PYTHON LEVEL 2 – TRANSACTION ANALYTICS PROJECT

Project Overview

In this project, you will develop a Python script to analyze customers' credit card spending patterns. The goal is to apply your knowledge of basic Python concepts (variables, loops, conditionals, data structures, and pandas) together with more advanced skills (functions, comprehensions, lambdas, built-in functions, and data visualization) while focusing on customer transaction analytics.

Objective

Build a Python application that:

- Processes customer credit card transactions.
- Computes spending statistics by customer and by category.
- Identifies "big spenders" and spending patterns.
- Generates a summary report for the analytics team.

Input Data

Customers will be represented as a list of dictionaries with fields: *customer_id*, *age, region, account_type* and *transactions* (where each transaction is a list with an amount, a category and a currency). Example:

```
customers = [{
        "customer_id": "C001",
        "age": 34,
        "region": "North",
        "account_type": "Gold",
        "transactions": [
          {"amount": 120, "category": "Food", "currency": "EUR"},
          {"amount": 300, "category": "Travel", "currency": "USD"},
          {"amount": 50, "category": "Entertainment", "currency": "EUR"},
        ]
        }]
```

Requirements

- Use functions to encapsulate logic (e.g., total spending, average per category).
- Use list/dict comprehensions to process transactions efficiently.
- Use lambda expressions and Python built-ins such as map, filter, zip, sorted, any, all, round.
- Yield transactions lazily or filter with generator expressions.
- Use modules: separate code into main.py, client.py, risk_tools.py, and data.py
- Use pandas to store processed data.
- Visualize spending patterns with matplotlib and seaborn.

Tasks

1. **Data Processing**
   - Print each customer_id, age, region, account_type, and number of transactions.
   - Write a helper function *to_eur(amount, currency)* that converts a transaction amount into EUR (assume 1 USD = 0.9 EUR).
   - Using this helper (and without changing the original data):
     - Compute total_spending (in EUR) for each customer, rounded to two decimal places.
     - Determine top_category per customer (the category where they spent the most €, based on converted amounts).

- Build a list of these row dictionaries, then create a pandas DataFrame with columns customer_id, age, region, account_type, and total_spending.
    - Compute total spending per region (group by region) and print it.
    - **Tip**: Do not remove the currency field from transactions. Convert on the fly when summing/aggregating.

2. **Functions and Comprehensions**
    - Write avg_transaction(customer) to compute the average transaction (in EUR) for a single customer. Print the average for the first 5 customers.
    - Extract all Food transactions (in EUR) across all customers into a single list using list comprehension.
        - Print the total number of food transactions.
        - Create a pandas Series from this list and display some summary statistics (describe()).
    - Build a dictionary mapping each customer_id to total_spending (already computed in Step 1; use the list you've built from the row dictionaries). Then, compare the first 5 values of this dictionary to the corresponding values from the DataFrame.
3. **Summary Report with Lambdas, Built-ins, and Generators**
    - Build a dictionary summary_report with:
        - Total number of customers
        - Total spending across all customers (use a **generator expression**)
        - Customer with the lowest spending (use *sort_values* with a **lambda**)
        - Average spending per customer (use the generator expression again or by applying a lambda to df["total_spending"])
        - Spending by region. Group df by region, but try to apply **lambda aggregation** (with df.apply) instead of the direct sum.
        - Find the category with the highest median spending (use groupby on top_category and apply with a **lambda** that computes the median of total spending). Store the result as a **tuple** (category_name, median_value) in the summary report.
    - Finally, print the summary_report in a readable way.
4. **Data Visualization**
    - Create **six figures** to explore customer spending patterns, using **matplotlib** or **seaborn** as specified below. Save at least two figures (e.g. the horizontal bar chart and the boxplot), one as JPEG, and the other as PNG.
        - Use **matplotlib** to draw:
            - A **horizontal bar chart** with the top ten customers by total spending
            - A **pie chart** with the share of spending by region (requires groupby and displaying percentages)
            - A **bar chart** with the median spending by category (requires a groupby)
        - Use **seaborn** to plot:
            - A **histogram** of total spending with a smooth density curve and vertical lines for the mean and median.
            - A **boxplot** with the spending distribution by account type (x: account; y: spending)
            - A **scatter plot** with age vs. spending, colored by top category.