**Build Instructions**

Create the following project directory structure

**Project Directory Structure**

**etl-project-1**

**data**

raw-data.csv

**docs**

notes.txt

**src**

app.py

You should then build your raw data CSV file called **raw-data.csv** first, containing good data, bad data and PII, you can do this using a python program or a text editor of your choice, you should have at least **100 records**, then create a blank file called **app.py** using VsCode as your IDE, that will contain your applications code, you can have this as a single monolithic structure or break it out into modules. The choice of Python Libraries you use is up to you, some examples of code snippets for reference, that you may wish to adapt are available below, but you may wish to create your own.

**Extract -**

**Example Code Snippets** – ***(****Can be adapted to café scenario)*

CSV Functions –

Building a CSV file from a Dictionary

A computer screen shot of a program

Description automatically generated

Building a Dictionary from a CSV File (Extract)

A computer screen shot of a computer program

Description automatically generated

**Transform –**

You now need to parse the data you have extracted into a Dictionary Data Structure transforming the data into it’s final format by removing any PII (Names, Card numbers), removing any blank or malformed records.

You need to do this in stages and there are various methods and operations you can perform on the data to do this, here are some examples you could build and adapt, but do research alternative methods of doing this. Essentially you need to transform the raw data taken from the source CSV file and then save the data to a clean CSV file ready for uploading into a database. Alternatively, you could also just move the data into another temporary data structure in your program ready for the load stage.

There are three type of transaction records you need to consider:

1. Card Transactions

25/08/2021 09:00,Chesterfield,Richard Copeland,"Regular Flavoured iced latte - Hazelnut - 2.75, Large Latte - 2.45",5.2,CARD,5494173772652516

1. Cash Transactions

25/08/2021 09:08,Chesterfield,Michael Sparrow,"Regular Latte - 2.15, Large Latte - 2.45",4.6,CASH,

1. Malformed Transactions

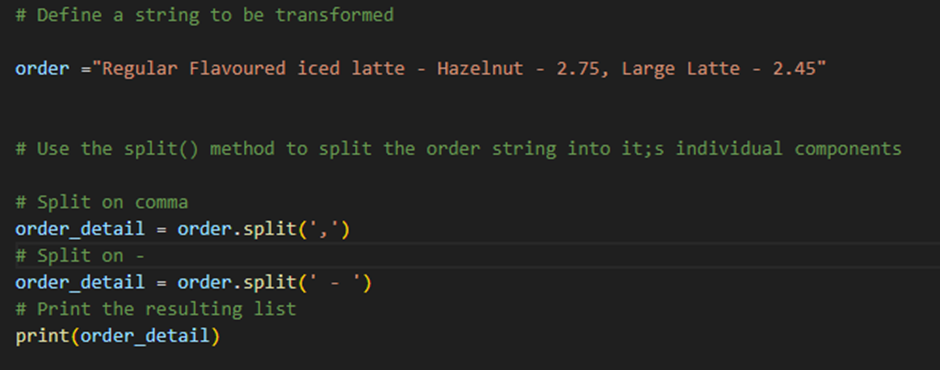
25/08/2021 10:11,Chesterfield,Donald Wilson,Regular Flavoured iced latte - Caramel - 2.75,2.75,CARD,9192463810678210

**Example Code Snippets** – ***(****Can be adapted to café scenario)*

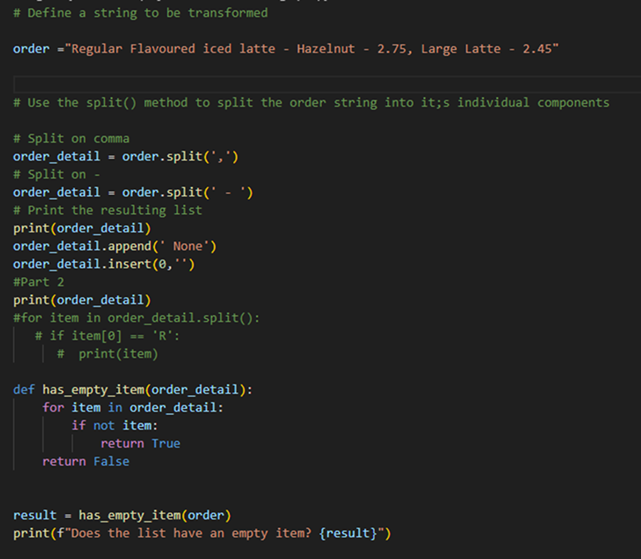
Transformation Functions –

Splitting a string into its individual components (Can be used for card and cash transactions)

