The “Three Strikes Before AI” rule:

Tagline: Try it yourself three times (documentation support is cool!). If you still can’t solve it, then AI becomes your assistant, not your shortcut.

Useful links:

- https://pandas.pydata.org/docs/user_guide/10min.html
- https://pandas.pydata.org/docs/user_guide/index.html#user-guide
- https://matplotlib.org/stable/tutorials/index.html
- https://matplotlib.org/stable/users/explain/quick_start.html

1. Dataset crafting

Let’s simulate a daily debit or credit bank account transactions.

Create a Data frame with multiple entries of payments for 15 customers in 50 different random days throughout random months of the year. Your Data frame should have the columns “Customer_Name”, “Customer_ID” (name and Id should always belong to the same person), “Day”, “Month”, and “Value” that can be a positive or negative float (2 decimals – 0.00) between -15.00 and 15.00. Save the data frame in a csv file named “*daily_entries.csv*”.

Suggested Methods:

```
import random,  calendar

amount = round(random.uniform(-15, 15), 2)

year, month = 2025, 3
day = random.randint(1, calendar.monthrange(year, month)[1])

idx = random.sample(range(n_rows), k)
```

2. Dataset handling

A. Part A:

Load the file “*daily_entries.csv*”. Create methods to group the customers’ balance per month and move it in another Data frame. Save the second data frame in a csv file named “*monthly_balances.csv*”.

B. Part B:

Use the same methods of item 1 to create a second daily entries with the same customers in a second Data frame. Create a method to insert noise in the daily entries' datasets (nan at random positions) and apply it method to the second data frame. Save the file with the applied noises to a csv named "*daily_entries_2.csv*"

C. Part C:

Implement a method to load the "*daily_entries_2.csv"* and update the "*monthly_balances.csv".*

- What happens then?

After updating the "*monthly_balances.csv",* combine the two datasets ("*daily_entries.csv"* and *"daily_entries_2"*) into one dataset and save it to a third file called "*daily_entries_total.csv*". (Hint: Check ways of doing it smartly).

Suggested Methods:

```
df.groupby(...)["Value"].sum().rename(columns={"Value":"........"}))
```

3. Data Plotting

Load the file "*daily_entries_total.csv"* and let's have some fun plotting.

Create a method for each situation below. You should also decide with your group the best graph type to use.

- Given a customer name or ID, plots all the entries of that customer.
- Given a day of the year (with the month), plots all the entries on that day. (Hint: use the Exception class to handle days with no entries)
- Given a customer name or ID, plots the balance of that customer per month in grouped column bars
- Given a starting and ending date, plots all the entries within this period per customer
- Given a starting and ending date, plots all the negative transactions within that period

4. Model-View-Controller (MVC) Desing

D. Part A:

Consider all the methods you have developed and divide them into a MVC architecture by drawing its UMLs.

E. Part B:

Perform the implementations in your code moving the corresponding methods to the corresponding python packages (files).

F. Part C:

Let’s assume now that we have 3 different types of users in our system:

- Bank Manager: can see all the information
- Customer: can only see the information of themselves
- Data analyst: can see everything BUT customers’ name

Modify your code so the policies above are respected. You can create a “login system” (a variable that controls who has logged into the system) that gives the right access to the users.