```
!pip install matplotlib
!pip install numpy
!pip install seaborn
!pip install pandas
```

```
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages (3.10.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplo
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matpl
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matpl
Requirement already satisfied: numpy>=1.23 in /usr/local/lib/python3.11/dist-packages (from matplotlib)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from matplot
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplo
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.11/dist-packages (from ma
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateuti
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (2.0.2)
Requirement already satisfied: seaborn in /usr/local/lib/python3.11/dist-packages (0.13.2)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in /usr/local/lib/python3.11/dist-packages (from se
Requirement already satisfied: pandas>=1.2 in /usr/local/lib/python3.11/dist-packages (from seaborn) (2
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in /usr/local/lib/python3.11/dist-packages (from
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplo
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matpl
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matpl
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from matplot
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages (from matplotlib!=3
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplo
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.11/dist-packages (from ma
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas>=
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateuti
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (2.2.2)
Requirement already satisfied: numpy>=1.23.2 in /usr/local/lib/python3.11/dist-packages (from pandas) (
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas) (2
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateuti
```

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


passmark = 40


import pandas as pd
df = pd.read_csv("/content/StudentsPerformance.csv")
df.head()
```

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

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
print (df.shape)
```

```
(1000, 12)
```

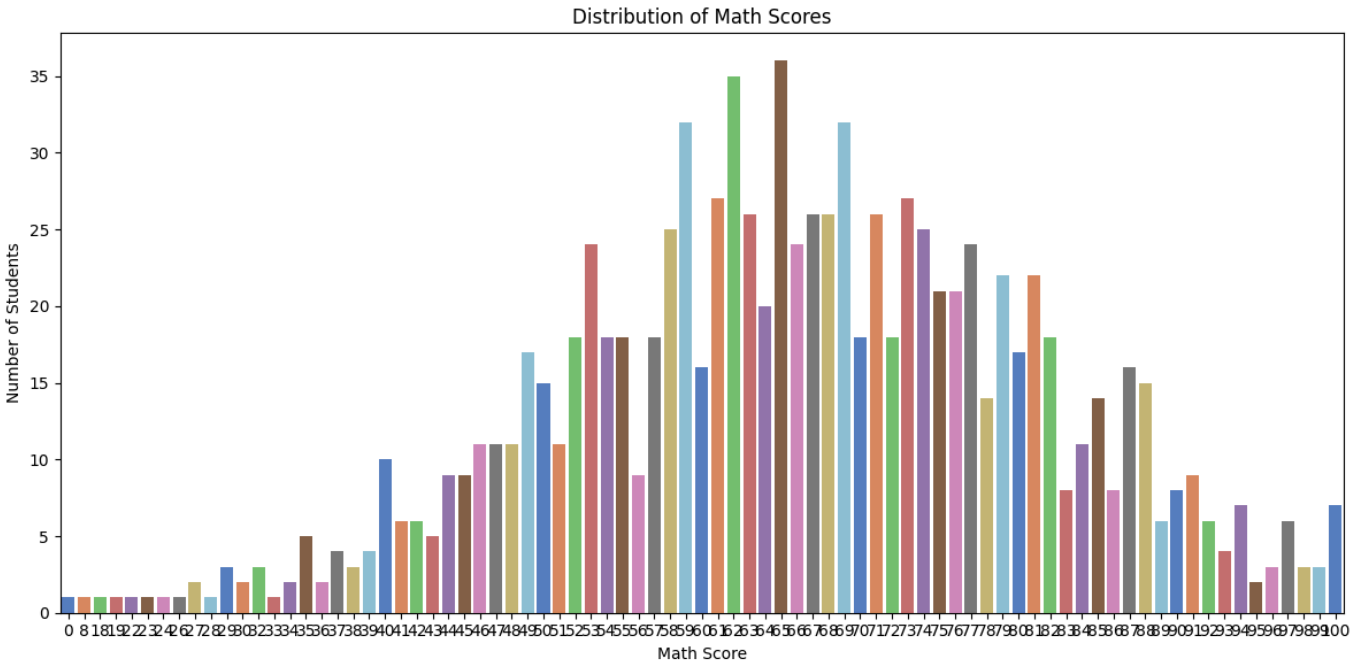
```
df.describe()
```

	math score	reading score	writing score
count	1000.00000	1000.000000	1000.000000
mean	66.08900	69.169000	68.054000
std	15.16308	14.600192	15.195657
min	0.00000	17.000000	10.000000
25%	57.00000	59.000000	57.750000
50%	66.00000	70.000000	69.000000
75%	77.00000	79.000000	79.000000

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

	0
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
Math_PassStatus	0
Reading_PassStatus	0
Writing_PassStatus	0
OverAll_PassStatus	0

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

# Load your dataset
df = pd.read_csv("/content/StudentsPerformance.csv")  # Ensure this file path is correct

# Create a count plot for math scores with updated usage of hue
plt.figure(figsize=(12, 6))  # Adjust the figure size for better visibility
p = sns.countplot(x="math score", data=df, hue="math score", palette="muted", legend=False)

# Set labels and title
plt.xlabel('Math Score')
plt.ylabel('Number of Students')
plt.title('Distribution of Math Scores')

