# Data Cleaning CLI Tool – Implementation and Execution Guide

## 1. Project Overview

The Data Cleaning CLI Tool is a Python command-line application designed to automate data cleaning processes. It reads raw CSV files, removes duplicates, fills missing values, trims extra spaces, and standardizes the data. The cleaned data can be exported as a CSV file or uploaded to a MySQL database.

## 2. Project Components

The project consists of the following main components:  
• cleaner.py – Contains functions for cleaning data using pandas.  
• db\_handler.py – Handles MySQL database connections and uploads cleaned data, creating tables dynamically.  
• exporter.py – Exports cleaned data to CSV files.  
• main.py – Command-line interface for the tool using argparse.  
• config.py – Stores database configuration credentials.

## 3. Tools and Libraries

• Python 3.13  
• pandas  
• mysql-connector-python  
• argparse  
• MySQL Workbench for database management

## 4. Implementation Steps

1. Set up the Python environment and install required libraries.  
2. Prepare a sample CSV file with intentional issues such as duplicates, missing values, and extra spaces.  
3. Implement cleaner.py to handle data cleaning operations (duplicates removal, missing value handling, trimming, standardization).  
4. Implement db\_handler.py to connect to MySQL and dynamically create tables and columns based on the CSV data.  
5. Implement exporter.py to export the cleaned data to CSV.  
6. Integrate all modules in main.py with argparse to allow CLI commands for cleaning, exporting, and uploading.  
7. Store database credentials securely in config.py.

## 5. Execution Steps

Follow these steps to execute the project:  
1. Place your input CSV file in the project directory.  
2. Run the CLI tool for different scenarios:  
 • Cleaning only: python main.py --input sample\_data.csv  
 • Cleaning + Export: python main.py --input sample\_data.csv --output cleaned\_data.csv  
 • Cleaning + Upload: python main.py --input sample\_data.csv --upload  
 • Full Workflow: python main.py --input sample\_data.csv --output cleaned\_data.csv --upload  
3. Check terminal messages for success or error logs.  
4. Verify exported CSV files in the project folder.  
5. Verify uploaded data in MySQL using:  
 USE data\_cleaner\_db;  
SELECT \* FROM cleaned\_data;

## 6. Problems Faced and Solutions

• Pandas fillna deprecation warning – fixed using 'ffill()'.  
• MySQL syntax errors when adding columns – fixed by checking INFORMATION\_SCHEMA before adding columns.  
• Access denied errors for MySQL – fixed by verifying credentials and updating config.py.  
• Database already exists – handled with CREATE DATABASE IF NOT EXISTS.  
• Dynamic column handling – implemented logic to create missing columns automatically based on CSV.

## 7. Testing and Verification

Testing was performed in multiple steps:  
1. Cleaning only – verified cleaned data in terminal.  
2. Cleaning + Export – verified cleaned CSV output.  
3. Cleaning + Upload – verified MySQL table and data insertion.  
4. Full workflow – verified both CSV export and MySQL upload.  
5. Negative testing – verified handling of wrong filenames, empty CSV, and invalid credentials.

## 8. Outcome

The project successfully:  
• Cleansed raw CSV data effectively.  
• Removes duplicates and handles missing values.  
• Uploads data dynamically to MySQL with automatic table/column creation.  
• Exports cleaned data to CSV.  
• Handles errors gracefully.  
This project demonstrates practical skills in Python programming, data cleaning with pandas, MySQL database operations, and building CLI tools.