实验一 《多源数据集成、清洗和统计》

题目

广州大学某班有同学100人，现要从两个数据源汇总学生数据。第一个数据源在数据库中，第二个数据源在txt文件中，两个数据源课程存在缺失、冗余和不一致性，请用C/C++/Java程序实现对两个数据源的一致性合并以及每个学生样本的数值量化。

● 数据库表：ID (int), 姓名(string), 家乡(string:限定为Beijing / Guangzhou / Shenzhen / Shanghai), 性别（string:boy/girl）、身高（float:单位是cm)）、课程1成绩（float）、课程2成绩（float）、...、课程10成绩(float)、体能测试成绩（string：bad/general/good/excellent）；其中课程1-课程5为百分制，课程6-课程10为十分制。

● txt文件：ID(string：6位学号)，性别（string:male/female）、身高（string:单位是m)）、课程1成绩（string）、课程2成绩（string）、...、课程10成绩(string)、体能测试成绩（string：差/一般/良好/优秀）；其中课程1-课程5为百分制，课程6-课程10为十分制。

两个数据源合并后读入内存，并统计：

1. 学生中家乡在Beijing的所有课程的平均成绩。

2. 学生中家乡在广州，课程1在80分以上，且课程9在9分以上的男同学的数量。(备注：该处做了修正，课程10数据为空，更改为课程9)

