# Synopsis

## Title of the Project: Student Result Data Analysis

## Introduction:

Student data analysis refers to the process of analyzing student-related data to gain insights and inform decision-making in education. This type of analysis can include data related to student performance, attendance, behavior, demographics, and other factors.

The goal of student data analysis is to identify patterns and trends that can inform instructional strategies, intervention programs, and policies that improve student outcomes. It can also be used to evaluate the effectiveness of existing programs and interventions, and to identify areas where improvements are needed.

## Problem statement:

Despite efforts to improve student outcomes, a school district is experiencing a persistent achievement gap between its low-income and non-low-income students. The district is seeking to better understand the factors contributing to this gap and develop targeted interventions to close it. To achieve this goal, the district plans to conduct a comprehensive analysis of student data, including academic performance, attendance, and demographic information, to identify patterns and trends that can inform intervention strategies. The district hopes that this analysis will lead to more effective interventions and ultimately improve outcomes for all students, regardless of their socioeconomic status.

## Literature survey (Brief):

Student data analysis has become an increasingly important tool for improving student outcomes and closing achievement gaps in education. Numerous studies have explored the use of data analysis techniques to identify factors that contribute to student success and to develop effective interventions to support struggling students.

## Objective of the Project work:

The objective of this project work is to conduct a comprehensive analysis of student data to identify factors that contribute to student success and to develop targeted interventions to support struggling students. Specifically, this project aims to identify patterns and trends in student data related to academic performance, attendance, behavior, and demographics, use predictive analytics techniques to identify students who are at risk of falling behind or dropping out, develop data visualization tools to present student data in a clear and accessible format that allows educators to quickly identify patterns and make informed decisions about how to best support student learning.

## Summary:

In summary, student data analysis involves the use of data analytics techniques to identify factors that contribute to student success and to develop targeted interventions to support struggling students. The process typically involves data collection, cleaning, and preprocessing, followed by descriptive and inferential analysis of the data to identify patterns and relationships between variables. Predictive analytics techniques can be used to identify students who are at risk of falling behind or dropping out, and data visualization tools can be developed to present student data in a clear and accessible format. Finally, student data analysis can be used to inform the development of targeted interventions to support struggling students, and these interventions can be evaluated and refined as necessary to ensure that they are meeting the needs of all students. Overall, student data analysis is a powerful tool for improving student outcomes and closing achievement gaps in education.

## References:

1. Baker, R. S. J. d., & Yacef, K. (2009). The state of educational data mining in 2009: A review and future visions. Journal of Educational Data Mining, 1(1), 3-17.
2. Means, B., Bakia, M., & Murphy, R. (2014). Learning analytics: Measurement innovations to support personalized learning. Washington, DC: US Department of Education.
3. Wiliam, D. (2011). Embedded formative assessment. Bloomington, IN: Solution Tree Press.

# 1. Introduction

Student result data analysis involves the process of examining and interpreting academic performance data to gain insights into student learning outcomes. This analysis can provide educators and administrators with valuable information about how well students are learning, which areas may require more attention, and how to make data-driven decisions to improve student achievement.

To perform student result data analysis, it's essential to collect relevant data, including grades, test scores, attendance records, and other performance indicators. This data can be organized and analyzed using a variety of techniques, including statistical analysis, data visualization.

Some common approaches to analyzing student result data include comparing the performance of different student groups, identifying trends and patterns in the data over time, and examining correlations between different performance indicators. By analyzing this data, educators can identify areas where students may need additional support or resources, and make informed decisions about curriculum, instruction, and assessment.

The purpose of student result data analysis is to help educators better understand how students are performing academically and to identify areas where additional support or interventions may be needed. By analyzing this data, educators can gain insights into the effectiveness of their teaching methods, the performance of individual students, and the overall success of their educational programs.Some of the key benefits of student result data analysis include:

- Identifying students who may be struggling and providing targeted support to help them improve.
- Evaluating the effectiveness of teaching methods and curriculum materials.
- Assessing the overall performance of educational programs and making data-driven decisions about improvements.
- Providing feedback to students and parents about academic progress and areas for improvement.
- Meeting accountability requirements for funding and accreditation.

# 2.Requirement Analysis

Requirement analysis is an important first step in any data analysis project, as it helps to ensure that the project is aligned with the needs and goals of the stakeholders involved. Here are some requirements that should be considered when conducting a student result data analysis:

1. Data sources: Identify the types of data sources that will be used in the analysis, such as student grades, attendance, test scores, and behavior. Determine how the data will be collected, stored, and integrated.
2. Data quality: Ensure that the data is accurate, complete, and consistent. Develop processes for verifying data quality, detecting errors, and resolving discrepancies.
3. Analysis tools: Select the appropriate analysis tools for the data sources and types of analysis required. For example, you might use statistical analysis tools to identify trends or machine learning algorithms to predict student outcomes.
4. Data security: Ensure that the student data is secure and protected from unauthorized access or use. Develop policies and procedures for data access, sharing, and retention.
5. User interface: Develop a user-friendly interface that allows educators to access and analyze the data easily. The interface should provide visualizations, dashboards, and reports that enable users to identify patterns, trends, and insights quickly.
6. Customization: Consider the needs of different users, such as teachers, administrators, and parents. needs.
7. Accessibility: Ensure that the system is accessible to all users, including those with disabilities or who use assistive technologies. Comply with accessibility standards, such as WCAG 2.0 or 2.1.
8. Integration: Ensure that the system integrates with other educational technologies, such as learning management systems or student information systems. Develop APIs or data exchange mechanisms to facilitate integration.

By carefully considering these requirements, a student result data analysis project can be designed and executed to meet the needs of all stakeholders involved.

# 3.Software Requirement Specification

The system must be able to perform the following functions:

- Collect and store student academic performance data, including grades, test scores, attendance, and behavior.
- Allow users to import data from external sources, such as learning management systems or student information systems.
- Provide data analysis tools, such as statistical analysis and machine learning algorithms, to identify trends and insights in the data.
- Allow users to create custom reports and visualizations based on the data analysis results.
- Provide real-time alerts to educators when a student's academic performance falls below a certain threshold.
- Integrate with other educational technologies, such as learning management systems or student information systems.

The system must meet the following non-functional requirements:

- Performance: The system must be able to handle large volumes of data and provide analysis results quickly.
- Security: The system must ensure the privacy and security of student data, comply with relevant data protection regulations, and prevent unauthorized access or use.
- Accessibility: The system must be accessible to users with disabilities or who use assistive technologies.
- Usability: The system must be user-friendly, with a clear and intuitive interface, and require minimal training to use.
- Compatibility: The system must be compatible with multiple web browsers and operating systems.

# 4.Analysis and Design

Analysis and design are critical stages in the development of a student data analysis system. Here is an overview of the analysis and design process:

1. Analysis During the analysis phase, you gather information about the system requirements, including functional and non-functional requirements, data sources, analysis tools, user interface, and security requirements. The analysis phase includes the following steps:

- Requirements gathering: Collect and document the system requirements, including user needs, system features, and performance requirements.
- Data modeling: Identify the data sources, including the types of data, the data structure, and the relationships between data entities.
- Use case analysis: Identify the use cases for the system, including the actors, scenarios, and expected outcomes.
- Risk analysis: Identify the risks associated with the system development, including security risks, data privacy risks, and system failure risks.

2. Design During the design phase, you create a blueprint for the system, including the system architecture, user interface, and data model. The design phase includes the following steps:

- System architecture design: Define the system components, including the presentation layer, application layer, and database layer.
- User interface design: Design the user interface, including the layout, navigation, and visual design.
- Data model design: Create a data model that maps to the system requirements, including the data entities, relationships, and constraints.
- Security design: Define the security mechanisms, including authentication, authorization, and encryption.

# 5. Data Understanding

Data understanding is an important step in the student data analysis process. This involves gathering information about the data that will be used for analysis. Here are the key steps in data understanding for student data analysis:

1. Data Collection The first step is to gather the data that will be used for analysis. This may involve collecting data from various sources, such as student information systems, learning management systems, or external sources.
2. Data Description Once the data has been collected, the next step is to describe the data. This involves understanding the data types, formats, and structures. Common data types for student data include grades, test scores, attendance records, and demographic data.
3. Data Cleaning Data cleaning is the process of identifying and correcting errors in the data. This may involve removing duplicate records, correcting typos, and fixing missing or inconsistent data.
4. Data Exploration Data exploration involves using statistical and visualization tools to explore the data and identify patterns and trends. This may involve creating graphs or charts to visualize the data, or using statistical analysis to identify correlations between different data points.
5. Data Preparation Data preparation involves transforming the data into a format that can be used for analysis. This may involve filtering the data to focus on specific variables or time periods, or combining data from multiple sources.
6. Data Quality Assessment Data quality assessment involves evaluating the quality of the data to ensure that it is accurate and reliable. This may involve testing the data against predefined rules or benchmarks, or comparing the data to external sources to identify discrepancies.

By following these steps, you can gain a deeper understanding of the data that will be used for analysis, which can help to identify insights and improve the accuracy of the analysis results.

# 6. Implementation and Results

Implementation and results are critical stages in the development of a student data analysis system. Here is an overview of the implementation and results process:

1. Implementation During the implementation phase, you build the system based on the requirements and design specifications. This includes the following steps:

- Software development: Develop the software components based on the system architecture and design.
- Database development: Build the database based on the data model and schema design.
- Integration testing: Test the system components to ensure that they work together as expected.

2. Results Once the system is implemented, you can use it to analyze the student data and generate insights. This includes the following steps:

- Data analysis: Use statistical and visualization tools to analyze the student data and identify patterns and trends.
- Reporting: Create reports and dashboards to present the analysis results to stakeholders, such as administrators, teachers, and students.
- Feedback and refinement: Gather feedback from users and stakeholders and refine the system based on feedback.

The results of the student data analysis can be used to inform decision-making and improve student outcomes. For example, the analysis may reveal areas where students are struggling, which can inform instructional strategies and interventions. The analysis may also identify areas where students are excelling, which can inform recognition and reward programs.

Overall, the implementation and results process is a critical component of the student data analysis system, and it requires close collaboration between developers, analysts, and stakeholders to ensure that the system meets user needs and generates valuable insights.