# Show the plot
plt.tight_layout()  # Adjust layout to fit labels
plt.show()
```

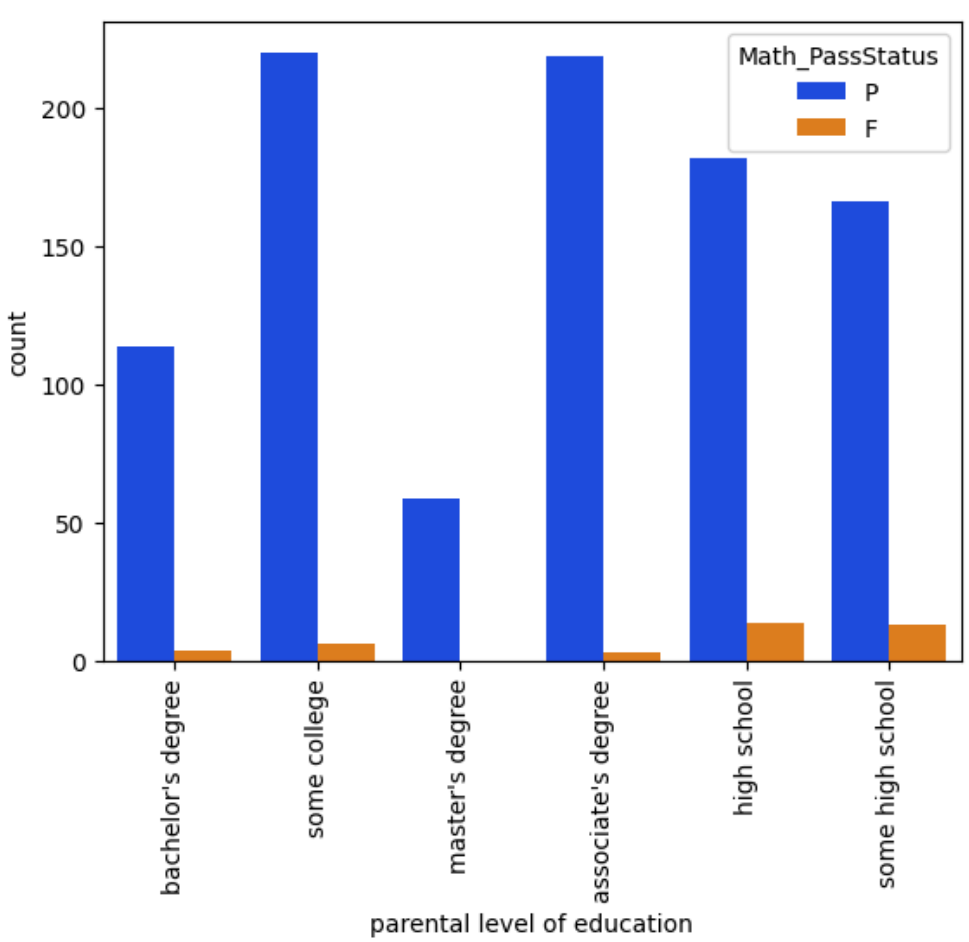



```
passmark = 40
df['Math_PassStatus'] = np.where(df['math score']<passmark, 'F', 'P')
df.Math_PassStatus.value_counts()
```

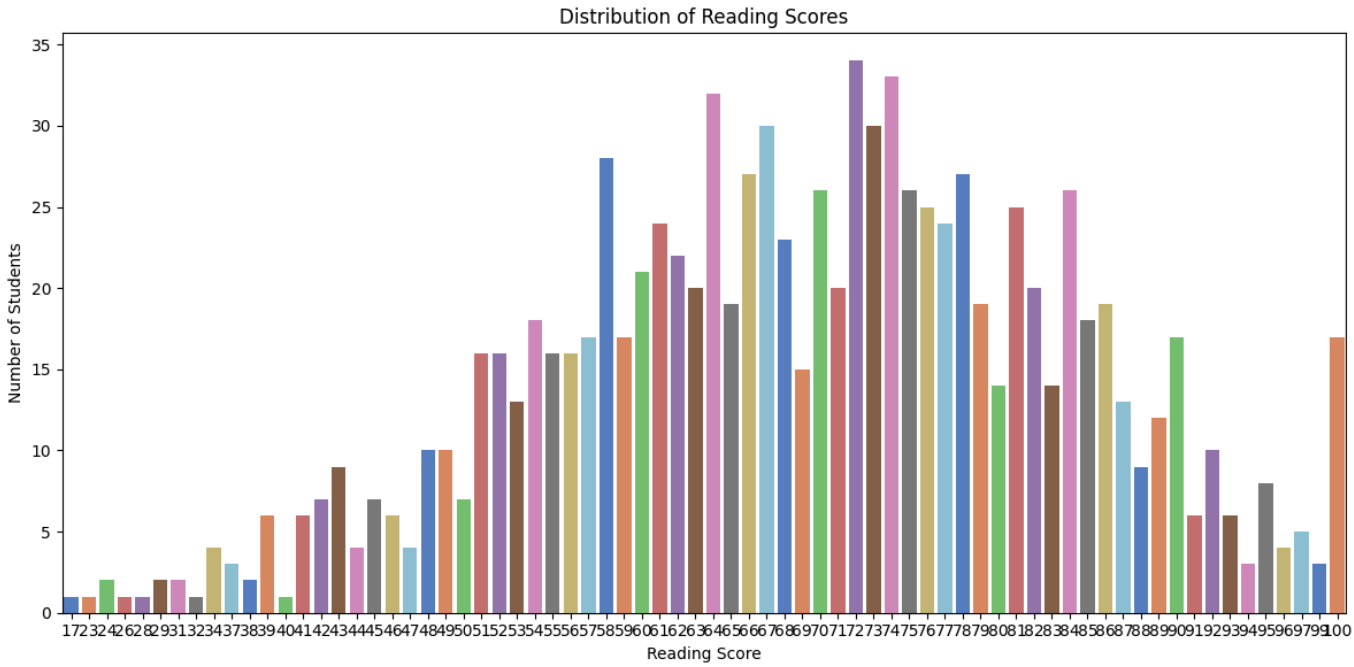
Math_PassStatus	count
P	960
F	40

dtype: int64