3. 比较广州和上海两地女生的平均体能测试成绩，哪个地区的更强些？

4. 学习成绩和体能测试成绩，两者的相关性是多少？（九门课的成绩分别与体能成绩计算相关性）

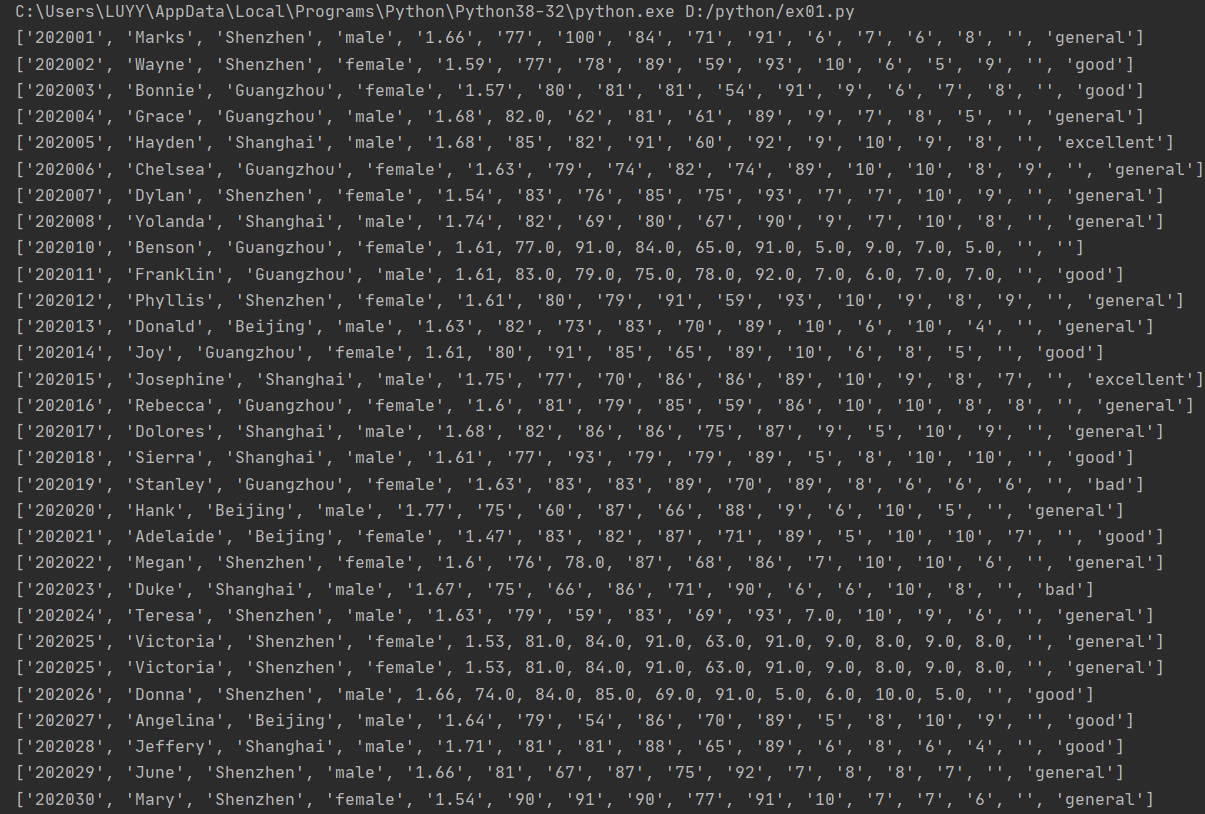
代码如下：

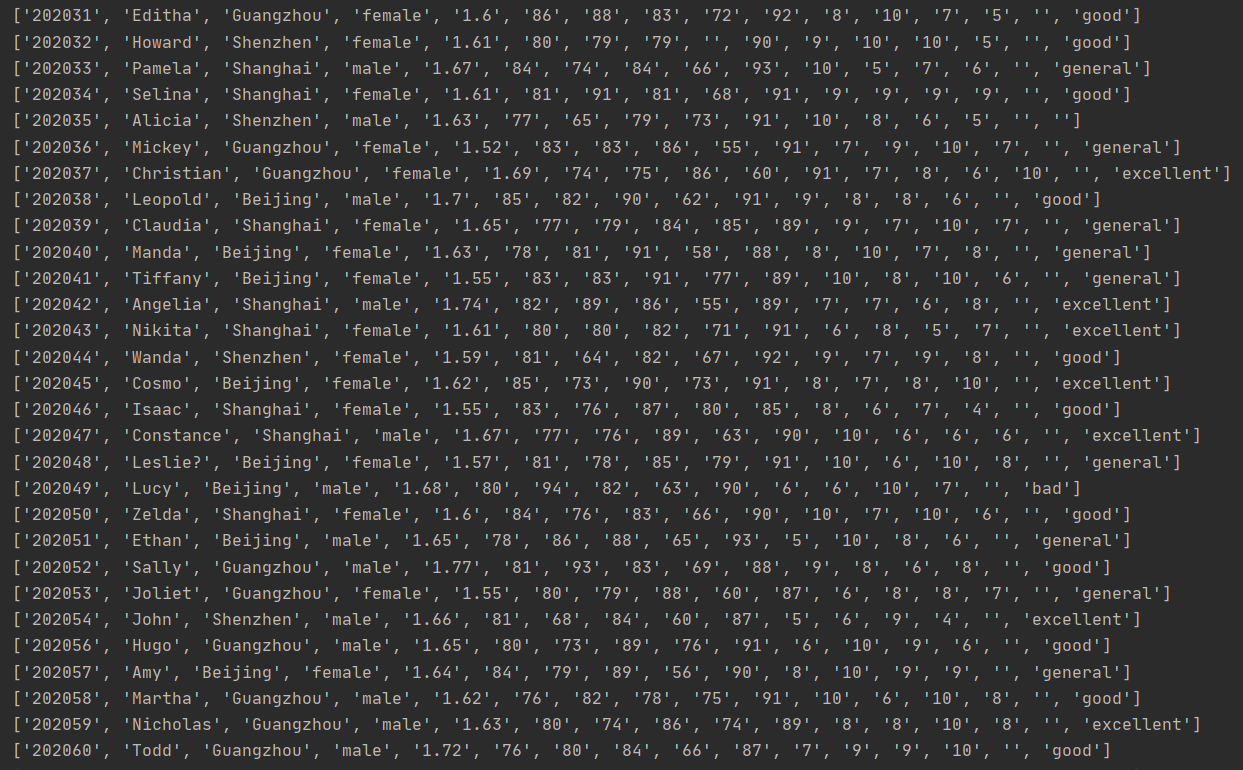
import sys  
import xlrd as xd  
import numpy as np  
import pyecharts.options as opts # for 散点图  
  
# 导入txt的数据  
d1 = []  
with open(r"F:\大三上\机器学习\data1.txt", 'r') as f:  
 for line in f:  
 d1.append(list(line.strip('\n').split(',')))  
  
