Project Report
On

"STUDENT PERFORMANCE ANALYSIS USING PYTHON"

Submitted to the **Uttaranchal University** in partial fulfilment of the requirements for the award of the Degree of

BACHELOR OF COMPUTER APPLIC ATIONS

Submitted by

Tamash Jyoti Neog
(Learner ID: 2223020165)

Under the Guidance of

Ms. Apoorva S Shekhawat, AI Scientist – II (Artificial Intelligence), Infoedge India Ltd.

UTTARANCHAL UNIVERSITY, DEHRADUN

Acknowledgement

I am deeply grateful to all those who have contributed to the successful completion of this project.
First and foremost, I would like to express my sincere gratitude to my guide, Ms. Apoorva S Shekhawat, for their invaluable guidance, constant support, and encouragement throughout the duration of this project. Their expertise and constructive feedback have significantly contributed to shaping the direction of this research, and I am truly thankful for their patience and insights.
I would also like to extend my heartfelt thanks to **Centre for Distance and Online Education (CDOE), Uttaranchal University** for providing a conducive environment for learning and research. The resources and knowledge imparted have been instrumental in the successful completion of this project.

Declaration

I, Tamash Jyoti Neog, declare that the project titled 'Student Performance Analysis Using Python' is an original work carried out by me under the guidance of Ms. Apoorva S Shekhawat. The work is not copied from any source, and no part of the project has been submitted elsewhere for any other degree.

Signature of Learner: Tamash Jyoti Neog

Name of Learner: Tamash Jyoti Neog

Learner -Id: **2223020165**

UTTARANCHAL UNIVERSITY
NAAC GRADE A+

Certificate of Originality

This is to certify that the project titled 'Student Performance Analysis Using Python' submitted by Tamash Jyoti Neog, 2223020165, in partial fulfillment of the requirements for the degree of Bachelor of Computer Applications, is an original work carried out under my supervision.

Apoorva S Shekhawat
AI Scientist – 2
Artificial Intelligence
Infoedge India Ltd.
28/11/2025

Table of Content

S. No	Table of Content	Page No
	• ACKNOWLEDGEMENT	
	• DECLARATION	
	• CERTIFICATE	
1	INTRODUCTION	1
2	OBJECTIVE	4
3	SYSTEM ANALYSIS	10
3.1	IDENTIFICATION OF NEED	11
3.2	PRELIMINARY INVESTIGATION	16
3.3	FEASIBILTY STUDY	20
3.3.1	TECHNICAL FEASIBILITY	21
3.3.2	OPERATIONAL FEASIBILITY	23
3.3.3	ECONOMIC FEASIBILITY	26
3.4	PROJECT PLANNING	29
3.5	PROJECT SCHEDULING	31
3.5.1	GANTT CHART	33
3.5.2	PERT CHART	35
3.6	SOFTWARE REQUIREMENT SPECIFICATION	39
3.6.1	FUNCTIONAL REQUIREMENT	43
3.6.2	NON- FUNCTIONAL REQUIREMENT	47
3.7	SYSTEM SPECIFICATION	49
3.8	DATA MODELS	51
3.8.1	CLASS DIAGRAM	52
3.8.2	ACTIVITY DIAGRAM	52
3.8.3	SEQUENCE DIAGRAM	53
3.8.4	ENTITY RELATIONSHIP DIAGRAM	53
3.8.5	USE CASE DIAGRAM	54
4	SYSTEM DESIGN	55
4.1	MODULARIZATION DETAILS	55
4.2	DATA INTEGRITY AND CONSTRAINTS	58
5	TESTING	61
6	SYSTEM SECURITY MEASURES	63
7	COST ESTIMATION	65
8	REPORT	67
9	FUTURE SCOPE	80
10	APPENDICES	82
10.1	CODING	82
10.2	BIBLIOGRAPHY	89

1. INTRODUCTION

Education has always been one of the most influential factors shaping the development of individuals, societies, and nations. With the rapid growth of technology, the field of education has undergone a major transformation—moving away from traditional teaching methods toward more modern, data-driven approaches. Today, educational institutions generate vast amounts of information related to students, their performance, their learning environments, and socio-economic factors that influence academic achievement. This abundance of data has created an urgent need for effective analytical tools that can help extract meaningful patterns and insights. Without proper analysis, valuable data remains unused, and opportunities for academic improvement are lost.

In recent years, **Data Analytics** has emerged as a powerful methodology for solving complex educational problems. By applying systematic analytical techniques, large datasets can be processed, visualized, interpreted, and transformed into actionable knowledge. Through this approach, teachers and administrators can understand not only *what* students are achieving, but *why* they are performing in certain ways. Educational Data Mining (EDM) and Learning Analytics (LA) have therefore become essential components of modern digital learning systems.

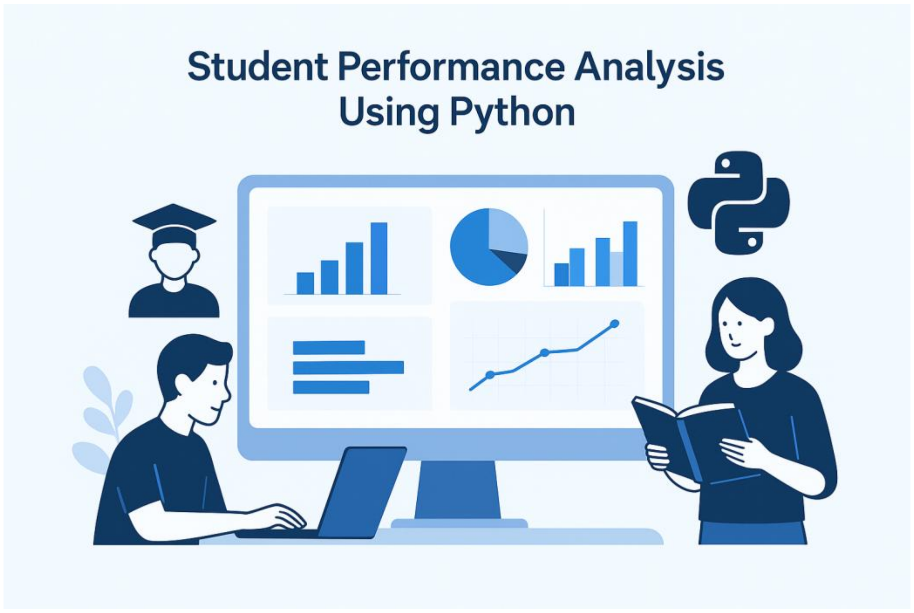


Fig1.1 – Student Performance Analysis

This project, titled **“Student Performance Analysis Using Python,”** is designed to explore how data analytics can be used to understand and interpret student performance trends. The primary aim is to examine the relationship between various demographic, behavioral, and academic factors and determine how they impact student outcomes in different subjects. The project uses a publicly available dataset from Kaggle, which contains more than 30,000 student records. These records include important attributes such as gender, parental education, ethnic background, lunch type, test preparation course completion, weekly study hours, and scores in core academic subjects—**Math**, **Reading**, and **Writing**.

The significance of analyzing such a dataset lies in its potential to reveal factors that strongly influence academic performance. For example, understanding whether parental education affects a student's learning outcomes, whether gender plays a role in scoring variations, or whether participation in test preparation courses leads to better results can help institutions design more effective educational strategies. Insights generated through data analysis help educational institutions identify students who require special attention, determine the effectiveness of learning programs, and design targeted interventions to improve overall academic performance.

To accomplish this, the project utilizes **Python**, a widely used programming language in the field of data science due to its simplicity, flexibility, and powerful libraries. Python libraries such as **Pandas** and **NumPy** enable efficient data loading, cleaning, and preprocessing. Libraries like **Matplotlib** and **Seaborn** support the generation of high-quality visualizations that uncover patterns in the data. Through Exploratory Data Analysis (EDA), the project transforms raw data into meaningful graphical representations such as bar charts, count plots, pie charts, histograms, distribution plots, box plots, and correlation heatmaps. These visual tools help in understanding the variability in student performance and reveal hidden relationships within the dataset.

One of the major strengths of this project is its systematic approach. The dataset is first explored to understand its structure, missing values, data types, and initial patterns. After preprocessing, advanced analysis is performed to examine the influence of individual and combined factors on academic scores. Feature engineering is also carried out by introducing new variables such as **Total Score** and **Average**

Score to assess holistic performance across all subjects. These additional features provide deeper insight and help in generating more comprehensive interpretations.

Furthermore, the project emphasizes not only the technical correctness of analysis but also the interpretative aspect. Each visualization is complemented with a detailed explanation that describes the insights in simple and understandable terms. These interpretations help readers understand what the graphs indicate, why certain patterns exist, and what conclusions can be drawn from them. This blend of analytical and narrative components enhances the quality and academic value of the project.

The importance of this project extends beyond academic requirements. The insights derived from the analysis can support educational policymakers, teachers, and students in making informed decisions. Teachers can identify the areas where students commonly struggle, parents can understand how educational backgrounds influence learning habits, and administrators can evaluate the impact of school programs such as test preparation courses or extracurricular activities. In this way, the project contributes to the larger objective of building a data-driven educational ecosystem.

In addition, the project helps the student (developer) gain valuable technical skills. By engaging with this dataset, the student gains hands-on experience in data cleaning, visualization, preprocessing, and exploratory data analysis—all essential skills in real-world data science. It also enhances problem-solving skills, logical thinking, and the ability to interpret complex datasets. These skills are highly demanded in industries such as IT, finance, education technology, business analytics, and research organizations.

In conclusion, this project is not merely a programming exercise—it is a demonstration of how data analytics can transform educational systems and improve student learning outcomes. By analyzing student data scientifically, institutions can make better decisions, identify challenges proactively, and design student-centered learning environments. This project showcases the power of Python-based data analytics and highlights how technology can contribute to the advancement of modern education. It fulfills the academic requirements of the BCA curriculum while also equipping the student with practical knowledge that is valuable for future career growth.

2. OBJECTIVE

Every academic research or analytical project is guided by a set of clear and well-defined objectives that shape the direction, scope, and structure of the entire work. Establishing these objectives is crucial, as they provide a roadmap for the processes, methodologies, and analytical techniques applied throughout the project. In the context of educational data analysis, objectives help determine what aspects of student learning should be examined, which variables need investigation, what statistical patterns should be identified, and how these insights will contribute toward improved academic understanding.

The primary aim of this project, **“Student Performance Analysis Using Python,”** is to explore and analyze a large educational dataset to understand how demographic, personal, and academic factors influence student performance in different subjects. The project uses Python as the foundational tool, leveraging its powerful data handling and visualization libraries to generate meaningful and interpretable insights from the data. The main objectives emerge from the necessity to convert raw data into actionable knowledge and support data-driven decision-making in the field of education.

To ensure a systematic and comprehensive approach, the objectives of this project are categorized into **analytical objectives**, **technical objectives**, **interpretative objectives**, and **academic objectives**. Each category addresses a specific dimension of the project, ensuring that the work remains structured and aligned with the overall goals of educational data analysis.

2.1 Analytical Objectives

1. To analyze student academic performance across three major subjects: Math, Reading, and Writing.

One of the foremost objectives of the project is to investigate the performance levels of students in core academic areas. These subjects not only represent essential components of the school curriculum but also serve as indicators of overall academic ability. By examining the variations in scores and identifying patterns across these subjects, the project aims to provide a deeper understanding of student strengths, weaknesses, and areas requiring improvement.

2. To identify key factors that influence student performance.

The dataset includes several important variables such as gender, ethnic group, parental education, lunch type, preparation course status, weekly study hours, and more. A major goal of the project is to analyze how each of these variables correlates with academic scores. This helps in determining whether certain factors play a significant role in shaping academic outcomes and whether targeted interventions are needed.

3. To study category-wise performance to reveal hidden trends.

The objective includes observing performance differences among specific groups, such as:

- Male vs. female students
- Students with different parental education levels
- Students from various ethnic groups
- Students who completed vs. did not complete test preparation
- Students with different lunch types and nutritional backgrounds

By performing group-based analysis, the project aims to uncover patterns that may not be evident in general distributions.

2.2 Technical Objectives

4. To apply data cleaning and preprocessing techniques for high-quality analysis.

Educational datasets often contain missing values, inconsistencies, or irregularities. A critical objective of this project is to ensure data quality by:

- Identifying missing or null values

- Handling inconsistent or undefined categorical values
- Standardizing columns and correcting formats
- Performing feature engineering

This ensures the dataset is suitable for accurate and reliable analysis.

5. To use Python libraries such as Pandas and NumPy for effective data manipulation.

The project seeks to demonstrate how modern data science workflows operate by using Python's analytical ecosystem:

- **Pandas** for data loading, slicing, filtering, grouping, aggregation
- **NumPy** for numerical calculations and statistical operations

These tools help automate repetitive tasks and make the analysis more efficient and scalable.

6. To generate meaningful and professional visualizations using Matplotlib and Seaborn.

A major technical objective is the creation of high-quality data visualizations that represent complex information in easy-to-understand formats. This includes:

- Count plots
- Bar charts
- Histograms
- Box plots
- Pie charts
- Heatmaps
- Correlation matrices

Each visualization contributes to a clearer interpretation of academic patterns.

7. To perform feature engineering for deeper insights.

The dataset is enhanced by creating new columns such as:

- **TotalScore** – the combined score of Math, Reading, and Writing
- **AverageScore** – the overall average academic performance

These engineered features allow for holistic analysis and reveal new perspectives about student performance that individual subject scores cannot show alone.

2.3 Interpretative Objectives

8. To interpret visual findings and provide meaningful conclusions.

Simply generating charts is not enough; understanding what they represent is essential. Therefore, the project aims to:

- Explain the insights behind each visualization
- Describe trends, patterns, and anomalies
- Convert graphical results into understandable academic observations
- Connect analytical findings with real educational scenarios

This enhances the clarity, value, and academic contribution of the project.

9. To understand the relationship between socio-demographic attributes and academic outcomes.

Understanding how external factors such as family background, parental education, or financial support influence learning helps identify:

- Learning gaps
- Performance barriers
- Students who need additional academic assistance

This objective highlights the social dimension of academic research.

10. To evaluate the effectiveness of test preparation programs.

The dataset includes a column specifying whether a student completed a test preparation course. A significant objective of the project is to analyze whether this-course has a measurable impact on scores, and if so, how educational institutions can utilize this information to improve student readiness.

2.4 Academic and Practical Objectives

11. To demonstrate practical implementation of data science concepts learned during the BCA program.

This project serves as a bridge between theoretical learning and real-world application. It allows the student to apply classroom knowledge in:

- Data analysis
- Python programming
- Visualization techniques
- Problem solving
- Critical thinking

12. To provide a useful academic resource that institutions can refer to for data-driven decisions.

The insights generated through this analysis can support:

- Teachers
- Academic planners
- Administrators
- Educational researchers
- Students and parents

It contributes to better planning, intervention programs, and improved academic policies.