```
p = sns.countplot(x='parental level of education', data = df, hue='Math_PassStatus', palette='bright')
_ = plt.setp(p.get_xticklabels(), rotation=90)
```




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

# Load your dataset
df = pd.read_csv("/content/StudentsPerformance.csv")  # Ensure this file path is correct

# Create a count plot for reading scores with updated usage of hue
plt.figure(figsize=(12, 6))  # Adjust the figure size for better visibility
p = sns.countplot(x="reading score", data=df, hue="reading score", palette="muted", legend=False)

# Set labels and title
plt.xlabel('Reading Score')
plt.ylabel('Number of Students')
plt.title('Distribution of Reading Scores')

# Show the plot
plt.tight_layout()  # Adjust layout to fit labels
plt.show()
```

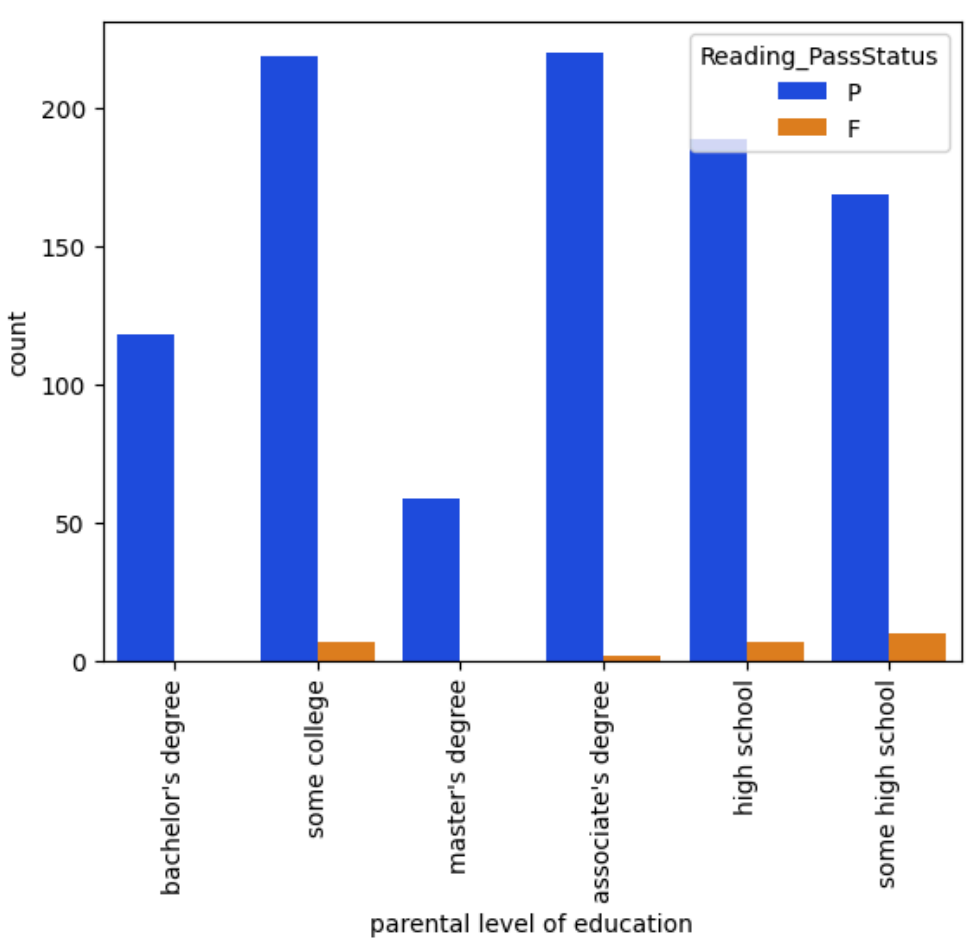



```
df['Reading_PassStatus'] = np.where(df['reading score']<passmark, 'F', 'P')
df.Reading_PassStatus.value_counts()
```

