ETL Technical Report

The Money Maker$: Agust Erlingsson, Eric Phieler, Nicole Rothfield, Matthew Terhune

ETL (Extract, Transform, Load) is the process by which data is extracted, transformed to suit specific business needs, and loaded into a final database that can be used for future analysis or business use. This report outlines an example of ETL using CSV files, Python Pandas scripting, and a PostgreSQL database.

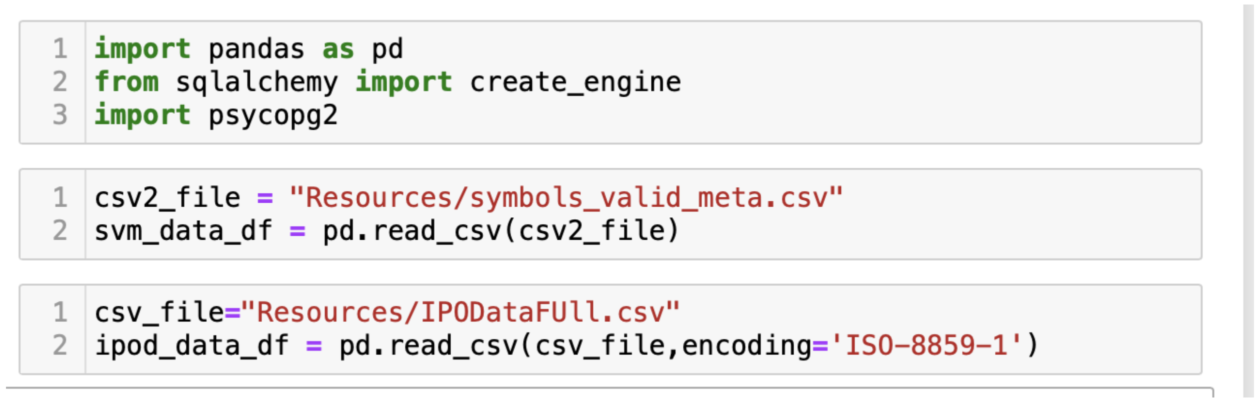
Extract

While the ‘E’ process can utilize many types of data sources (CSV files, JSON files, HTML tables, SQL databases, Spreadsheets, etc.), this project used two CSV files regarding stock market data sourced from Kaggle.com.

Link to dataset 1: https://www.kaggle.com/jacksoncrow/stock-market-dataset

Link to dataset 2: https://www.kaggle.com/proselotis/financial-ipo-data

The datasets were downloaded and added to a directory named “Resources”. Using Pandas in a Jupyter Notebook, the CSV files were extracted to be transformed as outlined below.



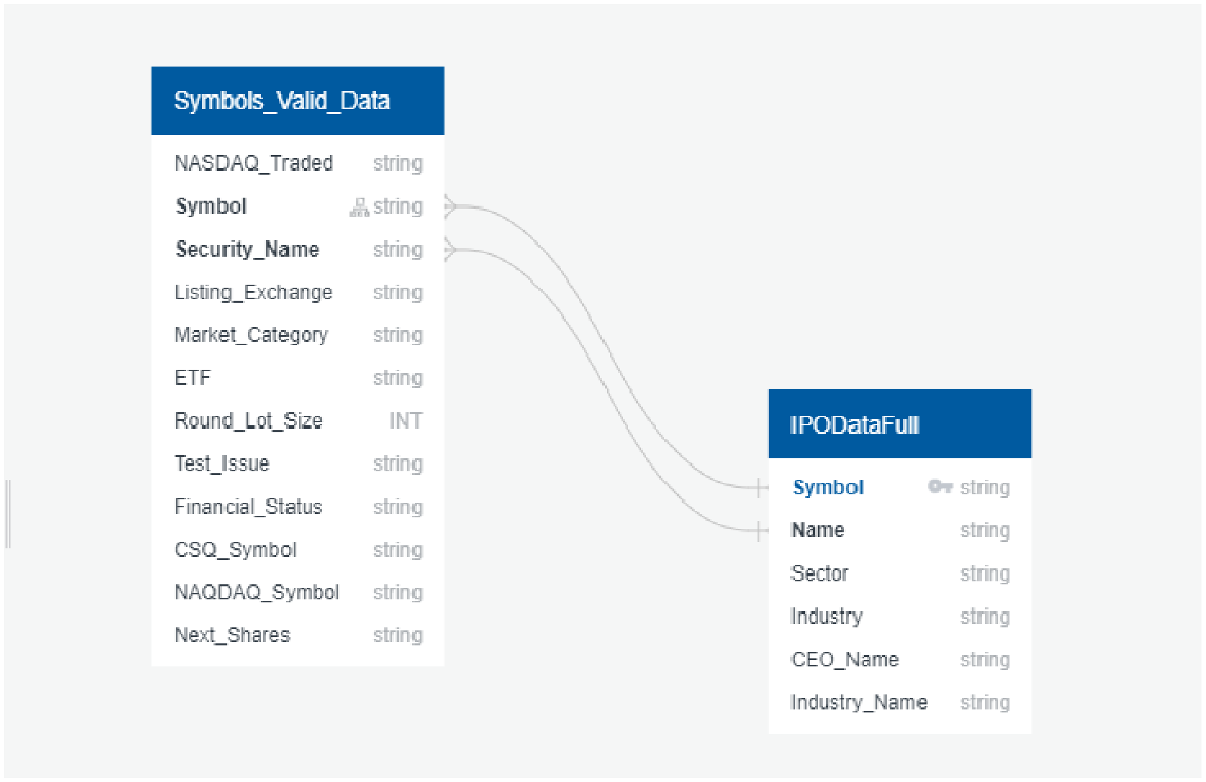
```
import pandas as pd
from sqlalchemy import create_engine
import psycopg2
```

```
csv2_file = "Resources/symbols_valid_meta.csv"
svm_data_df = pd.read_csv(csv2_file)
```

```
csv_file="Resources/IPODataFUll.csv"
ipod_data_df = pd.read_csv(csv_file,encoding='ISO-8859-1')
```