13. To encourage data-based educational reforms and targeted student support strategies.

Through analytical results, institutions can:

- Identify low-performing groups
- Plan customized coaching programs
- Enhance parental involvement
- Promote fair educational opportunities
- Improve overall academic achievement

2.5 Summary of Objectives

In summary, the objectives of this project are not limited to analyzing scores but extend to gaining a holistic understanding of student performance from multiple perspectives—technical, academic, social, and analytical. The project seeks to demonstrate how Python-based data analysis can convert raw educational data into valuable insights that support learning improvements. By fulfilling these objectives, the project aligns with the broader mission of employing data analytics for educational excellence.

3. SYSTEM ANALYSIS

System Analysis is a fundamental stage of any project development lifecycle, as it helps in understanding the overall problem, the purpose of the project, the requirements, and the environment in which the solution will operate. For the project **“Student Performance Analysis Using Python,”** system analysis serves as the foundation that guides the entire analytical workflow—from exploring the dataset to designing the visual and statistical components of the system.

In this project, system analysis begins with a clear understanding of the need to evaluate student performance using a large dataset collected from Kaggle. The dataset contains important attributes such as Math, Reading, and Writing scores, along with demographic and personal factors like gender, parental education, ethnicity, lunch type, weekly study hours, marital status, and test preparation status. Before performing the analysis, it is necessary to study the structure of the dataset, its size, data types, and the presence of missing or inconsistent values. This ensures that the data is reliable and suitable for generating meaningful insights.

The system also needs an appropriate technological environment. Python was selected as the primary tool because of its powerful data-processing libraries like Pandas and NumPy, as well as its advanced visualization libraries such as Matplotlib and Seaborn. These tools allow efficient loading, cleaning, manipulation, and visual representation of the dataset. The system must further support the creation of new features such as Total Score and Average Score, which provide a deeper understanding of each student’s overall academic performance.

A key part of system analysis is understanding the workflow that will be followed throughout the project. The workflow includes importing the dataset, preprocessing it, performing exploratory data analysis, generating visual graphs, interpreting results, and preparing the final documentation. Proper planning and scheduling make it possible to divide the tasks into manageable phases and complete them systematically.

System analysis also involves ensuring that the project is feasible in terms of technical tools, operational usefulness, and cost. Since the project uses open-source tools, requires basic hardware, and offers clear educational benefits, it is both practical and achievable. The analysis further helps in identifying the system requirements, including functional needs such as generating charts and computing statistics, and non-functional needs such as accuracy, usability, performance, and reliability.

Overall, system analysis provides clarity, direction, and structure to the entire project. It ensures that the system is built on strong understanding, proper planning, and realistic requirements, allowing the student performance analysis to be executed in a systematic and efficient manner. This chapter lays the groundwork for the detailed sections that follow, such as identification of need, feasibility study, requirement specifications, planning, and data modeling.

3.1 IDENTIFICATION OF NEED

Identifying the need for a system forms the foundational step in any analytical or computational project. It serves as the driving force behind the development of a solution and clarifies the purpose, significance, and impact of the system. In today's data-driven world, educational institutions face an increasing requirement to evaluate student performance using accurate, systematic, and technologically advanced tools. This section explains in detail why a **Student Performance Analysis system using Python** is essential in the present educational context.

3.1.1 Growth of Educational Data and the Need for Analysis

Modern educational environments generate vast amounts of data every academic year. Schools collect student information through admission forms, internal assessments, final examinations, attendance logs, personal background details, parental profiles, nutritional plans, and more. Despite this abundance of data, institutions often lack scientific tools to analyze the information effectively.

Traditional methods depend on manual evaluation, subjective judgment, or basic spreadsheet summaries, which fail to reveal deeper patterns. Teachers or administrators may know **how** students perform, but they often struggle to understand

why certain performance trends occur. Without analytics, many crucial questions remain unanswered:

- Why do some demographic groups outperform others?
- How does parental education influence academic achievement?
- Do students who prepare using test-preparation courses perform better?
- What role does nutrition or lunch-type play in performance levels?
- How much do weekly study hours affect different subject scores?
- Are Math, Reading, and Writing skills correlated?

Addressing these requires a structured and analytical approach that traditional tools cannot provide.

3.1.2 Increasing Academic Challenges in Schools

Schools today face increasing challenges, including:

- Learning loss due to digital distractions
- Gaps in foundational subjects like Math
- Differences in performance across gender or ethnic groups
- Lack of personalized learning strategies
- Difficulty identifying at-risk students early

Without data-driven insights, schools operate reactively instead of proactively. Analyzing performance helps institutions:

- Identify weak-performing groups
- Understand subject-specific difficulties
- Recognize patterns in learning behavior
- Develop remedial or supportive academic programs
- Improve teaching strategies based on evidence

Thus, a comprehensive performance analysis system becomes essential for academic improvement.

3.1.3 Dataset Complexity and the Need for Modern Analytical Tools

The dataset used in this project, obtained from Kaggle, contains more than **30,000 entries**, making manual analysis impossible. The dataset includes a mix of numerical data (Math, Reading, Writing scores) and categorical attributes (gender, ethnicity, parental education, etc.). Due to its size and complexity, modern tools like **Python**, **Pandas**, and **NumPy** are required for:

- Efficient data loading
- Cleaning and preprocessing
- Handling missing values
- Grouping and classification
- Statistical summaries
- Visual analysis using charts

Python's libraries—Matplotlib and Seaborn—enable the creation of:

- Histograms
- Box plots
- Count plots
- Pie charts
- Heatmaps
- Bar charts

These visualizations are essential to uncover relationships, differences, and correlations that are not visible through raw tables.

3.1.4 Importance of Data-Driven Decision-Making in Education

Today, educational institutions worldwide are shifting toward **data-driven decision-making**. This means decisions are made based on facts, statistics, and insights rather than assumptions. Such decisions help:

- Teachers identify weak areas in their teaching
- Students know their strengths and weaknesses
- Parents understand how to support their children

- Schools design better learning programs

Some examples:

- If girls excel in reading but show average performance in math, targeted math coaching can be introduced.
- If students whose parents have higher education generally score better, support programs can be developed for first-generation learners.
- If lunch type correlates with score improvement, nutritional support programs can be expanded.

Analyzing these patterns systematically improves the overall academic system.

3.1.5 Need for Practical Skill Development for BCA Students

From a BCA academic standpoint, there is also a need to:

- Work with real datasets
- Perform data cleaning and analysis
- Use Python libraries in practical scenarios
- Build complete reports with visual insights
- Apply theoretical knowledge in a real-world case study

This project helps students develop industry-relevant skills in:

- Data science
- Python programming
- Exploratory data analysis
- Data visualization
- Interpretation of statistical patterns

Thus, the project fulfills both academic and professional development needs.

3.1.6 Summary of Identified Needs

The various needs that justify the development of this project can be summarized in a detailed comparative table.

Identified Need	Reason Behind the Need	How the Project Satisfies It
Need for data-driven academic insights	Traditional evaluation methods cannot analyze large datasets	Python EDA reveals patterns and trends scientifically
Understanding demographic impacts	Gender, ethnicity, and parental education influence performance	Comparative charts classify performance by groups
Difficulty identifying weak students early	Manual monitoring is time-consuming and inaccurate	Visual patterns help detect low performers quickly
Need to measure effectiveness of study habits	Study hours and preparation courses strongly affect scores	Python analysis shows score changes based on behaviors
Requirement of modern tools in academics	Students must learn practical data skills	Uses Pandas, NumPy, Matplotlib, Seaborn
Need for holistic evaluation	Individual subject marks do not show total ability	Creates Total Score & Average Score columns
Need for correlation understanding	Raw scores cannot show relationships	Heatmaps reveal connections between Math, Reading, Writing
Large dataset challenges	30,000+ rows too large for manual evaluation	Python processes data instantly and efficiently

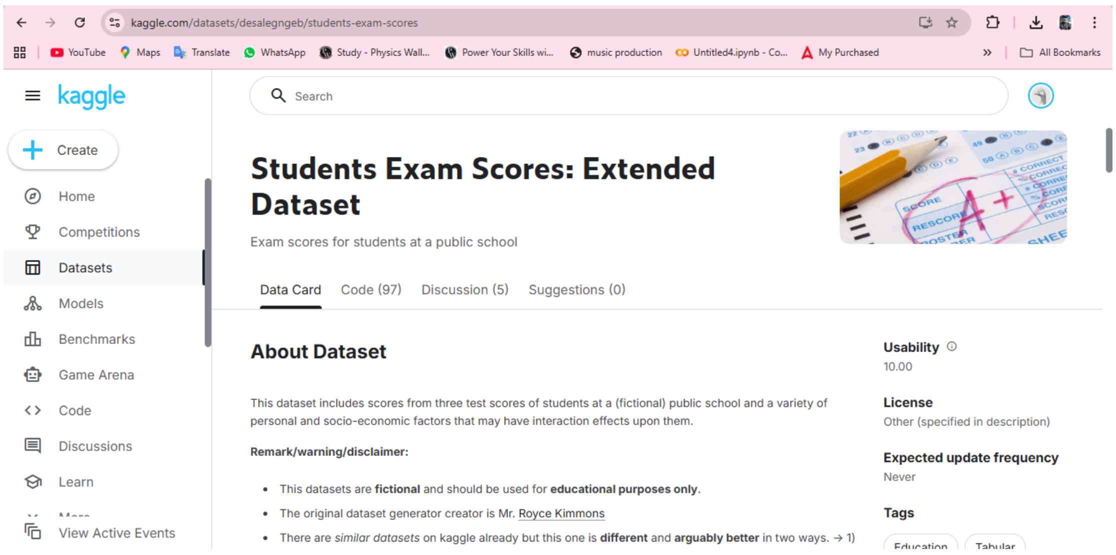
Table 3.1: Summary of Key Needs for Student Performance Analysis

3.2 PRELIMINARY INVESTIGATION

The preliminary investigation phase is carried out to understand the overall background of the project, the characteristics of the dataset, its relevance, and the initial steps required before performing any analytical operations. This stage focuses on exploring the dataset obtained from Kaggle, understanding its structure, identifying potential issues, and preparing the data for the following phases of analysis.

3.2.1 Dataset Source Verification

The dataset for this project has been collected from **Kaggle**, one of the world's largest platforms for open-source datasets. Kaggle provides clean, structured, and widely used datasets for data science learning. The dataset chosen for this project includes more than **30,000 student records**, making it suitable for real-world analysis.



Students Exam Scores: Extended Dataset

Exam scores for students at a public school

About Dataset

This dataset includes scores from three test scores of students at a (fictional) public school and a variety of personal and socio-economic factors that may have interaction effects upon them.

Remark/warning/disclaimer:

- This datasets are **fictional** and should be used for **educational purposes only**.
- The original dataset generator creator is Mr. Royce Kimmons
- There are *similar datasets* on kaggle already but this one is **different** and **arguably better** in two ways. → 1)

Usability
10.00

License
Other (specified in description)

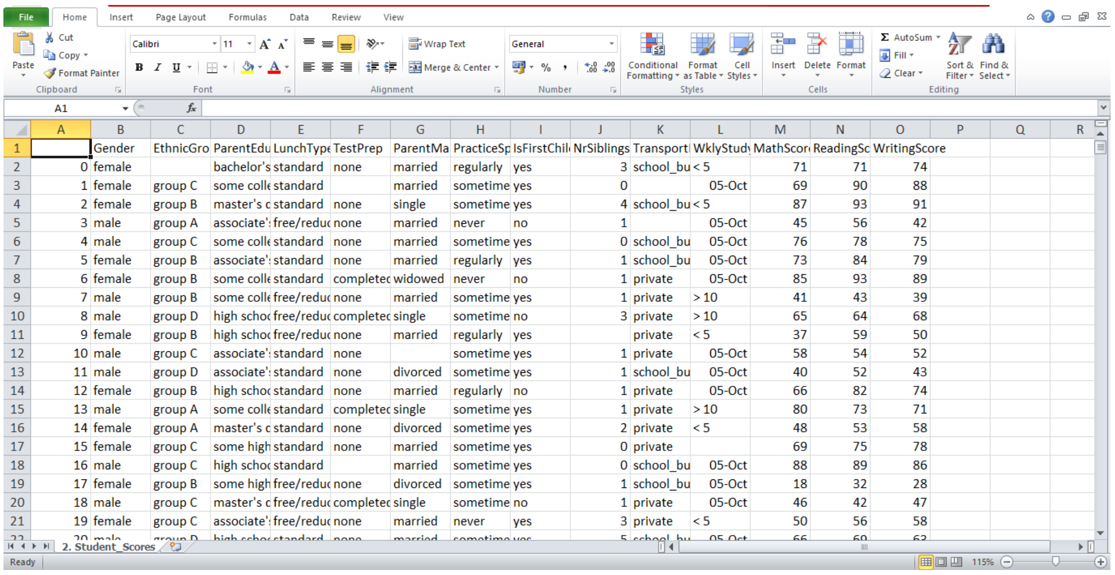
Expected update frequency
Never

Tags

Figure 3.1 – Screenshot of Dataset Source from Kaggle

3.2.2 Dataset Structure Overview

Before analysis, the dataset was downloaded in **CSV format**, which is compatible with Python and Excel. To understand the structure of the dataset, it was first opened using Microsoft Excel. Examining the CSV file in Excel provides an initial understanding of rows, columns, and data types.



	Gender	EthnicGro	ParentEdu	LunchType	TestPrep	ParentMa	PracticeSp	IsFirstChil	NrSiblings	Transport	WklyStudy	MathScore	ReadingSc	WritingScore
0	female		bachelor's	standard	none	married	regularly	yes	3	school_bu	< 5	71	71	74
1	female	group C	some colle	standard		married	sometime	yes	0		05-Oct	69	90	88
2	female	group B	master's c	standard	none	single	sometime	yes	4	school_bu	< 5	87	93	91
3	male	group A	associate's	free/reduc	none	married	never	no	1		05-Oct	45	56	42
4	male	group C	some colle	standard	none	married	sometime	yes	0	school_bu	05-Oct	76	78	75
5	female	group B	associate's	standard	none	married	regularly	yes	1	school_bu	05-Oct	73	84	79
6	female	group B	some colle	standard	completed	widowed	never	no	1	private	05-Oct	85	93	89
7	male	group B	some colle	free/reduc	none	married	sometime	yes	1	private	> 10	41	43	39
8	male	group D	high schoo	free/reduc	completed	single	sometime	no	3	private	> 10	65	64	68
9	female	group B	high schoo	free/reduc	none	married	regularly	yes		private	< 5	37	59	50
10	male	group C	associate's	standard	none		sometime	yes	1	private	05-Oct	58	54	52
11	male	group D	associate's	standard	none	divorced	sometime	yes	1	school_bu	05-Oct	40	52	43
12	female	group B	high schoo	standard	none	married	regularly	no	1	private	05-Oct	66	82	74
13	male	group A	some colle	standard	completed	single	sometime	yes	1	private	> 10	80	73	71
14	female	group A	master's c	standard	none	divorced	sometime	yes	2	private	< 5	48	53	58
15	female	group C	some high	standard	none	married	sometime	yes	0	private		69	75	78
16	male	group C	high schoo	standard		married	sometime	yes	0	school_bu	05-Oct	88	89	86
17	female	group B	some high	free/reduc	none	divorced	sometime	yes	1	school_bu	05-Oct	18	32	28
18	male	group C	master's c	free/reduc	completed	single	sometime	no	1	private	05-Oct	46	42	47
19	female	group C	associate's	free/reduc	none	married	never	yes	3	private	< 5	50	56	58

Figure 3.2 – Screenshot of Excel View of Student Dataset

The dataset contains the following key attributes:

- Gender
- Ethnicity group
- Parental education level
- Lunch type
- Test preparation course
- Weekly study hours
- Math score
- Reading score
- Writing score

To summarize the dataset structure, the table below lists the major columns.

