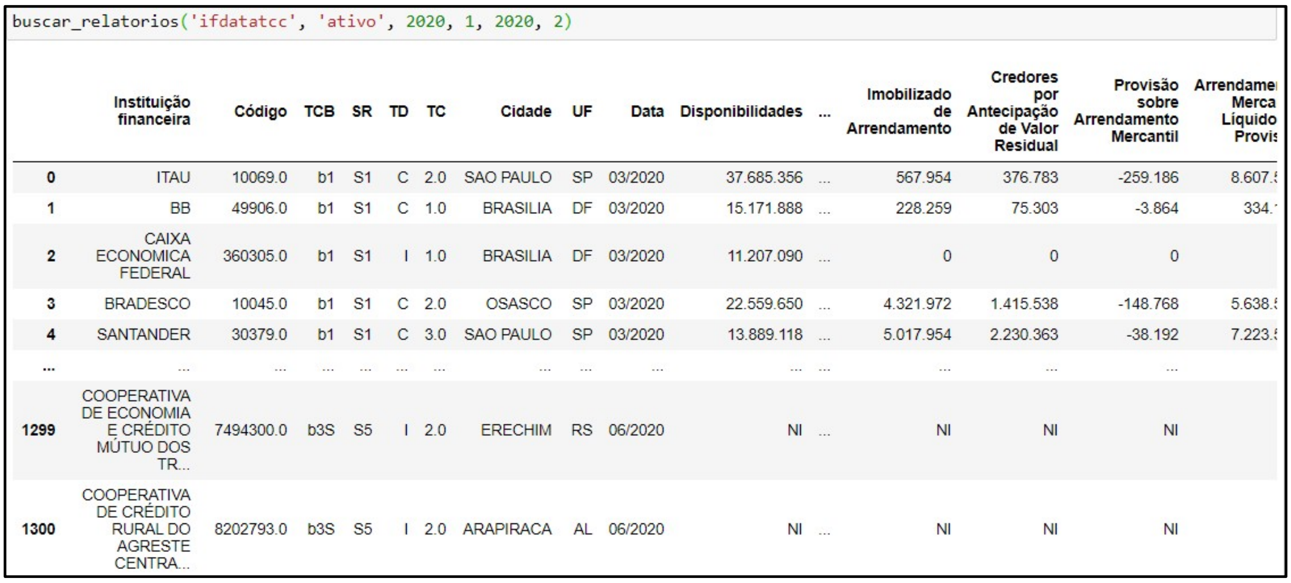
Figure 30. Return of the function "fetch_balancets". Source: Own authorship.



```
buscar_relatorio('ifdatatcc', 'ativo', 2020, 1)
```

	Instituição financeira	Código	TCB	SR	TD	TC	Cidade	UF	Data	Disponibilidades	...	Imobilizado de Arrendamento	Credores por Antecipação de Valor Residual	Provisão sobre Arrendamento Mercantil	Arrendame Merca Líquido Provis
0	ITAU	10069.0	b1	S1	C	2.0	SAO PAULO	SP	03/2020	37.685.356	...	567.954	376.783	-259.186	8.607.
1	BB	49906.0	b1	S1	C	1.0	BRASILIA	DF	03/2020	15.171.888	...	228.259	75.303	-3.864	334.
2	CAIXA ECONOMICA FEDERAL	360305.0	b1	S1	I	1.0	BRASILIA	DF	03/2020	11.207.090	...	0	0	0	
3	BRADESCO	10045.0	b1	S1	C	2.0	OSASCO	SP	03/2020	22.559.650	...	4.321.972	1.415.538	-148.768	5.638.
4	SANTANDER	30379.0	b1	S1	C	3.0	SAO PAULO	SP	03/2020	13.889.118	...	5.017.954	2.230.363	-38.192	7.223.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

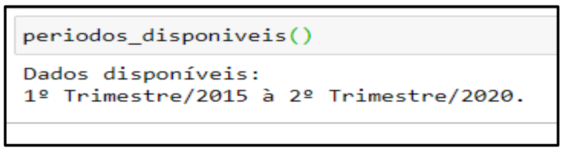
Figure 31. Return of the function "buscar_relatorio". Source: Own authorship.



```
buscar_relatorios('ifdatatcc', 'ativo', 2020, 1, 2020, 2)
```

