

System Requirements

Specification Index

For

Exporting Employee Data to Excel for Financial Analysis

(Topic: Data Import and Export)

Version 1.0

Employee Data Analysis Console

Project Abstract

The Employee Data Analysis Console is a Python-based application that helps organizations analyze and manage employee data. This system loads employee data from a CSV file, performs essential operations such as displaying the top rows, analyzing the structure of the data, identifying the employee with the highest age, and exporting the data into an Excel file. The application ensures that the employee data is presented in a structured and accessible manner, allowing organizations to easily track and analyze important employee information.

Business Requirements:

1. Display Head of Data:

- **The system must display the first five rows of employee data from the CSV file upon request.**

2. Data Information:

- **The system must provide information about the data structure, including column names and data types.**

3. Find Highest Age Employee:

- **The system must be capable of identifying the employee with the highest age by analyzing the 'Age' column in the CSV file.**

4. Export to Excel:

- **The system must provide an option to save the employee data to an Excel file. The user must be able to specify the file name or use a default name.**

5. Verify Saved Excel File:

- **The system must be able to verify whether the Excel file was saved successfully by reloading it and ensuring it's not empty.**

Constraints:

Input Requirements

- The CSV file must contain the following columns:
 - Employee ID: A unique identifier for each employee.
 - Name: The name of the employee.
 - Age: The age of the employee, which must be a positive integer.
 - Other columns as required by the organization.

Output Requirements

- Data Display:
 - The system should display the first five rows of data when requested.
- Data Information:
 - The system must output the column names and data types for the DataFrame when requested.
- Highest Age Employee:
 - The system must identify the employee with the highest age and display the corresponding Employee ID.
- Export to Excel:
 - The system must save the data into an Excel file with the option to specify the output file name. If no name is provided, the default name should be **employee_data.xlsx**.
- File Verification:
 - The system must verify if the exported Excel file is saved correctly by loading it back and checking its content.

Required Output Format:

1. Display Head:

- Show the first five rows of the DataFrame.

2. Data Information:

- Show column names and data types.

3. Highest Age Employee:

- Show the Employee ID of the employee with the highest age.

4. Export to Excel:

- Ensure the data is saved into an Excel file with the option to specify a file name.

Template Code Structure:

1. File Handling Functions:

- `load_csv_file()`
- `save_to_excel()`
- `verify_excel_saved()`

2. Data Display Functions:

- `display_head()`
- `dataframe_info()`
- `highest_age_employee()`

3. Input Section:

- Load employee data from a specified CSV file.

4. Output Section:

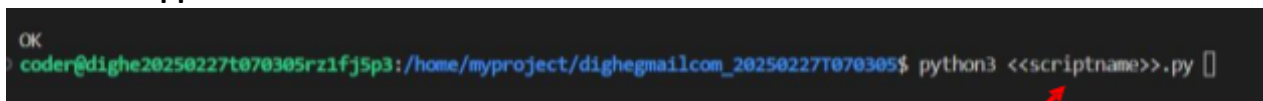
- **Display the first few rows of data.**
- **Provide details of the data (column names and data types).**
- **Display the employee with the highest age.**
- **Export the data to an Excel file.**

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
```

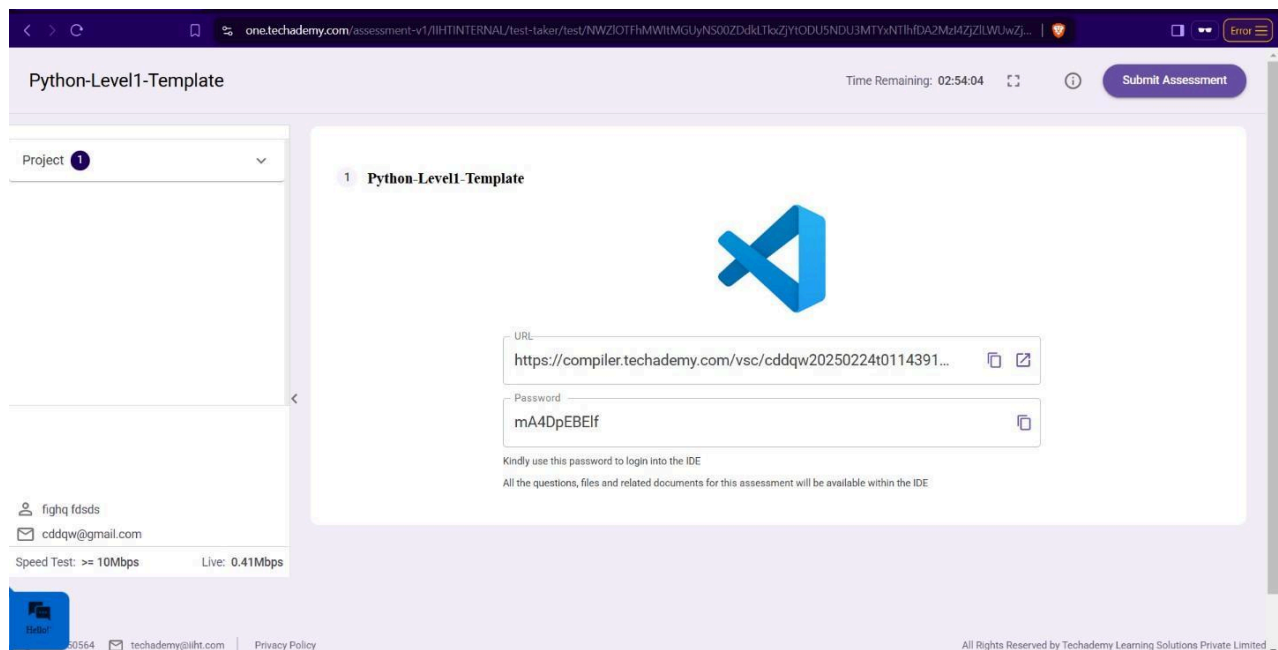
python3 mainclass.py

```
● 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.**