ETL Pipeline Local Project 1

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.

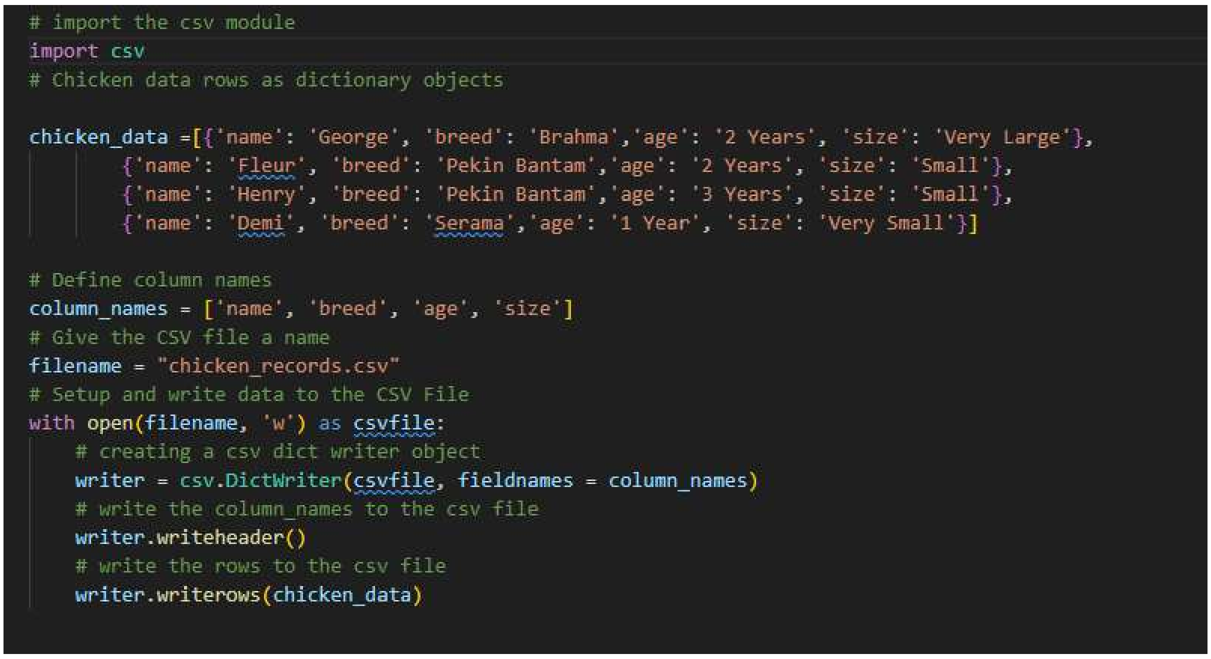
Extract -

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

CSV Functions –

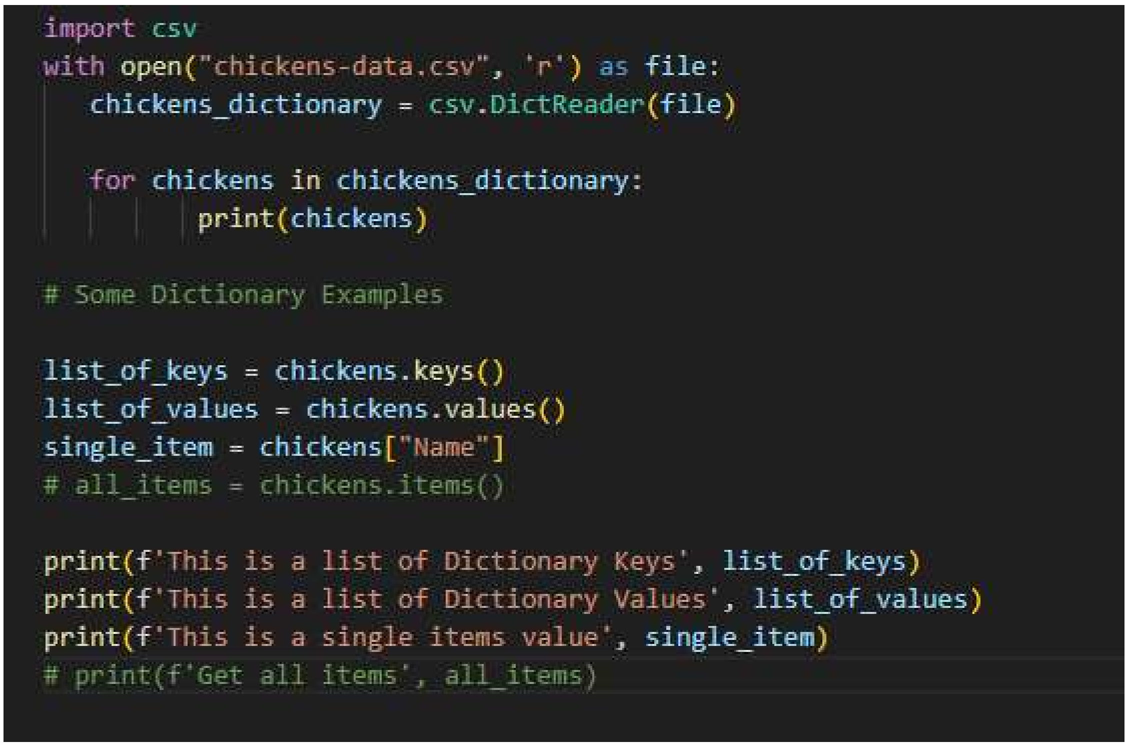
Building a CSV file from a Dictionary

ETL Pipeline Local Project 1



```python
# import the csv module
import csv
# Chicken data rows as dictionary objects

chicken_data =[{'name': 'George', 'breed': 'Brahma','age': '2 Years', 'size': 'Very Large'},
        {'name': 'Fleur', 'breed': 'Pekin Bantam','age': '2 Years', 'size': 'Small'},
        {'name': 'Henry', 'breed': 'Pekin Bantam','age': '3 Years', 'size': 'Small'},
        {'name': 'Demi', 'breed': 'Serama','age': '1 Year', 'size': 'Very Small'}]

# Define column names
column_names = ['name', 'breed', 'age', 'size']
# Give the CSV file a name
filename = "chicken_records.csv"
# Setup and write data to the CSV File
with open(filename, 'w') as csvfile:
    # creating a csv dict writer object
    writer = csv.DictWriter(csvfile, fieldnames = column_names)
    # write the column_names to the csv file
    writer.writeheader()
    # write the rows to the csv file
    writer.writerows(chicken_data)
```

Building a Dictionary from a CSV File (Extract)



```python
import csv
with open("chickens-data.csv", 'r') as file:
    chickens_dictionary = csv.DictReader(file)

    for chickens in chickens_dictionary:
            print(chickens)

# Some Dictionary Examples

list_of_keys = chickens.keys()
list_of_values = chickens.values()
single_item = chickens["Name"]
# all_items = chickens.items()

print(f'This is a list of Dictionary Keys', list_of_keys)
print(f'This is a list of Dictionary Values', list_of_values)
print(f'This is a single items value', single_item)
# print(f'Get all items', all_items)
```

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.

