**національний університет БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ україни**

**ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

Кафедра комп’ютерних наук

**ЛАБОРАТОРНА РОБОТА №4**

Виконав:

студент групи ІПЗ-21007б

Щур В. М.

Київ – 2024

**Хід виконання**

***1.*** *Створити віртуальне оточення (ім'я оточення - прізвище студента). В цьому оточенні створити проект Python – “Employees”*

*Зображення, що містить текст, знімок екрана, програмне забезпечення, монітор

Автоматично згенерований опис*

***2.*** *В проекті створити три програми. Перша зберігає в файл CSV таблицю наступної структури:*

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Прізвище | Ім’я | По батькові | Стать | Дата народження | Посада | Місто проживання | Адреса проживання | Телефон | Email |
|  |  |  |  |  |  |  |  |  |  |

І додає в неї не менш ніж 2000 записів. При чому 40% жіночої і 60% чоловічої статі. Всі назви і імена повинні бути коректними. Дата народження від 1938 до 2008 року.

**Код:**

import pandas as pd  
import random  
from faker import Faker  
from datetime import datetime  
  
faker\_ua = Faker('uk\_UA')  
  
def create\_birthdate():  
 start\_date = datetime(1938, 1, 1)  
 end\_date = datetime(2008, 12, 31)  
 return faker\_ua.date\_between(start\_date=start\_date, end\_date=end\_date)  
  
def assign\_gender():  
 return random.choices(["Чоловіча", "Жіноча"], [0.6, 0.4])[0]  
  
def assign\_job():  
 return faker\_ua.job()  
  
def create\_personal\_record(gender):  
 if gender == "Чоловіча":  
 first\_name = faker\_ua.first\_name\_male()  
 last\_name = faker\_ua.last\_name\_male()  
 middle\_name = faker\_ua.middle\_name\_male()  
 else:  
 first\_name = faker\_ua.first\_name\_female()  
 last\_name = faker\_ua.last\_name\_female()  
 middle\_name = faker\_ua.middle\_name\_female()  
  
 return {  
 "Прізвище": last\_name,  
 "Ім’я": first\_name,  
 "По батькові": middle\_name,  
 "Стать": gender,  
 "Дата народження": create\_birthdate(),  
 "Посада": assign\_job(),  
 "Місто проживання": faker\_ua.city(),  
 "Адреса проживання": faker\_ua.address().replace("\n", ", "),  
 "Телефон": faker\_ua.phone\_number(),  
 "Email": faker\_ua.email()  
 }  
  
def export\_to\_csv(filename, total\_records):  
 data\_records = []  
  
 male\_records = int(total\_records \* 0.6)  
 female\_records = total\_records - male\_records  
  
 for \_ in range(male\_records):  
 data\_records.append(create\_personal\_record("Чоловіча"))  
  
 for \_ in range(female\_records):  
 data\_records.append(create\_personal\_record("Жіноча"))  
  
 df = pd.DataFrame(data\_records)  
 df.to\_csv(filename, index=False)  
  
export\_to\_csv("employees.csv", 2000)

**Результат:**

**Зображення, що містить текст, знімок екрана

Автоматично згенерований опис**

***3.*** *Друга програма створює XLSX. В ньому 5 аркушів – “all”, “younger\_18”, “18-45”, “45-70”, “older\_70”. В аркуш “all” записує всі данні із CSV файлу, який створено в другому пункті. На інших аркушах відображає данні про співробітників, які належать до відповідної вікової категорії наступної структури:*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Прізвище | Ім’я | По батькові | Дата народження | Вік |
|  |  |  |  |  |  |

Вік – кількість повних років на момент поточної дати. Програма повинна виводити в консоль наступні повідомлення:

* Ok, якщо програма завершила свою роботу успішно;
* Повідомлення про неможливість створення XLSX файлу;
* Повідомлення про відсутність, або проблеми при відкритті файлу CSV.

**Код:**

import pandas as pd  
from datetime import datetime  
import os  
  
# Функція для обчислення віку  
def get\_age(birthdate):  
 today = datetime.today()  
 age = today.year - birthdate.year - ((today.month, today.day) < (birthdate.month, birthdate.day))  
 return age  
  
# Функція для класифікації віку  
def age\_group(age):  
 if age < 18:  
 return 'до\_18'  
 elif 18 <= age <= 45:  
 return '18-45'  
 elif 46 <= age <= 70:  
 return '45-70'  
 else:  
 return 'старше\_70'  
  
def convert\_csv\_to\_xlsx(csv\_file, xlsx\_file):  
 try:  
 df = pd.read\_csv(csv\_file)  
  
 required\_columns = ["Прізвище", "Ім’я", "По батькові", "Дата народження"]  
 if not all(column in df.columns for column in required\_columns):  
 raise ValueError("CSV файл не містить усіх необхідних колонок!")  
  
 df['Дата народження'] = pd.to\_datetime(df['Дата народження'], errors='coerce')  
 df['Вік'] = df['Дата народження'].apply(lambda x: get\_age(x) if pd.notnull(x) else None)  
  
 age\_categories = ['до\_18', '18-45', '45-70', 'старше\_70']  
 categorized\_dataframes = {category: df[df['Вік'].apply(lambda x: age\_group(x) == category)] for category in age\_categories}  
  
 with pd.ExcelWriter(xlsx\_file, engine='openpyxl') as writer:  
 df.to\_excel(writer, sheet\_name='Усі записи', index=False)  
  
 for category in age\_categories:  
 age\_df = categorized\_dataframes[category]  
 age\_df = age\_df[['Прізвище', 'Ім’я', 'По батькові', 'Дата народження', 'Вік']]  
 age\_df.index += 1  
 age\_df.index.name = '№'  
 age\_df.to\_excel(writer, sheet\_name=category, index=True)  
  
 print("Файл XLSX успішно створено.")  
  