## 6.1 Data Acquisition

Data acquisition is a critical step in the student data analysis process. It involves identifying the sources of data, extracting the data from those sources, and preparing the data for analysis. Here are the key steps in data acquisition for student data analysis:

1. Identify Data Sources: The first step in data acquisition is to identify the sources of data. Common data sources for student data analysis include student information systems, learning management systems, and external sources such as surveys or social media.
2. Extract Data: Once the data sources have been identified, the next step is to extract the data from those sources. This may involve using APIs or scripts to automate the extraction process.
3. Clean Data : After the data has been extracted, the next step is to clean the data. This involves identifying and correcting errors in the data, such as missing or inconsistent values.
4. Transform Data: Once the data has been cleaned, the next step is to transform the data into a format that can be used for analysis. This may involve converting data types, renaming columns, or aggregating data.
5. Load Data: The final step in data acquisition is to load the data into a database or data warehouse where it can be accessed for analysis.

## 6.2 Data Preparation

Data preparation is a critical step in the student data analysis process. It involves transforming the raw data into a format that can be used for analysis. Here are the key steps in data preparation for student data analysis:

1. Data Cleaning: The first step in data preparation is to clean the data. This involves identifying and correcting errors in the data, such as missing or inconsistent values.
2. Data Integration: The next step is to integrate data from multiple sources. This may involve combining data from different databases or merging data from multiple spreadsheets.

3. Data Transformation: Once the data has been cleaned and integrated, the next step is to transform the data into a format that can be used for analysis. This may involve creating new variables or aggregating data to create summary statistics.
4. Data Reduction: After the data has been transformed, the next step is to reduce the data to focus on the most relevant variables. This may involve filtering the data to exclude variables that are not relevant to the analysis.
5. Data Sampling: Finally, you may choose to sample the data to reduce the size of the dataset. This can make it easier to work with the data and reduce processing time.

## 6.3 Data Exploration and Visualization

# Importing the libraries

In [1]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
df = pd.read_csv("Student2.csv")
```

# Displaying the Top 5 rows in the dataset

In [2]:
```
df.head()
```

Out[2]:

| | Name | Maths | Physics | Chemistry | Result | Gender | State |
|---|---|---|---|---|---|---|---|
| 0 | Yash | 17.0 | 27.0 | 22.0 | 0.0 | male | Gujarat |
| 1 | Prit | 72.0 | 82.0 | 77.0 | 1.0 | male | Haryana |
| 2 | Meet | 97.0 | 18.0 | 13.0 | 0.0 | male | Himachal Pradesh |
| 3 | Drashti | 8.0 | 42.0 | 37.0 | 0.0 | female | Jammu and Kashmir |
| 4 | Saloni | 32.0 | 25.0 | 20.0 | 0.0 | female | Karnataka |

## Displaying the last 5 rows in the dataset

In [3]:
```
df.tail()
```

Out[3]:

| | Name | Maths | Physics | Chemistry | Result | Gender | State |
|---|---|---|---|---|---|---|---|
| **1069** | Binny | 50.0 | 66.0 | 42.0 | 1.0 | NaN | Karnataka |
| **1070** | Srisa | NaN | 67.0 | 40.0 | 1.0 | female | Kerala |
| **1071** | Ritika | 81.0 | 54.0 | 36.0 | 0.0 | female | Madhya Pradesh |
| **1072** | Niyati | 44.0 | 49.0 | 53.0 | 1.0 | female | Maharashtra |
| **1073** | Barkha | 49.0 | 79.0 | 76.0 | 1.0 | female | Karnataka |

## Displaying the 10 sample rows in the dataset

In [4]:
```
df.sample(10)
```

Out[4]:

| | Name | Maths | Physics | Chemistry | Result | Gender | State |
|---|---|---|---|---|---|---|---|
| **145** | Rohan | 65.0 | 91.0 | 57.0 | 1.0 | male | Karnataka |
| **802** | Tirth | 94.0 | 19.0 | 95.0 | 0.0 | male | Himachal Pradesh |
| **589** | Keyur | 35.0 | 58.0 | 87.0 | 0.0 | male | Karnataka |
| **736** | Yashraj | 32.0 | 75.0 | 98.0 | 0.0 | male | Kerala |
| **932** | Lalit | 42.0 | 32.0 | 68.0 | 0.0 | male | Jammu and Kashmir |
| **1073** | Barkha | 49.0 | 79.0 | 76.0 | 1.0 | female | Karnataka |
| **752** | Viral | 3.0 | 21.0 | 98.0 | 0.0 | male | Himachal Pradesh |
| **405** | Dhanvarsha | 24.0 | 21.0 | 6.0 | 0.0 | female | Karnataka |
| **981** | Avika | 13.0 | 21.0 | 87.0 | 0.0 | female | Himachal Pradesh |
| **14** | Minal | 12.0 | 59.0 | 54.0 | 0.0 | female | Karnataka |

## Displaying Description and summary of data

In [5]:
```
df.info()
print("-----------------------------------------")
df.shape
print("-----------------------------------------")
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1074 entries, 0 to 1073
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Name       1074 non-null   object
 1   Maths      1072 non-null   float64
 2   Physics    1072 non-null   float64
 3   Chemistry  1073 non-null   float64
 4   Result     1073 non-null   float64
 5   Gender     1072 non-null   object
 6   State      1073 non-null   object
dtypes: float64(4), object(3)
memory usage: 58.9+ KB
-----------------------------------------
-----------------------------------------
```

## Displaying the column headings

In [6]:
```
df.columns
```

Out[6]:
```
Index(['Name', 'Maths', 'Physics', 'Chemistry', 'Result', 'Gender', 'State'], dtype='object')
```

## Display the statistical information of the data

In [7]:
```
df.describe()
```

Out[7]:

| | Maths | Physics | Chemistry | Result |
|---|---|---|---|---|
| count | 1072.000000 | 1072.000000 | 1073.000000 | 1073.000000 |
| mean | 51.074627 | 52.155784 | 52.013048 | 0.249767 |
| std | 29.352076 | 24.652183 | 27.403253 | 0.433080 |
| min | 0.000000 | 10.000000 | 5.000000 | 0.000000 |
| 25% | 26.000000 | 31.000000 | 29.000000 | 0.000000 |
| 50% | 51.000000 | 51.000000 | 53.000000 | 0.000000 |
| 75% | 76.000000 | 74.000000 | 75.000000 | 0.000000 |
| max | 100.000000 | 95.000000 | 99.000000 | 1.000000 |

## Data Preparation

## 1.Checking for Duplicate data

In [8]:
```
df.duplicated()
```

Out[8]:
```
0       False
1       False
2       False
3       False
4       False
        ...
1069    False
1070    False
1071     True
1072     True
1073     True
Length: 1074, dtype: bool
```

## 2.Checking for Missing data

In [9]:
```
df.isnull().sum()
```

Out[9]:
```
Name         0
Maths        2
Physics      2
Chemistry    1
Result       1
Gender       2
State        1
dtype: int64
```

## The data contains some missing values so we drop those rows

```
In [10]: df.dropna()
```

Out[10]:

| | Name | Maths | Physics | Chemistry | Result | Gender | State |
|---|---|---|---|---|---|---|---|
| 0 | Yash | 17.0 | 27.0 | 22.0 | 0.0 | male | Gujarat |
| 1 | Prit | 72.0 | 82.0 | 77.0 | 1.0 | male | Haryana |
| 2 | Meet | 97.0 | 18.0 | 13.0 | 0.0 | male | Himachal Pradesh |
| 3 | Drashti | 8.0 | 42.0 | 37.0 | 0.0 | female | Jammu and Kashmir |
| 4 | Saloni | 32.0 | 25.0 | 20.0 | 0.0 | female | Karnataka |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 1066 | Binjal | 6.0 | 51.0 | 57.0 | 0.0 | female | Himachal Pradesh |
| 1068 | Sindu | 87.0 | 25.0 | 89.0 | 0.0 | male | Karnataka |
| 1071 | Ritika | 81.0 | 54.0 | 36.0 | 0.0 | female | Madhya Pradesh |
| 1072 | Niyati | 44.0 | 49.0 | 53.0 | 1.0 | female | Maharashtra |
| 1073 | Barkha | 49.0 | 79.0 | 76.0 | 1.0 | female | Karnataka |

1065 rows × 7 columns