ETL Pipeline Local Project 1

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
```

2. Cash Transactions

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

3. Malformed Transactions

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

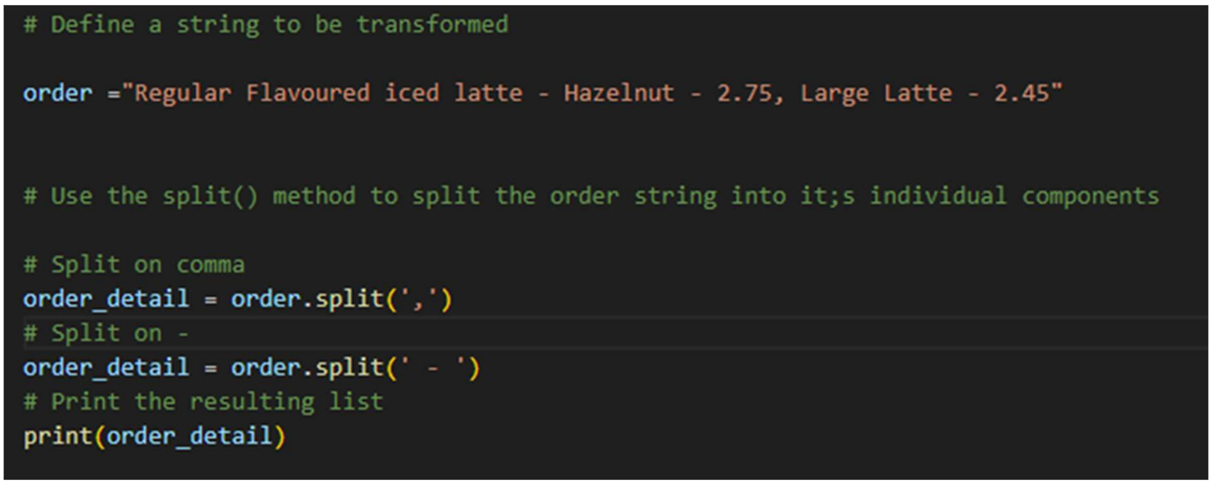
Example Code Snippets – ***(****Can be adapted to café scenario)*

Transformation Functions –

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

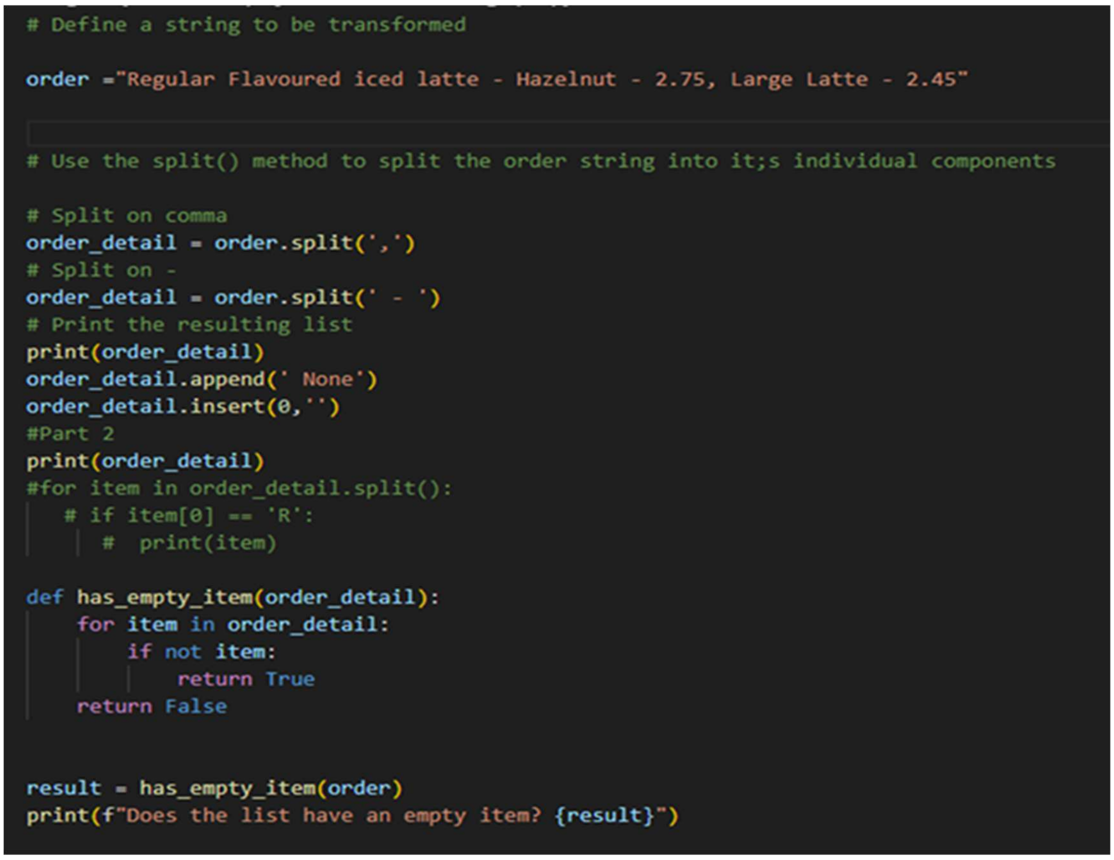
String Splitting Example 1

ETL Pipeline Local Project 1



```python
# Define a string to be transformed

order ="Regular Flavoured iced latte - Hazelnut - 2.75, Large Latte - 2.45"


# Use the split() method to split the order string into it;s individual components

# Split on comma
order_detail = order.split(',')
# Split on -
order_detail = order.split(' - ')
# Print the resulting list
print(order_detail)
```

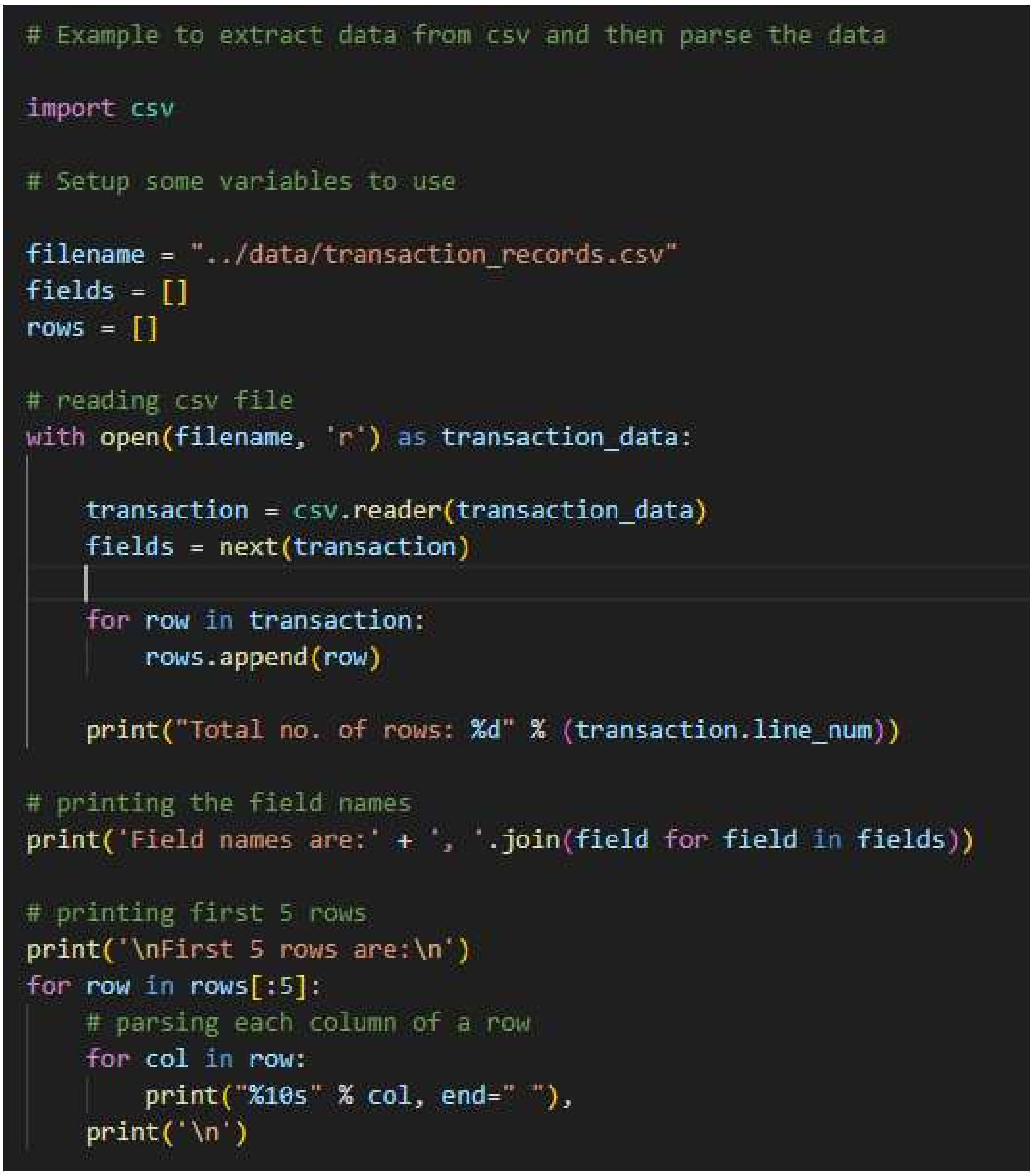
String Splitting Example 2