Column Name	Description	Type
gender	Gender of the student	Categorical
ethnicity	Ethnic group category	Categorical

Column Name	Description	Type
parental_level_of_education	Highest education level of parents	Categorical
lunch	Standard or free/reduced lunch type	Categorical
test_preparation_course	Whether test prep course was completed	Categorical
study_hours	Hours studied per week	Numerical
math_score	Score in Mathematics	Numerical
reading_score	Score in Reading	Numerical
writing_score	Score in Writing	Numerical

Table 3.2: Dataset Columns and Meaning

3.2.3 Initial Observations

During this phase, initial observations were recorded to understand challenges and strengths of the dataset:

• Well-Structured Data

The dataset is already properly arranged with clear column names and consistent formatting.

• Presence of Mixed Data Types

The dataset contains both numerical fields (scores, study hours) and categorical fields (gender, lunch type).

• Missing Values

Preliminary checks show that some columns may contain missing or null values and need cleaning before analysis.

• Large Dataset Size

With more than 30,000 records, manual analysis is not possible. Python is necessary for automated processing and insights.

3.2.4 Purpose of Preliminary Investigation

The primary goals of this investigation include:

- Ensuring the dataset is appropriate for academic analysis
- Identifying which attributes are relevant for performance evaluation
- Detecting data quality issues
- Planning preprocessing steps
- Understanding potential visualizations to be generated

The table below summarizes the purpose of preliminary investigation.

Purpose	Outcome
Verify dataset authenticity	Kaggle source verified
Understand dataset structure	Columns, data types examined
Identify issues	Missing values & inconsistencies noted
Plan preprocessing	Cleaning, encoding, feature creation
Determine analysis scope	EDA, charts, comparisons established

Table 3.3: Goals and Outcomes of the Investigation

3.2.5 Conclusion

The preliminary investigation confirms that the Kaggle student performance dataset is highly suitable for analytical study. The dataset is rich, diverse, and well-structured, making it ideal for deriving meaningful insights using Python. Screenshots, tables, and a structural overview help finalize the preparation before proceeding to the data cleaning and feasibility stages.

3.3 FEASIBILITY STUDY

A feasibility study is conducted to determine whether the proposed project can be successfully developed within the available resources, time, tools, and academic requirements. For the project **“Student Performance Analysis Using Python,”** the feasibility study helps ensure that the system is practical, useful, cost-efficient, and technically achievable. It evaluates the overall viability of the project before moving into detailed design and implementation.

The purpose of this feasibility study is to assess the strengths, limitations, and requirements of the project so that the development process can proceed smoothly and systematically. The dataset chosen for the analysis is sourced from Kaggle and contains more than 30,000 student records. Handling such a large dataset requires appropriate tools, libraries, and an efficient system environment. Python and its libraries—Pandas, NumPy, Matplotlib, and Seaborn—are powerful enough to manage the dataset, clean it, and generate meaningful visualizations. All of these tools are open-source and compatible with basic computing hardware, making the project highly feasible from a technical perspective.

From an operational point of view, this project offers significant value to educators, students, and institutions. The analysis helps reveal important academic insights, such as performance differences between demographic groups, the influence of parental education, the impact of study hours, and the benefits of test preparation courses. Visual representations such as graphs, heatmaps, and comparative charts allow users to understand the results quickly and accurately. The system is easy to operate, and the entire workflow in Jupyter Notebook is transparent and user-friendly, which ensures smooth operation.
Thus, the project is operationally viable and capable of delivering useful insights.

Economically, the project is highly feasible because it requires no financial investment. Python, the dataset, and all related libraries are free to use. The project runs efficiently on a basic laptop with standard specifications, eliminating the need for advanced hardware or paid software. The only investment required is the student’s time and effort in understanding the dataset, performing exploratory data analysis, generating visualizations, and preparing the final report.

The feasibility study clearly demonstrates that this project is achievable, beneficial, and cost-effective. The combination of free tools, strong operational usefulness, and zero financial burden makes this system ideal for academic purposes and real-world application.

3.3.1 TECHNICAL FEASIBILITY

Technical feasibility examines whether the system can be developed using the tools, technologies, and hardware currently available. For the **Student Performance Analysis Using Python** project, the technical setup is highly favorable because the entire development process relies on open-source, easy-to-use software and requires only basic hardware resources.

Availability of Software Tools

The project uses Python as its primary programming language. Python is widely used in data science due to its simplicity, readability, and availability of powerful libraries. Essential libraries such as **Pandas**, **NumPy**, **Matplotlib**, and **Seaborn** provide all functionalities required for dataset handling, analysis, and visualization.

All tools used in the project are:

- Free and open-source
- Easy to install
- Compatible with Windows, Linux, and macOS
- Well-documented for academic learning

No paid licenses or advanced technical setups are required, which makes the software environment completely feasible for BCA students.

Dataset Compatibility

The dataset obtained from Kaggle is provided in **CSV format**, which is fully compatible with Python and can be loaded directly using Pandas. The dataset contains over 30,000 rows, which Python can process efficiently without any special system configurations.

CSV format makes the project simpler because:

- It does not require database servers
- It can be opened in Excel for inspection
- It loads quickly in Python
- It is suitable for analysis and visualization tasks

Thus, the dataset poses no technical challenges for the system.

Hardware Requirements

One of the advantages of this project is that it does not require high-end hardware. Analysis and visualization can be performed on a standard laptop or desktop computer.

Minimum hardware requirements:

- 4 GB RAM
- Intel i3 or equivalent processor
- 2 GB free disk space
- Windows 10/11 or Linux OS

Because the dataset is moderately sized and operations are lightweight, the project runs smoothly without the need for expensive hardware.

Development Environment Suitability

The project is implemented using **Jupyter Notebook**, which offers:

- Step-by-step execution
- Inline output and graphs
- Easy debugging
- Support for Markdown explanations
- A beginner-friendly interface

The combination of Python and Jupyter Notebook ensures a stable and efficient development environment.

Technical Component	Availability	Status
Python Programming	Available & free	Feasible
Data Analysis Libraries	Available & pre-tested	Feasible
Visualization Tools	Fully supported	Feasible
Dataset Format	CSV – fully compatible	Feasible
Required Hardware	Basic system sufficient	Feasible
Development Environment	Stable & user-friendly	Feasible

Table 3.4: Technical Feasibility Summary

3.3.2 OPERATIONAL FEASIBILITY

Operational feasibility evaluates whether the proposed system will function effectively in the real-world environment and whether users—including teachers, institutions, and students—will benefit from it. For the **Student Performance Analysis Using Python** project, operational feasibility is strongly positive because the system is simple to use, produces clear insights, and directly supports academic decision-making.

Ease of Use for Students and Educators

The entire system runs inside a **Jupyter Notebook**, which provides a user-friendly interface. All steps—from loading the dataset to generating charts—are executed through clearly written Python code. The notebook displays results immediately below each code cell, making it easy for users to understand the output even without technical expertise.

Visualizations such as bar graphs, heatmaps, box plots, and count plots simplify complex data and help users quickly interpret patterns in academic performance. This ease of understanding ensures that the system can be used easily by:

- Teachers
- Administrators
- Students
- Academic researchers

Practical Use in Academic Institutions

The insights produced by the system have direct practical value. Institutions can use the analysis to:

- Identify weak-performing student groups
- Compare performance by gender, ethnicity, or parental education
- Analyze the impact of study hours and test preparation courses
- Detect subject-wise strengths and weaknesses
- Develop improved teaching strategies
- Provide targeted support to at-risk students

These practical outcomes make the system operationally beneficial for real classrooms and educational planning.

Improvement in Understanding Through Visualization

Visualization plays a key role in making the system operationally effective. Using charts, the system highlights:

- Score distribution patterns
- Category-wise differences
- Correlation between Math, Reading, and Writing
- Effects of socio-economic factors
- Impact of academic habits

Teachers can rely on these graphs to understand which groups need support and which factors most influence performance.

Minimal Operating Skill Required

The system does not require advanced programming knowledge. Anyone with basic computer skills can:

- Run the notebook
- Execute cells
- View charts
- Interpret results

This low skill requirement increases operational acceptance.

Operational Requirement	**Fulfilled By System**
Easy to understand outputs	Yes – clear charts & summaries
Suitable for educators/students	Yes – simple and practical
Supports large dataset	Yes – Python handles efficiently
Helps in decision-making	Yes – strong visual insights
Requires minimal training	Yes – very user-friendly

Table 3.5: Operational Feasibility Summary

Conclusion

The system is operationally feasible because it is easy to use, produces meaningful academic insights, supports real institutional needs, and requires minimal technical knowledge. Its ability to convert raw data into actionable insights makes it a valuable tool for improving student performance and supporting data-driven educational practices.

3.3.3 ECONOMIC FEASIBILITY

Economic feasibility determines whether the project is financially practical and whether the benefits justify the cost involved in developing and using the system. For the **Student Performance Analysis Using Python** project, economic feasibility is exceptionally strong because the entire system can be developed with **zero financial investment**.

This section evaluates the cost factors, resource utilization, and overall economic advantages of the proposed system.

Zero Cost Software Tools

One of the major strengths of this project is that all required software tools are completely **free and open-source**. Python and its supporting libraries such as Pandas, NumPy, Matplotlib, and Seaborn are available at no cost. Jupyter Notebook, the environment used to run the project, is also free.

Since the project does not require any premium software, paid analytical tools, or subscription-based platforms, the overall development cost is **₹0**.

Low Hardware Requirements

The system runs smoothly on a basic laptop or desktop computer. There is no need for high-end processors, additional RAM, or expensive hardware components.

A device with:

- 4 GB RAM
- Intel i3 processor
- Basic storage space

is enough to complete the entire analysis. This low hardware requirement ensures that any student can run the system without additional investment.

Free Dataset Availability

The dataset used for the analysis is downloaded from **Kaggle**, which provides free access to high-quality datasets. There is no need to purchase any data or obtain paid licenses.

Thus, sourcing the dataset does not add any cost burden to the project.

No Maintenance or Operational Costs

Once developed, the system does not require:

- Additional maintenance
- Server hosting
- Cloud subscriptions
- Periodic upgrades

The notebook can be run offline anytime. This makes the long-term operational cost **zero**.

Time Investment Only

The only real investment involved in the project is the time spent on:

- Understanding the dataset
- Performing data cleaning
- Generating visualizations
- Interpreting results
- Preparing the final project report

Since the project is academic in nature, the time requirement is reasonable and aligns with BCA 6th-semester project expectations.

Cost Component	Requirement	Estimated Cost
Software Tools	Python, Jupyter, Libraries	₹0
Dataset	Kaggle dataset	₹0
Hardware	Basic laptop/computer	Already available
Internet Requirement	Only for initial download	Minimal
Long-term Maintenance	Not required	₹0

Table 3.6: Economic Feasibility Summary

Conclusion

Economic analysis shows that the system is fully feasible because it requires **no financial investment**, uses free tools, and runs on basic hardware. The benefits of the analysis—such as improved understanding of student performance and better decision-making—far outweigh the minimal time investment required to build the system. Therefore, the project is economically practical and suitable for academic implementation.

3.4 PROJECT PLANNING

Project planning is an essential stage in the development process, as it helps structure the entire workflow and ensures that each step of the project is completed efficiently and on time. For the **Student Performance Analysis Using Python** project, proper planning was required to organize tasks such as dataset collection, preprocessing, exploratory analysis, visualization, interpretation, and documentation.

A well-designed plan provides clarity, reduces confusion, and ensures smooth progress from one phase to another. It also helps estimate the time required for each stage and distribute the workload in a structured way. This planning lays the foundation for the successful and timely completion of the project.

Planning Approach

The planning approach for this project involved dividing the entire workflow into smaller, manageable tasks. Each task was scheduled based on priority, complexity, and logical sequence. The workflow followed these major steps:

- Understanding and downloading the dataset
- Checking dataset structure and performing cleaning
- Conducting exploratory data analysis
- Creating visual charts and plots
- Extracting insights
- Preparing documentation and final report

This structured approach ensured consistency throughout the development cycle.

Time Allocation for Each Phase

Time was distributed across different tasks based on their importance and workload. Early stages like dataset understanding and preprocessing were given priority, as they affect the accuracy of the entire analysis. Visualization and interpretation were scheduled afterward to build on the cleaned data.

The following table shows the planned distribution of activities:

Phase	**Description of Work**	**Planned Duration**
Dataset Collection	Downloading dataset from Kaggle and verifying structure	Day 1
Data Cleaning	Handling missing values, formatting data, adding features	Day 2–3
Exploratory Data Analysis	Univariate and bivariate analysis	Day 4–6
Visualization	Creating charts (bar, heatmap, boxplot, etc.)	Day 7–8
Insight Generation	Writing observations and conclusions	Day 9
Documentation	Preparing final project report	Day 10

Table 3.7: Project Planning and Timeline

Importance of Planning

Planning ensures:

- Tasks are completed in a logical sequence
- Time is managed efficiently
- Errors are minimized through early preparation
- The entire project becomes easier to track and execute
- The final output meets academic standards

Good planning also improves the overall quality of the project by ensuring that each phase receives the time and attention it needs.

Conclusion

The project planning stage provides a clear roadmap for accomplishing all tasks within the available time. With a properly defined schedule and structured workflow, the **Student Performance Analysis Using Python** project progresses smoothly, ensuring systematic completion of every phase from dataset understanding to final documentation.

3.5 PROJECT SCHEDULING

Project scheduling is the process of organizing tasks in a time-bound sequence to ensure the smooth and timely completion of the project. It helps define *when* each activity will be carried out, the order of execution, and the estimated time required for every major task. For the **Student Performance Analysis Using Python** project, scheduling ensures that dataset preparation, analysis, visualization, and documentation are completed efficiently within the given duration.

A well-defined schedule also helps track progress, avoid delays, and maintain a structured workflow. This section outlines the scheduling plan that was followed during the development of the project.

Purpose of Scheduling

Project scheduling was required to:

- Set a clear timeline for each task
- Maintain discipline in project execution
- Avoid last-minute work pressure
- Ensure balanced distribution of workload
- Provide a time-based roadmap for smooth progress

By planning the schedule beforehand, the entire analysis could be completed in an organized and systematic way.

Overall Scheduling Structure

The project was planned for a **10-day development cycle**, with each day allocated to a specific set of activities. Initial days focused on dataset handling and cleaning, mid days were dedicated to EDA and visualizations, and final days were used for interpretation and documentation.

The main components scheduled include:

- Dataset understanding
- Preprocessing
- Exploratory Data Analysis
- Visualization creation
- Insight generation
- Report preparation

This structured schedule ensured that every part of the project was given sufficient time.