	Instituição financeira	Código	TCB	SR	TD	TC	Cidade	UF	Data	Disponibilidades	...	Imobilizado de Arrendamento	Credores por Antecipação de Valor Residual	Provisão sobre Arrendamento Mercantil	Arrendame Merca Líquido Provis
0	ITAU	10069.0	b1	S1	C	2.0	SAO PAULO	SP	03/2020	37.685.356	...	567.954	376.783	-259.186	8.607.
1	BB	49906.0	b1	S1	C	1.0	BRASILIA	DF	03/2020	15.171.888	...	228.259	75.303	-3.864	334.
2	CAIXA ECONOMICA FEDERAL	360305.0	b1	S1	I	1.0	BRASILIA	DF	03/2020	11.207.090	...	0	0	0	
3	BRADESCO	10045.0	b1	S1	C	2.0	OSASCO	SP	03/2020	22.559.650	...	4.321.972	1.415.538	-148.768	5.638.
4	SANTANDER	30379.0	b1	S1	C	3.0	SAO PAULO	SP	03/2020	13.889.118	...	5.017.954	2.230.363	-38.192	7.223.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1299	COOPERATIVA DE ECONOMIA E CRÉDITO MÚTUO DOS TR...	7494300.0	b3S	S5	I	2.0	ERECHIM	RS	06/2020	NI	...	NI	NI	NI	
1300	COOPERATIVA DE CRÉDITO RURAL DO AGRESTE CENTRA...	8202793.0	b3S	S5	I	2.0	ARAPIRACA	AL	06/2020	NI	...	NI	NI	NI	

Figure 32. Return of the function "buscar_relatorios". Source: Own authorship.



```
periodos_disponiveis()

Dados disponíveis:
1º Trimestre/2015 à 2º Trimestre/2020.
```

Figure 33. Return of the function "periods_available". Source: Own authorship.

5. Conclusion

In summary, it is possible to conclude that data extraction tools are currently of extreme importance, in view of the large informational volume processed and the different sources of information available to users. It is interesting to point out that the profession of data analyst has become increasingly essential for companies (Lima, 2018). There is applicability for the techniques of capturing information in several fields, as in the area of health, in which evidence can be collected for research proof (Bradley, Curry, & Devers, 2007), as in the Business and Competitive Intelligence, where a company can probe the Web to acquire and analyze information about the activity of its competitors (Baumgartner et al., 2005; Chen et al., 2002), and as in crawling os Social Web platforms, to map consumer desires and understand, model and predict human behavior (Gkotsis et al., 2013; Catanese et al., 2011).

This work explored the *Web Scraping* technique, which is a main method in the context of the digital era. The extraction can be made from simple texts, as well as, with the use of more advanced techniques, from audio and even video material using the aid of artificial intelligence. Thus, the main contribution of the work was to structure the use of three distinct methods to structure a data collection process, the first being the search for stock trading prices on the stock exchange and graphically analyzing the data, the second the search for files in ".zip" format extracted directly from the HyperText Markup Language (HTML) code of the website's server page, and the third, the creation of files in the "Comma-Separated Values" (CSV) format in order to manipulate and organize data for further analysis. The codes are applied to information from SGBDs in Brazil; however, the method and coding can be adapted to other DBMSs in other countries. Therefore, the code of the work used is available for public enjoyment on the GitHub platform (Github, 2020). The results show that the construction of programs for data extraction can be a great ally in the efforts of information users. Once created, the time saving of such tools for the structuring of reports that bring value generation becomes obvious.

The main limitation of this study is related to the need to check whether downloading data from the site with scripts is allowed and does not violate local data protection laws. Finally, regarding future work for the evolution of the solutions developed in this project, we would like to mention the study of other techniques using resources such as the AWS EMR to create a virtual machine capable of processing large-scale files as well as the demonstration of the use of *machine learning* and *deep learning* techniques.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

Anaconda (2020) Individual Edition. https://www.anaconda.com/products/individual

Baumgartner, R., Fröhlich, O., Gottlob, G., Harz, P., Herzog, M., & Lehmann, P. (2005). *Web Data Extraction for Business Intelligence: The Lixto Approach*. Gesellschaft für Informatik eV.