```python
# Define a string to be transformed

order ="Regular Flavoured iced latte - Hazelnut - 2.75, Large Latte - 2.45"


# Use the split() method to split the order string into it;s individual components

# Split on comma
order_detail = order.split(',')
# Split on -
order_detail = order.split(' - ')
# Print the resulting list
print(order_detail)
order_detail.append(' None')
order_detail.insert(0,'')
#Part 2
print(order_detail)
#for item in order_detail.split():
    # if item[0] == 'R':
        #  print(item)

def has_empty_item(order_detail):
    for item in order_detail:
        if not item:
            return True
    return False


result = has_empty_item(order)
print(f"Does the list have an empty item? {result}")
```

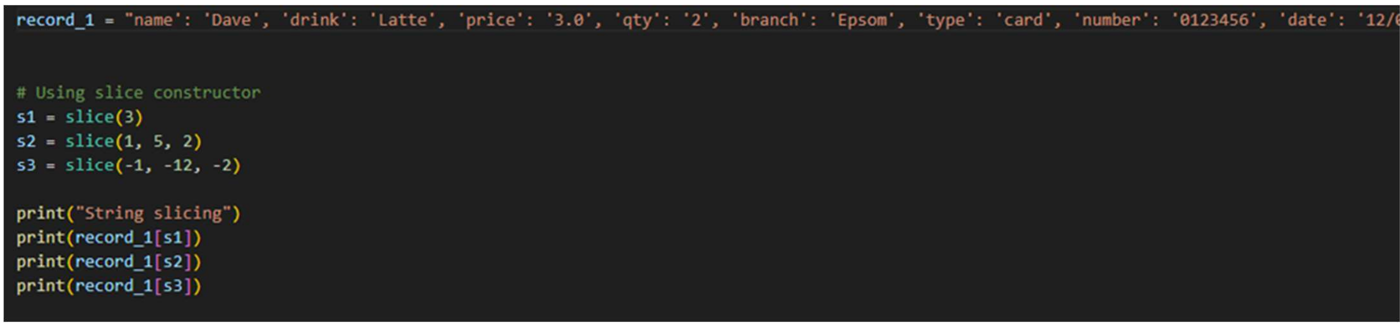
Parsing data example 1

ETL Pipeline Local Project 1



```python
# Example to extract data from csv and then parse the data

import csv

# Setup some variables to use

filename = "../data/transaction_records.csv"
fields = []
rows = []

# reading csv file
with open(filename, 'r') as transaction_data:

    transaction = csv.reader(transaction_data)
    fields = next(transaction)

    for row in transaction:
        rows.append(row)

    print("Total no. of rows: %d" % (transaction.line_num))

# printing the field names
print('Field names are:' + ', '.join(field for field in fields))

# printing first 5 rows
print('\nFirst 5 rows are:\n')
for row in rows[:5]:
    # parsing each column of a row
    for col in row:
        print("%10s" % col, end=" "),
    print('\n')
```

Slicing Examples



```python
record_1 = "name': 'Dave', 'drink': 'Latte', 'price': '3.0', 'qty': '2', 'branch': 'Epsom', 'type': 'card', 'number': '0123456', 'date': '12/


# Using slice constructor
s1 = slice(3)
s2 = slice(1, 5, 2)
s3 = slice(-1, -12, -2)

print("String slicing")
print(record_1[s1])
print(record_1[s2])
print(record_1[s3])
```

ETL Pipeline Local Project 1

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

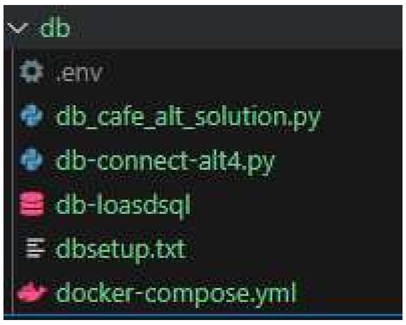
Setup a Container with an Adminer Image in It

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 :-

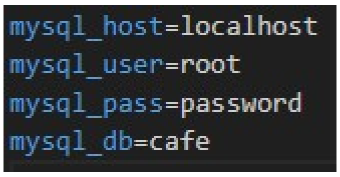
.env file, db-loadsql.sql, docker-compose.yml and the Database load program called db_cafe_alt_solution.py



```
db
  .env
  db_cafe_alt_solution.py
  db-connect-alt4.py
  db-loasdsql
  dbsetup.txt
  docker-compose.yml
```

File Content

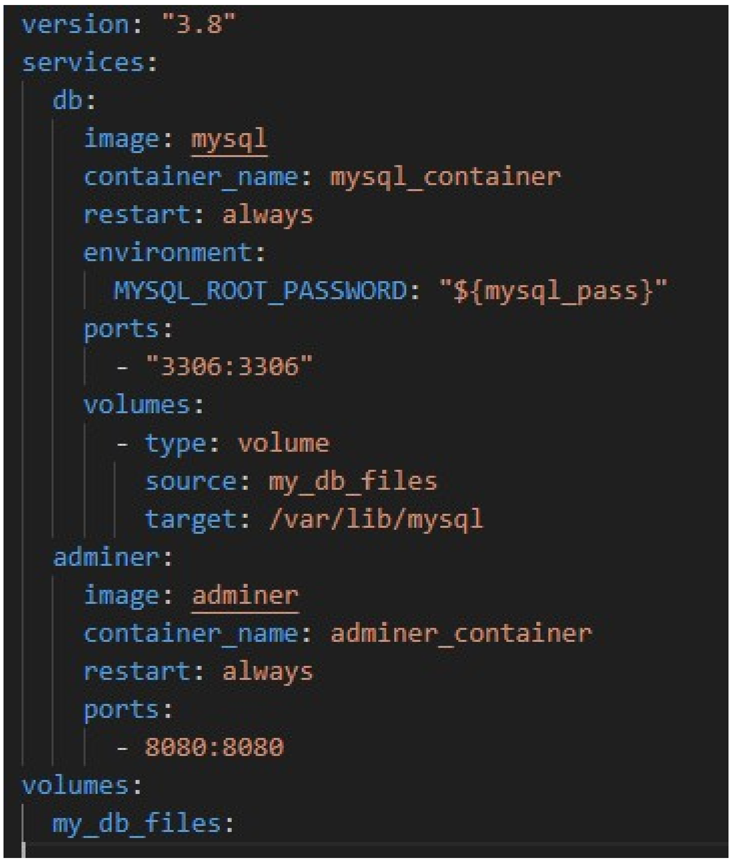
.env *file content*, this sets up the environment variables that contain the credentials to connect to the database.



