# **Title**: Step by Step document to the ETL Process using local files

## **Author**: Imanpreet Singh

## **Date**: 27th Jan 2022

## **Change Logging:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version |  | Author | Date | Description |
|  |  |  |  |  |
|  |  |  |  |  |

## **Scenario**:

Every weekday stock open and close price is provided.

Stock open price is available in a database. (Table = **int.openstockprice**)

Stock close price file is provided at the end of the day in a csv file (name = **closestockprice.csv**) and loaded in the landing area.

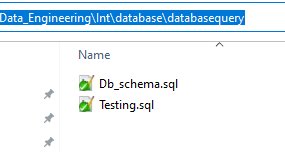
## **Requirement**:

At the end of the day, open stock price and close stock price needs to be loaded into one table in the database over which data analysis can be done. (Table = int.stockprice)

## **Pre-requisites:**

Before using the scripts and modules please follow below step to set up the system.

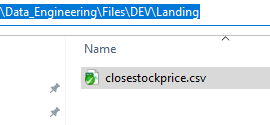
* Create Database Schema. Use ‘**Db\_schema.sql’** from below path.



Insert Stock open price data in the table **int.openstockprice** for current date using **Testing.sql** script

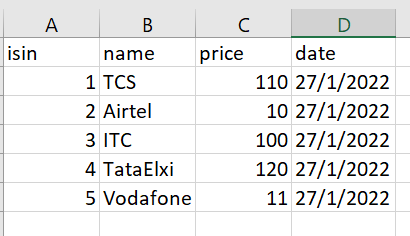


* Update below file in the landing area with the current date data.:

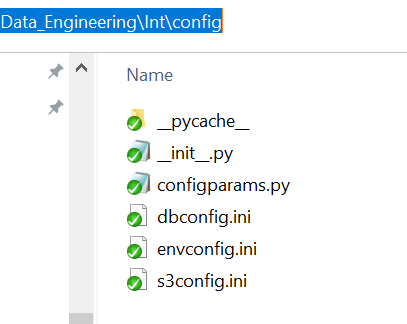


For 27th Jan below is the example.

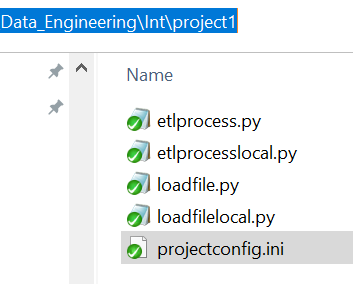
*As it is a stock price data, at weekend there will be no file.*



* Update config files as per your system:
  + **dbconfig.ini:** Update configuration about your source and target database. In our case, both are same.
  + **envconfig.ini:** Update configuration that will change as and when moved to different environments. e.g., archive path, landing path, file path etc.



* Update config files related to the project you are working.
  + **projectconfig.ini:** For our **stock price** solution, we need to provide source and target tables, file name etc. that can be provided here and will not be specific to other projects.



## **Testing:**

Once pre-requisites are met, we are ready with our process.

### **Load file from Landing to File area:**

#### Script - loadfilelocal.py environment project

* + Input variable
    1. environment 🡪 DEV
    2. project 🡪 stockprice
  + **python C:\Git\Data\_Engineering\Int\project1\loadfilelocal.py DEV stockprice**
  + This script will accept environment name and project name. This script can be used across different projects.

This will perform following tasks:

1. Load the file from **landing area to process area.**
2. If there is already a file in landing area then it will archive that file first to Archive area and add stock price date as suffix to the file. e.g., if file contains 27Jan 2022 data then that file will be archived as **filename\_27012022**
3. Once file is archived then file is moved from landing to process area

### **Perform ETL process:**

#### Script -etlprocesslocal**.py environment project**

* + Input variable
    1. environment 🡪 DEV
    2. project 🡪 stockprice
  + **python C:\Git\Data\_Engineering\Int\project1\etlprocesslocal.py DEV stockprice**
  + This script will accept environment name and project name.

This will perform following tasks:

##### Get the date for which data need to be processed.

* + 1. For first time, it will be current date
    2. Post that, maximum date from the target tables - **int.stockprice** is fetched and a day is added to it.
    3. If new date is weekend, then accordingly dates are added to get next weekday. (this is to make sure we are not loading same day into target table)

##### Load the data from database into the data frame – **open stock price data.**

1. Create WHERE condition to get the data from the database based on the date expected (as per step 1)
2. Create the select query based on the configuration and where condition (only columns defined in config will be fetched)
3. Use the query to get data from the database into data frame.
4. Check if data is there or not
5. Standardize DATE columns and PRICE columns

##### Load the data from csv file in the process area into data frame. – **close stock price data.**

1. Check file is not empty
2. Check the date present is for the date expected (as per step 1)
3. Check the columns for mandatory columns i.e., columns matching the open price data frame.
4. Standardize DATE columns and PRICE columns

##### Join both the data frames to create a final data frame containing both open and close stock price.

1. Derive day, month, year part from the date column and add it as separate column in data frame
2. Sort the column as per the target table column list. This target column list is defined in the config.

##### Load the data into the target table **int.stockprice**

1. Generate the dynamic insert query based on the schema, table, and column list.
2. Insert the data using this query

### **Repeat the process for next day:**

* Load the Open price data in the database for next day.
* Using Testing.sql. In our case next day data is 28th Jan 2022.
* Load close price file in the Landing area for next day manually
* Load the file from Landing area to processing area by running below script
  + ***loadfilelocal*.py environment project**
* Load the open price and close price data with all required transformation into target table by running below script.
  + ***etlprocesslocal*.py environment project**

\*\*\* if next data is not available in either of the data set i.e., open price or close price data will not be loaded. Into the target database.

\*\*\* all logging for both file loading and ETL process is maintained in loadfie.log and etlprocess.log respectively under Log folder. Log folder is configurable.

## **Package and Underlying Module Details:**

*(\*\* are for Python Dictionary Arguments, where we pass Dictionary as input parameter)*

### **Configparams**

#### ConfigParams

* + - * + To get parameters from configuration file and return it them as Python Dictionary

Input parameters - Config file, Section

Returns – Python dictionary

### **Database**

#### GetData

* + - * + To get data from the database in the form of Python Data Frame

Input parameters - query, logfile, \*\*dbparams

Returns – Python data frame

#### GetInsertQuery

* + - * + To get data from the database in the form of Python Data Frame

Input parameters - schemaname, tablename, columnlist, logfile

Returns – String

#### PutData

* + - * + To get data from the database in the form of Python Data Frame

Input parameters - query, data frame, logfile, \*\*dbparams

Returns –

### **Localfile**

#### PutFile

* + - * + To move file from one folder to another and archive the file if already present in the target path.

Input parameters - filesourcepath, filetargetpath, filearchivepath, filename, logfile

Returns -