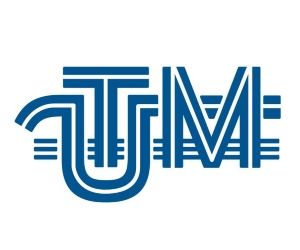
Ministerul Educației și Cercetării al Republicii Moldova

Universitatea Tehnică a Moldovei

Facultatea Calculatoare, Informatică și Microelectronică

Departamentul Ingineria Software și Automatică

**Raport**

la disciplina **”Programarea Declarativa”**

Tema: **Manipularea datelor folosind limbajul Python**

Efectuat de: studentul/studenta gr.**TI-216 Vlasitchi Stefan**

Verificat de: asis.univ **Viorel Rusu**

Chișinău-2023

**Exercitiul 1.**

Scrieți secvența de cod care regrupeză datele din cele 3 fișiere csv (obținute la laboratul precedent) după o caracteristică de bază (de ex: cursul valutar de la 3 bănci pentru aceeași zi)

from pathlib import Path  
import pandas as pd  
  
  
def filter\_and\_save(input\_file, output\_file=None, start\_date='2023-10-24', end\_date='2023-10-31'):  
 # Load data from the CSV file  
 df = pd.read\_csv(input\_file)  
  
 # Convert the 'Date' column to the datetime data type  
 df['Date'] = pd.to\_datetime(df['Date'])  
  
 # Select only the data between the specified date range  
 filtered\_df = df[df['Date'].between(start\_date, end\_date)]  
  
 # Extract the file name without extension using pathlib  
 input\_path = Path(input\_file)  
 file\_name = input\_path.stem  
 file\_extension = input\_path.suffix  
  
 # Construct the name of the new output file  
 new\_output\_file = Path(output\_file) if output\_file else input\_path.parent / f"new\_{file\_name}{file\_extension}"  
  
 # Save the sorted and filtered DataFrame to a new CSV file  
 filtered\_df.to\_csv(new\_output\_file, index=False)  
  
 print(  
 f"Data from {input\_file} has been sorted and filtered for the period {start\_date} - {end\_date} and saved in {new\_output\_file}.")  
  
  
# List of CSV file names  
files = ['1\_exchange\_rates\_october.csv', '2\_exchange\_rates\_october.csv', '3\_exchange\_rates\_october.csv']  
  
# Iterate through each file and apply the sorting function  
for file in files:  
 filter\_and\_save(file)

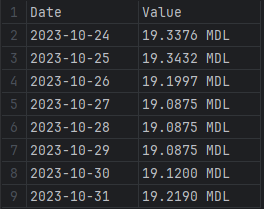
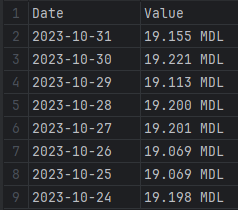
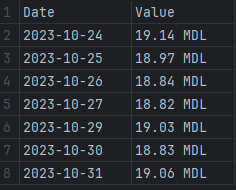
****Figura 1. Codul exercitiului 1

Figura 2 . Rezultatul exercitiului 1

**Exercitiul 2.**

Determinați min, max, media pentru valorile regrupate pentru fiecare zi aparte și min, max media pentru toate datele stocate.

import pandas as pd  
import os  
  
  
def load\_and\_process\_data(file\_path):  
 df = pd.read\_csv(file\_path)  
 df['Date'] = pd.to\_datetime(df['Date'])  
 df['Value'] = pd.to\_numeric(df['Value'].astype(str).str.replace(' MDL', ''), errors='coerce')  
 df = df.dropna(subset=['Value'])  
 return df  
  
  
def calculate\_daily\_statistics(df):  
 return df.groupby('Date')['Value'].agg(['min', 'max', 'mean'])  
  
  
def display\_statistics\_for\_date(date, stats\_dict):  
 print(f"\nComparare pentru data: {date} 00:00:00")  
 for stat\_type in ['min', 'max', 'mean']:  
 values = [stats\_dict[file\_key].loc[date, stat\_type] for file\_key in stats\_dict]  
 result = min(values) if stat\_type == 'min' else max(values) if stat\_type == 'max' else sum(values) / len(values)  
 print(f"{stat\_type}: {result}")  
  
  
def calculate\_and\_display\_statistics(file\_names, file\_type):  
 daily\_statistics\_dict = {}  
  
 for file\_name in file\_names:  
 df = load\_and\_process\_data(file\_name)  
 daily\_statistics = calculate\_daily\_statistics(df)  
 file\_key = os.path.splitext(os.path.basename(file\_name))[0]  
 daily\_statistics\_dict[file\_key] = daily\_statistics  
  
 all\_dates = set().union(\*[daily\_statistics\_dict[file\_key].index for file\_key in daily\_statistics\_dict])  
  
 for date\_to\_check in sorted(all\_dates):  
 display\_statistics\_for\_date(date\_to\_check, daily\_statistics\_dict)  
  
  
def main():  
 new\_files = [f for f in os.listdir() if f.startswith("new")]  
 specified\_files = ['1\_exchange\_rates\_october.csv', '2\_exchange\_rates\_october.csv', '3\_exchange\_rates\_october.csv']  
  
 print("\nApelul funcției pentru calculul statisticilor pentru fișierele create anterior cu prefixul 'new'")  
 calculate\_and\_display\_statistics(new\_files, "new")  
  
 print("\n----------------------------------------------------------------------------")  
  
 print("\nApelul funcției pentru calculul statisticilor pentru fișierele specificate")  
 calculate\_and\_display\_statistics(specified\_files, "specified")  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