```
In [11]: df=df.dropna()
```

```
In [12]: df.dropna()
```

Out[12]:

| | Name | Maths | Physics | Chemistry | Result | Gender | State |
|---|---|---|---|---|---|---|---|
| 0 | Yash | 17.0 | 27.0 | 22.0 | 0.0 | male | Gujarat |
| 1 | Prit | 72.0 | 82.0 | 77.0 | 1.0 | male | Haryana |
| 2 | Meet | 97.0 | 18.0 | 13.0 | 0.0 | male | Himachal Pradesh |
| 3 | Drashti | 8.0 | 42.0 | 37.0 | 0.0 | female | Jammu and Kashmir |
| 4 | Saloni | 32.0 | 25.0 | 20.0 | 0.0 | female | Karnataka |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 1066 | Binjal | 6.0 | 51.0 | 57.0 | 0.0 | female | Himachal Pradesh |
| 1068 | Sindu | 87.0 | 25.0 | 89.0 | 0.0 | male | Karnataka |
| 1071 | Ritika | 81.0 | 54.0 | 36.0 | 0.0 | female | Madhya Pradesh |
| 1072 | Niyati | 44.0 | 49.0 | 53.0 | 1.0 | female | Maharashtra |
| 1073 | Barkha | 49.0 | 79.0 | 76.0 | 1.0 | female | Karnataka |

1065 rows × 7 columns

# Deleting the Duplicate values in the data

In [13]:
```
df=df.drop_duplicates()
df
```

Out[13]:

| | Name | Maths | Physics | Chemistry | Result | Gender | State |
|---|---|---|---|---|---|---|---|
| 0 | Yash | 17.0 | 27.0 | 22.0 | 0.0 | male | Gujarat |
| 1 | Prit | 72.0 | 82.0 | 77.0 | 1.0 | male | Haryana |
| 2 | Meet | 97.0 | 18.0 | 13.0 | 0.0 | male | Himachal Pradesh |
| 3 | Drashti | 8.0 | 42.0 | 37.0 | 0.0 | female | Jammu and Kashmir |
| 4 | Saloni | 32.0 | 25.0 | 20.0 | 0.0 | female | Karnataka |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 995 | Karanvir | 4.0 | 48.0 | 64.0 | 0.0 | male | Karnataka |
| 996 | Kinshuk | 63.0 | 22.0 | 88.0 | 0.0 | male | Gujarat |
| 997 | Sharda | 90.0 | 64.0 | 43.0 | 1.0 | female | Madhya Pradesh |
| 998 | Kunal | 67.0 | 41.0 | 6.0 | 0.0 | male | Madhya Pradesh |
| 999 | Giaa | 92.0 | 74.0 | 9.0 | 0.0 | female | Madhya Pradesh |

1000 rows × 7 columns

# 3.Checking for Data Inconsistencies

In [14]:
```
df['Result'].value_counts()
```

Out[14]:
```
0.0    755
1.0    245
Name: Result, dtype: int64
```

## The Result Values Contains zeros and ones so we change 0 to fail and 1 to pass

In [15]:
```
df['Result']=''
a= list(zip(df['Maths'],df['Physics'],df['Chemistry']))
#print(a)
for i,v in enumerate(a):
    if v[0]>35 and v[1]>35 and v[2]>35:
        df['Result'][i]="Pass"
    else:
        df['Result'][i]="Fail"
df
```

Out[15]:

| | Name | Maths | Physics | Chemistry | Result | Gender | State |
|---|---|---|---|---|---|---|---|
| **0** | Yash | 17.0 | 27.0 | 22.0 | Fail | male | Gujarat |
| **1** | Prit | 72.0 | 82.0 | 77.0 | Pass | male | Haryana |
| **2** | Meet | 97.0 | 18.0 | 13.0 | Fail | male | Himachal Pradesh |
| **3** | Drashti | 8.0 | 42.0 | 37.0 | Fail | female | Jammu and Kashmir |
| **4** | Saloni | 32.0 | 25.0 | 20.0 | Fail | female | Karnataka |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **995** | Karanvir | 4.0 | 48.0 | 64.0 | Fail | male | Karnataka |
| **996** | Kinshuk | 63.0 | 22.0 | 88.0 | Fail | male | Gujarat |
| **997** | Sharda | 90.0 | 64.0 | 43.0 | Pass | female | Madhya Pradesh |
| **998** | Kunal | 67.0 | 41.0 | 6.0 | Fail | male | Madhya Pradesh |
| **999** | Giaa | 92.0 | 74.0 | 9.0 | Fail | female | Madhya Pradesh |

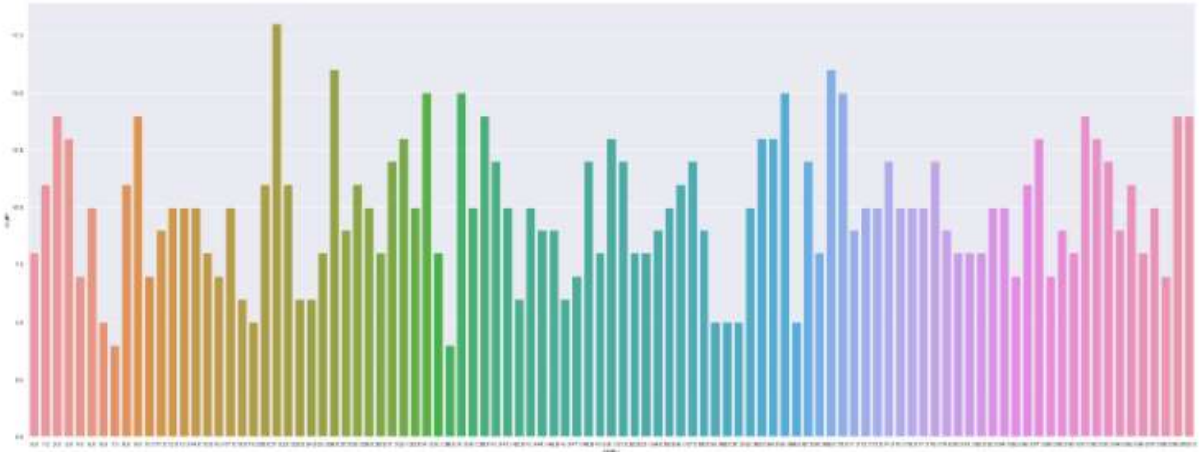
1000 rows × 7 columns

## Copying the Processed data to a new CSV file

In [16]:
```
df.to_csv('StudentResult.csv')
```

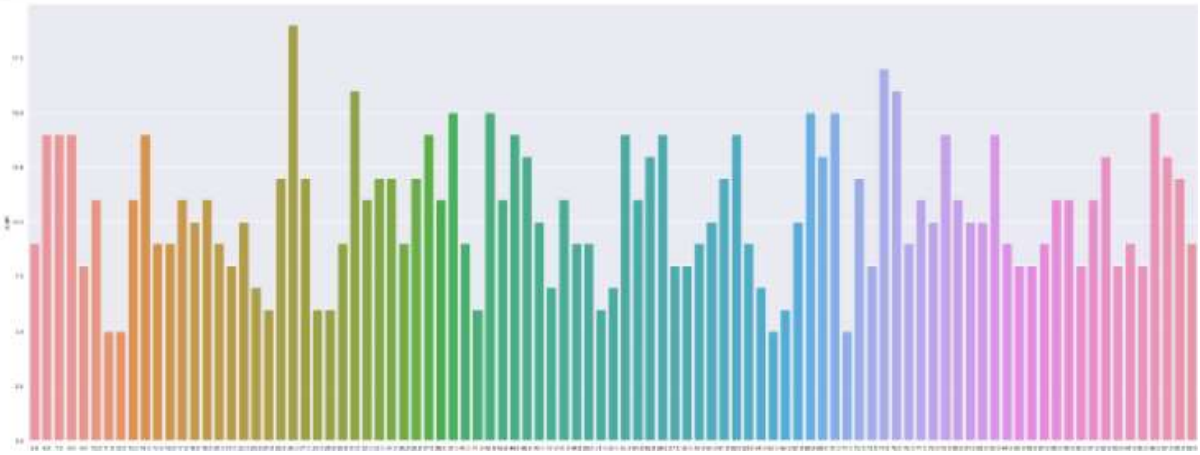
## Count Plot for Maths subject

In [17]:
```
plt.figure(figsize=(40,15))
sns.set_theme(style="darkgrid")
ax = sns.countplot(x="Maths", data=df)
plt.show()
```



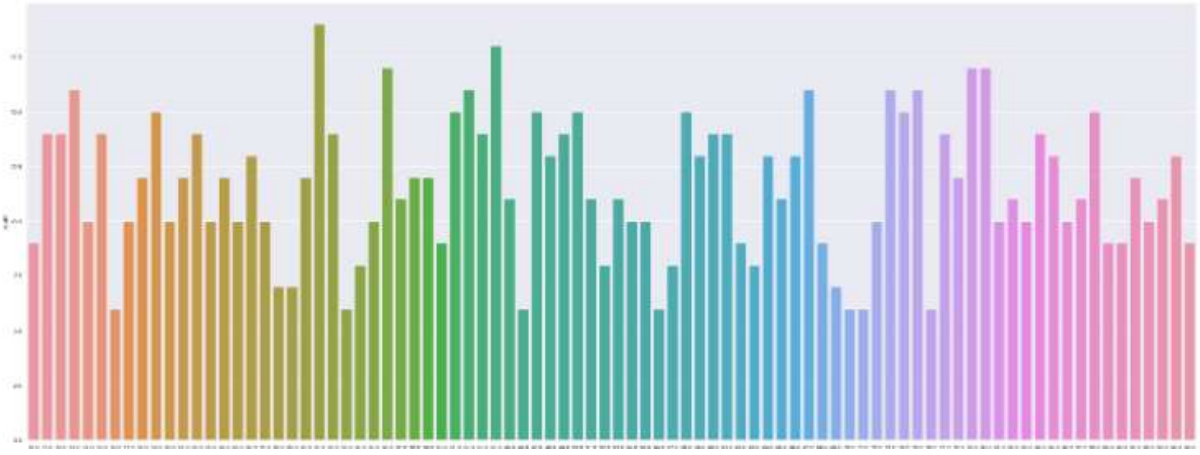
## Count Plot for Chemistry subject

In [18]:
```
plt.figure(figsize=(40,15))
sns.set_theme(style="darkgrid")
ax = sns.countplot(x="Chemistry", data=df)
plt.show()
```



## Count Plot for Physics subject

In [19]:
```
plt.figure(figsize=(40,15))
sns.set_theme(style="darkgrid")
ax = sns.countplot(x="Physics", data=df)
plt.show()
```



## Adding Total column to the dataframe

In [20]:
```
df1 = df.copy()
```

In [21]:
```
df1['Total']=df1['Maths']+df1['Physics']+df1['Chemistry']
df1
```

Out[21]:

| | Name | Maths | Physics | Chemistry | Result | Gender | State | Total |
|---|---|---|---|---|---|---|---|---|
| 0 | Yash | 17.0 | 27.0 | 22.0 | Fail | male | Gujarat | 66.0 |
| 1 | Prit | 72.0 | 82.0 | 77.0 | Pass | male | Haryana | 231.0 |
| 2 | Meet | 97.0 | 18.0 | 13.0 | Fail | male | Himachal Pradesh | 128.0 |
| 3 | Drashti | 8.0 | 42.0 | 37.0 | Fail | female | Jammu and Kashmir | 87.0 |
| 4 | Saloni | 32.0 | 25.0 | 20.0 | Fail | female | Karnataka | 77.0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 995 | Karanvir | 4.0 | 48.0 | 64.0 | Fail | male | Karnataka | 116.0 |
| 996 | Kinshuk | 63.0 | 22.0 | 88.0 | Fail | male | Gujarat | 173.0 |
| 997 | Sharda | 90.0 | 64.0 | 43.0 | Pass | female | Madhya Pradesh | 197.0 |
| 998 | Kunal | 67.0 | 41.0 | 6.0 | Fail | male | Madhya Pradesh | 114.0 |
| 999 | Giaa | 92.0 | 74.0 | 9.0 | Fail | female | Madhya Pradesh | 175.0 |

1000 rows × 8 columns

# Adding percentage column to the dataframe

In [22]:
```
df1['Percentage']=(df1['Total']/300)*100
df1['Percentage']=df1['Percentage'].round(2)
df1
```

Out[22]:

| | Name | Maths | Physics | Chemistry | Result | Gender | State | Total | Percentage |
|---|---|---|---|---|---|---|---|---|---|
| 0 | Yash | 17.0 | 27.0 | 22.0 | Fail | male | Gujarat | 66.0 | 22.00 |
| 1 | Prit | 72.0 | 82.0 | 77.0 | Pass | male | Haryana | 231.0 | 77.00 |
| 2 | Meet | 97.0 | 18.0 | 13.0 | Fail | male | Himachal Pradesh | 128.0 | 42.67 |
| 3 | Drashti | 8.0 | 42.0 | 37.0 | Fail | female | Jammu and Kashmir | 87.0 | 29.00 |
| 4 | Saloni | 32.0 | 25.0 | 20.0 | Fail | female | Karnataka | 77.0 | 25.67 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 995 | Karanvir | 4.0 | 48.0 | 64.0 | Fail | male | Karnataka | 116.0 | 38.67 |
| 996 | Kinshuk | 63.0 | 22.0 | 88.0 | Fail | male | Gujarat | 173.0 | 57.67 |
| 997 | Sharda | 90.0 | 64.0 | 43.0 | Pass | female | Madhya Pradesh | 197.0 | 65.67 |
| 998 | Kunal | 67.0 | 41.0 | 6.0 | Fail | male | Madhya Pradesh | 114.0 | 38.00 |
| 999 | Giaa | 92.0 | 74.0 | 9.0 | Fail | female | Madhya Pradesh | 175.0 | 58.33 |

1000 rows × 9 columns

# Calculating grade

In [23]:
```
def gd(no):
    if no>85:
        return "O"
    elif (no<85 and no>75):
        return "A"
    elif (no<75 and no>65):
        return "B"
    elif (no<65 and no>35):
        return "C"
    else:
        return "Fail"
df1['Grade']=df1['Percentage'].apply(gd)
df1
```

Out[23]:

| | Name | Maths | Physics | Chemistry | Result | Gender | State | Total | Percentage | Grade |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Yash | 17.0 | 27.0 | 22.0 | Fail | male | Gujarat | 66.0 | 22.00 | Fail |
| **1** | Prit | 72.0 | 82.0 | 77.0 | Pass | male | Haryana | 231.0 | 77.00 | A |
| **2** | Meet | 97.0 | 18.0 | 13.0 | Fail | male | Himachal Pradesh | 128.0 | 42.67 | C |
| **3** | Drashti | 8.0 | 42.0 | 37.0 | Fail | female | Jammu and Kashmir | 87.0 | 29.00 | Fail |
| **4** | Saloni | 32.0 | 25.0 | 20.0 | Fail | female | Karnataka | 77.0 | 25.67 | Fail |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **995** | Karanvir | 4.0 | 48.0 | 64.0 | Fail | male | Karnataka | 116.0 | 38.67 | C |
| **996** | Kinshuk | 63.0 | 22.0 | 88.0 | Fail | male | Gujarat | 173.0 | 57.67 | C |
| **997** | Sharda | 90.0 | 64.0 | 43.0 | Pass | female | Madhya Pradesh | 197.0 | 65.67 | B |
| **998** | Kunal | 67.0 | 41.0 | 6.0 | Fail | male | Madhya Pradesh | 114.0 | 38.00 | C |
| **999** | Giaa | 92.0 | 74.0 | 9.0 | Fail | female | Madhya Pradesh | 175.0 | 58.33 | C |

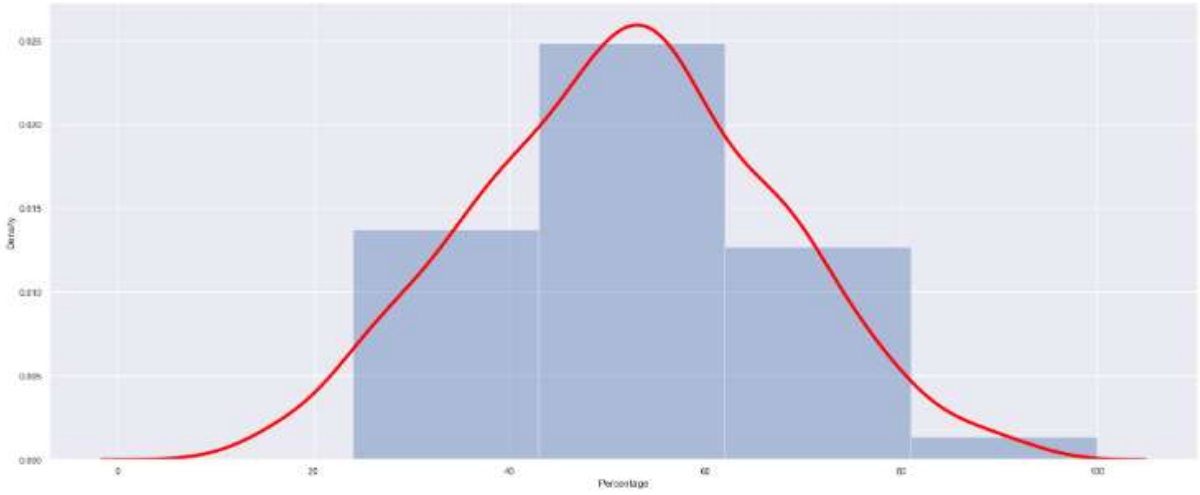
1000 rows × 10 columns

# Data Distribution of percentage

In [24]:
```
bn = np.linspace(24,100,5)
bn
```

Out[24]:
```
array([ 24.,  43.,  62.,  81., 100.])
```

In [25]:
```
plt.figure(figsize=(25,10))
sns.distplot(df1['Percentage'],bins=bn,kde_kws={'linewidth':4,'color':'red','label':'before_modify'})
plt.show()
```




# All subjects marks distribution

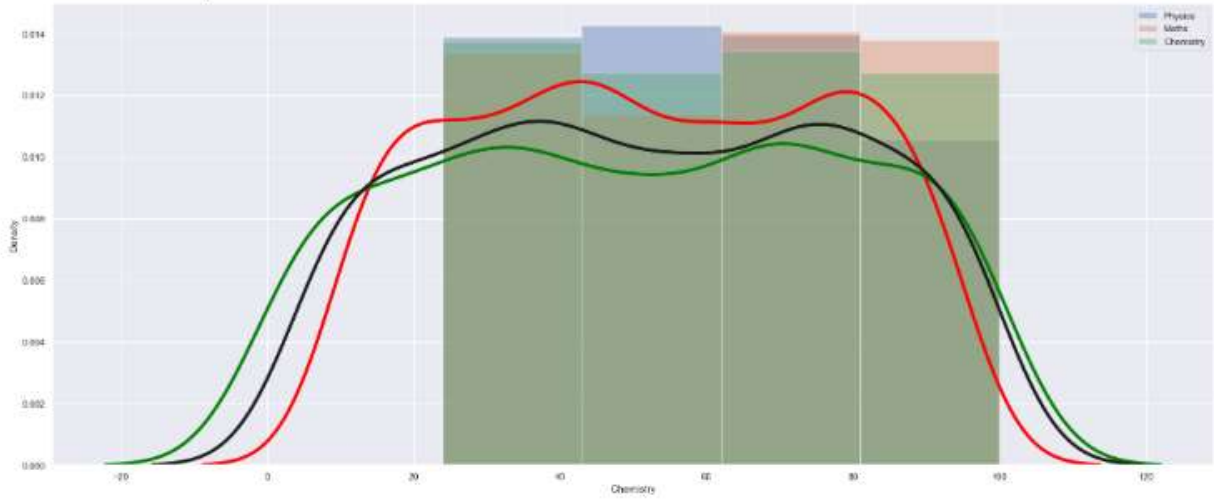
In [26]:
```
# Statistical Info of all subject columns
df1[['Physics','Maths','Chemistry']].describe()
```

Out[26]:

| | Physics | Maths | Chemistry |
|---|---|---|---|
| **count** | 1000.000000 | 1000.000000 | 1000.000000 |
| **mean** | 52.316000 | 50.842000 | 52.239000 |
| **std** | 24.801114 | 29.238775 | 27.397052 |
| **min** | 10.000000 | 0.000000 | 5.000000 |
| **25%** | 31.000000 | 26.000000 | 29.750000 |
| **50%** | 51.000000 | 51.000000 | 53.000000 |
| **75%** | 74.000000 | 76.000000 | 76.000000 |
| **max** | 95.000000 | 100.000000 | 99.000000 |

In [27]:
```
plt.figure(figsize=(25,10))
print("Green : Maths")
print("Red : Physics")
print("Black : Chemistry")
sns.distplot(df1['Physics'],bins=bn,kde_kws={'linewidth':4,'color':'red'},label='Physics')
sns.distplot(df1['Maths'],bins=bn,kde_kws={'linewidth':4,'color':'green'},label='Maths')
sns.distplot(df1['Chemistry'],bins=bn,kde_kws={'linewidth':4,'color':'k'},label='Chemistry')
plt.legend()
plt.show()
```

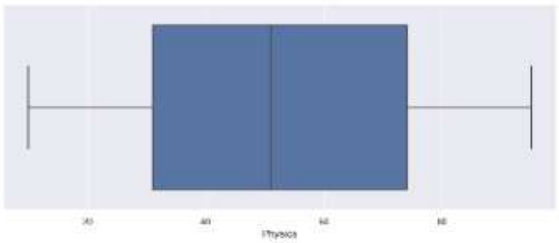
```
Green : Maths
Red : Physics
Black : Chemistry
```

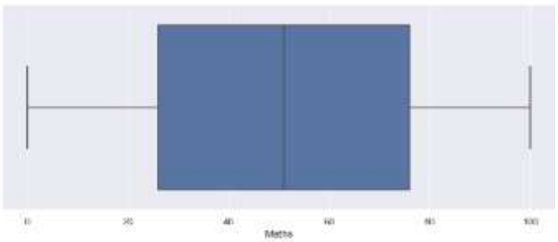


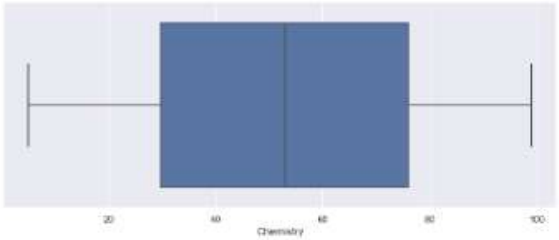

## Box Plot : To Detect Outliers

In [28]:
```
col=['Physics', 'Maths', 'Chemistry'] #Create a list of input columns
plt.figure(figsize=(25,10))
for i,v in enumerate(col):
    print(i,v)
    plt.subplot(2,2,i+1)
    sns.boxplot(x=v, data=df1,meanline="True")
plt.show()
```

