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):

```python
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:

```
Name,Average
Aarav,73.25
Isha,80.75
Rohan,75.0
Meera,45.0
Vikram,62.0
Priya,76.25
Kabir,75.5
Ananya,82.5
Siddharth,61.75
Pooja,68.75
```

✅ 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 numpy as np
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
```

```python
    try:
        for root, _, files in os.walk(sales_dir):
            sales = pd.concat([sales] + [pd.read_csv(os.path.join(root, file)) for file in
files])
            months += len(files)
        sales.reset_index(drop=True, inplace=True)
        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(by="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 occured -> {errors}")

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

🧪 **Sample Output:**

Console Output:

```
The top 5 products by total quantity are
         Name  Total
3      Tablet     22
6       Mouse     19
1  Smartphone     17
7  Smartwatch     14
2  Headphones     13
```

CSV Output (`sales_summary.csv`):

```
Product_ID,Name,Total,Average
P001,Laptop,12,6.0
P002,Smartphone,17,8.5
P003,Headphones,13,6.5
P004,Tablet,22,11.0
P005,Monitor,11,5.5
P006,Keyboard,11,5.5
P007,Mouse,19,9.5
P008,Smartwatch,14,7.0
P009,Printer,7,3.5
P010,Camera,8,4.0
```

✅ **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.