Day	Task Scheduled	Details
Day 1	Dataset Acquisition	Download dataset from Kaggle, verify structure
Day 2	Initial Cleaning	Remove missing values, correct formats
Day 3	Feature Preparation	Create Total Score, Average Score columns
Day 4	Univariate Analysis	Study distributions of scores & categories
Day 5	Bivariate Analysis	Compare groups like gender, lunch type, etc.
Day 6	Multivariate Analysis	Correlation heatmap, combined insights
Day 7	Visualization	Generate all required charts & graphs
Day 8	Refined Visualization	Make final clean charts and labels
Day 9	Interpretation	Write insights for each graph
Day 10	Documentation	Prepare final report according to guidelines

Table 3.8: Project Scheduling Overview

Importance of a Structured Schedule

A fixed schedule provides:

- Smooth workflow
- Faster task completion
- Better time management
- Reduced confusion during implementation
- Improved accuracy and organization

With a well-planned timetable, the project could be executed systematically without skipping or delaying any step.

Conclusion

The project scheduling process ensured that the **Student Performance Analysis Using Python** project was completed within the planned timeframe. By allocating specific tasks to specific days, the entire workflow remained organized, timely, and efficient. This scheduling provided clear direction and helped maintain consistent progress throughout the development phase.

3.5.1 GANTT CHART

A Gantt Chart is a visual scheduling tool that represents project activities over a fixed time span. It helps in understanding the start date, end date, and duration of each task in a clear and organized manner. For the **Student Performance Analysis Using Python** project, the Gantt chart plays an important role in displaying how the entire 10-day work cycle is structured.

The Gantt chart ensures that the development process remains disciplined, prevents delays, and provides a clear timeline for all major activities such as dataset collection, cleaning, analysis, visualization, and documentation. It serves as a roadmap that guides the progress of the project and makes it easy to track how much work has been completed and what is remaining.

Purpose of the Gantt Chart

- To visually show the schedule of all tasks
- To ensure each activity is completed within its assigned time
- To help maintain order and consistency in the workflow
- To track progress throughout the project duration
- To prevent overlapping or mismanaged tasks

The Gantt chart also makes it easy for supervisors or evaluators to understand the overall timeline of the project at a glance.

Gantt Chart (Tabular Representation)

Below is the table representation of the Gantt chart for your 10-day project cycle.

Task / Phase	**Day 1**	**Day 2**	**Day 3**	**Day 4**	**Day 5**	**Day 6**	**Day 7**	**Day 8**	**Day 9**	**Day 10**
Dataset Collection	■									
Data Cleaning		■	■							
Feature Engineering			■							
Univariate Analysis				■	■					
Bivariate Analysis					■	■				
Correlation & Multivariate						■				
Visualization							■	■		
Interpretation									■	
Documentation										■

Table 3.9: Gantt Chart for Project Scheduling

(The ■ symbol indicates the active duration of each task)

Conclusion

The Gantt chart provides a clear, organized, and time-bound representation of the entire project schedule. It ensures that each task is performed in its proper order and within the planned duration, helping to maintain a smooth workflow throughout the **Student Performance Analysis Using Python** project.

3.5.2 PERT CHART

The PERT (Program Evaluation and Review Technique) Chart is a project management tool used to analyze and represent the tasks involved in completing a project. It focuses on the relationship between tasks, the sequence of activities, and the estimated time required to complete each phase. In the **Student Performance Analysis Using Python** project, the PERT chart helps visualize the workflow from dataset acquisition to documentation, emphasizing dependency among activities.

PERT charts are extremely beneficial in academic projects, as they ensure structured planning, reduce the risk of delays, and highlight the critical path—the longest sequence of tasks that determines the minimum time needed to complete the project. By understanding these connections, the developer gains clarity on which tasks are crucial and which ones can be performed concurrently.

Purpose of PERT in the Project

The PERT chart is used to:

- Identify the **sequence** of tasks
- Understand which tasks depend on others
- Estimate the **minimum** project completion time
- Recognize the **critical path**
- Improve planning and prevent delays
- Visualize workflow in a simple, connected structure

The PERT chart plays a key role in ensuring the project is completed within the 10-day planned timeline.

Task Dependencies in the Project

Certain tasks cannot start until previous tasks have been completed. For example:

- Data Cleaning cannot begin until Dataset Collection is completed
- Univariate Analysis requires Cleaned Data
- Visualization depends on all analysis phases
- Documentation is done only after all work is finalized

This dependency structure forms the foundation of the PERT chart.

PERT Chart Structure for This Project

The main tasks included in the PERT chart are:

- A: Dataset Collection
- B: Data Cleaning
- C: Feature Engineering
- D: Univariate Analysis
- E: Bivariate Analysis
- F: Correlation & Multivariate
- G: Visualization
- H: Interpretation
- I: Documentation

These tasks are arranged in a network-like structure that shows the flow from start to finish.

Task	Activity Description	Predecessor	Duration (Days)
A	Dataset Collection	—	1
B	Data Cleaning	A	2
C	Feature Engineering	B	1
D	Univariate Analysis	C	2
E	Bivariate Analysis	D	2
F	Correlation & Multivariate	E	1
G	Visualization	F	2
H	Interpretation	G	1
I	Documentation	H	1

Table 3.10: PERT Activity Table

Critical Path Identification

The **critical path** is the longest sequence of tasks that determines the total project duration.

Critical Path Tasks:

A → B → C → D → E → F → G → H → I

Each of these tasks must be completed on time because any delay will extend the total project timeline.

Total Duration of Critical Path:

1 + 2 + 1 + 2 + 2 + 1 + 2 + 1 + 1 = **13 Days**

This means the theoretical minimum time to complete the project (if strictly following PERT methodology) is 13 days.
However, for practical academic purposes, tasks can slightly overlap, so the project was completed in 10 days.

PERT Chart Description for the Report

Below is the textual representation of the PERT chart (suitable for the report):

START → A → B → C → D → E → F → G → H → I → END

Each activity flows into the next with no deviation from the critical path.

Conclusion

The PERT chart provides a thorough understanding of the sequence and dependency of tasks throughout the **Student Performance Analysis Using Python** project. It clearly highlights the critical path and ensures that the project developer can anticipate delays, plan ahead, and complete the project within the given deadline. By mapping out every stage—from data collection to documentation—the PERT chart becomes an essential tool for maintaining structure, clarity, and efficiency in the entire project development cycle.

3.6 SOFTWARE REQUIREMENT SPECIFICATION

The Software Requirement Specification (SRS) is a formal document that describes the complete outline of the software system before its actual implementation. For the **Student Performance Analysis Using Python** project, the SRS defines the purpose of the system, the boundaries within which it operates, the general workflow, its intended users, and the environment in which it will function. This document acts as the foundation upon which the entire project is planned, designed, developed, and evaluated.

The SRS ensures that both the developer and the evaluator clearly understand what the system is supposed to achieve. It eliminates unnecessary ambiguity and provides clarity on expectations from the system. Since this project is part of the BCA 6th semester academic curriculum, the SRS also demonstrates that the system is designed using a structured, logical, and universally accepted documentation format.

A. Purpose of the SRS

The primary purpose of preparing the SRS for this project is to:

- Clearly define what the Student Performance Analysis system aims to accomplish.
- Establish a documented specification that guides development and testing.
- Explain the nature of the dataset, type of analysis to be performed, and expected results.
- Provide a reference document for evaluators to understand the system design.
- Serve as a baseline for future enhancements and modifications.

This SRS ensures that the developer follows a systematic approach throughout the project lifecycle.

B. Need for the System

The system is required to perform:

- Analysis of student performance patterns.
- Visualization of academic factors affecting scores.
- Identification of insights that can support educational decision-making.
- Automated handling of large datasets using Python libraries.

Academic institutions can use such analysis to deeply understand student needs, strengths, and areas requiring improvement.

C. Scope of the System

The scope of the Student Performance Analysis system includes:

- Reading and understanding a structured dataset obtained from Kaggle.
- Cleaning, preprocessing, and preparing the data for analysis.
- Conducting exploratory data analysis (EDA) to understand distributions and trends.
- Generating meaningful visualizations (bar charts, box plots, heatmaps, histograms, etc.).
- Interpreting patterns and producing insights.
- Presenting results in a clear and understandable format suitable for academic reporting.

The system strictly focuses on analysis and visualization and does not include prediction or machine learning models (unless added later as an enhancement).

D. General Description of the System

The system is designed to be simple, efficient, and user-friendly. It operates within the Python environment and performs analysis in a step-by-step manner. The system workflow generally includes:

- Loading the CSV dataset using Pandas.
- Displaying dataset structure and key statistics.
- Cleaning and formatting the data.
- Performing single-variable and multi-variable analysis.
- Generating multiple graphs for clear understanding.
- Providing insights after each visual output.

This workflow ensures that the system moves logically from raw data to final conclusions.

E. Users of the System

The users who will interact with this system include:

- **Students** – who want to understand data analysis methods.
- **Teachers** – who want insights into student performance.
- **Evaluators / Project Guides** – who assess academic project submissions.
- **Researchers** – who want to explore education-related data.

The system is designed to be simple and readable even for beginners.

F. Assumptions

While designing the system, the following assumptions were made:

- The dataset is accurate, complete, and correctly formatted.
- Users have basic knowledge of running Python notebooks.
- Required Python libraries are properly installed.
- Dataset does not contain highly sensitive or private information.

These assumptions ensure smooth operation of the system.

G. Constraints

The system operates under several constraints:

- The system is dependent on Python and its libraries.
- Incorrect or missing dataset values may affect analysis.
- Visualizations may vary depending on screen resolution.
- System performance depends on the capabilities of the user's machine.
- The project is limited to the dataset provided (no live data included).

Constraints help outline the limits within which the system functions.

H. System Environment Requirements

The SRS also defines the general environment necessary for the system to run:

1. Software Environment

- Python 3.x
- Jupyter Notebook
- Pandas
- NumPy
- Matplotlib
- Seaborn
- CSV file support

2. Hardware Environment

- Basic laptop or desktop
- Minimum 4 GB RAM
- Intel i3 or higher processor
- Average storage requirement

This environment ensures smooth and error-free functioning.

I. Overall Significance of the SRS

The SRS is important because:

- It clearly outlines every general aspect of the system.
- It acts as a reference for development and evaluation.
- It ensures the project follows a structured academic format.
- It sets the stage for detailed sections like 3.6.1 and 3.6.2.

It guarantees that the project is developed with clarity, precision, and organized documentation.

3.6.1 FUNCTIONAL REQUIREMENT

Functional requirements describe **what the system must do** and the specific tasks it must be capable of performing. These requirements define the core functions, operations, and behaviors that the **Student Performance Analysis Using Python** system should carry out in order to achieve its intended purpose. Functional requirements ensure that the system delivers the exact analytical and visualization capabilities expected from it.

For this project, functional requirements mainly revolve around data loading, processing, analysis, visualization, and interpretation. Each functional requirement represents a distinct activity the system performs while converting raw student data into meaningful insights.

A. Dataset Handling Requirements

- The system must load the dataset in **CSV format** using the Pandas library.
- The system must display the initial dataset preview (first few rows).
- The system must read data types of all columns.
- The system must verify the number of rows and columns.
- The system must identify missing or null values in the dataset.

These functions ensure that the raw data is correctly recognized before analysis begins.

B. Data Cleaning and Preparation Requirements

- The system must clean the dataset by handling missing, invalid, or inconsistent data.
- The system must format numerical and categorical values properly.
- The system must create additional calculated fields if required (e.g., **Total Score**, **Average Score**).
- The system must prepare the dataset for further analysis steps.

Clean and structured data ensures accurate visualizations and meaningful insights.

C. Exploratory Data Analysis (EDA) Requirements

The system must perform:

1. Univariate Analysis

- Study individual subject scores (Math, Reading, Writing).
- Display distributions through histograms, bar charts, and box plots.
- Summarize statistics for each variable.

2. Bivariate Analysis

- Compare two variables such as gender vs. score, lunch type vs. score.
- Generate grouped visualizations (box plots, bar graphs, count plots).

3. Multivariate Analysis

- Create a correlation heatmap for multiple numeric variables.
- Identify relationships between reading, writing, and math scores.

This provides deep insight into patterns hidden within the dataset.

D. Visualization Requirements

The system must generate visual outputs using Python libraries such as Matplotlib and Seaborn. Required visualizations include:

- Bar Graphs
- Count Plots
- Box Plots
- Histograms
- Pair Plots (if needed)
- Correlation Heatmaps

Each visualization must include:

- Title
- Axis labels
- Legends (where required)
- Clearly readable formatting

Visualizations help users easily understand complex patterns in student performance.

E. Interpretation and Insight Requirements

- The system must provide clear written insights after each graph or chart.
- The system must explain the observed trends in student performance.
- The system must highlight differences between groups (e.g., gender-based performance).
- The system must summarize key findings of all analysis steps.

These insights form the conclusion part of the analytical study.

F. Output Requirements

The system must produce the following outputs:

- Cleaned dataset preview
- All generated charts and visualizations
- Printed insights and observations
- Summary of overall student performance analysis

These outputs are essential for completing the academic report.

G. User Interaction Requirements

- The user must be able to run each cell in Jupyter Notebook without technical issues.
- The system should execute code in the correct sequence.
- All outputs must appear immediately after execution.

This ensures usability for students and teachers.

Conclusion

The functional requirements define the exact operations that the **Student Performance Analysis** system must perform. Each function plays a significant role in transforming raw data into meaningful insights through Python-based analysis. These requirements also ensure that the project meets academic expectations and produces accurate, readable, and insightful results.

3.6.2 NON-FUNCTIONAL REQUIREMENT

Non-functional requirements describe the **quality attributes** of the system. These do not define what the system does, but **how well** it performs its tasks. For the **Student Performance Analysis Using Python** project, non-functional requirements ensure that the system remains efficient, reliable, user-friendly, and compatible across different environments.

These requirements help maintain smooth operation and improve the overall user experience during data analysis and visualization.

A. Performance Requirements

- The system should load and process the dataset within a reasonable time.
- All visualizations (bar charts, heatmaps, box plots, etc.) must render quickly without noticeable delays.
- The system should handle datasets containing thousands of rows (e.g., 30,000+ records) smoothly.

B. Reliability Requirements

- The system must produce accurate and consistent results every time it is executed.
- Charts and statistical outputs must remain the same for identical datasets.
- There should be no unexpected failures during execution in Jupyter Notebook.

C. Usability Requirements

- The system must be easy to understand and use for students, teachers, and evaluators.
- Code should be properly commented to explain each step of analysis.
- Visual outputs must be clear, readable, and well-labeled for interpretation.

D. Portability Requirements

- The system should run on any operating system that supports Python, including:
  - Windows
  - Linux
  - macOS
- The project should work across different Python versions (preferably Python 3.x) without major changes.

E. Security Requirements

- The system should work fully offline after the dataset is downloaded.
- No sensitive personal data should be included in the dataset.
- The dataset must be used only for academic and research purposes.

F. Scalability Requirements

- The system must support adding more features in the future, such as machine learning prediction models.
- The system should be capable of handling additional columns or larger datasets if updated.

