# Data Cleaning CLI Tool – Project Implementation & Testing Guide

## 1. Overview

The Data Cleaning CLI Tool is a Python-based command-line application designed to automate data preprocessing tasks such as removing duplicates, filling missing values, trimming spaces, and validating data before storing it in a MySQL database. It also supports CSV import/export for easy integration with other tools used by engineers and analysts.

## 2. Project Components

The project consists of three main Python files:

• cleaner.py – Handles all data cleaning operations using Pandas.

• db\_handler.py – Manages database connections and uploads cleaned data to MySQL.

• main.py – Acts as the CLI entry point, integrating argparse for command-line use.

## 3. Tools & Libraries Used

• Python 3.13  
• pandas  
• mysql-connector-python  
• argparse  
• MySQL Workbench or phpMyAdmin for database setup

## 4. Steps to Test the Project

Follow these steps to properly test and verify the tool’s functionality:

1. Step 1: Prepare a sample CSV file

Create a file named 'sample\_data.csv' with the following sample content:  
  
name,age,email,salary  
John,25,john@example.com,50000  
Alice,30,alice@example.com,60000  
Bob,,bob@example.com,45000  
John,25,john@example.com,50000  
Charlie,35,,70000

1. Step 2: Run the CLI tool

Use the command line to execute the script:  
  
python main.py --input sample\_data.csv --output cleaned\_data.csv

1. Step 3: Check console output

The terminal should show logs confirming data cleaning steps, MySQL upload success, and output file generation.

1. Step 4: Verify cleaned CSV output

Open the 'cleaned\_data.csv' file to confirm that missing values have been filled, duplicates removed, and data standardized.

1. Step 5: Verify MySQL upload

Open MySQL Workbench and run:  
  
USE data\_cleaner\_db;  
SELECT \* FROM cleaned\_data;  
  
This should show the cleaned dataset successfully stored in the database.

## 5. Problems Faced During Development

While building the project, several issues were encountered and resolved:

• Deprecation Warning: Pandas’ ‘fillna(method=...)’ is now deprecated. Fixed by using ‘ffill()’.

• SQL Syntax Error: MySQL does not support 'ADD COLUMN IF NOT EXISTS' directly. Fixed by modifying the logic to check column existence before altering the table.

• Database Error: 'Database exists' error occurred when recreating the database. Fixed by skipping database creation if it already exists.

These issues helped in understanding real-world debugging scenarios, improving SQL query handling, and writing more stable, production-ready Python code.

## 6. Outcome

After successful testing, the CLI tool was able to:  
• Clean and validate data effectively.  
• Store results securely in a MySQL database.  
• Export cleaned data into a new CSV file.  
  
This project demonstrates practical knowledge of Python, Pandas, MySQL, and command-line interface design.