МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра інформаційних систем та мереж

Лабораторна робота №8

з дисципліни

СПЕЦІАЛІЗОВАНІ МОВИ ПРОГРАМУВАННЯ

на тему

ВІЗУАЛІЗАЦІЯ ТА ОБРОБКА ДАНИХ ЗА ДОПОМОГОЮ СПЕЦІАЛІЗОВАНИХ БІБЛІОТЕК PYTHON

Виконав:

ст. гр. РІ-31

ЛАЗАР В.С.

Прийняв:

ЩЕРБАК С.С.

Львів-2024

**Мета роботи:**

Розробка додатка для візуалізації CSV-наборів даних за допомогою Matplotlib та базових принципів ООП (наслідування, інкапсуляція, поліморфізм).

**Хід роботи:**

**Завдання 1: Вибір CSV-набору даних**

Обрати CSV-набір даних, який треба візуалізувати. Переконатися, що він містить відповідні дані для створення змістовних візуалізацій.

**Завдання 2: Завантаження даних з CSV**

Написати код для завантаження даних з CSV-файлу в додаток Python. Використовувати бібліотеки, такі як Pandas, для спрощення обробки даних**.**

**Завдання 3: Дослідження даних**

Визначити екстремальні значення по стовпцям.

**Завдання 4: Вибір типів візуалізацій**

Визначити, які типи візуалізацій підходять для представлення вибраних наборів даних. Зазвичай це може бути лінійні графіки, стовпчикові діаграми, діаграми розсіювання, гістограми та секторні діаграми.

**Завдання 5: Підготовка даних**

Попередньо обробити набір даних за необхідністю для візуалізації. Це може включати виправлення даних, фільтрацію, агрегацію або трансформацію.

**Завдання 6: Базова візуалізація**

Створити базову візуалізацію набору даних, щоб переконатися, що можна відображати дані правильно за допомогою Matplotlib. Розпочати з простої діаграми для візуалізації однієї змінної.

**Завдання 7: Розширені візуалізації**

Реалізувати більш складні візуалізації, виходячи з характеристик набору. Поекспериментувати з різними функціями Matplotlib та налаштуваннями.

**Завдання 8: Декілька піддіаграм**

Навчитися створювати кілька піддіаграм в межах одного малюнка для відображення декількох візуалізацій поруч для кращого порівняння.

**Завдання 9: Експорт і обмін**

Реалізувати функціональність для експорту візуалізацій як зображень (наприклад, PNG, SVG) або інтерактивних веб-додатків (наприклад, HTML).

**Код програми:**

"""The user interface of the lab work"""

import logging

import seaborn as sns

import matplotlib.pyplot as plt

from Data.Shared.classes.data\_io import DataIO

from Data.Lab8.BLL.classes.data\_processor import DataProcessor

logger = logging.getLogger(\_\_name\_\_)

logging.basicConfig(filename='Logs/logs.log', encoding='utf-8', level=logging.DEBUG)

class Console:

"""The console class of this lab work"""

instance = None

def \_\_new\_\_(cls):

if cls.instance is None:

cls.instance = super(Console, cls).\_\_new\_\_(cls)

return cls.instance

def \_\_init\_\_(self, file\_path='data.csv', group\_column='Category',

value\_column='Purchase Amount (USD)'):

self.file\_path = f"Data/Lab8/Imports/{file\_path}"

self.data = None

self.group\_column = group\_column

self.value\_column = value\_column

self.main()

@staticmethod

def annotate(ax, group\_column, data, is\_boxplot=False):

"""Creates the annotations for the plots"""

unique\_categories = data[group\_column].unique()

for i, element in enumerate(ax.patches[:len(unique\_categories)]):

category\_name = unique\_categories[i]

center\_x = element.get\_x() + element.get\_width() / 2 # Центр стовпця

height\_y = element.get\_height() # Висота стовпця

if is\_boxplot:

center\_x = element.get\_x() + element.get\_width() / 2

height\_y = element.get\_y() + element.get\_height()

ax.text(center\_x, height\_y + 0.5, category\_name, ha='center', fontsize=10)

def basic\_visualization(self):