```
mysql_host=localhost
mysql_user=root
mysql_pass=password
mysql_db=cafe
```

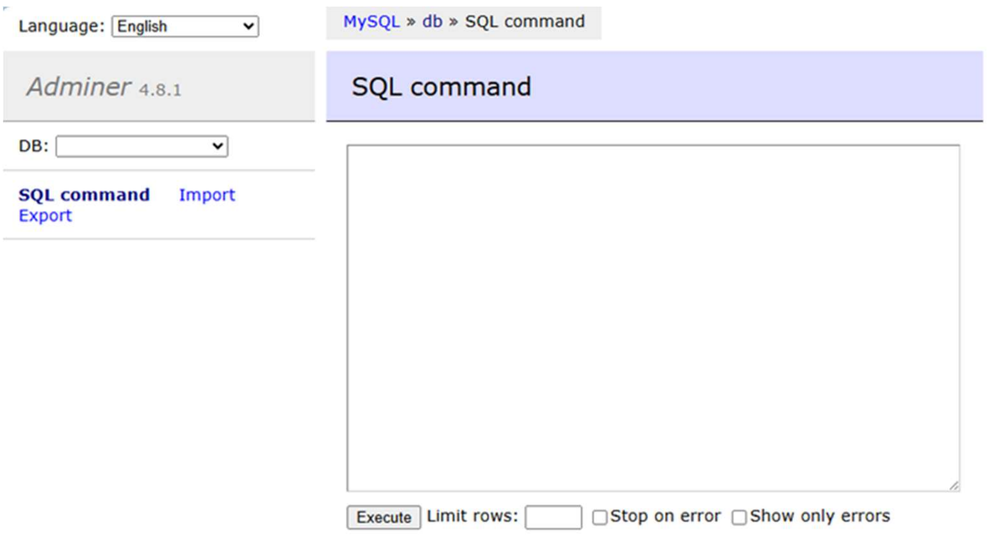
docker-compose.yml file content this provides the Docker Engine with configuration information in order to setup the MySql and Adminer Containers, it also sets up a virtual drive for the database data.

ETL Pipeline Local Project 1

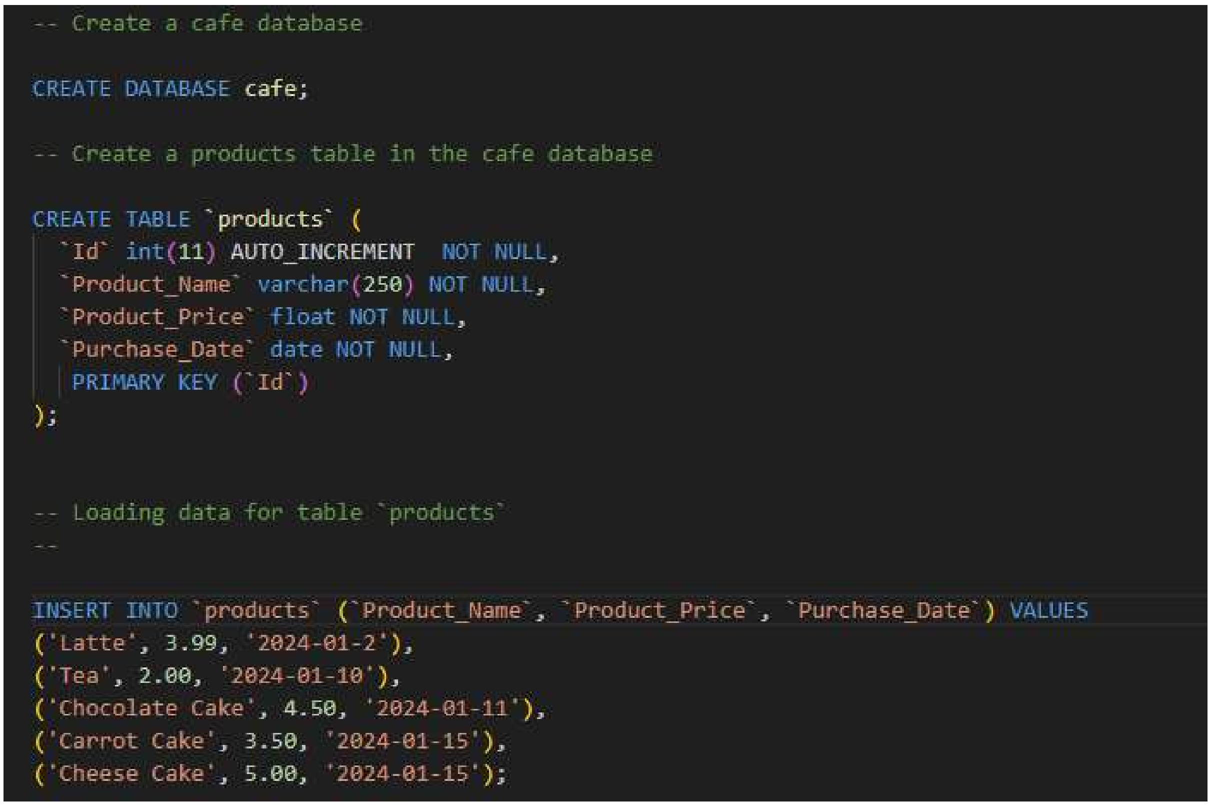


```
version: "3.8"
services:
  db:
    image: mysql
    container_name: mysql_container
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: "${mysql_pass}"
    ports:
      - "3306:3306"
    volumes:
      - type: volume
        source: my_db_files
        target: /var/lib/mysql
  adminer:
    image: adminer
    container_name: adminer_container
    restart: always
    ports:
      - 8080:8080
volumes:
  my_db_files:
```

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.



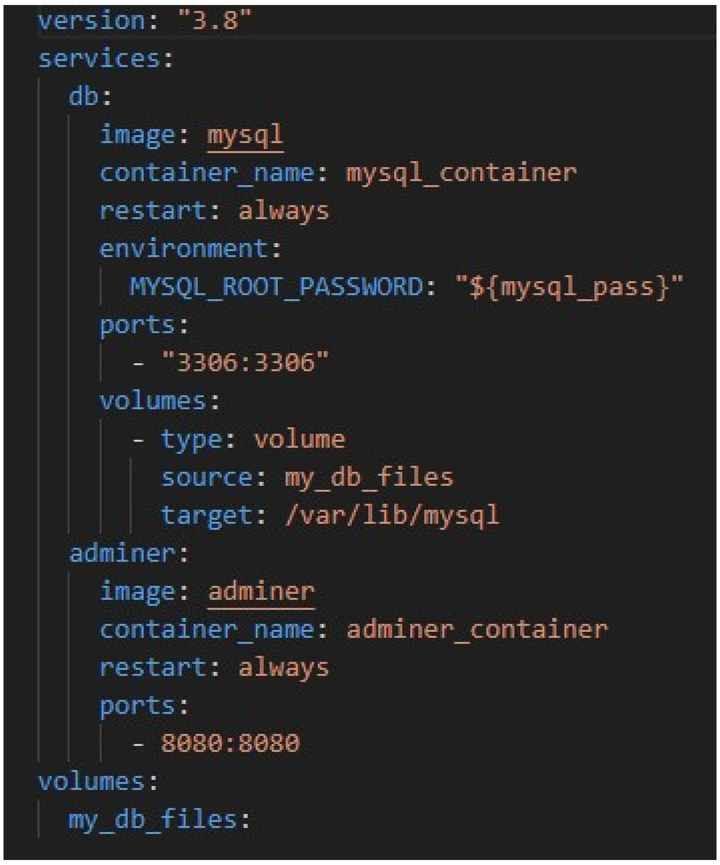
ETL Pipeline Local Project 1



```sql
-- Create a cafe database

CREATE DATABASE cafe;

-- Create a products table in the cafe database

CREATE TABLE `products` (
  `Id` int(11) AUTO_INCREMENT  NOT NULL,
  `Product_Name` varchar(250) NOT NULL,
  `Product_Price` float NOT NULL,
  `Purchase_Date` date NOT NULL,
  PRIMARY KEY (`Id`)
);


-- Loading data for table `products`
--

INSERT INTO `products` (`Product_Name`, `Product_Price`, `Purchase_Date`) VALUES
('Latte', 3.99, '2024-01-2'),
('Tea', 2.00, '2024-01-10'),
('Chocolate Cake', 4.50, '2024-01-11'),
('Carrot Cake', 3.50, '2024-01-15'),
('Cheese Cake', 5.00, '2024-01-15');
```

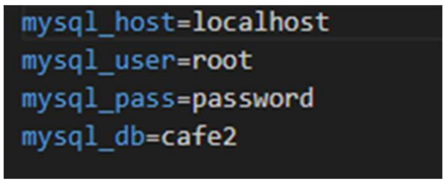
ETL Pipeline Local Project 1

Docker Compose YAML file to setup MYsql and adminer containers