# 导入xlsx的数据  
data = xd.open\_workbook ('F:/大三上/机器学习/data2.xlsx') # 打开excel表所在路径  
sheet = data.sheet\_by\_name('Sheet1') # 读取数据，以excel表名来打开  
d2 = []  
for r in range(sheet.nrows): # 将表中数据按行逐步添加到列表中，最后转换为list结构  
 data1 = []  
 for c in range(sheet.ncols):  
 data1.append(sheet.cell\_value(r, c))  
 d2.append(list(data1))  
  
# 将d2中的ID与d1一致化  
for n in range(1, len(d2)):  
 d2[n][0] = str(int(d2[n][0]) + 202000)  
  
# 比对两个数据源的数据进行合并，当数据不一致时以txt数据源d1为准  
d = d1  
  
# 把d1,d2的ID存放在另外的列表中  
d1\_ID = []  
d2\_ID = []  
for x in range(len(d1)):  
 d1\_ID.append(d1[x][0])  
for y in range(len(d2)):  
 d2\_ID.append(d2[y][0])  
  
# 去除重复的ID  
Delete\_Index = []  
for p in range(1, len(d)):  
 if d[p][0] == d[p - 1][0]:  
 Delete\_Index.append(p) # 保存重复的ID的下标  
  
sign = 0 # 记下已删除的行的个数，当已删除d中的一行时，下标值会有变化，实际的下标值为原本的Delete\_Index加上标志sign  
for q in range(len(Delete\_Index)): # 删除重复的ID  
 del d[Delete\_Index[q] - sign]  
 sign += 1

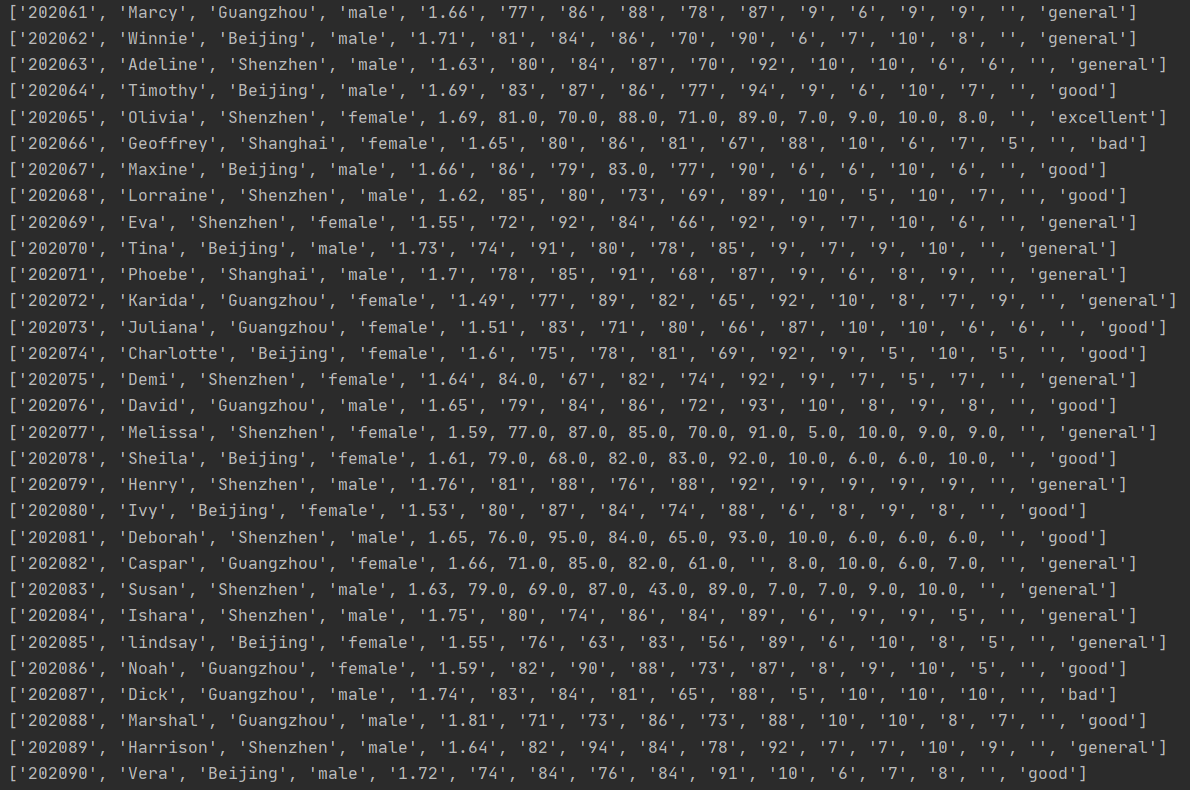
# 定义一个函数查判断某id是否在d1中  
def FindID\_d1(ID):  
 if ID in d1\_ID:  
 return 1  
 else:  
 return 0  
  