Borges, L. E. (2014). *Python para desenvolvedores: Aborda Python 3.3* (p. 14). Novatec Editora. https://books.google.com.br/books?hl=pt-BR&lr=lang_pt&id=eZmtBAAAQBAJ&oi=fnd&pg=PA14&dq=python&ots=VDSosqIkmu&sig=TZ0MbKn058lnRJ9zgZrNLmoOFh4#v=onepage&q=python&f=false

Bradley, E. H., Curry, L. A., & Devers, K. J. (2007). Qualitative Data Analysis for Health Services Research: Developing Taxonomy, Themes, and Theory. *Health Services Research, 42,* 1758-1772. https://doi.org/10.1111/j.1475-6773.2006.00684.x

Buriol, T. M., Marco, B., & Argenta, M. A. (2009). *Acelerando o desenvolvimento eo processamento de análises numéricas computacionais utilizando python e cuda*. https://www.researchgate.net/profile/Marco_Argenta/publication/228683446_ACELERANDO_O_DESENVOLVIMENTO_EO_PROCESSAMENTO_DE_ANALISES_NUMERICAS_COMPUTACIONAIS_UTILIZANDO_PYTHON_E_CUDA/links/5630d6c908ae0530378cdf06.pdf

Catanese, S. A., De Meo, P., Ferrara, E., Fiumara, G., & Provetti, A. (2011, May). Crawling Facebook for Social Network Analysis Purposes. In *Proceedings of the International Conference on Web Intelligence, Mining and Semantics* (pp. 1-8). https://doi.org/10.1145/1988688.1988749

Chen, H., Chau, M., & Zeng, D. (2002). CI Spider: A Tool for Competitive Intelligence on the Web. *Decision Support Systems, 34,* 1-17. https://doi.org/10.1016/S0167-9236(02)00002-7

Chen, M., Mao, S., & Liu, Y. (2014). Big Data: A Survey. *Mobile Networks and Applications, 19,* 171-209. https://doi.org/10.1007/s11036-013-0489-0

Ferrara, E., De Meo, P., Fiumara, G., & Baumgartner, R. (2014). Web Data Extraction, Applications and Techniques: A Survey. *Knowledge-Based Systems, 70,* 301-323. https://doi.org/10.1016/j.knosys.2014.07.007

Github (2020) Fabiobragato/Finances. https://github.com/fabiobragato/finances.git

Gkotsis, G., Stepanyan, K., Cristea, A. I., & Joy, M. (2013, July). Self-Supervised Automated Wrapper Generation for Weblog Data Extraction. In *British National Conference on Databases* (pp. 292-302). Springer. https://doi.org/10.1007/978-3-642-39467-6_26

Laender, A. H., Ribeiro-Neto, B. A., Da Silva, A. S., & Teixeira, J. S. (2002). A Brief Survey of Web Data Extraction Tools. *ACM SIGMOD Record, 31,* 84-93. https://doi.org/10.1145/565117.565137

Laudon, K. C., & Laudon, J. P. (2014). *Management Information Systems: Managing the Digital Firm* (13th ed., p. 37). Prentice Hall. http://dinus.ac.id/repository/docs/ajar/Kenneth_C.Laudon,Jane_P_.Laudon_-_Management_Information_Sysrem_13th_Edition_.pdf

Lima, F. G. (2018). Análise de Riscos. Ed. Atlas.

Miyagusku, R. (2008). Curso Prático de SQL. Guia de Referência Completo Para Usar a linguagem SQL nos Bancos de Dados: MS SQL Server, Oracle, PostgreSQL, MySQL.

Naeem, M., Jamal, T., Diaz-Martinez, J., Butt, S. A., Montesano, N., Tariq, M. I., De-La-Hoz-Valdiris, E. et al. (2022). Trends and Future Perspective Challenges in Big Data. In *Advances in Intelligent Data Analysis and Applications* (pp. 309-325). Springer. https://doi.org/10.1007/978-981-16-5036-9_30