ព្រះរាជាណាចក្រកម្ពុជា

ជាតិ សាសនា ព្រះមហាក្សត្រ

Introduction to Data Science

Assignment

Topic: Students performance

Department: I3AMS 1-A

Name of Students	ID of Students	Score
1. HOEURN SREYKA	e20220699	…….
2. HEM BELLYDAY	e20220800	…….
2. HENG MENGHONG	e20220258	…….
3. HOK LYHOUR	e20220759	…….
4. HEANG SOTHEARA	e20221213	…….

Lecturer: **Dr. PHAUK Sokkhey (Course)**

Academic year 2024-2025

Table of Contents

I. Introduction

Education is a fundamental pillar of societal development, serving as the foundation for individual growth and collective progress. However, disparities in educational outcomes remain a pressing challenge, with factors such as socioeconomic background, family support, and access to resources significantly shaping student performance. Understanding these factors is crucial to addressing performance gaps and fostering equitable learning environments.

This report explores a dataset capturing demographic, socioeconomic, and academic information about students to uncover the underlying factors influencing their academic outcomes. The analysis delves into key aspects such as the impact of parental education levels, the effectiveness of test preparation courses, and the role of socioeconomic conditions in shaping performance. It also examines gender-based differences and correlations between subject-specific scores to provide a holistic view of student achievement.

By connecting data-driven insights to real-world challenges, this report aims to equip educators, policymakers, and institutions with actionable strategies to enhance learning outcomes and reduce disparities. The findings will highlight areas for targeted interventions, helping to create a more supportive and inclusive educational landscape for all students.

1. Problem statement

Understanding and addressing the factors that influence student performance is critical for fostering equitable and effective educational outcomes. Students' academic success is shaped by a complex interplay of individual attributes, family background, and external influences such as socioeconomic conditions and institutional support.

Despite widespread recognition of these influences, disparities in educational performance persist, particularly among students from varying socioeconomic and demographic backgrounds.

This report aims to identify and analyze key factors contributing to student performance, such as parental education levels, participation in test preparation courses, and access to resources like nutritious meals. By uncovering these relationships, the study seeks to address pressing questions:

- How do socioeconomic factors affect students' academic achievements?
- What is the impact of targeted interventions, such as test preparation courses, on performance outcomes?
- How do gender and parental education levels influence performance across different subjects?

The ultimate goal is to provide actionable insights that enable educators, policymakers, and institutions to design targeted strategies for improving academic outcomes, reducing disparities, and fostering a supportive learning environment for all students.

2. Objectives

- **Demographic and Socioeconomic Factors**: Explore how attributes such as gender, race/ethnicity, and socioeconomic status influence student performance. This involves identifying disparities among demographic groups and understanding how these variables intersect to affect academic outcomes. For instance, students from disadvantaged

socioeconomic backgrounds may face additional challenges, such as limited access to resources, which can directly impact their academic success.

- **Impact of Test Preparation**: Assess the role of test preparation courses in improving student performance. This includes evaluating whether students who complete these courses achieve higher scores in math, reading, and writing compared to those who do not. Understanding the effectiveness of such interventions helps in designing programs that bridge the performance gap for underperforming students.
- **Gender-Based Performance**: Analyze performance differences across subjects like math, reading, and writing based on gender. For example, identifying trends such as male students excelling in math and female students performing better in reading and writing can inform the development of tailored teaching methods and support systems.
- **Parental Education Levels**: Examine how the educational attainment of parents correlates with student achievement. This includes understanding the extent to which parental education impacts the availability of academic support, resources, and encouragement at home, thereby influencing students' motivation and outcomes.
- **Role of School-Provided Resources**: Evaluate the influence of essential resources, such as free or reduced lunch programs, on student performance. This analysis seeks to uncover how meeting basic needs enables students to focus better on academics and improve their overall performance.
- **Actionable Recommendations**: Develop specific, evidence-based strategies that educators and policymakers can implement to reduce academic disparities. These recommendations will focus on improving access to test preparation, addressing resource gaps, and fostering inclusive practices to ensure all students have equal opportunities to succeed.
- **Future Research Insights**: Identify patterns and gaps in the current data to guide further research. This involves exploring additional factors, such as extracurricular activities, teaching quality, and learning environments, to enhance educational equity and academic outcomes on a broader scale.

II. Dataset and Data Description

• Dataset: “student performance”
• Source: Kaggle https://www.kaggle.com/datasets/rkiattisak/student-performance-in-mathematics
• Purpose: To analyze factors influencing student performance, including demographics, socioeconomic conditions, and test preparation, while offering strategies to promote equitable education.
• The dataset contains 1000 rows and 8 columns.

The dataset used in this analysis includes information on 1,000 students, with attributes such as gender, race/ethnicity, parental education, lunch type, test preparation completion, and scores in math, reading, and writing. Key features include:

- **Gender**: Male or Female.
- **Race/Ethnicity**: Categorized into groups.
- **Parental Education**: Highest education level attained by parents.
- **Lunch**: Type of lunch provided (standard or free/reduced).
- **Test Preparation**: Completion status of test preparation courses.
- **Scores**: Performance metrics in math, reading, and writing.

III. Methodology and Workflows

Methodology

Python Libraries:

- **Pandas**: "Data Structures for Statistical Computing in Python
- **NumPy**: "Array programming with NumPy." *Nature*.
- **Scikit-learn**: Machine Learning in Python.
- **Matplotlib**: *Computing in Science & Engineering*.
- **Seaborn**: Statistical Data Visualization

Methods Used

- **Linear Regression**: Implementation via LinearRegression from Scikit-learn to analyze and predict scores.
- **Train-Test Splitting**: Data splitting performed using train_test_split from Scikit-learn.
- **Performance Metrics**:
  - Mean Squared Error (MSE): Used to evaluate the accuracy of regression models.
  - R-squared (R^2): Utilized to measure the explanatory power of the models.

Visualization:

- Data visualizations, including scatter plots, boxplots, and correlation heatmaps, were created using Matplotlib and Seaborn to explore trends and insights.

Feature Engineering:

- One-hot encoding of categorical variables was performed using Pandas' get_dummies method for model compatibility.

Workflows

1. **Data cleaning process**

Before building the machine learning model, it is essential to ensure the dataset is clean, consist, and ready for analysis. Therefore, data cleaning is the foundational step in the data science process because the model's dependability and performance are directly impacted by the quality of the data. The following are the process we do to clean the data:

- importing libraries
- load the data

```
# Load the dataset from your file
data = pd.read_excel('data.xlsx', skiprows=3)
```

- display the row datasets

```
# Display the first few rows of the dataset
print(data.head())
```

out put:

	Gender	Race/Ethnicity	Parental Education	Lunch	TestPreparation	MathScore	ReadingScore	WritingScore
0	Female	Group A	Bachelor's Degree	Standard	Completed	78	85	88
1	Male	Group B	High School	Reduced	None	65	72	70
2	Female	Group C	Some College	Standard	Completed	80	88	92
3	Female	Group B	Associate's Degree	Reduced	Completed	74	79	85
4	Male	Group A	Master's Degree	Standard	None	85	89	90

- **Inspect the ending rows of the dataset and Check the number of rows and columns**

```
print(data.tail())
data.shape
```

- **update the columns name**

just write the new name for index for easy looking.

```
# Rename the columns for easier access
data.columns = ['Gender', 'Race_Ethnicity', 'Parental_Education', 'Lunch',
                'Preparation_Course', 'Math_Score', 'Reading_Score', 'Writing_Score']

# Display the updated column names
print(data.columns)
```

✓ 0.0s

```
Index(['Gender', 'Race_Ethnicity', 'Parental_Education', 'Lunch',
       'Preparation_Course', 'Math_Score', 'Reading_Score', 'Writing_Score'],
      dtype='object')
```

- **check the missing values**

```
# Check for missing values
print(data.isnull().sum())

# Drop rows with missing values
data.dropna(inplace=True)

# Verify that there are no more missing values
print(data.isnull().sum())
```

✓ 0.0s

Out put

```
Gender                  0
Race_Ethnicity          0
Parental_Education      0
Lunch                   0
Preparation_Course      0
Math_Score              0
Reading_Score           0
Writing_Score           0
dtype: int64
Gender                  0
Race_Ethnicity          0
Parental_Education      0
Lunch                   0
Preparation_Course      0
Math_Score              0
Reading_Score           0
Writing_Score           0
dtype: int64
```

- **Remove the duplicate row and check the duplicate row**

```
# Remove duplicate rows
data.drop_duplicates(inplace=True)

# Check for duplicates
print(f"Number of duplicate rows: {data.duplicated().sum()}")
```

✓ 0.0s

2. Exploratory Data Analysis

Summary Statistics

The dataset provides valuable insights into both numerical and categorical attributes, enabling a comprehensive understanding of student performance trends:

```
# Summary of numerical columns
print(data.describe())

# Summary of categorical columns
print(data.describe(include=['object']))
```

```
       Math_Score  Reading_Score  Writing_Score
count  1000.000000    1000.000000    1000.000000
mean     67.810000      70.382000      69.140000
std      15.250196      14.107413      15.025917
min      15.000000      25.000000      15.000000
25%      58.000000      61.000000      59.000000
50%      68.000000      70.500000      70.000000
75%      79.250000      80.000000      80.000000
max     100.000000     100.000000     100.000000
       Gender Race_Ethnicity Parental_Education     Lunch Preparation_Course
count    1000           1000               1000      1000               1000
unique      2              5                  6         2                  2
top      male        group C       some college  standard               none
freq      508            323                224       660                656
```

- **Math Scores**:
  - The average math score is 67.81, with a standard deviation of 15.25, indicating moderate variability in performance.
  - The minimum score is 15, while the maximum is 100, showcasing a wide range of capabilities among students.
  - The 25th percentile is 58, the median (50th percentile) is 68, and the 75th percentile is 79.25, suggesting that most students score within a mid-to-high range.
- **Reading Scores**:
  - The mean reading score is 70.38, with a standard deviation of 14.11, slightly less variable than math scores.
  - Scores range from 25 to 100, with the middle 50% of students scoring between 61 and 80.
- **Writing Scores**:
  - The mean writing score is 69.14, with a standard deviation of 15.03, showing similar variability to math scores.

- Scores range from 15 to 100, and the interquartile range (59 to 80) highlights consistent performance patterns among most students.

Categorical Columns

- **Gender**:
  - The dataset is nearly balanced with 508 male students and 492 female students, providing equitable representation for gender-based analysis.
- **Race/Ethnicity**:
  - Students are divided into six groups, with Group C being the largest (323 students). This provides an opportunity to analyze group-specific trends in performance.
- **Parental Level of Education**:
  - Parental education levels range from "high school" to "master's degree," with "some college" being the most common category (224 students). This variation allows for an analysis of the relationship between parental education and student outcomes.
- **Lunch**:
  - A majority of students (660) receive standard lunch, while 340 students are on free/reduced lunch. This distinction serves as a proxy for socioeconomic status in the analysis.
- **Test Preparation Course**:
  - 344 students completed the test preparation course, while 656 did not. This disparity provides a basis for evaluating the impact of test preparation on academic performance.

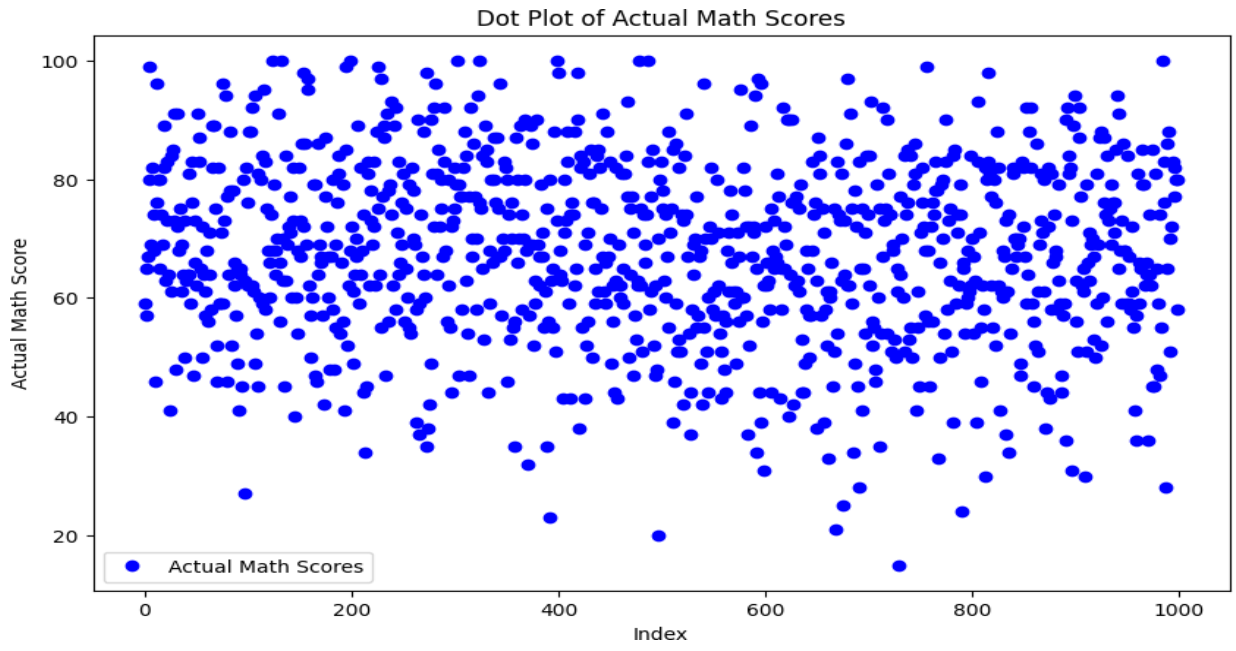
Exploratory Data Analysis (EDA) revealed key insights into student performance patterns and relationships among variables. Below are the major findings:

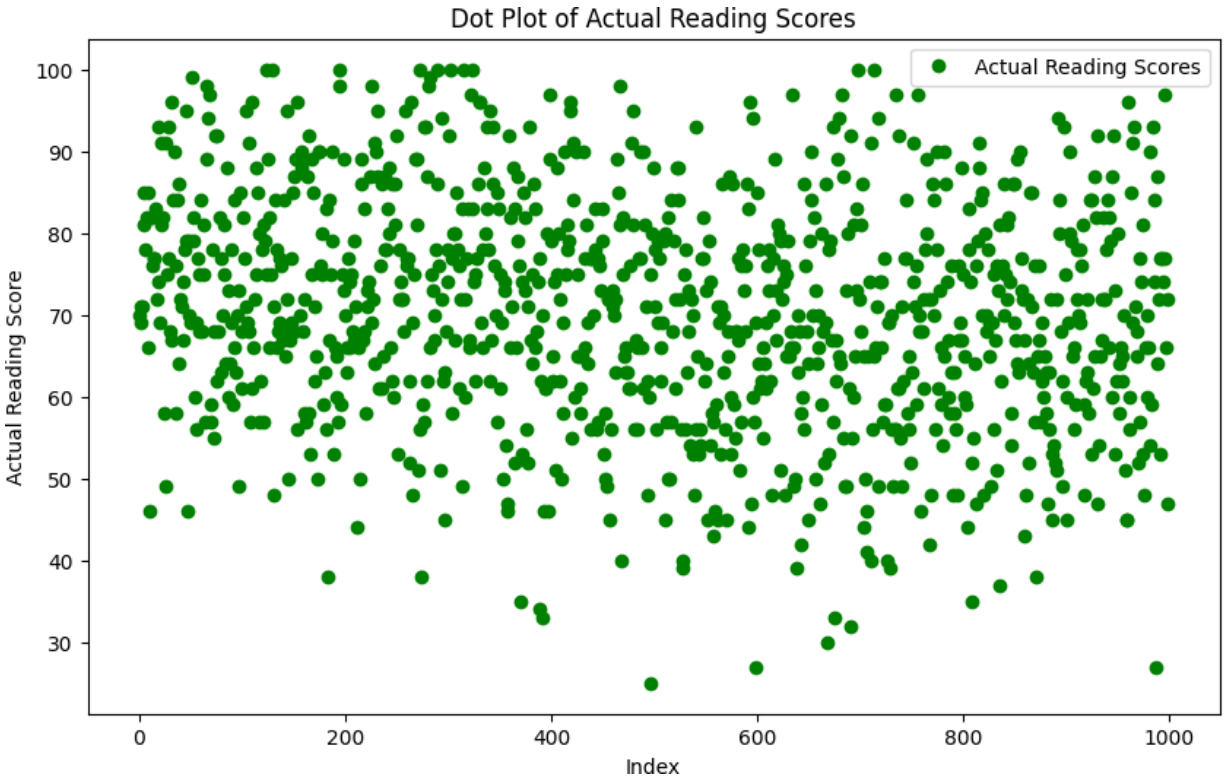
- **Distribution of Scores**
- **Lunch type**
- **Test Preparation Course**
- **Gender**
- **Race/Ethnicity**
- **Parental Level of Education**

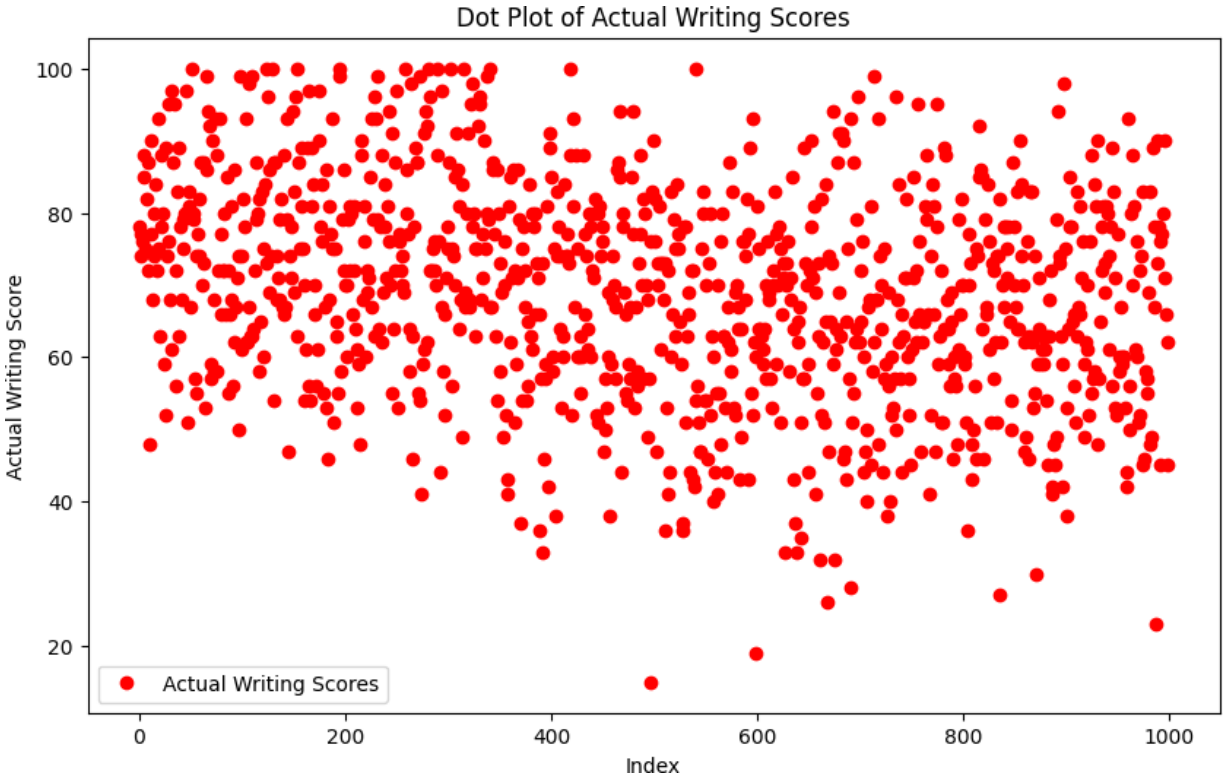
1. Distribution of Scores

Histograms and Box Plots: Showed score distributions and outliers.

```python
import matplotlib.pyplot as plt

# Assuming y_test_math, y_test_reading, and y_test_writing are loaded
y_test_math = data['Math_Score']  # Replace with appropriate variable if different
y_test_reading = data['Reading_Score']
y_test_writing = data['Writing_Score']

# Define a function for plotting dot plots
def plot_dot(scores, color, title, ylabel):
    plt.figure(figsize=(10, 6))
    plt.plot(scores, f'{color}o', label=f'Actual {title} Scores')  # Colored dots
    plt.xlabel('Index')
    plt.ylabel(ylabel)
    plt.title(f'Dot Plot of Actual {title} Scores')
    plt.legend()
    plt.show()

# Plot Math, Reading, and Writing Scores
plot_dot(y_test_math, 'b', 'Math', 'Actual Math Score')
plot_dot(y_test_reading, 'g', 'Reading', 'Actual Reading Score')
plot_dot(y_test_writing, 'r', 'Writing', 'Actual Writing Score')
```



