

# README for the Stock Portfolio MYSQL Database Management Script

## Team Formation

Team name: HealthEdBot

Names & USC ID:

Cici Chang (8706354957)

Daoyang Li (7168949829)

Yifan Yang(8386626967)

<https://github.com/DSCI560>

## Introduction

This Python script is designed for managing stock portfolios through interactions with a MySQL database and fetching real-time stock data using the yfinance library. It enables users to add and remove stocks from portfolios, display portfolio contents, and ensure the uniqueness of portfolio names.

## Requirements

- Linux ubuntu
- Python 3.x
- MySQL Database
- Libraries: yfinance, mysql.connector

Install the necessary Python libraries using pip:

```
pip install yfinance mysql-connector-python
```

## Setup

MySQL Database Configuration:

Ensure the MySQL database server is running.

Create a database named stock\_portfolio and the tables (portfolio, stocks).

```

mysql> USE stock_database;
Database changed
mysql> CREATE TABLE IF NOT EXISTS portfolios(
-> id INT AUTO_INCREMENT PRIMARY KEY,
-> name VARCHAR(255) NOT NULL,
-> creation_date DATE NOT NULL
-> );
Query OK, 0 rows affected (0.04 sec)

mysql> SHOW TABLES;
+-----+
| Tables_in_stock_database |
+-----+
| portfolios                |
+-----+
1 row in set (0.00 sec)

mysql> CREATE TABLE IF NOT EXISTS stocks(
-> id INT AUTO_INCREMENT PRIMARY KEY,
-> symbol VARCHAR(10) NOT NULL,
-> date DATE NOT NULL,
-> open DECIMAL(10,2),
-> high DECIMAL(10,2),
-> low DECIMAL(10,2),
-> close DECIMAL(10,2),
-> dividends DECIMAL(10,2),
-> portfolio_id INT,
-> FOREIGN KEY (portfolio_id) REFERENCES portfolios(id) ON DELETE CASCADE);
Query OK, 0 rows affected (0.03 sec)

mysql> SHOW TABLES;
+-----+
| Tables_in_stock_database |
+-----+
| portfolios                |
| stocks                    |
+-----+
2 rows in set (0.00 sec)

mysql>

```

#### Database Connection:

Establish the database connection parameters (host, user, password, database) in the Python script to match MySQL server credentials.

```

daoyang@daoyang-virtual-machine:~/Desktop/daoyang_7168949829/lab3$ python3 stock.py
(1, 'apple', datetime.date(2023, 8, 20))
(2, '[value-2]', datetime.date(2023, 1, 31))
(3, 'tech', datetime.date(2024, 2, 1))
(4, 'tech', datetime.date(2024, 2, 1))
(5, 'new', datetime.date(2024, 2, 1))
/home/daoyang/.local/lib/python3.10/site-packages/yfinance/utils.py:775: FutureWarning:
re version. Use pd.to_timedelta instead.
    df.index += _pd.TimedeltaIndex(dst_error_hours, 'h')

```

Date	Open	High	...	Dividends	Stock Splits
2023-01-03 00:00:00-05:00	129.555841	130.172390	...	0.0	0.0
2023-01-04 00:00:00-05:00	126.184691	127.944857	...	0.0	0.0
2023-01-05 00:00:00-05:00	126.423353	127.059795	...	0.0	0.0
2023-01-06 00:00:00-05:00	125.309594	129.565795	...	0.0	0.0
2023-01-09 00:00:00-05:00	129.744788	132.668449	...	0.0	0.0

```

[5 rows x 7 columns]
Stock added successfully
Stock added successfully
Stock added successfully
Stock added successfully
Stock added successfully
Stock added successfully
Stock added successfully
Stock added successfully
Stock added successfully
Stock added successfully
Stock added successfully

```

```

Portfolio ID: 1, Creation Date: apple
Stocks included:

Portfolio ID: 2, Creation Date: [value-2]
Stocks included:

Portfolio ID: 3, Creation Date: tech
Stocks included: AAPL, MSFT, AMZN, GOOGL, JPM, JNJ, KO, BA, NVDA, AAPL

Portfolio ID: 4, Creation Date: tech
Stocks included:

Portfolio ID: 5, Creation Date: new
Stocks included: AAPL, MSFT, AMZN, GOOGL, TSLA, JPM, JNJ, KO, BA, NVDA, AAPL, MSFT, AMZN, GOOGL,

Database error: Unread result found
Portfolio ID: 1, Creation Date: apple
Stocks included:

Portfolio ID: 2, Creation Date: [value-2]
Stocks included:

Portfolio ID: 3, Creation Date: tech
Stocks included: AAPL, MSFT, AMZN, GOOGL, JPM, JNJ, KO, BA, NVDA, AAPL

Portfolio ID: 4, Creation Date: tech
Stocks included:

Portfolio ID: 5, Creation Date: new
Stocks included: AAPL, MSFT, AMZN, GOOGL, TSLA, JPM, JNJ, KO, BA, NVDA, AAPL, MSFT, AMZN, GOOGL,

