

# **System Requirements**

## **Specification Index**

**For**

### **Employee Data Management and Analysis**

**(Topic: Data Selection and Indexing )**

**Version 1.0**

# Employee Data Analysis Console

## Project Abstract

The Employee Data Analysis Console is a Python-based console application designed to analyze employee data from CSV files. This system provides various functionalities such as retrieving columns based on labels or positions, displaying the first few rows of the dataset, and handling missing data by filling in the missing values with the column mean. Additionally, it allows users to export the updated data back to a new CSV file. The system aims to streamline employee data management, making it easier to manipulate and clean the data for further analysis or reporting.

## Business Requirements:

- **Data Handling:** The system must handle large datasets efficiently.
- **Missing Data Management:** The system should offer an automated way to handle missing values.
- **Data Exporting:** It should allow the user to export modified datasets to CSV format.
- **Column Retrieval:** The system should support both label-based and position-based column retrieval.
- **Information Display:** The system should be able to display the first few rows of the data and provide column-wise data type information.

## Constraints

## Input Requirements

- **CSV File Path:**
  - Must be stored as a string in the variable `file_path`.
  - Must be a valid file path to a CSV file.
  - The file should contain structured tabular data with rows and columns.

## Data Handling Constraints:

- Missing values in numeric columns should be handled by replacing them with the column mean.
- Numeric columns are identified by the `float64` or `int64` data types in Pandas.

## Output Constraints

### 1. Display Format:

- The first 5 rows of the DataFrame must be displayed when `display_head()` is called.
- The column names and data types of the DataFrame should be displayed when `dataframe_info()` is called.

### 2. Export Format:

- The system should save the updated DataFrame to a CSV file when `export_updated_csv()` is called.
- The updated CSV file should be saved with the name "updated\_employee\_data.csv" by default or with a user-specified name.

## Template Code Structure:

### 1. Class Definition:

- `EmployeeDataAnalysis` class that will hold the main logic.

### 2. Functions:

- Data Loading Functions:
  - `__init__(self, file_path)`: Loads the CSV file into a Pandas DataFrame.
- Data Retrieval Functions:

- `retrieve_columns_label_based(self, column_name)`: Retrieves columns based on the column label.
  - `retrieve_columns_position_based(self, column_index)`: Retrieves columns based on the column index.
- Display Functions:
  - `display_head(self)`: Displays the first 5 rows of the DataFrame.
  - `dataframe_info(self)`: Displays the column names and data types of the DataFrame.
- Data Cleaning Functions:
  - `handle_missing_values(self)`: Replaces missing values with the column mean for numeric columns.
- Export Functions:
  - `export_updated_csv(self, output_file="updated_employee_data.csv")`: Exports the updated DataFrame to a CSV file.

### 3. Input Section:

- Get file path for the CSV file.

### 4. Processing Section:

- Perform operations like retrieving columns, handling missing values, and exporting the data.

### 5. Output Section:

- Display the first 5 rows of the data and column-wise data type information.
- Export the cleaned data to a new CSV 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

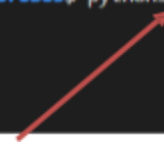
To run the application

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



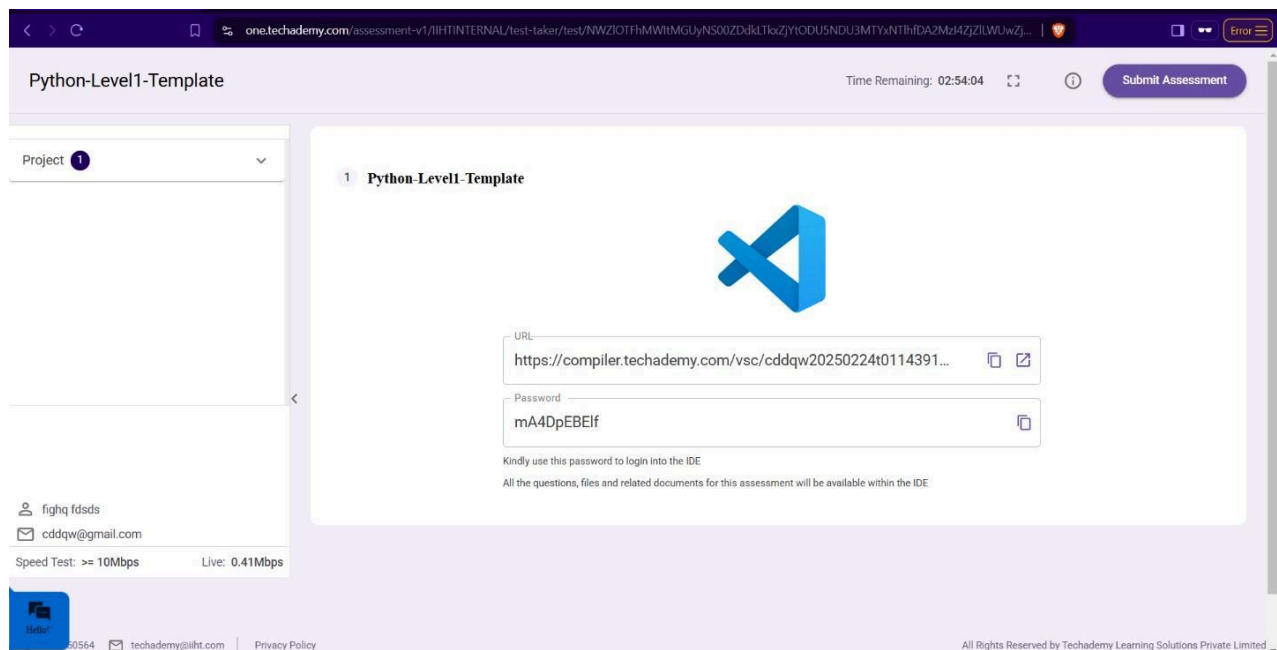
**python3 mainclass.py**

```
● 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.