

System Requirements

Specification Index

For

Pivot a DataFrame to Summarize Data and Compute Multiple Aggregate Functions

(Topic: Advanced Pandas)

Version 1.0

Sales Data Analysis Console

Project Abstract

The Sales Data Analysis Console is a Python-based application developed to provide comprehensive insights into sales data. This system processes sales data stored in a CSV file, allowing the user to perform various operations like displaying the first few rows, retrieving data frame information, pivoting data, calculating total revenue, and exporting the summarized data. The main objective of this system is to provide efficient ways to analyze sales data and create summary reports that facilitate business decision-making. The application uses the powerful Pandas library to handle and manipulate the data.

Business Requirements

1. Data Loading:

- The system must load sales data from a CSV file.
- The system should support any standard CSV file containing columns like **Category**, **Units_Sold**, **Price**, and **Product_ID**.

2. Display Data:

- The system should provide a way to display the first five rows of the dataset for quick inspection.

3. Data Inspection:

- The system must provide information about the columns and data types of the dataset.

4. Data Pivoting and Summarizing:

- The system should allow pivoting of data to compute multiple aggregate functions like sum and mean.
- The system must calculate total revenue by multiplying units sold by the mean price.
- The system should calculate the product count by determining the number of unique products within each category.

5. Data Export:

- The system must save the summarized sales data into a new CSV file.

Constraints

Input Requirements

1. CSV File:

- The input data should be stored in a CSV file.
- The file must contain columns: **Category**, **Units_Sold**, **Price**, and **Product_ID**.

2. Output File:

- The system must export the summarized data to a new CSV file with a default name of **summary_sales_data.csv** or a custom name provided by the user.

Processing Constraints

1. Data Manipulation:

- The system should load the CSV file into a Pandas DataFrame.
- The DataFrame must be processed to summarize sales by category.
- The system should handle missing or invalid data gracefully (e.g., empty fields, incorrect data types).

2. Pivoting and Aggregation:

- The system must use Pandas pivot functionality to summarize data.
- The pivot operation should calculate:
 - Total units sold per category.
 - Total price and average price per category.
 - Total revenue per category (calculated by multiplying total units sold by average price).

- The number of unique products per category.

3. Output Constraints:

- The system must format the output CSV file correctly, ensuring that it can be opened and read by other applications (e.g., Excel).

Required Output Format

1. Summarized CSV Output:

- The summarized CSV file should contain the following columns:
 - **Category**: The product category.
 - **Units_Sold**: The total units sold within each category.
 - **Price**: The total and mean price for products within each category.
 - **Total_Revenue**: The calculated total revenue for each category.
 - **Product_Count**: The count of unique products within each category.

Template Code Structure

1. Class: SalesAnalysis

- **Methods**:
 - **`__init__(self, file_path)`**: Loads CSV data into a Pandas DataFrame.
 - **`display_head(self)`**: Displays the first 5 rows of the DataFrame.
 - **`dataframe_info(self)`**: Displays information about DataFrame columns and data types.
 - **`pivot_and_summarize(self)`**: Pivots the DataFrame and calculates total revenue and product count.

- `export_updated_csv(self, output_file="summary_sales_data.csv")`: Exports the summarized data to a CSV file.

2. Data Input Section:

- Load the CSV file provided by the user.

3. Data Processing Section:

- Perform pivoting and aggregation on the sales data.
- Calculate total revenue and product counts.

4. Data Export Section:

- Save the processed data into a new CSV file for further use or reporting.

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
```

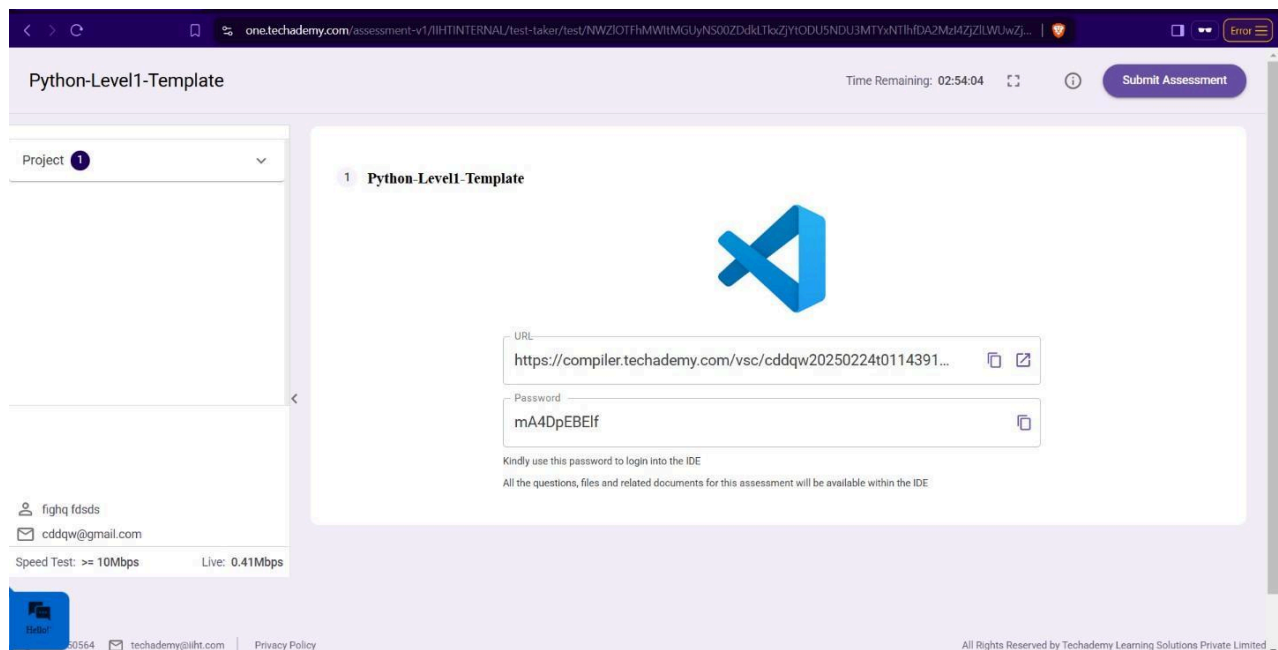
python3 mainclass.py

```
● 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.**