Bravisa Project

Overview

The Bravisa Project is an automated tool developed to streamline the creation of various reports, such as Bravisa Temple Tree lists (BTT List), EPS, SMR, PRS, ERS, IRS, etc. The software is designed to handle daily FB data, which can be inserted into a PostgreSQL database through the application. This data is then processed through a series of transformations that involve combining columns across different tables to generate the reports.

Key features of the Bravisa Project include:

- Data Insertion and Management: The tool allows users to insert daily FB data into the database and manage the data by deleting records for a specific day or a span of days.
- Report Generation: Users can generate various reports either together or separately. The reports can also be downloaded for further use.
- Data Transformation: The application automatically processes the inserted data through complex transformations to create the required reports by combining information from different tables.
- Industry Details Management: The tool includes a GUI page for adding new industry details into the database.

The project is built using Flask as the web framework, PostgreSQL as the database, and Python 3.10.9 for the backend logic.

The requirements.txt file lists the dependencies required for the project, which include various libraries and tools used for tasks such as data handling, report generation, and web interface management. Here are some key packages,

- Flask: For building the web application and handling HTTP requests.
- PostgreSQL: For managing the database.
- Pandas: For data manipulation and transformation.
- Numpy: For mathematical and statistical calculations.

Installation and setup

This section will guide you through the process of installing and setting up the Bravisa Project. Follow the steps below to get started.

Prerequisites

Before you begin, ensure you have the following installed on your system:

- Operating System:Windows.
- Python: Version 3.10.9 or later. Download Python.
- Package Manager: (pip or conda) Ensure pip is installed with Python.
- Database: Download PgAdmin. Download PostgreSQL.

Step 1: Clone the Repository

First, clone the project repository to your local machine using Git.

```
git clone https://github.com/yourusername/yourproject.git
cd yourproject
```

Step 2: Create a Virtual Environment (Optional but Recommended)

It's recommended to create a virtual environment to manage project dependencies.

```
python -m venv venv
```

Activate the virtual environment

```
venv\Scripts\activate
```

Step 3: Install Dependencies

Install the required Python packages using pip.

```
pip install -r requirements.txt
```

Step 4: Set Up the Database

Import the database from the backup file into PgAdmin.

Step 5: Configure Environment Variables

Go to the specified files and change the ENVIRONMENT VARIABLES to match your directory.

app.py

```
ohlc_folder = 'your working directory.......app\\\OHLCFiles\\'
index_ohlc_folder = 'your working directory.......app\\\IndexOHLCFiles\\'
index_files_folder = 'your working directory.......app\\index-files\\'
```

db_helper.py

```
    def db_connect(self):
        """ Creating the connection with local database to access data
        """
        if os.name == 'nt':
            ...
conn = psycopg2.connect(
    host='localhost',   # IP address from PgAdmin or localhost
    user='postgres',
    password='edsols',  # the password you gave while setting up the database
    database='BravisaDB', # database Name
)
```

fb_insert.py

```
my_path=os.path.abspath(os.path.dirname('your working directory''))
# change it to the directory of the app folder
```

All the directories variables for all the functions to intermediate_insert should be changed for the feature to work. If you're adding the intermediate insert into the software again, right now this feature is disabled.

```
 if(fbname == 'intermediate_insert'):

                    file_to_check = 'Working directory.....\\MissingData'+ '\\' +
'AnnualGeneralMeeting.csv'
```

ohlc.py
my_path = os.path.abspath(os.path.dirname('your working directory'))# working directory
 filepath = os.path.join(my_path, "IndexOHLCFiles\\")

Index_ohlc.py

my_path = os.path.abspath(os.path.dirname('you working directory'))# working directory
 filepath = os.path.join(my_path, "IndexOHLCFiles\\")

btt_list.py

my_path = os.path.abspath(os.path.dirname('your working directory'))
filepath = os.path.join(my_path, "index-files\\")

Step 5: Run the Application

To start the application, run the following command:

To update the changes made in the app.py or install the Bravisa.exe file use this command.

```
pyinstaller .\Bravisa.spec
```

For development purposes you can use this,

```
python app.py
```

Directory

1. Build - this folder contains all the build files for the bravisa.exe
2. dash_process - contains the modules to enable dashboard processes to visualize data.
3. dist - contains the downloaded files folder with all the downloaded reports
4. DownloadFiles - this serves the same purpose as the dist folder.
5. env - has the libraries and dependencies that you install if you create a virtual environment.
6. FBFiles - this folder is where FB file folders 1,2,3 must be extracted from the zip file.

7. index-files - this folder is where BSE500_Index.csv and ind_nifty500list.csv must be changed monthly.
8. IndexOHLCFiles - this is the folder where daily index OHLC files must be placed,example : ind_close_all_DATE.csv.
9. lib - this folder contains all the lib modules, such as OHLC, IndexOHLC, PE etc.
10. Log folder will contain the log files, if you enable the log functions across the software.
11. mf_analysis - this folder contains all the modules to generate mutual fund reports.
12. missing - this folder should be used to place all the missing data while using the intermediate insert.
13. OHLCFiles - this is the folder where the daily OHLC files are placed to generate the OHLC report, example files are, BhavCopy_BSE_CM_0_0_0_20240729_F_0000.CSV, cm01APR2024bhav.csv
14. outputs - folder will contain all the cleaned data, if the utils/sanitize.py module's san_in function is enabled.
15. reports - this folder contains all the modules to generate reports like EPS, EERS, SMR, PRS, IRS etc.
16. static - this folder has the static image used in the index.html
17. templates folder contains the static front end GUI pages.
18. utils - this folder contains all the utility modules such as db_helper.py, script_helper.py, sanitize.py, holiday.py etc.
19. app.py is the main entry point to the flask app and the software.
20. Bravisa.spec is used to build the Bravisa.exe file from the flask application.
21. requirements.txt - this contains all the libraries and their versions that are essential for running the application.
22. Other miscellaneous files will be explained later in the Database and Reports Relation chapter and the Development and Testing chapter.

Codebase and Codeflow

1. The software starts from the app.py python file, and progressively extends into multiple modules using all their functionality.
2. check_file_presence function checks for the required files to start any kind of action.
3. Statically there are two pages index.html and the industrymapping.html.
4. industrymapping.html - is responsible for inserting new industry details into the database.
5. Once a new Industry is inserted, make sure to add the newly added industry into the index_mapping variable in the download_csv() function.
6. The main process() connects to the db and gets required details along with what type of action to perform like, start_date, end_date, is_holiday, insert or

generate or insert and generate, delete reports etc. Download data offers all reports or single reports.

7. insert_data uses the lib/fb_insert.py to insert all the FB files into the database with specific delete query for every table.
8. report_generation uses the utils/script_helper.py where the module generates a series of reports based on the day, depending on the day the if it's the first day of the month BTT list is generated and so on. EPS() uses reports.EPS.py , EERS() uses reports/EERS.py, SMR() - uses reports/SMR.py, OHLC - uses lib/ohlc.py, IndexOHLC - uses lib/indexohlc.py, PE - uses lib/ PE.py, PRS() - uses reports/PRS.py, IRS - uses reports/IRS.py
9. The script_helper.py file contains more reports and dash_process modules which are disabled.
10. delete_data() function deletes all the reports data from the database within the given date.
11. download_csv() function lets you download reports within the date range, make sure to add the index_map for a new industry to ensure proper index names for industries while downloading the csv files.

Development and Testing

Development and testing is done with jupyter notebook files found throughout the directory, Every ipynb file is well documented with what every cell executes. Data flow for every report is documented in a flowchart manner and is available in the flow chart folder.