```
# Step 1: Import essential libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

```
# Step 2: Upload dataset file to Colab
from google.colab import files
uploaded = files.upload()
```

Choose Files StudentsPe…ance[1].csv

- **StudentsPerformance[1].csv**(text/csv) - 72036 bytes, last modified: 11/13/2025 - 100% done

```
Saving StudentsPerformance[1].csv to StudentsPerformance[1] (1).csv
```

```
# Step 3: Load dataset
df = pd.read_csv("StudentsPerformance[1].csv")

# Display first 5 rows
print("First 5 Rows:")
display(df.head())

# Show basic info
print("\nDataset Info:")
df.info()

# Check for missing values
print("\nMissing Values:")
print(df.isnull().sum())
```

```
First 5 Rows:
```

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group A	associate's degree	free/reduced	none	47	57	44
4	male	group C	some college	standard	none	76	78	75

```
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column                       Non-Null Count  Dtype 
---  ------                       --------------  ----- 
 0   gender                       1000 non-null   object
 1   race/ethnicity               1000 non-null   object
 2   parental level of education  1000 non-null   object
 3   lunch                        1000 non-null   object
 4   test preparation course      1000 non-null   object
 5   math score                   1000 non-null   int64 
 6   reading score                1000 non-null   int64 
 7   writing score                1000 non-null   int64 
dtypes: int64(3), object(5)
memory usage: 62.6+ KB

Missing Values:
gender                         0
race/ethnicity                 0
parental level of education    0
lunch                          0
test preparation course        0
math score                     0
reading score                  0
writing score                  0
```

```
# Step 4: Summary statistics
print("Summary Statistics:")
display(df.describe(include='all'))
```

Summary Statistics:

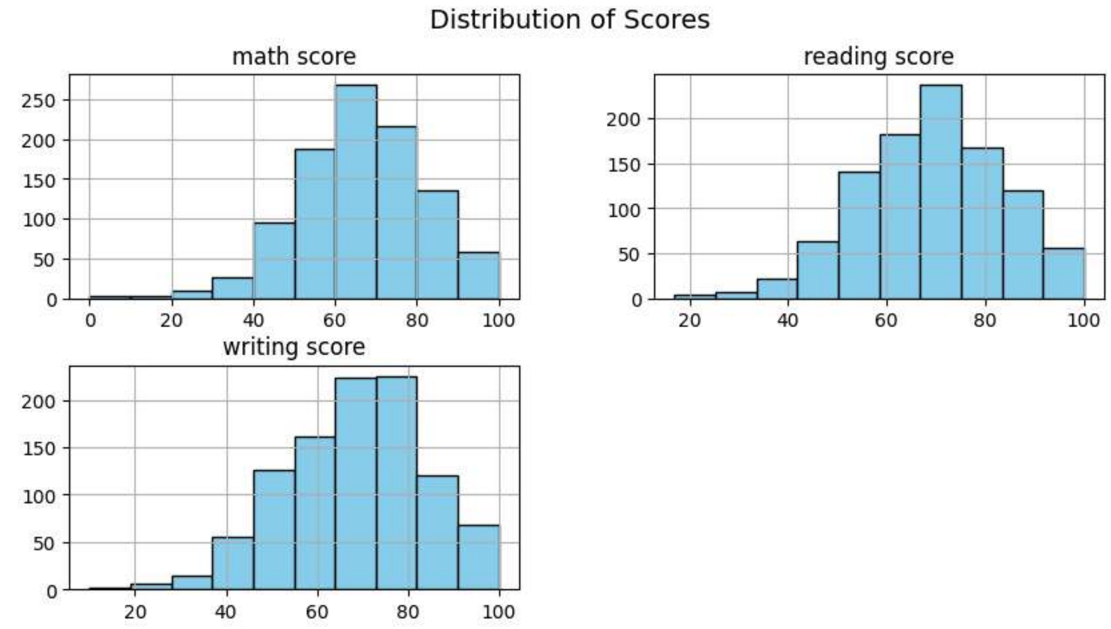
	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
count	1000	1000	1000	1000	1000	1000.00000	1000.000000	1000.000000
unique	2	5	6	2	2	NaN	NaN	NaN
top	female	group C	some college	standard	none	NaN	NaN	NaN
freq	518	319	226	645	642	NaN	NaN	NaN
mean	NaN	NaN	NaN	NaN	NaN	66.08900	69.169000	68.054000
std	NaN	NaN	NaN	NaN	NaN	15.16308	14.600192	15.195657
min	NaN	NaN	NaN	NaN	NaN	0.00000	17.000000	10.000000
25%	NaN	NaN	NaN	NaN	NaN	57.00000	59.000000	57.750000
50%	NaN	NaN	NaN	NaN	NaN	66.00000	70.000000	69.000000
75%	NaN	NaN	NaN	NaN	NaN	77.00000	79.000000	79.000000

```
# Step 5: Encode categorical columns
le = LabelEncoder()
for c in cat_cols:
    df[c] = le.fit_transform(df[c])

print("After Encoding:")
display(df.head())
```

After Encoding:

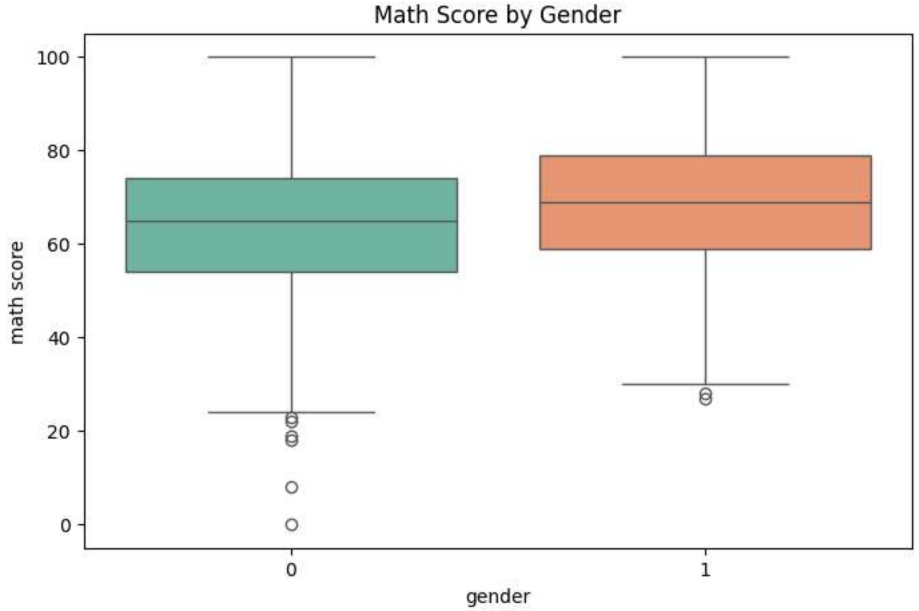
	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	0	1	1	1	1	72	72	74
1	0	2	4	1	0	69	90	88
2	0	1	3	1	1	90	95	93
3	1	0	0	0	1	47	57	44
4	1	2	4	1	1	76	78	75

```
# Step 5a: Distribution of numerical scores
num_cols = ['math score', 'reading score', 'writing score']
df[num_cols].hist(figsize=(10,5), bins=10, color='skyblue', edgecolor='black')
plt.suptitle("Distribution of Scores", fontsize=14)
plt.show()
```



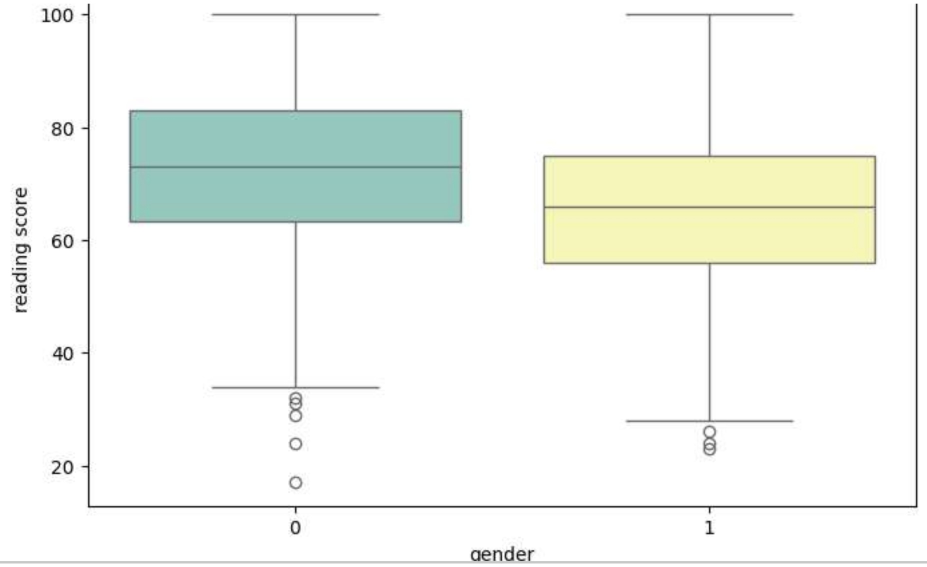
```
# Step 5b: Gender-wise performance comparison
plt.figure(figsize=(8,5))
sns.boxplot(x='gender', y='math score', data=df, palette='Set2')
plt.title("Math Score by Gender")
plt.show()

plt.figure(figsize=(8,5))
sns.boxplot(x='gender', y='reading score', data=df, palette='Set3')
plt.title("Reading Score by Gender")
plt.show()