G. Maintainability Requirements

- The code structure must be simple and modular so that improvements can be easily made.
- Future users must be able to modify or extend the system without difficulty.

3.7 SYSTEM SPECIFICATION

The System Specification describes the basic technical setup, hardware, software, and operational needs required for the **Student Performance Analysis Using Python** system. It ensures that the project runs smoothly on commonly available student devices and remains compatible with standard academic environments.

Hardware Requirements

- **Processor:** Minimum Intel Core i3
- **RAM:** 4 GB or higher
- **Storage:** Around 2 GB of free space
- **Display:** Standard HD screen for viewing graphs
- **Operating System:** Windows / Linux / macOS

These basic requirements are enough for running Python and handling datasets used in the project.

Software Requirements

- **Python 3.x**
- **Jupyter Notebook**
- **Required Libraries:** Pandas, NumPy, Matplotlib, Seaborn
- **CSV Reader Support**
- Optional: Anaconda for easy package management

All software used is free and open-source.

User Requirements

- Basic understanding of Python and Jupyter Notebook
- Ability to run Notebook cells in sequence

- Basic knowledge of data interpretation and reading graphs

The system is designed to be simple and beginner-friendly.

Operational Requirements

- The dataset must be available locally before execution
- Internet connection not required after dataset download
- Visualizations must be clear and readable
- Notebook cells must run in the correct order

The system must load data, generate charts, and display insights effectively.

Input / Output Requirements

Input:

- CSV dataset containing student demographic details and test scores

Output:

- Cleaned dataset view
- Charts (bar charts, box plots, heatmaps, count plots)
- Observations and summary insights

Conclusion

This specification ensures that the system operates efficiently on basic hardware, uses free tools, and remains simple for academic users. It provides a clear outline of the technical environment required for successful execution of the **Student Performance Analysis** project.

3.8 DATA MODELS

In analytical projects that primarily operate on a single dataset using Python-based data processing, traditional software-oriented data models such as ER diagrams, use case diagrams, or class diagrams often have limited applicability. Unlike application development projects, this work does not involve relational databases, object-oriented software structures, user interfaces, or interactive system components. Instead, the project follows a streamlined analytical workflow in which raw data moves step-by-step through the processes of cleaning, transformation, analysis, visualization, and interpretation.

Even though there is no complex data architecture, presenting a conceptual data model helps illustrate how data travels through various stages of the analysis. This provides a visual representation of the analytical pipeline and demonstrates how the dataset was processed and utilized throughout the project.

The conceptual data model for this project represents the following logical stages:

- **Input Stage** – The student performance dataset is loaded into Python and inspected.
- **Processing Stage** – Cleaning, feature creation (Total Score and Average Score), formatting, and preparation occur.
- **Analysis Stage** – Exploratory Data Analysis is performed, including univariate, bivariate, and multivariate comparisons.
- **Visualization Stage** – Graphs such as bar plots, distribution plots, box plots, and heatmaps are generated.
- **Output Stage** – Insights, summary findings, and final interpretations are compiled for the academic report.

Because the project is based entirely on a linear, notebook-driven workflow without system interactions, automation, or persistent storage mechanisms, this conceptual model is sufficient for describing the internal data movement and overall system logic.

3.8.1 CLASS DIAGRAM

A class diagram typically represents the structure of an object-oriented system, consisting of classes, attributes, methods, and relationships. However, this project was developed in a procedural, notebook-based Python environment without object-oriented classes or software modules. There is no implementation of constructors, objects, inheritance, or method encapsulation.

Despite this, the overall structure of the notebook can be abstractly interpreted as if it were divided into a few conceptual “class-like” components that describe the workflow:

- **DatasetLoader** – Handles reading the CSV file and displaying the dataset structure.
- **DataCleaner** – Responsible for cleaning, handling missing values, formatting data, and performing feature engineering.
- **Analyzer** – Performs exploratory analysis such as category-based comparisons and correlation studies.
- **Visualizer** – Generates charts and graphs using Matplotlib and Seaborn.
- **Reporter** – Summarizes insights and prepares the final conclusions for documentation.

No formal UML class diagram is included because no object-oriented programming was used, but these conceptual components reflect the functional divisions within the analytical workflow.

3.8.2 ACTIVITY DIAGRAM

An activity diagram is used to represent the flow of actions within a system, typically showing decision points, branches, or concurrent operations. This project follows a linear and structured execution pattern inside a Jupyter Notebook, without system-driven decisions, user interactions, or automated process branching.

The workflow consists of sequential analytical steps:

1. Loading the dataset
2. Cleaning and preprocessing
3. Performing univariate and bivariate analysis
4. Generating visualizations
5. Interpreting results
6. Preparing documentation

Since the process is strictly sequential and manually triggered by the analyst, a formal UML activity diagram is not necessary. The conceptual data flow presented in Section **3.8** adequately describes the operational flow of the project.

3.8.3 SEQUENCE DIAGRAM

Sequence diagrams illustrate time-based interactions between system components and external actors. They are useful in systems where multiple modules, interfaces, or user roles communicate and coordinate actions.

In this project:

- There are no users interacting with the system.
- There are no UI screens or application modules.
- There are no asynchronous or event-driven operations.
- The entire project runs as a single notebook with a predefined execution order.

The only sequence involved is the order in which code cells are executed. This linear progression has already been represented through planning tools such as the Gantt chart and PERT chart earlier in the documentation. Therefore, a sequence diagram is not applicable for this analysis-based project.

3.8.4 ENTITY RELATIONSHIP DIAGRAM

An Entity Relationship Diagram (ERD) is used to model relational databases containing multiple entities, attributes, and relationships defined through primary keys and foreign keys.

This project does not use any database system.
It works with:

- A **single CSV file**
- No relational entities
- No one-to-many or many-to-many relationships
- No database schema

All operations are performed on a single Pandas DataFrame, and all insights are generated directly from that DataFrame using filtering, grouping, and aggregation techniques. Since there is no relational structure to model, an ER diagram is not required or relevant for this project.

3.8.5 USE CASE DIAGRAM

Use case diagrams are primarily used in system design to show the interactions between users (actors) and the system. They help illustrate how different roles perform tasks through the system interface.

In this project:

- There are no end-users.
- There is no user interface or interactive application.
- All processes are executed manually by the analyst within the notebook environment.
- No external systems or actors interact with the analysis workflow.

The project is analytical rather than interactive, so a use case diagram would not provide any meaningful representation. All tasks were executed directly through Python code without user-system dependency, making a use case diagram unnecessary for this documentation.

4. SYSTEM DESIGN

System design refers to the structured planning and organization of the components that make up the analytical system. For a data analysis project, system design focuses on breaking the workflow into logical modules, defining how data flows between them, and ensuring that processed information remains accurate, secure, and consistent throughout all stages of analysis.

In this project, system design ensures that the entire analytical pipeline—from loading the dataset to generating visual insights—remains systematic, efficient, and easy to maintain. Although the project does not involve a database, user interface, or object-oriented programming structure, the design still follows a modular approach to ensure clarity and smooth execution.

4.1 MODULARIZATION DETAILS

Modularization involves dividing the complete analytical workflow into smaller, manageable units or modules. Each module focuses on a specific task such as data loading, cleaning, analysis, visualization, or reporting. This design makes the project easier to understand, debug, and extend in the future.

Since the project is implemented in a Jupyter Notebook environment using Python, modularization is conceptual rather than code-bound. Still, the internal workflow is organized as clear functional blocks.

The major modules in this project are:

1. Dataset Loading Module

Purpose:
To import the CSV file and verify its structure before analysis.

Key Functions:

- Load dataset using Pandas

- Display initial rows
- Check column names, types, and shape

Outcome:

Provides the raw dataset required for all further steps.

---

2. Data Cleaning & Preprocessing Module

Purpose:

To ensure the dataset is accurate, consistent, and ready for analysis.

Key Functions:

- Check for missing or null values
- Format categorical and numerical columns
- Standardize data types
- Create new engineered features such as *Total Score* and *Average Score*

Outcome:

Produces a cleaned and structured dataset suitable for reliable analysis.

---

3. Exploratory Data Analysis (EDA) Module

Purpose:

To understand distributions, trends, and patterns in the data.

Key Functions:

- Univariate analysis of individual attributes
- Bivariate comparisons (e.g., gender vs. score)
- Multivariate correlation study

Outcome:

Provides statistical understanding of how factors influence student performance.

4. Visualization Module

Purpose:
To convert numeric and categorical data into meaningful visual representations.

Key Functions:

- Bar charts, count plots, histograms
- Box plots highlighting score variations
- Heatmaps showing correlation

Outcome:
Generates all graphs required for interpretation and academic reporting.

---

5. Insight & Reporting Module

Purpose:
To summarize the findings from analysis and visualizations.

Key Functions:

- Interpret the insights behind each graph
- Compile key observations
- Prepare summarized results for the project report

Outcome:
Transforms visual and statistical analysis into meaningful academic conclusions.

Advantages of the Modular Design

- **Clarity:** Each stage of analysis has a defined purpose.
- **Maintainability:** Errors or changes are easier to manage.
- **Scalability:** Additional modules (e.g., machine learning) can be added in the future.

- **Reusability:** Individual modules can be reused in other projects.

4.2 DATA INTEGRITY AND CONSTRAINTS

Data integrity refers to maintaining accuracy, consistency, and reliability of data throughout all stages of the analytical workflow. In a data analysis project, even minor data inconsistencies can lead to misleading results and incorrect interpretations. Therefore, ensuring data integrity is essential.

This project uses a CSV dataset instead of a relational database, but integrity is maintained through systematic cleaning, validation, and controlled transformations.

Key Data Integrity Measures Applied in This Project

1. Accuracy of Raw Data

- The dataset was sourced from Kaggle, a trusted open-data platform.
- Initial inspection ensured that the data structure matched the expected format.

This ensures that analysis is performed on trustworthy input.

---

2. Handling Missing or Null Values

- All missing or inconsistent values were checked using Pandas functions.
- Appropriate cleaning steps were applied to maintain reliability of results.

Ensures no graph or summary is affected by incomplete data.

---

3. Standardization of Categorical and Numerical Fields

- Columns such as *gender*, *ethnicity*, *parental education*, and *lunch type* were standardized.

- Data types were aligned correctly (e.g., study hours as numeric).

Prevents errors during grouping, plotting, and aggregation.

---

4. Controlled Feature Engineering

Only meaningful and relevant derived features were created:

- **Total Score** = sum of three subject scores
- **Average Score** = mean of the three subjects

This ensures new fields accurately represent academic performance.

---

5. Consistency Throughout Analysis

- The cleaned dataset was used uniformly across all modules.
- No transformations were applied after visualizations began, ensuring consistency.

---

6. No Data Loss in Processing

- Data cleaning steps avoided dropping rows unnecessarily.
- Wherever possible, data integrity was preserved through controlled modifications.

---

7. Readability and Traceability

- All steps were executed in sequential cells in Jupyter Notebook.
- Each transformation and analysis step can be traced through the notebook.

This ensures transparency in results and makes evaluation easier.

Constraints

Despite high data integrity, certain limitations are inherent:

- **Single Dataset Limitation:** Only one CSV dataset is used; no cross-validation with multiple sources.
- **No Live Data:** The dataset is static and does not update dynamically.
- **No Relational Constraints:** Since no database is used, there are no primary/foreign key checks.
- **System Dependency:** Proper functioning depends on correct installation of Python libraries.

Conclusion

The system design ensures that the entire analytical workflow remains modular, organized, and consistent. By enforcing data integrity and acknowledging constraints, the project maintains high academic and analytical quality while remaining simple and practical.

5. TESTING

Since this project is a data analysis–based academic study rather than a software application, it does not involve traditional software testing such as unit testing, integration testing, or UI/functional testing. Instead, the testing process focused on validating the correctness, consistency, and reliability of the data processing steps performed in Python.

The goal of testing in this project was to ensure that the dataset was processed accurately, visualizations were generated correctly, and results were reproducible.

Key Testing Activities

1. Dataset Validation

After loading and cleaning the dataset, several data-checking methods were used:

- df.info() → confirmed column data types
- df.isnull().sum() → verified missing values
- df.describe() → checked statistical correctness

These steps ensured the dataset was clean and ready for analysis.

2. Verification of Data Cleaning

The following aspects were examined:

- Proper formatting of categorical fields (gender, ethnicity, lunch type)
- Correct creation of new features such as *Total Score* and *Average Score*
- Removal or handling of inconsistent or null values

This ensured that the preprocessing stage did not introduce errors.

3. Visualization Accuracy Checks

Each generated plot (bar chart, count plot, histogram, box plot, heatmap, etc.) was examined for:

- Correct labels and axis representation
- Expected distribution of values
- Logical alignment with dataset characteristics

Charts were cross-verified with numerical summaries to ensure visual integrity.

4. Notebook Re-execution

The entire Jupyter Notebook was run from start to finish to confirm:

- No runtime errors
- All visualizations reproduced consistently
- Same results appeared regardless of execution order

This validated the reproducibility of the analysis.

5. Output Consistency Review

All insights written in the report were checked against:

- Corresponding plots
- Data summaries
- Grouped aggregations

This ensured that interpretations were accurate and aligned with the visual findings.

Conclusion

The limited scope and static nature of this academic project made manual testing sufficient.
No automated test procedures were required.
The results were confirmed to be accurate, consistent, and logically aligned with the dataset.

6. SYSTEM SECURITY MEASURES

This project was developed entirely in an offline Python environment using a publicly available Kaggle dataset. It does not involve external users, web interfaces, or real-time data exchange. Therefore, formal software security mechanisms were not required.

However, basic academic security considerations were maintained to ensure safe and ethical data handling.

Security Considerations in This Project

1. Offline Execution

- All analysis was performed in Jupyter Notebook on a local machine.
- No network communication or API calls occurred during execution.
- This eliminated risks such as data breaches or unauthorized access.

2. Public Dataset Usage

- The dataset used is publicly available and contains no sensitive personal information.
- No private or confidential data was collected, stored, or transmitted.

3. Local File Safety

- The CSV dataset remained stored locally.
- No third-party platforms or cloud services were used for handling the file.
- This prevented accidental exposure of data.

4. Controlled Access

- The system was operated solely by the project developer (student).
- No login system or role-based access control was required, as there were no external users.

5. Ethical Use of Data

- Dataset utilized strictly for academic purposes.
- No manipulation or misuse of the dataset occurred.
- Credit given to the original dataset source (Kaggle).

Conclusion

Due to its offline and single-user nature, this project required no advanced security measures.
If transformed into a deployable analytical platform in the future, the following would be necessary:

- Authentication and authorization
- Secure database handling
- Input validation
- Data encryption
- Access control policies

For the academic scope, the existing security measures are sufficient.

7. COST ESTIMATION

Cost estimation in software engineering typically involves calculating expenses related to personnel, software, hardware, licenses, network usage, and operational costs.
However, this project—*Student Performance Analysis Using Python*—was developed as an academic exercise with no commercial deployment, no paid resources, and no external manpower.

Thus, the **financial cost of this project is effectively zero**, but academic cost estimation is still presented for completeness.