```yaml
version: "3.8"
services:
  db:
    image: mysql
    container_name: mysql_container
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: "${mysql_pass}"
    ports:
      - "3306:3306"
    volumes:
      - type: volume
        source: my_db_files
        target: /var/lib/mysql
  adminer:
    image: adminer
    container_name: adminer_container
    restart: always
    ports:
      - 8080:8080
volumes:
  my_db_files:
```

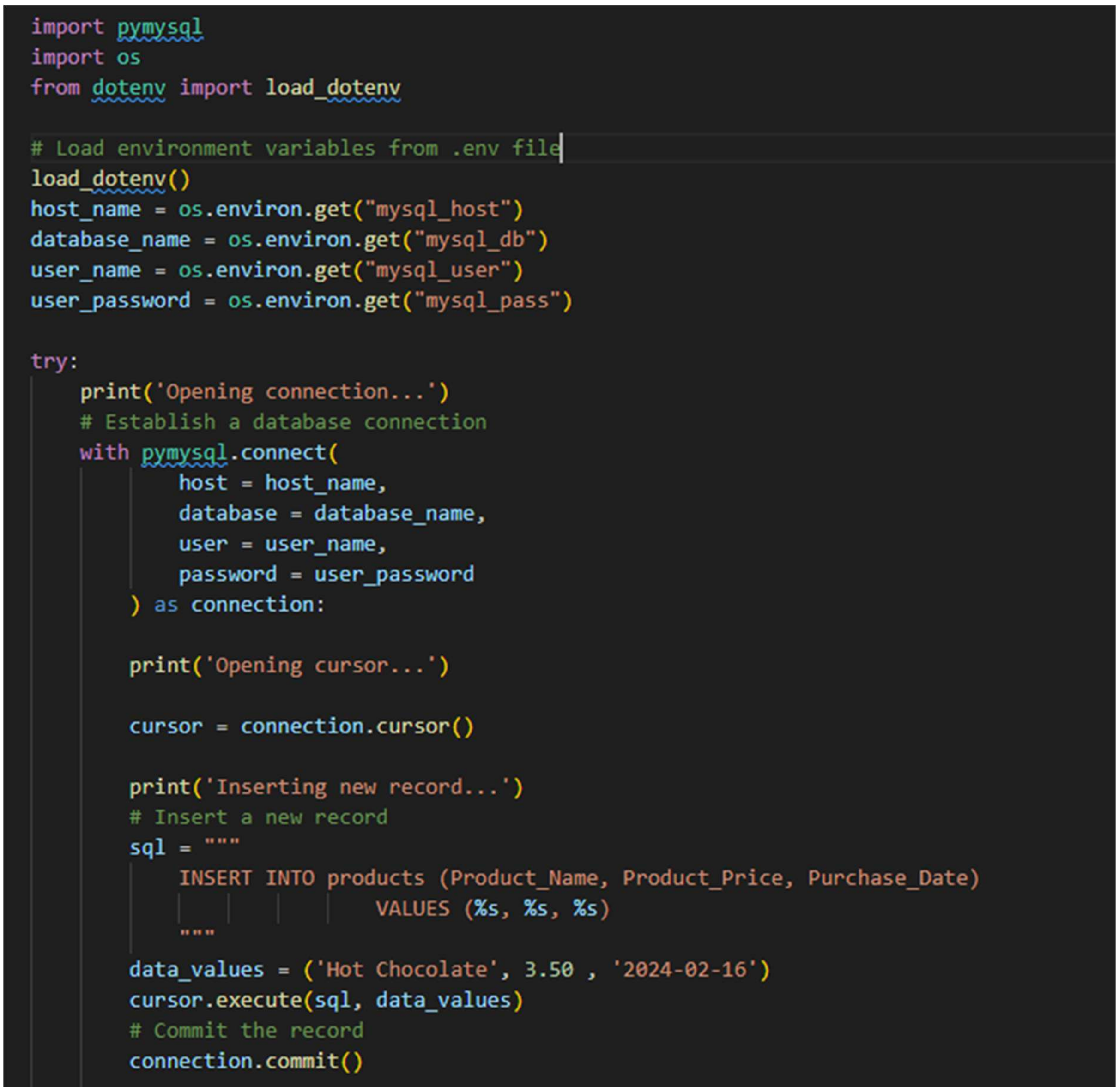
Dot env file to setup DB credentials (create a blank file called .env, containing the following)



```
mysql_host=localhost
mysql_user=root
mysql_pass=password
mysql_db=cafe2
```

ETL Pipeline Local Project 1

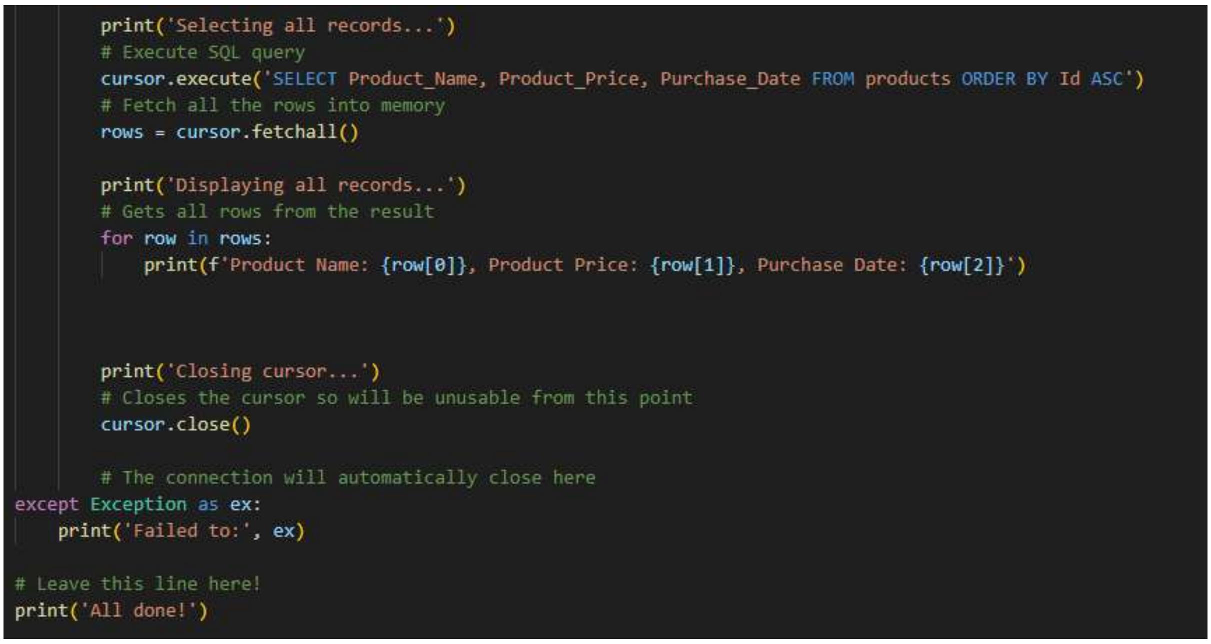
Load Function Python Program Example Part 1



```python
import pymysql
import os
from dotenv import load_dotenv

# Load environment variables from .env file
load_dotenv()
host_name = os.environ.get("mysql_host")
database_name = os.environ.get("mysql_db")
user_name = os.environ.get("mysql_user")
user_password = os.environ.get("mysql_pass")

try:
    print('Opening connection...')
    # Establish a database connection
    with pymysql.connect(
            host = host_name,
            database = database_name,
            user = user_name,
            password = user_password
        ) as connection:

        print('Opening cursor...')

        cursor = connection.cursor()

        print('Inserting new record...')
        # Insert a new record
        sql = """
            INSERT INTO products (Product_Name, Product_Price, Purchase_Date)
                        VALUES (%s, %s, %s)
            """
        data_values = ('Hot Chocolate', 3.50 , '2024-02-16')
        cursor.execute(sql, data_values)
        # Commit the record
        connection.commit()
```

ETL Pipeline Local Project 1

Load Function Python Program Example Part 2



```
        print('Selecting all records...')
        # Execute SQL query
        cursor.execute('SELECT Product_Name, Product_Price, Purchase_Date FROM products ORDER BY Id ASC')
        # Fetch all the rows into memory
        rows = cursor.fetchall()

        print('Displaying all records...')
        # Gets all rows from the result
        for row in rows:
            print(f'Product Name: {row[0]}, Product Price: {row[1]}, Purchase Date: {row[2]}')



        print('Closing cursor...')
        # Closes the cursor so will be unusable from this point
        cursor.close()