plt.figure(figsize=(8,5))
sns.boxplot(x='gender', y='writing score', data=df, palette='Set1')
plt.title("Writing Score by Gender")
plt.show()
```

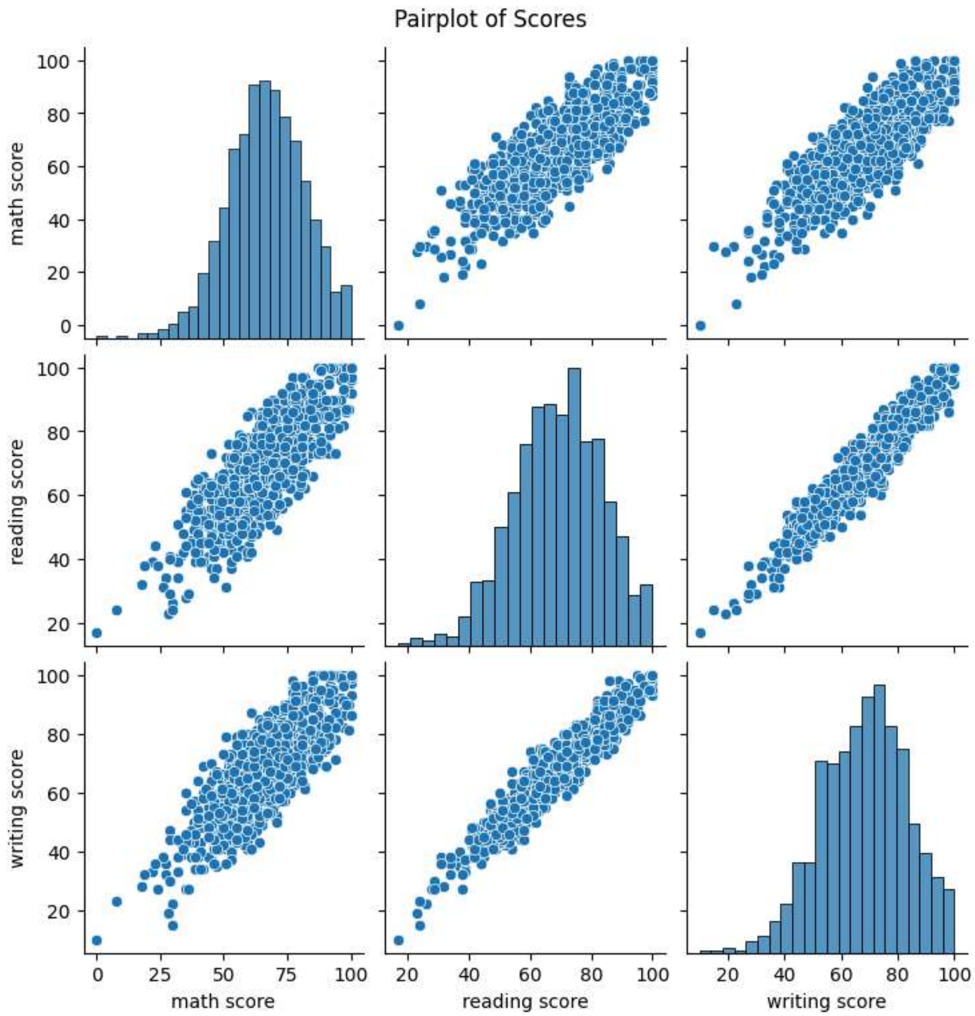
```
/tmp/ipython-input-312818870.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variabl

  sns.boxplot(x='gender', y='math score', data=df, palette='Set2')
```



```
/tmp/ipython-input-312818870.py:8: FutureWarning:
```

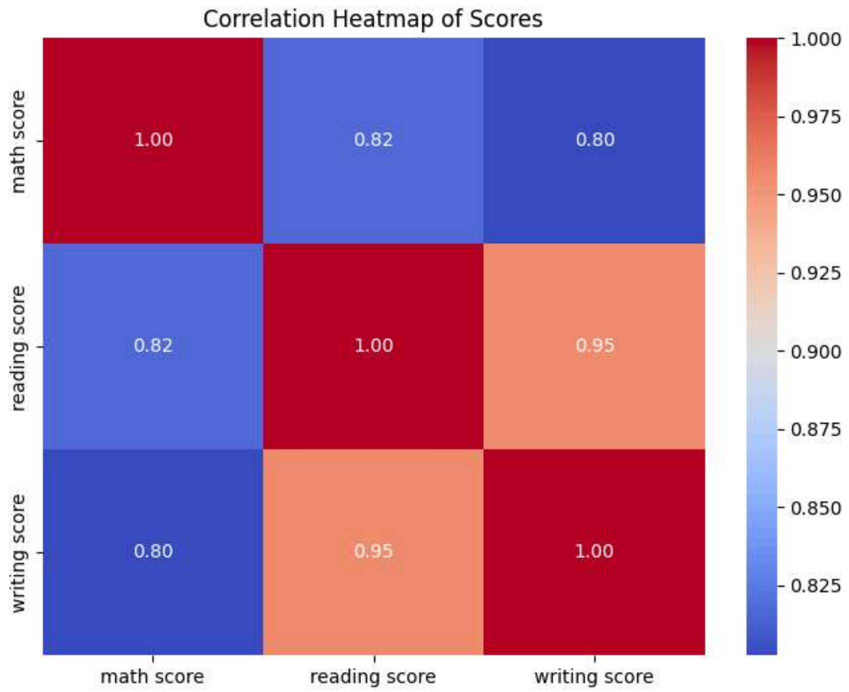
```
# Step 5c: Relationship between scores
sns.pairplot(df[num_cols])
plt.suptitle("Pairplot of Scores", y=1.02)
plt.show()
```





```
ne `x` variabl
```

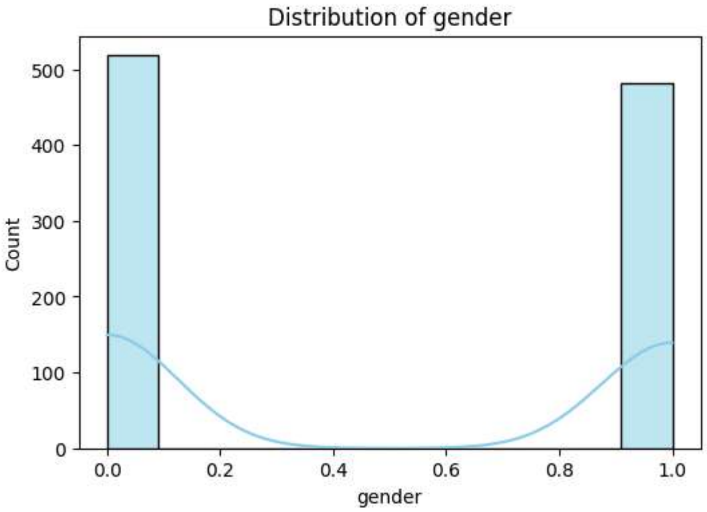
```
# Step 5d: Correlation heatmap
plt.figure(figsize=(8,6))
sns.heatmap(df[num_cols].corr(), annot=True, cmap='coolwarm', fmt='.2f')
plt.title("Correlation Heatmap of Scores")
plt.show()
```

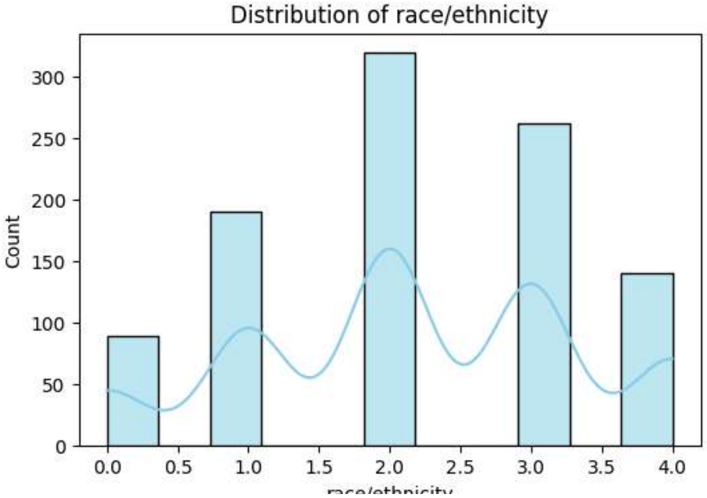

Correlation Heatmap of Scores
math score
reading score
writing score
1.00
0.82
0.80
0.82
1.00
0.95
0.80
0.95
1.00
math score
reading score
writing score
1.000
0.975
0.950
0.925
0.900
0.875
0.850
0.825

```
# ---- UNIVARIATE ANALYSIS ----

print("Dataset Shape:", df.shape)
print("\nSummary Statistics (Numeric Columns):")
display(df.describe())

# The following line is removed as all object columns were encoded in a previous step,
# leading to a ValueError. If univariate analysis on original categorical features is needed,
# it should be performed before encoding.
# print("\nSummary of Categorical Columns:")
# display(df.describe(include='object'))

# Numerical Columns
num_cols = df.select_dtypes(include=np.number).columns.tolist()

# Categorical Columns (will be empty if all were encoded)
cat_cols = df.select_dtypes(include='object').columns.tolist()

# Histograms (Distribution)
for col in num_cols:
    plt.figure(figsize=(6,4))
    sns.histplot(df[col], kde=True, color='skyblue')
    plt.title(f"Distribution of {col}")
    plt.show()

# Boxplots (Outliers + Spread)
for col in num_cols:
    plt.figure(figsize=(6,3))
    sns.boxplot(x=df[col])
    plt.title(f"Boxplot of {col}")
    plt.show()

# Countplots (Categorical) - This loop will not execute if cat_cols is empty
for col in cat_cols:
    plt.figure(figsize=(7,4))
    sns.countplot(x=df[col], palette='Set2')
    plt.title(f"Count of {col}")
    plt.xticks(rotation=45)
    plt.show()
```

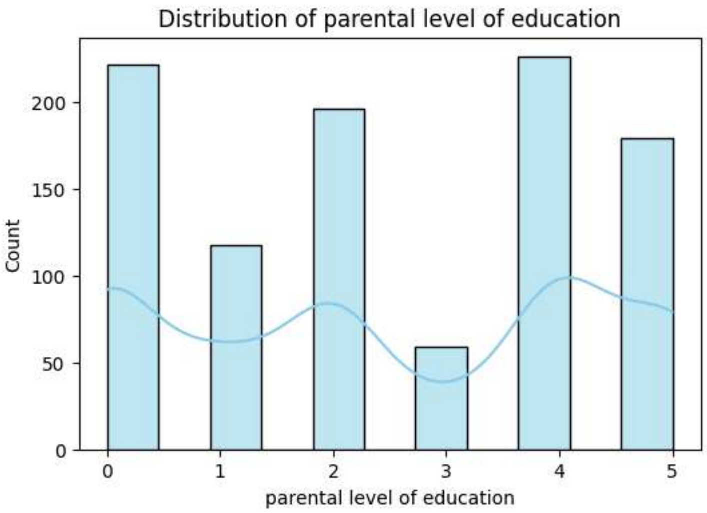
```
Dataset Shape: (1000, 8)