```

```

daoyang@daoyang-virtual-machine:~/Desktop/daoyang_7168949829/lab3$ python3 data_preprocessing.py
/home/daoyang/.local/lib/python3.10/site-packages/yfinance/utils.py:775: FutureWarning: The 'unit' keyword in TimedeltaIndex construction is deprecated and will be removed in a future version. Use pd.to_timedelta instead.
df.index += _pd.TimedeltaIndex(dst_error_hours, 'h')

```

	Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
0	2023-01-03 00:00:00-05:00	129.555841	130.172390	123.479803	124.374802	112117500	0.0	0.0
1	2023-01-04 00:00:00-05:00	126.184691	127.944857	124.384755	125.657639	89113600	0.0	0.0
2	2023-01-05 00:00:00-05:00	126.423345	127.059787	124.066524	124.325073	80962700	0.0	0.0
3	2023-01-06 00:00:00-05:00	125.309579	129.565780	124.195801	128.899506	87754700	0.0	0.0
4	2023-01-09 00:00:00-05:00	129.744803	132.668464	129.168025	129.426575	70790800	0.0	0.0

```

Please choose a type among = (Forward Filling: ffill, Backward Filling: bfill, Interpolate: linear) --> ffill
/home/daoyang/Desktop/daoyang_7168949829/lab3/data_preprocessing.py:17: FutureWarning: DataFrame.fillna with 'method' is deprecated and will raise in a future version. Use obj.ffill() or obj.bfill() instead.
data.fillna(method = miss_val, inplace = True)

```

	Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
0	2023-01-03 00:00:00-05:00	129.555841	130.172390	123.479803	124.374802	112117500	0.0	0.0
1	2023-01-04 00:00:00-05:00	126.184691	127.944857	124.384755	125.657639	89113600	0.0	0.0
2	2023-01-05 00:00:00-05:00	126.423345	127.059787	124.066524	124.325073	80962700	0.0	0.0
3	2023-01-06 00:00:00-05:00	125.309579	129.565780	124.195801	128.899506	87754700	0.0	0.0
4	2023-01-09 00:00:00-05:00	129.744803	132.668464	129.168025	129.426575	70790800	0.0	0.0

```

The 'date' attribute is not in datetime format.
converted to datetime format
/home/daoyang/Desktop/daoyang_7168949829/lab3/data_preprocessing.py:32: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.
For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method([col]: value, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.

```

	Date	Open	High	Low	Close	Volume	Dividends	Stock Splits	daily_returns
0	2023-01-03 00:00:00-05:00	129.555841	130.172390	123.479803	124.374802	112117500	0.0	0.0	0.000000
1	2023-01-04 00:00:00-05:00	126.184691	127.944857	124.384755	125.657639	89113600	0.0	0.0	0.010314
2	2023-01-05 00:00:00-05:00	126.423345	127.059787	124.066524	124.325073	80962700	0.0	0.0	-0.010605
3	2023-01-06 00:00:00-05:00	125.309579	129.565780	124.195801	128.899506	87754700	0.0	0.0	0.036794
4	2023-01-09 00:00:00-05:00	129.744803	132.668464	129.168025	129.426575	70790800	0.0	0.0	0.004089

## Usage

Run the python script 'stock.py' directly through the terminal. The script includes a test function **test\_portfolio\_management()** demonstrating the functionalities, such as adding stocks to a portfolio, removing a stock, and displaying portfolios.

Run the python script 'data\_preprocessing.py' directly through the terminal.

The script consists of a function `fetch_stock_data` that downloads stock data for a given symbol between specified start and end dates. It then processes this data by filling missing values as specified by the user, ensures the date column is in datetime format, calculates daily returns, and saves the processed data to a CSV file.

## Main Functions

- **fetch\_stock\_data(stock\_symbol, start\_date, end\_date):** Fetches historical data for a specified stock symbol from Yahoo Finance.
- **check\_stock\_validity(stock\_symbol):** Validates the existence of a stock symbol in the market data.
- **add\_portfolio(portfolio\_name, creation\_date):** Adds a new portfolio to the database if the name is unique.
- **add\_stock\_to\_portfolio(portfolio\_name, stock\_symbol):** Adds a stock to an existing portfolio after validating the stock symbol and checking if the portfolio exists.
- **remove\_stock\_from\_portfolio(portfolio\_name, stock\_symbol):** Removes a stock from a specified portfolio.
- **display\_portfolios():** Displays all portfolios and their associated stocks.

