In [63]:

```
import numpy as np # linear algebra
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [64]:

```
df = pd.read_csv(r'C:/Users/jerusha josine/Documents/studentdataset/StudentsPerformance.csv')
df.head(20)
```

Out[64]:

	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
5	female	group B	associate's degree	standard	none	71	83	78
6	female	group B	some college	standard	completed	88	95	92
7	male	group B	some college	free/reduced	none	40	43	39
8	male	group D	high school	free/reduced	completed	64	64	67
9	female	group B	high school	free/reduced	none	38	60	50
10	male	group C	associate's degree	standard	none	58	54	52
11	male	group D	associate's degree	standard	none	40	52	43
12	female	group B	high school	standard	none	65	81	73
13	male	group A	some college	standard	completed	78	72	70
14	female	group A	master's degree	standard	none	50	53	58
15	female	group C	some high school	standard	none	69	75	78
16	male	group C	high school	standard	none	88	89	86
17	female	group B	some high school	free/reduced	none	18	32	28
18	male	group C	master's degree	free/reduced	completed	46	42	46
19	female	group C	associate's degree	free/reduced	none	54	58	61

In [65]:

```
df.shape # 1000 rows and 8 columns
```

Out[65]:

```
(1000, 8)
```

In [66]:

```
df.describe()
```

Out[66]:

	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
max	100.00000	100.000000	100.000000

In [67]:

```
# to check the score is add the 3 columns and divide by 3
df["mean score"] = ((df["math score"] + df["reading score"] + df["writing score"]) / 3).round()
df.head()
```

Out[67]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score	mean score
0	female	group B	bachelor's degree	standard	none	72	72	74	73.0
1	female	group C	some college	standard	completed	69	90	88	82.0
2	female	group B	master's degree	standard	none	90	95	93	93.0
3	male	group A	associate's degree	free/reduced	none	47	57	44	49.0
4	male	group C	some college	standard	none	76	78	75	76.0

In [68]:

```
# find the no. of males and females in the class
df['gender'].value_counts()
```

```
Out[68]: gender
         female    518
         male      482
         Name: count, dtype: int64
```

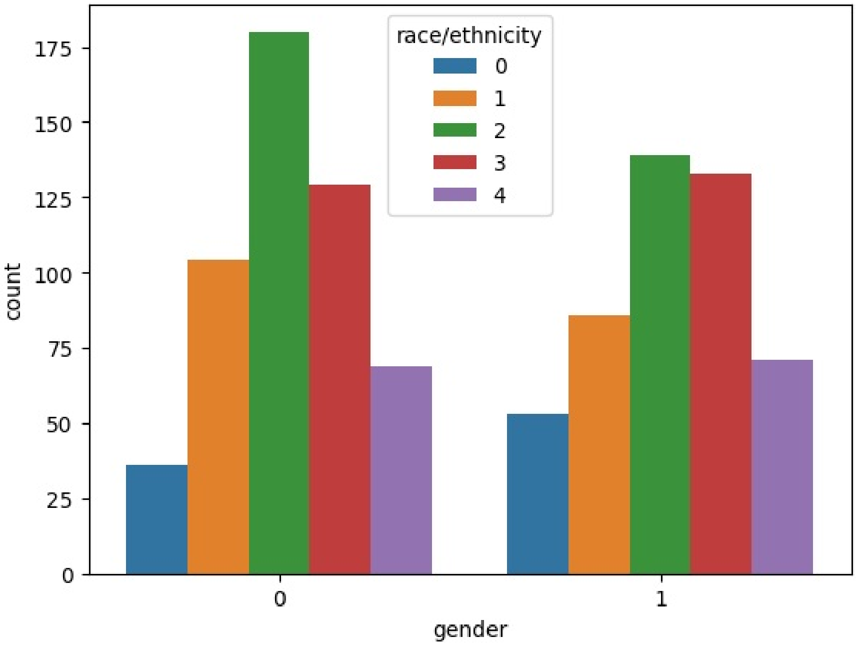
```
In [69]: # label Encoding
         # converting the string values into numeric form to understand the data