Summary Statistics (Numeric Columns):
```

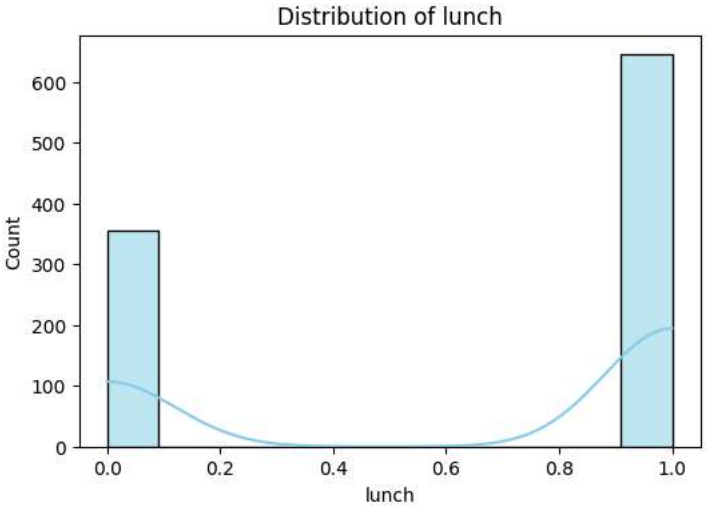
	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.00000	1000.000000	1000.000000
mean	0.482000	2.174000	2.486000	0.645000	0.642000	66.08900	69.169000	68.054000
std	0.499926	1.157179	1.829522	0.478753	0.479652	15.16308	14.600192	15.195657
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000	17.000000	10.000000
25%	0.000000	1.000000	1.000000	0.000000	0.000000	57.00000	59.000000	57.750000
50%	0.000000	2.000000	2.000000	1.000000	1.000000	66.00000	70.000000	69.000000
75%	1.000000	3.000000	4.000000	1.000000	1.000000	77.00000	79.000000	79.000000
max	1.000000	4.000000	5.000000	1.000000	1.000000	100.00000	100.000000	100.000000

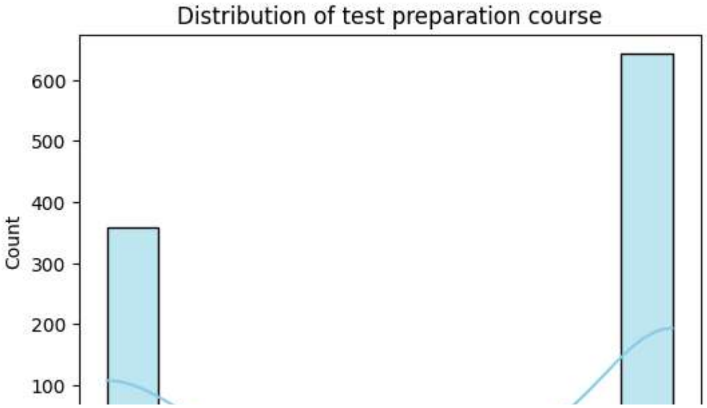


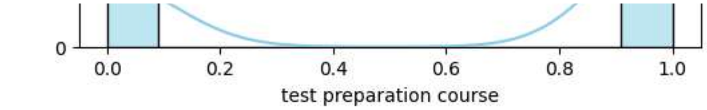


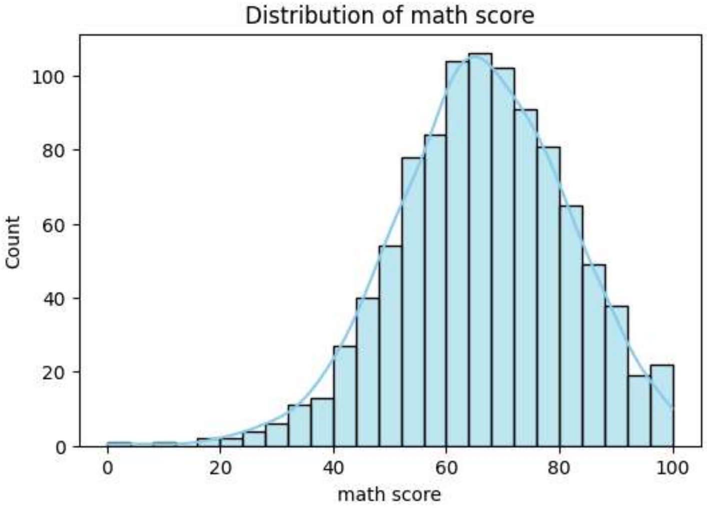
race/ethnicity

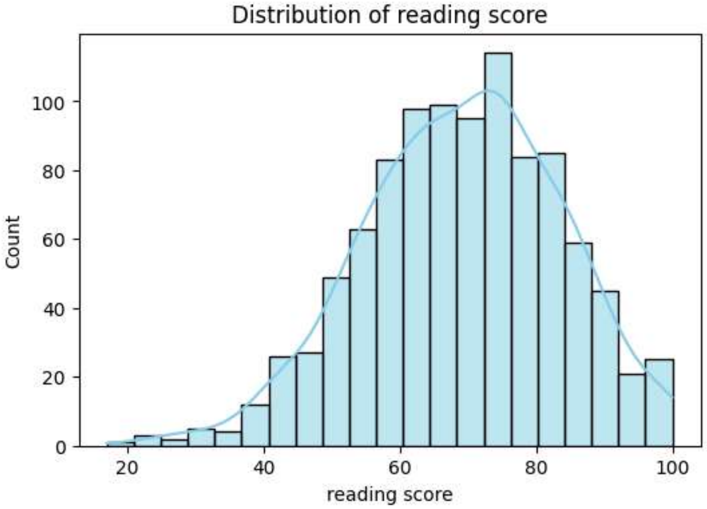


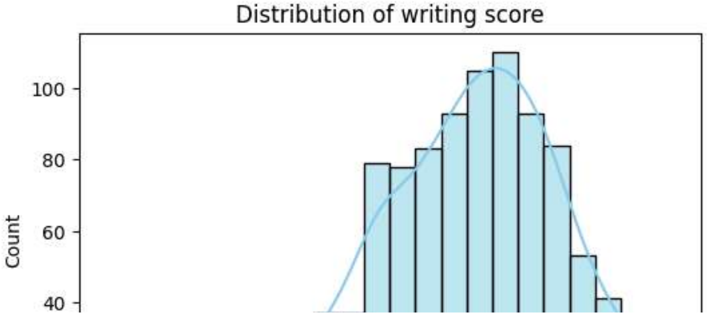


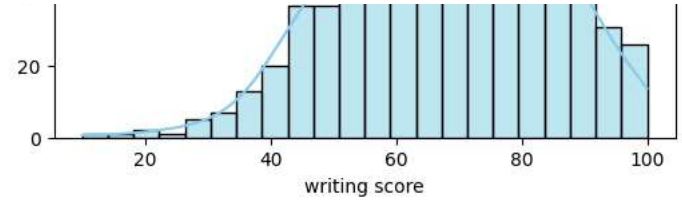



0
0.0
0.2
0.4
0.6
0.8
1.0
test preparation course

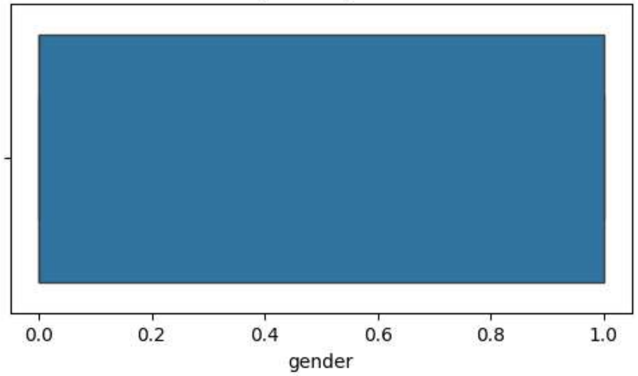

Distribution of math score
Count
0
20
40
60
80
100
0
20
40
60
80
100
math score


Distribution of reading score
Count
0
20
40
60
80
100
20
40
60
80
100
reading score

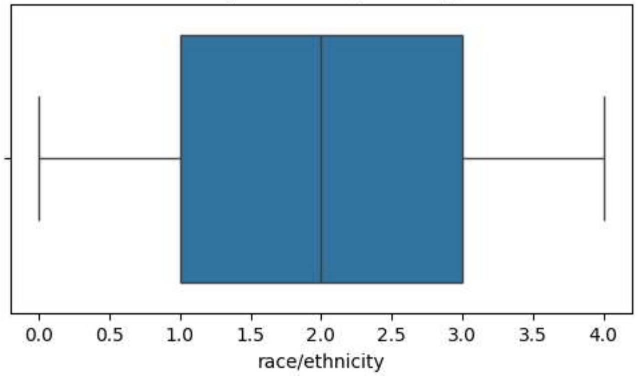

Distribution of writing score
Count
40
60
80
100



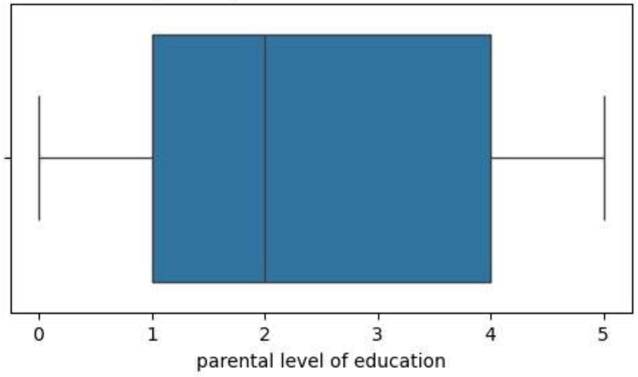
Boxplot of gender

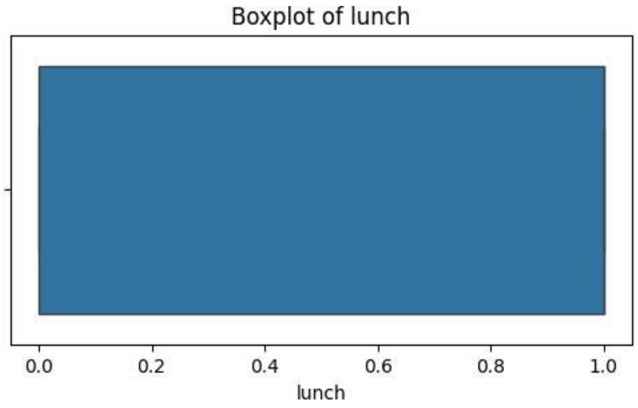


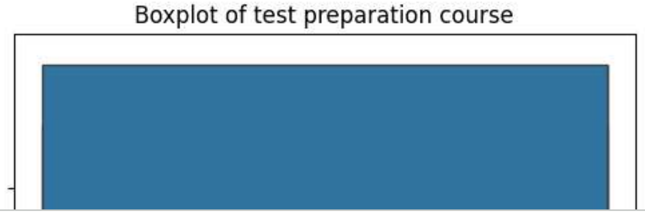
Boxplot of race/ethnicity



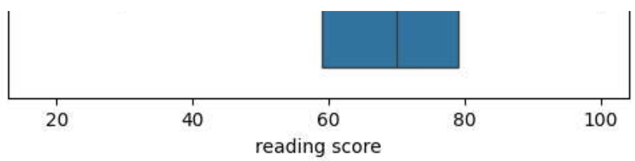
Boxplot of parental level of education

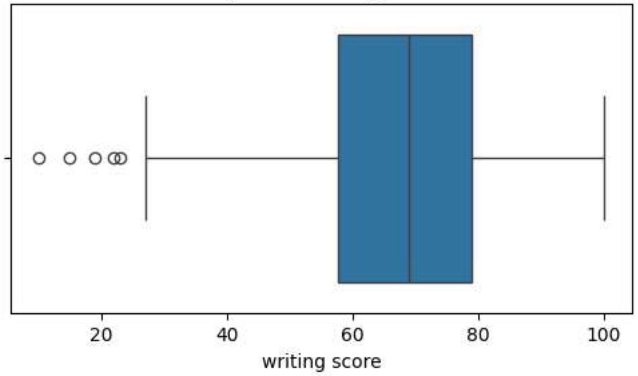


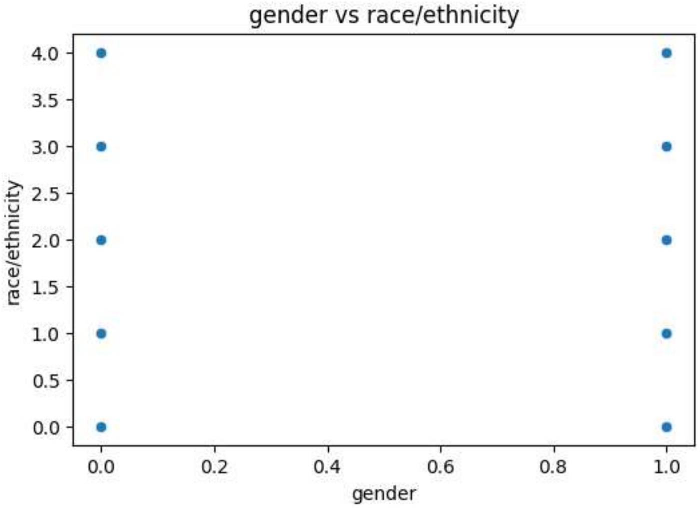


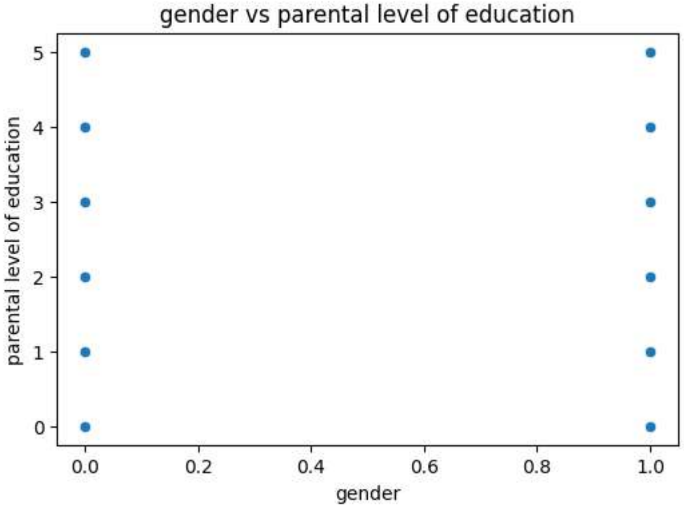


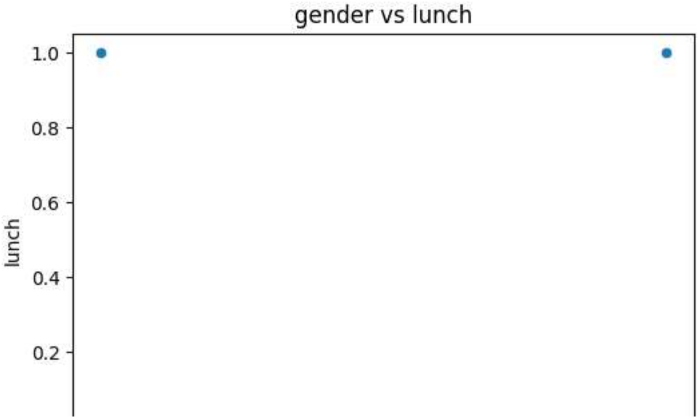
```
# ---- BIVARIATE ANALYSIS ----