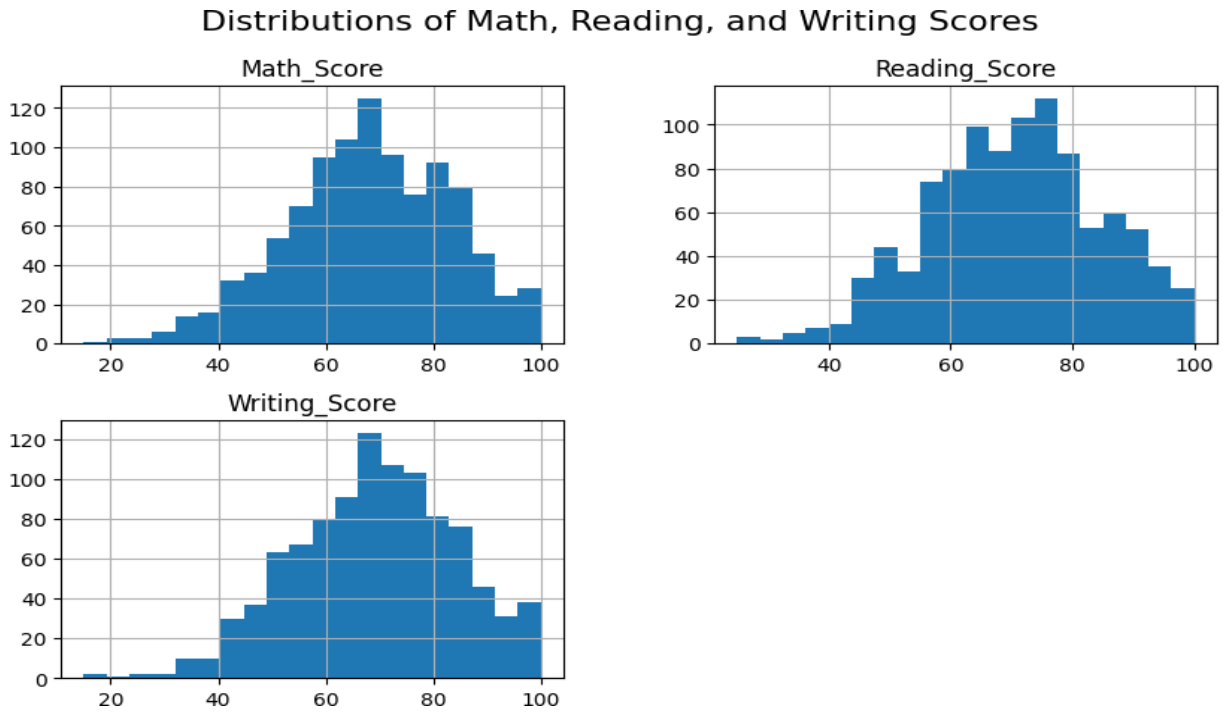

Dot Plot of Actual Reading Scores
Actual Reading Scores
Actual Reading Score
Index
100
90
80
70
60
50
40
30
0
200
400
600
800
1000


Dot Plot of Actual Writing Scores
Actual Writing Scores
Actual Writing Score
Index
100
80
60
40
20
0
200
400
600
800
1000

Below is the histogram that make us easy to understand more than dot plot.

The histograms illustrate the distributions of scores in math, reading, and writing for students in the dataset

```
# Plot histograms for all subjects
data[['Math_Score', 'Reading_Score', 'Writing_Score']].hist(bins=20, figsize=(10, 6))
plt.suptitle('Distributions of Math, Reading, and Writing Scores', fontsize=16)
plt.show()
```



Here's a detailed breakdown:

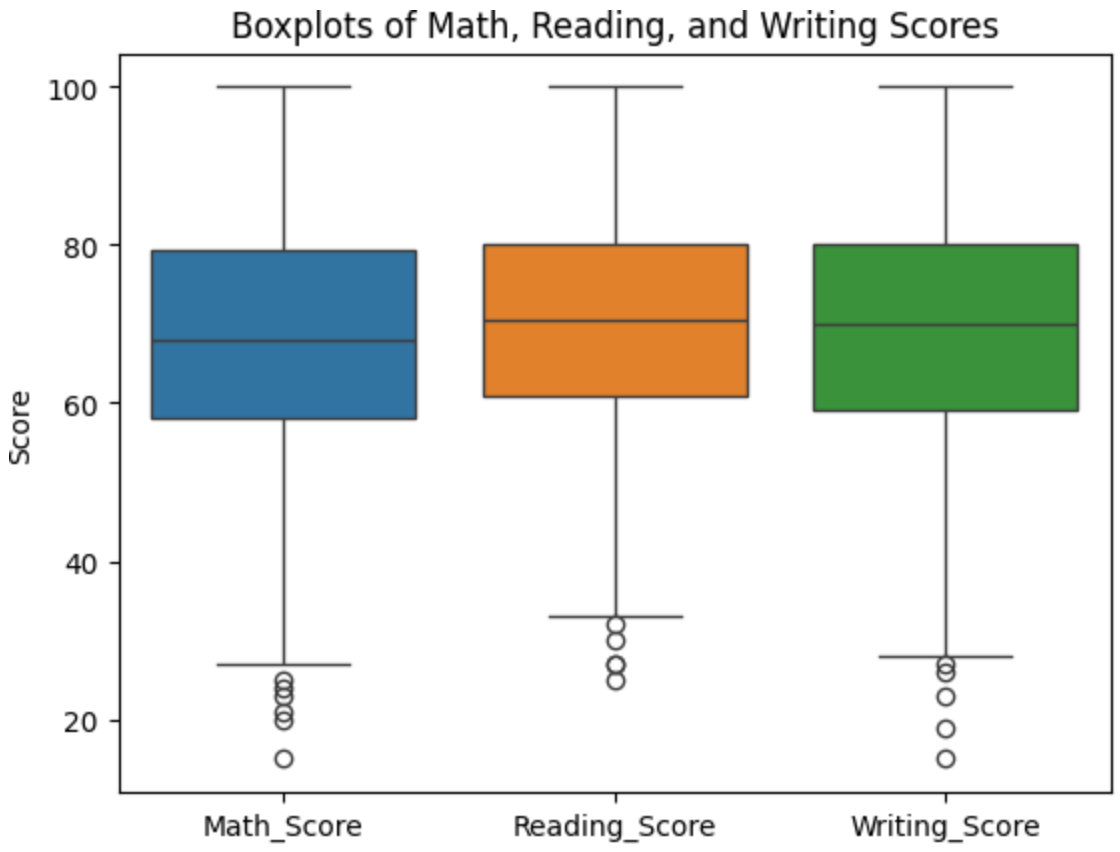
- **Math Scores**:
  - **Shape**: The math scores exhibit a slightly right-skewed distribution, with most scores clustering between 50 and 80.
  - **Peak Frequency**: The highest number of students scored around the 70 mark.
  - **Range**: Scores range from approximately 20 to 100, indicating a wide variation in math performance.
  - **Outliers**: A small number of students scored exceptionally low, pulling the distribution slightly to the right.
- **Reading Scores**:
  - **Shape**: The distribution is approximately symmetric, resembling a normal distribution.

- **Peak Frequency**: The most common scores fall between 70 and 80, suggesting higher performance in reading compared to math.
- **Range**: Scores range from about 40 to 100, indicating less variation compared to math.

- **Writing Scores**:
  - **Shape**: Similar to reading scores, writing scores display a near-normal distribution.
  - **Peak Frequency**: The majority of students scored between 65 and 80.
  - **Range**: Scores range from approximately 20 to 100, with a distribution that closely mirrors reading scores.

The box plot provides a comparative summary of the distribution of scores for math, reading, and writing subjects.

```
# Boxplots for all subjects
sns.boxplot(data=data[['Math_Score', 'Reading_Score', 'Writing_Score']])
plt.title('Boxplots of Math, Reading, and Writing Scores')
plt.ylabel('Score')
plt.show()
```



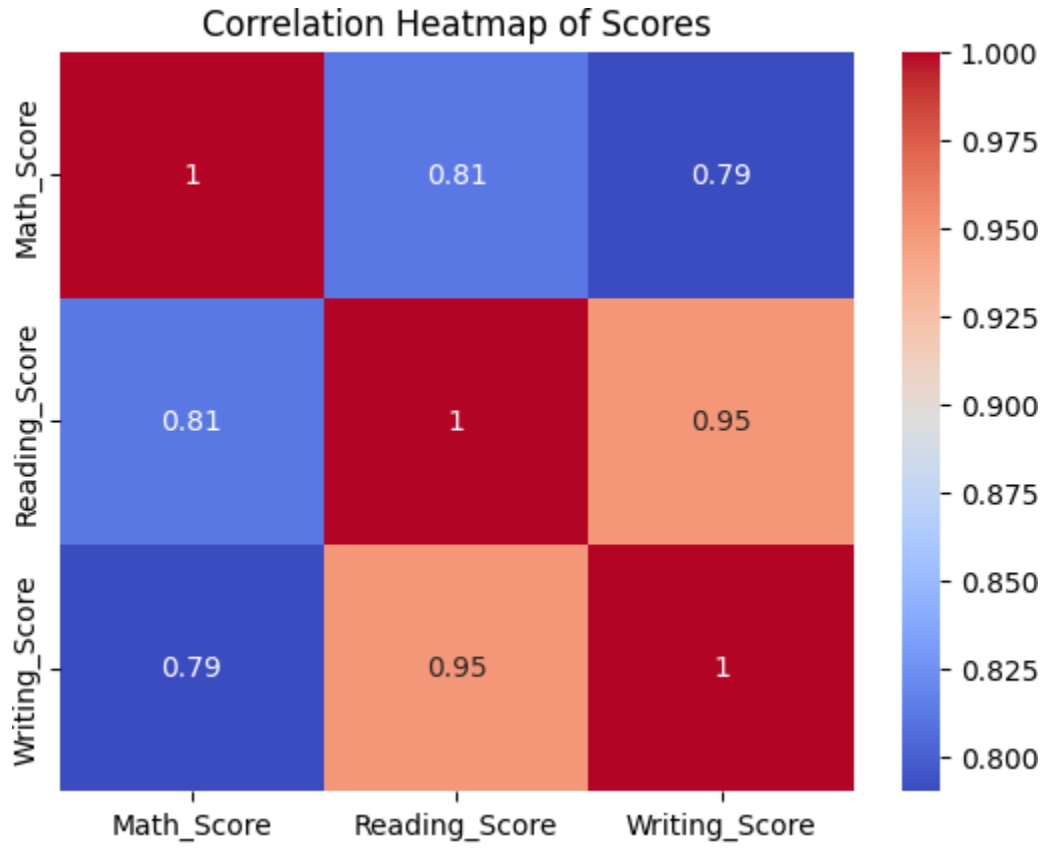
Here are the key observations:

- **Central Tendency**:
  - The median (central line in each box) for math scores is slightly lower than that for reading and writing. This indicates that students generally perform better in reading and writing compared to math.
  - Reading and writing scores have similar medians, reflecting comparable performance levels in these two subjects.
- **Range of Scores**:
  - Math scores have a wider range (indicated by the whiskers), extending from 15 to 100. This suggests greater variability in math performance.
  - Reading and writing scores are slightly more consistent, with a narrower range compared to math.
- **Outliers**:
  - Outliers (points outside the whiskers) are visible for all three subjects. These represent students who performed exceptionally lower than the general cohort.
  - Math has the most visible outliers, indicating that a subset of students struggled significantly in this subject.
- **Spread (Interquartile Range - IQR)**:
  - The IQR (height of each box) for all three subjects is relatively similar, indicating consistent spread in the middle 50% of scores.
  - Writing has a slightly larger IQR compared to reading, reflecting a broader performance range among average performers.
- **Performance Insights**:
  - Students appear to perform slightly better in reading and writing compared to math, as indicated by higher medians and fewer extreme low scores.

3. Correlation Heatmap

The heatmap visualizes the relationships between the three academic performance metrics: math, reading, and writing scores.

```
import seaborn as sns
import matplotlib.pyplot as plt

# Compute the correlation matrix
correlation_matrix = data[['Math_Score', 'Reading_Score', 'Writing_Score']].corr()

# Plot the heatmap
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap of Scores')
plt.show()
```



Here's what it reveals:

- **Correlation Strength**:
  - **Math and Reading**: Correlation coefficient of 0.81 indicates a strong positive relationship, meaning students who score well in reading are also likely to perform well in math.
  - **Math and Writing**: Correlation coefficient of 0.79 shows a moderately strong positive relationship between these subjects.
  - **Reading and Writing**: Correlation coefficient of 0.95 highlights an exceptionally strong positive relationship, suggesting that students tend to excel in both literacy-based subjects simultaneously.
- **Implications**:
  - The high correlation between reading and writing scores suggests overlapping skill sets or shared influencing factors, such as language proficiency.
  - The slightly lower correlation of math scores with the other two subjects may indicate that mathematical ability is influenced by different cognitive or instructional factors.
- **Use in Modeling**:
  - These correlations provide insights into feature relationships, which can improve the performance of predictive models by considering how these scores interact.

Lunch-Based Summary Statistics Analysis

```python
# Group scores by lunch type and calculate summary statistics
lunch_analysis = data.groupby('Lunch')[['Math_Score', 'Reading_Score', 'Writing_Score']].describe()

# Simplify the multi-index column names for better readability
lunch_analysis.columns = [f"{stat}_{score}" for score, stat in lunch_analysis.columns]

# Reset index for better formatting
lunch_analysis.reset_index(inplace=True)

# Format and print the output
print("Lunch-Based Summary Statistics:")
print("=" * 70)
for _, row in lunch_analysis.iterrows():
    print(f"Lunch Type: {row['Lunch']}")
    print("=" * 70)
    print(f"{'Statistic':<15} {'Math':<10} {'Reading':<10} {'Writing':<10}")
    print("-" * 70)
    print(f"{'Mean':<15} {row['mean_Math_Score']:<10.2f} {row['mean_Reading_Score']:<10.2f} {row['mean_Writing_Score']:<10.2f}")
    print(f"{'25% (Q1)':<15} {row['25%_Math_Score']:<10.2f} {row['25%_Reading_Score']:<10.2f} {row['25%_Writing_Score']:<10.2f}")
    print(f"{'50% (Median)':<15} {row['50%_Math_Score']:<10.2f} {row['50%_Reading_Score']:<10.2f} {row['50%_Writing_Score']:<10.2f}")
    print(f"{'75% (Q3)':<15} {row['75%_Math_Score']:<10.2f} {row['75%_Reading_Score']:<10.2f} {row['75%_Writing_Score']:<10.2f}")
    print(f"{'Min':<15} {row['min_Math_Score']:<10.2f} {row['min_Reading_Score']:<10.2f} {row['min_Writing_Score']:<10.2f}")
    print(f"{'Max':<15} {row['max_Math_Score']:<10.2f} {row['max_Reading_Score']:<10.2f} {row['max_Writing_Score']:<10.2f}")
    print(f"{'Std':<15} {row['std_Math_Score']:<10.2f} {row['std_Reading_Score']:<10.2f} {row['std_Writing_Score']:<10.2f}")
    print(f"{'Count':<15} {row['count_Math_Score']:<10.0f} {row['count_Reading_Score']:<10.0f} {row['count_Writing_Score']:<10.0f}")
    print("=" * 70)
```

```
Lunch-Based Summary Statistics:
======================================================================
Lunch Type: free/reduced
======================================================================
Statistic       Math       Reading    Writing
----------------------------------------------------------------------
Mean            59.90      65.64      64.24
25% (Q1)        50.00      57.00      54.00
50% (Median)    61.00      66.00      64.00
75% (Q3)        70.00      75.00      75.00
Min             15.00      25.00      15.00
Max             93.00      98.00      99.00
Std             13.97      13.24      14.44
Count           340        340        340
======================================================================
Lunch Type: standard
======================================================================
Statistic       Math       Reading    Writing
----------------------------------------------------------------------
Mean            71.88      72.82      71.67
25% (Q1)        62.00      64.00      62.00
50% (Median)    72.50      73.00      72.00
75% (Q3)        82.00      83.00      81.00
Min             21.00      27.00      19.00
Max             100.00     100.00     100.00
Std             14.26      13.93      14.70
Count           660        660        660
======================================================================
```

The table presents summary statistics for student performance in math, reading, and writing, categorized by lunch type (free/reduced and standard). Here's a detailed analysis:

Free/Reduced Lunch

- **Mean Scores**:
  - Students with free/reduced lunch have lower mean scores: Math (59.90), Reading (65.64), Writing (64.24).
- **Score Spread**:
  - Scores are distributed in the lower range, with a 75th percentile (Q3) of 70 (Math), 75 (Reading), and 75 (Writing).
- **Variability**:
  - Standard deviations (Std) indicate moderate variability: Math (13.97), Reading (13.24), Writing (14.44).
- **Performance Insight**:
  - Students receiving free/reduced lunch generally underperform, likely reflecting challenges linked to socioeconomic factors.

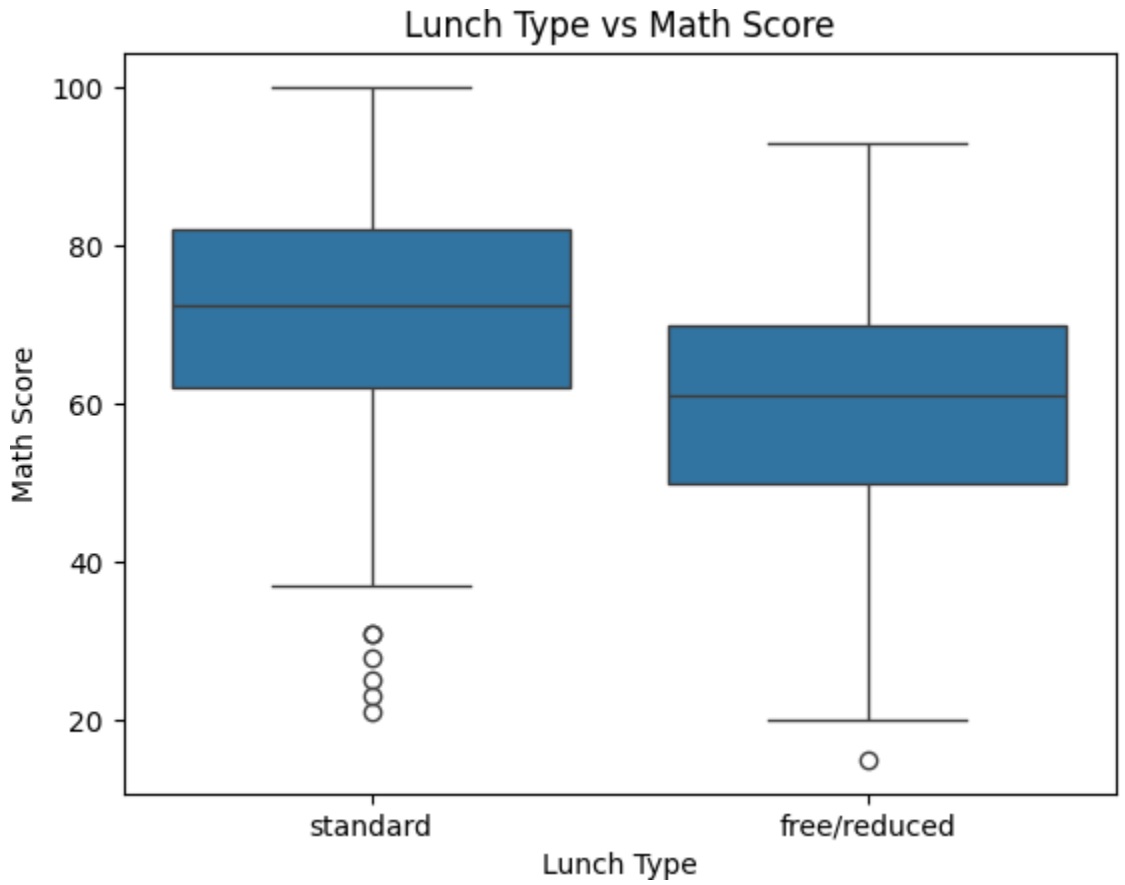
Standard Lunch

- **Mean Scores**:
  - Students with standard lunch show higher mean scores: Math (71.88), Reading (72.82), Writing (71.67).
- **Score Spread**:
  - The 75th percentile (Q3) scores are significantly higher: Math (82), Reading (83), Writing (81).
- **Variability**:
  - Standard deviations are slightly higher: Math (14.26), Reading (13.93), Writing (14.70), indicating a broader spread in performance.
- **Performance Insight**:
  - Students with standard lunch outperform their peers, suggesting greater stability and access to resources.