String Splitting Example 1



String Splitting Example 2



Parsing data example 1

A screen shot of a computer program

Description automatically generated

Slicing Examples



**Load** –

You can use Docker Containers with MySQL and Adminer images in them to create a database infrastructure locally for your ETL Pipeline.

**Setup Docker Containers Instructions**

*How to Install and setup Docker, MySQL and Adminer*

Download Docker Desktop - <https://docs.docker.com/desktop/install/windows-install/>

Setup a container with a MySQL image in it

Setup a Container with an Adminer Image in It

**Directory Structure**

Add the following Directory Structure to you project folder, this is for you to store all database related files in.

You must have the following files in your db folder :-

.env file, db-loadsql.sql, docker-compose.yml and the Database load program called db\_cafe\_alt\_solution.py

A screenshot of a computer

Description automatically generated

**File Content**

A screen shot of a computer screen

Description automatically generated**.env** *file content*, this sets up the environment variables that contain the credentials to connect to the database.

**docker-compose.yml** file content this provides the Docker Engine with configuration information in order to setup the MySql and Adminer Containers, it also sets up a virtual drive for the database data.

A screenshot of a computer screen

Description automatically generated

**db-loadsql.sql** File content this is used for the Database Build and Setup so contains the SQL Commands needed. This needs to be run in the Adminer SQL Command Window.