# 1. NUMERIC vs NUMERIC → SCATTERPLOTS
num_cols = df.select_dtypes(include=np.number).columns.tolist()

for i in range(len(num_cols)):
    for j in range(i+1, len(num_cols)):
        plt.figure(figsize=(6,4))
        sns.scatterplot(x=df[num_cols[i]], y=df[num_cols[j]])
        plt.title(f"{num_cols[i]} vs {num_cols[j]}")
        plt.show()

# 2. CATEGORICAL vs NUMERIC → BOXPLOTS
cat_cols = df.select_dtypes(include='object').columns.tolist()

for cat in cat_cols:
    for num in num_cols:
        plt.figure(figsize=(7,4))
        sns.boxplot(x=df[cat], y=df[num], palette='Set3')
        plt.title(f"{num} by {cat}")
        plt.xticks(rotation=45)
        plt.show()

# 3. CORRELATION MATRIX (NUMERIC VARIABLES)
plt.figure(figsize=(8,6))
sns.heatmap(df[num_cols].corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Matrix")
plt.show()

# 4. GROUPED MEANS (Categorical vs Numeric)
for cat in cat_cols:
    print(f"\nAverage Values grouped by {cat}:")
    display(df.groupby(cat)[num_cols].mean().round(2))
```

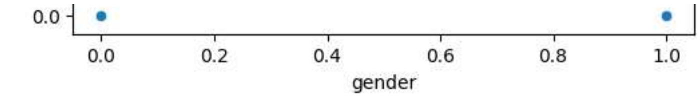


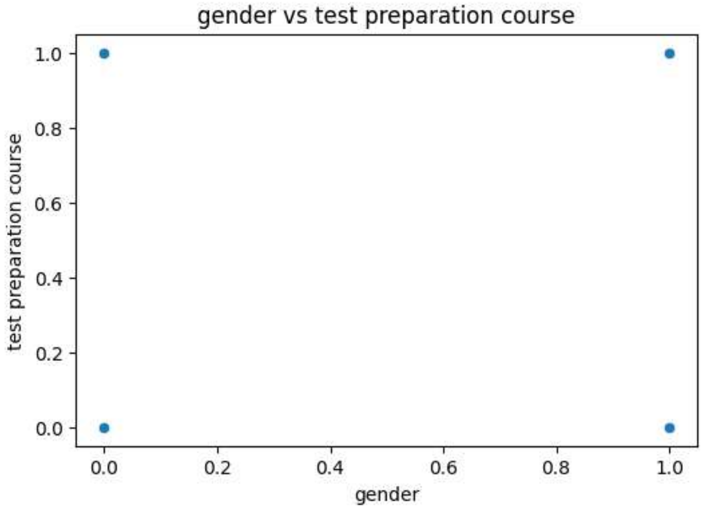

Boxplot of writing score

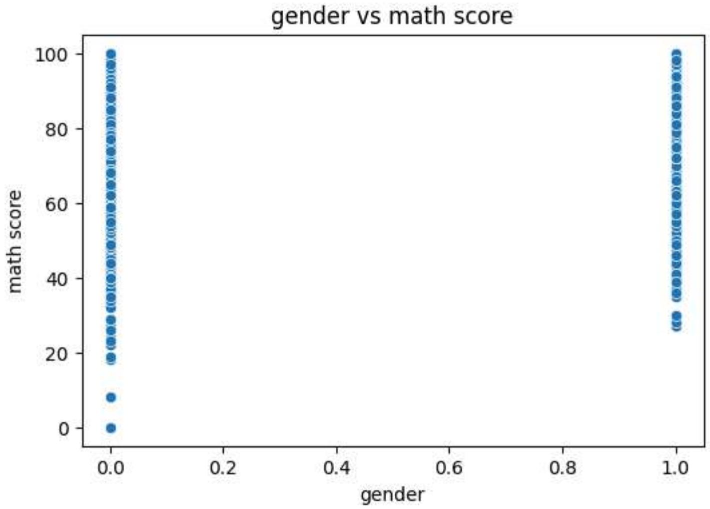

gender vs race/ethnicity
race/ethnicity
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0
gender

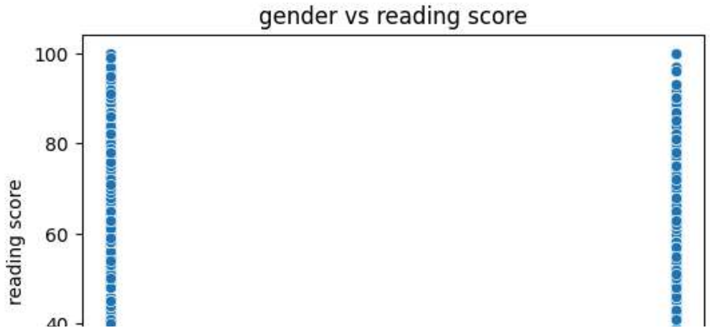

gender vs parental level of education
parental level of education
5
4
3
2
1
0
0.0
0.2
0.4
0.6
0.8
1.0
gender

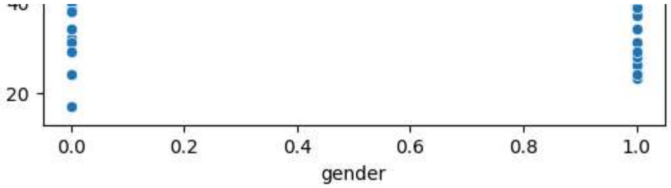

gender vs lunch
lunch
1.0
0.8
0.6
0.4
0.2

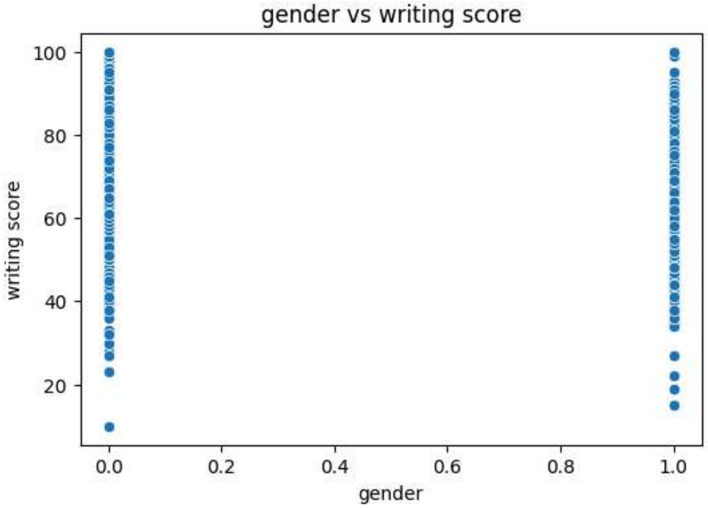


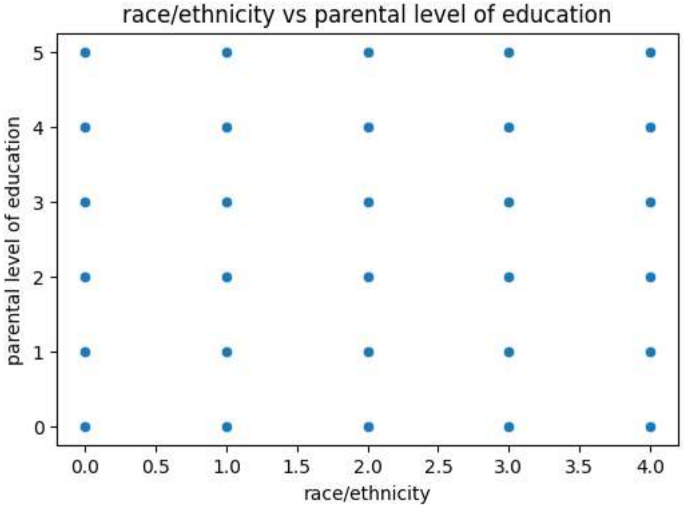

gender vs test preparation course

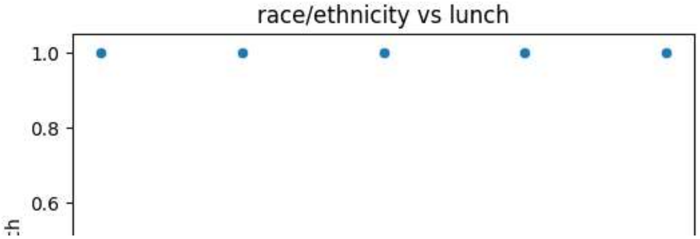

gender vs math score

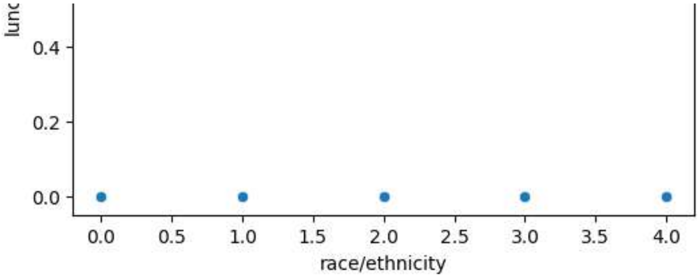

gender vs reading score


40
20
0.0
0.2
0.4
0.6
0.8
1.0
gender

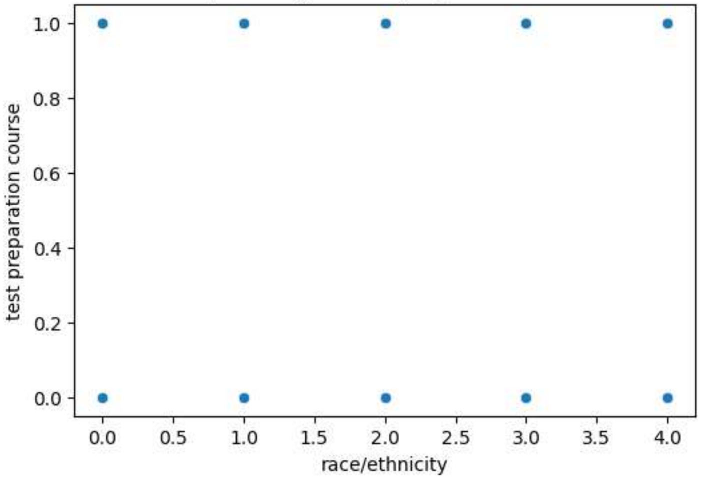

gender vs writing score
writing score
100
80
60
40
20
0.0
0.2
0.4
0.6
0.8
