Project Title:

CLI-Based Statistical Calculator Using Python

Submission Date: August 4, 2025
Course Code: CSC202
Department: Computer Science
Institution: Ladoke Akintola University of Technology

TEAM MEMBERS

NAMES	MATRIC NUMBER
Abdulsalam Fareeda Adebimpe	2023002555
Ndudim Jesse	2023008473
Fatiloro Oluwamakinwa Favour	2023003425
Idowu Temitope Samuel	2023005981
Onyendu Emmanuel	2023006872

Executive Summary

This project aims to develop a **Command-Line Interface (CLI) calculator** that performs a wide range of statistical computations using real-world numerical data from CSV files.

The application demonstrates the use of:

- File handling
- Object-Oriented Programming (OOP)
- Error handling
- Statistical analysis
- Data visualization (in text format)
- Hypothesis testing

Problem Statement

In many low-resource environments or CLI-based systems, researchers, students, or analysts lack access to GUI-based tools like Excel or SPSS.

This project solves that problem by:

- Allowing CSV input from any platform
- Providing a menu-driven, lightweight CLI solution
- Returning rich statistical insights directly on the terminal

Project Objectives

- Design a user-friendly CLI tool for statistical analysis
- Implement Python modules for extensible, reusable functions
- Ensure robust error handling and input validation
- Produce interpretative statistical reports
- Apply principles of modularity and unit testing

System Architecture Diagram

System Components Diagram

```
mathematica
CopyEdit
User Input ───────▶ CLI App ───────▶ CSV Reader
                    │              │
                    ▼              ▼
              Stats Module    Correlation Module
                    │              │
                    ▼              ▼
         Hypothesis Testing    Report Writer
                    ▼
             Text-Based Output
                    ▼
          Formatted Reports (.txt)
```

Tools and Technologies

Tool	Purpose
Python 3.10+	Core programming language
pandas	CSV parsing, basic stats
numpy	Math and numeric support
scipy.stats	Advanced statistical tests

unittest	Testing modules
Visual Studio Code / PyCharm	Development

P CSV File Format

`input_data.csv` (sample rows):

StudentID	MathScore	EnglishScore	Gender
S001	78	82	M
S002	85	79	F
S003	69	70	F

Column Descriptions:

- Numeric values analyzed
- Gender supports categorical analysis
- Identifiers are ignored in math logic

Code Structure

File	Responsibility
`cli_calculator.py`	Main menu and CLI
`stats_module.py`	Descriptive statistics
`correlation_module.py`	Pearson analysis
`hypothesis_module.py`	T-tests, Chi-square
`file_handler.py`	Input and validation
`report_writer.py`	Text file generation
`test_all.py`	Unit tests
`analysis_report.txt`	Output example

`input_data.csv`	Sample input data

Descriptive Statistics Module

Highlights:

```
def calculate_mean(data): return np.mean(data)
def calculate_median(data): return np.median(data)
```

Outputs:

- Mean, median, mode
- Standard deviation & variance
- Skewness & kurtosis

Interpretation:

- Skewness near 0 = symmetric
- Kurtosis > 3 = heavy-tailed

Correlation Analysis

```
def compute_pearson_corr(x, y): return stats.pearsonr(x, y)
```

Example:

- Correlation between Math and English scores: **0.88**
- Interpretation: Strong positive linear relationship

Hypothesis Testing

`One-sample t-test:`

```
stats.ttest_1samp(data, popmean)
```

Two-sample t-test:

```
stats.ttest_ind(group1, group2)
```

Chi-square test:

```
stats.chisquare(observed_freq)
```

Real use cases:

- Checking gender bias in exam success
- Testing if a class scored above a national average

Text Histogram and Frequency Distribution

Example:

```
75.0: ███
78.0: █
```

Histogram shows frequency of scores visually using ASCII bars.
Ideal for CLI environments and easy for comparison.

Error Handling and Validation

Handled Errors:

- FileNotFoundError → if CSV path is invalid
- ValueError → non-numeric data in math operations
- ZeroDivisionError → empty datasets

- IndexError → wrong column index

Validation Techniques:

- Confirm numeric columns
- Check for NaNs or empty cells
- Handle duplicate or missing column headers

Testing and Results

`test_all.py` runs unit tests on:

- Mean, median, std dev
- Correlation logic
- T-tests
- File reading and report generation

Sample Output:

```
Ran 12 tests in 0.09s
```

Conclusion and Future Work

This project successfully demonstrates:

- Modular Python programming
- End-to-end CLI interaction
- Practical data analysis
- Solid software engineering principles

Future Enhancements:

- Add GUI version with Tkinter or PyQt
- Support exporting `.csv` and `.pdf` reports
- Integrate confidence intervals and regression