 except FileNotFoundError:  
 print("Помилка - CSV файл не знайдено або виникла проблема при його відкритті!")  
 except Exception as e:  
 print(f"Помилка при створенні XLSX файлу: {e}")  
  
convert\_csv\_to\_xlsx("employees.csv", "employees\_groups.xlsx")

**Результат:**

**Зображення, що містить текст, знімок екрана, число, програмне забезпечення

Автоматично згенерований опис**

***4.*** *Третя програма зчитує дані з CSV файлу і виконує наступні функції:*

* Виводить повідомлення про відсутність, або проблеми при відкритті файлу CSV, або Ок.
* Рахує кількість співробітників чоловічої і жіночої статі. Результати виводить в консоль і будує відповідну діаграму.
* Рахує кількість співробітників кожної вікової категорії (вказані в пункті 3).
* Результати виводить в консоль і будує відповідну діаграму.
* Рахує кількість співробітників жіночої та чоловічої статі кожної вікової категорії (вказані в пункті 3).
* Результати виводить в консоль і будує відповідні діаграми.

**Код:**

import pandas as pd  
import matplotlib.pyplot as plt  
from datetime import datetime  
  
def read\_csv(file\_path):  
 try:  
 data = pd.read\_csv(file\_path)  
 return data, None  
 except Exception as e:  
 return None, f"Не вдалося відкрити файл: {e}"  
  
def calculate\_age(birth\_date\_str):  
 try:  
 birth\_date = datetime.strptime(birth\_date\_str, '%Y-%m-%d')  
 today = datetime.today()  
 age = today.year - birth\_date.year - ((today.month, today.day) < (birth\_date.month, birth\_date.day))  
 return age  
 except Exception as e:  
 return None  
  
def categorize\_age(age):  
 if age is None:  
 return None  
 elif age < 18:  
 return 'до 18'  
 elif 18 <= age <= 45:  
 return '18-45'  
 elif 45 < age <= 70:  
 return '46-70'  
 else:  
 return 'старші 70'  
  
def main():  
 file\_path = 'employees.csv'  
 data, error = read\_csv(file\_path)  
  
 if error:  
 print(error)  
 return  
 else:  
 print("Файл успішно відкрито!")  
  
 if 'Стать' not in data.columns or 'Дата народження' not in data.columns:  
 print("Невірний формат CSV файлу. Перевірте наявність колонок 'Стать' і 'Дата народження'!")  
 return  
  
 gender\_counts = data['Стать'].value\_counts()  
 print("\nРозподіл співробітників по статі:")  
 print(f"Чоловіків: {gender\_counts.get('Чоловіча', 0)}")  
 print(f"Жінок: {gender\_counts.get('Жіноча', 0)}")  
  
 plt.figure(figsize=(10, 6))  
 gender\_counts.plot(kind='bar', color=['#1f77b4', '#ff69b4'], edgecolor='black') # Нові кольори  
 plt.title('Розподіл співробітників по статі')  
 plt.xlabel('Стать')  
 plt.ylabel('Кількість')  
 plt.xticks(rotation=0)  
 plt.grid(axis='y')  
 plt.show()  
  
 data['Вік'] = data['Дата народження'].apply(calculate\_age)  
  
 if data['Вік'].isnull().any():  
 print("Помилка обчислення віку. Перевірте правильність формату дати народження.")  
 return  
  
 age\_categories = data['Вік'].apply(categorize\_age).value\_counts()  
 print("\nКількість співробітників у кожній віковій категорії:")  
 for category, count in age\_categories.items():  
 print(f"{category}: {count}")  
  
 plt.figure(figsize=(10, 6))  
 age\_categories.plot(kind='bar', color=['#98fb98', '#ffcc99', '#ffa07a', '#20b2aa'], edgecolor='black') # Нові кольори  
 plt.title('Розподіл співробітників по вікових категоріях')  
 plt.xlabel('Вікова категорія')  
 plt.ylabel('Кількість')  
 plt.xticks(rotation=45)  
 plt.grid(axis='y')  
 plt.show()  
  
 gender\_age\_distribution = data.groupby(['Стать', data['Вік'].apply(categorize\_age)]).size().unstack().fillna(0)  
 print("\nРозподіл співробітників за статтю і віковими категоріями:")  
 print(gender\_age\_distribution)  
  
 gender\_age\_distribution.plot(kind='bar', figsize=(12, 8), edgecolor='black', color=['#1e90ff', '#ff69b4', '#32cd32', '#ffa07a']) # Нові кольори  
 plt.title('Розподіл співробітників по вікових категоріях та статі')  
 plt.xlabel('Стать')  
 plt.ylabel('Кількість')  
 plt.xticks(rotation=45)  
 plt.grid(axis='y')  
 plt.legend(title='Вікова категорія')  
 plt.tight\_layout()  
 plt.show()  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

**Результат:**

**Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис**

**Зображення, що містить текст, знімок екрана, схема, Графік

Автоматично згенерований опис**

**Зображення, що містить текст, схема, знімок екрана, Графік

Автоматично згенерований опис**

**Зображення, що містить текст, знімок екрана, схема, Графік

Автоматично згенерований опис**

***5.*** *Створити файл requirments.txt, в якому зберігаються всі встановлені модулі та пакети.*

**

*Зображення, що містить текст, знімок екрана, програмне забезпечення, Мультимедійне програмне забезпечення

Автоматично згенерований опис*

***6.*** *Завантажити проект на сервіс GitHub.*