Figura 3 . Codul exercitiului 2

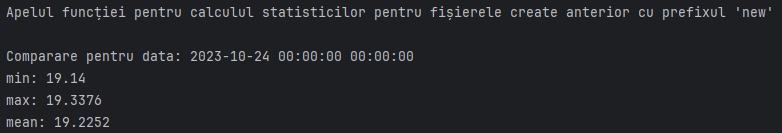
****

Figura 4.1. Rezultatul exercitiului 2

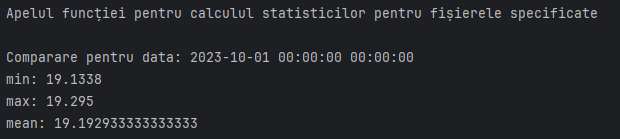
****

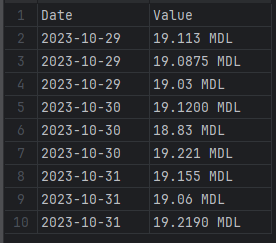
Figura 4.2. Rezultatul exercitiului 2

**Exercitiul 3.**

Scrieți secvența de cod care va împărți fișierul csv obținut după concatinare în N fișiere, unde fiecare fișier individual va corespunde unei săptămâini (7 rânduri). Fișierele sunt denumite după prima și ultima dată pe care le conțin (de ex 20231103\_20231109.csv).

import pandas as pd  
  
END\_DATE = pd.to\_datetime('2023-10-31')  
  
files\_to\_concat = ['1\_exchange\_rates\_october.csv', '2\_exchange\_rates\_october.csv', '3\_exchange\_rates\_october.csv']  
  
# List Comprehension for loading data  
dataframes = [pd.read\_csv(file) for file in files\_to\_concat]  
  
concatenated\_df = pd.concat(dataframes, ignore\_index=True)  
concatenated\_df.to\_csv('fisier\_concatenat.csv', index=False)  
print("Fișierul concatenat a fost creat cu succes.")  
  
# Date parsing with error handling  
concatenated\_df['Date'] = pd.to\_datetime(concatenated\_df['Date'], format='%d %B %Y', errors='coerce')  
  
if concatenated\_df['Date'].isna().any():  
 print("Warning: Some dates could not be parsed.")  
  
# Sorting by date  
concatenated\_df = concatenated\_df.sort\_values(by='Date')  
  
# Using a fixed interval for weeks  
distinct\_weeks = pd.date\_range(start=concatenated\_df['Date'].min(), end=concatenated\_df['Date'].max(), freq='7D')  
  
for start\_date in distinct\_weeks:  
 end\_date = start\_date + pd.DateOffset(days=6)  
 end\_date = min(end\_date, END\_DATE) # Adjust end date to END\_DATE constant  
  
 week\_df = concatenated\_df[(concatenated\_df['Date'] >= start\_date) & (concatenated\_df['Date'] <= end\_date)]  
  
 if not week\_df.empty:  
 weekly\_output\_file = f"{start\_date.strftime('%Y%m%d')}\_{end\_date.strftime('%Y%m%d')}.csv"  
 week\_df.to\_csv(weekly\_output\_file, index=False)  
 print(f"Created file: {weekly\_output\_file}")  
 else:  
 print("Could not create file: no valid data.")

Figura 5. Codul exercitiului 3

****

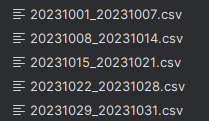
****

Figura 6. Rezultatul exercitiului 3

**Exercitiul 4.**

Scrieți funcția ce are o dată de intrare tip datetime și returnează liniile din DataFrame (din fișierul cu date concatinate) pentru această dată sau None dacă nu există date pentru această dată.

import pandas as pd  
  
# Încarcă fișierul CSV obținut după concatinare  
try:  
 concatenated\_df = pd.read\_csv("fisier\_concatenat.csv")  
except FileNotFoundError:  
 print("Fișierul fisier\_concatenat.csv nu a fost găsit.")  
 concatenated\_df = None  
  
  
def filter\_data\_by\_date(df, target\_date):  
 if df is not None and 'Date' in df.columns:  
 try:  
 formatted\_target\_date = target\_date.strftime('%d %B %Y')  
 filtered\_rows = df[df['Date'] == formatted\_target\_date]  
  
 if not filtered\_rows.empty:  
 return filtered\_rows  
 else:  
 print(f'Nu există date pentru data: {target\_date}')  
 return None  
 except ValueError as e:  
 print(f"Eroare la filtrarea datelor: {e}")  
 return None  
 else:  
 print("DataFrame invalid pentru filtrare.")  
 return None  
  