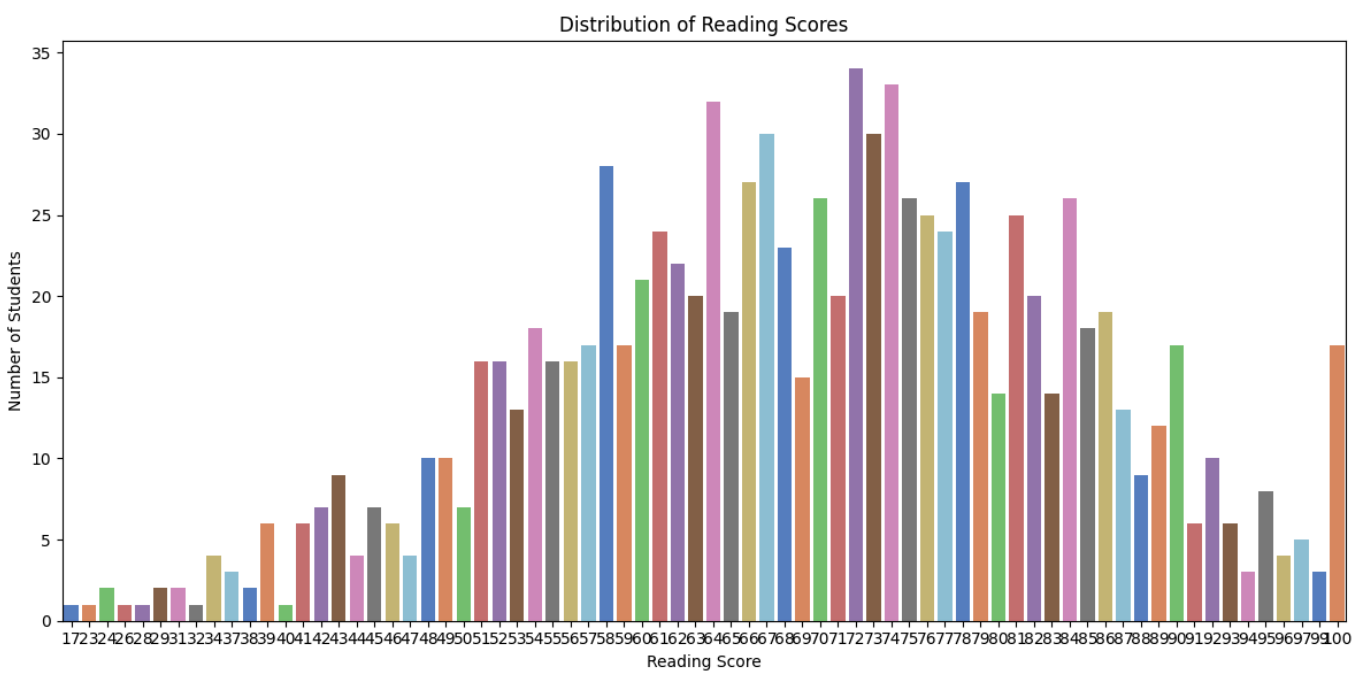
Reading_PassStatus	count
P	974
F	26

dtype: int64

```
p = sns.countplot(x='parental level of education', data = df, hue='Reading_PassStatus', palette='bright')
_ = plt.setp(p.get_xticklabels(), rotation=90)
```




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

# Load your dataset
df = pd.read_csv("/content/StudentsPerformance.csv")  # Ensure this file path is correct

# Create a count plot for reading scores with updated usage of hue
plt.figure(figsize=(12, 6))  # Adjust the figure size for better visibility
p = sns.countplot(x="reading score", data=df, hue="reading score", palette="muted", legend=False)

# Set labels and title
plt.xlabel('Reading Score')
plt.ylabel('Number of Students')
plt.title('Distribution of Reading Scores')

# Show the plot
plt.tight_layout()  # Adjust layout to fit labels
plt.show()
```

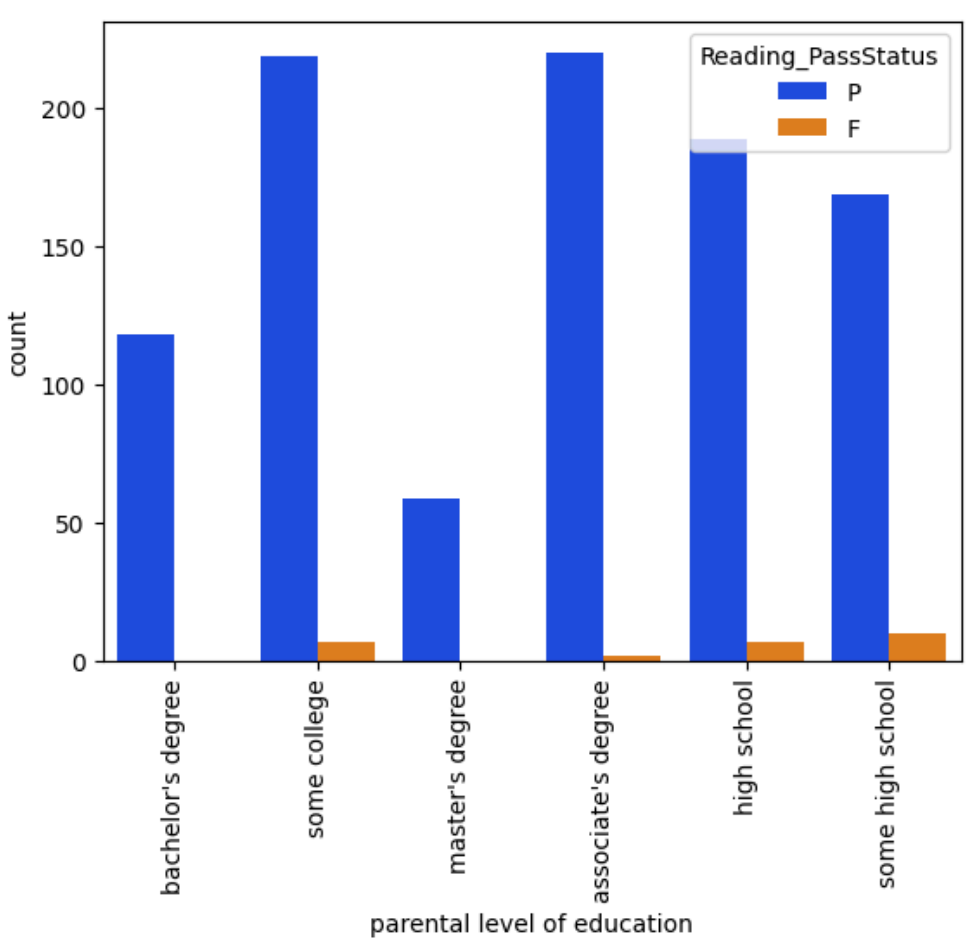