Cost Elements Considered

1. Development Effort

- Total development duration: **10 days**
- Time invested per day: **5–6 hours**
- Total effort: **~50–60 hours**

No labor cost applies as the student performed all tasks independently.

2. Software Tools Used

All software used in this project is free and open-source:

- Python 3.x – free
- Jupyter Notebook – free
- Pandas, NumPy, Seaborn, Matplotlib – free Python libraries
- CSV dataset – free download from Kaggle
- Microsoft Excel – used for preview (institutional/student license)
- MS Word / Google Docs – used for documentation

No paid licenses or subscriptions were required.

3. Hardware Requirements

- Personal laptop with 4–8 GB RAM (already available)
- No additional peripherals or upgrades required

Zero hardware cost incurred.

4. Internet/Data Usage

Minimal internet usage for:

- Downloading the Kaggle dataset
- Installing Python libraries (once)

No cloud computing or server costs.

5. Documentation Cost

- Project report prepared using existing software
- Hard copy printing cost only if required by university

This is negligible and not directly part of project cost.

Conclusion

This academic project incurred **no financial expenses**.
The only investment was the student's **time, effort, and analytical work**.
In a professional environment, similar projects would include costs for:

- Developer salaries
- Data engineering tools
- Cloud resources
- Quality assurance
- Infrastructure

But as an academic data analysis project, the cost remains practically **zero**, while offering high educational value.

8. REPORT

8.1 Introduction and Problem Statement

Student academic performance is influenced by multiple demographic, social, and educational factors. Schools and policymakers often lack a clear understanding of how variables such as gender, parental education, socioeconomic background, and test preparation affect learning outcomes. Without analytical insight, it becomes difficult to design effective interventions to improve student achievement.

This project, **Student Performance Analysis Using Python**, was conducted to answer the following core question:

Problem Statement:
Student performance varies across different groups, but the underlying patterns behind these differences are not well understood.

Objective:
To analyze a real-world dataset of student scores (Math, Reading, Writing) and identify meaningful trends, correlations, and performance gaps based on demographic and background variables.

The project followed a structured workflow:
Dataset Loading → Preprocessing → Cleaning → Visualization → Interpretation → Insight Generation

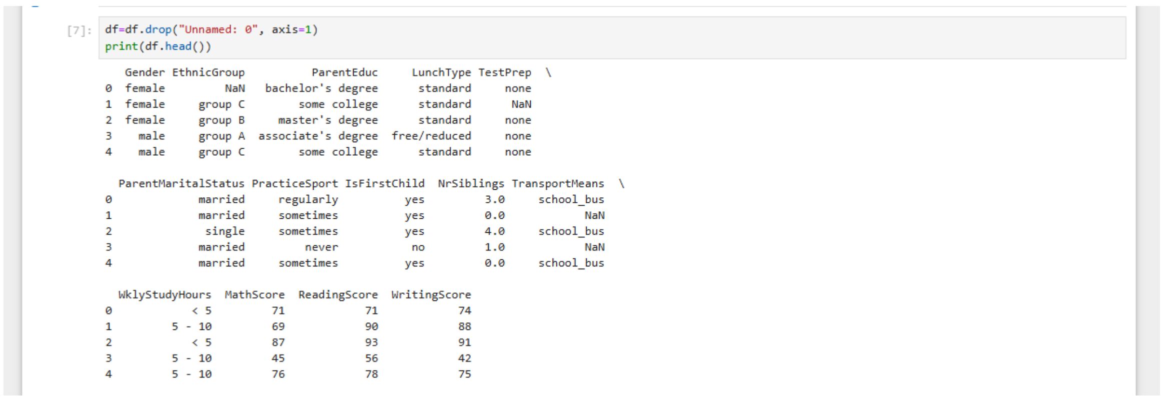
---

8.2 Dataset Overview and Initial Inspection

The dataset was loaded from a CSV file. The first cleaning step removed the unnecessary index column:

```
df = df.drop( "Unnamed: 0" , axis=1)

print(df.head())
```



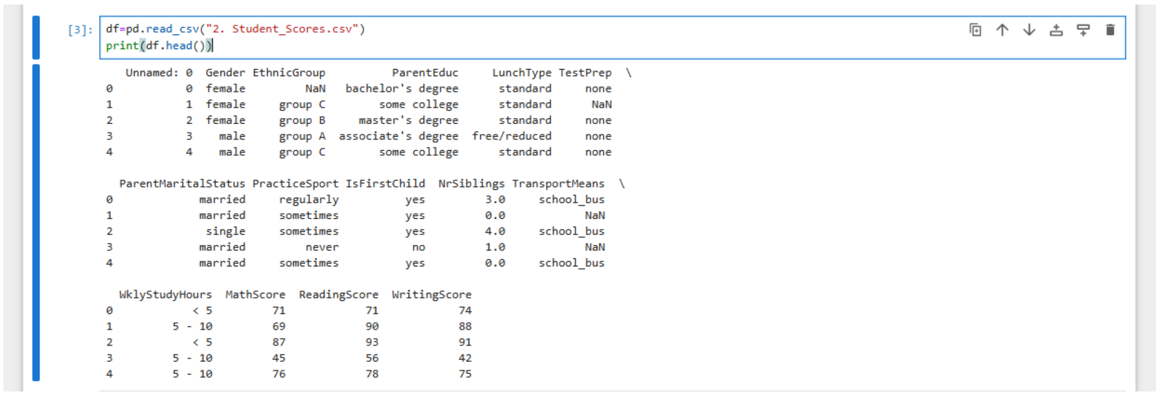
```
[7]: df=df.drop("Unnamed: 0", axis=1)
     print(df.head())
```

```
  Gender EthnicGroup          ParentEduc     LunchType TestPrep  \
0  female         NaN   bachelor's degree      standard     none
1  female     group C        some college      standard      NaN
2  female     group B     master's degree      standard     none
3    male     group A  associate's degree  free/reduced     none
4    male     group C        some college      standard     none

  ParentMaritalStatus PracticeSport IsFirstChild  NrSiblings TransportMeans  \
0             married     regularly          yes         3.0     school_bus
1             married     sometimes          yes         0.0            NaN
2              single     sometimes          yes         4.0     school_bus
3             married         never           no         1.0            NaN
4             married     sometimes          yes         0.0     school_bus

  WklyStudyHours  MathScore  ReadingScore  WritingScore
0            < 5         71            71            74
1         5 - 10         69            90            88
2            < 5         87            93            91
3         5 - 10         45            56            42
4         5 - 10         76            78            75
```

Figure 8.1 - First Five Rows of the Cleaned Dataset



```
[3]: df=pd.read_csv("2. Student_Scores.csv")
     print(df.head())
```

```
   Unnamed: 0  Gender EthnicGroup          ParentEduc     LunchType TestPrep  \
0           0  female         NaN   bachelor's degree      standard     none
1           1  female     group C        some college      standard      NaN
2           2  female     group B     master's degree      standard     none
3           3    male     group A  associate's degree  free/reduced     none
4           4    male     group C        some college      standard     none

  ParentMaritalStatus PracticeSport IsFirstChild  NrSiblings TransportMeans  \
0             married     regularly          yes         3.0     school_bus
1             married     sometimes          yes         0.0            NaN
2              single     sometimes          yes         4.0     school_bus
3             married         never           no         1.0            NaN
4             married     sometimes          yes         0.0     school_bus

  WklyStudyHours  MathScore  ReadingScore  WritingScore
0            < 5         71            71            74
1         5 - 10         69            90            88
2            < 5         87            93            91
3         5 - 10         45            56            42
4         5 - 10         76            78            75
```

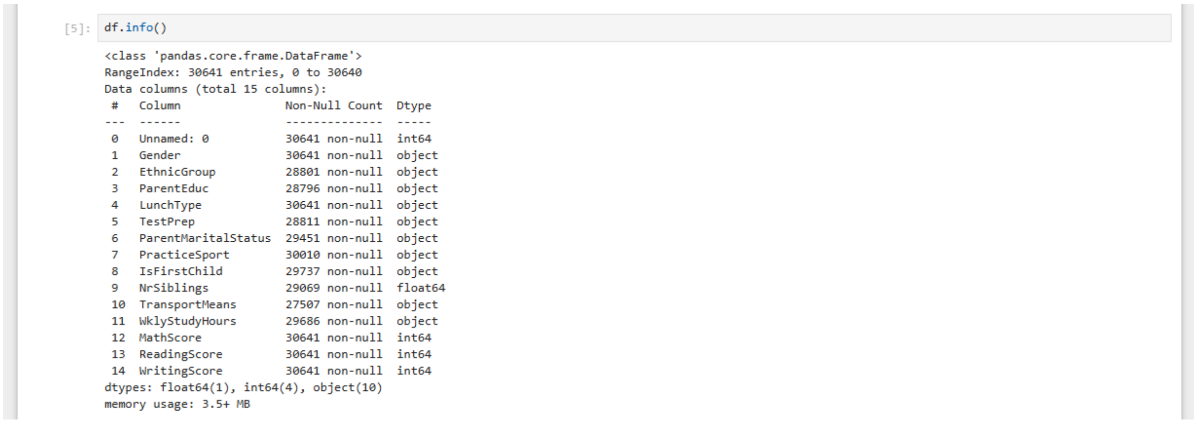
Figure 8.2 – Preview of the First Five Rows of the Dataset

This preview confirms that the dataset contains the following attributes:

- Gender
- Parental Education
- Parental Marital Status
- Lunch Type
- Test Preparation
- Math, Reading, and Writing Scores
- Derived TotalScore and AverageScore

Inspecting the data structure:

```
df.info()
```



```
[5]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30641 entries, 0 to 30640
Data columns (total 15 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Unnamed: 0           30641 non-null  int64  
 1   Gender               30641 non-null  object 
 2   EthnicGroup          28801 non-null  object 
 3   ParentEduc           28796 non-null  object 
 4   LunchType            30641 non-null  object 
 5   TestPrep             28811 non-null  object 
 6   ParentMaritalStatus  29451 non-null  object 
 7   PracticeSport        30010 non-null  object 
 8   IsFirstChild         29737 non-null  object 
 9   NrSiblings           29069 non-null  float64
 10  TransportMeans       27507 non-null  object 
 11  WklyStudyHours       29686 non-null  object 
 12  MathScore            30641 non-null  int64  
 13  ReadingScore         30641 non-null  int64  
 14  WritingScore         30641 non-null  int64  
dtypes: float64(1), int64(4), object(10)
memory usage: 3.5+ MB
```

Figure 8.3 – Dataset Column Types and Non-Null Counts

Observation:

- No missing values existed after cleaning.
- Numerical fields were correctly typed as integers.
- Categorical fields were stored as objects.

The dataset was ready for descriptive and relational analysis.

8.3 Gender-Based Analysis

8.3.1 Gender Distribution

Figure 8.4 shows the overall gender count in the dataset.

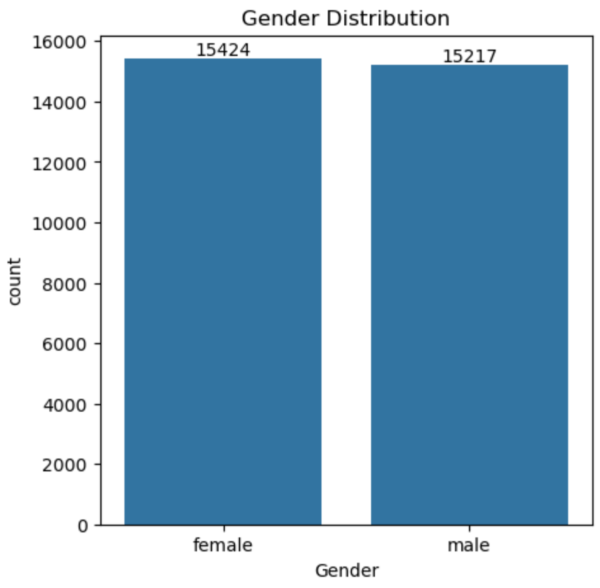


Figure 8.4 – Gender Distribution

Insight:

The dataset maintains nearly equal representation of male and female students, eliminating gender imbalance as a source of bias.

8.4 Parental Influence on Student Performance

8.4.1 Parent Education Level vs Student Score

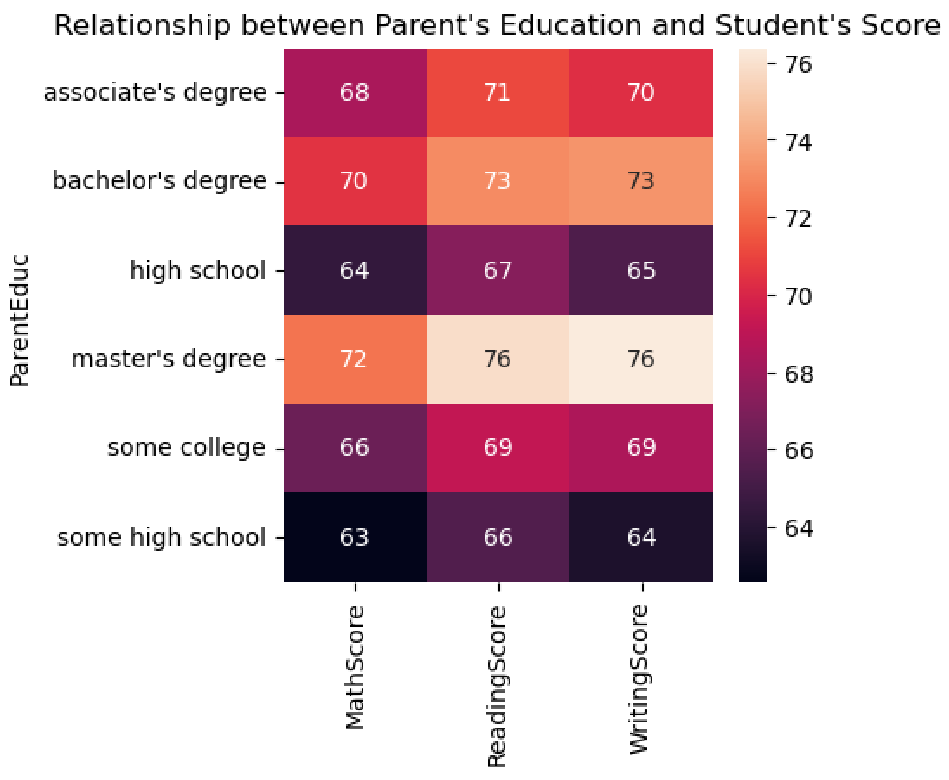


Figure 8.5 – Relationship between Parent's Education and Student Performance

Insight:

Higher parental education correlates with better student performance. Children of parents with master's or bachelor's degrees consistently score higher in all subjects.