1.0
gender


race/ethnicity vs parental level of education
parental level of education
5
4
3
2
1
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
race/ethnicity

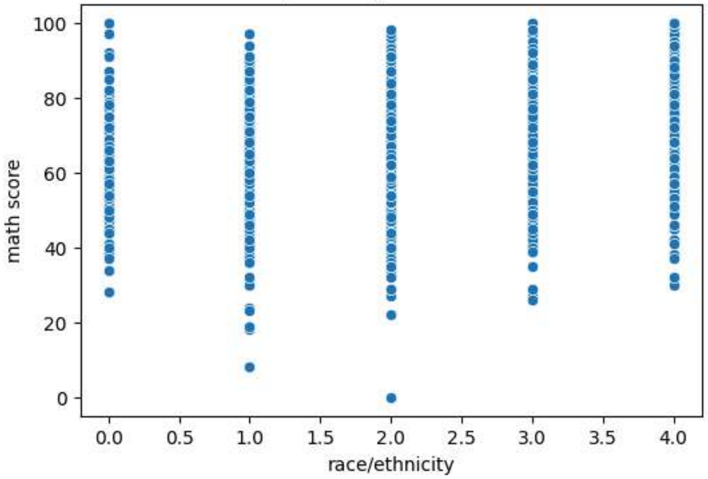

race/ethnicity vs lunch
1.0
0.8
0.6



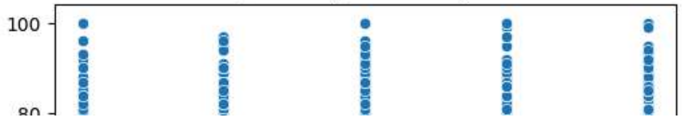
race/ethnicity vs test preparation course

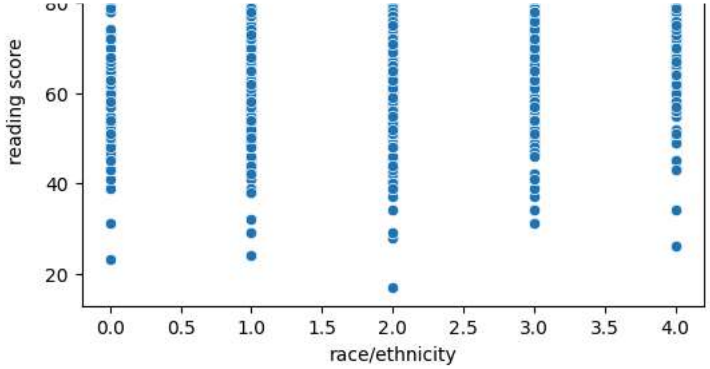


race/ethnicity vs math score

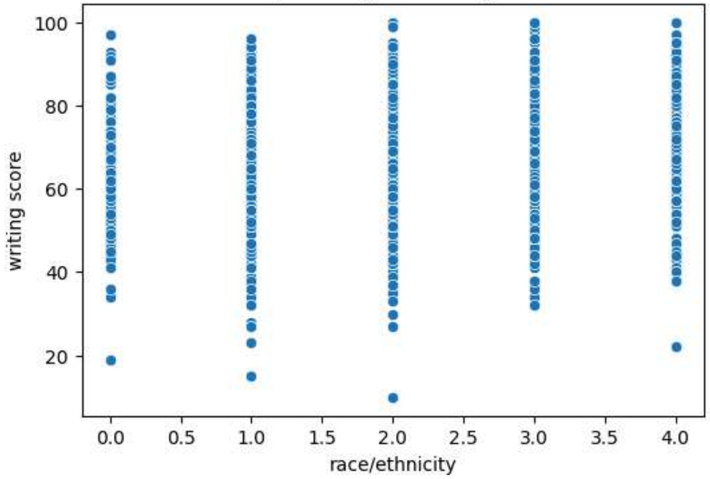


race/ethnicity vs reading score

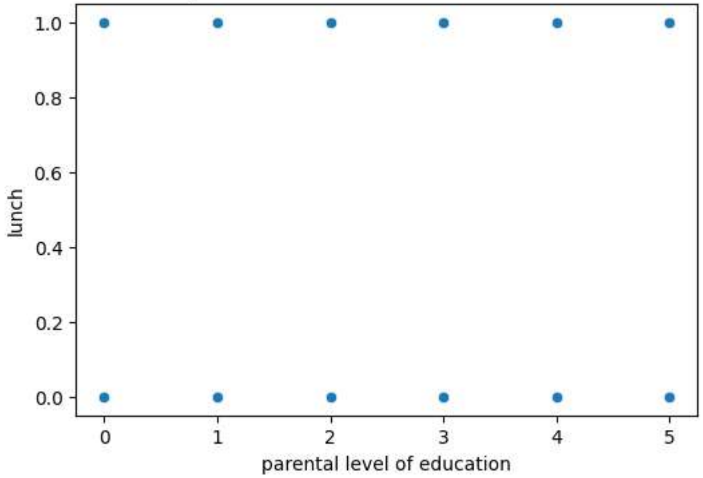




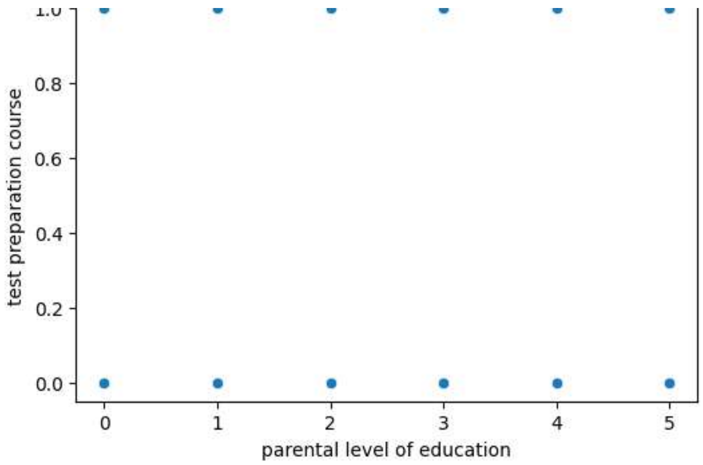
race/ethnicity vs writing score



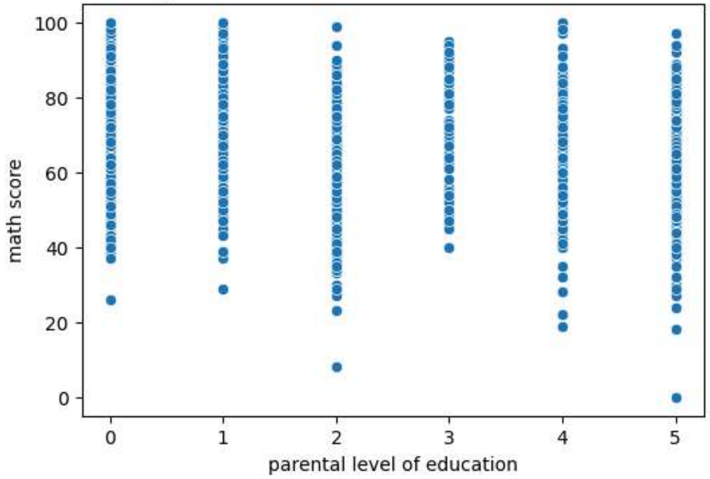
parental level of education vs lunch



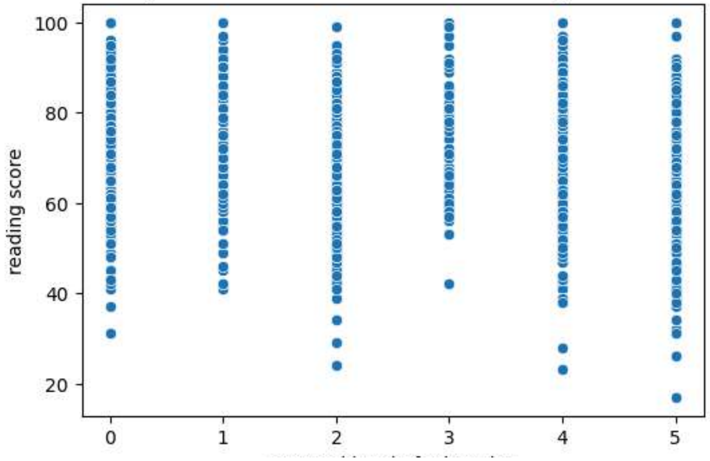
parental level of education vs test preparation course



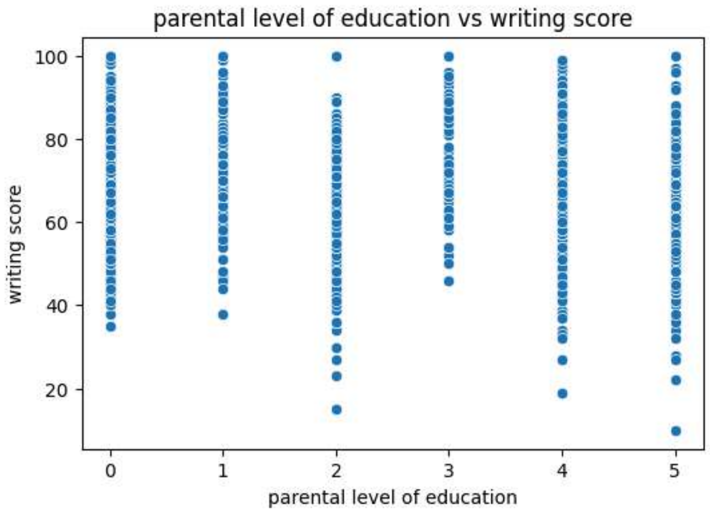
parental level of education vs math score

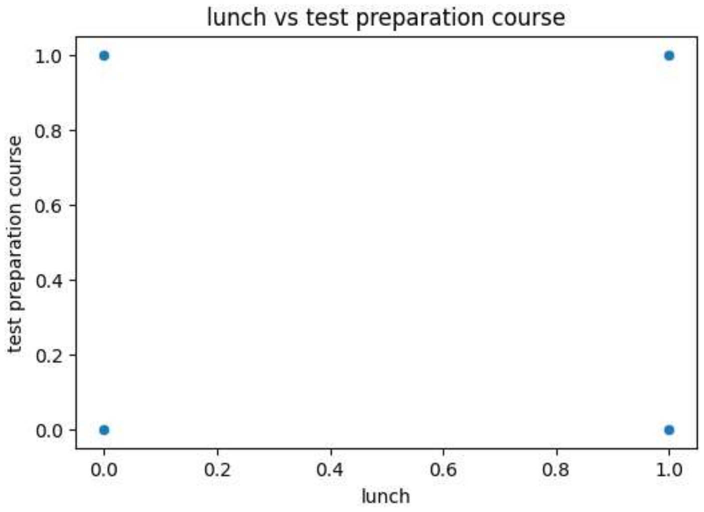


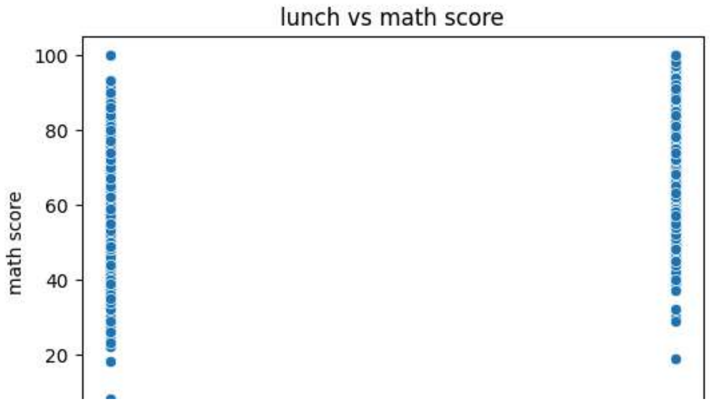
