**AP LAB\_2**

**~ Prof. Ishan Maheta**

✅ **Title 2.1: Student Grades Analysis from CSV**

🎯 **Objective:**

* To write a Python program that reads student grades from a CSV file.
* To calculate the average score for each student.
* To write the results into a new CSV file.
* To demonstrate effective CSV file manipulation and modular coding using functions.

📘 **Task Description:**  
In this experiment, the task is to process student grade data stored in CSV format. The program reads a CSV file containing students’ marks across different subjects, computes the average score for each student, and saves the results in a new CSV file.

Two approaches are demonstrated:

1. Using Python’s built-in **csv module** for precise control.
2. Using **pandas library** for faster and more concise operations.

💻 **Python Code (Using csv module):**

import csv

dir = "files/program\_1"

# Read data from input.csv

with open(f"{dir}/input.csv", "r", newline="") as input:

    reader = csv.DictReader(input)

    rows = []

    for row in reader:

        scores = [float(row[col]) for col in row if col != "Name"]

        avg = sum(scores) / len(scores)

        rows.append({"Name": row["Name"], "Average": avg})

# Write to output.csv

with open(f"{dir}/output.csv", "w", newline="") as outfile:

    fieldnames = ["Name", "Average"]

    writer = csv.DictWriter(outfile, fieldnames=fieldnames)

    writer.writeheader()

    writer.writerows(rows)

💻 **Python Code (Using pandas module):**

import pandas as pd

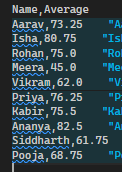
dir = "files/program\_1"

df = pd.read\_csv(f"{dir}/input.csv")  # Read input file

df["Average"] = df.drop("Name", axis=1).mean(axis=1)  # Compute average

df[["Name", "Average"]].to\_csv(f"{dir}/output.csv", index=False)  # Save result

🧪 **Sample Output:**



✅ **Conclusion:**  
This experiment demonstrates how to process CSV data in Python. The **csv module** provides detailed control over reading and writing, while **pandas** offers faster, more concise syntax. Both approaches successfully calculated and exported average marks for each student.

✅ **Title 2.2: Sales Data Analysis from Multiple CSV Files**

🎯 **Objective:**

* To read sales data from multiple CSV files across years and stores.
* To combine sales data with product information.
* To calculate total and average monthly sales.
* To identify the top 5 best-selling products.
* To generate a summarized CSV report.

📘 **Task Description:**  
As a data engineer for a retail company, the goal is to process nationwide sales data spread across multiple CSV files. Each file represents a month’s sales, while a separate file maps product IDs to names.

The program should:

* Read all sales CSV files recursively from a directory.
* Aggregate sales quantities per product.
* Compute total and average monthly sales.
* Identify the **top 5 best-selling products**.
* Save the final results to sales\_summary.csv.

💻 **Python Code:**

import pandas as pd

import os

import traceback

dir = r"files/program\_2"

sales\_dir = os.path.join(dir, "sales\_data")

products\_file = os.path.join(dir, "product\_names.csv")

output\_file = os.path.join(dir, "sales\_summary.csv")

errors = 0

def print\_error(e: Exception) -> None:

    tb = traceback.extract\_tb(e.\_\_traceback\_\_)

    for filename, line, funcname, text in tb:

        print(f"Error -> {e}")

        print(f"File -> {filename}")

        print(f"Function -> {funcname}, Line -> {line}")

        print(f"Code -> {text}\n")

    global errors

    errors += 1

def load(sales\_dir: str, products\_file: str):

    sales = pd.DataFrame()

    months = 0

    try:

        for root, \_, files in os.walk(sales\_dir):

            for file in files:

                sales = pd.concat([sales, pd.read\_csv(os.path.join(root, file))], ignore\_index=True)

                months += 1

        products = pd.read\_csv(products\_file)

    except Exception as e:

        print\_error(e)

        return None, None, 0

    return sales, products, months

def process(sales: pd.DataFrame, products: pd.DataFrame, months: int):

    try:

        totals = sales.groupby("Product\_ID")["Quantity"].sum()

        products["Total"] = products["Product\_ID"].map(totals).fillna(0)

        products["Average"] = (products["Total"] / months).round(2)

        top\_5 = products.sort\_values(by="Total", ascending=False).head(5)

        print("The top 5 products by total quantity are")

        print(top\_5[["Name", "Total"]])

    except Exception as e:

        print\_error(e)

def func(sales\_dir: str, products\_file: str, output\_file: str):

    sales, products, months = load(sales\_dir, products\_file)

    if sales is None or products is None:

        return

    process(sales, products, months)

    try:

        products.to\_csv(output\_file, index=False)

    except Exception as e:

        print\_error(e)

def main():

    func(sales\_dir, products\_file, output\_file)

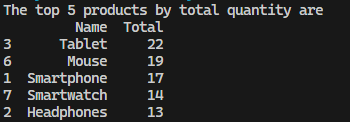
    if errors:

        print(f"Total number of errors occurred -> {errors}")

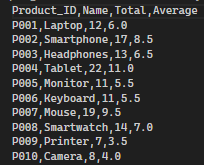
if \_\_name\_\_ == "\_\_main\_\_":

    main()

🧪 **Sample Output:**  
Console Output:



CSV Output (sales\_summary.csv):



✅ **Conclusion:**  
This program successfully processed multi-year, multi-store sales data and generated product-wise sales summaries. It identified the **top 5 products**, calculated **monthly averages**, and produced a clean CSV summary. The solution demonstrates the use of pandas for efficient data handling and ensures scalability for large datasets.