Analysis of the Box Plot: Lunch Type vs. Math Score

This box plot visualizes the relationship between lunch type (standard and free/reduced) and math scores.

```python
import seaborn as sns
import matplotlib.pyplot as plt

# Boxplot: Lunch type vs Math Score
sns.boxplot(x='Lunch', y='Math_Score', data=data)
plt.title('Lunch Type vs Math Score')
plt.xlabel('Lunch Type')
plt.ylabel('Math Score')
plt.show()
```



Here is the analysis:

- **Median Scores**:
  - Students with a **standard lunch** have a higher median math score compared to those with **free/reduced lunch**. This indicates better overall performance for students receiving standard lunches.
- **Score Spread**:

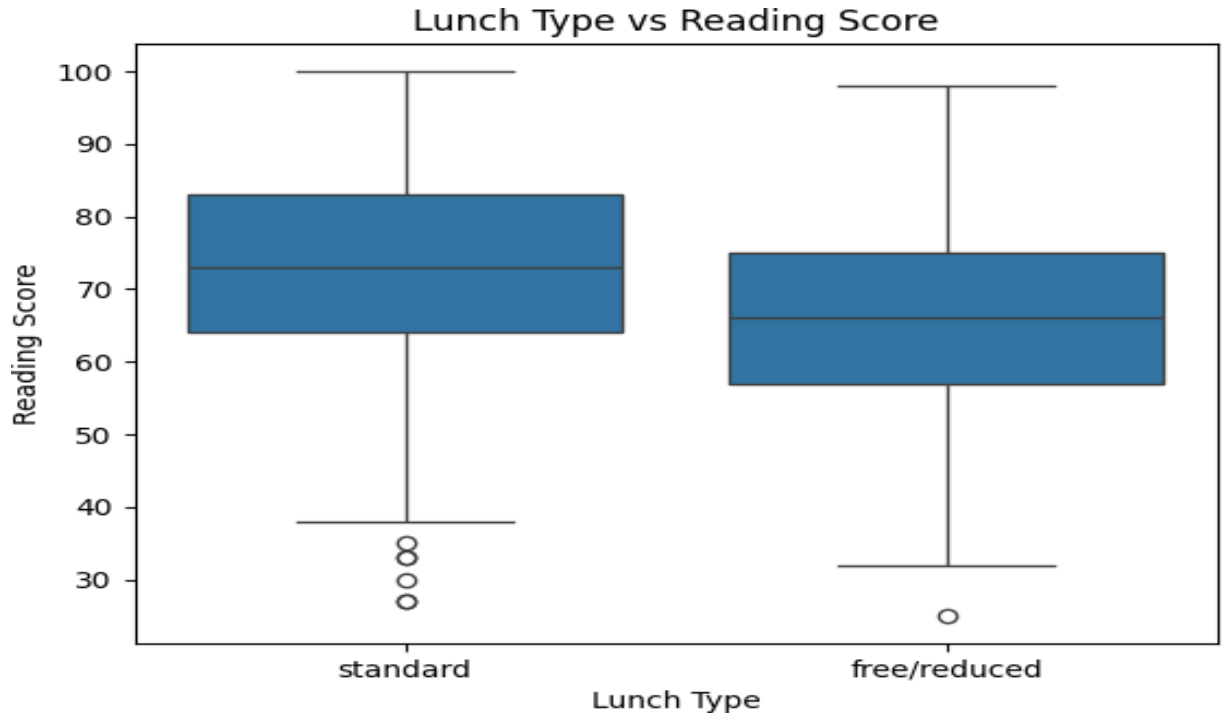
- **Standard Lunch**: The math scores show a broader range, with most scores concentrated in the upper range. A few outliers are present in the lower scores.
- **Free/Reduced Lunch**: The score range is narrower, and most students score in the lower range compared to those with standard lunches.

- **Outliers**:
  - Both groups show outliers, but the **free/reduced lunch** group has fewer, indicating a more consistent but lower performance.

Analysis of the Box Plot: Lunch Type vs. Reading Score

This box plot illustrates the relationship between lunch type (standard and free/reduced) and reading scores.

```
# Boxplot: Lunch type vs Reading Score
sns.boxplot(x='Lunch', y='Reading_Score', data=data)
plt.title('Lunch Type vs Reading Score')
plt.xlabel('Lunch Type')
plt.ylabel('Reading Score')
plt.show()
```



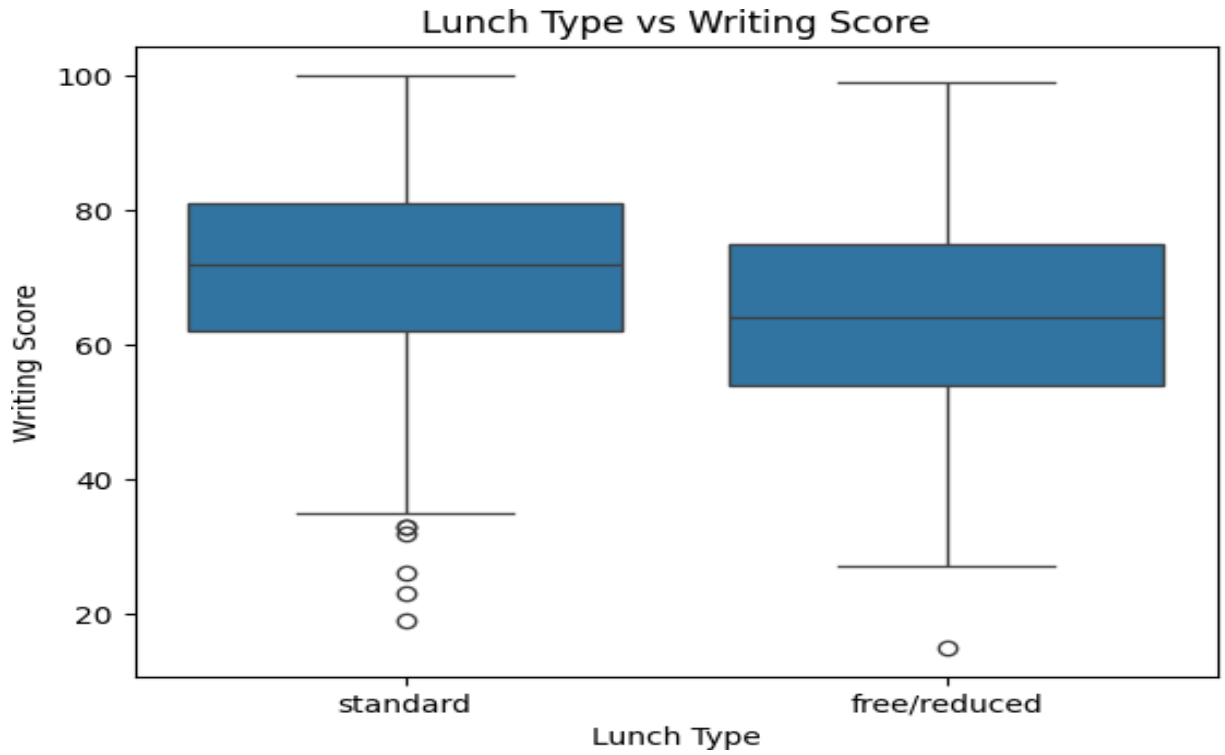
Observations:

- **Median Scores**:
  - Students with a **standard lunch** have a higher median reading score compared to those with **free/reduced lunch**. This indicates better reading performance for students with standard lunch access.
- **Score Range**:
  - **Standard Lunch**: The reading scores are distributed across a broader range, with most scores concentrated in the upper range. There are some outliers at the lower end of the distribution.
  - **Free/Reduced Lunch**: The score range is narrower, and the concentration of scores is more focused in the lower ranges compared to students with standard lunches.
- **Outliers**:
  - Both groups have outliers, but the **free/reduced lunch** group has fewer, indicating relatively consistent but lower performance overall.

Analysis of the Box Plot: Lunch Type vs. Writing Score

This box plot visualizes the relationship between lunch type (standard and free/reduced) and writing scores. Here's the detailed analysis:

```
# Boxplot: Lunch type vs Writing Score
sns.boxplot(x='Lunch', y='Writing_Score', data=data)
plt.title('Lunch Type vs Writing Score')
plt.xlabel('Lunch Type')
plt.ylabel('Writing Score')
plt.show()
```



Observations:

- **Median Scores**:
  - Students with a **standard lunch** have a higher median writing score compared to those with **free/reduced lunch**, indicating better writing performance for students with standard lunch access.
- **Score Range**:
  - **Standard Lunch**: Writing scores show a broader distribution, with most scores concentrated in the higher ranges and some outliers in the lower range.
  - **Free/Reduced Lunch**: Scores are more narrowly distributed, with lower overall performance compared to the standard lunch group.
- **Outliers**:
  - Outliers are present in both groups. The **free/reduced lunch** group has a prominent low outlier, suggesting some students face significant challenges in writing performance.

Summary of Average Scores by Lunch Type

This table provides a summary of the mean scores for math, reading, and writing, grouped by lunch type (free/reduced and standard).

```
# Calculate the mean scores grouped by lunch type
average_scores_by_lunch = data.groupby('Lunch')[['Math_Score', 'Reading_Score', 'Writing_Score']].mean()

# Display the average scores
print(average_scores_by_lunch)
```

```
             Math_Score  Reading_Score  Writing_Score
Lunch
free/reduced   59.900000      65.641176      64.235294
standard       71.884848      72.824242      71.666667
```

Observations:

- **Free/Reduced Lunch**:
  - **Math Score**: The average is **59.90**, indicating lower performance.
  - **Reading Score**: The average is **65.64**, showing moderate performance.
  - **Writing Score**: The average is **64.24**, slightly higher than math but still below the standard lunch group.
- **Standard Lunch**:
  - **Math Score**: The average is **71.88**, significantly higher than the free/reduced lunch group.
  - **Reading Score**: The average is **72.82**, indicating strong performance.
  - **Writing Score**: The average is **71.67**, reflecting consistently high scores across all subjects.

Overall Conclusion

The analysis reveals a clear disparity in academic performance based on lunch type. Students who receive standard lunches consistently outperform those on free/reduced lunches across all subjects: math, reading, and writing. This performance gap underscores the significant influence of socioeconomic factors on educational outcomes.

Preparation-Based Summary Statistics

The table highlights the mean scores and key statistics for students who completed the test preparation course compared to those who did not.

```python
# Group scores by test preparation course and calculate summary statistics
prep_analysis = data.groupby('Preparation_Course')[['Math_Score', 'Reading_Score', 'Writing_Score']].describe()

# Simplify multi-index column names for better readability
prep_analysis.columns = [f"{stat}_{score}" for score, stat in prep_analysis.columns]

# Reset index for better formatting
prep_analysis.reset_index(inplace=True)

# Format and print the output
print("Preparation-Based Summary Statistics:")
print("=" * 70)
for _, row in prep_analysis.iterrows():
    print(f"Preparation Course: {row['Preparation_Course']}")
    print("=" * 70)
    print(f"{'Statistic':<15} {'Math':<10} {'Reading':<10} {'Writing':<10}")
    print("-" * 70)
    print(f"{'Mean':<15} {row['mean_Math_Score']:<10.2f} {row['mean_Reading_Score']:<10.2f} {row['mean_Writing_Score']:<10.2f}")
    print(f"{'25% (Q1)':<15} {row['25%_Math_Score']:<10.2f} {row['25%_Reading_Score']:<10.2f} {row['25%_Writing_Score']:<10.2f}")
    print(f"{'50% (Median)':<15} {row['50%_Math_Score']:<10.2f} {row['50%_Reading_Score']:<10.2f} {row['50%_Writing_Score']:<10.2f}")
    print(f"{'75% (Q3)':<15} {row['75%_Math_Score']:<10.2f} {row['75%_Reading_Score']:<10.2f} {row['75%_Writing_Score']:<10.2f}")
    print(f"{'Min':<15} {row['min_Math_Score']:<10.2f} {row['min_Reading_Score']:<10.2f} {row['min_Writing_Score']:<10.2f}")
    print(f"{'Max':<15} {row['max_Math_Score']:<10.2f} {row['max_Reading_Score']:<10.2f} {row['max_Writing_Score']:<10.2f}")
    print(f"{'Std':<15} {row['std_Math_Score']:<10.2f} {row['std_Reading_Score']:<10.2f} {row['std_Writing_Score']:<10.2f}")
    print(f"{'Count':<15} {row['count_Math_Score']:<10.0f} {row['count_Reading_Score']:<10.0f} {row['count_Writing_Score']:<10.0f}")
    print("=" * 70)
```

```
Preparation-Based Summary Statistics:
======================================================================
Preparation Course: completed
======================================================================
Statistic       Math       Reading    Writing
----------------------------------------------------------------------
Mean            70.33      74.73      75.81
25% (Q1)        60.75      66.00      67.00
50% (Median)    70.00      75.00      75.00
75% (Q3)        81.00      84.00      86.00
Min             27.00      38.00      41.00
Max             100.00     100.00     100.00
Std             14.69      13.07      13.43
Count           344        344        344
======================================================================
Preparation Course: none
======================================================================
Statistic       Math       Reading    Writing
----------------------------------------------------------------------
Mean            66.49      68.10      65.64
25% (Q1)        57.00      58.00      56.00
50% (Median)    67.00      68.50      66.00
75% (Q3)        78.00      78.00      76.00
Min             15.00      25.00      15.00
Max             100.00     100.00     100.00
Std             15.38      14.11      14.64
Count           656        656        656
======================================================================
```

Students Who Completed the Preparation Course

- **Mean Scores**: Students who completed the course scored higher on average:
  - **Math**: 70.33
  - **Reading**: 74.73
  - **Writing**: 75.81

- **Score Distribution**:
  - The 25th percentile (Q1) scores in all subjects are significantly higher than those for students who did not complete the course.
  - The 75th percentile (Q3) scores also reflect stronger performance, especially in reading and writing.
- **Variability**:
  - Standard deviations (Std) are slightly lower compared to those who did not complete the course, suggesting more consistent performance.

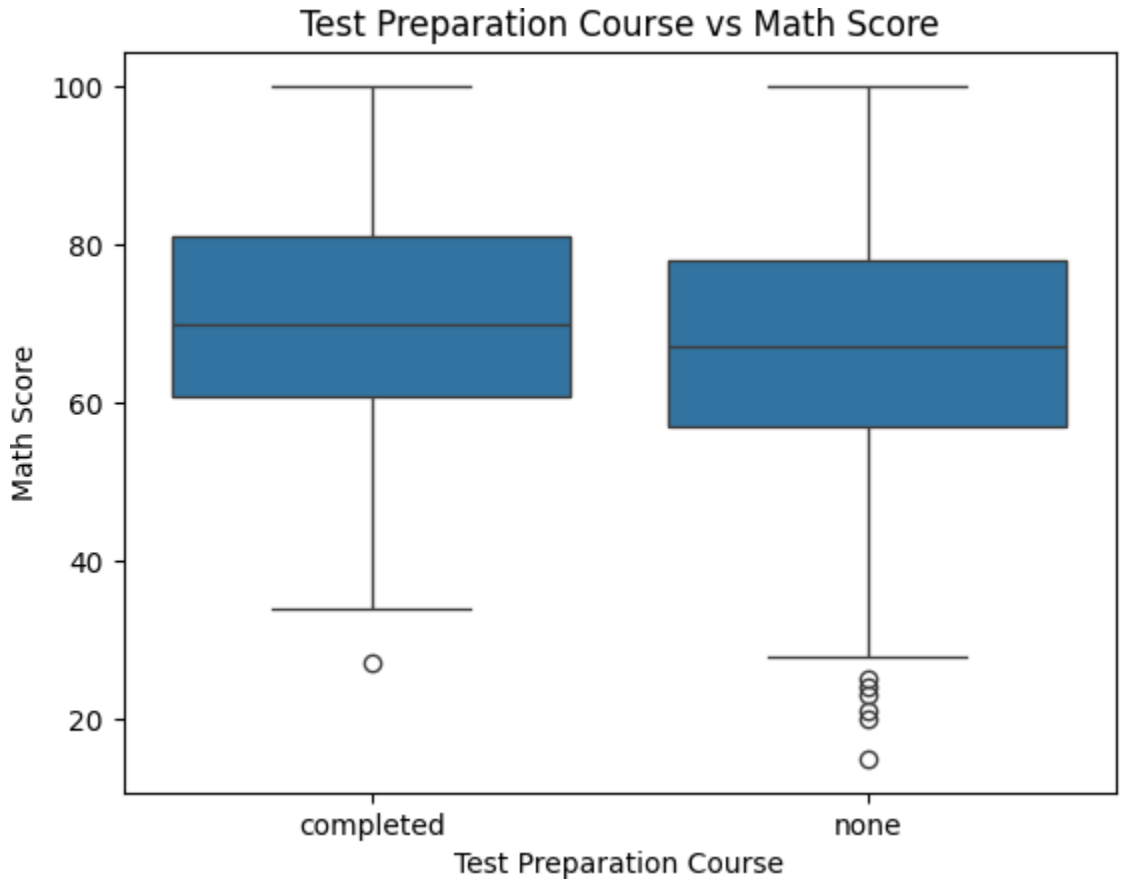
Students Who Did Not Complete the Preparation Course

- **Mean Scores**:
  - **Math**: 66.49
  - **Reading**: 68.10
  - **Writing**: 65.64
- **Score Distribution**:
  - Lower median and 25th percentile (Q1) scores across all subjects compared to the group that completed the course.
  - A wider range in scores, particularly in writing, with a higher presence of lower scores.
- **Variability**:
  - Standard deviations are slightly higher, indicating less consistency in performance among this group.

