Thyago De Oliveira Alves

2018400@student.cct.ie

CCT College

Machine Learning CA1

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

filename = "StudentsPerformance.csv"  
score\_df = pd.read\_csv(filename)  
score\_df.head()

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

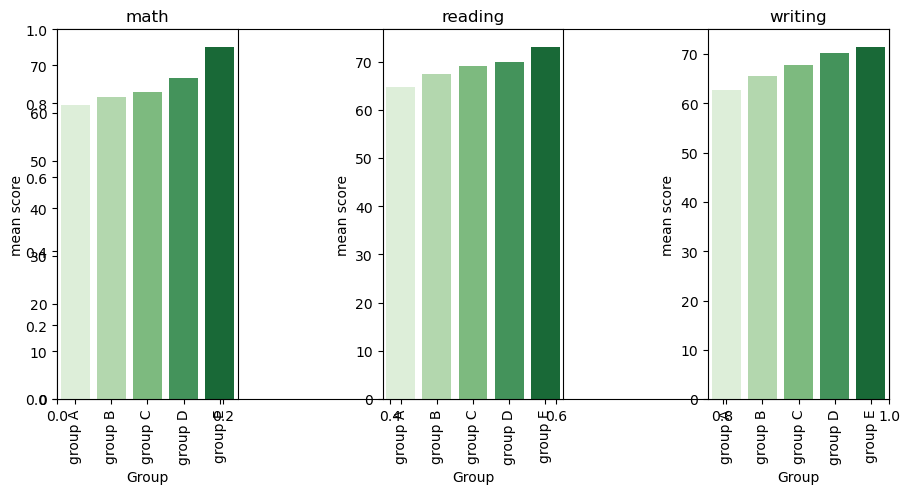
score\_df.rename(columns={"race/ethnicity":"ethnicity","parental level of education":"parent\_education"  
 ,"math score":"math","reading score":"reading","writing score":"writing",  
 "test preparation course":"pre"},inplace=True)  
score\_df.head()

gender ethnicity parent\_education lunch pre math \  
0 female group B bachelor's degree standard none 72   
1 female group C some college standard completed 69   
2 female group B master's degree standard none 90   
3 male group A associate's degree free/reduced none 47   
4 male group C some college standard none 76   
  
 reading writing   
0 72 74   
1 90 88   
2 95 93   
3 57 44   
4 78 75

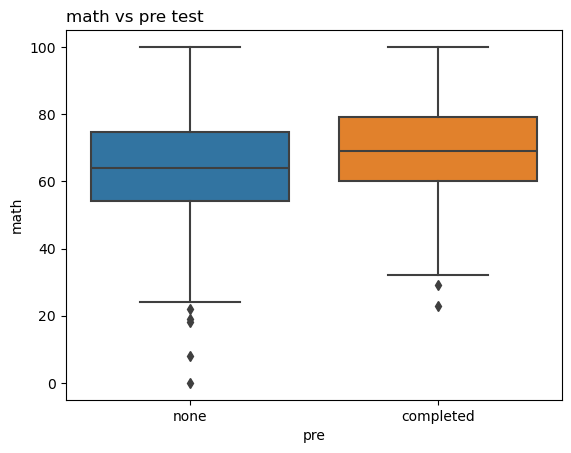
score\_df.dtypes

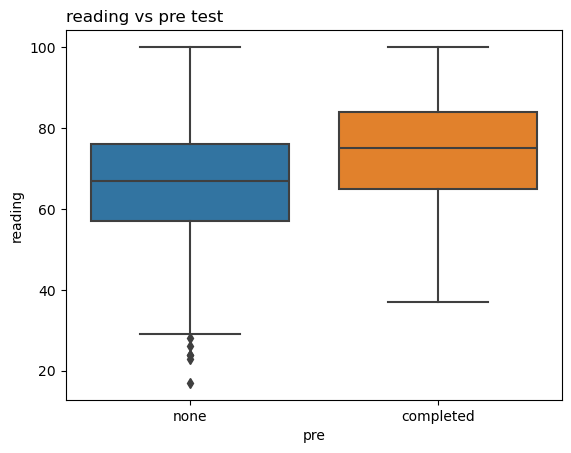
gender object  
ethnicity object  
parent\_education object  
lunch object  
pre object  
math int64  
reading int64  
writing int64  
dtype: object

