In [24]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix
```

In [25]:
```
df= pd.read_csv("/content/onsite-DataModeling_nilaiSiswa_UTS.csv")
df
```

Out[25]:

	gender	race	parental level of education	lunch	test preparation course	math score	reading score	writing score	total_score	result
0	female	group B	bachelor's degree	standard	none	72	72	74	72.67	PASS
1	female	group C	some college	standard	completed	69	90	88	82.33	PASS
2	female	group B	master's degree	standard	none	90	95	93	92.67	PASS
3	male	group A	associate's degree	free/reduced	none	47	57	44	49.33	FAIL
4	male	group C	some college	standard	none	76	78	75	76.33	PASS
...	...	...	...	...	...	...	...	...	...	...
995	female	group E	master's degree	standard	completed	88	99	95	94.00	NaN
996	male	group C	high school	free/reduced	none	62	55	55	57.33	FAIL
997	female	group C	high school	free/reduced	completed	59	71	65	65.00	PASS
998	female	group D	some college	standard	completed	68	78	77	74.33	NaN
999	female	group D	some college	free/reduced	none	77	86	86	83.00	PASS

1000 rows × 10 columns

In [26]:
```
df.isnull().sum()
```

Out[26]:
```
gender                         0
race                           0
parental level of education    0
lunch                          0
test preparation course        0
math score                     0
reading score                  0
writing score                  0
total_score                    0
result                        61
dtype: int64
```

1. Question 1's answer

In [27]:
```
def get_grade(score):
    if score > 60:
        return 'PASS'
    else:
        return 'FAIL'
```

In [28]:
```
df['result'] = df['result'].fillna(df['total_score'].apply(get_grade))
```

In [29]:
```
df.isnull().sum()
```

Out[29]:
```
gender                         0
race                           0
parental level of education    0
lunch                          0
test preparation course        0
math score                     0
reading score                  0
writing score                  0
total_score                    0
result                         0
dtype: int64
```

In [30]:
```
df
```

Out[30]:

	gender	race	parental level of education	lunch	test preparation course	math score	reading score	writing score	total_score	result
0	female	group B	bachelor's degree	standard	none	72	72	74	72.67	PASS
1	female	group C	some college	standard	completed	69	90	88	82.33	PASS
2	female	group B	master's degree	standard	none	90	95	93	92.67	PASS
3	male	group A	associate's degree	free/reduced	none	47	57	44	49.33	FAIL
4	male	group C	some college	standard	none	76	78	75	76.33	PASS
...	...	...	...	...	...	...	...	...	...	...
995	female	group E	master's degree	standard	completed	88	99	95	94.00	PASS
996	male	group C	high school	free/reduced	none	62	55	55	57.33	FAIL
997	female	group C	high school	free/reduced	completed	59	71	65	65.00	PASS
998	female	group D	some college	standard	completed	68	78	77	74.33	PASS
999	female	group D	some college	free/reduced	none	77	86	86	83.00	PASS

1000 rows × 10 columns

2. Question 2's answer

Compute the mean score for each course

In [31]:
```
math_mean = df['math score'].mean()
reading_mean = df['reading score'].mean()
writing_mean = df['writing score'].mean()
```

Find the highest and lowest mean score of each course

In [32]:
```
math_highest_mean = df['math score'].max()
math_lowest_mean = df['math score'].min()
reading_highest_mean = df['reading score'].max()
reading_lowest_mean = df['reading score'].min()
writing_highest_mean = df['writing score'].max()
writing_lowest_mean = df['writing score'].min()
```

Results

In [33]:
```
print("The average score in math is:", math_mean)
print("The highest mean score in math is:", math_highest_mean)
print("The lowest mean score in math is:", math_lowest_mean)
```

```
The average score in math is: 66.089
The highest mean score in math is: 100
The lowest mean score in math is: 0
```

In [34]:
```
print("The average score in reading is:", reading_mean)
print("The highest mean score in reading is:", reading_highest_mean)
print("The lowest mean score in reading is:", reading_lowest_mean)
```

```
The average score in reading is: 69.169
The highest mean score in reading is: 100
The lowest mean score in reading is: 17
```

In [35]:
```
print("The average score in writing is:", writing_mean)
print("The highest mean score in writing is:", writing_highest_mean)
print("The lowest mean score in writing is:", writing_lowest_mean)
```