  
# 定义一个函数判断某ID是否在d2，有则返回下标  
def FindID\_d2(ID):  
 if ID in d2\_ID:  
 for r in range(len(d2\_ID)):  
 if ID == d2\_ID[r]:  
 return r  
 else:  
 return -1  
  
  
# 把d1数据为空的地方按照d2补充上  
for i in range(1, len(d)):  
 for j in range(1, len(d[0])):  
 if d[i][j] == '':  
 tmp\_index =FindID\_d2(d[i][0])  
 if tmp\_index >= 0:  
 d[i][j] = d2[tmp\_index][j]  
  
  
# 把d2不在d1中的ID加入到d1中  
for y in range(len(d2)):  
 if FindID\_d1(d2\_ID[y]) == 0:  
 d.append(list(d2[y]))  
  
for n in range(1, len(d)):  
 for m in range(1, len(d[0])):  
 if d[n][3] == 'girl': # 如果性别显示为girl则改为female  
 d[n][3] = 'female'  
 if d[n][3] == 'boy':  
 d[n][3] = 'male'  
 if float(d[n][4]) > 2: # 如果身高为三位数则改为小数形式  
 d[n][4] = float(d[n][4])/100  
  
del d[0] # 删除第一行，只剩下数据  
d.sort(key = lambda x: x[0], reverse=False) # 按照ID排序  
  
for line in d: # 输出最后结果  
 print(line)  
  
  
# 把最终结果写入到Final\_Data.xlsx中  
output = open('F:/大三上/机器学习/Final\_Data.xlsx', 'w', encoding='gbk')  
output.write('ID\tName\tCity\tGender\tHeight\tC1\tC2\tC3\tC4\tC5\tC6\tC7\tC8\tC9\tC10\tConstitution\n')  
for i in range(len(d)):  
 for j in range(len(d[0])):  
 output.write(str(d[i][j])) # write函数不能写int类型的参数，所以使用str()转化  
 output.write('\t') # 相当于Tab一下，换一个单元格  
 output.write('\n') # 写完一行立马换行  
output.close()  
  
