Data Science Assignment

CSCU9M3 – Scripting for Data Science

This assignment contributes to **60% of your overall grade** in the module. It is designed to evaluate your Python scripting abilities in a practical, real-world scenario, where handling and processing complex datasets is essential. Through this project, you will demonstrate technical programming skills, and your capacity to make critical decisions about data handling, representation, and tool selection, that are essential in professional settings.

The assignment assesses your achievement of all five Intended Learning Outcomes (ILOs) for the course:

* **ILO1**: You will develop and test a Python program to load, manipulate, analyse, and store data from a nested file structure, applying key concepts in data processing.
* **ILO2**: You will select and apply Python’s standard libraries for tasks such as data traversal, file handling, and JSON manipulation, showcasing your ability to choose tools appropriate for processing data in a secure environment.
* **ILO3**: As you navigate varied data formats, you’ll need to design suitable representations to capture, aggregate, and report essential information from different JSON files.
* **ILO4**: You will implement and customize basic calculations to generate statistics and summarize data, translating mathematical concepts into workable Python code.
* **ILO5**: Finally, by adhering to real-world constraints—such as security and error handling—you will demonstrate your capacity to choose and employ the right scripting approach for this data-processing task.

This assignment provides a valuable opportunity to integrate all aspects of your learning, helping you build confidence in tackling complex data processing projects from start to finish.

## Premise

You have been hired by **Heather Glen Rehabilitation Centre**, a Scottish rehabilitation and recovery facility that offers a variety of supportive healthcare services, including physiotherapy, occupational therapy, post-surgery rehabilitation, and mental health counselling. Each year, the centre compiles an annual report to assess patient outcomes, service usage, and treatment trends across its departments. As part of their data analysis team, your task is to create a Python script that will aggregate the data from the past years, stored across numerous files, into a structured report for the management team.

Given the sensitive nature of the data, you will work with a synthetic sample provided for development. Your final script will be tested on the actual data and is expected to generate two outputs: a structured JSON file for internal reporting and a text summary for Heather Glen’s annual newsletter. This project provides a real-world opportunity to demonstrate your data processing skills in Python, as well as your ability to handle data responsibly and with attention to detail.

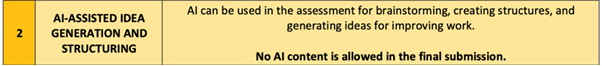
# Academic Integrity

This is an **Individual Assessment**, and all University policies on Assessments and Academic Integrity apply. <https://www.stir.ac.uk/media/stirling/services/academic-registry/documents/academic-integrity-policy.pdf>

Adhering to these guidelines is crucial for ensuring that your work reflects your own understanding and skills. Here are the key points to consider:

1. You must complete this assignment **independently**—collaboration with other students is not permitted.
2. Open discussion with other students to deepen your understanding of the concepts is encouraged; however, sharing or discussing specific solutions to the assignment, in whole or in part, is strictly prohibited and will be considered academic misconduct.
3. While you may use tools like Large Language Models (LLMs) or generative AI for brainstorming or to help structure your approach, **you are not allowed to submit any code or text generated by such tools** as part of your final submission.
4. You are welcome to consult any sources (online or offline) to aid in creating your solution, provided you follow the rules above regarding independent work and the limited use of AI tools.
5. If you refer to any external source for any part of your solution—no matter how minor—you must properly acknowledge the source. Cite it both where you use it within your code and in a separate bibliography document accompanying your submission. Failure to do so may result in an Academic Integrity investigation.
6. If you incorporate code from an external source into your solution, you must also explain how you modified it to integrate effectively with your own work, in addition to providing proper citation.

By following these guidelines, you uphold the principles of academic integrity and contribute to a fair, honest learning environment.



# Task Instructions from Heather Glen Rehabilitation Centre

Dear Data Analysis Team,

Welcome aboard! We at **Heather Glen Rehabilitation Centre** are delighted to have your assistance in preparing our annual report. Our goal is to create an insightful, data-driven overview of our services, patient outcomes, and treatment trends over the past year. This report will help us assess our service usage, support planning for the future, and share highlights with our community.