         from sklearn.preprocessing import LabelEncoder
         lc = LabelEncoder() # use a variable to fit and transform the data
         df['gender'] = lc.fit_transform(df['gender'])
         df['race/ethnicity'] = lc.fit_transform(df['race/ethnicity'])
         df['parental level of education'] = lc.fit_transform(df['parental level of education'])
         df['lunch'] = lc.fit_transform(df['lunch'])
         df['test preparation course'] = lc.fit_transform(df['test preparation course'])
         df.head(20)
```

Out[69]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score	mean score
0	0	1	1	1	1	72	72	74	73.0
1	0	2	4	1	0	69	90	88	82.0
2	0	1	3	1	1	90	95	93	93.0
3	1	0	0	0	1	47	57	44	49.0
4	1	2	4	1	1	76	78	75	76.0
5	0	1	0	1	1	71	83	78	77.0
6	0	1	4	1	0	88	95	92	92.0
7	1	1	4	0	1	40	43	39	41.0
8	1	3	2	0	0	64	64	67	65.0
9	0	1	2	0	1	38	60	50	49.0
10	1	2	0	1	1	58	54	52	55.0
11	1	3	0	1	1	40	52	43	45.0
12	0	1	2	1	1	65	81	73	73.0
13	1	0	4	1	0	78	72	70	73.0
14	0	0	3	1	1	50	53	58	54.0
15	0	2	5	1	1	69	75	78	74.0
16	1	2	2	1	1	88	89	86	88.0
17	0	1	5	0	1	18	32	28	26.0
18	1	2	3	0	0	46	42	46	45.0
19	0	2	0	0	1	54	58	61	58.0

```
In [70]: # Analyzing the Gender and Race
         sns.countplot(x=df['gender'], hue = df['race/ethnicity'])
         # it differentiate the data based on the race of the students
```

```
Out[70]: <Axes: xlabel='gender', ylabel='count'>
```



```
In [71]: # Analysing Test Preparation
```

```
df['test preparation course'].value_counts()
# 1 - None
# 0 - Completed
```

Out[71]:

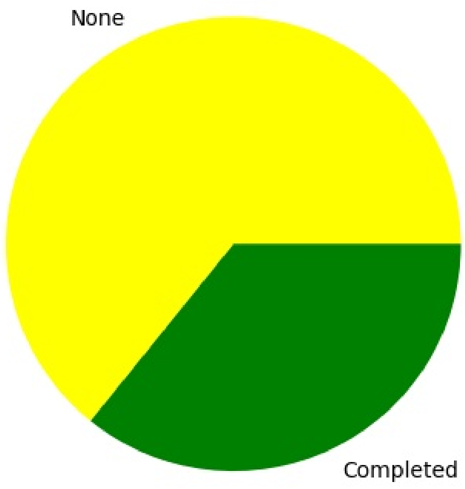
```
test preparation course
1    642
0    358
Name: count, dtype: int64
```

In [72]:

```
labels = ['None', 'Completed'] # pie chart
colors = ['yellow', 'green']
plt.pie(df['test preparation course'].value_counts() , labels = labels, colors = colors)
```

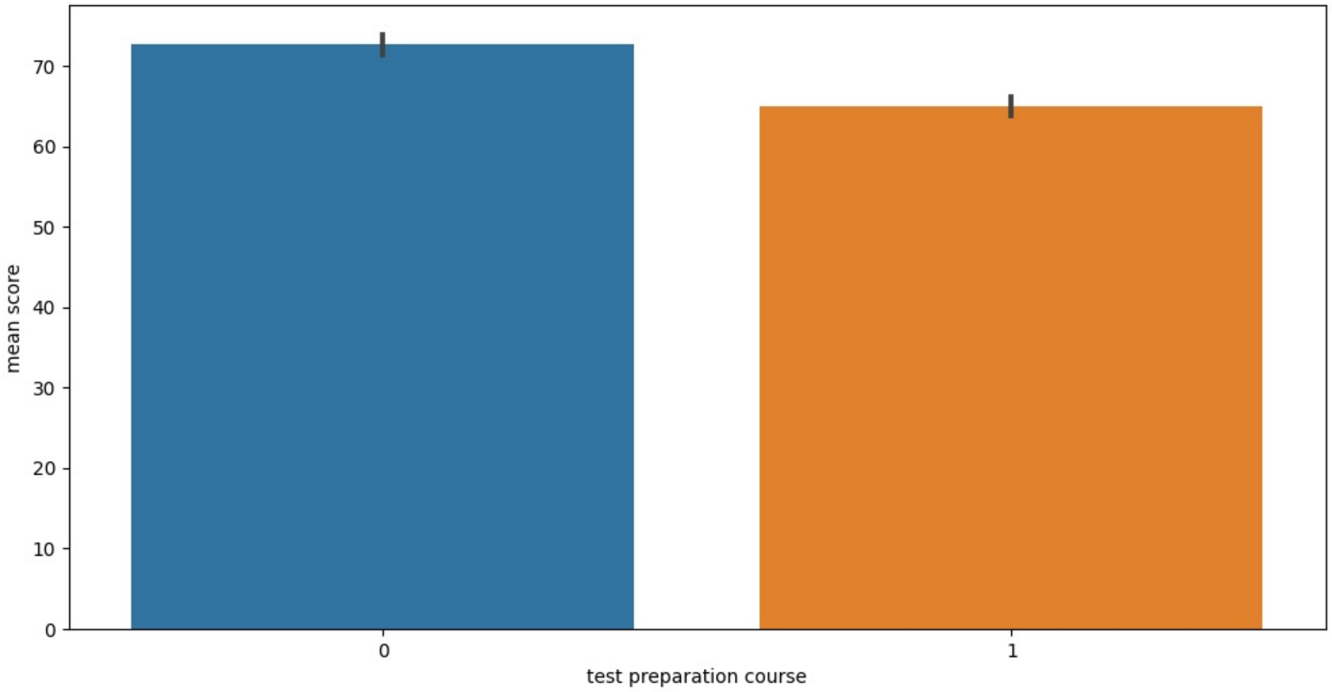
Out[72]:

```
([<matplotlib.patches.Wedge at 0x2801742a590>,
  <matplotlib.patches.Wedge at 0x2801733abd0>],
 [Text(-0.47460171119818767, 0.9923473261553901, 'None'),
  Text(0.4746018041084478, -0.9923472817199666, 'Completed')])
```



In [73]:

```
# Assuming you have a pandas DataFrame called "df" with the columns "test preparation course" and "mean score"
plt.figure(figsize=(12, 6))
sns.barplot(x ='test preparation course', y ='mean score', data = df)
plt.show()
# students who are done the score means 0 they are getting more mean score.
# students who are not done the score they are getting less mean score.
```



In [74]:

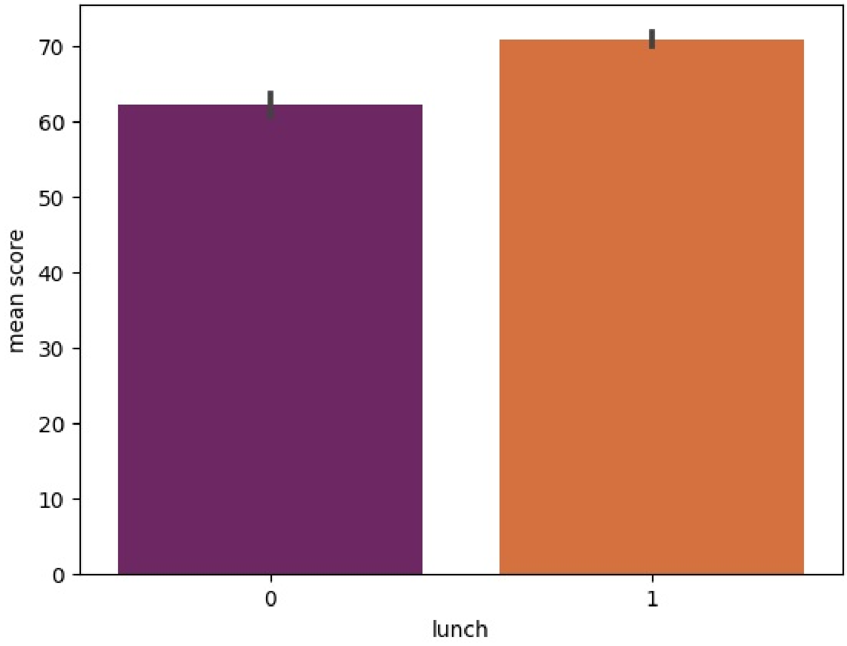
```
# Analyzing Lunch
# some students are depending on school for lunch and some are depending on their homes
```

In [75]:

```
sns.barplot(x = df['lunch'], y = df['mean score'], palette = 'inferno')
# 0 - free lunch
# 1 - Tiffin
```

Out[75]:

```
<Axes: xlabel='lunch', ylabel='mean score'>
```



In [76]:

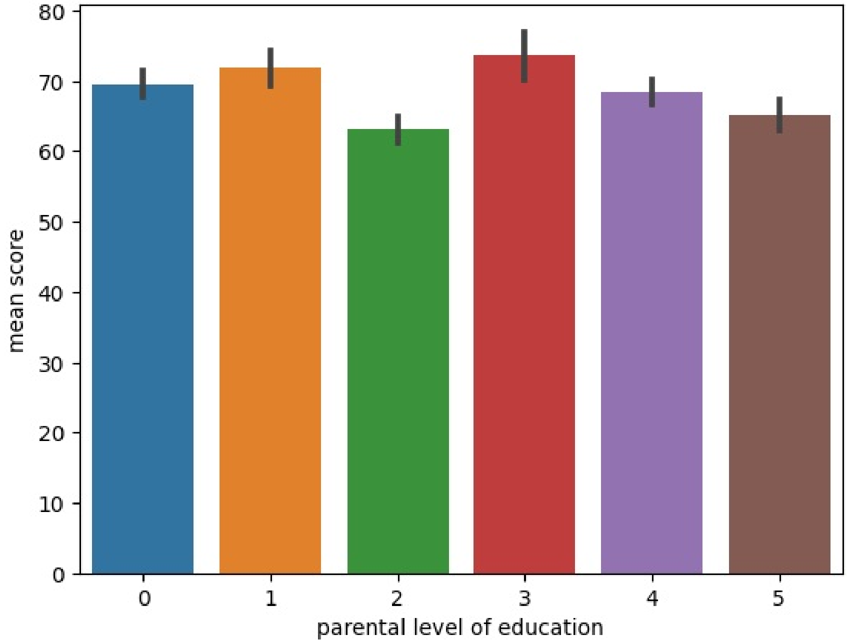
```
# Analysing parental level of education.
# green showing the high score
# check the third column (parental level of education in index 2 )
# which means parents who studied master's degree, their kids are performing good in their education.
```

In [77]:

```
sns.barplot(x = 'parental level of education', y = 'mean score', data = df)
```

Out[77]:

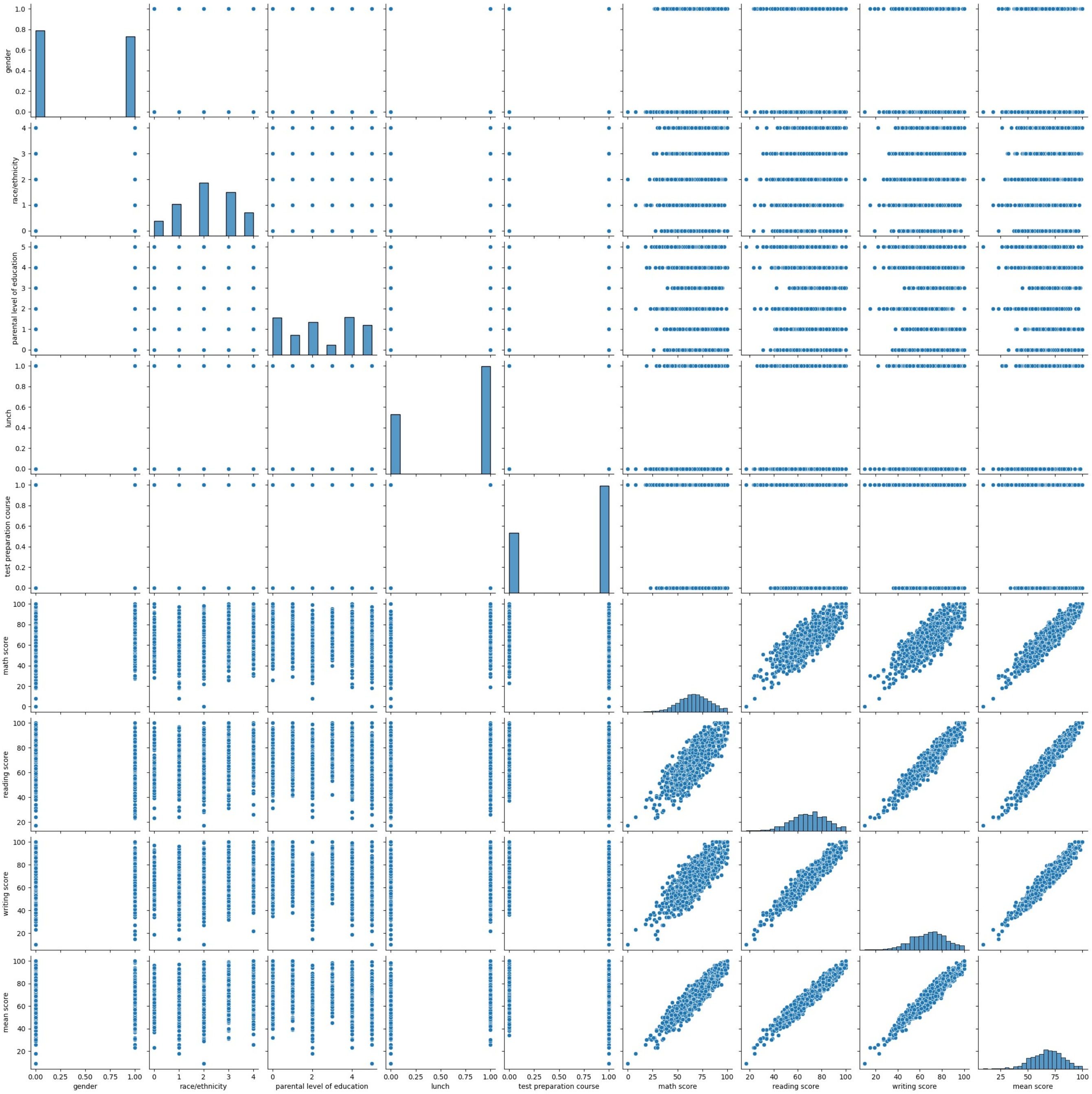
```
<Axes: xlabel='parental level of education', ylabel='mean score'>
```



In [78]:

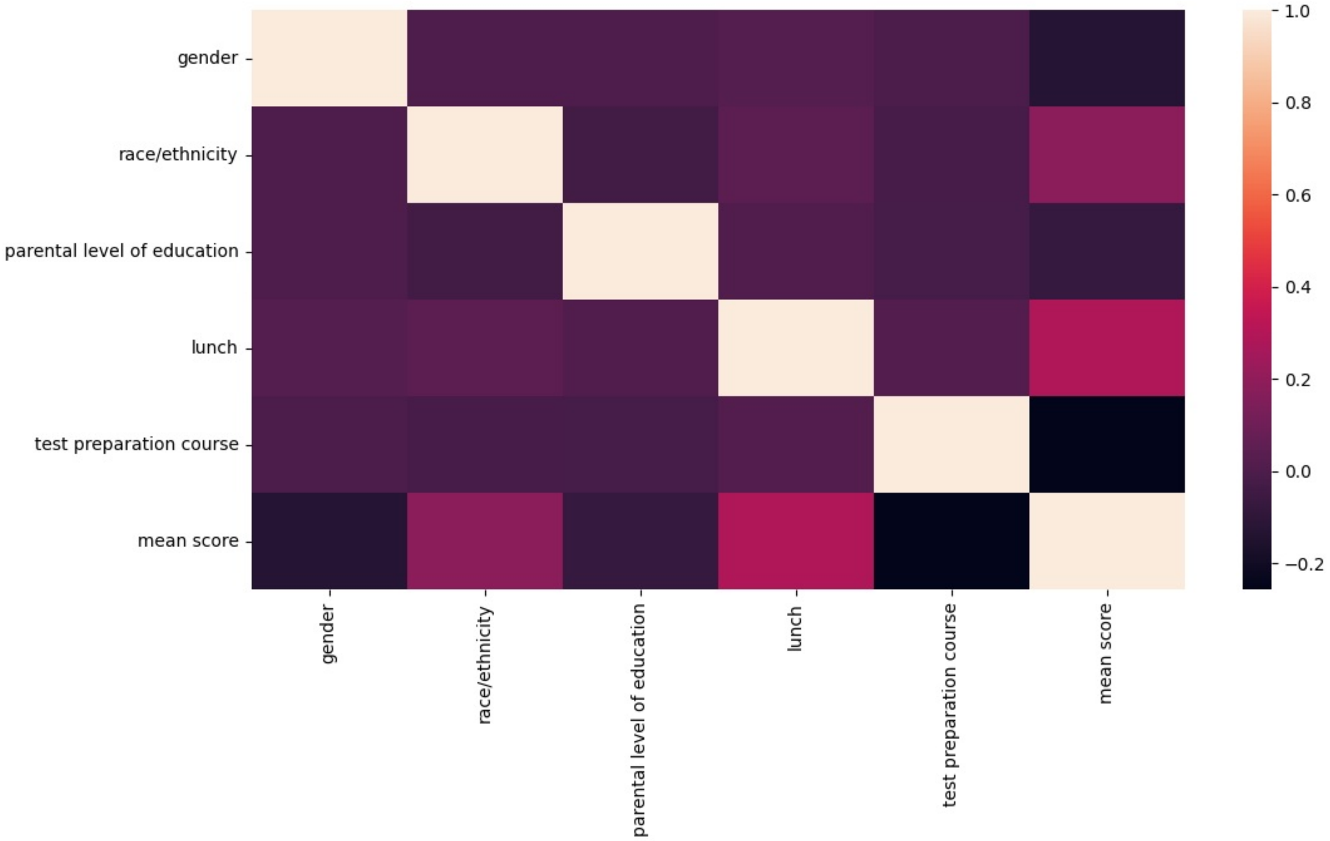
```
plt.figure(figsize = (12,6)) # pair plot
sns.pairplot(df)
plt.show()
```