Transform

An Entity Relationship Diagram (ERD) was made by using This Website to determine existing relationships between entities via the two data sets. The .csv files were opened and analyzed to determine the names of the columns to be input into each entity. Then, it was determined which parts of each entity would be strings or integers. After all data points were inserted in both of the entities, two foreign keys were set, “Symbol” and “Security_Name.” The primary keys, as set in the SQL Schema, were an ID number automatically set, corresponding to each stock ticker symbol.



Once the table columns for the database are identified, use Pandas to reduce the original data to the columns necessary by creating a new DataFrames as seen below:

```
info_svm_data_df=svm_data_df [["Symbol","Nasdaq Traded","Security Name","CQS Symbol","NASDAQ Symbol"]]
info_svm_data_df.head()
```

	Symbol	Nasdaq Traded	Security Name	CQS Symbol	NASDAQ Symbol
0	A	Y	Agilent Technologies, Inc. Common Stock	A	A
1	AA	Y	Alcoa Corporation Common Stock	AA	AA
2	AAAU	Y	Perth Mint Physical Gold ETF	AAAU	AAAU
3	AACG	Y	ATA Creativity Global - American Depositary Sh...	NaN	AACG
4	AADR	Y	AdvisorShares Dorsey Wright ADR ETF	AADR	AADR

```
info_ipod_data_df=ipod_data_df[["Symbol","Name","Sector","Industry","CEOName","PresidentName"]]
info_ipod_data_df.head()
```

	Symbol	Name	Sector	Industry	CEOName	PresidentName
0	A	Agilent Technologies, Inc.	Capital Goods	Biotechnology: Laboratory Analytical Instruments	Michael McMullen	Michael McMullen
1	AAC	AAC Holdings, Inc.	Health Care	Medical Specialities	Michael Cartwright	Michael Nanko
2	AAOI	Applied Optoelectronics, Inc.	Technology	Semiconductors	Chih-Hsiang Lin	Chih-Hsiang Lin
3	AAP	Advance Auto Parts Inc	Consumer Services	Other Specialty Stores	Thomas Greco	Thomas Greco
4	AAT	American Assets Trust, Inc.	Consumer Services	Real Estate Investment Trusts	Ernest Rady	Ernest Rady

These are the final tables that will be loaded into the database.

Load

Load the data into a PostgreSQL Relational Database.

Open pgAdmin 4 and connect to a local server. Once connected, create a new database and tables accordingly using the SQL Schema from the ERD tool.

Using sqlalchemy in the jupyter notebook, connect to the database:

```
con_string = "postgres:PASSWORDHERE@localhost:5432/Stocks"
engine=create_engine(f'postgresql://{con_string}')
```

Check for successful connection to the database and confirm that the tables have been created:

```
engine.table_names()
```

Use the Pandas .to_sql function to upload the transformed DataFrames to the database as shown below:

```
: info_ipod_data_df.to_sql(name='IPODataFull',con=engine,if_exists='append', index=False)
```

```
: info_svm_data_df.to_sql(name='symbols_valid_meta',con=engine,if_exists='append', index=False)
```

Ensure that all desired data is loaded correctly by running the SELECT * FROM command in the jupyter notebook.

```
pd.read_sql_query('SELECT * FROM "IPODataFull"', con=engine).head()
```

```
pd.read_sql_query('SELECT * FROM symbols_valid_meta', con=engine).head()
```