As part of this project, we ask that you develop a Python script to analyse and compile data from our records into two essential outputs:

1. A structured **JSON report** that provides a detailed overview for our internal use.
2. A **text summary** with key statistics and trends, ready for inclusion in our annual newsletter.

Following are the details and specifications for the task.

## Assignment Scope

The data you’ll be working with is organised in a folder structure by **year** and **month**, with individual .json files containing records of patient treatments and services. Each file has a consistent set of core fields but may occasionally contain additional fields specific to certain treatments or cases.

Our team has provided a synthetic sample of this data for your development process. Once your script is complete, it will be tested on the actual data stored in our secure environment.

## Server Specifications

Your script will be tested on our server, which is configured with the following specifications. Please ensure that your code is compatible with these settings to avoid runtime issues:

* **Python Version**: 3.9
* **Available Libraries**:
  + **Standard Libraries**: Including os, json, datetime, collections, math, statistics, and itertools.
  + **Pandas**: Installed and available for data manipulation, aggregation, and analysis.
  + **Matplotlib**: Installed and available for producing graphs and visualisations if needed.

**Note**: No internet access is available on the server, so all data sources must be local. Make sure your code does not attempt to access external resources, as it will result in errors.

## Requirements for Data Aggregation

Your script should:

* **Traverse the nested folder structure** to locate all .json files, ensuring that each record is processed.
* **Aggregate data on an annual basis**, compiling statistics and insights for the year as a whole. The primary focus should include:
  + **Total record count** for the year
  + **Most common diagnoses and treatments**
  + **Monthly record breakdown** to show usage trends over the year
  + Any additional insights or trends that you find relevant to highlight

This level of detail will allow our team to gain valuable insights into our service offerings and adjust our resources to meet patient needs effectively.

## Output Specifications

### 1. JSON Report

**Overview**

The JSON output file should be named ZZZZ\_annual\_report.json (replace ZZZ with the year). It will aggregate data from all visit records for a given year and present it in a structured, machine-readable format. This report is intended for internal analysis by Heather Glen Rehabilitation Centre, summarising patient visits, treatments, and trends across departments.

The structure of the JSON output should include key statistics and insights derived from the data.

**Required Structure**

The JSON report should contain the following fields at the top level:

1. **year**: *(Integer)*
   * The year for which the report is generated.
2. **total\_records**: *(Integer)*
   * The total number of patient visits for the year.
3. **departments**: *(List of Department Summaries)*
   * A list containing summaries for each department, with details on common diagnoses, treatments, and overall visit trends.
   * Each department entry should include:
     + **department\_name**: *(String)* – Name of the department (e.g., "Physiotherapy").
     + **total\_visits**: *(Integer)* – Total number of visits for this department.
     + **common\_diagnoses**: *(List of Dictionaries)* – List of the most common diagnoses within this department, sorted by frequency in descending order.
       - Each entry in this list should contain:
         * **diagnosis**: *(String)* – The diagnosis name.
         * **frequency**: *(Integer)* – Number of occurrences of this diagnosis.
     + **common\_treatments**: *(List of Dictionaries)* – List of the most common treatments within this department, sorted by frequency in descending order.
       - Each entry in this list should contain:
         * **treatment**: *(String)* – The treatment type.
         * **frequency**: *(Integer)* – Number of occurrences of this treatment.
     + **average\_treatment\_duration**: *(Float)* – The average duration of treatments for this department, in minutes.
4. **monthly\_summary**: *(List of Monthly Summaries)*
   * A list containing monthly summaries to help identify trends over the year.
   * Each month entry should include:
     + **month**: *(String)* – The month name (e.g., "January").
     + **record\_count**: *(Integer)* – The number of visits recorded in this month.
     + **follow\_up\_visits**: *(Integer)* – The number of visits where follow-up was recommended.
5. **overall\_outcomes**: *(Dictionary)*
   * Summary of outcomes across all departments for the year.
   * Contains:
     + **improved**: *(Integer)* – Number of visits where the outcome was "Improved".
     + **no\_change**: *(Integer)* – Number of visits where the outcome was "No Change".
     + **worsened**: *(Integer)* – Number of visits where the outcome was "Worsened".