  
# 两个数据源合并后读入内存，并统计：  
# 1. 学生中家乡在Beijing的所有课程的平均成绩。  
print("--------------------------------------------------------")  
print("1. 学生中家乡在Beijing的所有课程的平均成绩。")  
Score\_tempC1 = []  
Score\_tempC2 = []  
Score\_tempC3 = []  
Score\_tempC4 = []  
Score\_tempC5 = []  
Score\_tempC6 = []  
Score\_tempC7 = []  
Score\_tempC8 = []  
Score\_tempC9 = []  
for i in range(len(d)):  
 for j in range(len(d[0])):  
 if d[i][2] == 'Beijing':  
 Score\_tempC1.append(d[i][5])  
 Score\_tempC2.append(d[i][6])  
 Score\_tempC3.append(d[i][7])  
 Score\_tempC4.append(d[i][8])  
 Score\_tempC5.append(d[i][9])  
 Score\_tempC6.append(d[i][10])  
 Score\_tempC7.append(d[i][11])  
 Score\_tempC8.append(d[i][12])  
 Score\_tempC9.append(d[i][13])  
# 去掉列表为空的元素  
Score\_C1 = list(filter(None, Score\_tempC1))  
Score\_C2 = list(filter(None, Score\_tempC2))  
Score\_C3 = list(filter(None, Score\_tempC3))  
Score\_C4 = list(filter(None, Score\_tempC4))  
Score\_C5 = list(filter(None, Score\_tempC5))  
Score\_C6 = list(filter(None, Score\_tempC6))  
Score\_C7 = list(filter(None, Score\_tempC7))  
Score\_C8 = list(filter(None, Score\_tempC8))  
Score\_C9 = list(filter(None, Score\_tempC9))  
# 输出平均成绩  
sum\_C1 = 0.0  
sum\_C2 = 0.0  
sum\_C3 = 0.0  
sum\_C4 = 0.0  
sum\_C5 = 0.0  
sum\_C6 = 0.0  
sum\_C7 = 0.0  
sum\_C8 = 0.0  
sum\_C9 = 0.0  
for m in range(len(Score\_C1)): # 计算C1的总成绩  
 sum\_C1 += float(Score\_C1[m])  
for m in range(len(Score\_C2)): # 计算C2的总成绩  
 sum\_C2 += float(Score\_C2[m])  
for m in range(len(Score\_C3)): # 计算C3的总成绩  
 sum\_C3 += float(Score\_C3[m])  
for m in range(len(Score\_C4)): # 计算C4的总成绩  
 sum\_C4 += float(Score\_C4[m])  
for m in range(len(Score\_C5)): # 计算C5的总成绩  
 sum\_C5 += float(Score\_C5[m])  
for m in range(len(Score\_C6)): # 计算C6的总成绩  
 sum\_C6 += float(Score\_C6[m])  
for m in range(len(Score\_C7)): # 计算C7的总成绩  
 sum\_C7 += float(Score\_C7[m])  
for m in range(len(Score\_C8)): # 计算C8的总成绩  
 sum\_C8 += float(Score\_C8[m])  
for m in range(len(Score\_C9)): # 计算C9的总成绩  
 sum\_C9 += float(Score\_C9[m])  
print("学生中家乡在Beijing的所有课程的平均成绩.")  
print("课程C1的平均成绩为：", sum\_C1/len(Score\_C1))  
print("课程C2的平均成绩为：", sum\_C2/len(Score\_C2))  
print("课程C3的平均成绩为：", sum\_C3/len(Score\_C3))  
print("课程C4的平均成绩为：", sum\_C4/len(Score\_C4))  
print("课程C5的平均成绩为：", sum\_C5/len(Score\_C5))  
print("课程C6的平均成绩为：", sum\_C6/len(Score\_C6))  
print("课程C7的平均成绩为：", sum\_C7/len(Score\_C7))  
print("课程C8的平均成绩为：", sum\_C8/len(Score\_C8))  
print("课程C9的平均成绩为：", sum\_C9/len(Score\_C9))  
  
# 2. 学生中家乡在广州，课程1在80分以上，且课程9在9分以上的男同学的数量  
print("--------------------------------------------------------")  
print("2. 学生中家乡在广州，课程1在80分以上，且课程9在9分以上的男同学的数量")  
count = 0  
for i in range(len(d)):  
 for j in range(len(d[0])):  
 if d[i][2] == 'Guangzhou' and d[i][3] == 'male' and float(d[i][5]) > 80 and float(d[i][13]) > 9:  
 count += 1  
print("学生中家乡在广州，课程1在80分以上，且课程9在9分以上的男同学的数量:", count)  
  
