

System Requirements

Specification Index

For

Create a Pandas DataFrame from a Dictionary and Perform Indexing Operations

(Topic: Pandas Data Structures)

Version 1.0

Employee Data Analysis Console

Project Abstract

The Employee Data Analysis Console is a Python application developed to handle and analyze employee data stored in tabular form. The application takes in employee records and allows for various operations such as displaying the first few records, retrieving employee information based on specific criteria, filtering employees by age, and saving the processed data to CSV. This system is essential for handling large employee datasets, streamlining data retrieval, and providing efficient insights into the workforce. The goal is to provide HR teams and data analysts with a versatile tool for managing employee records and conducting basic analyses on employee attributes.

Business Requirements

- Efficient handling of employee data stored in tabular format.
- Provide functionality to display, filter, and save employee records.
- Allow HR teams to quickly access employee details based on unique identifiers or attributes like age.
- Provide outputs in a standard format, which can be exported for further use.

Constraints

Input Requirements

- Employee Data:
 - Must be provided as a list of dictionaries or other appropriate data structures.
 - Each dictionary must contain keys: **Employee ID**, **Name**, **Age**, and any other relevant attributes.
 - Example: `{"Employee ID": 1, "Name": "John Doe", "Age":`

Output Requirements

- **Display Data:**
 - The output should be presented in tabular form (first 5 rows of data).
 - Must allow for filtering and displaying specific employee data based on input parameters.
- **Save to CSV:**
 - The DataFrame must be saved to a CSV file with the employee data in the format: "Employee ID", "Name", "Age", ...

Operations

The system should allow for the following operations:

1. **Displaying Head of Data:**
 - Display the first 5 rows of the employee dataset.
 - Use case: User wants to quickly view a sample of the dataset for review.
2. **Dataframe Info:**
 - Provide basic information about the dataset, including column names and data types.
 - Use case: User needs to check the structure of the dataset to understand its columns and types before performing analysis.
3. **Get Employee by ID:**
 - Retrieve the details of an employee based on a unique employee ID.
 - Use case: HR manager needs to retrieve specific details of an employee using their Employee ID.
4. **Get Employees Above Age:**
 - Filter the dataset to show employees whose age is greater than a

specified value.

- Use case: User wants to analyze employees older than a specific age threshold (e.g., retirement age).

5. Save Data to CSV:

- Save the current employee data in a CSV file.
- Use case: User wants to export the employee dataset for further use, reporting, or analysis.

Output Format

The output should be presented in the following formats:

- **Display Format:**
Show the first 5 rows of the employee data in tabular form.

Template Code Structure

1. Initialization Section:

- Initialize DataFrame with input employee data.

2. Display Section:

- Display the first 5 rows of the data.
- Show DataFrame information including column names and data types.

3. Filter Section:

- Filter employees by employee ID.
- Filter employees by age.

4. Output Section:

- Save the current employee data to a CSV file.
- Print results in tabular format.

Execution Steps to Follow:

- All actions like build, compile, running application, running test cases will be through Command Terminal.
- To open the command terminal the test takers, need to go to Application menu (Three horizontal lines at left top) -> Terminal -> New Terminal
- This editor Auto Saves the code
- If you want to exit(logout) and continue the coding later anytime (using Save & Exit option on Assessment Landing Page) then you need to use **CTRL+Shift+B** -command compulsorily on code IDE. This will push or save the updated contents in the internal git/repository. Else the code will not be available in the next login.
- These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.
- To setup environment:
You can run the application without importing any packages
- To launch application:
python3 mainclass.py
- To run Test cases:
python3 -m unittest
- Before Final Submission also, you need to use **CTRL+Shift+B** - command compulsorily on code IDE, before final submission as well. This will push or save the updated contents in the internal git/repository, and will be used to evaluate the code quality.

Screen shot to run the program

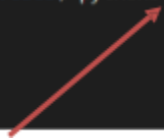
To run the application

```
OK
coder@dighe20250227t070305rz1fj5p3:/home/myproject/dighegmailcom_20250227T070305$ python3 <<scriptname>>.py
```



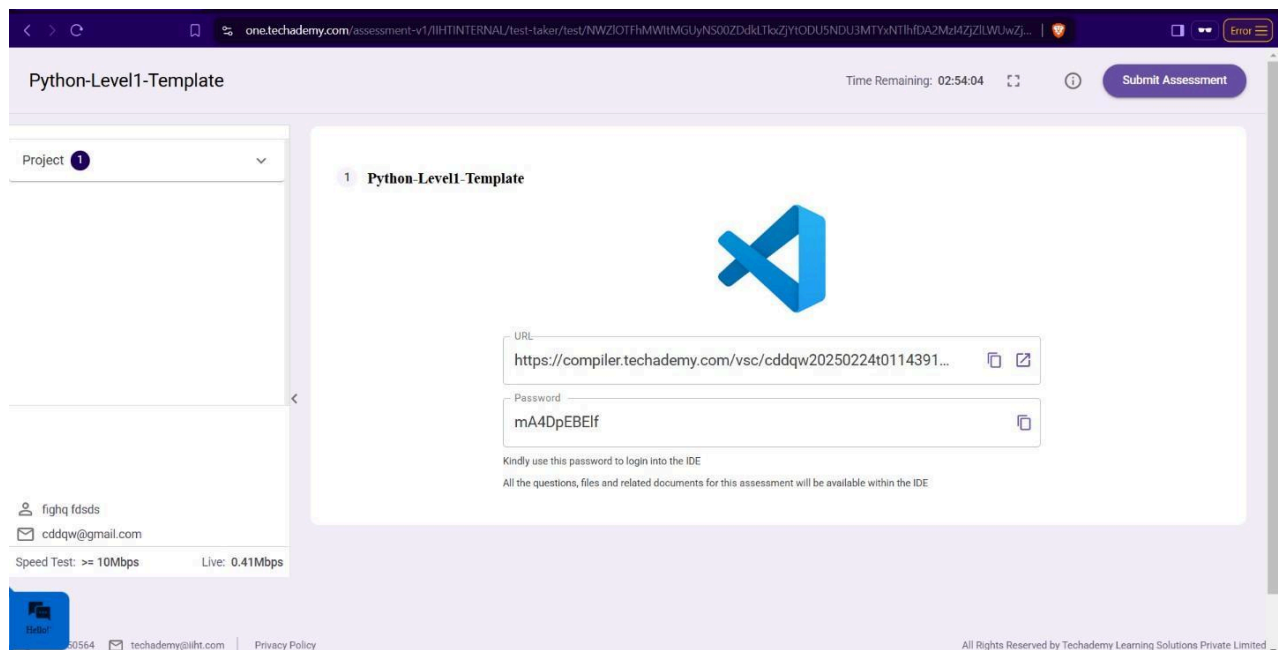
python3 mainclass.py

```
● coder@dighe20250227t070305rz1fj5p3:/home/myproject/dighegmailcom_20250227T070305$ python3 -m unittest
TestBoundary = Passed
.TestExceptional = Passed
.TestCalculateTotalDonations = Failed
.TestCalculateTotalStockValue = Failed
.TestCheckFrankWhiteDonated = Failed
```



To run the testcase

python3 -m unittest



- **Once you are done with development and ready with submission, you may navigate to the previous tab and submit the workspace. It is mandatory to click on “Submit Assessment” after you are done with code.**