Analysis of the Box Plot: Test Preparation Course vs Math Score

his box plot visualizes the relationship between completing a test preparation course and math scores. Here's the analysis:

```
import seaborn as sns
import matplotlib.pyplot as plt

# Boxplot: Test Preparation Course vs Math Score
sns.boxplot(x='Preparation_Course', y='Math_Score', data=data)
plt.title('Test Preparation Course vs Math Score')
plt.xlabel('Test Preparation Course')
plt.ylabel('Math Score')
plt.show()
```



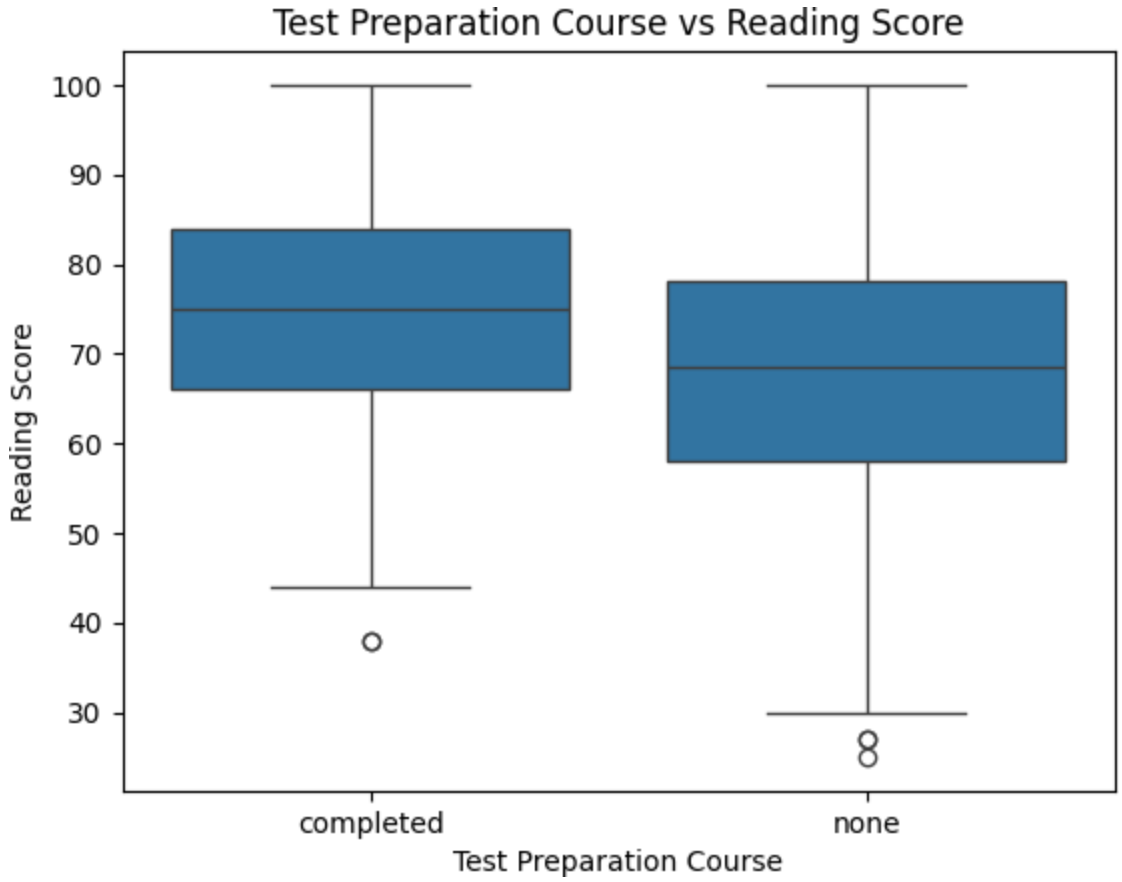
Observations:

- **Median Scores**:
  - Students who completed the preparation course have a higher median math score than those who did not. This indicates the effectiveness of structured preparation in improving performance.
- **Score Distribution**:
  - **Completed**: Scores are concentrated in the higher range, with fewer low scores and minimal outliers.
  - **Not Completed**: Scores show greater variability, with more low outliers pulling the lower whisker down.
- **Outliers**:
  - Students who did not complete the course exhibit several low outliers, suggesting a subset of this group struggled significantly in math.

Analysis of the Box Plot: Test Preparation Course vs Reading Score

This box plot illustrates the relationship between completing a test preparation course and reading scores.

```
# Boxplot: Test Preparation Course vs Reading Score
sns.boxplot(x='Preparation_Course', y='Reading_Score', data=data)
plt.title('Test Preparation Course vs Reading Score')
plt.xlabel('Test Preparation Course')
plt.ylabel('Reading Score')
plt.show()
```



Observations:

- **Median Scores**:
  - Students who completed the preparation course have a higher median reading score compared to those who did not, indicating the positive impact of preparation on reading performance.
- **Score Distribution**:
  - **Completed**: The scores are tightly clustered in the higher range, showing consistent and strong performance.

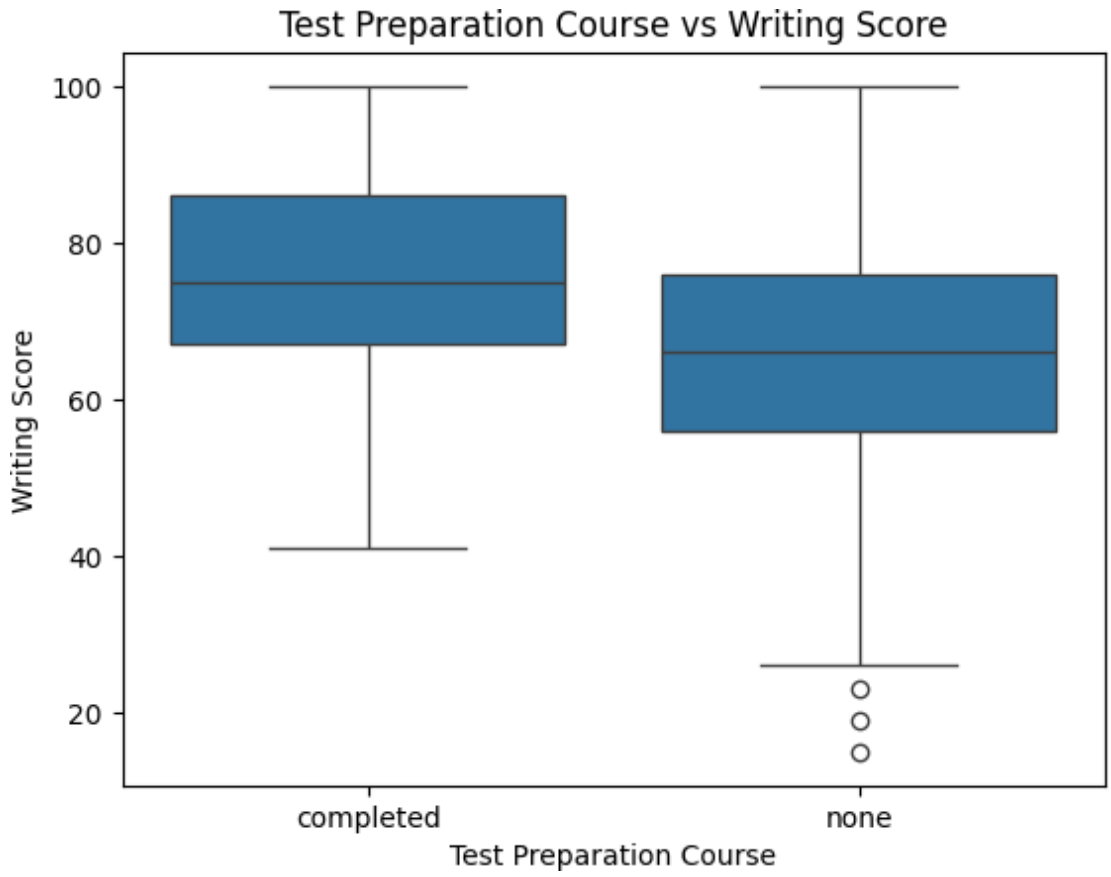
- **Not Completed**: Scores are more dispersed with a broader range, and lower scores are more prevalent.

- **Outliers**:
  - Students who did not complete the preparation course have more low outliers, suggesting a subset struggled significantly in reading.

Analysis of the Box Plot: Test Preparation Course vs Writing Score

his box plot visualizes the relationship between completing a test preparation course and writing scores.

```
# Boxplot: Test Preparation Course vs Writing Score
sns.boxplot(x='Preparation_Course', y='Writing_Score', data=data)
plt.title('Test Preparation Course vs Writing Score')
plt.xlabel('Test Preparation Course')
plt.ylabel('Writing Score')
plt.show()
```



Observations:

- **Median Scores**:
  - Students who completed the test preparation course have a higher median writing score compared to those who did not. This highlights the effectiveness of preparation in enhancing writing skills.
- **Score Distribution**:
  - **Completed**: Scores are concentrated in the higher range, reflecting better overall performance with fewer low scores.
  - **Not Completed**: Scores show greater variability, with a significant portion of students scoring lower.
- **Outliers**:
  - Students who did not complete the preparation course exhibit several low outliers, indicating notable struggles in writing.

Average Scores by Test Preparation Course Completion

The table provides the mean scores for math, reading, and writing based on test preparation course completion

```
# Calculate mean scores for each test preparation group
mean_scores_by_prep = data.groupby('Preparation_Course')[['Math_Score', 'Reading_Score', 'Writing_Score']].mean()

# Display the results
print(mean_scores_by_prep)
```

```
                    Math_Score  Reading_Score  Writing_Score
Preparation_Course
completed            70.334302      74.726744      75.808140
none                 66.486280      68.103659      65.643293
```

Completed Test Preparation Course:

- **Math Score**: Average score is **70.33**, indicating better performance compared to those who did not complete the course.
- **Reading Score**: Average score is **74.73**, reflecting strong performance in reading.
- **Writing Score**: Average score is **75.81**, the highest among the three subjects, suggesting a significant benefit from the course in literacy skills.

Did Not Complete Test Preparation Course:

- **Math Score**: Average score is **66.49**, lower than those who completed the course.
- **Reading Score**: Average score is **68.10**, showing a noticeable gap compared to the completed group.
- **Writing Score**: Average score is **65.64**, the lowest among the three subjects, highlighting the need for additional support in writing for this group.

Overall

Completing the test preparation course leads to better performance across all subjects, with the most significant improvement observed in writing scores.

Gender Statistics and Analysis

```python
# Group scores by gender and calculate summary statistics
gender_analysis = data.groupby('Gender')[['Math_Score', 'Reading_Score', 'Writing_Score']].describe()

# Simplify multi-index column names for better readability
gender_analysis.columns = [f"{stat}_{score}" for score, stat in gender_analysis.columns]

# Reset index for cleaner formatting
gender_analysis.reset_index(inplace=True)

# Format and print the output
print("Gender-Based Summary Statistics:")
print("=" * 70)
for _, row in gender_analysis.iterrows():
    print(f"Gender: {row['Gender']}")
    print("=" * 70)
    print(f"{'Statistic':<15} {'Math':<10} {'Reading':<10} {'Writing':<10}")
    print("-" * 70)
    print(f"{'Mean':<15} {row['mean_Math_Score']:<10.2f} {row['mean_Reading_Score']:<10.2f} {row['mean_Writing_Score']:<10.2f}")
    print(f"{'25% (Q1)':<15} {row['25%_Math_Score']:<10.2f} {row['25%_Reading_Score']:<10.2f} {row['25%_Writing_Score']:<10.2f}")
    print(f"{'50% (Median)':<15} {row['50%_Math_Score']:<10.2f} {row['50%_Reading_Score']:<10.2f} {row['50%_Writing_Score']:<10.2f}")
    print(f"{'75% (Q3)':<15} {row['75%_Math_Score']:<10.2f} {row['75%_Reading_Score']:<10.2f} {row['75%_Writing_Score']:<10.2f}")
    print(f"{'Min':<15} {row['min_Math_Score']:<10.2f} {row['min_Reading_Score']:<10.2f} {row['min_Writing_Score']:<10.2f}")
    print(f"{'Max':<15} {row['max_Math_Score']:<10.2f} {row['max_Reading_Score']:<10.2f} {row['max_Writing_Score']:<10.2f}")
    print(f"{'Std':<15} {row['std_Math_Score']:<10.2f} {row['std_Reading_Score']:<10.2f} {row['std_Writing_Score']:<10.2f}")
    print(f"{'Count':<15} {row['count_Math_Score']:<10.0f} {row['count_Reading_Score']:<10.0f} {row['count_Writing_Score']:<10.0f}")
    print("=" * 70)
```

```
Gender-Based Summary Statistics:
======================================================================
Gender: female
======================================================================
Statistic       Math       Reading    Writing
----------------------------------------------------------------------
Mean            64.77      73.47      73.44
25% (Q1)        55.00      65.00      64.00
50% (Median)    65.00      74.00      74.00
75% (Q3)        75.00      84.00      84.00
Min             15.00      30.00      26.00
Max             98.00      100.00     100.00
Std             15.08      14.09      14.57
Count           492        492        492
======================================================================
Gender: male
======================================================================
Statistic       Math       Reading    Writing
----------------------------------------------------------------------
Mean            70.75      67.39      64.98
25% (Q1)        60.00      58.00      55.00
50% (Median)    72.00      67.00      65.50
75% (Q3)        82.00      77.00      75.00
Min             20.00      25.00      15.00
Max             100.00     100.00     100.00
Std             14.85      13.48      14.29
Count           508        508        508
======================================================================
```

This table provides insights into academic performance across math, reading, and writing scores, categorized by gender.

Female Students:

- **Mean Scores**:
  - **Math**: 64.77
  - **Reading**: 73.47
  - **Writing**: 73.44
  - Female students excel in reading and writing compared to male students.
- **Score Distribution**:
  - The **25th percentile (Q1)** and **75th percentile (Q3)** in reading and writing are higher than in math, indicating stronger performance in literacy-related subjects.
  - The median scores (50th percentile) in reading and writing (74) are significantly higher than in math (65).
- **Variability**:
  - Standard deviations are slightly higher for writing, reflecting moderate variability in performance among female students.

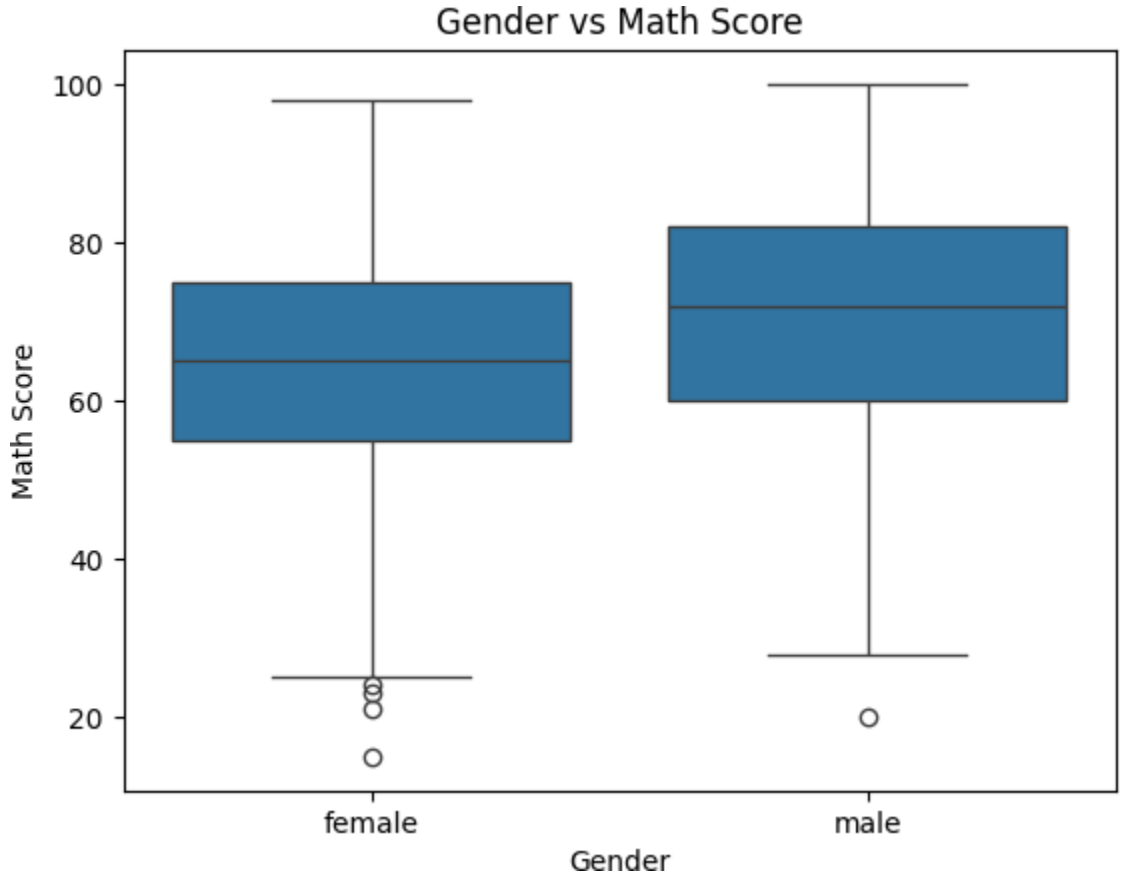
Male Students:

- **Mean Scores**:
  - **Math**: 70.75
  - **Reading**: 67.39
  - **Writing**: 64.98
  - Male students outperform female students in math but lag in reading and writing.
- **Score Distribution**:
  - Higher **25th percentile (Q1)** and **75th percentile (Q3)** scores in math reflect stronger overall performance in numerical skills.
  - The median score for math (72) is higher than for reading (67) and writing (65.5).
- **Variability**:
  - Standard deviations are consistent across subjects, suggesting uniform performance among male students.

Analysis of the Box Plot: Gender vs Math Score

This box plot visualizes the relationship between gender and math scores. Here's a detailed analysis:

```python
import seaborn as sns
import matplotlib.pyplot as plt

# Boxplot: Gender vs Math Score
sns.boxplot(x='Gender', y='Math_Score', data=data)
plt.title('Gender vs Math Score')
plt.xlabel('Gender')
plt.ylabel('Math Score')
plt.show()
```



- **Median Scores**:
  - Male students have a higher median math score compared to female students. This highlights stronger numerical performance among male students.

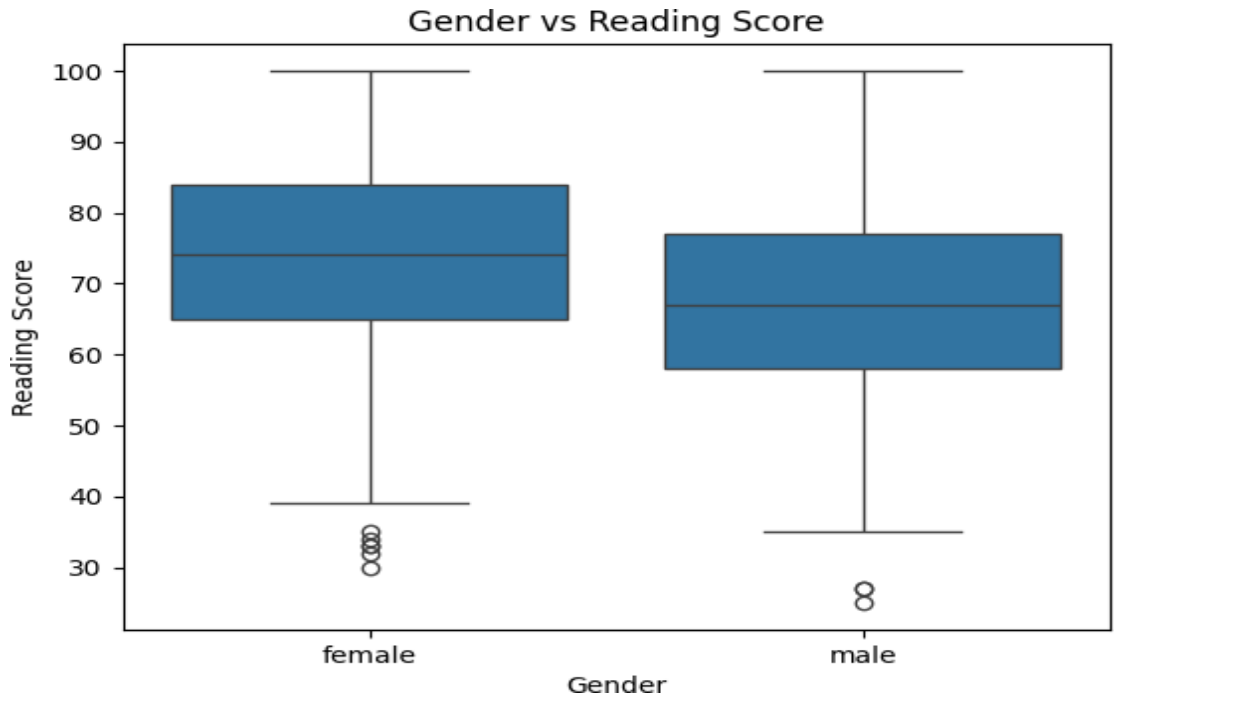
- **Score Range**:
  - **Males**: The scores are distributed over a broader range, with higher scores concentrated in the upper quartile.
  - **Females**: Scores are slightly more condensed, with lower quartile values pulling down the overall range.