# 3. 比较广州和上海两地女生的平均体能测试成绩，哪个地区的更强些？  
# bad/general/good/excellent分别为1分/2分/3分/4分  
print("--------------------------------------------------------")  
print("3. 比较广州和上海两地女生的平均体能测试成绩，哪个地区的更强些？")  
Constitution\_SH = 0.0 # 上海女生体能成绩  
Constitution\_GZ = 0.0 # 广州女生体能成绩  
count\_SH = 0 # 上海女生人数  
count\_GZ = 0 # 广州女生人数  
for i in range(len(d)):  
 for j in range(len(d[0])):  
 if d[i][2] == 'Shanghai' and d[i][3] == 'female': # 按照条件筛选  
 if d[i][15] != '':  
 count\_SH += 1  
 if d[i][15] == 'bad': # 不同的评价有不同的分数  
 Constitution\_SH += 1  
 if d[i][15] == 'general':  
 Constitution\_SH += 2  
 if d[i][15] == 'good':  
 Constitution\_SH += 3  
 if d[i][15] == 'excellent':  
 Constitution\_SH += 4  
 if d[i][2] == 'Guangzhou' and d[i][3] == 'female':  
 if d[i][15] != '':  
 count\_GZ += 1  
 if d[i][15] == 'bad':  
 Constitution\_GZ += 1  
 if d[i][15] == 'general':  
 Constitution\_GZ += 2  
 if d[i][15] == 'good':  
 Constitution\_GZ += 3  
 if d[i][15] == 'excellent':  
 Constitution\_GZ += 4  
print("上海女生体能平均成绩为：", Constitution\_SH/count\_SH)  
print("广州女生体能平均成绩为：", Constitution\_GZ/count\_GZ)  
if Constitution\_SH/count\_SH > Constitution\_GZ/count\_GZ:  
 print("上海女生体能更强。")  
elif Constitution\_SH/count\_SH < Constitution\_GZ/count\_GZ:  
 print("广州女生体能更强。")  
else:  
 print("两个地区女生体能不分上下。")  
  
# 4. 学习成绩和体能测试成绩，两者的相关性是多少？（九门课的成绩分别与体能成绩计算相关性）  
# 把体能测试转de成绩化为数字表示的形式  
print("--------------------------------------------------------")  
print("4. 学习成绩和体能测试成绩，两者的相关性是多少？（九门课的成绩分别与体能成绩计算相关性）")  
Constitution\_Socre = []  
for i in range(len(d)):  
 if d[i][15] == 'bad':   
 Constitution\_Socre.append(1)  
 if d[i][15] == 'general':  
 Constitution\_Socre.append(2)  
 if d[i][15] == 'good':  
 Constitution\_Socre.append(3)  
 if d[i][15] == 'excellent':  
 Constitution\_Socre.append(4)  
# 求协方差  
ss\_Constitution = 0.0 # 记录体能成绩的协方差  
from numpy import \*  
mean\_Constitution = mean(Constitution\_Socre)  
for j in range(len(Constitution\_Socre)): # 协方差公式，先计算和  
 ss\_Constitution += (Constitution\_Socre[j] - mean\_Constitution) \* (Constitution\_Socre[j] - mean\_Constitution)  
ss\_Constitution = ss\_Constitution/(len(Constitution\_Socre) - 1)  
print("体能成绩的协方差是：", ss\_Constitution)  
std\_Constitution = ss\_Constitution \*\* 0.5  
print("体能成绩的标准差是：", std\_Constitution)  
Constitution\_Socre1 = [] # 体能成绩数组B`  
for m in range(len(Constitution\_Socre)):  
 Constitution\_Socre1.append((Constitution\_Socre[m] - mean\_Constitution)/std\_Constitution)  
  
# 计算C1  
C1\_Score = []  
for i in range(len(d)):  
 if d[i][5] != '' and d[i][15] != '': # 体能成绩有两处为空，要舍去  
 C1\_Score.append(float(d[i][5]))  
# 求协方差  
ss\_C1 = 0.0  
mean\_C1 = mean(C1\_Score) # C1成绩的平均值  
for i in range(len(C1\_Score)): # 协方差公式，先计算和  
 ss\_C1 += (C1\_Score[i] - mean\_C1) \* (C1\_Score[i] - mean\_C1)  
ss\_C1 = ss\_C1/(len(C1\_Score) - 1)  
std\_C1 = ss\_C1 \*\* 0.5  
print("C1成绩的平均值是：", mean\_C1)  
print("C1成绩的协方差是：", ss\_C1)  
print("C1成绩的标准差是：", std\_C1)  
C1\_Score1 = []  
for m in range(len(C1\_Score)): # C1成绩数组C1`  
 C1\_Score1.append((C1\_Score[m] - mean\_C1)/std\_C1)  
  
# 体能 成绩数组和C1成绩数组的相关性  
correlation\_Con\_C1 = 0.0  
for m in range(len(Constitution\_Socre1)):  
 correlation\_Con\_C1 += Constitution\_Socre1[m] \* C1\_Score[m]  
print("C1和体能测试成绩的相关性是（C1\*B)：", correlation\_Con\_C1)  
print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  
  