**Example Structure**

Below is an example JSON structure based on the requirements above:



### 2. Text Summary

Create a .txt file named **XXXX**\_summary.txt (replace **XXXX** with the corresponding year) that highlights key statistics from the year’s data. This summary will serve as a reader-friendly version for our annual newsletter, so it should be written in clear, concise paragraphs.

The text summary should cover:

* **Overall statistics** such as the total number of records and common treatments.
* **Notable trends or insights** from the data, such as high-demand months for specific therapies, or noteworthy shifts in patient needs.

Please organise the summary in a way that is easy to understand and ready for publication.

## Important Considerations

As this data is sensitive, we have some guidelines to ensure the security and integrity of your work:

* **Your final Python script that you submit should not use logging or debugging print statements** that might inadvertently expose sensitive information. Obviously, you are free to use these in your development but make sure you remove them before submitting.
* **Do not rely on any external packages** that require internet access; please use only the Python libraries listed in the server’s specifications.
* **Handle missing fields gracefully.** If an expected field is missing, skip or flag it without causing an error in the script.

## Deliverables

Please submit the following:

1. **Contract Agreement**: A template contract is provided for you to submit, acknowledging responsibilities and accepting the terms of the project.
2. **Python Script**: Your final .py file that generates the JSON and text outputs, as well as any other reasonable or useful outputs when run on our secure server.
3. **README file**: A .txt file that explains the Python script’s requirements and the outputs it generates.
4. **Optional Draft Submission**: We encourage you to submit a draft of your script ahead of the final deadline for preliminary testing. This will be run on a test server to ensure it executes without errors (content correctness will not be evaluated at this stage).

## Final Remarks

We look forward to reviewing your work and seeing the insights you uncover from our data. This report is a crucial part of our planning and community outreach, and we appreciate your attention to detail, accuracy, and respect for data security.

Thank you for your dedication to helping Heather Glen Rehabilitation Centre continue to support our patients with excellence.

Warm regards,  
**Heather Glen Rehabilitation Centre Management Team**

# Data Structure Description for Heather Glen Rehabilitation Centre

**Overview**

As part of your task, you will work with a dataset representing patient visits at Heather Glen Rehabilitation Centre. Each visit is recorded as an individual JSON file, organised in a folder structure by year and month (data/<year>/<month>/). Each JSON file contains the details of a single visit, including patient information, diagnosis, treatment, and session outcomes. The data provides an opportunity to analyse trends in patient care, service usage, and treatment effectiveness.

**JSON Schema**

Each JSON file will follow a consistent schema. Below is a list of attributes, with details on expected values or constraints for each.

* **patient\_id**: *(String)*
  + Unique identifier for each patient, prefixed with "HG" followed by digits (e.g., "HG12345").
* **visit\_date**: *(String, Date in ISO format)*
  + The date of the visit, formatted as YYYY-MM-DD (e.g., "2023-06-15").
* **department**: *(String, Enum)*
  + The department providing the treatment. Possible values are:
    - "Physiotherapy"
    - "Occupational Therapy"
    - "Mental Health Counselling"
    - "Post-Surgery Rehabilitation"
* **therapist\_id**: *(String)*
  + Unique identifier for the therapist who conducted the session, prefixed with "T" followed by digits (e.g., "T789").
* **diagnosis**: *(String)*
  + Primary diagnosis or reason for the visit (e.g., "Lower back pain", "Wrist sprain", "Anxiety").
* **treatment**: *(String)*
  + Type of treatment administered (e.g., "Manual therapy", "Cognitive Behavioral Therapy", "Strength training").
* **treatment\_duration**: *(Integer, Minutes)*
  + Duration of the treatment in minutes (e.g., 45). This field may be missing in some records.
* **outcome**: *(String, Enum)*
  + Outcome of the session. Possible values are:
    - "Improved"
    - "No Change"
    - "Worsened"
  + This field may be missing in some records, this should then be assumed to mean “No Change”.
* **follow\_up\_needed**: *(Boolean)*
  + Indicates if a follow-up appointment is recommended (true or false).
* **additional\_notes**: *(String, Optional)*
  + Optional field with additional notes from the therapist (e.g., "Patient reported 50% pain reduction").