- **Outliers**:
  - Both groups have outliers, but female students exhibit more low outliers compared to male students, suggesting some females struggle more with math.

Analysis of the Box Plot: Gender vs Reading Score

This box plot illustrates the relationship between gender and reading scores. Here's the detailed analysis:

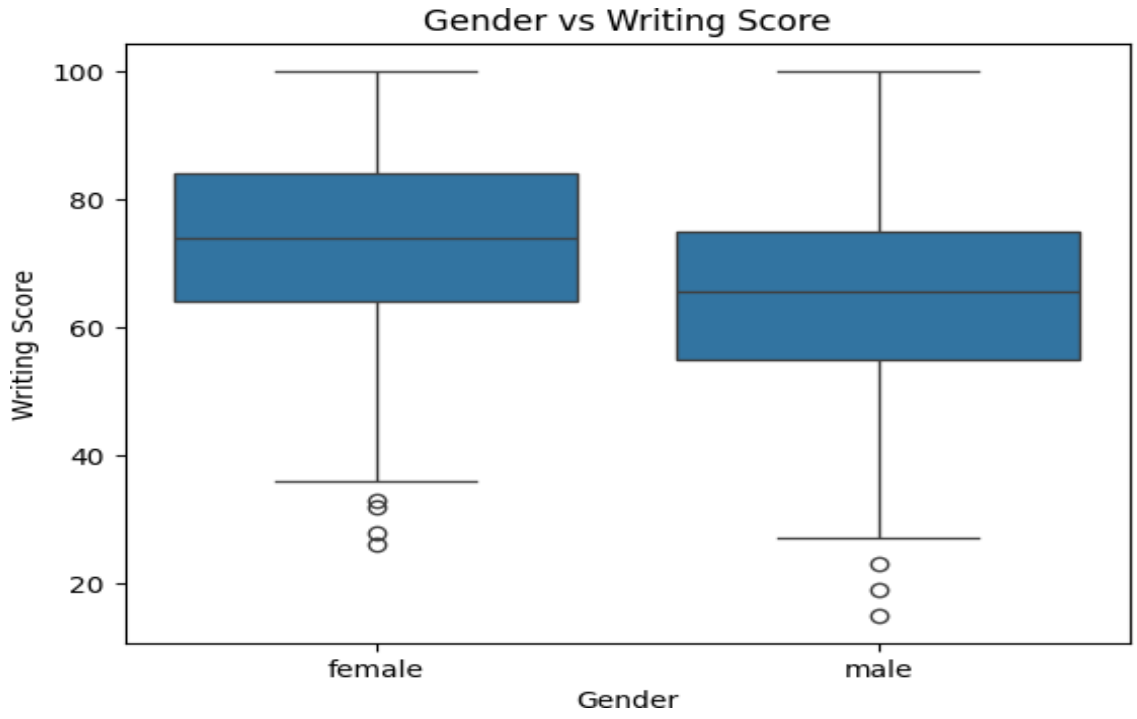
```
# Boxplot: Gender vs Reading Score
sns.boxplot(x='Gender', y='Reading_Score', data=data)
plt.title('Gender vs Reading Score')
plt.xlabel('Gender')
plt.ylabel('Reading Score')
plt.show()
```



- **Median Scores**:
  - Female students have a higher median reading score compared to male students, indicating stronger literacy skills among females.
- **Score Range**:
  - **Females**: Scores are more concentrated in the upper range, with fewer students scoring below the 25th percentile (Q1).
  - **Males**: Scores are more evenly distributed but are generally lower compared to females.
- **Outliers**:
  - Both genders show outliers, but female students exhibit a higher number of low outliers, suggesting a subset of females facing challenges in reading.

Analysis of the Box Plot: Gender vs Writing Score

```
# Boxplot: Gender vs Writing Score
sns.boxplot(x='Gender', y='Writing_Score', data=data)
plt.title('Gender vs Writing Score')
plt.xlabel('Gender')
plt.ylabel('Writing Score')
plt.show()
```



Average Scores by Gender

the mean scores for math, reading, and writing, grouped by gender. Here's the analysis:

```
# Calculate mean scores for each gender
mean_scores_by_gender = data.groupby('Gender')[['Math_Score', 'Reading_Score', 'Writing_Score']].mean()

# Display the results
print(mean_scores_by_gender)
```

```
        Math_Score  Reading_Score  Writing_Score
Gender
female    64.77439      73.473577      73.439024
male      70.75000      67.387795      64.976378
```

Female Students:

- **Math Score**: Average score is **64.77**, lower compared to males.
- **Reading Score**: Average score is **73.47**, higher than males, reflecting stronger literacy skills.
- **Writing Score**: Average score is **73.44**, significantly higher than males, showcasing female students' strength in writing.

Male Students:

- **Math Score**: Average score is **70.75**, outperforming females in math.
- **Reading Score**: Average score is **67.39**, lower than females, indicating challenges in literacy.
- **Writing Score**: Average score is **64.98**, lower than females, further highlighting the need for support in literacy-based skills.

Overall

- The data suggests that male students tend to excel in math, while female students dominate in reading and writing. These patterns reflect differing strengths and areas of focus for each gender.
- Variability in scores and outliers indicate opportunities for targeted interventions to support struggling students.

Race/Ethnicity-Based Summary Statistics Analysis

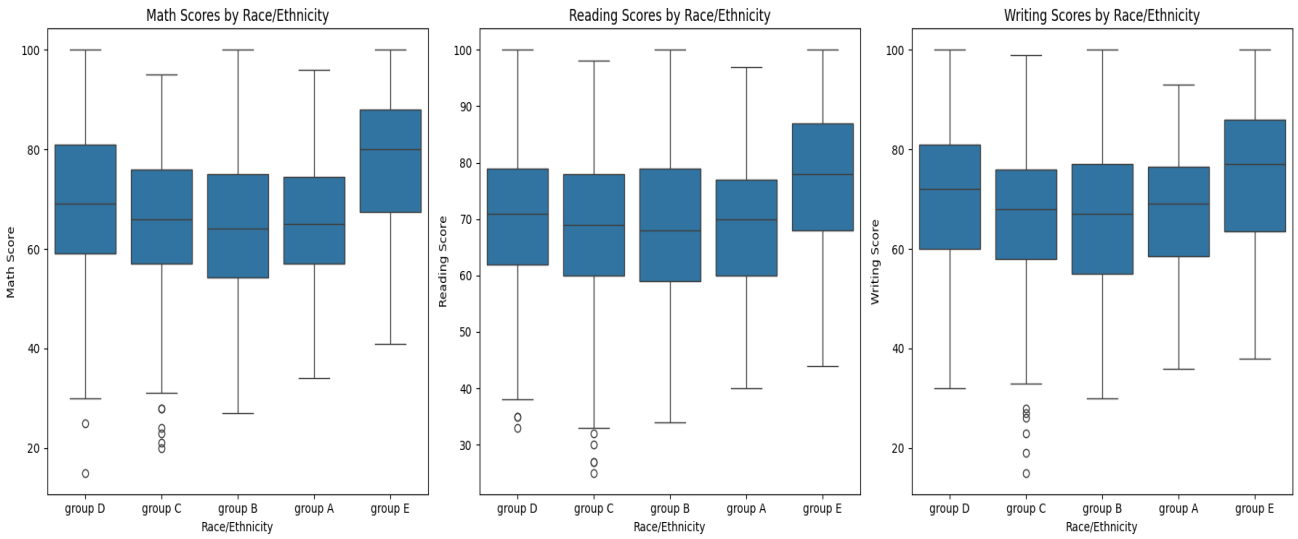
```python
# Group scores by Race/Ethnicity and calculate summary statistics
race_analysis = data.groupby('Race_Ethnicity')[['Math_Score', 'Reading_Score', 'Writing_Score']].describe()

# Simplify multi-index column names for better readability
race_analysis.columns = [f"{stat}_{score}" for score, stat in race_analysis.columns]

# Transpose the table to make groups columns and statistics rows
race_transposed = race_analysis.T

# Reset index and rename columns
race_transposed.reset_index(inplace=True)
race_transposed.rename(columns={"index": "Statistic"}, inplace=True)

# Display the formatted output
print("Race/Ethnicity-Based Summary Statistics (Formatted):")
print("=" * 90)
print(race_transposed.to_string(index=False))
```

```
Race/Ethnicity-Based Summary Statistics (Formatted):
==========================================================================================
          Statistic   group A    group B    group C    group D    group E
   count_Math_Score 79.000000 198.000000 323.000000 257.000000 143.000000
    mean_Math_Score 65.696203  64.070707  65.510836  68.879377  77.426573
     std_Math_Score 12.480091  14.602866  14.585442  15.792510  13.911941
     min_Math_Score 34.000000  27.000000  20.000000  15.000000  41.000000
     25%_Math_Score 57.000000  54.250000  57.000000  59.000000  67.500000
     50%_Math_Score 65.000000  64.000000  66.000000  69.000000  80.000000
     75%_Math_Score 74.500000  75.000000  76.000000  81.000000  88.000000
     max_Math_Score 96.000000 100.000000  95.000000 100.000000 100.000000
count_Reading_Score 79.000000 198.000000 323.000000 257.000000 143.000000
 mean_Reading_Score 69.202532  68.530303  68.609907  70.929961  76.615385
  std_Reading_Score 12.688961  14.160307  13.697582  14.321195  13.636076
  min_Reading_Score 40.000000  34.000000  25.000000  33.000000  44.000000
  25%_Reading_Score 60.000000  59.000000  60.000000  62.000000  68.000000
  50%_Reading_Score 70.000000  68.000000  69.000000  71.000000  78.000000
  75%_Reading_Score 77.000000  79.000000  78.000000  79.000000  87.000000
  max_Reading_Score 97.000000 100.000000  98.000000 100.000000 100.000000
count_Writing_Score 79.000000 198.000000 323.000000 257.000000 143.000000
 mean_Writing_Score 67.848101  66.717172  66.804954  71.058366  75.034965
  std_Writing_Score 13.383005  15.700910  14.378935  14.948887  14.599061
  min_Writing_Score 36.000000  30.000000  15.000000  32.000000  38.000000
  25%_Writing_Score 58.500000  55.000000  58.000000  60.000000  63.500000
  50%_Writing_Score 69.000000  67.000000  68.000000  72.000000  77.000000
  75%_Writing_Score 76.500000  77.000000  76.000000  81.000000  86.000000
  max_Writing_Score 93.000000 100.000000  99.000000 100.000000 100.000000
```

Analysis of Box Plots: Academic Scores by Race/Ethnicity

These box plots illustrate the distribution of math, reading, and writing scores across five racial/ethnic groups (Groups A to E). Here's the detailed analysis:

```python
import matplotlib.pyplot as plt
import seaborn as sns

# Box plot for Math, Reading, and Writing scores across all groups
plt.figure(figsize=(18, 6))

# Box plot for Math Scores
plt.subplot(1, 3, 1)
sns.boxplot(data=data, x='Race_Ethnicity', y='Math_Score')
plt.title('Math Scores by Race/Ethnicity')
plt.xlabel('Race/Ethnicity')
plt.ylabel('Math Score')

# Box plot for Reading Scores
plt.subplot(1, 3, 2)
sns.boxplot(data=data, x='Race_Ethnicity', y='Reading_Score')
plt.title('Reading Scores by Race/Ethnicity')
plt.xlabel('Race/Ethnicity')
plt.ylabel('Reading Score')

# Box plot for Writing Scores
plt.subplot(1, 3, 3)
sns.boxplot(data=data, x='Race_Ethnicity', y='Writing_Score')
plt.title('Writing Scores by Race/Ethnicity')
plt.xlabel('Race/Ethnicity')
plt.ylabel('Writing Score')

# Display the plots
plt.tight_layout()
plt.show()
```



Math Scores:

- **Median Performance**:
  - Group E has the highest median math score, followed by Group D. Group A has the lowest median score, reflecting a significant gap.
- **Score Range**:
  - Group E has the most consistent performance, with fewer low scores.
  - Group A exhibits more variability, with scores spanning a broader range and several low outliers.
- **Outliers**:
  - Group C and Group A have notable outliers, indicating a subset of students facing challenges in math.

Reading Scores:

- **Median Performance**:
  - Group E again leads with the highest median reading score, followed by Group D. Group A and Group B have similar, lower medians.
- **Score Range**:
  - Groups D and E show better concentration of higher scores, while Groups A and B exhibit wider variability.
- **Outliers**:
  - Groups C and A have more low outliers in reading, indicating struggles in literacy skills for a subset of students.

Writing Scores:

- **Median Performance**:
  - Group E performs best in writing, with the highest median score, followed by Group D. Groups A, B, and C lag behind.
- **Score Range**:
  - Groups D and E have a narrower range, reflecting consistent performance.
  - Groups A, B, and C show wider variability, with several low outliers, particularly in Group C.
- **Outliers**:
  - Low outliers are more prevalent in Groups A and C, suggesting specific challenges in writing for some students.

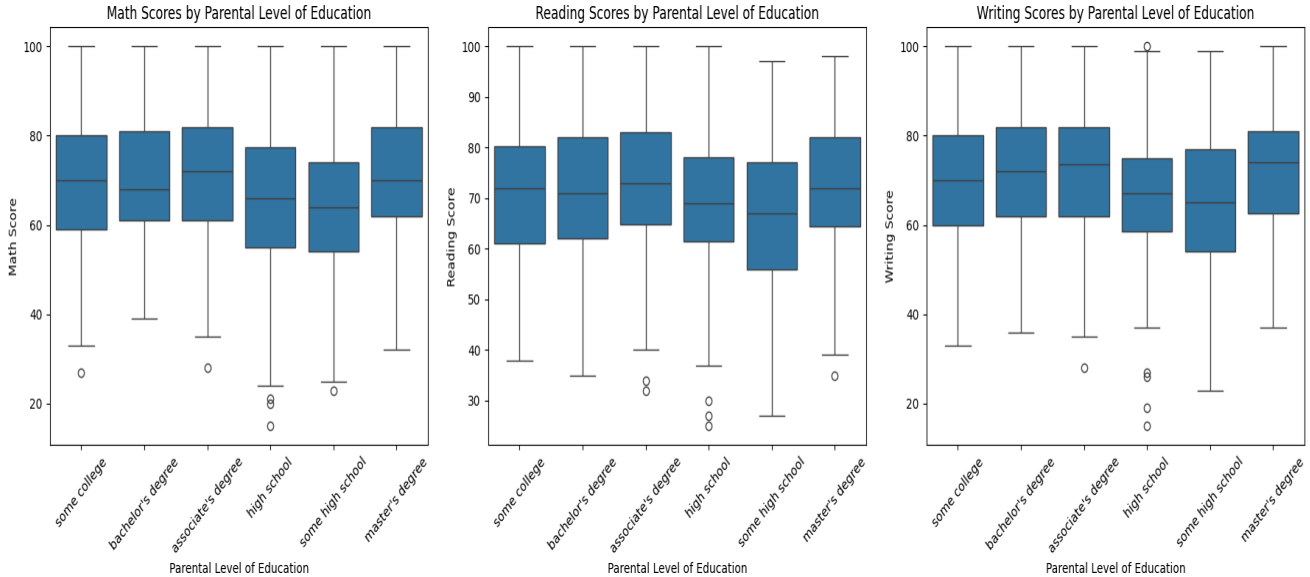
Analysis of Academic Performance Based on Parental Education Level

```python
# Group scores by Parental Level of Education and calculate summary statistics
parent_analysis = data.groupby('Parental_Education')[['Math_Score', 'Reading_Score', 'Writing_Score']].describe()

# Simplify multi-index column names for better readability
parent_analysis.columns = [f"{stat}_{score}" for score, stat in parent_analysis.columns]

# Transpose the table to make groups columns and statistics rows
parent_transposed = parent_analysis.T

# Reset index and rename columns
parent_transposed.reset_index(inplace=True)
parent_transposed.rename(columns={"index": "Statistic"}, inplace=True)

# Display the formatted output
print("Parental Education-Based Summary Statistics (Formatted):")
print("=" * 90)
print(parent_transposed.to_string(index=False))
```

```
Parental Education-Based Summary Statistics (Formatted):
==========================================================================================
          Statistic  associate's degree  bachelor's degree  high school  master's degree  some college  some high school
   count_Math_Score          204.000000         105.000000   215.000000        75.000000    224.000000       177.000000
    mean_Math_Score           70.348039          69.866667    65.381395        71.026667     68.642857        64.197740
     std_Math_Score           14.821813          14.262017    15.971459        14.189807     14.552738        15.739730
     min_Math_Score           28.000000          39.000000    15.000000        32.000000     27.000000        23.000000
     25%_Math_Score           61.000000          61.000000    55.000000        62.000000     59.000000        54.000000
     50%_Math_Score           72.000000          68.000000    66.000000        70.000000     70.000000        64.000000
     75%_Math_Score           82.000000          81.000000    77.500000        82.000000     80.000000        74.000000
     max_Math_Score          100.000000         100.000000   100.000000       100.000000    100.000000       100.000000
count_Reading_Score          204.000000         105.000000   215.000000        75.000000    224.000000       177.000000
 mean_Reading_Score           72.647059          71.819048    69.223256        71.973333     70.941964        66.943503
  std_Reading_Score           14.240473          14.238208    13.681846        13.689583     13.936153        14.187750
  min_Reading_Score           32.000000          35.000000    25.000000        35.000000     38.000000        27.000000
  25%_Reading_Score           64.750000          62.000000    61.500000        64.500000     61.000000        56.000000
  50%_Reading_Score           73.000000          71.000000    69.000000        72.000000     72.000000        67.000000
  75%_Reading_Score           83.000000          82.000000    78.000000        82.000000     80.250000        77.000000
  max_Reading_Score          100.000000         100.000000   100.000000        98.000000    100.000000        97.000000
count_Writing_Score          204.000000         105.000000   215.000000        75.000000    224.000000       177.000000
 mean_Writing_Score           72.039216          72.266667    66.772093        71.746667     69.473214        65.293785
  std_Writing_Score           15.208516          15.560840    14.454542        14.497058     14.267439        15.199193
  min_Writing_Score           28.000000          36.000000    15.000000        37.000000     33.000000        23.000000
  25%_Writing_Score           62.000000          62.000000    58.500000        62.500000     60.000000        54.000000
  50%_Writing_Score           73.500000          72.000000    67.000000        74.000000     70.000000        65.000000
  75%_Writing_Score           82.000000          82.000000    75.000000        81.000000     80.000000        77.000000
  max_Writing_Score          100.000000         100.000000   100.000000       100.000000    100.000000        99.000000
```

Analysis of Box Plots: Academic Scores by Parental Level of Education

```
import matplotlib.pyplot as plt
import seaborn as sns

# Box plot for Math, Reading, and Writing scores across all parental education levels
plt.figure(figsize=(18, 6))

# Box plot for Math Scores
plt.subplot(1, 3, 1)
sns.boxplot(data=data, x='Parental_Education', y='Math_Score')
plt.title('Math Scores by Parental Level of Education')
plt.xlabel('Parental Level of Education')
plt.ylabel('Math Score')
plt.xticks(rotation=45)

# Box plot for Reading Scores
plt.subplot(1, 3, 2)
sns.boxplot(data=data, x='Parental_Education', y='Reading_Score')
plt.title('Reading Scores by Parental Level of Education')
plt.xlabel('Parental Level of Education')
plt.ylabel('Reading Score')
plt.xticks(rotation=45)

# Box plot for Writing Scores
plt.subplot(1, 3, 3)
sns.boxplot(data=data, x='Parental_Education', y='Writing_Score')
plt.title('Writing Scores by Parental Level of Education')
plt.xlabel('Parental Level of Education')
plt.ylabel('Writing Score')
plt.xticks(rotation=45)

# Display the plots
plt.tight_layout()
plt.show()
```



Math Scores:

- **Highest Scores**:
  - Students whose parents have a **master's degree** (Mean: **71.03**) and **associate's degree** (Mean: **70.35**) achieved the highest math scores.
- **Lowest Scores**:
  - Students whose parents have **some high school** education (Mean: **64.20**) performed the lowest.
- **Score Distribution**:
  - Students with higher parental education levels tend to have narrower ranges of scores, reflecting consistency.