```
df['Reading_PassStatus'] = np.where(df['reading score']<passmark, 'F', 'P')
df.Reading_PassStatus.value_counts()
```

Reading_PassStatus	count
P	974
F	26

dtype: int64

```
p = sns.countplot(x='parental level of education', data = df, hue='Reading_PassStatus', palette='bright')
_ = plt.setp(p.get_xticklabels(), rotation=90)
```



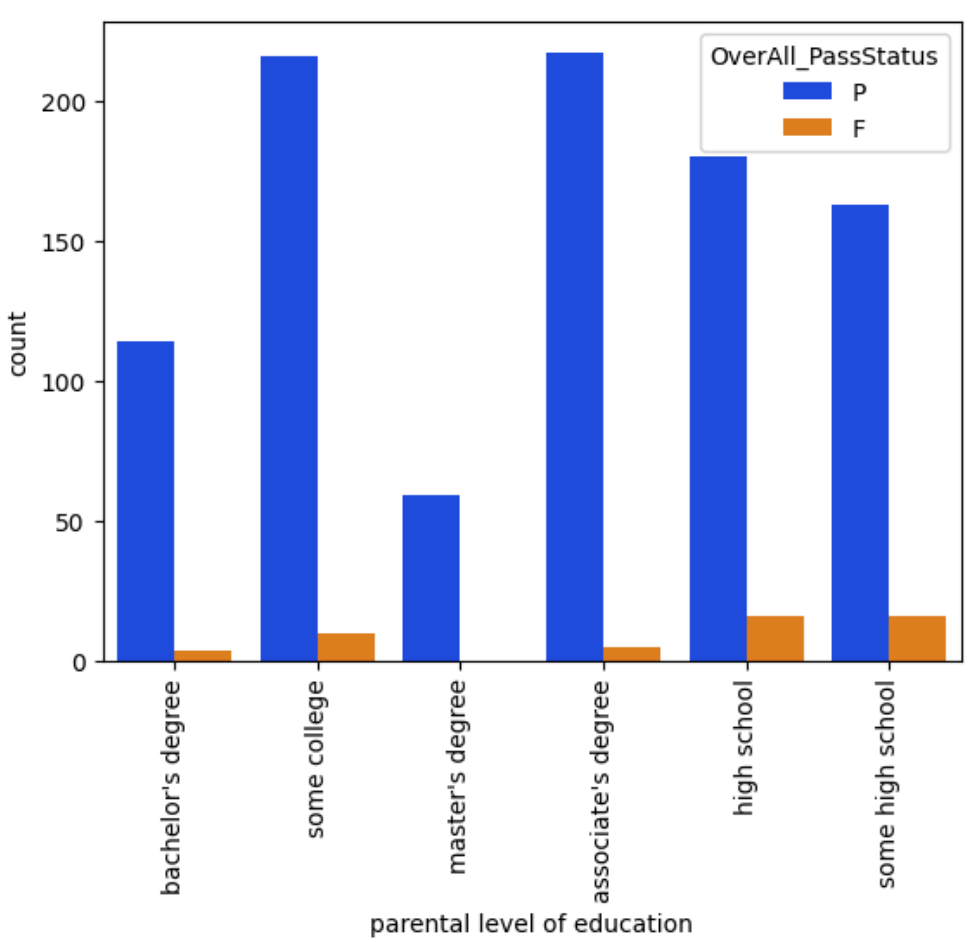

```
df['OverAll_PassStatus'] = df.apply(lambda x : 'F' if x['Math_PassStatus'] == 'F' or
                                    x['Reading_PassStatus'] == 'F' or x['Writing_PassStatus'] == 'F' else '
```

```
df.OverAll_PassStatus.value_counts()
```

OverAll_PassStatus	count
P	949
F	51

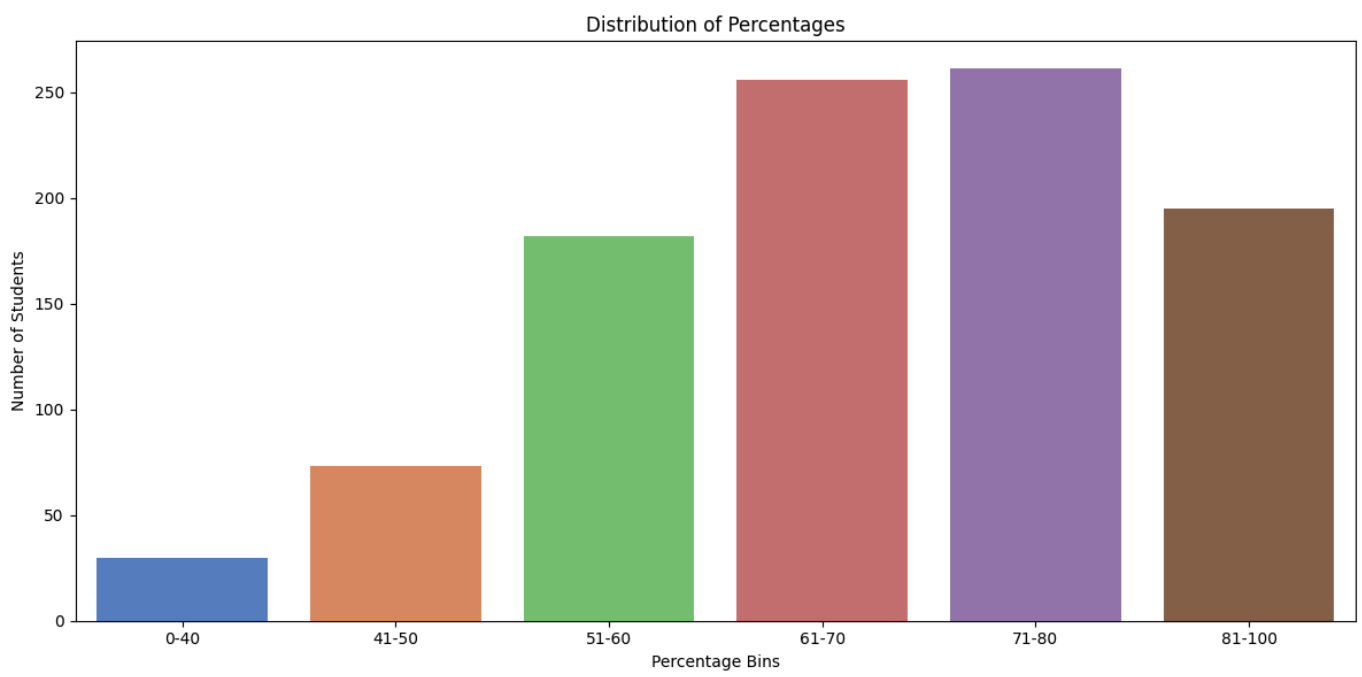
dtype: int64