# 计算C2  
C2\_Score = []  
for i in range(len(d)):  
 if d[i][6] != '' and d[i][15] != '': # 体能成绩有两处为空，要舍去  
 C2\_Score.append(float(d[i][6]))  
# 求协方差  
ss\_C2 = 0.0  
mean\_C2 = mean(C2\_Score) # C2成绩的平均值  
for i in range(len(C2\_Score)): # 协方差公式，先计算和  
 ss\_C2 += (C2\_Score[i] - mean\_C2) \* (C2\_Score[i] - mean\_C2)  
ss\_C2 = ss\_C2/(len(C2\_Score) - 1)  
std\_C2 = ss\_C2 \*\* 0.5  
print("C2成绩的平均值是：", mean\_C2)  
print("C2成绩的协方差是：", ss\_C2)  
print("C2成绩的标准差是：", std\_C2)  
C2\_Score1 = []  
for m in range(len(C2\_Score)): # C2成绩数组C2`  
 C2\_Score1.append((C2\_Score[m] - mean\_C2)/std\_C2)  
  
# 体能 成绩数组和C2成绩数组的相关性  
correlation\_Con\_C2 = 0.0  
for m in range(len(Constitution\_Socre1)):  
 correlation\_Con\_C2 += Constitution\_Socre1[m] \* C2\_Score[m]  
print("C2和体能测试成绩的相关性是（C2\*B)：", correlation\_Con\_C2)  
print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  
  
# 计算C3  
C3\_Score = []  
for i in range(len(d)):  
 if d[i][7] != '' and d[i][15] != '': # 体能成绩有两处为空，要舍去  
 C3\_Score.append(float(d[i][7]))  
# 求协方差  
ss\_C3 = 0.0  
mean\_C3 = mean(C3\_Score) # C3成绩的平均值  
for i in range(len(C3\_Score)): # 协方差公式，先计算和  
 ss\_C3 += (C3\_Score[i] - mean\_C3) \* (C3\_Score[i] - mean\_C3)  
ss\_C3 = ss\_C3/(len(C3\_Score) - 1)  
std\_C3 = ss\_C3 \*\* 0.5  
print("C3成绩的平均值是：", mean\_C3)  
print("C3成绩的协方差是：", ss\_C3)  
print("C3成绩的标准差是：", std\_C3)  
C3\_Score1 = []  
for m in range(len(C3\_Score)): # C3成绩数组C3`  
 C3\_Score1.append((C3\_Score[m] - mean\_C3)/std\_C3)  
  
# 体能 成绩数组和C3成绩数组的相关性  
correlation\_Con\_C3 = 0.0  
for m in range(len(Constitution\_Socre1)):  
 correlation\_Con\_C3 += Constitution\_Socre1[m] \* C3\_Score[m]  
print("C3和体能测试成绩的相关性是（C3\*B)：", correlation\_Con\_C3)  
print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  
  
# 因为C4成绩有一个为空，所以需要按照C4重新计算体能测试数组B`  
# 把体能测试转de成绩化为数字表示的形式  
Constitution\_SocreA = []  
for i in range(len(d)):  
 if d[i][8] != '':  
 if d[i][15] == 'bad':  
 Constitution\_SocreA.append(1)  
 if d[i][15] == 'general':  
 Constitution\_SocreA.append(2)  
 if d[i][15] == 'good':  
 Constitution\_SocreA.append(3)  
 if d[i][15] == 'excellent':  
 Constitution\_SocreA.append(4)  
# 求协方差  
ss\_ConstitutionA = 0.0 # 记录体能成绩的协方差  
from numpy import \*  
mean\_ConstitutionA = mean(Constitution\_SocreA)  
for j in range(len(Constitution\_SocreA)): # 协方差公式，先计算和  
 ss\_ConstitutionA += (Constitution\_SocreA[j] - mean\_ConstitutionA) \* (Constitution\_SocreA[j] - mean\_ConstitutionA)  
ss\_ConstitutionA = ss\_ConstitutionA/(len(Constitution\_SocreA) - 1)  
print("体能成绩的协方差是(按照C4重新计算后)：", ss\_ConstitutionA)  
std\_ConstitutionA = ss\_ConstitutionA \*\* 0.5  
print("体能成绩的标准差是(按照C4重新计算后)：", std\_ConstitutionA)  
Constitution\_SocreA1 = [] # 体能成绩数组B`  
for m in range(len(Constitution\_SocreA)):  
 Constitution\_SocreA1.append((Constitution\_SocreA[m] - mean\_ConstitutionA)/std\_ConstitutionA)  
  