```
0 Physics
1 Maths
2 Chemistry
```



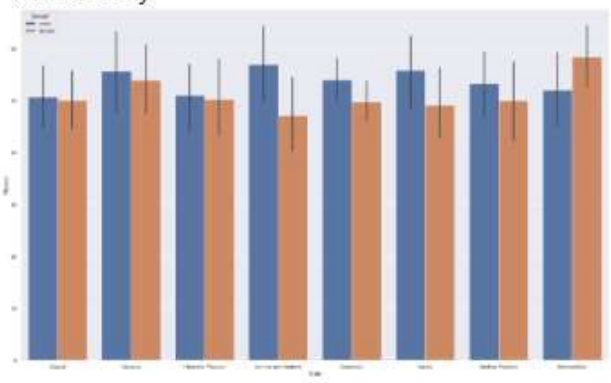



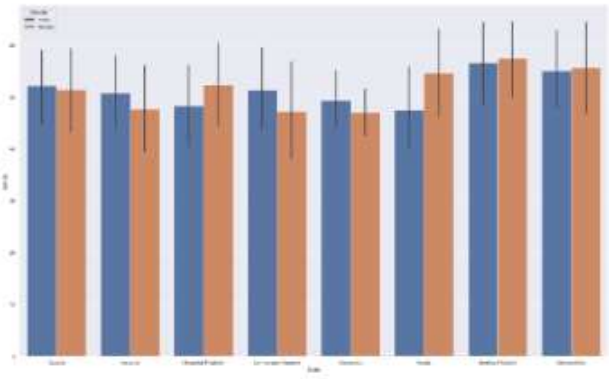



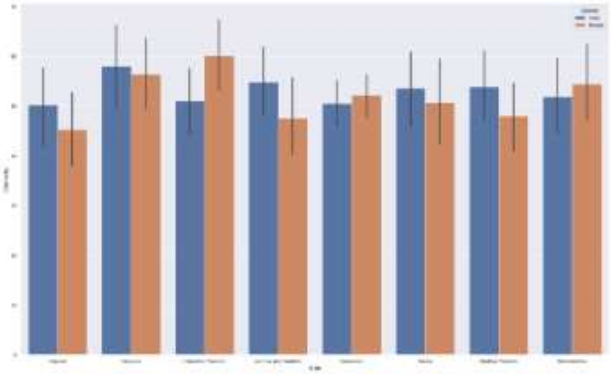
# Visualization by Bar Plot