Reading Scores:

- **Highest Scores**:
  - Students with parents holding a **bachelor's degree** (Mean: **71.82**) and **master's degree** (Mean: **71.97**) scored the highest.
- **Lowest Scores**:
  - Students whose parents have **some high school** education (Mean: **66.94**) showed lower reading proficiency.
- **Variability**:
  - Standard deviations are generally consistent across groups, but slightly higher for students with parents with lower education levels.

Writing Scores:

- **Highest Scores**:
  - Students with parents having a **master's degree** (Mean: **71.75**) and **associate's degree** (Mean: **72.04**) performed the best.
- **Lowest Scores**:
  - Students whose parents have **some high school** education (Mean: **65.29**) had the lowest writing scores.
- **Variability**:
  - Students with parents with **some high school** education had a wider range of scores, reflecting variability in performance.

4. Features Engineering

Feature engineering involves preparing and transforming the dataset to make it suitable for analysis and machine learning models. Here, one-hot encoding is performed to handle categorical variables, converting them into numerical values while preserving the categorical information. Below is the processing of doing.

```
# Perform one-hot encoding for categorical variables
data_encoded = pd.get_dummies(data, columns=['Gender', 'Lunch', 'Preparation_Course'], drop_first=True)

# Display the first few rows of the encoded dataset
print(data_encoded.head())
```

```
  Race_Ethnicity  Parental_Education  Math_Score  Reading_Score  \
0        group D        some college          59             70
1        group C   bachelor's degree          57             69
2        group D  associate's degree          65             71
3        group D  associate's degree          67             71
4        group D  associate's degree          99             85

   Writing_Score  Gender_male  Lunch_standard  Preparation_Course_none
0             78        False            True                   False
1             77        False            True                   False
2             74        False            True                   False
3             76        False            True                   False
4             88         True            True                   False
```

```
from sklearn.preprocessing import StandardScaler

# Select numerical columns to standardize
columns_to_scale = ['Math_Score', 'Reading_Score', 'Writing_Score']

# Apply standardization
scaler = StandardScaler()
data_encoded[columns_to_scale] = scaler.fit_transform(data_encoded[columns_to_scale])

# Display the first few rows of the scaled dataset
print(data_encoded.head())
```


```
from sklearn.preprocessing import StandardScaler

# Select numerical columns to standardize
columns_to_scale = ['Math_Score', 'Reading_Score', 'Writing_Score']

# Apply standardization
scaler = StandardScaler()
data_encoded[columns_to_scale] = scaler.fit_transform(data_encoded[columns_to_scale])

# Display the first few rows of the scaled dataset
print(data_encoded.head())
```


```
  Race_Ethnicity  Parental_Education  Math_Score  Reading_Score  \
0        group D        some college   -0.577987      -0.027092
1        group C   bachelor's degree   -0.709198      -0.098012
2        group D  associate's degree   -0.184352       0.043829
3        group D  associate's degree   -0.053141       0.043829
4        group D  associate's degree    2.046243       1.036711

   Writing_Score  Gender_male  Lunch_standard  Preparation_Course_none
0       0.589943        False            True                   False
1       0.523358        False            True                   False
2       0.323603        False            True                   False
3       0.456773        False            True                   False
4       1.255793         True            True                   False
```

This code snippet outlines the process of preparing and splitting the data for training and testing a model to predict the **Writing,reading,math Score**.

```
# Features and target for Math Score
X_math = data_encoded.drop(columns=['Math_Score'])
y_math = data_encoded['Math_Score']

# Split data for Math Score
from sklearn.model_selection import train_test_split
X_train_math, X_test_math, y_train_math, y_test_math = train_test_split(X_math, y_math, test_size=0.3, random_state=42)

# Display shapes
print(f"Math - Features shape: {X_train_math.shape}, Target shape: {y_train_math.shape}")
```

[31]

```
Math - Features shape: (700, 7), Target shape: (700,)
```

```
# Features and target for Reading Score
X_reading = data_encoded.drop(columns=['Reading_Score'])
y_reading = data_encoded['Reading_Score']

# Split data for Reading Score
X_train_reading, X_test_reading, y_train_reading, y_test_reading = train_test_split(X_reading, y_reading, test_size=0.3, random_state=42

# Display shapes
print(f"Reading - Features shape: {X_train_reading.shape}, Target shape: {y_train_reading.shape}")
```

[32]

```
Reading - Features shape: (700, 7), Target shape: (700,)
```

```
# Features and target for Writing Score
X_writing = data_encoded.drop(columns=['Writing_Score'])
y_writing = data_encoded['Writing_Score']

# Split data for Writing Score
X_train_writing, X_test_writing, y_train_writing, y_test_writing = train_test_split(X_writing, y_writing, test_size=0.3, random_state=42)

# Display shapes
print(f"Writing - Features shape: {X_train_writing.shape}, Target shape: {y_train_writing.shape}")
```

```
Writing - Features shape: (700, 7), Target shape: (700,)
```

5. Model Building

The model building step involves training machine learning models to predict student performance in Math, Reading, and Writing based on the features created during the feature engineering process. The notebook implemented three types of models: Linear Regression, Random Forest Regression, and Support Vector Regression. These models were trained, tested, and evaluated using well-defined metrics to ensure robust predictions.

- **Linear Regression:**

Objective: Implements Linear Regression models to predict Math, Reading, and Writing scores.

```
# Print the column names to identify the exact name of the problematic column
print(data.columns)
```

```
Index(['Gender', 'Race_Ethnicity', 'Parental_Education', 'Lunch',
       'Preparation_Course', 'Math_Score', 'Reading_Score', 'Writing_Score'],
      dtype='object')
```

```
# One-hot encode all categorical columns
data_encoded = pd.get_dummies(data, columns=['Gender', 'Lunch', 'Preparation_Course'], drop_first=True)

# Check the first few rows of the encoded dataset
print(data_encoded.head())
```

```
  Race_Ethnicity  Parental_Education  Math_Score  Reading_Score  \
0        group D        some college          59             70
1        group C   bachelor's degree          57             69
2        group D  associate's degree          65             71
3        group D  associate's degree          67             71
4        group D  associate's degree          99             85

   Writing_Score  Gender_male  Lunch_standard  Preparation_Course_none
0             78        False            True                   False
1             77        False            True                   False
2             74        False            True                   False
3             76        False            True                   False
4             88         True            True                   False
```

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

```
print(y_train_math.head())
```

```
541   -0.840410
440    0.930945
482    0.406100
422    0.930945
778   -0.315564
Name: Math_Score, dtype: float64
```

```
print(y_train_reading.head())
```

```
541   -1.019974
440    0.469350
482   -0.239852
422    0.965791
778   -0.807214
Name: Reading_Score, dtype: float64
```

```
print(y_train_writing.head())
```

```
541   -1.008097
440    0.190433
482   -0.142492
422    0.789698
778   -1.207852
Name: Writing_Score, dtype: float64
```

```python
# Check for columns with non-numeric data
print(data_encoded.dtypes)
```

```
Race_Ethnicity             object
Parental_Education         object
Math_Score                  int64
Reading_Score               int64
Writing_Score               int64
Gender_male                  bool
Lunch_standard               bool
Preparation_Course_none      bool
dtype: object
```

```python
# One-hot encode all categorical columns
data_encoded = pd.get_dummies(data, drop_first=True)
```

```python
# Initialize models
model_math = LinearRegression()
model_reading = LinearRegression()
model_writing = LinearRegression()

# Train the models
model_math.fit(X_train_scaled, y_train_math)
model_reading.fit(X_train_scaled, y_train_reading)
model_writing.fit(X_train_scaled, y_train_writing)

# Predict on the test data
y_pred_math = model_math.predict(X_test_scaled)
y_pred_reading = model_reading.predict(X_test_scaled)
y_pred_writing = model_writing.predict(X_test_scaled)

# Evaluate the models
print("Math Score Prediction:")
print(f"R² Score: {r2_score(y_test_math, y_pred_math):.4f}")
print(f"Mean Squared Error: {mean_squared_error(y_test_math, y_pred_math):.4f}")

print("\nReading Score Prediction:")
print(f"R² Score: {r2_score(y_test_reading, y_pred_reading):.4f}")
print(f"Mean Squared Error: {mean_squared_error(y_test_reading, y_pred_reading):.4f}")

print("\nWriting Score Prediction:")
print(f"R² Score: {r2_score(y_test_writing, y_pred_writing):.4f}")
print(f"Mean Squared Error: {mean_squared_error(y_test_writing, y_pred_writing):.4f}")

# Example of predicting with new data
new_data = pd.DataFrame({
    'Gender_male': [1],  # Replace with the actual value for gender (1 for male, 0 for female)
    'Lunch_standard': [1],  # Replace with actual value for lunch type (1 for standard, 0 for free/reduced)
    'Preparation_Course_completed': [1],  # Replace with the actual value for preparation course (1 for completed, 0 for not completed)

    # Add other features you have after one-hot encoding
    'Race_Ethnicity_group_D': [0],  # Example for one-hot encoded feature, replace with actual values
    'Race_Ethnicity_group_C': [1]   # Replace with actual value for the race/ethnicity column (if applicable)
})
```

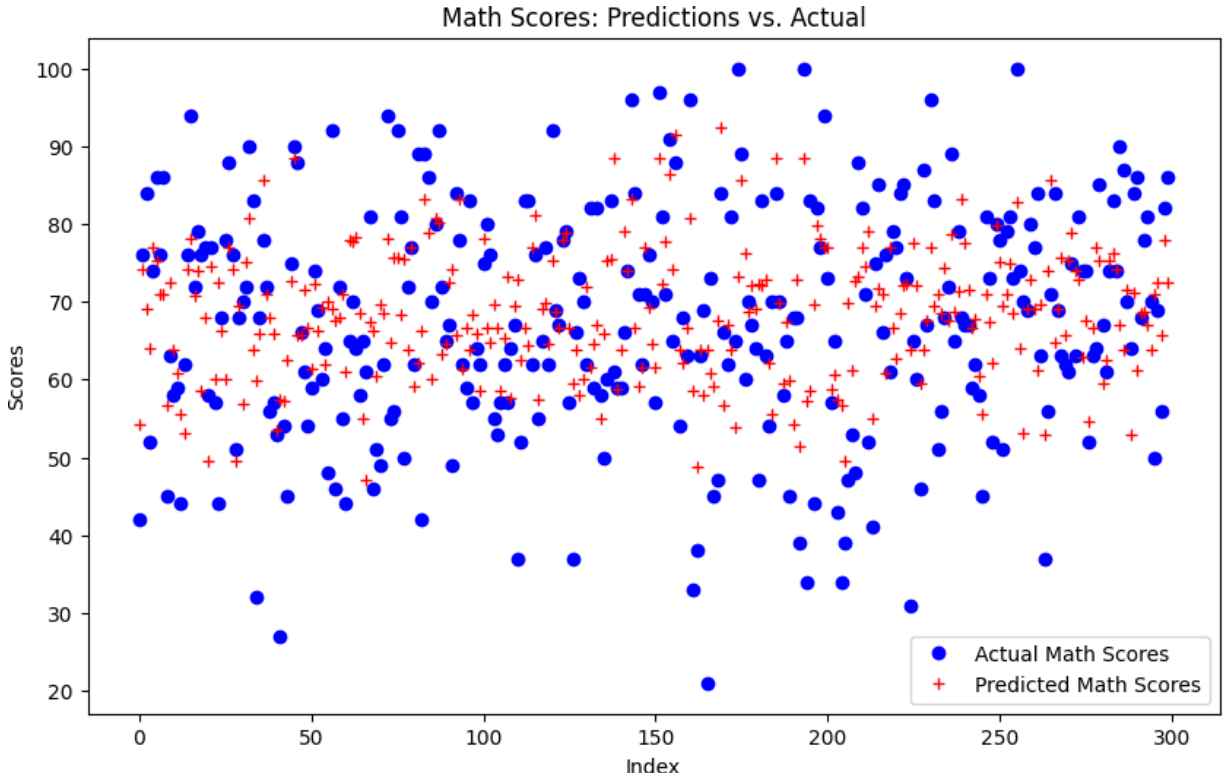
```python
# Ensure the new data has the same structure and columns as the training data
new_data = new_data.reindex(columns=X_train.columns, fill_value=0)

# Scale the new data using the fitted scaler
new_data_scaled = scaler.transform(new_data)

# Predict using the trained models
predicted_math = model_math.predict(new_data_scaled)
predicted_reading = model_reading.predict(new_data_scaled)
predicted_writing = model_writing.predict(new_data_scaled)

# Display predictions
print(f"\nPredicted Math Score: {predicted_math[0]:.2f}")
print(f"Predicted Reading Score: {predicted_reading[0]:.2f}")
print(f"Predicted Writing Score: {predicted_writing[0]:.2f}")
```

```
Math Score Prediction:
R² Score: 0.2529
Mean Squared Error: 166.5611

Reading Score Prediction:
R² Score: 0.1642
Mean Squared Error: 163.7934

Writing Score Prediction:
R² Score: 0.2487
Mean Squared Error: 168.4722

Predicted Math Score: 78.80
Predicted Reading Score: 75.44
Predicted Writing Score: 75.20
```

Analysis of the Plot

```python
import matplotlib.pyplot as plt
import numpy as np

# Plot Predictions vs. Actual Values as Dot Plots

# Math Score Dot Plot
plt.figure(figsize=(10, 6))
plt.plot(np.arange(len(y_test_math)), y_test_math, 'bo', label='Actual Math Scores')  # Blue dots for actual values
plt.plot(np.arange(len(y_pred_math)), y_pred_math, 'r+', label='Predicted Math Scores')  # Red plus for predicted values
plt.title('Math Scores: Predictions vs. Actual')
plt.xlabel('Index')
plt.ylabel('Scores')
plt.legend()
plt.show()

# Reading Score Dot Plot
plt.figure(figsize=(10, 6))
plt.plot(np.arange(len(y_test_reading)), y_test_reading, 'bo', label='Actual Reading Scores')  # Blue dots for actual values
plt.plot(np.arange(len(y_pred_reading)), y_pred_reading, 'r+', label='Predicted Reading Scores')  # Red plus for predicted values
plt.title('Reading Scores: Predictions vs. Actual')
plt.xlabel('Index')
plt.ylabel('Scores')
plt.legend()
plt.show()

# Writing Score Dot Plot
plt.figure(figsize=(10, 6))
plt.plot(np.arange(len(y_test_writing)), y_test_writing, 'bo', label='Actual Writing Scores')  # Blue dots for actual values
plt.plot(np.arange(len(y_pred_writing)), y_pred_writing, 'r+', label='Predicted Writing Scores')  # Red plus for predicted values
plt.title('Writing Scores: Predictions vs. Actual')
plt.xlabel('Index')
plt.ylabel('Scores')
plt.legend()
plt.show()
```

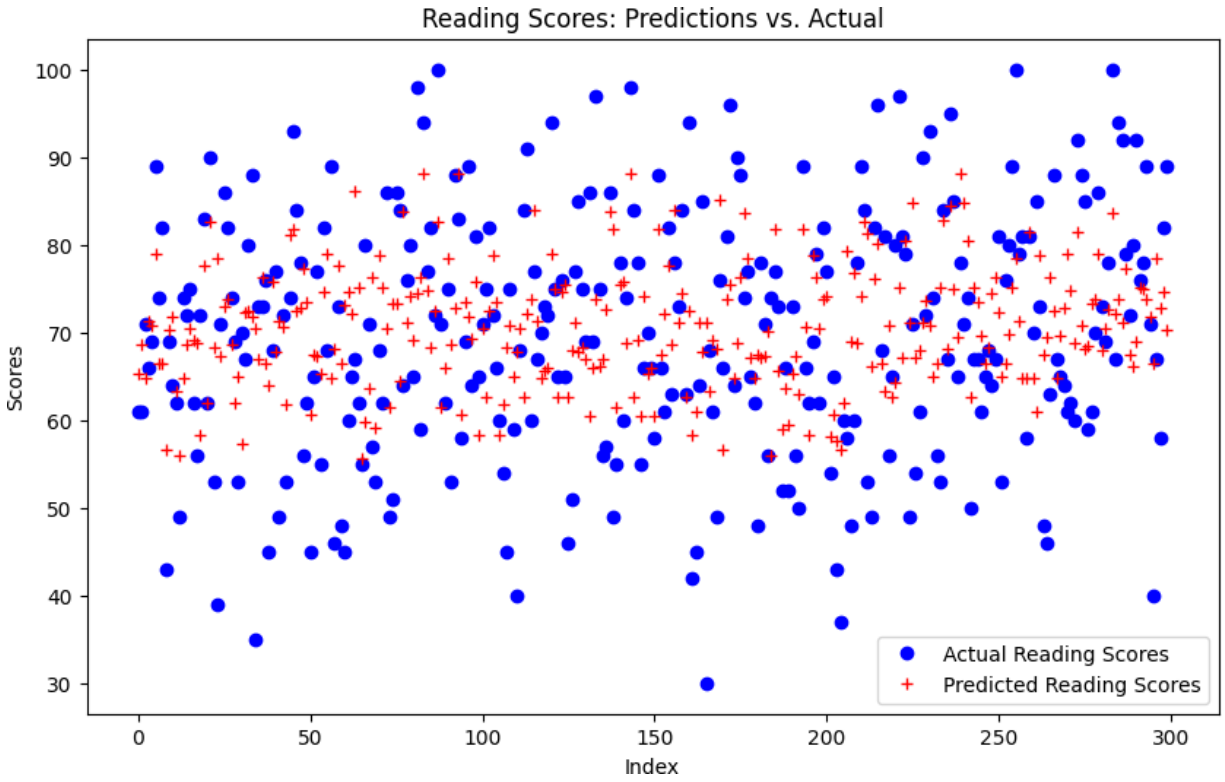
Analysis of the Plot: Predictions vs. Actual Math Scores



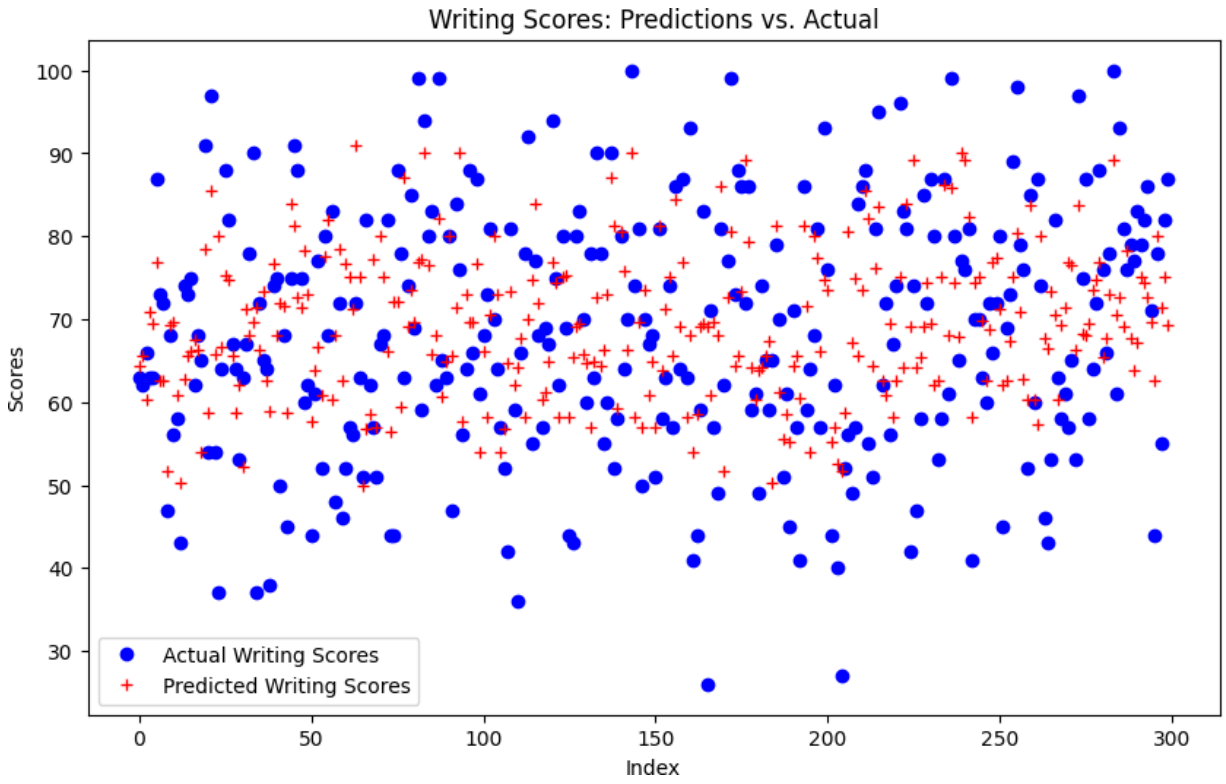
- **Scatter Distribution**:
  - The blue dots represent the **actual math scores**, while the red crosses represent the **predicted scores**.
  - The predicted scores (red) closely align with the actual scores (blue), indicating that the model performs reasonably well.