        # The connection will automatically close here
except Exception as ex:
    print('Failed to:', ex)

# Leave this line here!
print('All done!')
```

ETL Pipeline Local Project 1

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.

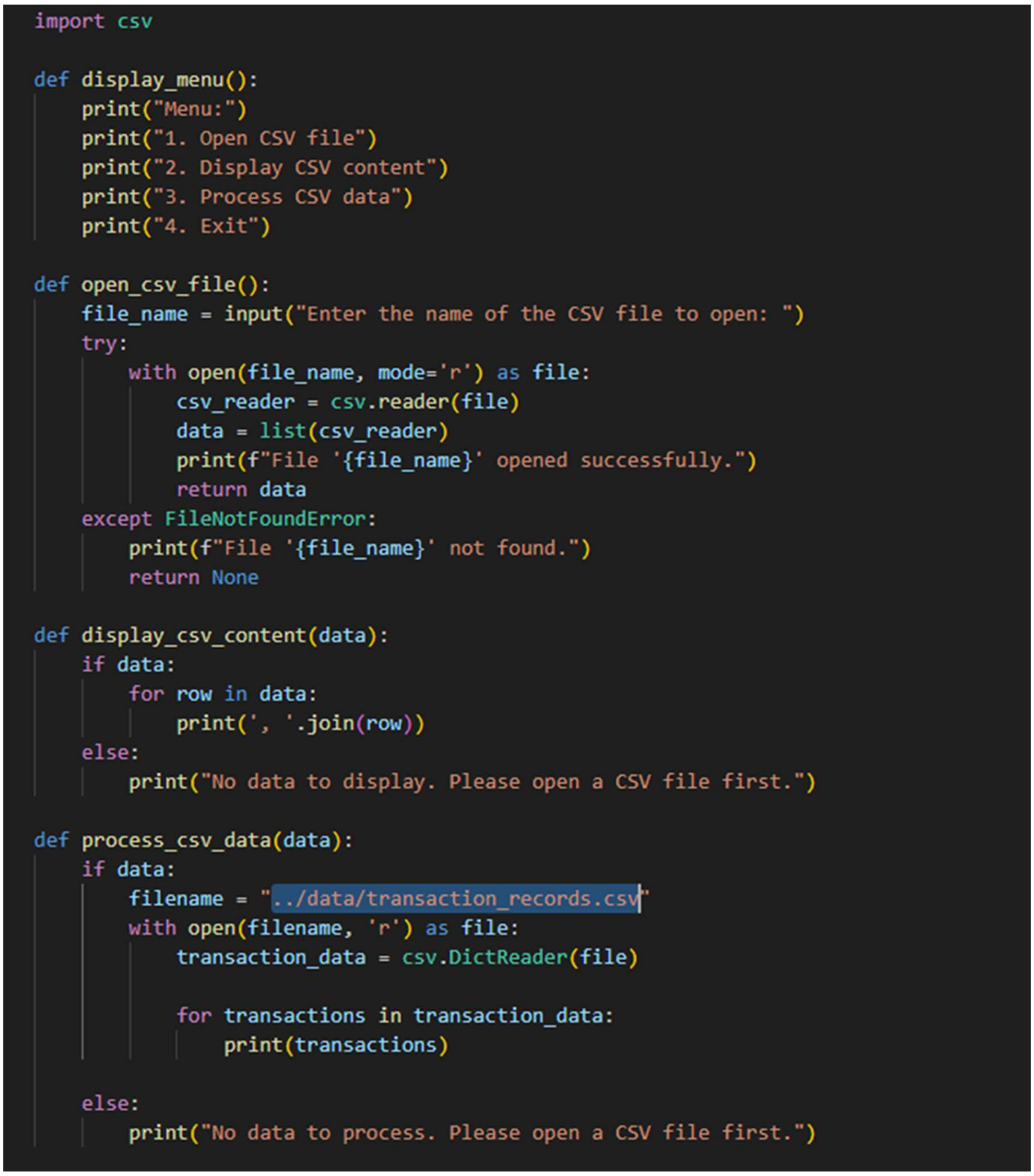
This can be created in python using a simple menu system or by using the Tkinter library to create a simple GUI.

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.

ETL Pipeline Local Project 1

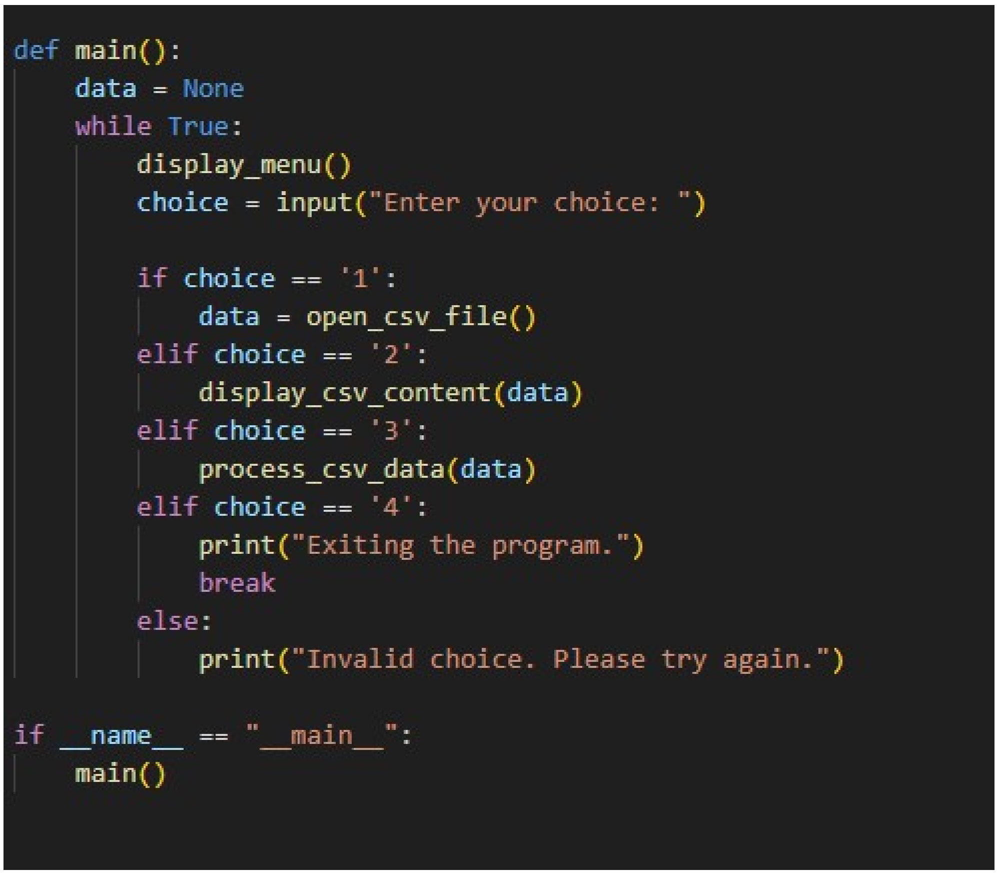
Part 1 of the program



```python
import csv

def display_menu():
    print("Menu:")
    print("1. Open CSV file")
    print("2. Display CSV content")
    print("3. Process CSV data")
    print("4. Exit")

def open_csv_file():
    file_name = input("Enter the name of the CSV file to open: ")
    try:
        with open(file_name, mode='r') as file:
            csv_reader = csv.reader(file)
            data = list(csv_reader)
            print(f"File '{file_name}' opened successfully.")
            return data
    except FileNotFoundError:
        print(f"File '{file_name}' not found.")
        return None

def display_csv_content(data):
    if data:
        for row in data:
            print(', '.join(row))
    else:
        print("No data to display. Please open a CSV file first.")

def process_csv_data(data):
    if data:
        filename = "../data/transaction_records.csv"
        with open(filename, 'r') as file:
            transaction_data = csv.DictReader(file)

            for transactions in transaction_data:
                print(transactions)

    else:
        print("No data to process. Please open a CSV file first.")
```

ETL Pipeline Local Project 1

Part 2 of the program



```python
def main():
    data = None
    while True:
        display_menu()
        choice = input("Enter your choice: ")

        if choice == '1':
            data = open_csv_file()
        elif choice == '2':
            display_csv_content(data)
        elif choice == '3':
            process_csv_data(data)
        elif choice == '4':
            print("Exiting the program.")
            break
        else:
            print("Invalid choice. Please try again.")