```
p = sns.countplot(x='parental level of education', data = df, hue='OverAll_PassStatus', palette='bright')
_ = plt.setp(p.get_xticklabels(), rotation=90)
```




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

# Load your dataset
df = pd.read_csv("/content/StudentsPerformance.csv")  # Ensure this file path is correct

# Calculate total marks and percentage
df['Total_Marks'] = df['math score'] + df['reading score'] + df['writing score']
df['Percentage'] = df['Total_Marks'] / 3

# Bin the percentages for better visualization
bins = [0, 40, 50, 60, 70, 80, 100]  # Define bins for percentages
labels = ['0-40', '41-50', '51-60', '61-70', '71-80', '81-100']  # Labels for the bins
df['Percentage_Binned'] = pd.cut(df['Percentage'], bins=bins, labels=labels, right=False)

# Create a count plot for binned percentages with updated usage of hue
plt.figure(figsize=(12, 6))  # Adjust the figure size for better visibility
p = sns.countplot(x="Percentage_Binned", data=df, hue="Percentage_Binned", palette="muted", legend=False)

# Set labels and title
plt.xlabel('Percentage Bins')
plt.ylabel('Number of Students')
plt.title('Distribution of Percentages')

# Show the plot
plt.tight_layout()  # Adjust layout to fit labels
```

```
plt.show()
```

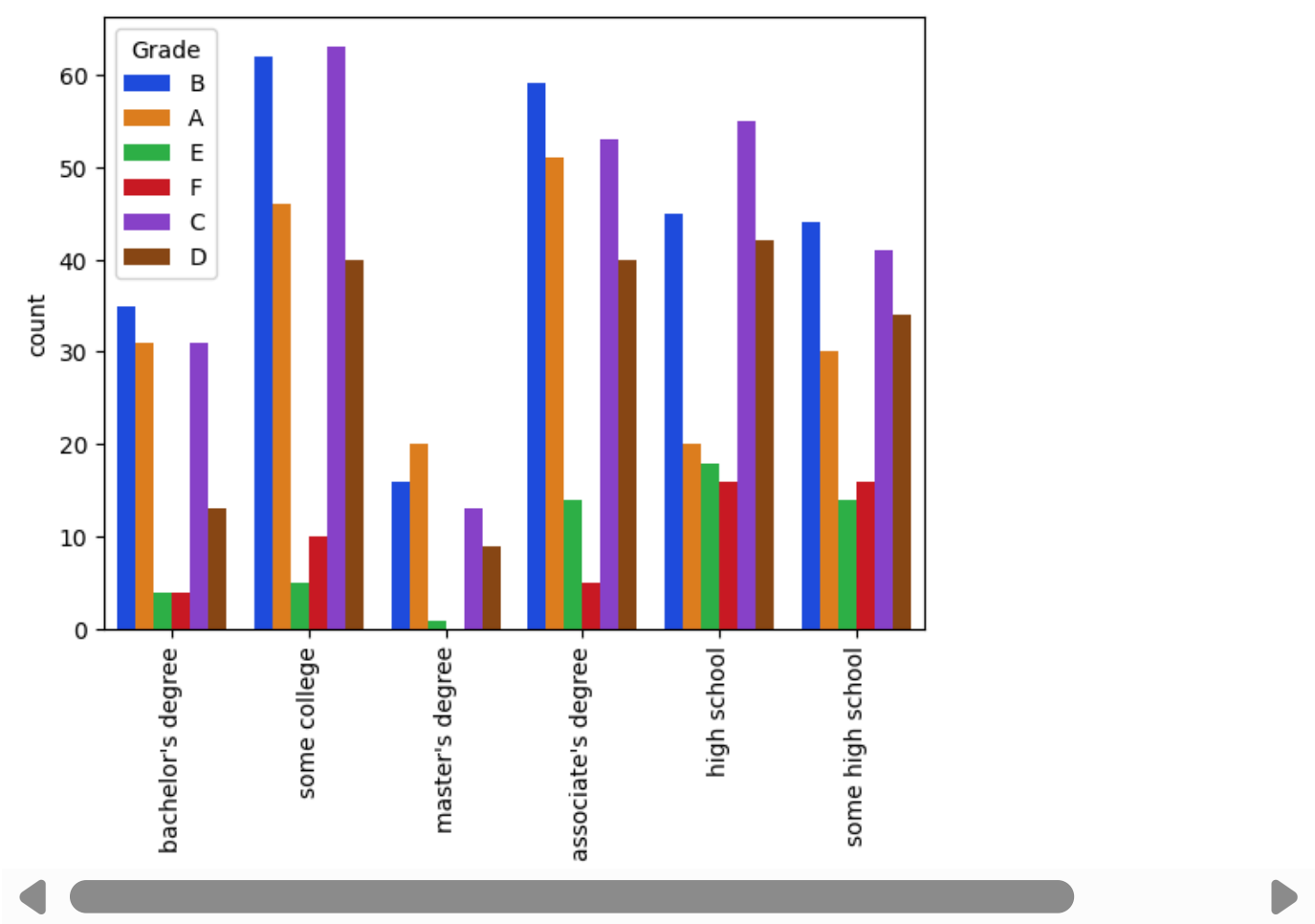



```
def GetGrade(Percentage, OverAll_PassStatus):
    if ( OverAll_PassStatus == 'F'):
        return 'F'
    if ( Percentage >= 80 ):
        return 'A'
    if ( Percentage >= 70):
        return 'B'
    if ( Percentage >= 60):
        return 'C'
    if ( Percentage >= 50):
        return 'D'
    if ( Percentage >= 40):
        return 'E'
    else:
        return 'F'
df['Grade'] = df.apply(lambda x : GetGrade(x['Percentage'], x['OverAll_PassStatus']), axis=1)

df.Grade.value_counts()
```

