Process Automation Specialist – Data Processing Take-Home Assignment

Overview
You are tasked with building an end-to-end data processing workflow that mirrors common challenges in process automation. The objective is to demonstrate your ability to design, implement, and test a solution that ingests raw data, processes and cleans it, applies normalization and enrichment routines, and outputs the results in a specified format. **You must build your solution from scratch and showcase your own approach while leveraging Python 3.11 features effectively.** Utilization of built-ins is acceptable, over-reliance on third-party libraries is discouraged.

Background & Scenario
Imagine you have been contracted by a retail company that collects sales data from multiple sources:

- **Sales Data:** A CSV file containing daily sales transactions.
- **Product Data:** A JSON API endpoint providing detailed product information (e.g., product ID, category, price, and availability).

Your goal is to integrate these sources, clean and normalize the data, enrich the sales records with product details, and produce an aggregated summary report (e.g., total sales per category, average transaction value, etc.). Sample data has been provided.

Objectives
1. **Data Ingestion:**
   - Read raw sales data from a CSV file.
   - Fetch product details from a JSON API (the static JSON file provided is meant to represent the API, building an API is unnecessary).

2. **Data Processing:**
   - Clean the sales data by handling missing values, correcting data types, and ensuring consistency.
   - Normalize the data (e.g., standardizing date formats, product codes, etc.).
   - Enrich the sales data by merging it with product information using product IDs as keys.
   - Handle any discrepancies or conflicts in the data.

3. **Output:**
   - Produce an output report in a specified format (e.g. a new CSV file) that aggregates key metrics.

Expanded Output Requirements

The final output must be a CSV file named `aggregated_report.csv` that summarizes the processed data by product category. The CSV file **must** include the following columns in the exact order specified:

1. **category**
   - **Data Type:** String
   - **Description:** The product category as provided by the product data.
   - **Example:** `Gadgets`

2. **total_sales**
   - **Data Type:** Numeric (float, with two decimal precision)
   - **Description:** The total sale amount for all transactions within the category.
   - **Example:** `375.00`

3. **total_transactions**
   - **Data Type:** Integer
   - **Description:** The total number of transactions contributing to the category's sales.
   - **Example:** `6`

4. **average_transaction_value**
   - **Data Type:** Numeric (float, with two decimal precision)
   - **Description:** The average sale amount per transaction, calculated as `total_sales / total_transactions`.
   - **Example:** `62.50`

5. **total_quantity**
   - **Data Type:** Integer
   - **Description:** The cumulative number of product units sold in the category.
   - **Example:** `14`

Additional Output File Instructions:
- Ensure that all numeric values in the CSV file are formatted appropriately (e.g., floats with exactly two decimal places).
- The CSV should not include any additional columns or headers beyond the ones specified.
- Rows in the CSV file should be sorted by the `category` name in alphabetical order.

Task Breakdown

1. Python Stand-Alone Script
- **Purpose:** Create a script that orchestrates the complete data processing workflow.
- **Requirements:**
  - Accept input file paths or endpoints as command-line arguments or prompt the user for them during script run.
  - Read the raw CSV sales data.
  - Read the raw JSON product information.
  - Clean, normalize, and enrich the sales data.
  - Generate and output a summary report.
  - Leverage Python 3.11.

2. Library Module(s)
- **Purpose:** Encapsulate reusable functions and/or classes to handle core processing tasks.
- **Requirements:**
  - Design one or more modules that separate concerns (e.g., one for data ingestion, one for processing, and one for enrichment).
  - Write clean, maintainable code with proper function definitions, error handling, and documentation strings.
  - Use modern Python best practices and language features.

3. Test Cases
- **Purpose:** Validate the correctness and robustness of your solution.
- **Requirements:**
  - Develop unit tests for individual functions/components.
  - Ensure tests cover edge cases, such as missing data, incorrect formats.
  - Use Python’s built-in `unittest` module or another testing framework (e.g., `pytest`) that supports Python 3.11.

Deliverables
- **Python Script:**
  - A stand-alone script (e.g., `process_data.py`) that implements the workflow.

- **Library Module(s):**
  - One or more Python modules (e.g., `data_ingestion.py`, `data_processing.py`, `data_enrichment.py`) encapsulating core functionality.