if __name__ == "__main__":
    main()
```

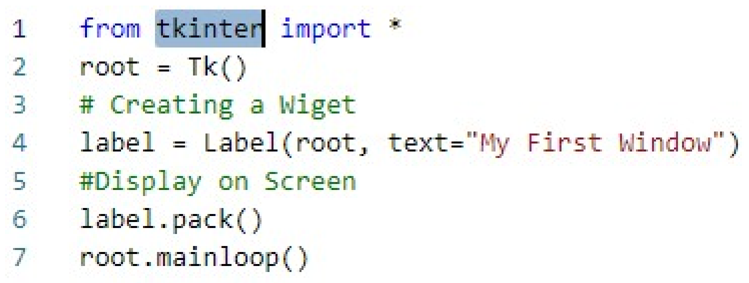
GUI – Using Tkinter with Python to build a GUI Interface that will trigger Data Processing.

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**

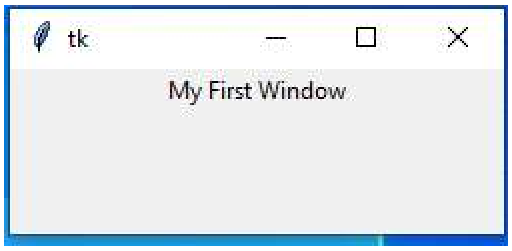
Example 1 Code (using pack display method)



```python
from tkinter import *
root = Tk()
# Creating a Wiget
label = Label(root, text="My First Window")
#Display on Screen
label.pack()
root.mainloop()
```

ETL Pipeline Local Project 1

Example 1 Output

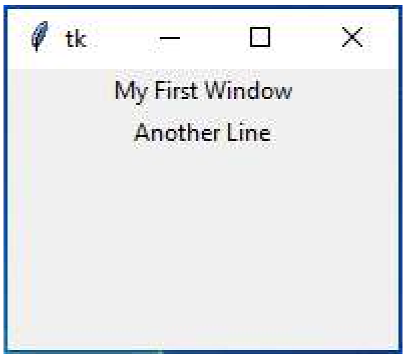


```
My First Window
```

Example 2 Code (using pack display method)

```python
from tkinter import *
root = Tk()
# Creating a Wiget
label1 = Label(root, text="My First Window")
label2 = Label(root, text="Another Line")
#Display on Screen
label1.pack()
label2.pack()
root.mainloop()
```

Example 2 Output



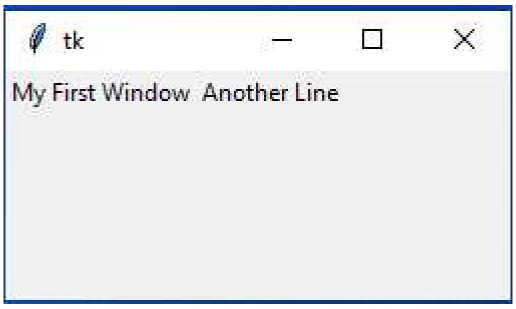
```
My First Window
Another Line
```

Example 3 Code (using grid display method)

ETL Pipeline Local Project 1

```python
from tkinter import *
root = Tk()
# Creating a Wiget
label1 = Label(root, text="My First Window")
label2 = Label(root, text="Another Line")
#Display on Screen
label1.grid(row=0, column=0)
label2.grid(row=0, column=1)
root.mainloop()
```

Example 3 Output (using grid display method)

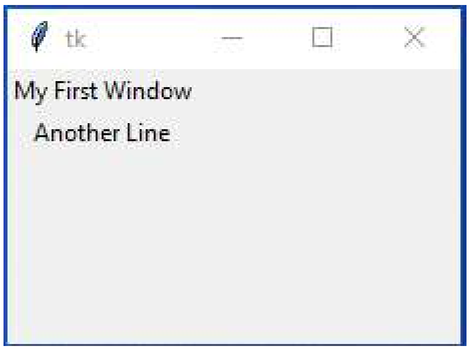


```
My First Window  Another Line
```

Example 4 Code (using grid display method)

```python
from tkinter import *
root = Tk()
# Creating a Wiget
label1 = Label(root, text="My First Window")
label2 = Label(root, text="Another Line")
#Display on Screen
label1.grid(row=0, column=0)
label2.grid(row=1, column=0)
root.mainloop()
```

Example 4 Output



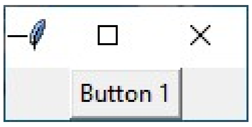
```
My First Window
  Another Line
```

Example 5 Code (Button using pack)

ETL Pipeline Local Project 1

```python
from tkinter import *
root = Tk()
# Creating a Button Wiget
firstButton = Button(root, text="Button 1")
firstButton.pack()
root.mainloop()
```

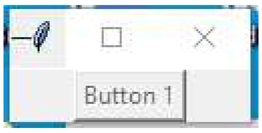
Example 5 Output



Example 6 Code

```python
from tkinter import *
root = Tk()
# Creating a Button Wiget
firstButton = Button(root, text="Button 1", state=DISABLED)
firstButton.pack()
root.mainloop()
```

Example 6 Output

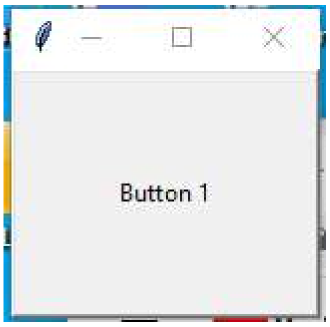


Example 7 Code – Button Size

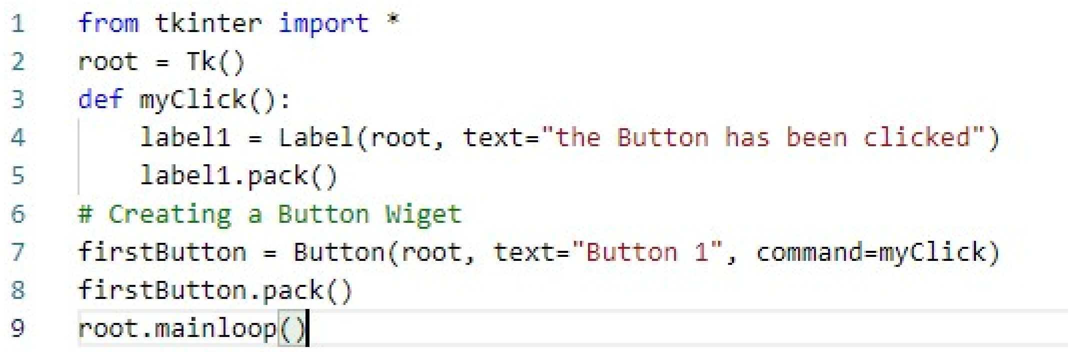
```python
from tkinter import *
root = Tk()
# Creating a Button Wiget
firstButton = Button(root, text="Button 1", padx=50, pady=50)
firstButton.pack()
root.mainloop()
```

ETL Pipeline Local Project 1

Example 7 Output

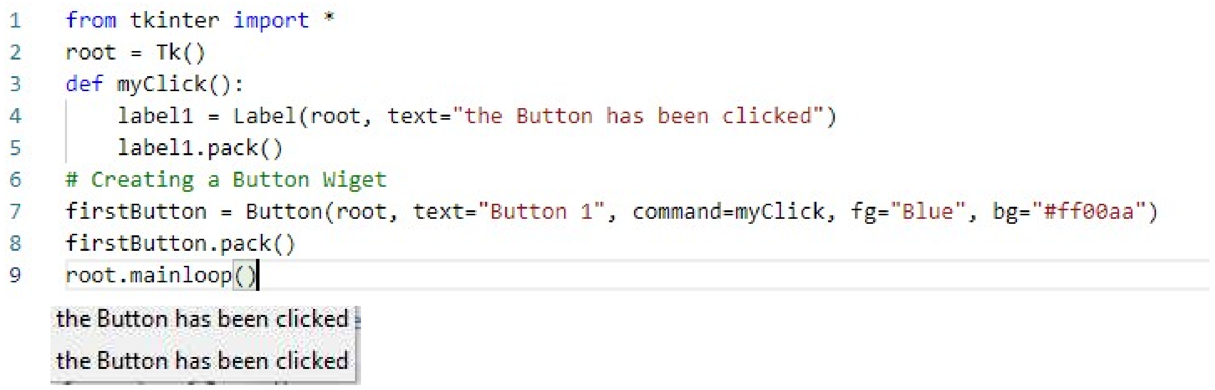


Example 8 Code – Using a function to make the button do something