parental level of education vs reading score

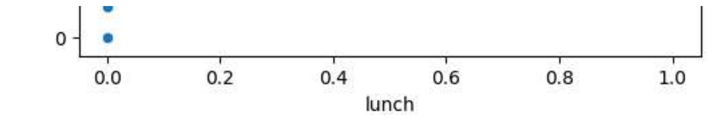


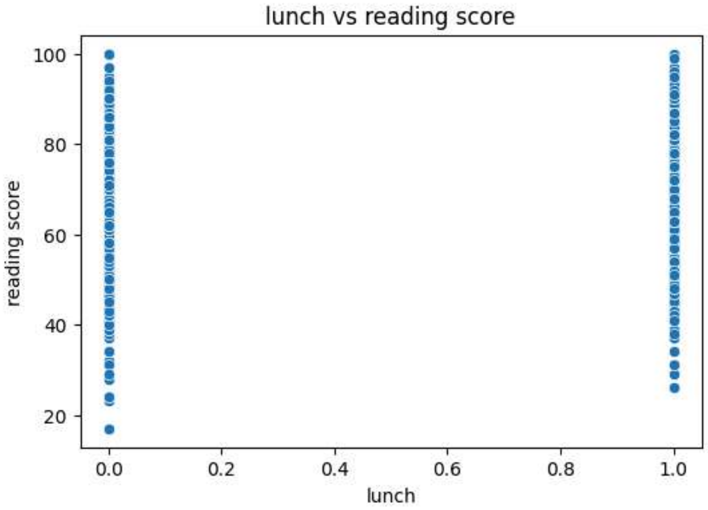
parental level of education

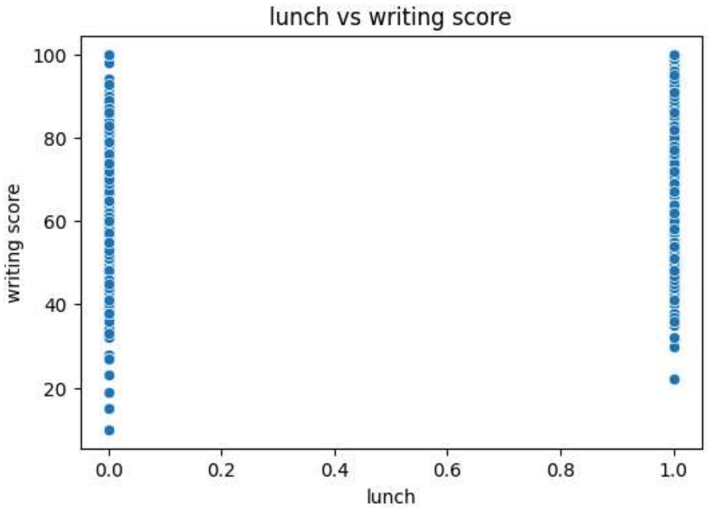


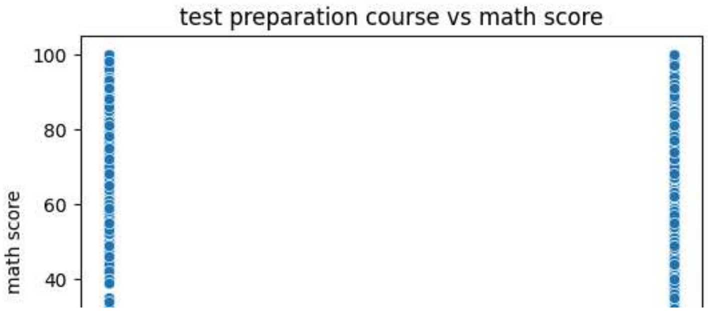


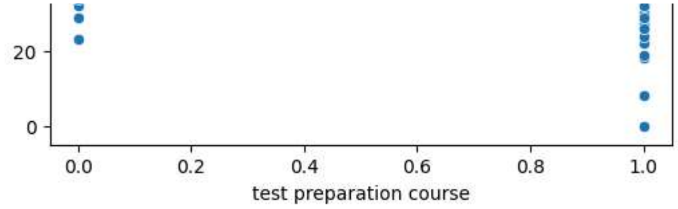


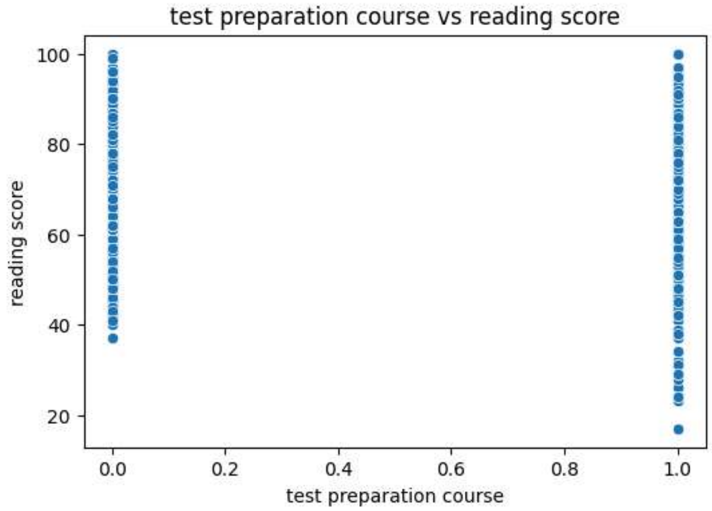

0
0.0
0.2
0.4
0.6
0.8
1.0
lunch

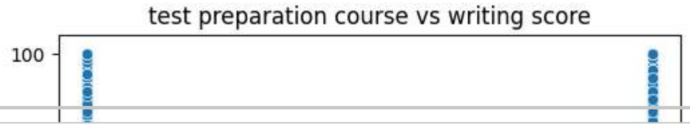

lunch vs reading score
reading score
100
80
60
40
20
0.0
0.2
0.4
0.6
0.8
1.0
lunch


lunch vs writing score
writing score
100
80
60
40
20
0.0
0.2
0.4
0.6
0.8
1.0
lunch

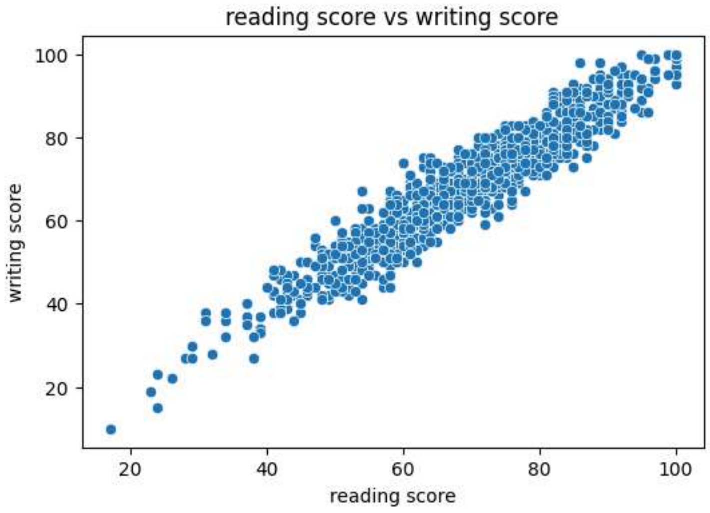

test preparation course vs math score
math score
100
80
60
40

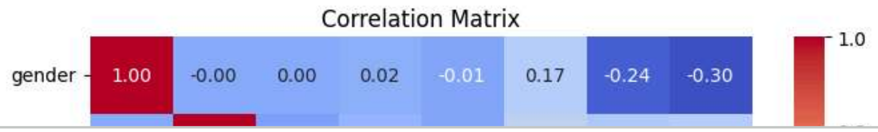


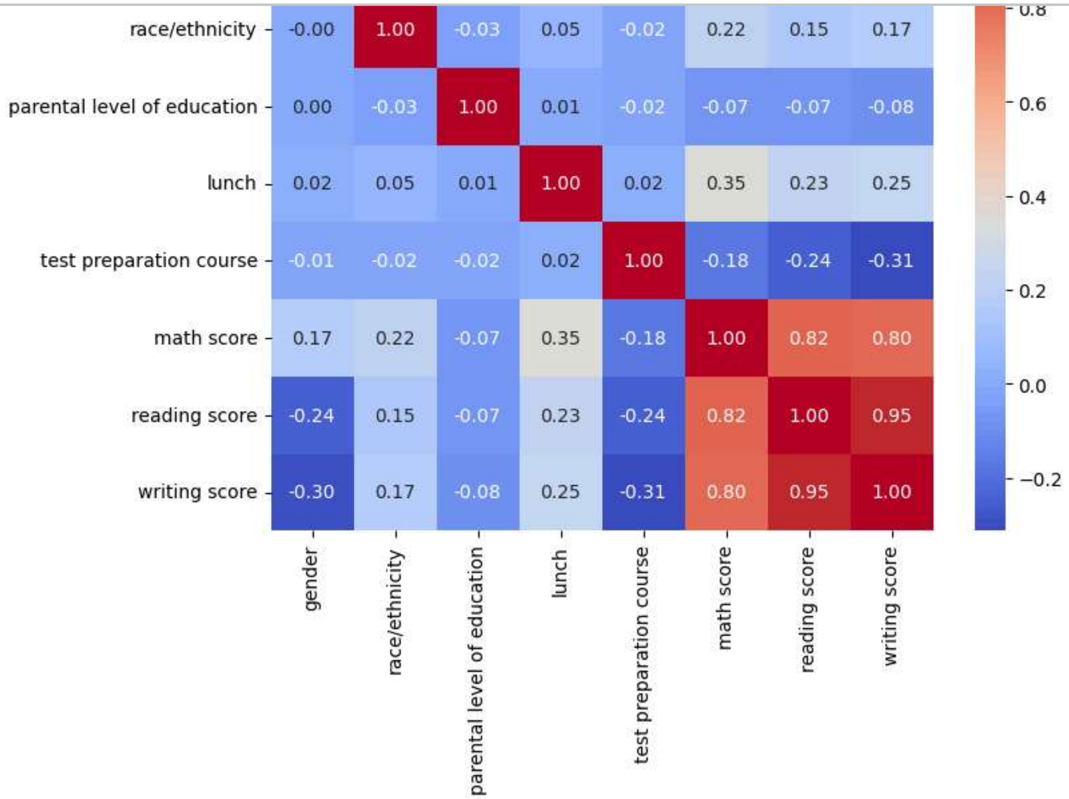

test preparation course vs reading score

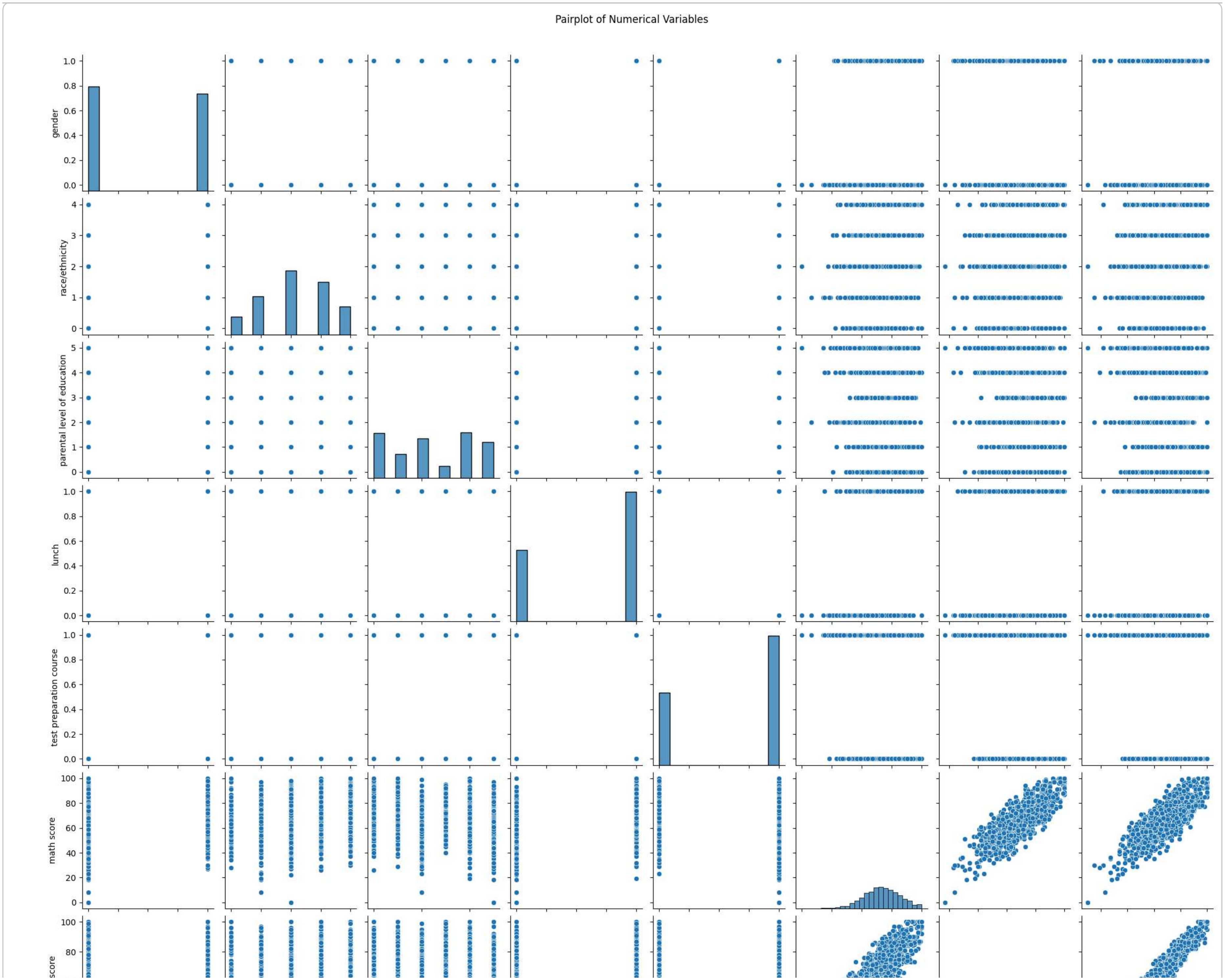

test preparation course vs writing score

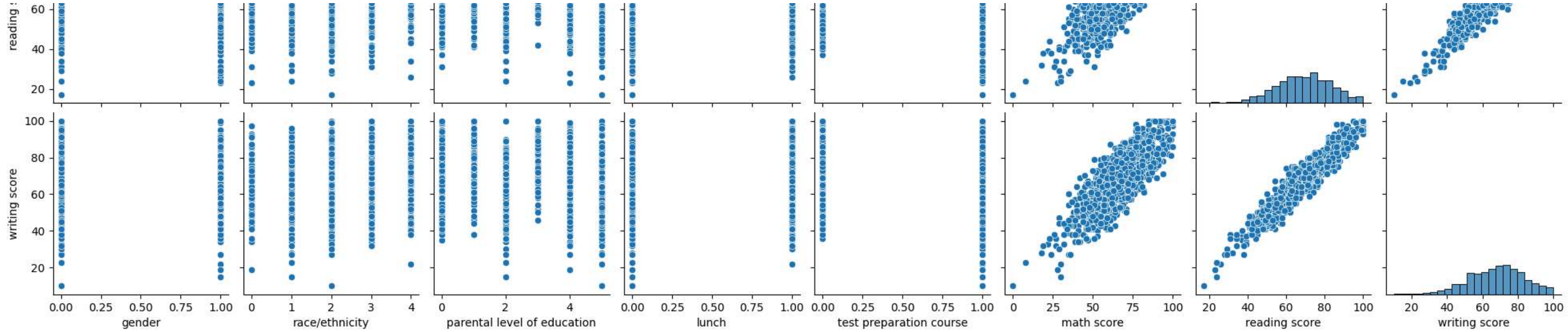
```
# ---- MULTIVARIATE ANALYSIS ----