```
In [29]: col=['Physics', 'Maths', 'Chemistry'] #Create a list of input columns
plt.figure(figsize=(50,30))
for i,v in enumerate(col):
    print(i,v)
    plt.subplot(2,2,i+1)
    sns.barplot(x="State", y=v, data=df, hue="Gender")
plt.show()
```

```
0 Physics
1 Maths
2 Chemistry
```







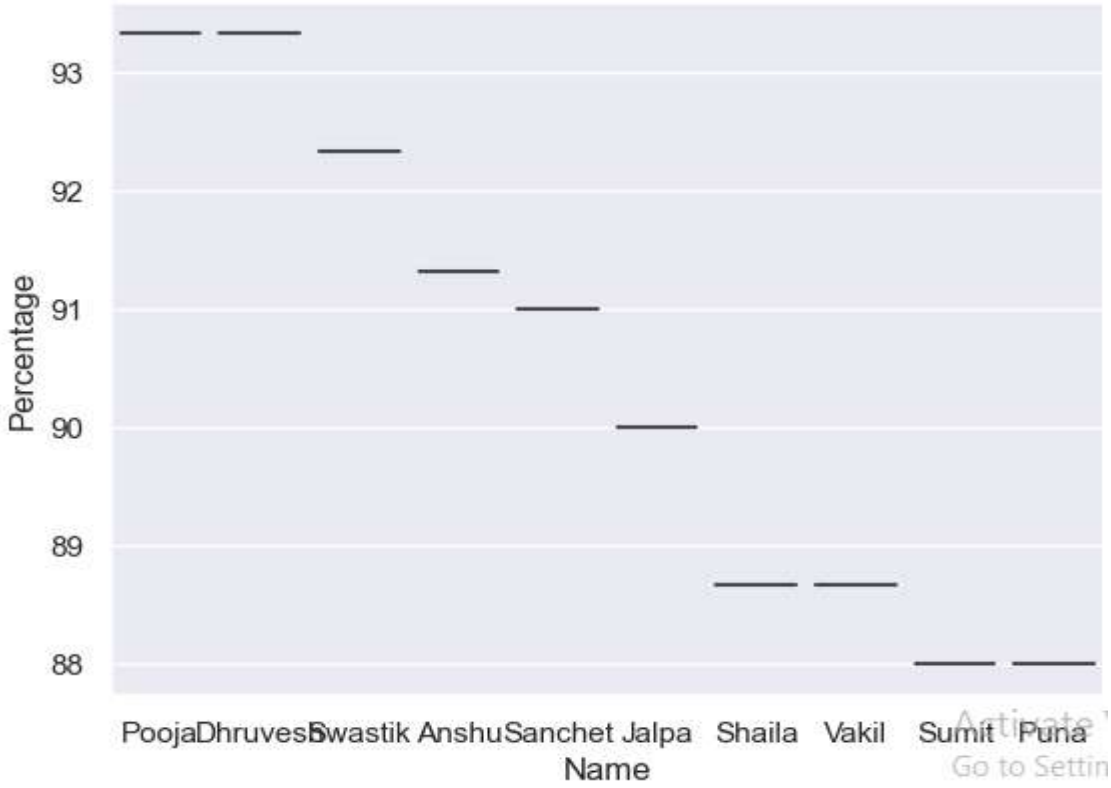
```
In [30]: #Finding the top 10 scorers

In [31]: df1["Percentage"].nlargest(10)

Out[31]: 21     93.33
         407    93.33
         779    92.33
         103    91.33
         713    91.00
         521    90.00
         86     88.67
         787    88.67
         212    88.00
         853    88.00
         Name: Percentage, dtype: float64
```

In [32]:
```
sns.violinplot(
    x='Name',
    y='Percentage',
    data=df1.nlargest(10, 'Percentage')
)
```

Out[32]:
```
<AxesSubplot:xlabel='Name', ylabel='Percentage'>
```

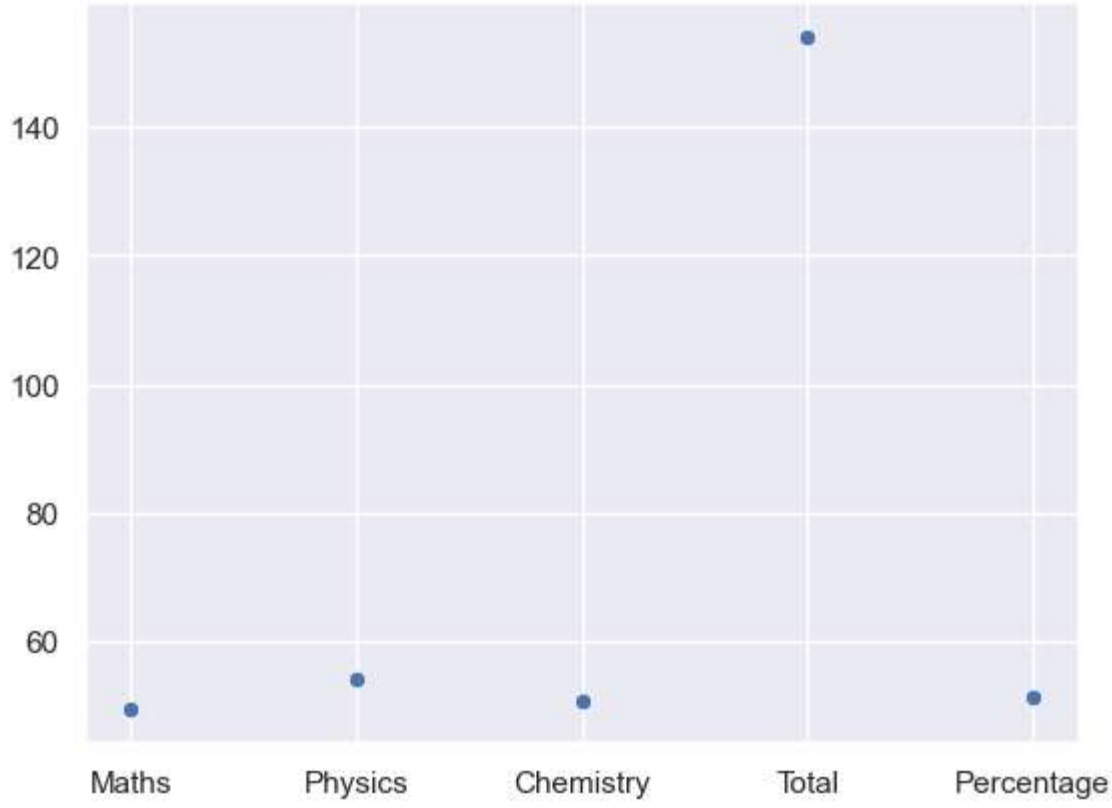



# Avg score of male in Karnataka

In [33]:
```
avgmale=df1[(df1['State']=='Karnataka') & (df1['Gender']=='male')].mean()
df1[(df1['State']=='Karnataka') & (df1['Gender']=='male')].mean()
```

Out[33]:
```
Maths          49.424460
Physics        54.043165
Chemistry      50.604317
Total         154.071942
Percentage     51.357338
dtype: float64
```

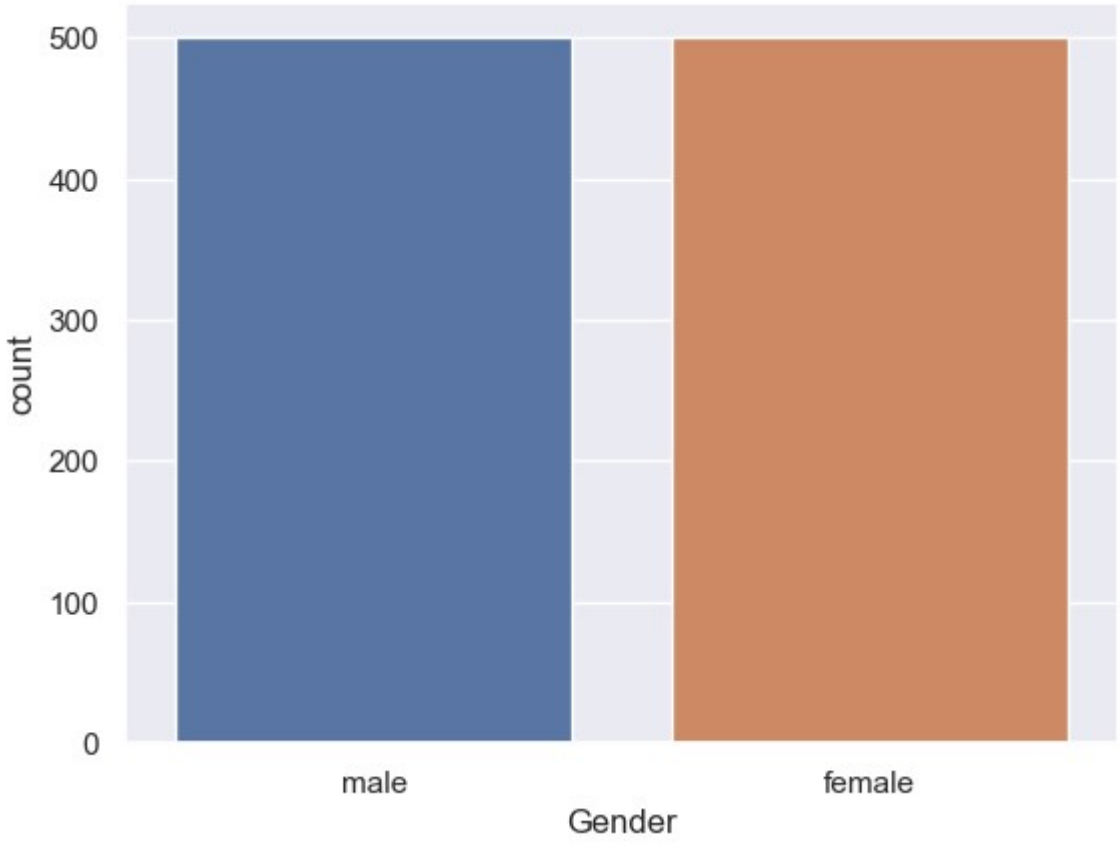
```
In [34]: sns.scatterplot(data = avgmale )
Out[34]: <AxesSubplot:>
```




# Overall total Percentage of Male and Female