```python
from tkinter import *
root = Tk()
def myClick():
    label1 = Label(root, text="the Button has been clicked")
    label1.pack()
# Creating a Button Wiget
firstButton = Button(root, text="Button 1", command=myClick)
firstButton.pack()
root.mainloop()
```

Example 8 Output



```python
from tkinter import *
root = Tk()
def myClick():
    label1 = Label(root, text="the Button has been clicked")
    label1.pack()
# Creating a Button Wiget
firstButton = Button(root, text="Button 1", command=myClick, fg="Blue", bg="#ff00aa")
firstButton.pack()
root.mainloop()
```

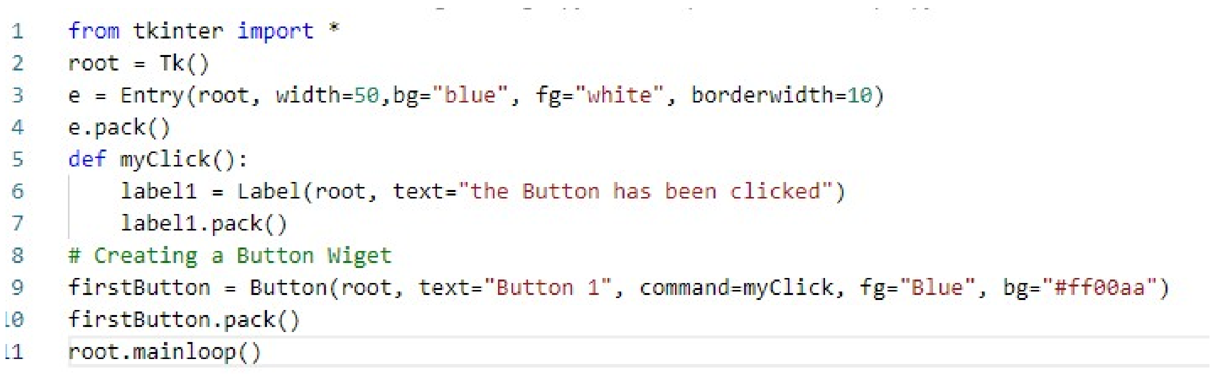
```
the Button has been clicked
the Button has been clicked
```

Example 9 Code

Example 9 Output – Button Colours

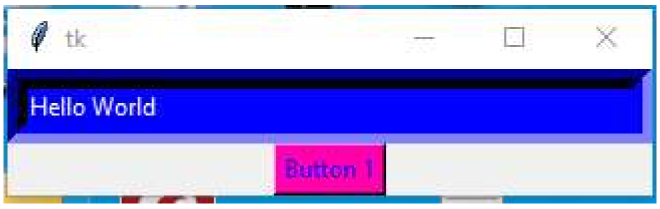
ETL Pipeline Local Project 1

Example 10 Code – Input Boxes



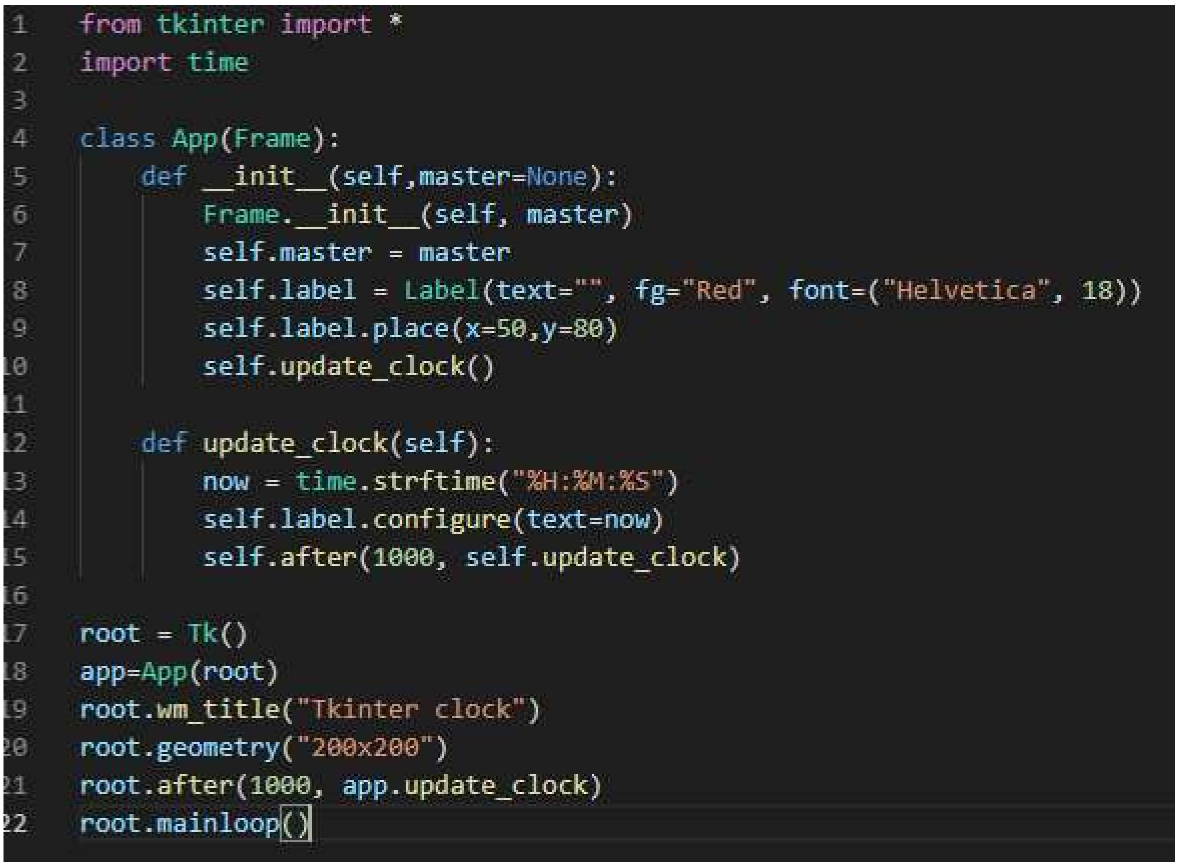
```python
from tkinter import *
root = Tk()
e = Entry(root, width=50,bg="blue", fg="white", borderwidth=10)
e.pack()
def myClick():
    label1 = Label(root, text="the Button has been clicked")
    label1.pack()
# Creating a Button Wiget
firstButton = Button(root, text="Button 1", command=myClick, fg="Blue", bg="#ff00aa")
firstButton.pack()
root.mainloop()
```

Example 10 Output



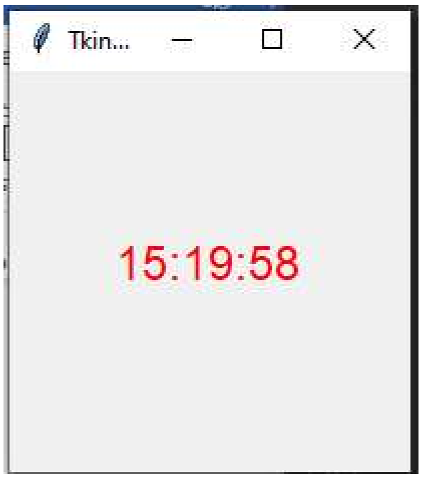
ETL Pipeline Local Project 1

Example 11 - Simple Clock



```
from tkinter import *
import time

class App(Frame):
    def __init__(self,master=None):
        Frame.__init__(self, master)
        self.master = master
        self.label = Label(text="", fg="Red", font=("Helvetica", 18))
        self.label.place(x=50,y=80)
        self.update_clock()

    def update_clock(self):
        now = time.strftime("%H:%M:%S")
        self.label.configure(text=now)
        self.after(1000, self.update_clock)

root = Tk()
app=App(root)
root.wm_title("Tkinter clock")
root.geometry("200x200")
root.after(1000, app.update_clock)
root.mainloop()
```

Example 11 - Output



```
15:19:58
```

ETL Pipeline Local Project 1

Additional research Task - Python Timers and clocks

https://youtu.be/ruohUTTo8Kw

Tkinter Tutorials

Python Tkinter Tutorial - GeeksforGeeks

TKinter Cheat Sheet

Tkinter Cheat Sheet (activestate.com)