# 1. PAIRPLOT (ALL NUMERIC VARIABLES)
sns.pairplot(df[num_cols])
plt.suptitle("Pairplot of Numerical Variables", y=1.02)
plt.show()

# 2. FULL HEATMAP (Including Encoded Categorical)
# First encode categorical variables for heatmap
df_encoded = df.copy()
for col in cat_cols:
    df_encoded[col] = df_encoded[col].astype('category').cat.codes

plt.figure(figsize=(10,7))
sns.heatmap(df_encoded.corr(), cmap='coolwarm', annot=False)
plt.title("Full Multivariate Correlation Heatmap")
plt.show()

# 3. FACETGRID (Categorical + Numeric Multivariate Comparison)
for cat in cat_cols:
    g = sns.FacetGrid(df, col=cat, height=4)
    g.map(plt.hist, num_cols[0])  # histogram of first numeric column by category
    plt.suptitle(f"Distribution of {num_cols[0]} by {cat}")
    plt.show()

# 4. MULTIVARIATE BOXPLOT GRID
plt.figure(figsize=(12,6))
sns.boxplot(data=df[num_cols])
plt.title("Multivariate Boxplot of All Numeric Variables")
plt.xticks(rotation=45)
plt.show()
```

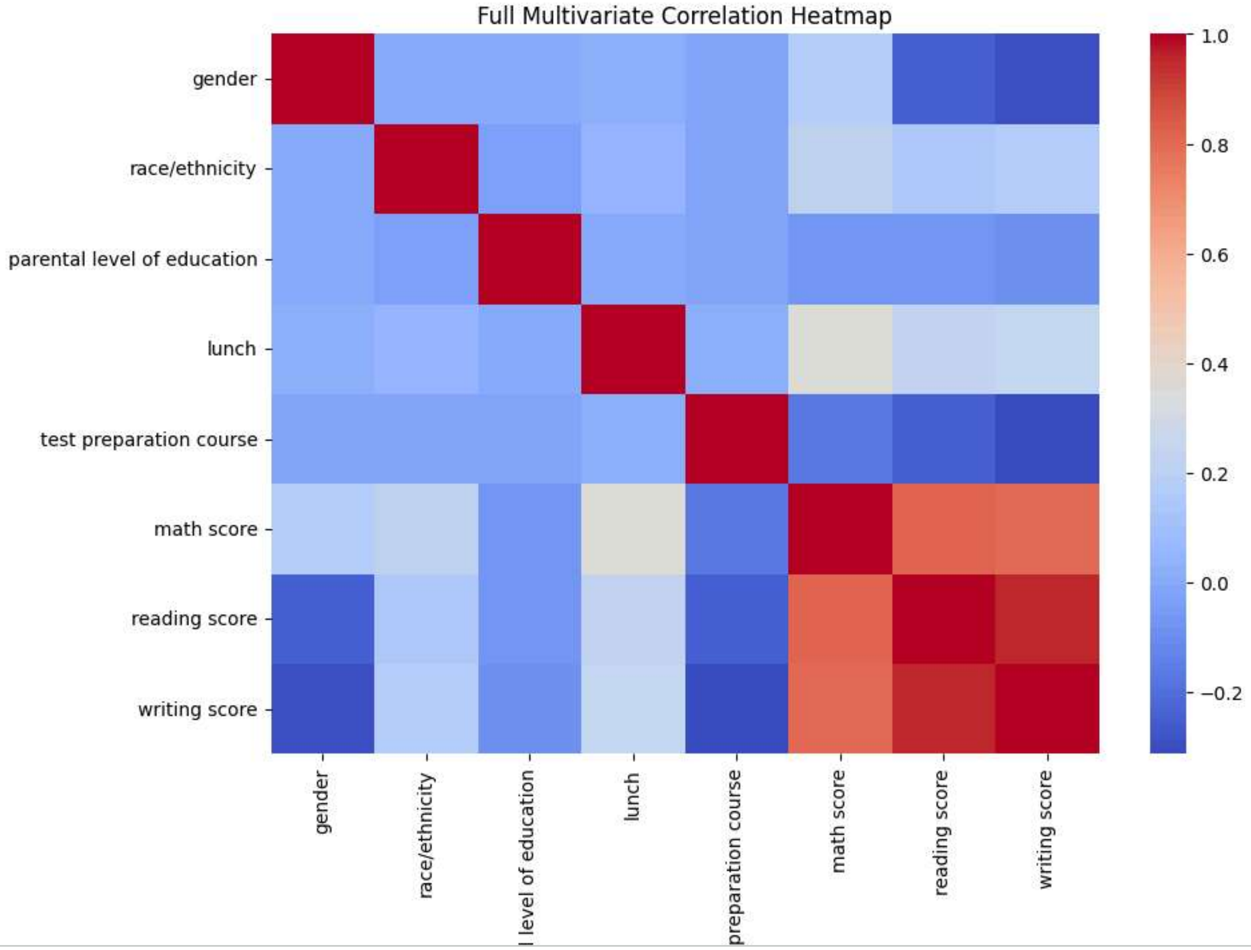




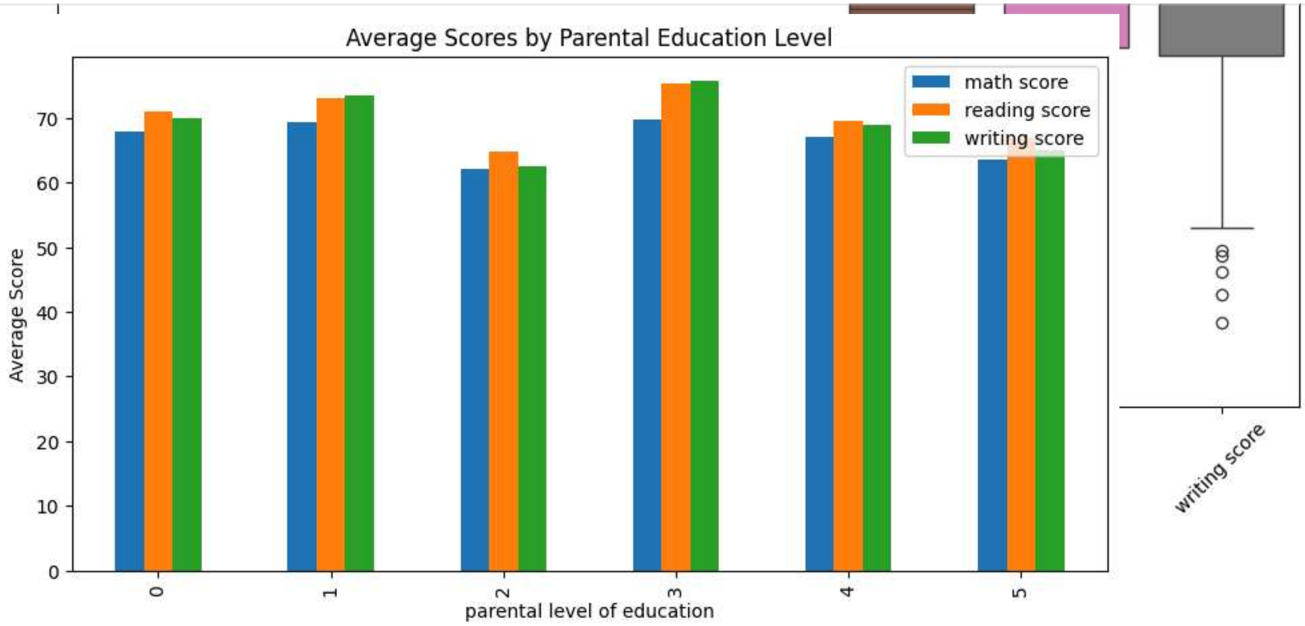

race/ethnicity -0.00 1.00 -0.03 0.05 -0.02 0.22 0.15 0.17
parental level of education 0.00 -0.03 1.00 0.01 -0.02 -0.07 -0.07 -0.08
lunch 0.02 0.05 0.01 1.00 0.02 0.35 0.23 0.25
test preparation course -0.01 -0.02 -0.02 0.02 1.00 -0.18 -0.24 -0.31
math score 0.17 0.22 -0.07 0.35 -0.18 1.00 0.82 0.80
reading score -0.24 0.15 -0.07 0.23 -0.24 0.82 1.00 0.95
writing score -0.30 0.17 -0.08 0.25 -0.31 0.80 0.95 1.00
gender
race/ethnicity
parental level of education
lunch
test preparation course
math score
reading score
writing score
0.8
0.6
0.4
0.2
0.0
−0.2


Pairplot of Numerical Variables
gender
race/ethnicity
parental level of education
lunch
test preparation course
math score
score
1.0
0.8
0.6
0.4
0.2
0.0
4
3
2
1
0
5
100
80
60
40
20


reading s
60
40
20
writing score
100
80
60
40
20
0.00 0.25 0.50 0.75 1.00
gender
0 1 2 3 4
race/ethnicity
0 2 4
parental level of education
0.00 0.25 0.50 0.75 1.00
lunch
0.00 0.25 0.50 0.75 1.00
test preparation course
0 25 50 75 100
math score
20 40 60 80 100
reading score
20 40 60 80 100
writing score


Full Multivariate Correlation Heatmap
gender
race/ethnicity
parental level of education
lunch
test preparation course
math score
reading score
writing score
gender
race/ethnicity
l level of education
lunch
preparation course
math score
reading score
writing score
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

```
# Step 5e: Average scores by parental level of education
avg_scores = df.groupby('parental level of education')[['math score', 'reading score', 'writing score']].mean()
avg_scores.plot(kind='bar', figsize=(10,5))
plt.title("Average Scores by Parental Education Level")
plt.ylabel("Average Score")
plt.show()
```



```
# Step 6: Encode categorical variables using LabelEncoder
cat_cols = df.select_dtypes(include=['object']).columns
le = LabelEncoder()

for col in cat_cols:
    df[col] = le.fit_transform(df[col])

print("After Encoding:")
display(df.head())
```

```
After Encoding:
```

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	0	1	1	1	1	72	72	74
1	0	2	4	1	0	69	90	88
2	0	1	3	1	1	90	95	93
3	1	0	0	0	1	47	57	44
4	1	2	4	1	1	76	78	75

```
# Step 7: Define X (features) and y (target)
```

```
    X = df.drop("writing score", axis=1)
    y = df["writing score"]

    print("Feature Columns:", X.columns.tolist())
    print("Shape of X:", X.shape)
```