"""Outputs basic visualization of the imported data"""

prepared\_data = DataProcessor.prepare\_data(self.data, self.group\_column, self.value\_column)

fig, ax = plt.subplots(figsize=(10, 6))

ax.bar(prepared\_data[self.group\_column], prepared\_data[self.value\_column], color='skyblue')

ax.set\_xlabel(self.group\_column)

ax.set\_ylabel(self.value\_column)

ax.set\_title(f"Average {self.value\_column} by {self.group\_column}")

ax.tick\_params(axis='x', rotation=45)

# noinspection PyTypeChecker

self.annotate(ax, self.group\_column, prepared\_data)

DataIO.save\_visualization(fig, "basic\_visualization")

plt.tight\_layout()

plt.show()

logger.info("[Lab 8] Ended making basic graph")

def advanced\_visualization(self):

"""Outputs advanced histogram of the imported data"""

prepared\_data = DataProcessor.prepare\_data(self.data, self.group\_column, self.value\_column)

fig, ax = plt.subplots(figsize=(10, 6))

sns.histplot(prepared\_data[self.value\_column], kde=True, ax=ax, color='green')

ax.set\_title(f"Histogram of {self.value\_column}")

ax.set\_xlabel(self.value\_column)

ax.set\_ylabel('Frequency')

DataIO.save\_visualization(fig, "advanced\_visualization\_histogram")

plt.tight\_layout()

plt.show()

logger.info("[Lab 8] Ended making advanced graph")

def multiple\_subplots(self):

"""Outputs multiple subplots of the imported data"""

prepared\_data = DataProcessor.prepare\_data(self.data, self.group\_column, self.value\_column)

fig, axs = plt.subplots(1, 2, figsize=(14, 6))

sns.barplot(x=self.group\_column, y=self.value\_column,

data=prepared\_data, ax=axs[0], color='skyblue')

axs[0].set\_title(f"Barplot of {self.value\_column}")

axs[0].set\_xlabel(self.group\_column)

axs[0].set\_ylabel(self.value\_column)

axs[0].tick\_params(axis='x', rotation=45)

self.annotate(axs[0], self.group\_column, prepared\_data)

sns.histplot(prepared\_data[self.value\_column], kde=True, ax=axs[1], color='green')

axs[1].set\_title(f"Histogram of {self.value\_column}")

axs[1].set\_xlabel(self.value\_column)

axs[1].set\_ylabel('Frequency')

DataIO.save\_visualization(fig, "multiple\_subplots\_with\_categories")

plt.tight\_layout()

plt.show()

logger.info("[Lab 8] Ended making multiple graphs")

def main(self):

"""The main menu of this lab work"""

try:

self.data = DataIO.load\_data(self.file\_path)

except FileNotFoundError as e:

print(e)

while True:

prompt = input("1 - Extreme values\n"

"2 - Display basic graph\n"

"3 - Display advanced graph\n"

"4 - Display multiple graphs\n"

"Your choice: ")

match prompt:

case '1':

logger.info("[Lab 8] Found extreme values")

DataProcessor.find\_extreme\_values(self.data)

case '2':

logger.info("[Lab 8] Started making basic graph")

self.basic\_visualization()

case '3':

logger.info("[Lab 8] Started making advanced graph")

self.advanced\_visualization()

case '4':

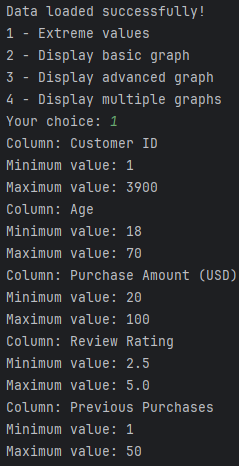
logger.info("[Lab 8] Started making multiple graphs")

self.multiple\_subplots()