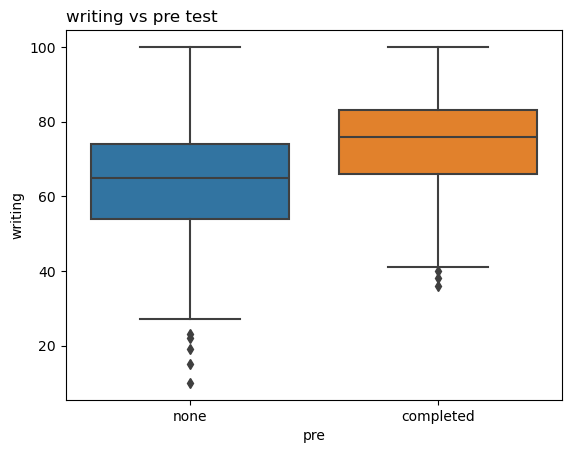
fig, ax = plt.subplots()  
fig.subplots\_adjust(hspace=0.8, wspace=0.8, left = 0.2, right = 1.5)  
for idx in range(3):  
 plt.subplot(1,3, idx+1)  
 ethn\_df = score\_df.groupby("ethnicity")[list(score\_df.columns[-3:])[idx]].mean()  
 sns.barplot(x=ethn\_df.index, y = ethn\_df.values, palette = "Greens")  
 plt.xlabel("Group")  
 plt.ylabel("mean score")  
 plt.xticks(rotation=90)  
 plt.title(list(score\_df.columns[-3:])[idx])  
plt.show()



for item in score\_df.columns[-3:]:  
 sns.boxplot(x=score\_df["pre"], y=score\_df[item])  
 plt.title(item+" vs pre test", loc="left")  
 plt.show()

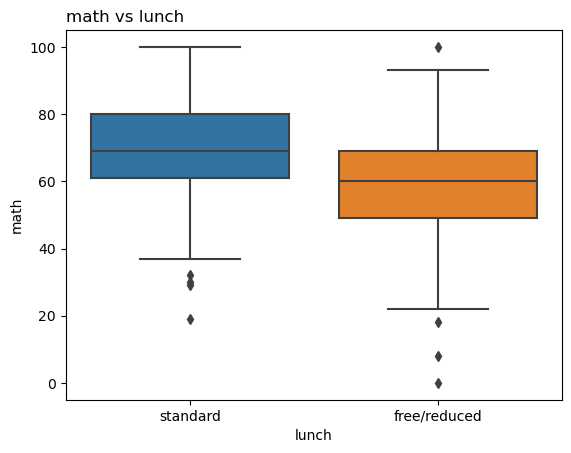


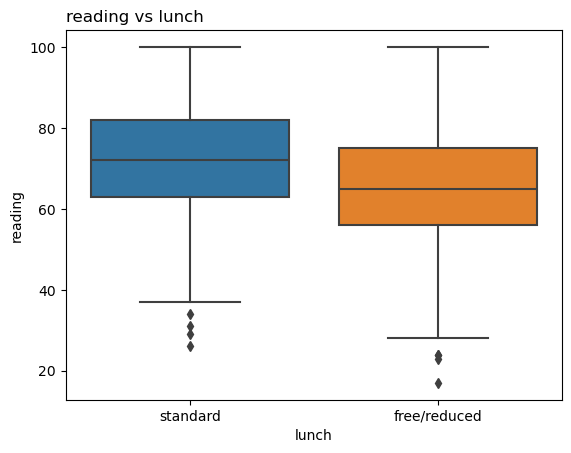


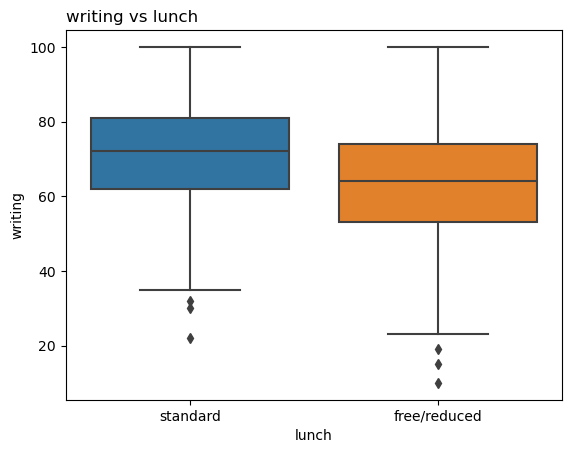


#The score distribution got narrower if   
#students complete the preparation before test, and also we can see that the average of the score is better.