```
In [35]: df['Gender'].value_counts()/df.shape[0]*100
Out[35]: male      50.0
         female    50.0
         Name: Gender, dtype: float64

In [36]: sns.countplot(x=df["Gender"])
Out[36]: <AxesSubplot:xlabel='Gender', ylabel='count'>
```



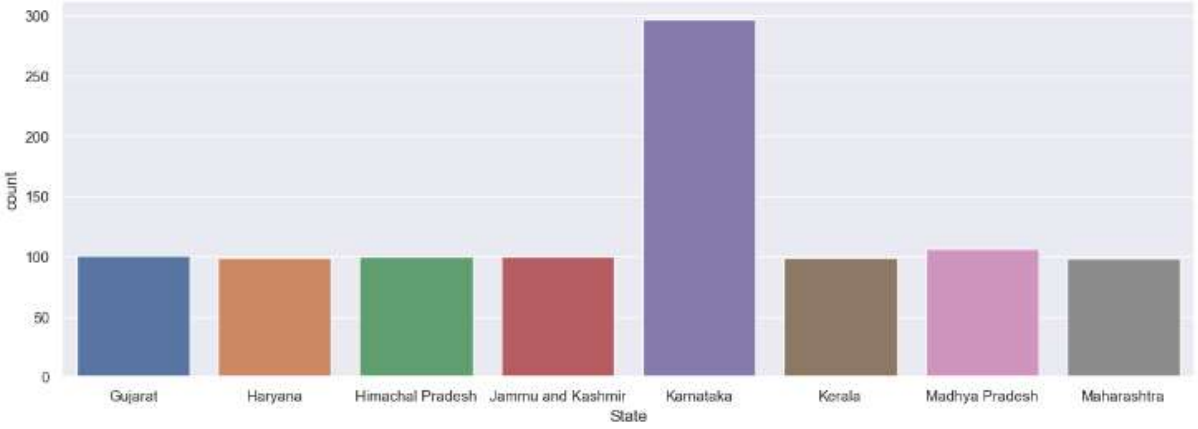

# Finding the State with maximum students

In [37]:
```
df1['State'].value_counts()
```

Out[37]:
```
Karnataka            297
Madhya Pradesh       106
Gujarat              101
Himachal Pradesh     100
Jammu and Kashmir    100
Haryana               99
Kerala                99
Maharashtra           98
Name: State, dtype: int64
```

In [38]:
```
plt.figure(figsize=(15,5))
sns.set_theme(style="darkgrid")
sns.countplot(x=df["State"])
```

Out[38]: <AxesSubplot:xlabel='State', ylabel='count'>




# Top 10 Highest Scorers Of Karnataka state

In [39]:
```
df1[(df1['State']=='Karnataka')].sort_values(by='Percentage',ascending = False).head(10)
```

Out[39]:

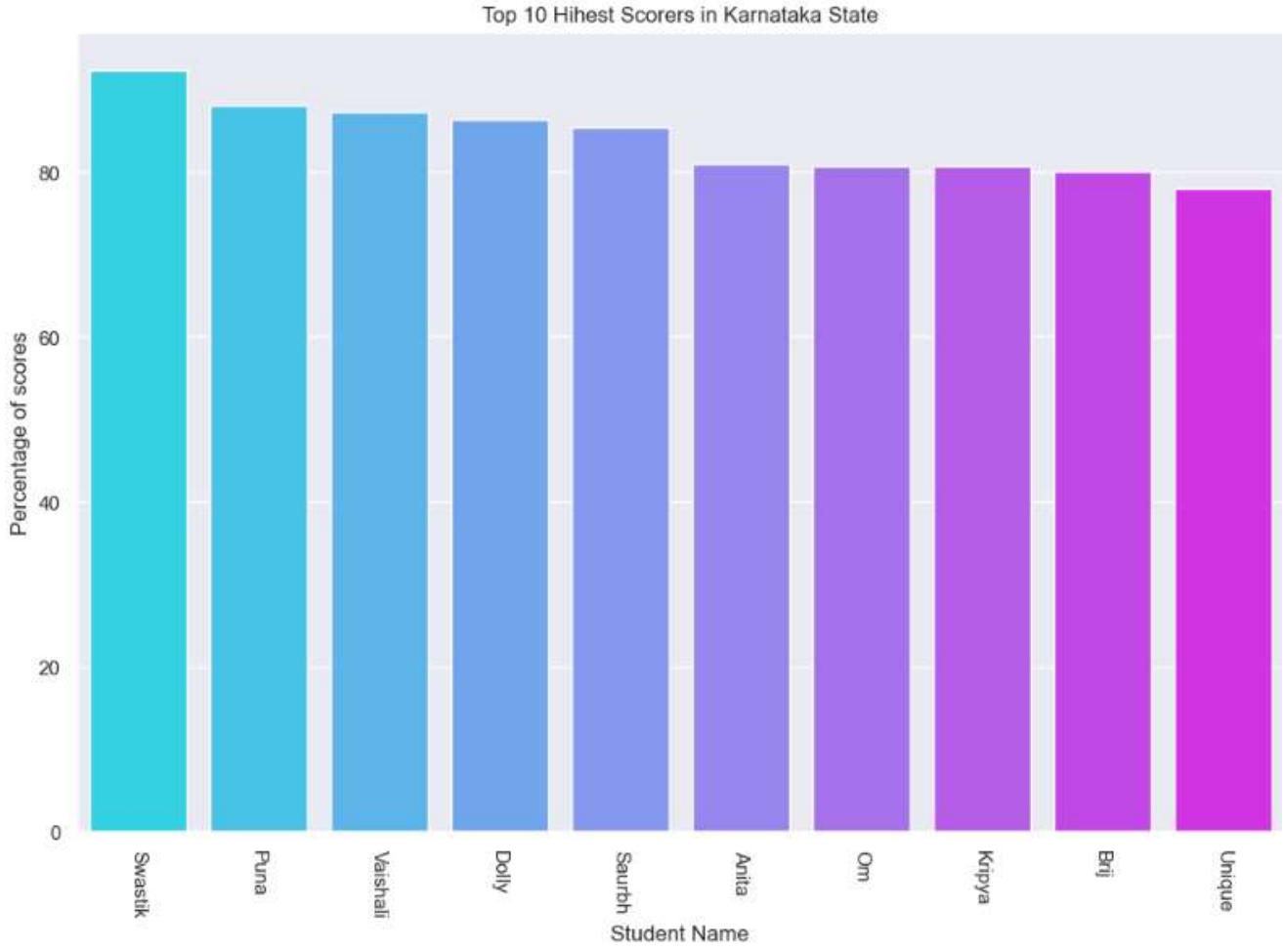
| | Name | Maths | Physics | Chemistry | Result | Gender | State | Total | Percentage | Grade |
|---|---|---|---|---|---|---|---|---|---|---|
| **779** | Swastik | 91.0 | 88.0 | 98.0 | Pass | male | Karnataka | 277.0 | 92.33 | O |
| **853** | Puna | 98.0 | 67.0 | 99.0 | Pass | male | Karnataka | 264.0 | 88.00 | O |
| **605** | Vaishali | 87.0 | 94.0 | 81.0 | Pass | female | Karnataka | 262.0 | 87.33 | O |
| **759** | Dolly | 86.0 | 77.0 | 96.0 | Pass | female | Karnataka | 259.0 | 86.33 | O |
| **209** | Saurbh | 91.0 | 67.0 | 98.0 | Pass | male | Karnataka | 256.0 | 85.33 | O |
| **125** | Anita | 98.0 | 56.0 | 89.0 | Pass | female | Karnataka | 243.0 | 81.00 | A |
| **425** | Om | 91.0 | 86.0 | 65.0 | Pass | male | Karnataka | 242.0 | 80.67 | A |
| **898** | Kripya | 67.0 | 93.0 | 82.0 | Pass | female | Karnataka | 242.0 | 80.67 | A |
| **933** | Brij | 91.0 | 54.0 | 95.0 | Pass | male | Karnataka | 240.0 | 80.00 | A |