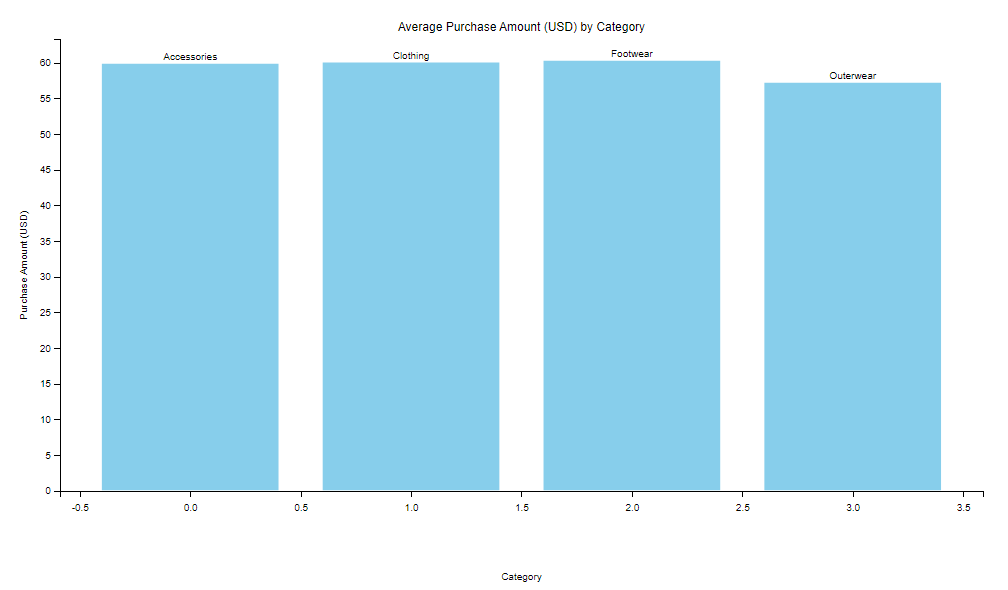
case \_:

