

# **System Requirements**

## **Specification Index**

**For**

### **Normalize Numerical Data in a DataFrame Using `apply()` or Vectorized Operations**

**(Topic: Advanced Pandas )**

**Version 1.0**

# Sales Data Analysis Console

## Project Abstract

The Sales Data Analysis Console is a Python application developed to manage and process sales data stored in CSV format. This system helps with loading, normalizing, and analyzing sales data for products. The application ensures that sales information, such as units sold and price, is transformed into a consistent format through Min-Max scaling. The goal is to generate useful insights, normalize the data, and allow for further analysis or reporting. The system is crucial for business analysts to derive trends and make data-driven decisions on sales performance.

## Business Requirements:

The system will allow users to:

- Load sales data from a CSV file.
- Display the first few rows of the sales data.
- Display detailed information about the dataset such as column names and data types.
- Normalize specific columns ('Units Sold' and 'Price') using Min-Max scaling for better analysis.
- Export the normalized data into a new CSV file for further processing or reporting.

---

## Constraints

### Input Requirements:

- CSV File Input:
  - The file must be in CSV format.
  - The file must contain columns including, but not limited to, **Units\_Sold** and **Price**.

- Example file path: **sales\_data.csv**
- **Data Types:**
  - **Units\_Sold:** Must be a numeric value (int or float).
  - **Price:** Must be a numeric value (float).
- **File Path:**
  - The path to the CSV file must be valid and accessible.

### Conversion Constraints

- **Normalization:**
  - For the **Units\_Sold** and **Price** columns, the Min-Max scaling will be applied to normalize the data.
  - The result will be stored in new columns **Normalized\_Units\_Sold** and **Normalized\_Price**.
- **Export:**
  - The system will export the updated CSV file containing normalized data to the specified location.

---

### Output Constraints

1. **Display Output:**
  - The first 5 rows of the sales data will be displayed for quick inspection of the data.
2. **Detailed Data Info:**
  - The system will output the DataFrame's column names and data types.
3. **Normalized Data:**

- The system will display the updated DataFrame containing the normalized **Units\_Sold** and **Price** columns.

#### 4. File Export:

- The system will save the normalized data to a new CSV file.
- 

### Template Code Structure

#### 1. Data Loading Functions:

- **load\_data\_from\_csv():**
  - Load CSV data into a Pandas DataFrame.

#### 2. Data Inspection:

- **display\_head():**
  - Display the first few rows of the DataFrame for preview.
- **dataframe\_info():**
  - Provide details about the columns and data types.

#### 3. Normalization Section:

- **normalize\_data():**
  - Normalize the **Units\_Sold** and **Price** columns using Min-Max scaling.

#### 4. Export Section:

- **export\_updated\_csv():**
  - Save the normalized data into a new CSV file for further use.

### **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

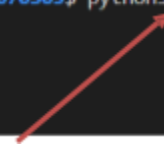
To run the application

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



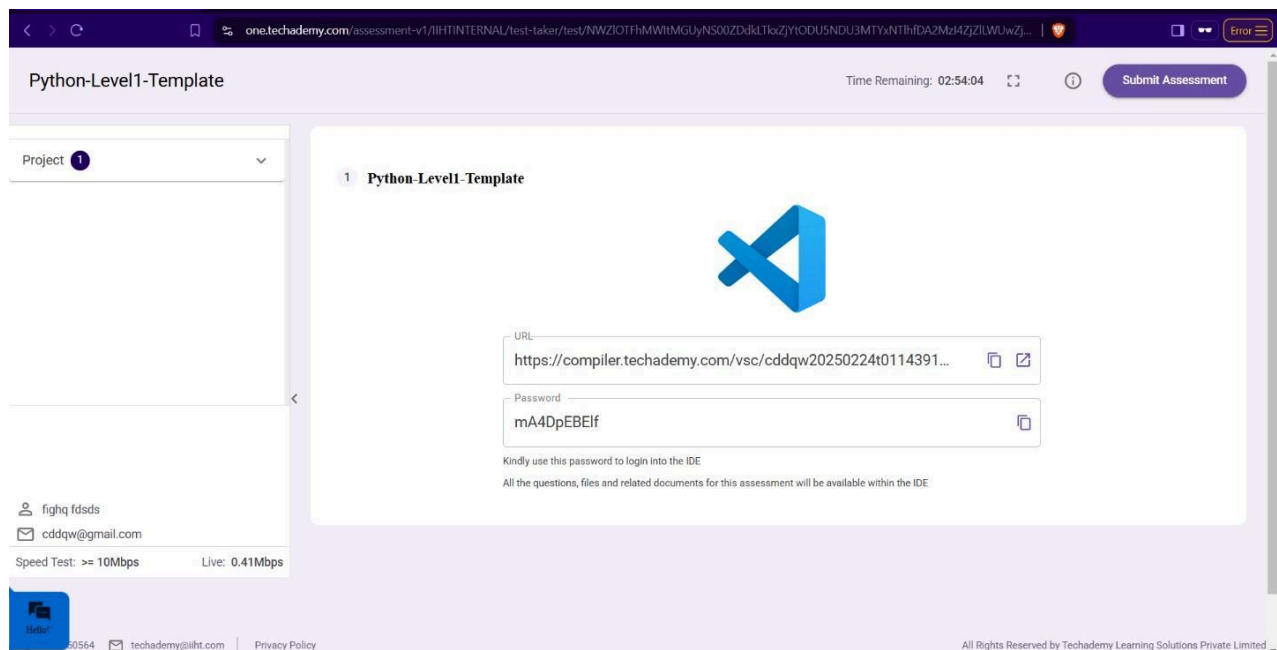
**python3 mainclass.py**

```
● 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.**