for item in score\_df.columns[-3:]:  
 sns.boxplot(x=score\_df["lunch"], y=score\_df[item])  
 plt.title(item+" vs lunch", loc="left")  
 plt.show()







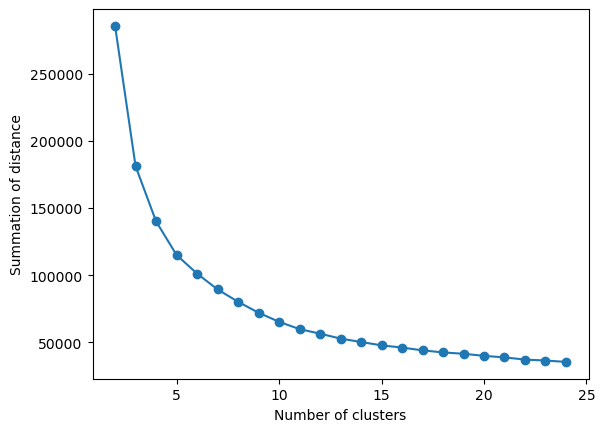
#Make sense! Students are easier to get better score once they eat standardly.  
  
#I check the data type earily. Then I transform some features by label encoder.

labelencoder = LabelEncoder()  
train\_df = score\_df.copy()  
train\_df["parent\_education"] = labelencoder.fit\_transform(train\_df["parent\_education"])  
train\_df["pre"] = labelencoder.fit\_transform(train\_df["pre"])  
train\_df["lunch"] = labelencoder.fit\_transform(train\_df["lunch"])  
train\_df.head()

gender ethnicity parent\_education lunch pre math reading writing  
0 female group B 1 1 1 72 72 74  
1 female group C 4 1 0 69 90 88  
2 female group B 3 1 1 90 95 93  
3 male group A 0 0 1 47 57 44  
4 male group C 4 1 1 76 78 75

#Great! The features "parent\_education",   
#"lunch" and "pre" are labeled by numbers. Next, we use KMeans argorithm to classify the dataset.

kmeans\_dis = list()  
for idx in range(2, 25):   
 kmeans = KMeans(init = "k-means++", n\_clusters = idx, n\_init = 20)  
 kmeans.fit\_transform(train\_df.iloc[:, 2:])  
 kmeans\_dis.append(kmeans.inertia\_)  
plt.plot(list(range(2,25)), kmeans\_dis, marker = "o")  
plt.xlabel("Number of clusters")  
plt.ylabel("Summation of distance")  
plt.show()



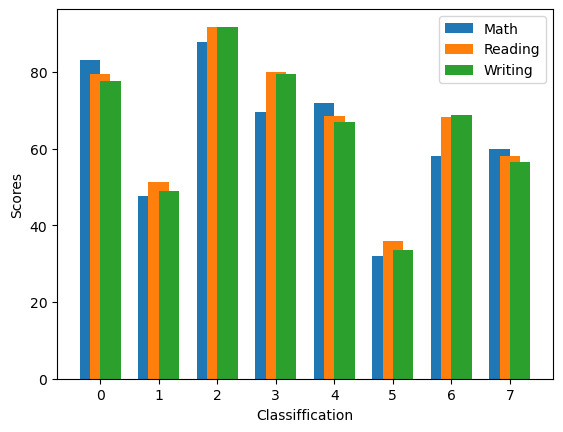
kmeans = KMeans(init = "k-means++", n\_clusters = 8)  
kmeans.fit\_transform(train\_df.iloc[:, 2:])  
kmeans\_label = kmeans.labels\_  
score\_df["classification"] = kmeans\_label  
score\_df.head(10)

gender ethnicity parent\_education lunch pre math \  
0 female group B bachelor's degree standard none 72   
1 female group C some college standard completed 69   
2 female group B master's degree standard none 90   
3 male group A associate's degree free/reduced none 47   
4 male group C some college standard none 76   
5 female group B associate's degree standard none 71   
6 female group B some college standard completed 88   
7 male group B some college free/reduced none 40   
8 male group D high school free/reduced completed 64   
9 female group B high school free/reduced none 38   
  