  
# Exemplu de utilizare:  
try:  
 day\_input = input("Introdu ziua (DD): ")  
 target\_date = pd.to\_datetime(f'2023-10-{day\_input}', format='%Y-%m-%d')  
except ValueError as e:  
 print(f"Eroare: {e}")  
 target\_date = None  
  
if target\_date is not None:  
 result = filter\_data\_by\_date(concatenated\_df, target\_date)  
  
 if result is not None:  
 print(f"\nDatele pentru data {target\_date} sunt:")  
 print(result)

Figura 7. Codul exercitiului 4

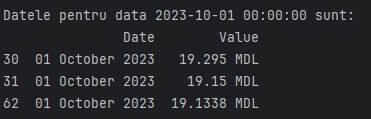
****

Figura 8. Rezultatul exercitiului 4

**Exercitiul 5.**

Scrieți secvența de cod care permite afișarea datelor pe secvențe de timp. De ex. pentru fiecare săptămănă separat (funcția are o dată de intrare tip datetime) și returnează liniile din DataFrame pentru această perioadă sau None dacă nu există date pentru zilele indicate.

import pandas as pd  
  
try:  
 # Încarcă fișierul CSV obținut după concatinare  
 concatenated\_df = pd.read\_csv("fisier\_concatenat.csv")  
except FileNotFoundError:  
 print("Fișierul fisier\_concatenat.csv nu a fost găsit.")  
 concatenated\_df = None  
  
def get\_data\_for\_time\_interval(dataframe, start\_date, end\_date):  
 try:  
 # Asigură-te că coloana 'Date' este de tipul datetime  
 dataframe['Date'] = pd.to\_datetime(dataframe['Date'], errors='coerce')  
  
 # Construiește datele de start și final cu luna și anul fixate la 2023-10  
 start\_date = pd.to\_datetime(f'2023-10-{start\_date:02}', format='%Y-%m-%d')  
 end\_date = pd.to\_datetime(f'2023-10-{end\_date:02}', format='%Y-%m-%d')  
  
 # Filtrarea dataframe-ului pentru intervalul specificat  
 time\_interval\_data = dataframe[(dataframe['Date'] >= start\_date) & (dataframe['Date'] <= end\_date)]  
  
 # Verificare dacă există date pentru intervalul specificat  
 if time\_interval\_data.empty:  
 print(f"Nu există date pentru intervalul de la {start\_date} la {end\_date}")  
 return None  
 else:  
 return time\_interval\_data  
  
 except ValueError as e:  
 print(f"Eroare la conversia datelor sau filtrarea datelor: {e}")  
 return None  
  
# Exemplu de utilizare  
try:  
 start\_date\_input = int(input("Introdu data inițială (format: DD): "))  
 end\_date\_input = int(input("Introdu data finală (format: DD): "))  
  
 # Verifică dacă datele introduse sunt în formatul corect și sunt valide  
 if 1 <= start\_date\_input <= 31 and 1 <= end\_date\_input <= 31:  
 result = get\_data\_for\_time\_interval(concatenated\_df, start\_date\_input, end\_date\_input)  
  
 if result is not None:  
 print(result)  
 else:  
 print("Ați introdus o dată greșită.")  
  
except ValueError:  
 print("Datele introduse nu sunt valide.")

Figura 9. Codul exercitiului 5

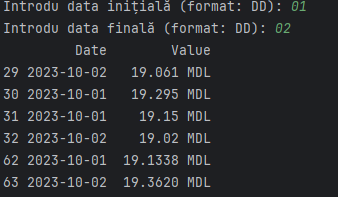


Figura 9. Rezultaul exercitiului 5

**Concluzie**

În cadrul laboratorului, s-au abordat diverse aspecte legate de manipularea datelor folosind limbajul Python. Am început prin concatenarea mai multor fișiere CSV într-un singur cadru de date, apoi am explorat diverse moduri de analiză și filtrare a acestora.

Pentru a face manipularea datelor mai interactivă și personalizată, am implementat funcții care permit utilizatorului să introducă date inițiale și finale pentru a obține intervalul dorit de informații din cadru. În aceste funcții, am folosit librăria Pandas pentru manipularea datelor și am integrat aspecte practice precum validarea intrărilor utilizatorului și gestionarea corectă a formatului datei.

În plus, am explorat moduri de sortare și exportare a datelor, creând fișiere CSV separate pentru săptămâni și afișând rezultatele într-un mod coerent și ușor de înțeles.