Data Collection and ETL

Three data tables were generated in the “stock\_db” by a series of python programs. Data flew from the state of extraction, transformation and finally uploading. During the extraction stage, the data files stock, market capacity and cpi would be collected through API from **data.nasdaq.com**. Afterwards, selected data would be transformed to suitable format and uploaded onto the Cloud database via MySQL with MySQL-Connector.

Database

1. stock\_db. /\* for captured and transformed data (with API) from data.nasdaq.com \*/

Table Schema for stock\_db

1. stocks - Stock Table
2. market\_capacity - Market Capacity Tables
3. cpi - Consumer Price Index (CPI)

1) stocks (Consolidate\_Stocks.csv)

|  |  |  |
| --- | --- | --- |
| Index | Field Name | Description |
| 1 | seq\_number | Record sequence number |
| 2 | ticker | Stock code |
| 3 | txdate | Transaction date |
| 4 | open\_price | Opening price |
| 5 | day\_high | Day High |
| 6 | day\_low | Day Low |
| 7 | close\_price | Closing price |
| 8 | ex\_dividend | Dividend |
| 9 | split\_ratio | Stock split ratio |
| 10 | adj\_open | Adjusted opening price |
| 11 | adj\_high | Adjusted high price |
| 12 | adj\_low | Adjusted low price |
| 13 | adj\_close | Adjusted closing price |
| 14 | adj\_volume | Adjusted transaction volume |

2) market\_capacity (Consolidate\_Market\_Capacity.csv)

|  |  |  |
| --- | --- | --- |
| Index | Field Name | Description |
| 1 | seq\_number | Sequence number |
| 2 | ticker | Stock code |
| 3 | txdate | Transaction date |
| 4 | short\_vol | Short volume |
| 5 | total\_vol | Total volume |

3) cpi (USA\_Transformed\_CPI.csv)

|  |  |  |
| --- | --- | --- |
| Index | Field Name | Description |
| 1 | seq\_number | Sequence number |
| 2 | txdate | Transaction date |
| 3 | CPI | Consumer Price Index per month |

Function Listing

|  |  |  |
| --- | --- | --- |
|  | Function | Description |
| 1 | .env | Environment files |
| 2 | requirements.txt | Installation package requirement files |
| 3 | connect\_db.py | Database connection |
| 4 | nasdaq\_cpi\_data.py | Collect CPI data |
| 5 | nasdaq\_market\_capacity.py | Collect market capacity data |
| 6 | nasdaq\_stock\_data.py | Collect stock data |
| 7 | transform\_cpi\_data.py | Transform CPI data |
| 8 | transform\_market\_capacity.py | Transform market capacity data |
| 9 | transform\_stock\_data.py | Transform stock data |
| 10 | load\_data.py | Load data to database |

Data Warehouse

Amazon Relational Database Service (AWS RDS), along with the DB engine MySQL, was chosen to operate our relational databases in the AWS Cloud in our project. By connecting AWS RDS and MySQL workbench with the RDS DB instance endpoint, all data was securely uploaded onto the Cloud and successfully retrieved by other admin users for further data analysis. On RDS, two connected databases have been deployed using MySQL script, which respectively contains three tables for raw-data storage and three tables for transformed-data analysis. We designed to scale our databases separately because of better security and higher efficiency.

Data Warehouse

1. stock\_warehouse /\* Clean data for data analysis and data visualisation \*/

Table Schema for stock\_warehouse

1. BI\_stocks – stocks for data analysis
2. BI\_stock\_mth – consolidated stocks for analysis by month
3. BI\_cpi – consolidated cpi data

1) BI\_stocks

|  |  |  |
| --- | --- | --- |
| Index | Field Name | Description |
| 1 | ticker | Stock code |
| 2 | txdate | Transaction date |
| 3 | open\_price | Opening price |
| 4 | day\_high | Day High |
| 5 | day\_low | Day Low |
| 6 | close\_price | Closing price |
| 7 | ex\_dividend | Dividend |
| 8 | short\_vol | Short volume |
| 9 | total\_vol | Total volume |