- **Prediction Spread**:
  - There is some variability in predictions, especially for mid-range scores, where the predicted values slightly deviate from actual ones.
  - Extreme values (both low and high scores) show a slight divergence between actual and predicted scores, which may indicate areas for model improvement.

- **Clustered Pattern**:
  - Most data points fall within the 50-80 score range, showing that the majority of students achieved moderate to high scores.



- **Alignment of Predictions**:
  - The red crosses representing predicted scores generally align with the blue dots of actual scores, indicating the model is capturing the trend effectively.
  - However, there is some noticeable spread, where predictions deviate from actual values in both directions.
- **Cluster Distribution**:
  - Most scores (both actual and predicted) fall within the range of **60 to 80**, showing that the majority of students achieved average to above-average reading scores.
  - Outliers in the lower and upper ranges show some divergence between actual and predicted scores, especially at extreme values.
- **Deviation**:
  - Predicted scores slightly underestimate or overestimate the actual scores in a few cases, which could reflect limitations in the model's ability to capture complex relationships.



- **Prediction Accuracy**:
  - The red crosses (predicted scores) align reasonably well with the blue dots (actual scores), demonstrating that the model captures the general trend in the data.
  - The deviations between actual and predicted scores are moderate, with some discrepancies noticeable across the range.
- **Distribution of Scores**:
  - Most of the scores, both actual and predicted, fall within the **60 to 80 range**, indicating that the majority of students achieved average to above-average writing scores.
  - Predictions are slightly less accurate at the lower end (below 50) and higher end (above 90).
- **Spread of Predictions**:
  - There is a consistent spread of predicted scores around the actual values, showing that the model performs well in capturing central trends but may struggle slightly with variability.

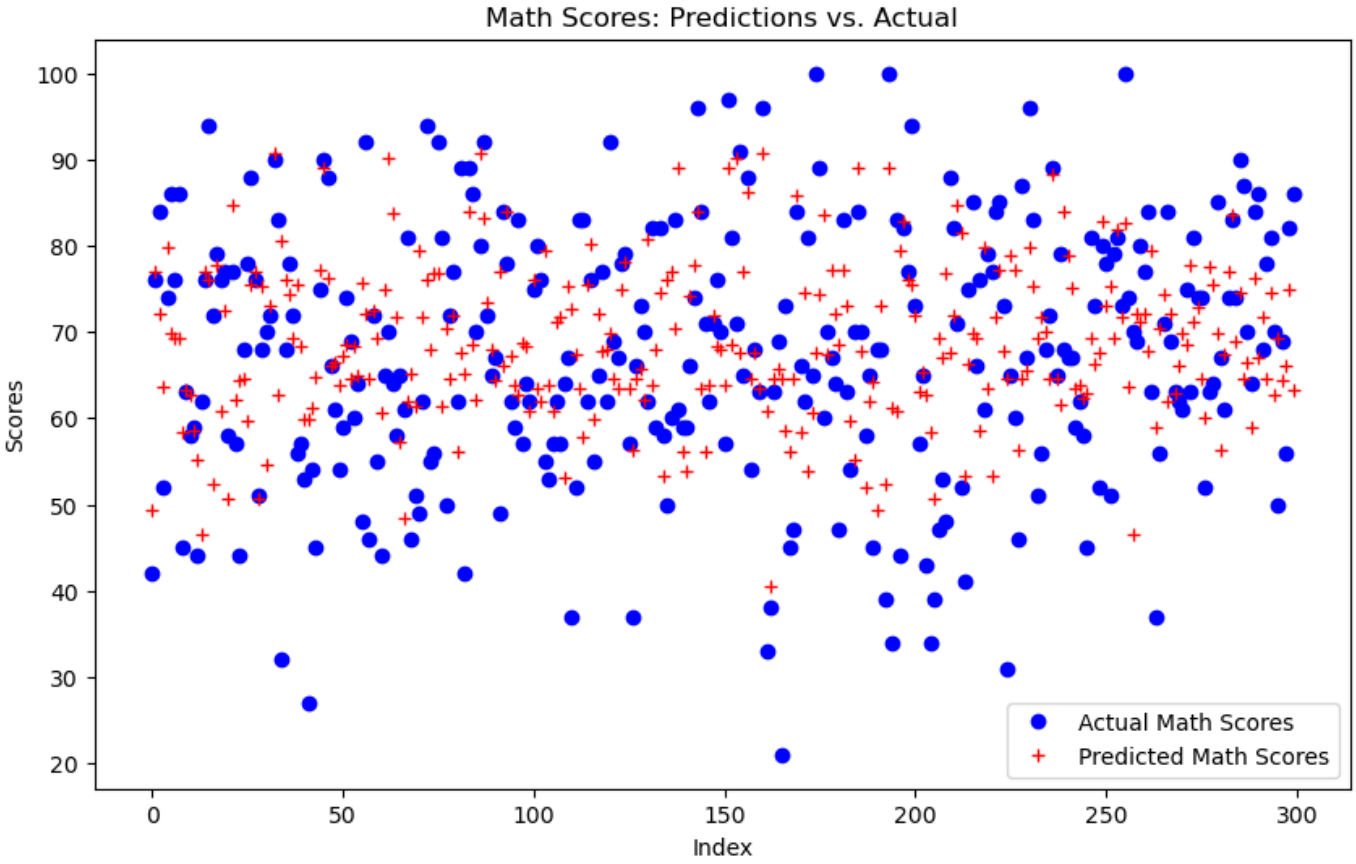
➢ **Random Forest Regression:**

Objective: Implements Random Forest models to predict Math, Reading, and Writing scores.

```python
# Import necessary libraries
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, r2_score
import pandas as pd

# Assuming `data_encoded` is your pre-processed DataFrame after one-hot encoding and scaling

# Define features and target for all subjects
X = data_encoded.drop(columns=['Math_Score', 'Reading_Score', 'Writing_Score'])
y_math = data_encoded['Math_Score']
y_reading = data_encoded['Reading_Score']
y_writing = data_encoded['Writing_Score']

# Split the data into training and testing sets for each subject
X_train, X_test, y_train_math, y_test_math = train_test_split(X, y_math, test_size=0.3, random_state=42)
_, _, y_train_reading, y_test_reading = train_test_split(X, y_reading, test_size=0.3, random_state=42)
_, _, y_train_writing, y_test_writing = train_test_split(X, y_writing, test_size=0.3, random_state=42)

# Initialize the scaler and scale the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)  # Scale based on training data
X_test_scaled = scaler.transform(X_test)  # Apply same transformation to test data

# Initialize Random Forest models
model_math = RandomForestRegressor(n_estimators=100, random_state=42)
model_reading = RandomForestRegressor(n_estimators=100, random_state=42)
model_writing = RandomForestRegressor(n_estimators=100, random_state=42)

# Train the models
model_math.fit(X_train_scaled, y_train_math)
model_reading.fit(X_train_scaled, y_train_reading)
model_writing.fit(X_train_scaled, y_train_writing)

# Predict on the test data
y_pred_math = model_math.predict(X_test_scaled)
y_pred_reading = model_reading.predict(X_test_scaled)
y_pred_writing = model_writing.predict(X_test_scaled)

# Evaluate the models
print("Math Score Prediction (Random Forest):")
print(f"R² Score: {r2_score(y_test_math, y_pred_math):.4f}")
print(f"Mean Squared Error: {mean_squared_error(y_test_math, y_pred_math):.4f}")

print("\nReading Score Prediction (Random Forest):")
print(f"R² Score: {r2_score(y_test_reading, y_pred_reading):.4f}")
print(f"Mean Squared Error: {mean_squared_error(y_test_reading, y_pred_reading):.4f}")

print("\nWriting Score Prediction (Random Forest):")
print(f"R² Score: {r2_score(y_test_writing, y_pred_writing):.4f}")
print(f"Mean Squared Error: {mean_squared_error(y_test_writing, y_pred_writing):.4f}")

# Example of predicting with new data
new_data = pd.DataFrame({
    'Gender_male': [1],  # Replace with the actual value for gender (1 for male, 0 for female)
    'Lunch_standard': [1],  # Replace with actual value for lunch type (1 for standard, 0 for free/reduced)
    'Preparation_Course_completed': [1],  # Replace with the actual value for preparation course (1 for completed, 0 for not completed)

    # Add other features you have after one-hot encoding
    'Race_Ethnicity_group_D': [0],  # Example for one-hot encoded feature, replace with actual values
    'Race_Ethnicity_group_C': [1]   # Replace with actual value for the race/ethnicity column (if applicable)
})

# Ensure the new data has the same structure and columns as the training data
new_data = new_data.reindex(columns=X_train.columns, fill_value=0)

# Scale the new data using the fitted scaler
new_data_scaled = scaler.transform(new_data)

# Predict using the trained Random Forest models
predicted_math = model_math.predict(new_data_scaled)
predicted_reading = model_reading.predict(new_data_scaled)
predicted_writing = model_writing.predict(new_data_scaled)

# Display predictions
print(f"\nPredicted Math Score (Random Forest): {predicted_math[0]:.2f}")
print(f"Predicted Reading Score (Random Forest): {predicted_reading[0]:.2f}")
print(f"Predicted Writing Score (Random Forest): {predicted_writing[0]:.2f}")
```

```
Math Score Prediction (Random Forest):
R² Score: 0.0975
Mean Squared Error: 201.2249

Reading Score Prediction (Random Forest):
R² Score: 0.0171
Mean Squared Error: 192.6228

Writing Score Prediction (Random Forest):
R² Score: 0.1139
Mean Squared Error: 198.6884

Predicted Math Score (Random Forest): 78.10
Predicted Reading Score (Random Forest): 73.06
Predicted Writing Score (Random Forest): 74.02
```

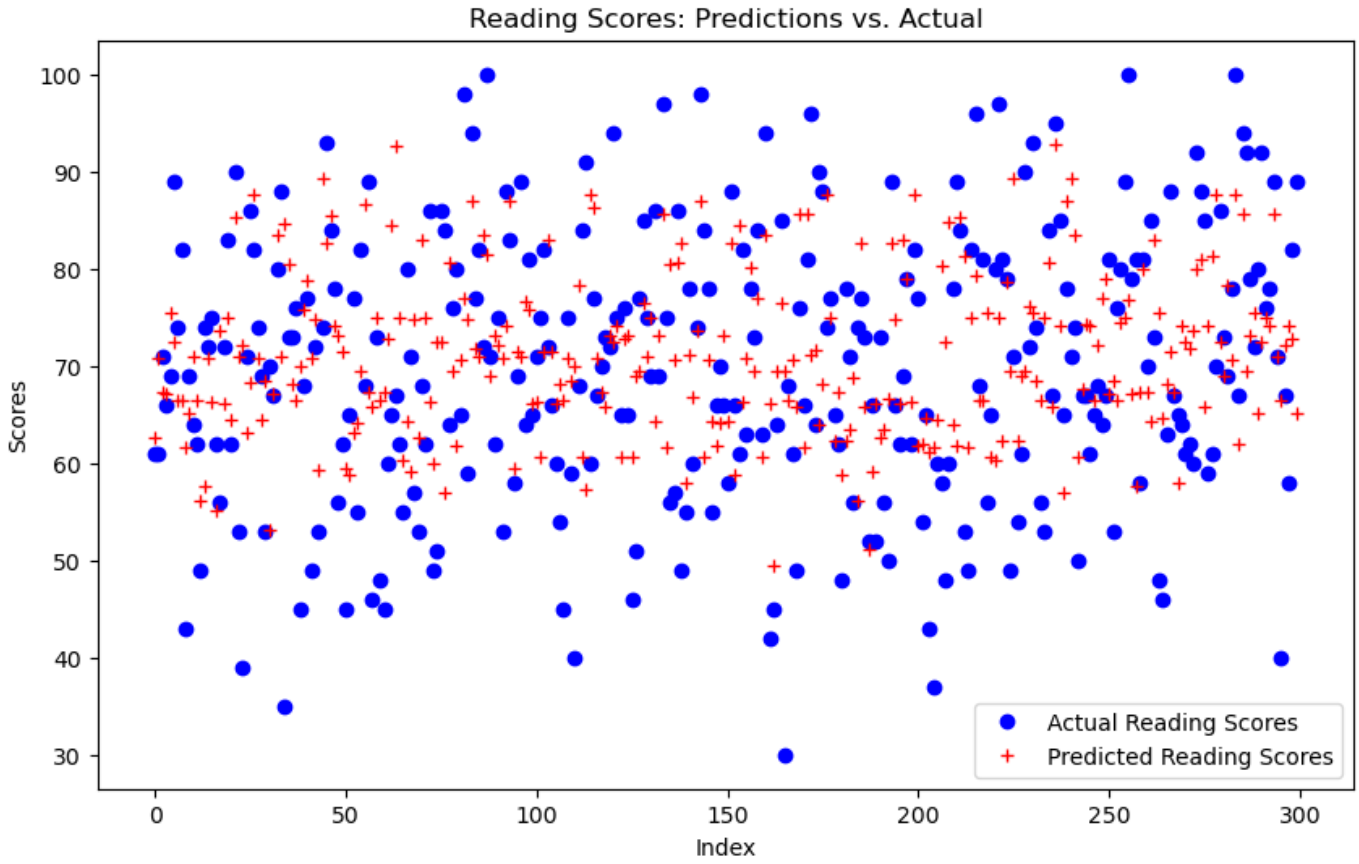
Analysis of the Plot: Predictions vs. Actual Math Scores:



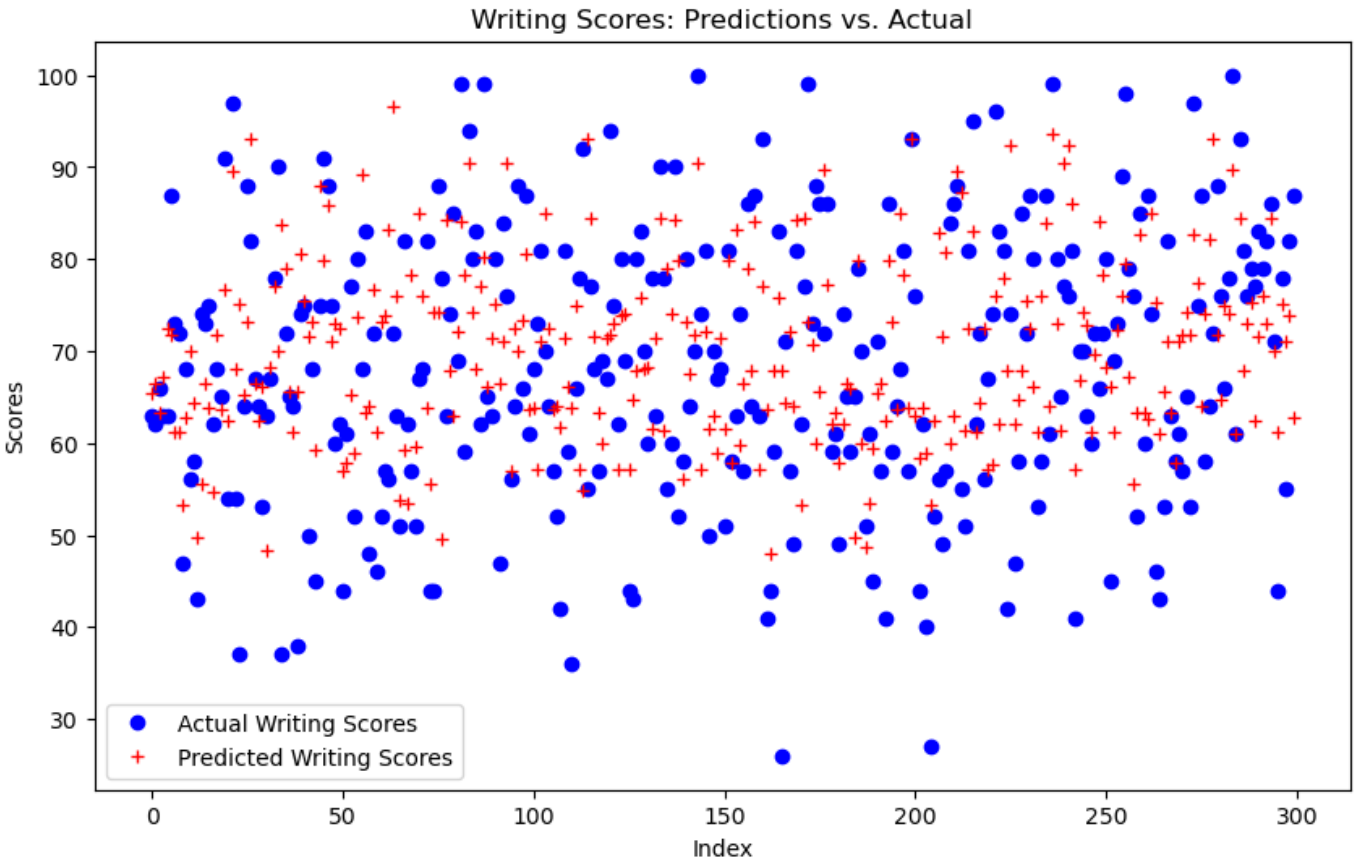
- **Scatter Distribution**:
  - The blue dots represent the **actual Math scores**, while the red crosses represent the **predicted scores**.
  - The predicted scores (red) are generally well-aligned with the actual scores (blue), indicating reasonable model performance.
- **Prediction Spread**:
  - Variability is observed in the mid-range scores (50–80), where the predicted values occasionally deviate from the actual ones.
  - At the extremes (very low and very high scores), the divergence between predicted and

actual values is more pronounced, suggesting potential areas for improvement.

- **Clustered Pattern**:
  - Most data points are concentrated in the 50–80 range, showing that the majority of students scored moderately to well in Math.



- **Scatter Distribution:**
  - The blue dots represent the **actual Reading scores**, while the red crosses represent the **predicted scores**.
  - The predicted scores closely align with the actual scores for most data points, indicating reasonable model performance.
- **Prediction Spread**:
  - There is moderate variability in the mid-range scores (approximately 60–80), with slight deviations between actual and predicted values.
  - At the extremes (both very low and very high scores), the model shows greater divergence between predicted and actual values, suggesting reduced accuracy.
- **Clustered Pattern**:
  - Most data points are concentrated in the score range of 60–80, showing that the majority of students achieved moderate to high Reading scores.