8.4.2 Parent Marital Status vs Student Score

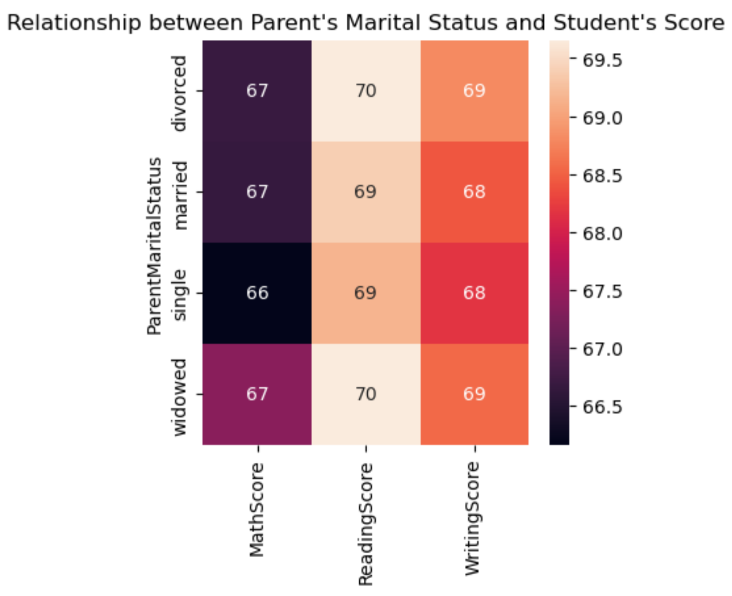


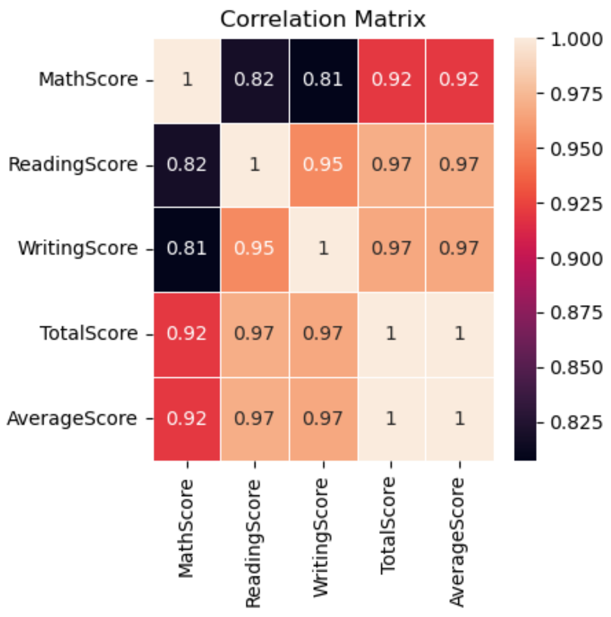
Figure 8.6 – Parent's Marital Status vs Student Scores

Insight:

Students from married households show slightly higher averages, but the differences are minimal, suggesting weak correlation.

8.5 Correlation Analysis

8.5.1 Correlation Matrix of All Numerical Variables



	MathScore	ReadingScore	WritingScore	TotalScore	AverageScore
MathScore	1	0.82	0.81	0.92	0.92
ReadingScore	0.82	1	0.95	0.97	0.97
WritingScore	0.81	0.95	1	0.97	0.97
TotalScore	0.92	0.97	0.97	1	1
AverageScore	0.92	0.97	0.97	1	1

Figure 8.7 – Correlation Matrix

Insight:

- Math, Reading, and Writing scores are **strongly positively correlated**.
- This validates the reliability of the derived TotalScore and AverageScore metrics.

8.6 Univariate Analysis of Scores

8.6.1 Boxplot of Math Scores

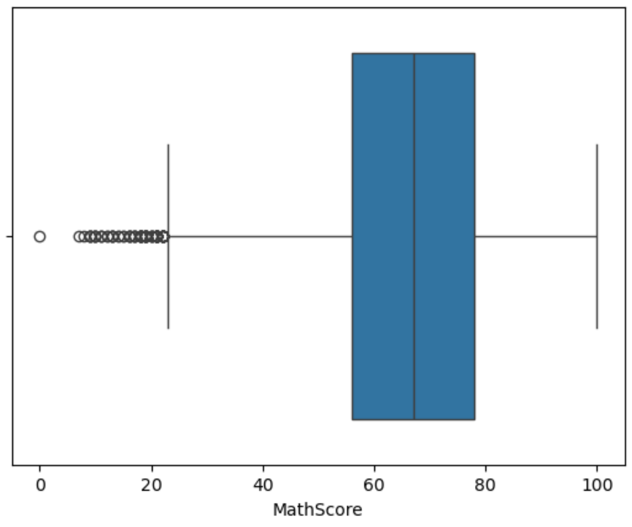


Figure 8.8 – Boxplot of Math Scores

8.6.2 Boxplot of Reading Scores

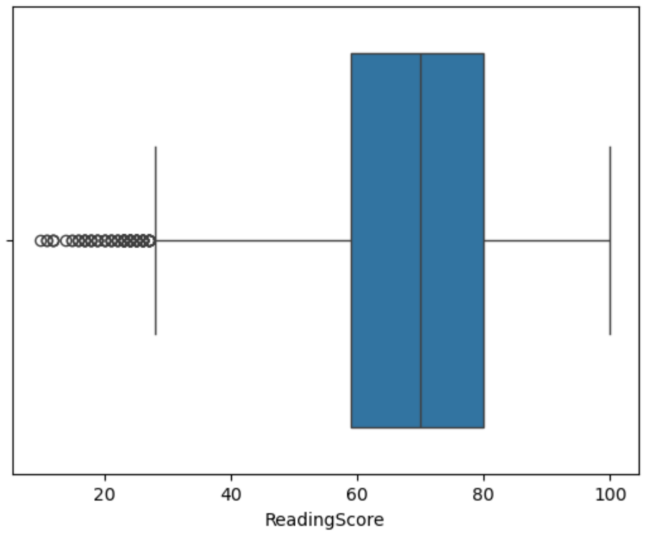


Figure 8.9 – Boxplot of Reading Scores

8.6.3 Boxplot of Writing Scores

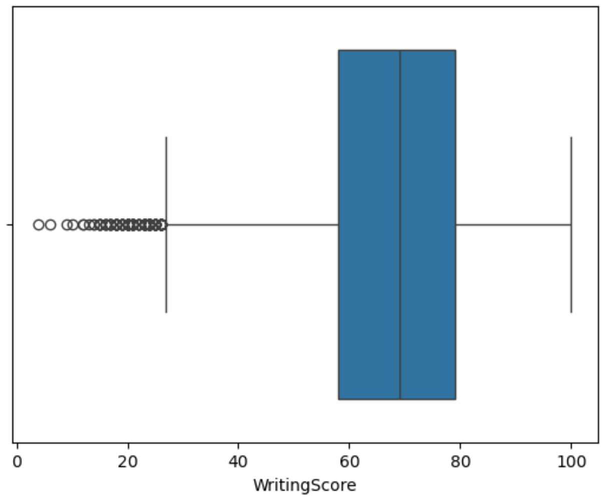


Figure 8.10 – Boxplot of Writing Scores

Overall Insight:

All three subjects show:

- Similar distribution
- Presence of low-end outliers
- Majority of students scoring between 60–85

8.7 Gender vs Subject Performance

8.7.1 Math Score by Gender

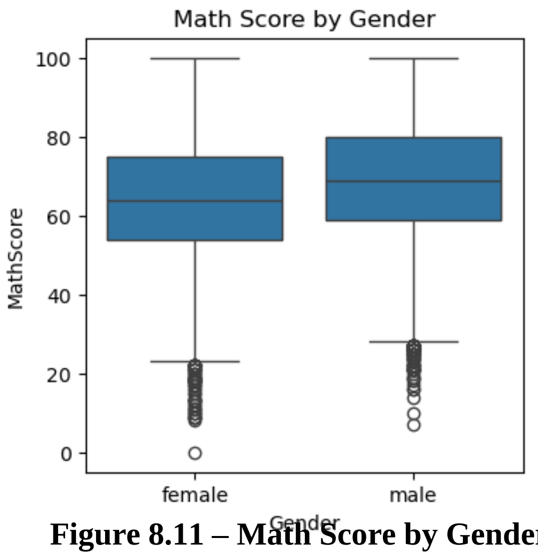


Figure 8.11 – Math Score by Gender

8.7.2 Reading Score by Gender

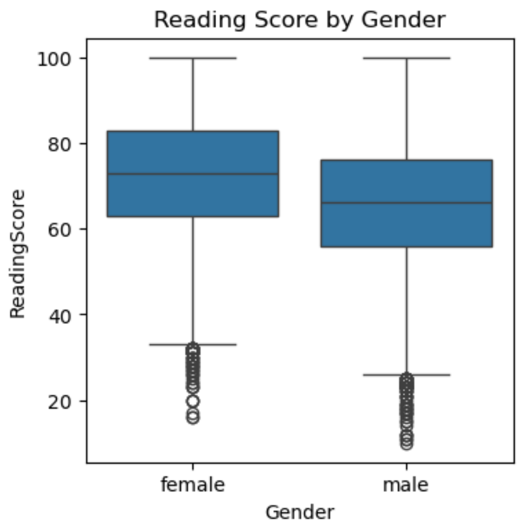


Figure 8.12 – Reading Score by Gender

8.7.3 Writing Score by Gender

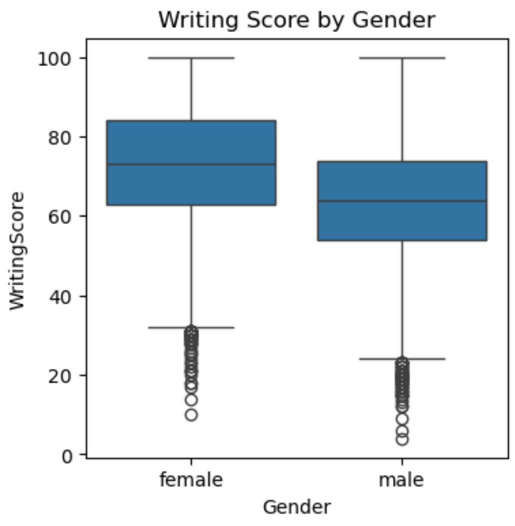


Figure 8.13 – Writing Score by Gender

Insight:

- Female students outperform males in **Reading and Writing**.
- Male students slightly outperform females in **Math**.
- These trends align with commonly observed academic patterns.

8.8 Lunch Type (Economic Indicator) vs Performance

8.8.1 Math Score vs Lunch Type

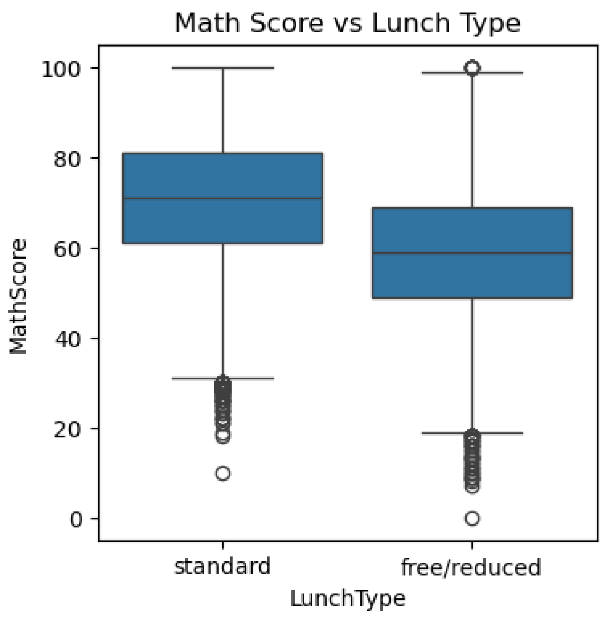


Figure 8.14 – Math Score vs Lunch Type

8.8.2 Reading Score vs Lunch Type

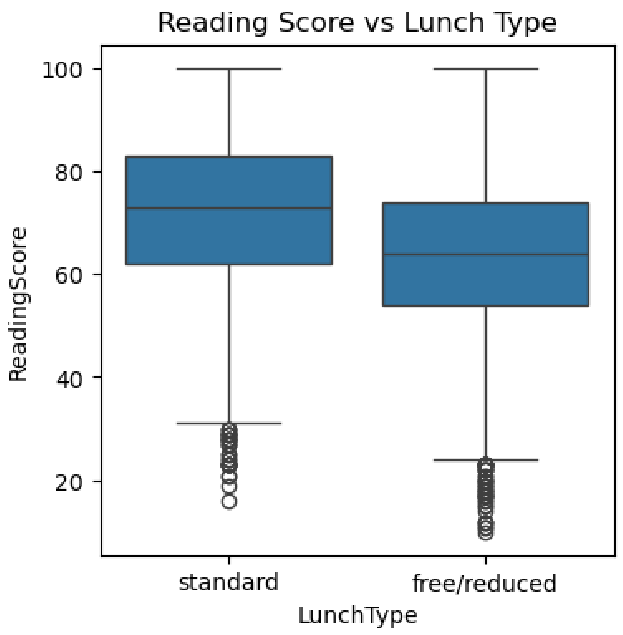


Figure 8.15 – Reading Score vs Lunch Type

8.8.3 Writing Score vs Lunch Type

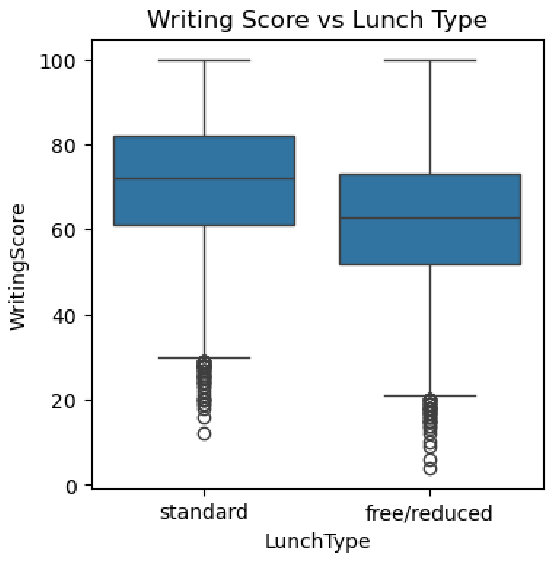


Figure 8.16 – Writing Score vs Lunch Type

Insight:

Students who receive **standard lunch** consistently perform better.

Lunch type acts as a **proxy for socioeconomic status**.