2) BI\_stocks\_mth

|  |  |  |
| --- | --- | --- |
| Index | Field Name | Description |
| 1 | ticker | Stock code |
| 2 | yearMonth | Transaction date |
| 3 | mth\_open\_price | Opening price |
| 4 | mth\_day\_high | Day High |
| 5 | mth\_day\_low | Day Low |
| 6 | mth\_close\_price | Closing price |
| 7 | mth\_short\_vol | Short volume |
| 8 | mth\_total\_vol | Total volume |

3) BI\_cpi

|  |  |  |
| --- | --- | --- |
| Index | Field Name | Description |
| 1 | txdate | Transaction date |
| 2 | cpi | Consumer Price Index per month |

The SQL Scripts For The Database

/\* database (raw data) \*/

Create database stock\_db;

use stock\_db;

/\* Tables for Samuel's raw data: stocks, market\_capacity, CPI \*/

Create table stocks (

seq\_number int UNSIGNED not null AUTO\_INCREMENT,

ticker VARCHAR(7),

txdate date,

open\_price decimal(10, 3),

day\_high decimal(10, 3),

day\_low decimal(10, 3),

close\_price decimal(10, 3),

ex\_dividend decimal(5, 3),

split\_ratio decimal(4, 2),

adj\_open decimal(10, 3),

adj\_high decimal(10, 3),

adj\_low decimal(10, 3),

adj\_close decimal(10, 3),

adj\_volume int(15),

PRIMARY KEY (seq\_number)

);

Create table market\_capacity (

seq\_number int UNSIGNED not null AUTO\_INCREMENT,

ticker VARCHAR(7),

txdate date,

short\_vol int(15),

total\_vol int(15),

PRIMARY KEY (seq\_number)

);

Create table cpi (

seq\_number int UNSIGNED not null AUTO\_INCREMENT,

txdate date,

CPI decimal(10, 4),

PRIMARY KEY (seq\_number)

);

The SQL Scripts For Data Warehouse

/\* data warehouse (data analysis) \*/

CREATE DATABASE stock\_warehouse;

USE stock\_warehouse;

/\* Tables Addison's BI analysis: BI\_stocks, BI\_stocks\_mth, BI\_cpi \*/

CREATE TABLE BI\_stocks SELECT stocks.ticker, stocks.txdate, stocks.open\_price, stocks.day\_high, stocks.day\_low, stocks.close\_price, stocks.ex\_dividend, market\_capacity.short\_vol, market\_capacity.total\_vol

FROM stock\_db.stocks

LEFT JOIN stock\_db.market\_capacity

USING (seq\_number);

Create table BI\_stocks\_mth

select ticker, DATE\_FORMAT(`txdate`, '%Y-%m') AS yearMonth,

CAST(avg(open\_price) AS decimal(10, 3)) AS mth\_open\_price,

CAST(avg(day\_high) AS decimal(10, 3)) AS mth\_day\_high,

CAST(avg(day\_low) AS decimal(10, 3)) AS mth\_day\_low,

CAST(avg(close\_price) AS decimal(10, 3)) AS mth\_close\_price,

CAST(avg(short\_vol) AS decimal(15, 2)) AS mth\_short\_vol,

CAST(avg(total\_vol) AS decimal(15, 2)) AS mth\_total\_vol

from BI\_stocks

group by ticker, yearMonth;

CREATE TABLE BI\_cpi SELECT txdate, cpi FROM stock\_db.cpi;

Data Visualisation

Microsoft Power BI is the data visualisation tool for presenting the behaviour of the dataset by creating a report for our project. It is only allowed to be installed on Microsoft Windows OS and needs to be connected with the MySQL database in our project. The three tables from MySQL data warehouse were imported for analysis. DAX (Data Analysis Expressions) was chosen to analyse the data because data can be better understood and interpreted. Through its library of functions and operators, it builds formulas and expressions to simplify the task.