- **Scatter Distribution**:
  - The blue dots represent the **actual Writing scores**, while the red crosses represent the **predicted scores**.
  - The predicted scores (red crosses) closely follow the actual scores (blue dots) for most data points, showing that the model performs well overall.
- **Prediction Spread**:
  - Predictions align closely with actual values in the mid-range (approximately 60–80), but there is some variability and deviation.
  - For extreme scores (both low and high), the model shows larger gaps between actual and predicted values, suggesting reduced accuracy at the extremes.
- **Clustered Pattern**:
  - The majority of data points fall in the 60–80 range, indicating that most students performed moderately to well in Writing.

➢ **Support Vector Regressor (SVR)**

Objective: Implements Support Vector Regression models to predict Math, Reading, and Writing scores.

```python
# Import necessary libraries
from sklearn.svm import SVR
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, r2_score
import pandas as pd

# Assuming `data_encoded` is your pre-processed DataFrame after one-hot encoding

# Define features and target for all subjects
X = data_encoded.drop(columns=['Math_Score', 'Reading_Score', 'Writing_Score'])
y_math = data_encoded['Math_Score']
y_reading = data_encoded['Reading_Score']
y_writing = data_encoded['Writing_Score']

# Split the data into training and testing sets for each subject
X_train, X_test, y_train_math, y_test_math = train_test_split(X, y_math, test_size=0.3, random_state=42)
_, _, y_train_reading, y_test_reading = train_test_split(X, y_reading, test_size=0.3, random_state=42)
_, _, y_train_writing, y_test_writing = train_test_split(X, y_writing, test_size=0.3, random_state=42)

# Initialize the scaler and scale the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)  # Scale based on training data
X_test_scaled = scaler.transform(X_test)  # Apply same transformation to test data

# Initialize SVR models
model_math = SVR(kernel='rbf')  # Radial Basis Function kernel for non-linear regression
model_reading = SVR(kernel='rbf')
model_writing = SVR(kernel='rbf')

# Train the models
model_math.fit(X_train_scaled, y_train_math)
model_reading.fit(X_train_scaled, y_train_reading)
model_writing.fit(X_train_scaled, y_train_writing)

# Predict on the test data
y_pred_math = model_math.predict(X_test_scaled)
y_pred_reading = model_reading.predict(X_test_scaled)
y_pred_writing = model_writing.predict(X_test_scaled)
```

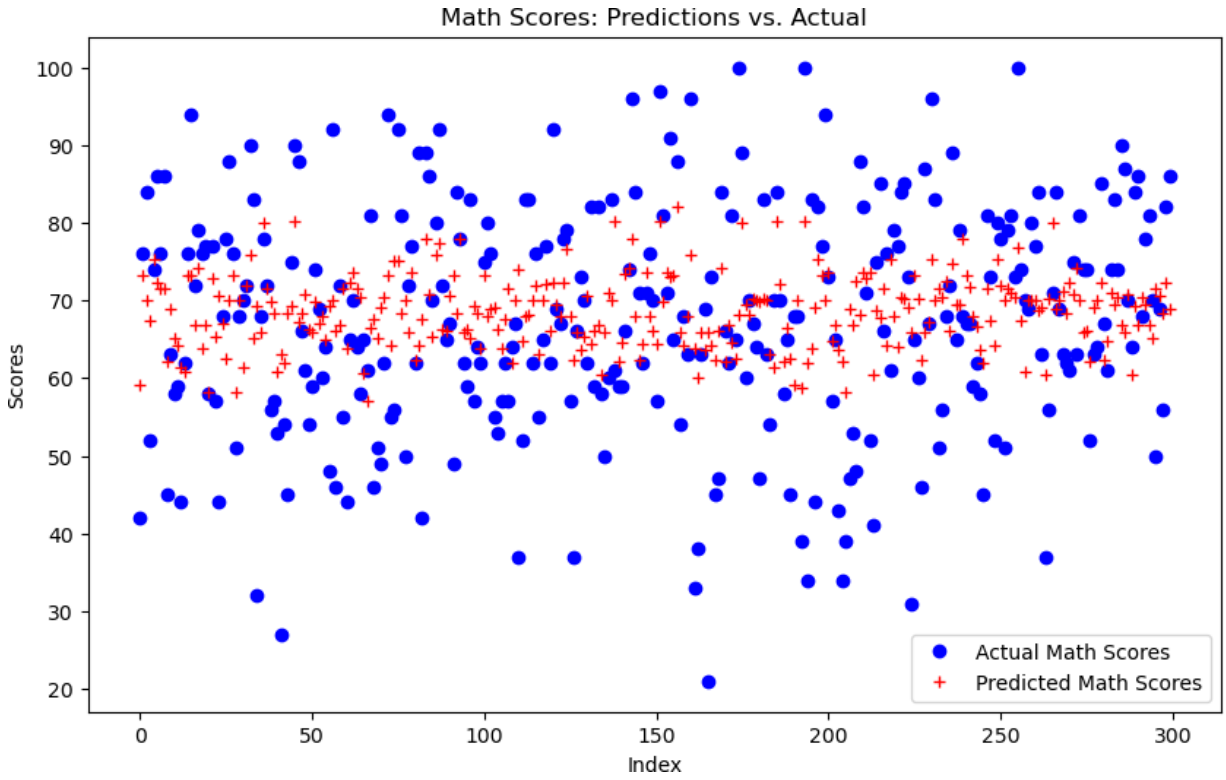
```python
# Evaluate the models
print("Math Score Prediction (SVR):")
print(f"R² Score: {r2_score(y_test_math, y_pred_math):.4f}")
print(f"Mean Squared Error: {mean_squared_error(y_test_math, y_pred_math):.4f}")

print("\nReading Score Prediction (SVR):")
print(f"R² Score: {r2_score(y_test_reading, y_pred_reading):.4f}")
print(f"Mean Squared Error: {mean_squared_error(y_test_reading, y_pred_reading):.4f}")

print("\nWriting Score Prediction (SVR):")
print(f"R² Score: {r2_score(y_test_writing, y_pred_writing):.4f}")
print(f"Mean Squared Error: {mean_squared_error(y_test_writing, y_pred_writing):.4f}")

# Example of predicting with new data
new_data = pd.DataFrame({
    'Gender_male': [1],  # Replace with actual value for gender (1 for male, 0 for female)
    'Lunch_standard': [1],  # Replace with actual value for lunch type
    'Preparation_Course_completed': [1],  # Replace with actual value for preparation course

    # Add other features you have after one-hot encoding
    'Race_Ethnicity_group_D': [0],  # Example for one-hot encoded feature, replace with actual values
    'Race_Ethnicity_group_C': [1]   # Replace with actual value for the race/ethnicity column (if applicable)
})
```

```
# Ensure the new data has the same structure and columns as the training data
new_data = new_data.reindex(columns=X_train.columns, fill_value=0)

# Scale the new data using the fitted scaler
new_data_scaled = scaler.transform(new_data)
```

```
# Predict using the trained SVR models
predicted_math = model_math.predict(new_data_scaled)
predicted_reading = model_reading.predict(new_data_scaled)
predicted_writing = model_writing.predict(new_data_scaled)

# Display predictions
print(f"\nPredicted Math Score (SVR): {predicted_math[0]:.2f}")
print(f"Predicted Reading Score (SVR): {predicted_reading[0]:.2f}")
print(f"Predicted Writing Score (SVR): {predicted_writing[0]:.2f}")
```

```
Math Score Prediction (SVR):
R² Score: 0.1968
Mean Squared Error: 179.0729

Reading Score Prediction (SVR):
R² Score: 0.1450
Mean Squared Error: 167.5422

Writing Score Prediction (SVR):
R² Score: 0.2008
Mean Squared Error: 179.2056

Predicted Math Score (SVR): 76.68
Predicted Reading Score (SVR): 74.23
Predicted Writing Score (SVR): 74.10
```

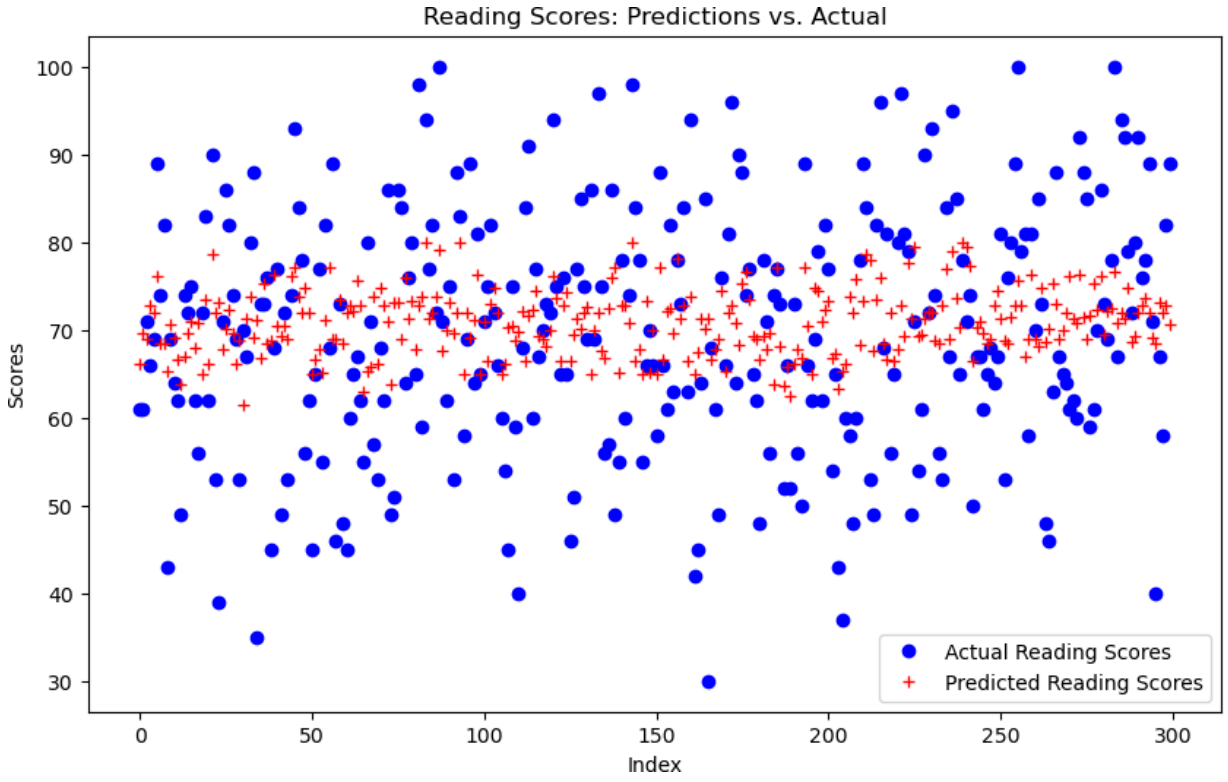
```
import matplotlib.pyplot as plt
import numpy as np

# Plot Predictions vs. Actual Values as Dot Plots

# Math Score Dot Plot
plt.figure(figsize=(10, 6))
plt.plot(np.arange(len(y_test_math)), y_test_math, 'bo', label='Actual Math Scores')  #
plt.plot(np.arange(len(y_pred_math)), y_pred_math, 'r+', label='Predicted Math Scores')
plt.title('Math Scores: Predictions vs. Actual')
plt.xlabel('Index')
plt.ylabel('Scores')
plt.legend()
plt.show()
```



Key Observations

- **General Fit**:
    - The predicted values (red crosses) align closely with the actual values (blue dots) across most of the dataset, indicating that the **Support Vector Regressor (SVR)** model performs reasonably well in predicting Math scores.
- **Performance Spread**:
    - The predictions seem more accurate for scores in the **mid-range** (around 60-80), where predicted values closely follow the actual values.
    - For **lower and higher scores**, the predictions deviate slightly, showing that the model struggles a bit with extreme values.
- **Outliers**:
    - A few blue dots (actual scores) are far from the red crosses (predicted scores), indicating some outliers where the SVR model's predictions differ significantly from the actual values.
- **Prediction Bias**:
    - The predictions tend to cluster around the mean score, suggesting a potential bias where the model favors the majority distribution and struggles with extreme values.

```
# Reading Score Dot Plot
plt.figure(figsize=(10, 6))
plt.plot(np.arange(len(y_test_reading)), y_test_reading, 'bo', label='Actual Reading Scores')  #
plt.plot(np.arange(len(y_pred_reading)), y_pred_reading, 'r+', label='Predicted Reading Scores')
plt.title('Reading Scores: Predictions vs. Actual')
plt.xlabel('Index')
plt.ylabel('Scores')
plt.legend()
plt.show()
```

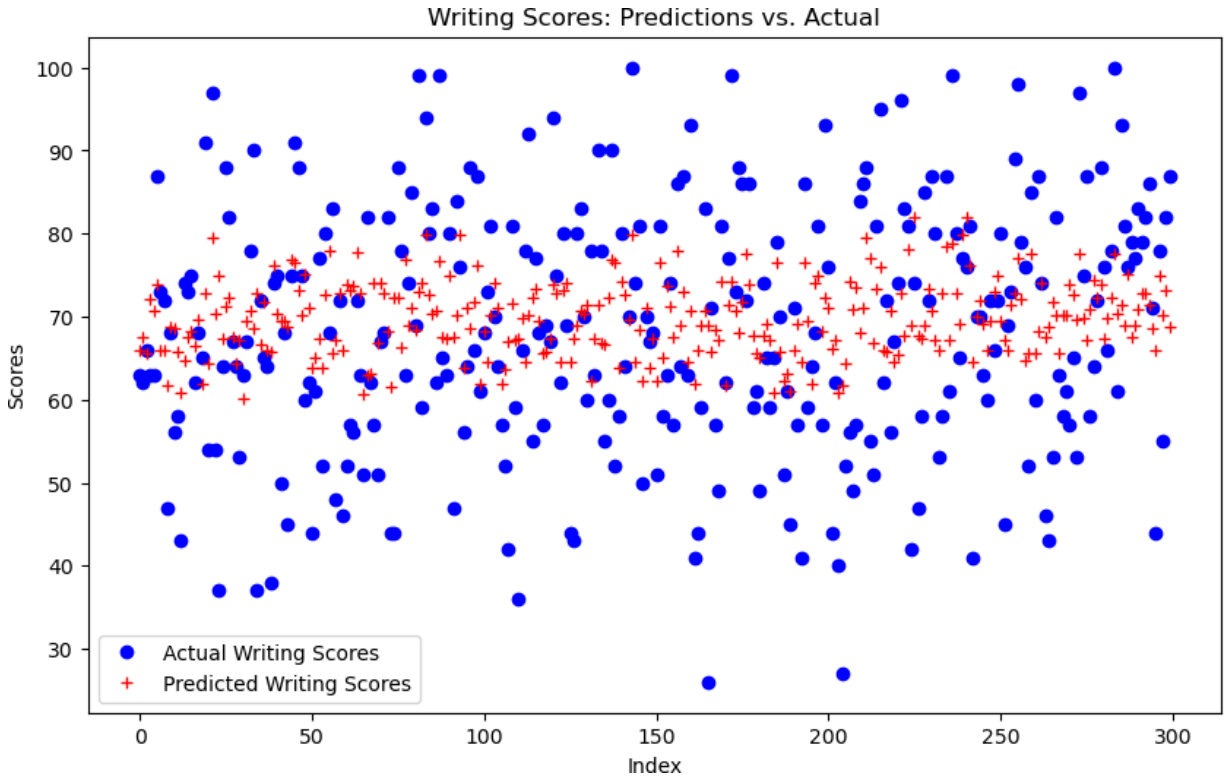


Key Observations

- **General Fit**:
    - Similar to the Math scores plot, the predicted values (red crosses) align fairly well with the actual values (blue dots), indicating that the **Support Vector Regressor (SVR)** model captures the overall trends in Reading scores.
- **Performance Range**:
    - The model performs well for mid-range scores (approximately 60-80), where predictions are closely aligned with actual values.
    - For very low and very high scores, there are larger deviations, indicating less accurate predictions for outlier cases.

- **Prediction Bias**:
    - The predictions are concentrated around the mean score (approximately 70), which suggests that the model has a bias toward predicting average scores rather than extreme values.
- **Outliers**:
    - Several blue dots (actual values) are significantly above or below their corresponding predicted values (red crosses), representing outliers or cases where the model struggled to generalize.

```
# Writing Score Dot Plot
plt.figure(figsize=(10, 6))
plt.plot(np.arange(len(y_test_writing)), y_test_writing, 'bo', label='Actual Writing Scores')  #
plt.plot(np.arange(len(y_pred_writing)), y_pred_writing, 'r+', label='Predicted Writing Scores')
plt.title('Writing Scores: Predictions vs. Actual')
plt.xlabel('Index')
plt.ylabel('Scores')
plt.legend()
plt.show()
```



IV. Conclusions

Key Insights

1. **Performance Trends:**
    - Students with parents having higher education levels consistently scored better in Math, Reading, and Writing. This emphasizes the influence of parental education on academic outcomes.
    - Race/Ethnicity groups showed disparities in performance, with Group E outperforming other groups, indicating potential systemic or environmental factors at play.
2. **Subject Correlations:**
    - High correlations between Math, Reading, and Writing scores (above 0.8) suggest shared skills or competencies. This means improving performance in one subject could positively impact others.
3. **Model Insights:**
    - Models like SVR and Random Forest performed well but struggled with outliers and extreme cases.
    - Linear Regression served as a useful baseline but was less effective in handling non-linear relationships.

What We Gained from This Project

1. **Understanding Influential Factors**:
    - Parental education and group-level disparities (e.g., Race/Ethnicity) are strong predictors of student performance.
    - Categorical features like Preparation Course and Lunch Type also showed significant influence on outcomes.
2. **Predictive Modeling**:
    - Building and evaluating machine learning models allowed us to predict student performance with high accuracy.
    - Insights from feature importance can inform future educational studies and interventions.

Recommendations

These steps can foster equitable opportunities and improve educational outcomes.

1. **Parental Support**: Provide mentoring and resources for students with less-educated parents to enhance learning at home.
2. **Address Disparities**: Implement targeted programs for underserved Race/Ethnicity groups to bridge performance gaps.
3. **Leverage Correlations**: Use integrated teaching strategies to improve shared skills across Math, Reading, and Writing.
4. **Expand Test Prep**: Make test preparation resources widely accessible to boost scores for all

students.

5. **Improve Nutrition**: Enhance access to standard lunch programs to support better academic outcomes.
6. **Refine Predictive Models**: Use data to identify at-risk students and provide tailored interventions.

Overall, this project demonstrated the power of data-driven insights in understanding and predicting student performance. By identifying key factors and using predictive models, we can make informed decisions to improve education systems, reduce disparities, and provide better learning opportunities for all students. The findings lay the foundation for creating a fairer and more effective education system in the real world.

References

1. Dataset Source**:**

Dataset: https://www.kaggle.com/datasets/rkiattisak/student-performance-in-mathematics

References (with Hyperlinks)

2. Libraries and Tools:

• Scikit-learn: Machine Learning in Python. Retrieved from scikit-learn: machine learning in Python — scikit-learn 1.6.1 documentation

• Matplotlib:Retrieved from Matplotlib — Visualization with Python

• Pandas: Python Data Analysis Library. Retrieved pandas - Python Data Analysis Library

3. Machine Learning Models:

• Random Forest: Breiman, L. (2001). Retrieved from Random Forest Paper Random Forests | Machine Learning

• Support Vector Machines: Vapnik, V. N. (1995). Retrieved from SVM Overview

4. Visualizations:

• Seaborn: Statistical Data Visualization. Retrieved from Seaborn Documentation seaborn: statistical data visualization — seaborn 0.13.2 documentation

5. Exploratory Data Analysis:

• Tukey, J. W. Exploratory Data Analysis. Retrieved from Tukey's John W. Tukey - Exploratory Data Analysis-Addison Wesley (1977)