| **615** | Unique | 91.0 | 84.0 | 59.0 | Pass | male | Karnataka | 234.0 | 78.00 | A |

# Top 10 Highest Scorers Of Karnataka state

In [40]:
```
top10= df1[(df1['State']=='Karnataka')].sort_values(by='Percentage',ascending = False).head(10)
```

In [41]:
```
plt.figure(figsize=(12,8))
sns.barplot(x='Name',y='Percentage',data=top10,palette='cool',ci=None);
plt.xlabel('Student Name')
plt.xticks(rotation=-90)
plt.ylabel('Percentage of scores')
plt.title('Top 10 Hihest Scorers in Karnataka State')
```

Out[41]:
```
Text(0.5, 1.0, 'Top 10 Hihest Scorers in Karnataka State')
```




# Score of M/F

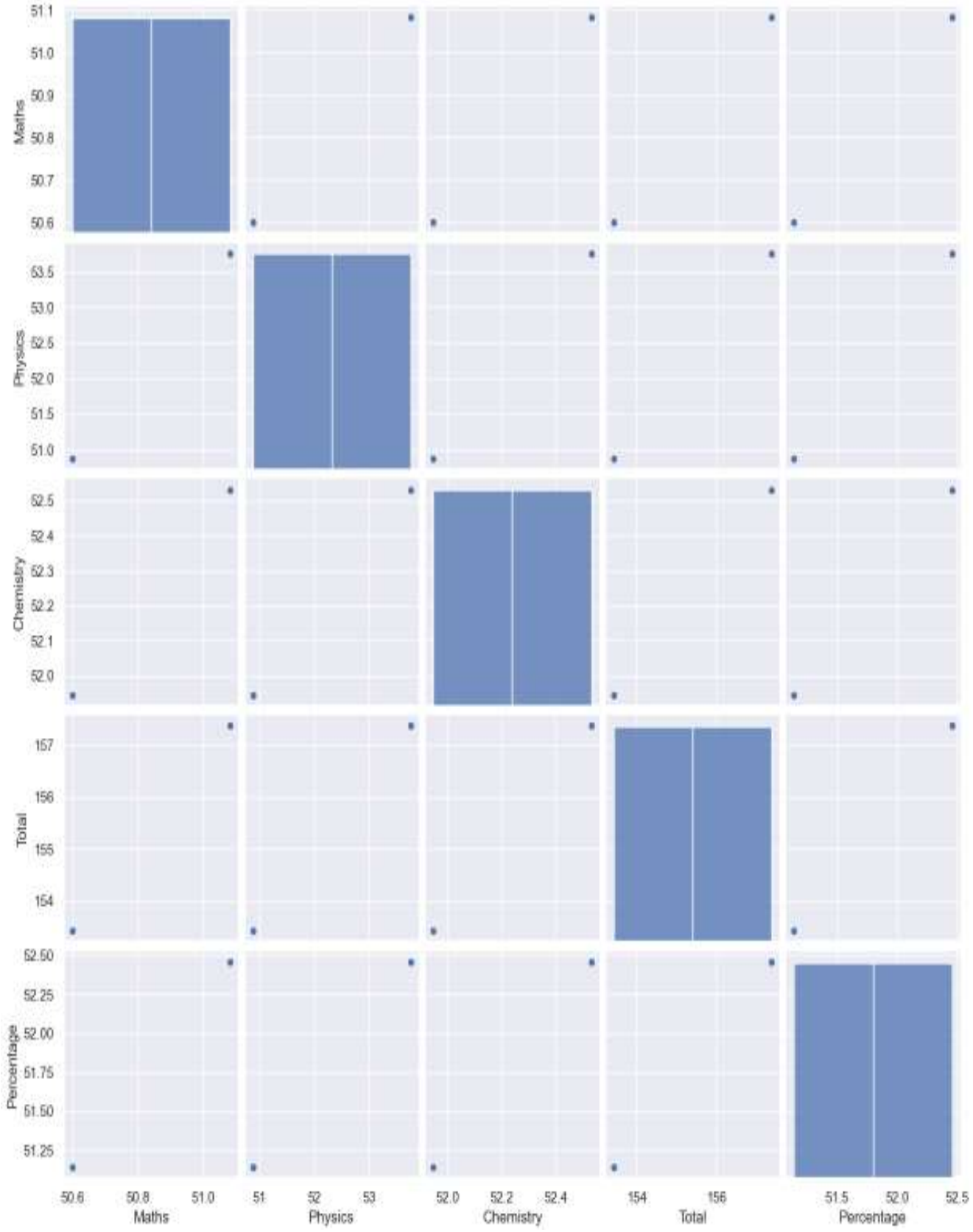
In [42]:
```
df4=df1.groupby(['Gender']).mean()
df1.groupby(['Gender']).mean()
```

Out[42]:

| Gender | Maths | Physics | Chemistry | Total | Percentage |
|---|---|---|---|---|---|
| **female** | 50.600 | 50.876 | 51.946 | 153.422 | 51.14074 |
| **male** | 51.084 | 53.756 | 52.532 | 157.372 | 52.45722 |

In [43]:
```
plt.figure(figsize=(25,10))
sns.pairplot(df4)
plt.show()
```

<Figure size 2500x1000 with 0 Axes>



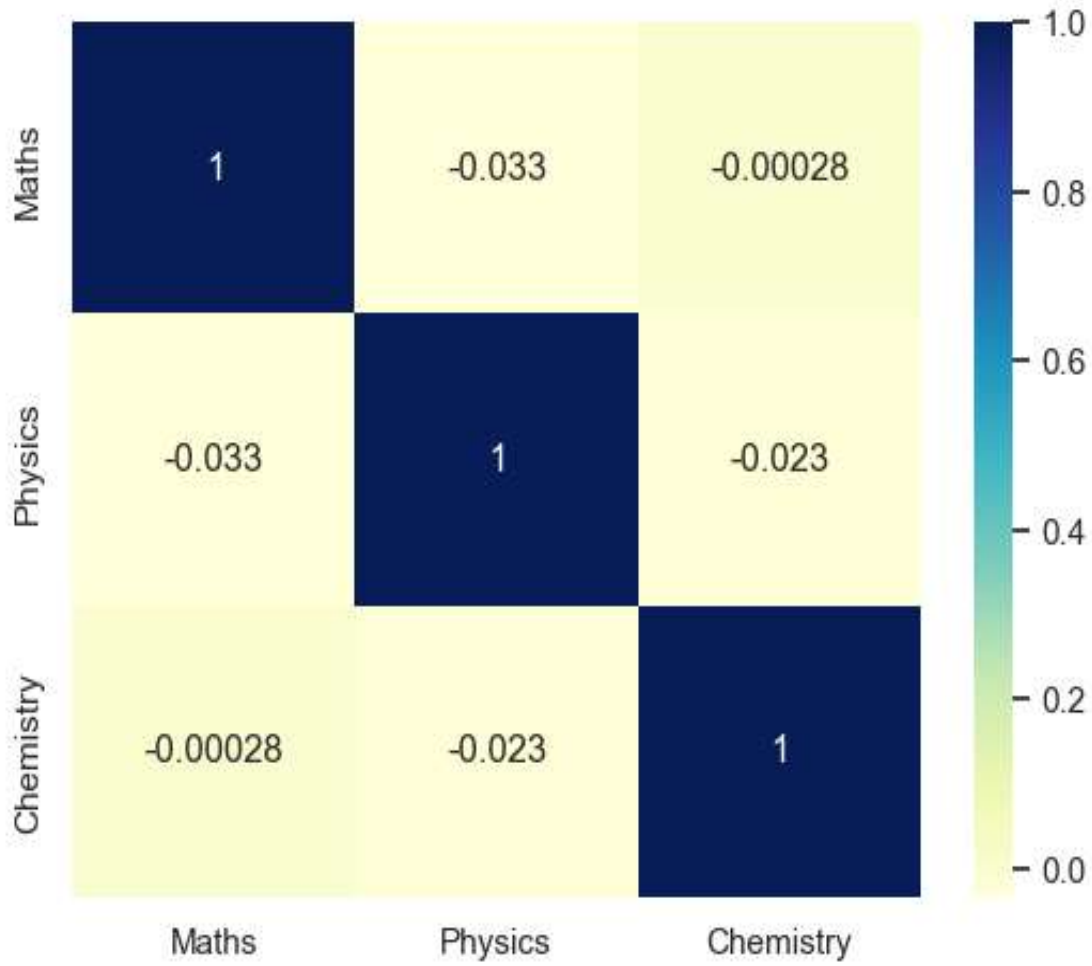
# Finding the Correlation of each column in the dataframe

In [46]:
```
df.corr()
```

Out[46]:

| | Maths | Physics | Chemistry |
|---|---|---|---|
| **Maths** | 1.000000 | -0.033428 | -0.000281 |
| **Physics** | -0.033428 | 1.000000 | -0.022521 |
| **Chemistry** | -0.000281 | -0.022521 | 1.000000 |

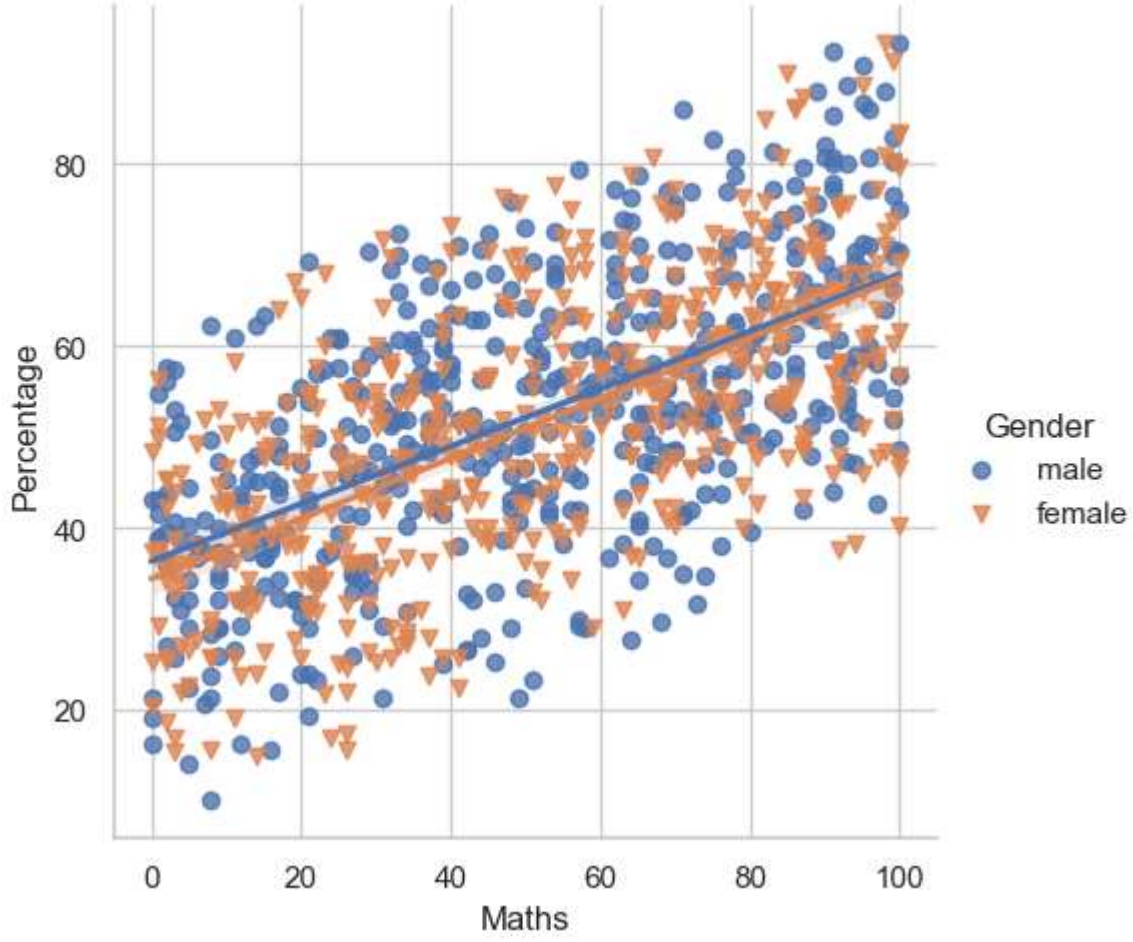
In [47]:
```
dataplot = sns.heatmap(df.corr(), cmap="YlGnBu", annot=True)
```



# Regression for Maths subject with Total Percentage for males and Females

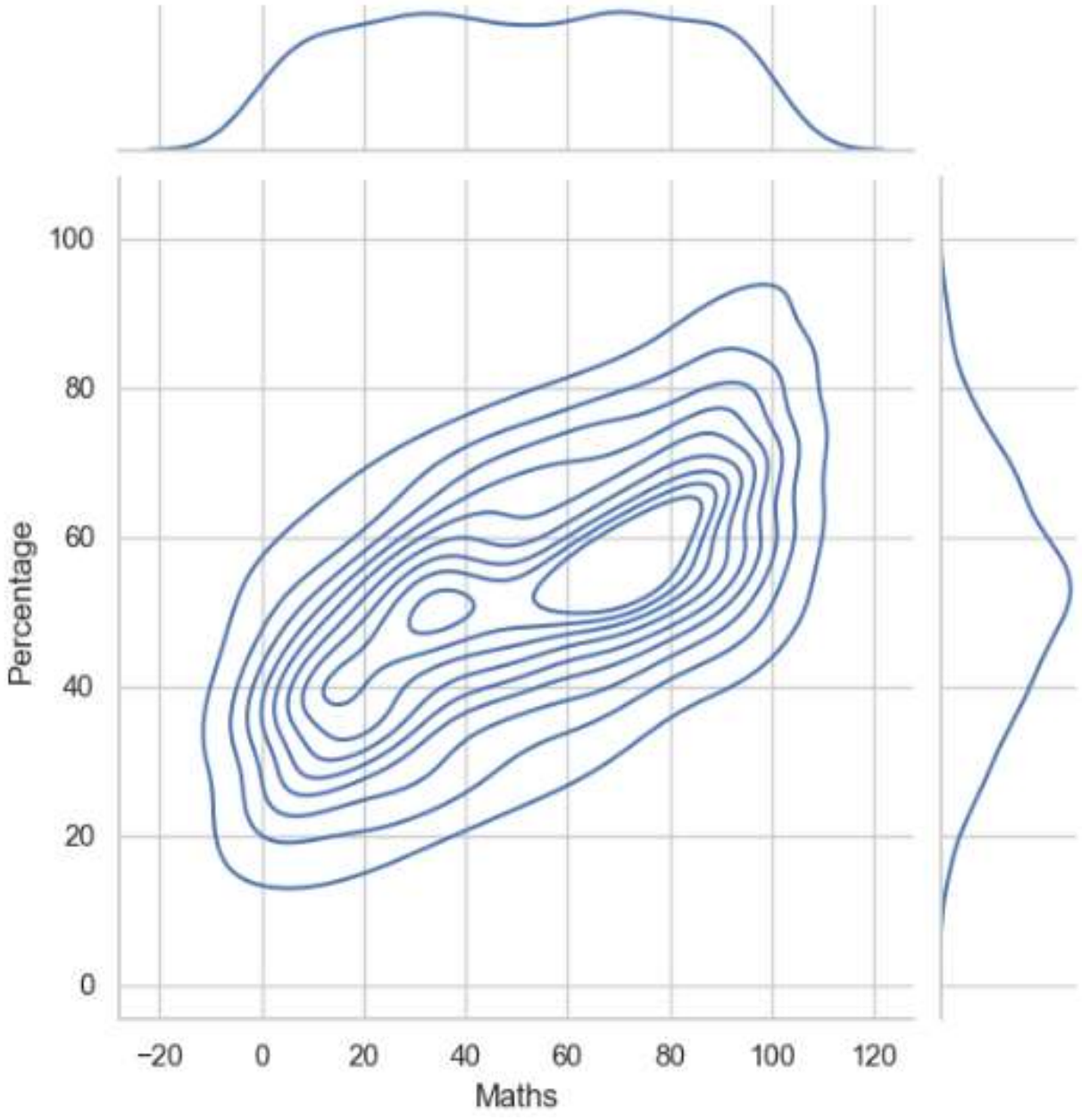
In [48]:
```
sns.set_style('whitegrid')
sns.lmplot(x ='Maths', y ='Percentage', data = df1,
           hue ='Gender', markers =['o', 'v'])
```

Out[48]: <seaborn.axisgrid.FacetGrid at 0x2006b116fd0>