Grade	count
B	261
C	256
A	198
D	178
E	56
F	51

dtype: int64

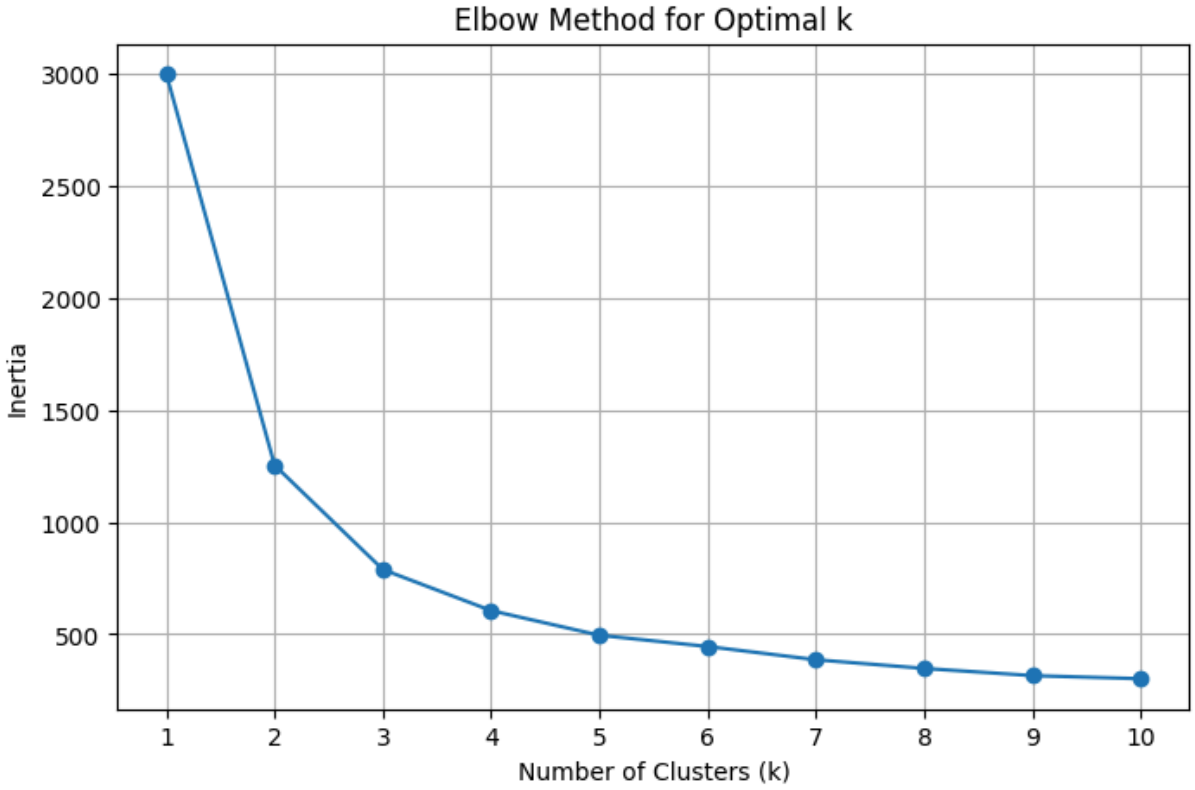
```
p = sns.countplot(x='parental level of education', data = df, hue='Grade', palette='bright')
_ = plt.setp(p.get_xticklabels(), rotation=90)
```




```
!pip install scikit-learn
```