```
The average score in writing is: 68.054
The highest mean score in writing is: 100
The lowest mean score in writing is: 10
```

3. Question 3's answer

In [36]:
```
df_copy = df.copy()
df_copy
```

Out[36]:

	gender	race	parental level of education	lunch	test preparation course	math score	reading score	writing score	total_score	result
0	female	group B	bachelor's degree	standard	none	72	72	74	72.67	PASS
1	female	group C	some college	standard	completed	69	90	88	82.33	PASS
2	female	group B	master's degree	standard	none	90	95	93	92.67	PASS
3	male	group A	associate's degree	free/reduced	none	47	57	44	49.33	FAIL
4	male	group C	some college	standard	none	76	78	75	76.33	PASS
...	...	...	...	...	...	...	...	...	...	...
995	female	group E	master's degree	standard	completed	88	99	95	94.00	PASS
996	male	group C	high school	free/reduced	none	62	55	55	57.33	FAIL
997	female	group C	high school	free/reduced	completed	59	71	65	65.00	PASS
998	female	group D	some college	standard	completed	68	78	77	74.33	PASS
999	female	group D	some college	free/reduced	none	77	86	86	83.00	PASS

1000 rows × 10 columns

In [37]:
```
df_copy.isnull().sum()
```

Out[37]:
```
gender                       0
race                         0
parental level of education  0
lunch                        0
test preparation course      0
math score                   0
reading score                0
writing score                0
total_score                  0
result                       0
dtype: int64
```

In [38]:
```
df_copy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 10 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   gender                       1000 non-null   object 
 1   race                         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  
 8   total_score                  1000 non-null   float64
 9   result                       1000 non-null   object 
dtypes: float64(1), int64(3), object(6)
memory usage: 78.2+ KB
```

Hapus feature yang tidak diperlukan

In [39]:
```
df_copy.drop(columns="gender",inplace=True)
df_copy.drop(columns="race",inplace=True)
df_copy.drop(columns="parental level of education",inplace=True)
df_copy.drop(columns="lunch",inplace=True)
df_copy.drop(columns="test preparation course",inplace=True)
```

Encoding data

In [40]:
```
df_knn = df_copy.copy()

def encode_data(feature_name):
  mapping_dict = {}
  unique_values = list(df_copy[feature_name].unique())
  for idx in range(len(unique_values)):
    mapping_dict[unique_values[idx]] = idx
  return mapping_dict
df_knn['result'].replace({'PASS':0, 'FAIL':1}, inplace = True)
df_knn
```

Out[40]:

	math score	reading score	writing score	total_score	result
0	72	72	74	72.67	0
1	69	90	88	82.33	0
2	90	95	93	92.67	0
3	47	57	44	49.33	1
4	76	78	75	76.33	0
...	...	...	...	...	...
995	88	99	95	94.00	0
996	62	55	55	57.33	1
997	59	71	65	65.00	0
998	68	78	77	74.33	0
999	77	86	86	83.00	0

1000 rows × 5 columns

In [41]:
```
df_copy
```

Out[41]:

	math score	reading score	writing score	total_score	result
0	72	72	74	72.67	PASS
1	69	90	88	82.33	PASS
2	90	95	93	92.67	PASS
3	47	57	44	49.33	FAIL
4	76	78	75	76.33	PASS
...	...	...	...	...	...
995	88	99	95	94.00	PASS
996	62	55	55	57.33	FAIL
997	59	71	65	65.00	PASS
998	68	78	77	74.33	PASS
999	77	86	86	83.00	PASS

1000 rows × 5 columns

In [42]:
```
Y =df_knn['result']
X =df_knn.drop(columns ='result')
```

In [43]:
```
from    sklearn.model_selection import  train_test_split
X_train,        X_test, y_train,        y_test  =       train_test_split(X,     Y, test_size=0.2,       random_
```

In [44]:
```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn.fit(X_train,y_train)
print('Accuracy of K-NN classifier on training set: {:.2f}'.format(knn.score(X_train,y_train)))
print('Accuracy of K-NN classifier on test set: {:.2f}'.format(knn.score(X_test,y_test)))
```

```
Accuracy of K-NN classifier on training set: 0.99
Accuracy of K-NN classifier on test set: 0.99
```

In [45]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Make predictions on the test set
y_pred = knn.predict(X_test)

# Calculate evaluation metrics
accuracy = accuracy_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print("Accuracy: {:.2f}%".format(accuracy*100))
print('F1-score: : {:.2f}%'.format(f1*100))
```

```
Accuracy: 99.00%
F1-score: : 98.31%
```

Visualisasi menggunakan heatmap

In [46]:
```
y_pred = knn.predict(X_test)
mat = confusion_matrix(y_test, y_pred)
sns.heatmap(mat.T, square = True, annot = True, cmap = 'Blues')
plt.xlabel('true label')
plt.ylabel('predicted label')
```

Out[46]: Text(77.92222222222227, 0.5, 'predicted label')