```
In [49]: plt.figure(figsize=(20,10))
         sns.set_style('whitegrid')
         sns.jointplot(x ='Maths', y ='Percentage', data = df1, kind ='kde')
         plt.show()
```

```
<Figure size 2000x1000 with 0 Axes>
```



### 6.4 Data Wrangling

Data wrangling is the process of cleaning, transforming, and preparing data for analysis. In the case of student data analysis, data wrangling may involve tasks such as:

1. Cleaning and formatting: This may include removing duplicate data, correcting typos or errors, and formatting the data in a consistent way.
2. Merging data from different sources: Student data may come from multiple sources, such as enrollment data, attendance data, and performance data. These data sources may need to be merged together to create a complete picture of each student's academic history.
3. Creating new variables: Additional variables may need to be created to analyze the data effectively. For example, a variable representing the number of absences or the average grade for each student may be created.
4. Handling missing data: Some data may be missing or incomplete, and strategies for dealing with missing data, such as imputation, may need to be employed.
5. Filtering and subset selection: Depending on the research question, it may be necessary to filter the data and select only certain subsets of students for analysis.
6. Aggregating data: Data may need to be aggregated at various levels, such as by student, by class, or by school, to provide a summary of the data for analysis.
7. Reshaping data: The data may need to be reshaped from a wide to a long format, or from a long to a wide format, depending on the analysis.

Overall, data wrangling is an important step in preparing student data for analysis and ensuring that the results are accurate and reliable.

## 7. Observations/Summary

1. Student demographics: Analysis of student data may reveal information about the demographics of the student population, such as gender, age, ethnicity, socioeconomic status, and more. This information can be used to better understand the needs and challenges of different groups of students.
2. Academic performance: Analysis of student data can provide insights into academic performance, such as grades, test scores, and graduation rates. This information can help

identify areas of strength and weakness in the education system and inform interventions to improve student outcomes.

3. Attendance: Student attendance is a critical factor in academic success, and analysis of attendance data can provide insights into patterns of absenteeism and potential barriers to attendance. This information can inform strategies to improve attendance and student engagement.
4. Disciplinary actions: Analysis of disciplinary data can provide insights into patterns of behavior and potential issues within the school environment. This information can inform interventions to improve school climate and prevent future disciplinary incidents.
5. Educational programs: Analysis of data related to educational programs, such as enrollment in advanced courses or participation in extracurricular activities, can provide insights into opportunities and barriers to academic success. This information can inform interventions to increase access to educational programs and support student achievement.

# 8. Conclusion/Future Enhancements

## Conclusion:

Student data analysis is a crucial process that can provide valuable insights into the academic performance, attendance, behavior, and other factors that contribute to student success. Through data analysis, educators, administrators, and policymakers can identify areas of strength and weakness in the education system and develop targeted interventions to improve student outcomes. However, data analysis is not a one-time process and requires ongoing monitoring and evaluation to ensure that interventions are effective.

## Future enhancements:

There are several ways to enhance student data analysis in the future, including:

1. Integration of multiple data sources: To provide a more comprehensive picture of student performance and behavior, student data analysis can be enhanced by integrating data from

multiple sources, such as social-emotional learning assessments, health records, and community data.

2. Use of predictive analytics: Predictive analytics can be used to forecast future student performance and behavior based on historical data. This can help educators and administrators identify at-risk students and intervene before academic or behavioral issues arise.
3. Development of interactive dashboards: Interactive dashboards can provide educators and administrators with real-time access to student data and allow them to monitor progress, identify trends, and make data-driven decisions.
4. Utilization of machine learning: Machine learning algorithms can be applied to student data to identify patterns and make predictions about student outcomes. This can help educators and administrators tailor interventions to individual students and improve the effectiveness of interventions.

Overall, enhancing student data analysis through the integration of multiple data sources, the use of predictive analytics, the development of interactive dashboards, and the utilization of machine learning can help improve student outcomes and ensure that all students have the opportunity to reach their full potential.

# 9. Bibliography

1. Baker, R. S. J. d., & Yacef, K. (2009). The state of educational data mining in 2009: A review and future visions. Journal of Educational Data Mining, 1(1), 3-17.

2. Means, B., Bakia, M., & Murphy, R. (2014). Learning analytics: Measurement innovations to support personalized learning. Washington, DC: US Department of Education.

3. Wiliam, D. (2011). Embedded formative assessment. Bloomington, IN: Solution Tree Press.

4. K-12 Education Data Governance: Lessons Learned from Leading States. Data Quality Campaign, 2018.

5. The Power of Longitudinal Data Systems: Improving Student Achievement. Data Quality Campaign, 2019.

6. Data-Driven Decision Making: A Handbook for School Leaders. National Comprehensive Center for Teacher Quality, 2019.

7. The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling. Ralph Kimball and Margy Ross, 2013.

8. Big Data and Education: The Role of Data Science in Improving Educational Outcomes. Journal of Educational Data Mining, 2017.

9. Using Student Data to Drive Instruction: A Comprehensive Guide to Building and Implementing Data Systems in Schools. Victoria L. Bernhardt, 2017.

10. Improving Educational Outcomes Through Analytics and Data-Driven Decision Making. EDUCAUSE Review, 2018.