 reading writing classification   
0 72 74 4   
1 90 88 3   
2 95 93 2   
3 57 44 1   
4 78 75 0   
5 83 78 3   
6 95 92 2   
7 43 39 5   
8 64 67 6   
9 60 50 1

class\_df = score\_df.groupby("classification")[score\_df.columns[-4:-1]].mean()  
class\_df

math reading writing  
classification   
0 83.243478 79.539130 77.573913  
1 47.759124 51.437956 49.043796  
2 87.754386 91.833333 91.736842  
3 69.528662 79.942675 79.509554  
4 71.966216 68.527027 66.986486  
5 32.135135 35.972973 33.594595  
6 58.047619 68.367347 68.755102  
7 59.862069 58.151724 56.613793

ind = np.arange(8)  
width = 0.35  
fig, ax = plt.subplots()  
rects1 = ax.bar(ind - width/2, class\_df.math, width, label='Math')  
rects2 = ax.bar(ind, class\_df.reading, width, label='Reading')  
rects3 = ax.bar(ind + width/2, class\_df.writing, width, label='Writing')  
  
ax.set\_xlabel('Classiffication')  
ax.set\_ylabel('Scores')  
ax.set\_xticks(ind)  
ax.legend()  
plt.show()



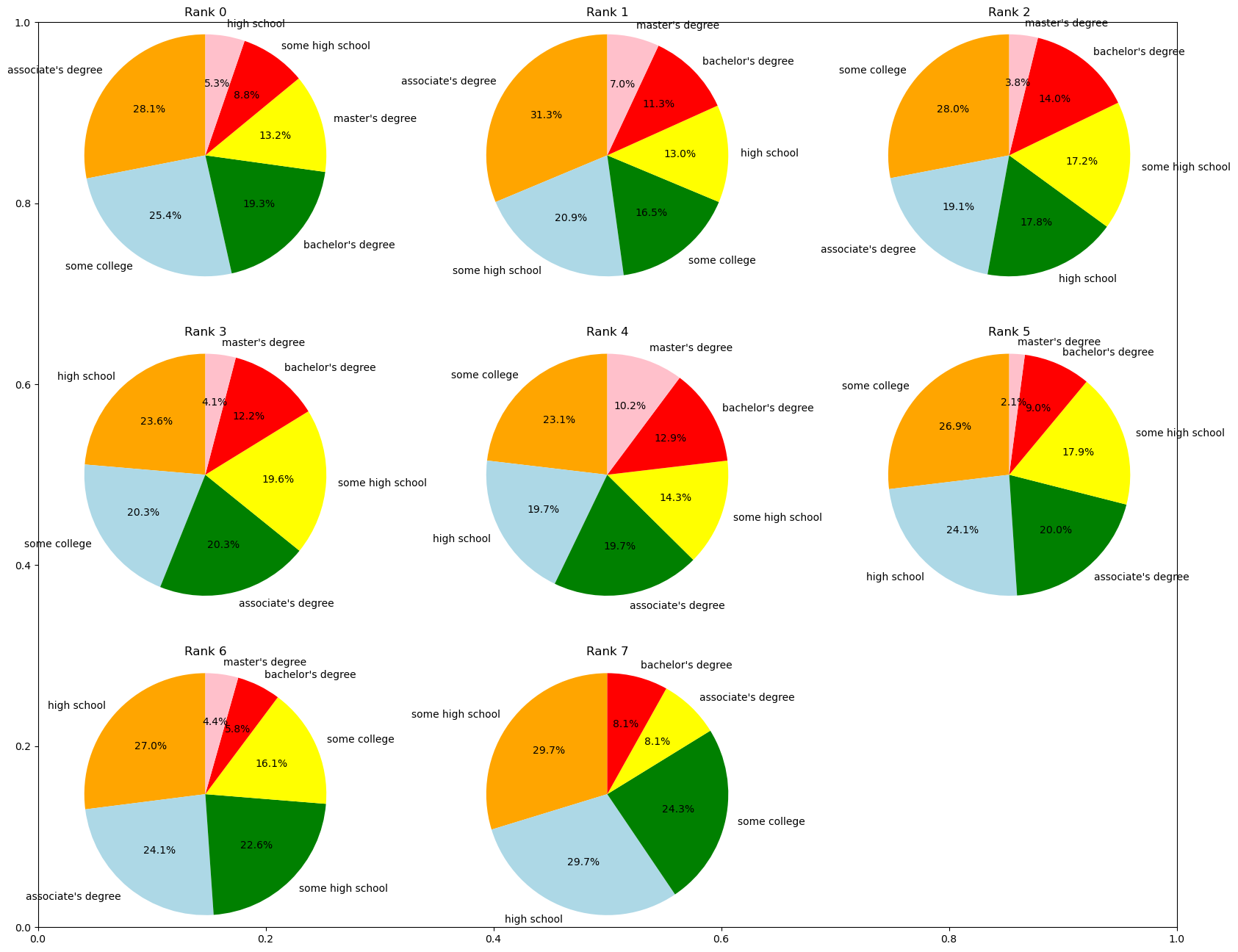
class\_df["total\_ave\_score"] = (class\_df.math + class\_df.reading + class\_df.writing)/3  
rank = class\_df["total\_ave\_score"].sort\_values(ascending = False)  
rank.index