```
<Figure size 1200x600 with 0 Axes>
```



In [81]:

```
plt.figure(figsize = (12,6)) # apply the heatmap
sns.heatmap(df.corr())
plt.show()
```



In [80]:

```
# data processing
# drop the math score, reading score and writing score
# target class is mean score
df = df.drop(['math score', 'writing score', 'reading score'],axis = 1)
df.head()
```

Out[80]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	mean score
0	0	1	1	1	1	73.0
1	0	2	4	1	0	82.0
2	0	1	3	1	1	93.0
3	1	0	0	0	1	49.0
4	1	2	4	1	1	76.0

In [90]:

```
from sklearn.model_selection import train_test_split
y = df['mean score']
x = df.drop(['mean score'], axis  = 1) # target class
# 80% data going to testing, 20% of data goingto training
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.2, random_state = 0)
```

In [83]:

```
# model building
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix
```

In [84]:

```
model = LogisticRegression(solver='liblinear', random_state=0)
```

In [85]:

```
model.fit(x_train, y_train)
```

Out[85]:

```
▾                LogisticRegression
LogisticRegression(random_state=0, solver='liblinear')
```

In [86]:

```
predictions = model.predict(x_test) # predit the value for the value for the testing
```

In [91]:

```
predictions
```

```
Out[91]: array([69., 56., 56., 59., 73., 76., 67., 71., 76., 69., 73., 54., 69.,
                71., 76., 54., 68., 74., 62., 49., 73., 69., 67., 68., 71., 56.,
                55., 68., 54., 74., 59., 76., 67., 73., 76., 71., 74., 68., 92.,
                69., 73., 56., 68., 75., 65., 92., 73., 65., 74., 54., 71., 55.,
                65., 68., 71., 69., 69., 69., 76., 69., 71., 74., 76., 71., 73.,
                58., 69., 73., 76., 68., 71., 71., 71., 75., 71., 71., 69., 69.,
                73., 73., 76., 69., 87., 73., 79., 69., 71., 92., 76., 54., 54.,
                73., 54., 55., 69., 68., 59., 54., 56., 68., 76., 71., 61., 50.,
                71., 75., 76., 65., 69., 79., 74., 75., 69., 59., 74., 74., 76.,
                59., 56., 76., 69., 65., 74., 68., 71., 76., 73., 76., 76., 74.,
                79., 73., 59., 69., 76., 69., 71., 69., 69., 73., 71., 73., 73.,
                68., 65., 59., 59., 59., 76., 69., 68., 74., 74., 71., 74., 69.,
                71., 73., 69., 68., 74., 69., 59., 71., 76., 73., 76., 68., 69.,
                73., 69., 71., 65., 75., 73., 69., 69., 73., 87., 73., 68., 65.,
                49., 68., 76., 69., 92., 87., 54., 68., 68., 48., 58., 67., 59.,
                59., 76., 73., 92., 61.])
```

```
In [87]: difference = abs(predictions - y_test)
```

```
In [92]: y_test
```

```
Out[92]: 993    69.0
         859    77.0
         298    45.0
         553    68.0
         672    74.0
                ...
         679    61.0
         722    84.0
         215    81.0
         653    70.0
         150    66.0
         Name: mean score, Length: 200, dtype: float64
```

```
In [93]: difference.mean() # find the difference betbeen the original value and predicted values
```

```
Out[93]: 11.03
```

```
In [94]: # Average error is 11.03 marks
         # the model is predicting 60 marks for a student
         # that means the avarage between the errors are either +11 for -11
```

```
In [ ]:
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js