Assessment and Marking Criteria

Your submission for the Heather Glen Rehabilitation Centre project will be assessed across three core components to ensure a comprehensive evaluation of your script's functionality, output quality, and code clarity. The grading structure is designed to reward not only correctness but also creativity, efficiency, and robust handling of real-world data complexities.

**Assessment Components**

**A) Automatic Testing on Unseen Datasets**

Your script will be tested against three unseen datasets to assess its accuracy, robustness, and error-handling capabilities:

1. **Dataset 1**: A clean dataset without errors or anomalies, testing basic functionality and correct aggregation.
2. **Dataset 2**: A dataset with anticipated errors or anomalies, such as missing fields (e.g., treatment\_duration, outcome). This tests your script’s ability to handle optional fields gracefully and produce consistent results.
3. **Dataset 3**: A larger dataset with additional anomalies and unexpected data issues beyond what is specified in the JSON schema. This will assess how well your code performs under less predictable conditions, including handling unexpected or malformed data gracefully without crashing.

To pass this component, your script must:

* Process Dataset 1 and Dataset 2 without errors.
* Correctly handle the specified anomalies in Dataset 2.
* Manage larger data volumes and additional errors in Dataset 3, with higher marks awarded for scripts that demonstrate strong error-handling and resilience.

**B) Quality and Completeness of Outputs**

The outputs generated by your script will be inspected to evaluate the accuracy, structure, and usefulness of the data presented. This includes both the JSON report and the text summary for the client’s newsletter:

* **JSON Report**: Should strictly follow the structure provided, with accurate, meaningful aggregations and trends, even in the presence of incomplete or varied data.
* **Text Summary**: Should be clear, informative, and well-organised, with at least two paragraphs summarising key statistics and insights in a way that would be accessible to a non-technical audience. Higher marks will be awarded for summaries that demonstrate thoughtful analysis and clear communication of trends and insights.

To achieve a passing mark, your script must generate basic, accurate outputs for the first two datasets. Higher marks will be awarded for outputs that go beyond basic requirements by incorporating thoughtful analysis and additional insights, particularly for the larger, more complex dataset.

**C) Code Clarity and Completeness**

Your code will be reviewed for readability, organisation, and adherence to Python best practices. This includes:

* **Clarity**: Clear naming conventions, modular structure, and logical flow, allowing someone with Python knowledge to easily understand and verify your code.
* **Documentation**: Reasonable inline comments, function docstrings, and overall documentation where necessary, especially where code complexity increases.
* **Error Handling**: Graceful handling of missing fields and unexpected data issues, with code that fails safely rather than crashes, even with unexpected errors.

Higher marks will be awarded for code that demonstrates both completeness and cleverness, such as efficient handling of large data volumes or creative error-handling mechanisms that account for a wide range of potential issues.

**Grading Categories**

* **Fail**: Does not pass the automatic testing on Dataset 1 or Dataset 2, or fails to generate basic JSON and text outputs.
* **3rd Class (Pass)**: Successfully passes automatic testing on Dataset 1 and Dataset 2, with a rudimentary JSON report and text summary output.
* **2:2 (Lower Second)**: Successfully processes Dataset 1 and Dataset 2 with correct outputs, and provides a well-structured JSON and text output with basic data aggregation and summaries.
* **2:1 (Upper Second)**: Passes automatic testing on all three datasets, with thorough JSON and text outputs that are accurate, clear, and contain well-presented data insights. Code is clear, well-documented, and handles expected anomalies gracefully.
* **1st (Distinction)**: Demonstrates a high level of distinction, cleverness, and creativity. Passes all automatic tests, including with Dataset 3, with outputs that showcase insightful analysis, well-presented trends, and a polished text summary. Code is exceptionally clear, concise, and demonstrates robust error handling, showing readiness for real-world data complexities.