- **Test Files:**
  - Test cases for both unit and integration testing (e.g., `test_data_processing.py`).

- **Documentation:**
  - A README or documentation file (e.g., `README.md`) that outlines:
    - Your overall approach and design decisions.

  - How to run the script and tests.
  - Any assumptions made and potential areas for future improvement.

Evaluation Criteria
- **Code Quality:**
  - Clear structure, readability, and adherence to Python best practices.
  - Effective use of Python 3.11 features.

- **Problem-Solving:**
  - Demonstrated ability to analyze the scenario and implement a comprehensive solution.
  - Robust handling of data issues (missing values, incorrect formats, etc.).
  - We want to see how you'd solve the problem, not how a library solves it for you.

- **Modularity & Reusability:**
  - Proper separation of concerns between the script and library modules.
  - Well-documented functions and clear code organization.

- **Testing:**
  - Unit tests covering functions you wrote.
  - Use of a clear and consistent testing approach.

- **Documentation:**
  - Clarity in the README regarding your design decisions and instructions for running the solution.
  - Concise and accurate explanations without unnecessary boilerplate or hints.

Submission Instructions
1. **Prepare Your Package:**
   - Organize your solution in a Git repository or a compressed folder.
   - Ensure that all deliverables (scripts, modules, tests, and documentation) are included.

2. **Include Documentation:**
   - Provide a README file explaining your approach, design choices, and how to execute your code and tests.

3. **Submission Format:**
   - Submit the package via the provided platform or as directed by the hiring team.

Asset Files for the Assignment

1. Sales Data CSV (`sales_data.csv`)
```csv
date,transaction_id,product_id,quantity,sale_amount
2023-03-01,TX001,P001,2,40.00
2023-03-01,TX002,P002,1,25.00
2023-03-02,TX003,P003,3,45.00
2023-03-02,TX004,P004,2,80.00
2023-03-03,TX005,P001,1,20.00
2023-03-03,TX006,P005,5,150.00
2023-03-03,TX007,,2,30.00
03/04/2023,TX008,P002,2,50.00
2023-03-04,TX009,P004,1,40.00
2023-03-04,TX010,P005,3,90.00
```

2. Product Data JSON (`product_data.json`)
```json
[
    {
        "product_id": "P001",
        "product_name": "Widget",
        "category": "Gadgets",
        "price": 20.00,
        "availability": "in stock"
    },
    {
        "product_id": "P002",
        "product_name": "Gizmo",
        "category": "Gadgets",
        "price": 25.00,
        "availability": "in stock"
    },
    {
        "product_id": "P003",
        "product_name": "Thingamajig",
        "category": "Tools",
        "price": 15.00,
        "availability": "out of stock"
    },
    {
        "product_id": "P004",
        "product_name": "Doohickey",
        "category": "Tools",
        "price": 40.00,
        "availability": "in stock"
    },
    {
        "product_id": "P005",
        "product_name": "Contraption",
        "category": "Gadgets",
        "price": 30.00,
        "availability": "in stock"
    }
]
```

3. Aggregated Report CSV (`aggregated_report.csv`)
```csv
category,total_sales,total_transactions,average_transaction_value,total_quantity
Gadgets,375.00,6,62.50,14
Tools,165.00,3,55.00,6
```

Explanation of the Aggregated Report Calculation:

- **Gadgets:**
  - **Transactions:** TX001 (P001), TX002 (P002), TX005 (P001), TX006 (P005), TX008 (P002), TX010 (P005)
  - **Total Sales:** 40.00 + 25.00 + 20.00 + 150.00 + 50.00 + 90.00 = 375.00
  - **Total Transactions:** 6
  - **Average Transaction Value:** 375.00 / 6 ≈ 62.50
  - **Total Quantity:** 2 + 1 + 1 + 5 + 2 + 3 = 14

- **Tools:**
  - **Transactions:** TX003 (P003), TX004 (P004), TX009 (P004)
  - **Total Sales:** 45.00 + 80.00 + 40.00 = 165.00
  - **Total Transactions:** 3
  - **Average Transaction Value:** 165.00 / 3 = 55.00
  - **Total Quantity:** 3 + 2 + 1 = 6