Index([2, 0, 3, 4, 6, 7, 1, 5], dtype='int32', name='classification')

rank

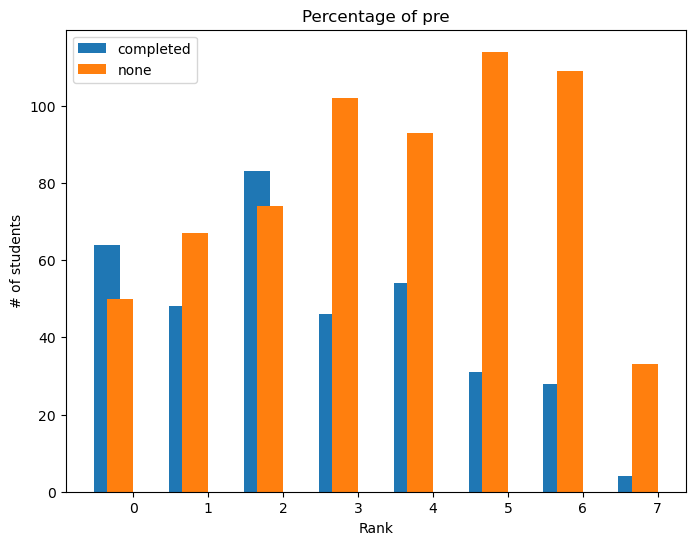
classification  
2 90.441520  
0 80.118841  
3 76.326964  
4 69.159910  
6 65.056689  
7 58.209195  
1 49.413625  
5 33.900901  
Name: total\_ave\_score, dtype: float64

def plot\_pie\_chart(column):  
 fig, ax = plt.subplots(figsize=(20,16))  
 color = ["orange","lightblue","green","yellow","red","pink","brown","gray"]  
 for idx in range(8):  
 plt.subplot(3, 3, idx+1)  
 num = "class"+ str(idx)  
 num = score\_df[score\_df["classification"]==rank.index[idx]]  
 percentage\_of\_parent\_edu = num[column].value\_counts()  
 percentage\_of\_parent\_edu.sort\_index()  
 label = percentage\_of\_parent\_edu.index  
 value = percentage\_of\_parent\_edu.values  
 plt.pie(value, labels = label, autopct = "%1.1f%%",  
 startangle=90, radius = 4, colors = color[:len(label)])  
 plt.axis("equal")  
 plt.title("Rank "+str(idx))  
 plt.show()  
plot\_pie\_chart("parent\_education")



