Problem Statement

A system (TCU) collects data and transmits data to the data warehouse. This data is read by another module named as data processor. Data processor reads the collected data and distributes the data to multiple modules. Our task is to test the functionality of the data processor and validate if data is processed as per the requirements.

# System Overview

TCU

Data Warehouse

Data Processor

Customer 1

TEST

Customer 2

ODI

# Input Data

* TCU sends data in .csv format for all the players who played between 1990 – 2000.
* TCU sends data in .json format for all the players who played from 2000 onwards.

\*Data is available in inputDataSet directory.

# Processing Logic for Data Processor Module:

* The Data Processor will read each entry/event and will create a corresponding entry/event at the output.
* If the event type is ODI, data will be sent to Customer 2 as odi\_results.csv.
* If the event type is Test data will be sent to Customer 1 test as test\_results.csv.
* The Data Processor will create a new col in the data named “Player Type”.
  + If a player has made more than 500 runs and taken 50+ wickets, he will be called an “All-Rounder”.
  + If a player has made more than 500 runs and taken less than 50 wickets, he will be called a “Batsman”.
  + If a player has less than 500 runs, he will be called a “Bowler”.
* If any player doesn’t have any data for runs and wickets, this player will be removed from the output data.
* If the age of a player is > 50 and < 15, this player will also be removed from the output.
* There shouldn’t be any extra entry in the output.
* There shouldn’t be any missing entry in the output.

\*Output data is available in outputDataSet directory.

# Functional Requirements (Mandatory)

* TCU will store data in .csv or .json format. Create a python module to read the .csv & .json files.
* Read the data from both the sources, merge them and store them in a temp folder.
* Create a new parameter “Player Type” and update the col values as mentioned above.
* Validate if the Data Processor’s output is correct as per the processing logic.
* Create a new col as “Result”.
  + “PASS”: if the output from step 3 matches with the data processor’s output for a given player
  + “FAIL” if the output from step 3 doesn’t match with the data processor’s output for a given player.
* Store the results as “test\_result.csv”.

# Bonus Requirements (Optional)

* Code should follow pep-8 coding standard and code should have 8+ score.
* Use any of the linting modules to format the code.
* Create requirements.txt file to maintain dependencies.
* Use pytest framework to run the test. Use pytest-html or Azure report.
* Write another test to check the schema. Output data should have the following parameters:
  + eventType: string
  + playerName: string
  + age: int
  + runs: int
  + wickets: int
  + playerType: string

# How to submit the code

* Create a git repository and publish your code and share the repo link with the team.
* **Bonus**: create a readMe.md file in the repo and add a description of your assignment along with one example of data processing.
* **Bonus**: add one action/workflow to your git repository.
* **Bonus**: create a pipeline to run the test cases

For any queries reach out to the team.