# Data Cleaning CLI Tool – Project Report

## 1. Overview

The Data Cleaning CLI Tool is a Python command-line application designed to clean and preprocess data from CSV files. It helps remove duplicates, fill missing values, trim spaces, and standardize the data. After cleaning, the data can be uploaded to a MySQL database or exported as a new CSV file.

## 2. Project Purpose

This project aims to automate data preprocessing tasks that engineers and analysts perform manually. It saves time, ensures data quality, and makes it easy to store and analyze data in MySQL databases.

## 3. Tools & Libraries Used

• Python 3.13  
• pandas (for data cleaning)  
• mysql-connector-python (to connect Python with MySQL)  
• argparse (to handle command-line arguments)  
• MySQL Workbench (for managing the database)

## 4. Steps We Followed

1. Set up the Python environment and installed all necessary libraries.  
2. Prepared a sample CSV file with intentional issues like duplicates and missing values.  
3. Wrote cleaner.py to handle data cleaning tasks (removing duplicates, filling missing values, trimming spaces).  
4. Wrote db\_handler.py to upload cleaned data to MySQL database, creating tables dynamically if they did not exist.  
5. Updated main.py to integrate CLI functionality using argparse.  
6. Tested the tool in multiple steps: cleaning only, cleaning + exporting CSV, cleaning + uploading to MySQL, full workflow.  
7. Fixed issues along the way such as Pandas deprecation warnings, MySQL syntax errors, and MySQL authentication errors.

## 5. Problems We Faced and Solved

• Pandas deprecation warning: 'fillna(method=...)' is deprecated, fixed by using 'ffill()'.  
• MySQL syntax errors: 'ADD COLUMN IF NOT EXISTS' not supported, fixed by checking columns in INFORMATION\_SCHEMA before adding.  
• Access denied for MySQL root user: solved by verifying credentials and updating config.py with correct password.  
• Database already exists errors: handled by using 'CREATE DATABASE IF NOT EXISTS'.  
• Ensuring dynamic columns: wrote Python code to add any missing CSV columns to MySQL table automatically.

## 6. Testing Steps

1. Cleaning only: Run the script with the input CSV and check terminal messages.  
2. Cleaning + Export: Run with output CSV argument and check the exported cleaned\_data.csv file.  
3. Cleaning + Upload: Run with --upload argument and check that data appears in MySQL table.  
4. Full Workflow: Run with both --output and --upload and verify both CSV and database.  
5. Negative Testing: Check with wrong file names, empty CSV, or invalid DB credentials to ensure script handles errors gracefully.

## 7. Outcome

After completing the project and testing:  
• The tool successfully cleans CSV data and handles missing or duplicate values.  
• The cleaned data can be uploaded to MySQL dynamically with columns created as needed.  
• The cleaned data can also be exported as CSV.  
• The script now handles errors like invalid files or database connection issues gracefully.  
• This project demonstrates practical skills in Python, Pandas, MySQL, and CLI application development.