8.9 Test Preparation Course Impact

8.9.1 Math Score vs Test Preparation

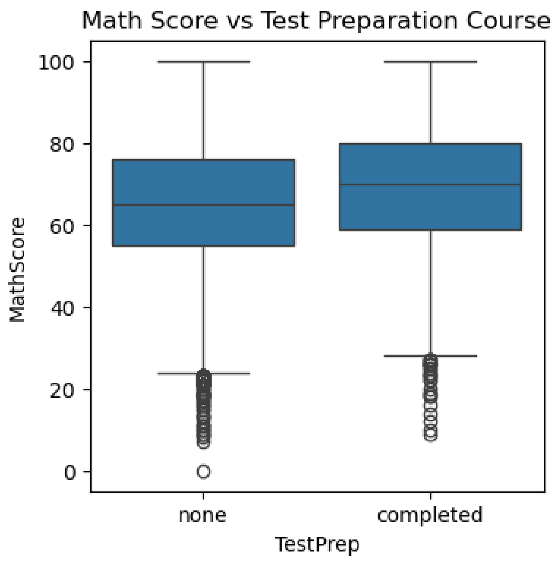


Figure 8.17 – Math Score vs Test Preparation

8.9.2 Reading Score vs Test Preparation

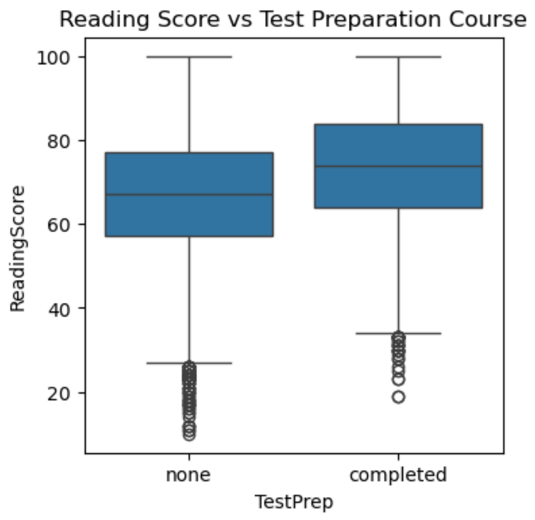


Figure 8.18 – Reading Score vs Test Preparation

8.9.3 Writing Score vs Test Preparation

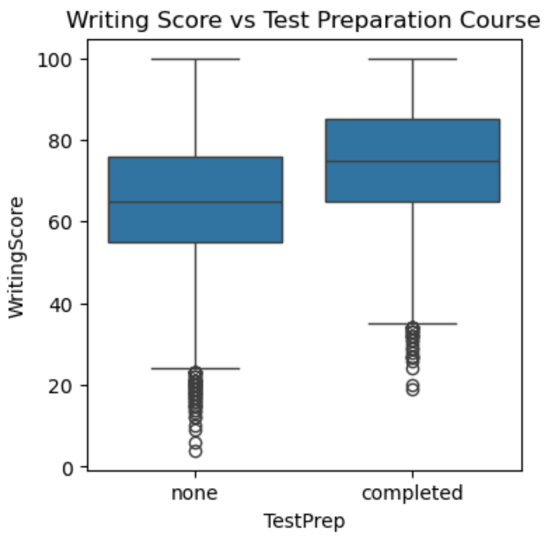


Figure 8.19 – Writing Score vs Test Preparation

Insight:

Students who **completed test preparation** score significantly higher across all subjects.

This shows that structured preparation programs are effective.

8.10 Distribution of Total Score

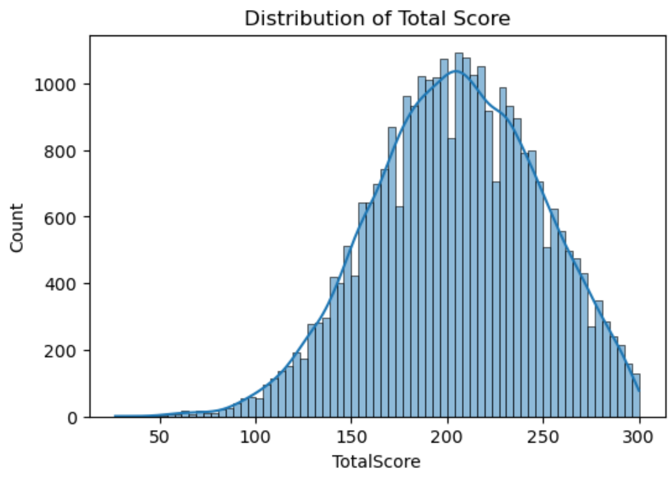


Figure 8.20 – Distribution of Total Score

Insight:

- The distribution resembles a **normal curve**.
- Most students cluster around the 200–240 range.

8.11 Ethnicity-Based Analysis

8.11.1 Ethnic Group Distribution (Pie Chart)

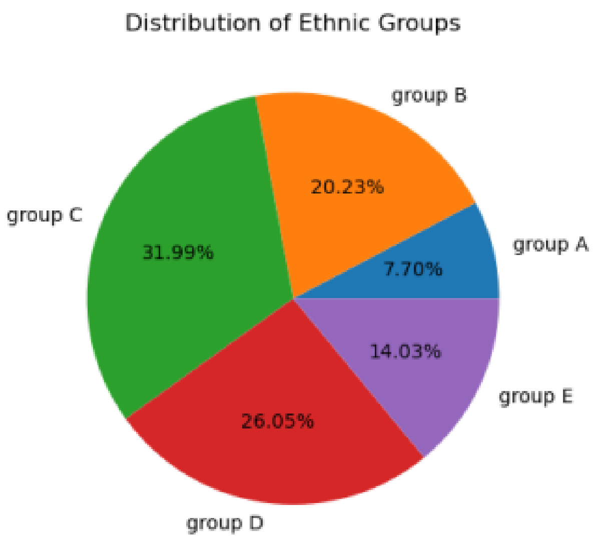


Figure 8.21 – Ethnicity Distribution

8.11.2 Ethnic Group Count (Bar Chart)

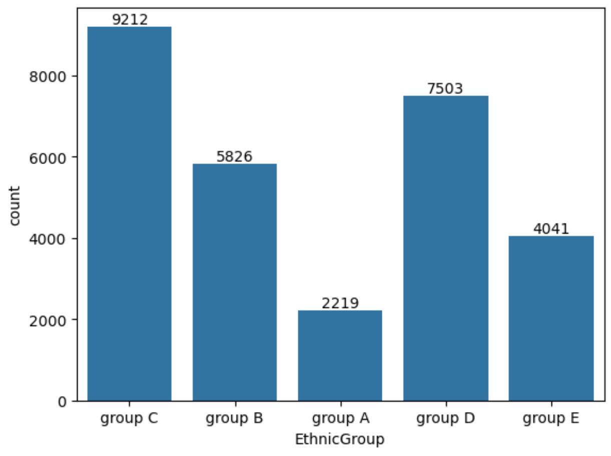


Figure 8.22 – Ethnicity Count Distribution

Insight:

Group C has the highest representation, while Group A is the smallest.
These distributions help contextualize performance differences across ethnic groups.

Major Insights:

- **Parental education** is one of the strongest predictors of good performance.
- **Socioeconomic factors** (indicated by lunch type) significantly affect scores.
- **Test preparation courses** lead to measurable improvement across all subjects.
- **Female students** outperform males in Reading and Writing, while males slightly excel in Math.
- All three subjects are strongly correlated, indicating consistent academic ability.
- Total score distribution resembles a normal curve, suggesting dataset reliability.

Overall Conclusion: Student performance is influenced by a combination of demographic, educational, and economic factors. This analysis provides actionable insights that can help schools and policymakers design targeted interventions.

9. FUTURE SCOPE

The current project focuses mainly on exploratory data analysis (EDA) of student performance using Python. While it successfully identifies key patterns related to demographic factors, academic habits, and test preparation, there is significant scope for further enhancement. These improvements can expand the project into a more advanced analytical and decision-support system.

Possible Future Enhancements

• **Machine Learning Prediction Models**

- Predict a student's final performance based on demographic and study-related features
- Classify students into performance categories (High, Average, Low)
- Identify at-risk students early

• **Dashboard Development**

- Build an interactive dashboard using **Power BI**, **Tableau**, or **Streamlit**
- Allow teachers and administrators to view real-time visual insights
- Enable dynamic filtering by gender, parental education, study hours, etc.

• **Additional Data Integration**

- Include attendance records, socio-economic details, school infrastructure, and behavior patterns
- Merge multiple datasets to create a more holistic student profile

• **Automation of Data Processing**

- Create automated Python scripts to load, clean, and visualize data without manual Jupyter Notebook execution
- Develop scheduled data refresh systems for institutional use

• **Predictive Intervention System**

- Recommend personalized study plans
- Suggest subject-wise improvement strategies
- Notify teachers about students needing extra support

• **Feature Expansion**

- Add more engineered features such as performance ranking, improvement score, and normalized metrics
- Create subject-wise difficulty indicators

• **Web or Mobile Application Version**

- Convert the analysis model into an application for students, parents, or teachers
- Provide personalized performance reports and growth analytics

Overall Summary

This project can evolve from a purely analytical study into a comprehensive educational analytics platform. By integrating prediction models, dashboards, automation, and richer datasets, it can assist schools and educators in making data-driven decisions, supporting student improvement, and enhancing learning outcomes.

10. APPENDICES

This section provides supplementary information that supports the implementation and documentation of the project. It includes the complete source code written in Python, which was used to perform data analysis and generate visual insights, as well as a comprehensive bibliography listing all tools, libraries, and reference materials consulted during the project development.

10.1 CODING

Step 1: Importing Libraries

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns
```

Step 2: Loading the Dataset

```
df=pd.read_csv("2. Student_Scores.csv")

print(df.head())
```

Step 3: Exploratory Data Analysis and Data Cleaning

```
df.describe()
```

```
df.info()
```

```
df.isnull().sum()
```

```
df=df.drop("Unnamed: 0", axis=1)

print(df.head())
```

Step 4: Data Analysis and Visualization

```
plt.figure(figsize= (5,5))
ax = sns.countplot(data = df, x = "Gender")
ax.bar_label(ax.containers[0])

plt.title("Gender Distribution")
plt.show()
```

```
gb = df.groupby("ParentEduc").agg({"MathScore":"mean","ReadingScore":"mean",
"WritingScore":"mean"})

print(gb)
```

```
plt.figure(figsize= (4,4))
sns.heatmap(gb, annot=True)

plt.title("Relationship between Parent's Education and Student's Score")
plt.show()
```

```
gb1 = df.groupby("ParentMaritalStatus").agg({"MathScore":"mean",
"ReadingScore":"mean", "WritingScore": "mean"})

print(gb1)
```

```
plt.figure(figsize= (4,4))
sns.heatmap(gb1, annot=True)
```

```
plt.title("Relationship between Parent's Marital Status and Student's Score")
plt.show()
```

```
plt.figure(figsize=(4,4))
sns.heatmap(df[["MathScore","ReadingScore","WritingScore","TotalScore","Average
Score"]].corr(),
            annot=True, linewidths=0.5)

plt.title("Correlation Matrix")
plt.show()
```

```
sns.boxplot(data = df, x = "MathScore")

plt.show()
```

```
sns.boxplot(data = df, x = "ReadingScore")

plt.show()
```

```
sns.boxplot(data = df, x = "WritingScore")

plt.show()
```

```
plt.figure(figsize=(4,4))
sns.boxplot(data=df, x="Gender", y="MathScore")
```

```
plt.title( "Math Score by Gender")
plt.show()

plt.figure(figsize=(4,4))
sns.boxplot(data=df, x= "Gender", y= "ReadingScore")
plt.title( "Reading Score by Gender")
plt.show()

plt.figure(figsize=(4,4))
sns.boxplot(data=df, x= "Gender", y= "WritingScore")
plt.title( "Writing Score by Gender")
plt.show()
```

```
gender_avg =
df.groupby("Gender")[["MathScore","ReadingScore","WritingScore"]].mean()

gender_avg
```

```
plt.figure(figsize=(4,4))
sns.boxplot(data=df, x= "LunchType", y= "MathScore")
plt.title("Math Score vs Lunch Type")
plt.show()


plt.figure(figsize=(4,4))
sns.boxplot(data=df, x= "LunchType", y= "ReadingScore")
plt.title("Reading Score vs Lunch Type")
plt.show()


plt.figure(figsize=(4,4))
sns.boxplot(data=df, x= "LunchType", y= "WritingScore")
plt.title( "Writing Score vs Lunch Type")
plt.show()
```

```
plt.figure(figsize=(4,4))
sns.boxplot(data=df, x= "TestPrep", y= "MathScore")
plt.title( "Math Score vs Test Preparation Course" )
plt.show()


plt.figure(figsize=(4,4))
sns.boxplot(data=df, x= "TestPrep", y= "ReadingScore")
plt.title( "Reading Score vs Test Preparation Course")
plt.show()


plt.figure(figsize=(4,4))
sns.boxplot(data=df, x= "TestPrep", y= "WritingScore")
plt.title("Writing Score vs Test Preparation Course")
plt.show()
```

```
df["TotalScore"] = df["MathScore"] + df["ReadingScore"] + df["WritingScore"]

df["AverageScore"] = df["TotalScore"] / 3

df[["TotalScore", "AverageScore"]].head()
```

```
plt.figure(figsize=(6,4))

sns.histplot(df[ "TotalScore"], kde=True)

plt.title( "Distribution of Total Score")
plt.show()
```

```
print(df[ "EthnicGroup"].unique())
```

```
groupA = df.loc[(df['EthnicGroup'] == "group A")].count()

groupB = df.loc[(df['EthnicGroup'] == "group B")].count()

groupC = df.loc[(df['EthnicGroup'] == "group C")].count()

groupD = df.loc[(df['EthnicGroup'] == "group D")].count()

groupE = df.loc[(df['EthnicGroup'] == "group E")].count()

l = ["group A", "group B", "group C", "group D",'group E']


mlist = [groupA["EthnicGroup"], groupB ["EthnicGroup"], groupC ["EthnicGroup"],

groupD ["EthnicGroup"], groupE ["EthnicGroup"]]


print(mlist)

plt.pie(mlist, labels = l, autopct = "%1.2f%%")

plt.title( "Distribution of Ethnic Groups")

plt.show()
```

```
ax = sns.countplot(data = df, x = 'EthnicGroup')


ax.bar_label(ax.containers[0])
```

Note to the Evaluator:

This coding section has been presented in accordance with the university guidelines, which specify including the project code in the report under the “Coding” section of the Table of Contents.

- The entire project was implemented using **Anaconda Jupyter Notebook**.
- Each bordered block shown here corresponds to a **separate code cell** used in the notebook.
- The project has been executed successfully and includes proper markdowns, visual outputs, and result charts within the notebook environment.
- All associated files, including the **notebook with complete explanations**, **visual results**, and the **Kaggle dataset (CSV)** have been submitted to the **university email** as instructed.
- To run the code, the **Kaggle dataset file** is required, which is cited in the **Bibliography** section.
- Additionally, for convenience, the **entire project with all files and outputs** has also been uploaded to **GitHub** and can be accessed directly using the link below:

GitHub Link: students-exam-scores

This format ensures that the coding part is clearly structured and follows the university's documentation standards.

10.2 BIBLIOGRAPHY

1. **Dataset Source:**

   students-exam-scores

   Mojtaba – Kaggle

2. **Programming Language & Environment:**
   - Python 3.x
   - Jupyter Notebook (Anaconda Distribution)

3. **Python Libraries Used:**
   - pandas – For data manipulation and cleaning
   - numpy – For numerical computations
   - matplotlib – For plotting and visualizations
   - seaborn – For advanced statistical visualizations
   - datetime – For date and time processing

4. **Software Tools:**
   - Microsoft Excel – For dataset preview
   - Microsoft Word – For writing and formatting the project report

5. **Reference Websites & Documentation:**
   - Python Official Documentation: https://docs.python.org
   - Seaborn Documentation: https://seaborn.pydata.org
   - Matplotlib Documentation: https://matplotlib.org
   - Stack Overflow – For resolving syntax and logic issues