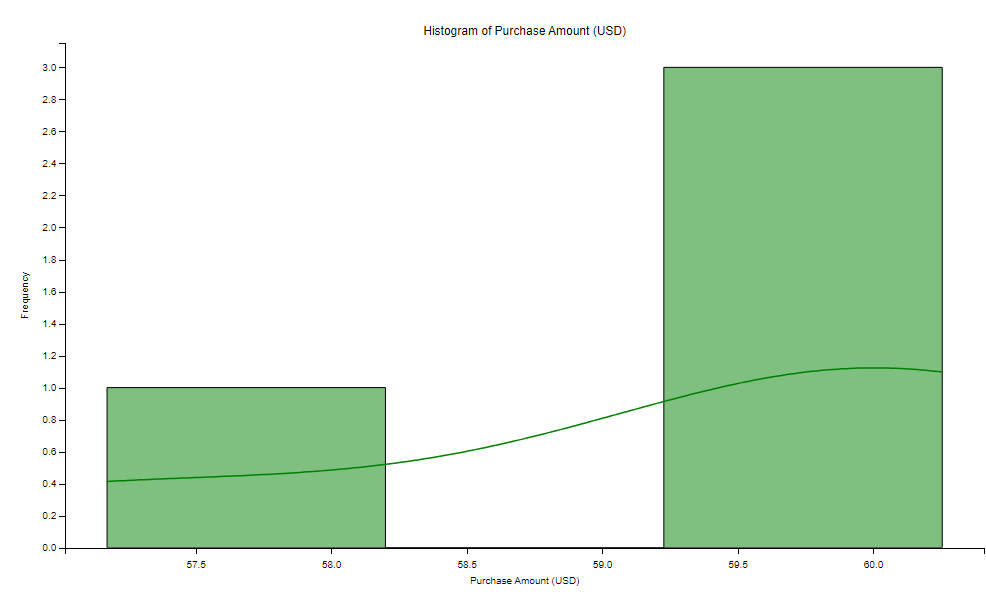
return

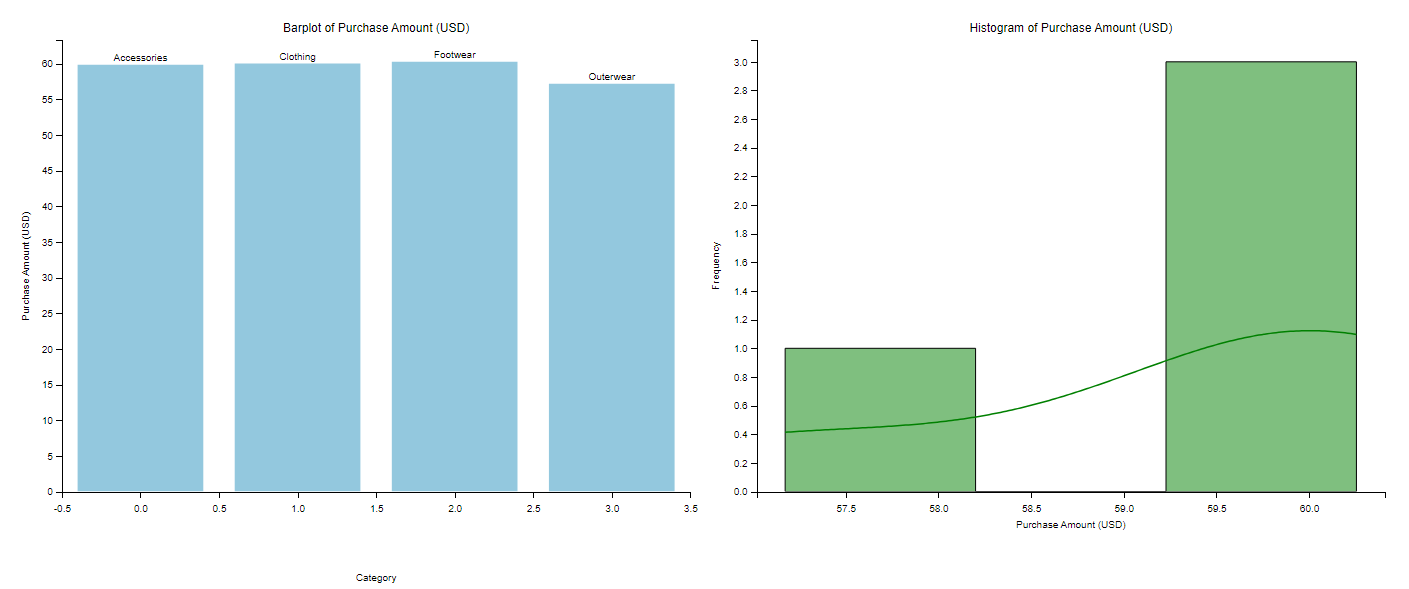
На рис. 1-4 зображено результат виконання програми:



*Рис. 1. Приклад роботи функції пошуку екстремальних значень таблиці*

*  
Рис. 2. Приклад базової візуалізації даних таблиці*

*  
Рис. 3. Приклад розширеної візуалізації даних таблиці*

*  
Рис. 4. Приклад візуалізації кількох піддіаграм даних таблиці*

Посилання на Github: [PaperGlit/Python\_Lab\_8](https://github.com/PaperGlit/Python_Lab_8)

**Висновок:**

Виконавши ці завдання, я створив багатофункціональний додаток для візуалізації CSV-наборів даних за допомогою Matplotlib. Цей проект покращив мої навички візуалізації даних, дозволяючи досліджувати результати з різноманітними наборами даних