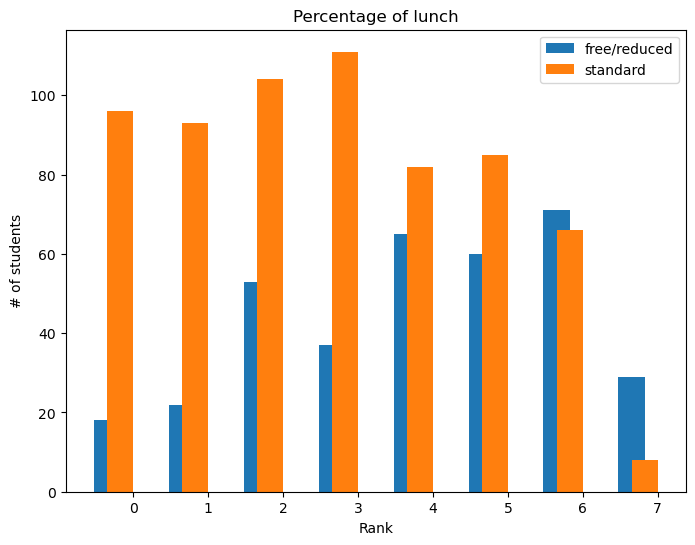
#Let's define the high degree of education. Parents having bachelor or master degree are high-level educated. So we focus on these two terms.  
  
#As pie chart were shown above, we can easily understand the ratio of high-degree education. For the rank0, its ratio is around 32%. In addition, there are no differences between rank1 to rank3, and the ratio are around 15~17%. Finally, the ratio is only 8% in rank7.  
  
#We calculated the average score of each rank before, so we can say that parent's education affect the score but not obviously, because there are still 70%~80% parents without high education degree.

def plot\_bar\_chart(column):  
 fig, ax = plt.subplots(figsize=(8,6))  
 index\_dict = dict()  
 width = 0.35  
 ind = np.arange(8)  
 for idx in range(8):  
 num = "class"+ str(idx)  
 num = score\_df[score\_df["classification"]==rank.index[idx]]  
 percentage\_of\_column = num[column].value\_counts()  
 percentage\_of\_column = percentage\_of\_column.sort\_index()  
 for key in percentage\_of\_column.index:  
 if key not in index\_dict.keys():  
 index\_dict[key] = []  
 index\_dict[key].append(percentage\_of\_column[key]) #/percentage\_of\_column.values.sum())  
 else:  
 index\_dict[key].append(percentage\_of\_column[key]) #/percentage\_of\_column.values.sum())  
   
 percentage\_of\_column = score\_df[score\_df["classification"]==rank.index[4]][column].value\_counts().sort\_index()  
 for i in range(len(percentage\_of\_column.index)):  
 rects = ax.bar(ind - width/(i+1),   
 index\_dict[percentage\_of\_column.index[i]],  
 width, label=percentage\_of\_column.index[i])  
  
 ax.set\_xlabel('Rank')  
 ax.set\_ylabel('# of students')  
 ax.set\_title("Percentage of " + column)  
 ax.set\_xticks(ind)  
 ax.legend()  
 plt.show()  
   
plot\_bar\_chart("pre")

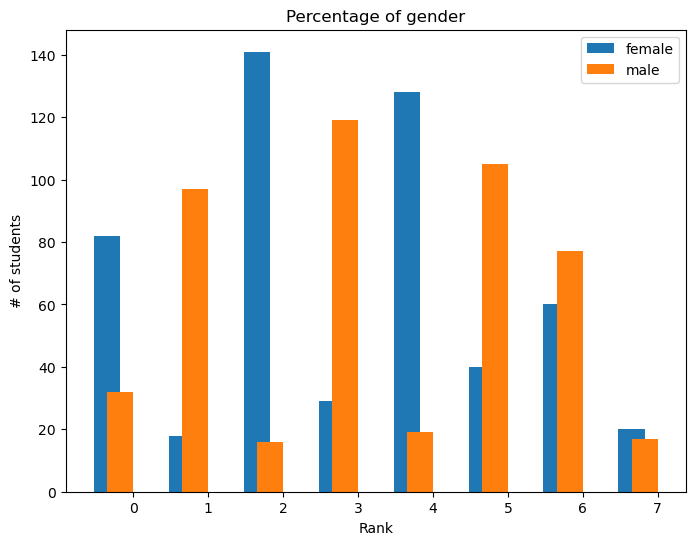


#Nice! Over 50% of students in rank0 completed the test preparation course,   
#and normally there is about 70~80% students in rank7 hadn't finished course.  
#It is say that preparation course can help students get better score.

plot\_bar\_chart("lunch")



plot\_bar\_chart("gender")



Cell In[2], line 1  
 pandoc jupyter\_file.ipynb -s -o new\_word\_file.docx  
 ^  
SyntaxError: invalid syntax