Problem Statement 1

Create a Pipeline in Python to read employees' data from a GCS bucket and upload it into BigQuery.
Points to follow:

- Technology Stack:
    - Python
    - Apache Beam
    - Dataflow
    - BigQuery
    - GCS
- Add audit columns - created_time, modified_time (current timestamp).
- Columns should map as integer, string, date, and float datatypes based on the input data.
- Use a unique identifier for naming the dataflow job.
- Read table schema from a JSON file and parse CSV according to the schema given, without using the headers.

Problem Statement 2

Create a Pipeline in python to read employees' data from a GCS bucket and load it into two different target tables in BigQuery.
Points to follow:

- Technology Stack:
    - Python
    - Apache Beam
    - Dataflow
    - BigQuery
    - GCS
- Read data from a common input CSV and output to two different BigQuery tables with different schemas.
- The two tables to be created are ‘employee’ and ‘employee_personal_info’.
- Table ‘employee’ would have the following columns:
    - Emp_ID
    - Name
    - Gender
    - Email
    - Date_of_Birth
    - Date_of_Joining
    - Age_in_Company
    - Salary
    - Created_Time
    - Modified_Time

- Table ‘employee_personal_info’ would have the following columns:
    - ID
    - Emp_ID
    - Father_Name
    - Mother_Name
    - Age_in_Yrs
    - Weight_in_Kgs
    - Phone_No
    - State
    - Zip
    - Region
    - Created_Time
    - Modified_Time

- Add audit columns - created_time, modified_time (current timestamp).
- Columns should map as integer, string, date, and float datatypes based on the input data.
- Use a unique identifier for naming the dataflow job.
- Read table schemas from JSON file.
- Compute values of some columns dynamically:
    - Calculate ‘Age_in_Yrs’ using ‘Date of Birth’ in the input data.
    - Concatenate ‘First Name’ and ‘Last Name’ to generate column ‘Name’.
    - Generate a unique ‘ID’ column for table ‘employee_personal_info’.