A computer screen shot of a program

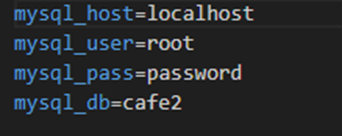
Description automatically generated

Docker Compose YAML file to setup MYsql and adminer containers

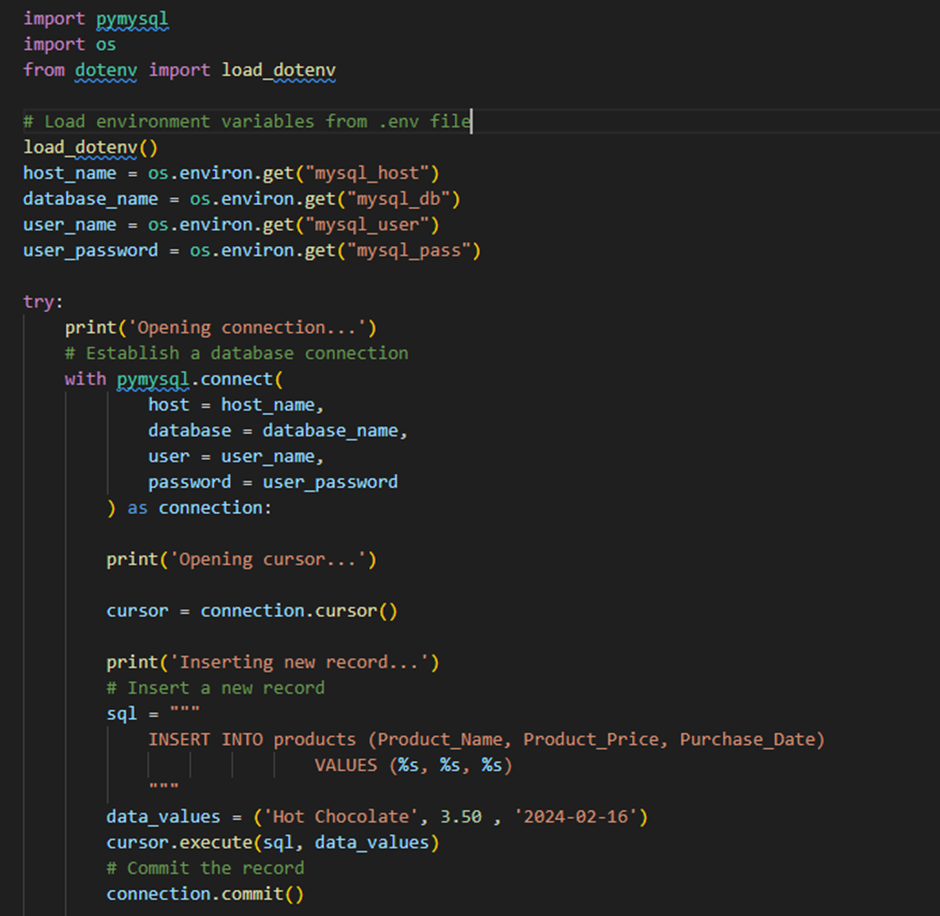
A screenshot of a computer screen

Description automatically generated

Dot env file to setup DB credentials (create a blank file called .env, containing the following)



Load Function Python Program Example Part 1



Load Function Python Program Example Part 2

A screen shot of a computer program

Description automatically generated

**User Interface – (CLI or GUI)**

The task is to create a GUI or CLI interface so the user can trigger the uploading/selection of a Raw Dat CSV file to use.

This can be created in python using a simple menu system or by using the Tkinter library to create a simple GUI.

**CLI - Using Just Python to build a CLI Interface that will trigger Data Processing**

This is an example of how you can use python statements to build a simple menu system that will display the contents of a CSV file, when this is built you can then modify it to process the CSV file further removing anomalies in the data, by combining some of the previous examples on slicing and splitting strings, so it displays cleaned data.

*Part 1 of the programA screen shot of a computer program

Description automatically generated*

Part 2 of the programA screen shot of a computer program

Description automatically generated

**GUI – Using Tkinter with Python to build a GUI Interface that will trigger Data Processing.**

TKinter can be used to create windows, buttons and display text and images.

Below is a set of code snippets you can use to explore the GUI functionality, you could add to your ETL App

To install Tkinter on windows use **pip install tk**

Graphical user interface, text

Description automatically generated**Example 1 Code** (using pack display method)

**Example 1 Output**

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated**Example 2 Code** (using pack display method)

**Example 2 Output**

Graphical user interface, text, application, Word

Description automatically generated

**Example 3 Code** (using grid display method)

Text, letter

Description automatically generated

**Example 3 Output** (using grid display method)

Graphical user interface, text, application

Description automatically generated

**Example 4 Code** (using grid display method)

Text, letter

Description automatically generated

Graphical user interface, text, application

Description automatically generated**Example 4 Output**

**Example 5 Code** (Button using pack)

Text, letter

Description automatically generated

Graphical user interface, application, Teams

Description automatically generated**Example 5 Output**

**Example 6 Code**

Text, letter

Description automatically generated

**Example 6 Output**

Graphical user interface, application, Teams

Description automatically generated

**Example 7 Code –** Button Size

Text, letter

Description automatically generated with medium confidence

Graphical user interface, application

Description automatically generated**Example 7 Output**

**Example 8 Code** – Using a function to make the button do something

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated**Example 8 Output**

Graphical user interface, text, application

Description automatically generated

**Example 9 Code**

**Example 9 Output** – Button Colours

Graphical user interface

Description automatically generated with medium confidence

**Example 10 Code** – Input Boxes

Graphical user interface, text, application, email

Description automatically generated

**Example 10 Output**

Graphical user interface

Description automatically generated

Text

Description automatically generated**Example 11 - Simple Clock**

**Example 11 - Output**

Graphical user interface, application, Word

Description automatically generated

**Additional research Task - Python Timers and clocks**

[**https://youtu.be/ruohUTTo8Kw**](https://youtu.be/ruohUTTo8Kw)

**Tkinter Tutorials**

[**Python Tkinter Tutorial - GeeksforGeeks**](https://www.geeksforgeeks.org/python-tkinter-tutorial/)

**TKinter Cheat Sheet**

[Tkinter Cheat Sheet (activestate.com)](https://cdn.activestate.com/wp-content/uploads/2021/02/Tkinter-CheatSheet.pdf)