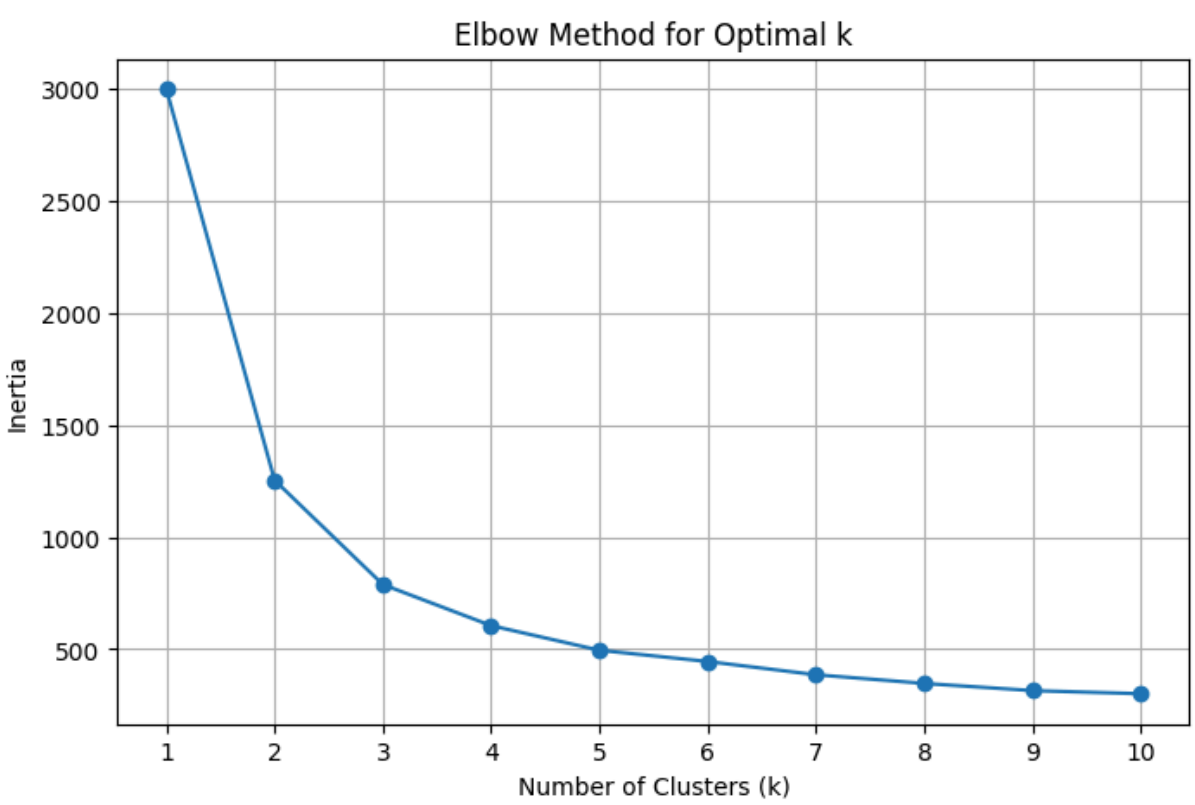
```
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packages (1.6.1)
Requirement already satisfied: numpy>=1.19.5 in /usr/local/lib/python3.11/dist-packages (from scikit-le
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from scikit-lea
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-le
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from sc
```

```
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler


import pandas as pd
df = pd.read_csv("/content/StudentsPerformance.csv")
df.head()
# Selecting features for clustering
features = df[['math score', 'reading score', 'writing score']]

# Standardizing the features
scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)


import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

# Load the dataset
df = pd.read_csv("/content/StudentsPerformance.csv")

# Select features for clustering
features = df[['math score', 'reading score', 'writing score']]

# Standardizing the features
scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)

# Calculate the inertia for different numbers of clusters
inertia = []
k_values = range(1, 11)

for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(features_scaled)
    inertia.append(kmeans.inertia_)

# Plotting the elbow graph
plt.figure(figsize=(8, 5))
plt.plot(k_values, inertia, marker='o')
plt.title('Elbow Method for Optimal k')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Inertia')
plt.xticks(k_values)
plt.grid()
plt.show()
```




```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

# Load the dataset
df = pd.read_csv("/content/StudentsPerformance.csv")

# Select features for clustering
features = df[['math score', 'reading score', 'writing score']]

# Standardizing the features
scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)

# Calculate the inertia for different numbers of clusters
inertia = []
k_values = range(1, 11)

for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(features_scaled)
    inertia.append(kmeans.inertia_)

# Plotting the elbow graph
plt.figure(figsize=(8, 5))
plt.plot(k_values, inertia, marker='o')
plt.title('Elbow Method for Optimal k')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Inertia')
plt.xticks(k_values)
```

```
plt.grid()
plt.show()

# Fit K-Means with the chosen number of clusters
k = 3  # Change this based on your elbow plot
kmeans = KMeans(n_clusters=k, random_state=42)
df['Cluster'] = kmeans.fit_predict(features_scaled)

# Output the clustered DataFrame
print(df[['math score', 'reading score', 'writing score', 'Cluster']].head())
```




```
   math score  reading score  writing score  Cluster
0          72             72             74        0
1          69             90             88        2
2          90             95             93        2
3          47             57             44        1
4          76             78             75        2
```

```
import pandas as pd
import numpy as np
import seaborn as sns  # Ensure this line is included
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

# Load your dataset (assuming you've done this already)
# df = pd.read_csv("StudentsPerformance (2).csv")

# Example: Selecting features and scaling
features = df[['math score', 'reading score', 'writing score']]
scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)
```

```
# Fit K-Means clustering (example for k=3)
k = 3  # You can adjust this based on your elbow method results
kmeans = KMeans(n_clusters=k, random_state=42)
df['Cluster'] = kmeans.fit_predict(features_scaled)

# Plotting the clusters
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='math score', y='reading score', hue='Cluster', palette='deep', style='Cluster',
plt.title('K-Means Clustering of Students by Math and Reading Scores')
plt.xlabel('Math Score')
plt.ylabel('Reading Score')
plt.legend(title='Cluster')
plt.show()
```