# 计算C4  
C4\_Score = []  
for i in range(len(d)):  
 if d[i][8] != '' and d[i][15] != '': # 体能成绩有两处为空，要舍去  
 C4\_Score.append(float(d[i][8]))  
# 求协方差  
ss\_C4 = 0.0  
mean\_C4 = mean(C4\_Score) # C4成绩的平均值  
for i in range(len(C4\_Score)): # 协方差公式，先计算和  
 ss\_C4 += (C4\_Score[i] - mean\_C4) \* (C4\_Score[i] - mean\_C4)  
ss\_C4 = ss\_C4/(len(C4\_Score) - 1)  
std\_C4 = ss\_C4 \*\* 0.5  
print("C4成绩的平均值是：", mean\_C4)  
print("C4成绩的协方差是：", ss\_C4)  
print("C4成绩的标准差是：", std\_C4)  
C4\_Score1 = []  
for m in range(len(C4\_Score)): # C4成绩数组C4`  
 C4\_Score1.append((C4\_Score[m] - mean\_C4)/std\_C4)  
  
# 体能 成绩数组和C4成绩数组的相关性  
correlation\_Con\_C4 = 0.0  
for m in range(len(Constitution\_SocreA1)):  
 correlation\_Con\_C4 += Constitution\_SocreA1[m] \* C4\_Score[m]  
print("C4和体能测试成绩的相关性是（C4\*B)：", correlation\_Con\_C4)  
print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  
  
# 因为C5成绩有一个为空，所以需要按照C5重新计算体能测试数组B`  
# 把体能测试转de成绩化为数字表示的形式  
Constitution\_SocreB = []  
for i in range(len(d)):  
 if d[i][9] != '':  
 if d[i][15] == 'bad':  
 Constitution\_SocreB.append(1)  
 if d[i][15] == 'general':  
 Constitution\_SocreB.append(2)  
 if d[i][15] == 'good':  
 Constitution\_SocreB.append(3)  
 if d[i][15] == 'excellent':  
 Constitution\_SocreB.append(4)  
# 求协方差  
ss\_ConstitutionB = 0.0 # 记录体能成绩的协方差  
from numpy import \*  
mean\_ConstitutionB = mean(Constitution\_SocreB)  
for j in range(len(Constitution\_SocreB)): # 协方差公式，先计算和  
 ss\_ConstitutionB += (Constitution\_SocreB[j] - mean\_ConstitutionB) \* (Constitution\_SocreB[j] - mean\_ConstitutionB)  
ss\_ConstitutionB = ss\_ConstitutionB/(len(Constitution\_SocreB) - 1)  
print("体能成绩的协方差是(按照C5重新计算后)：", ss\_ConstitutionB)  
std\_ConstitutionB = ss\_ConstitutionB \*\* 0.5  
print("体能成绩的标准差是(按照C5重新计算后)：", std\_ConstitutionB)  
Constitution\_SocreB1 = [] # 体能成绩数组B`  
for m in range(len(Constitution\_SocreB)):  
 Constitution\_SocreB1.append((Constitution\_SocreB[m] - mean\_ConstitutionB)/std\_ConstitutionB)  
  
# 计算C5  
C5\_Score = []  
for i in range(len(d)):  
 if d[i][9] != '' and d[i][15] != '': # 体能成绩有两处为空，要舍去  
 C5\_Score.append(float(d[i][9]))  
# 求协方差  
ss\_C5 = 0.0  
mean\_C5 = mean(C5\_Score) # C5成绩的平均值  
for i in range(len(C5\_Score)): # 协方差公式，先计算和  
 ss\_C5 += (C5\_Score[i] - mean\_C5) \* (C5\_Score[i] - mean\_C5)  
ss\_C5 = ss\_C5/(len(C5\_Score) - 1)  
std\_C5 = ss\_C5 \*\* 0.5  
print("C5成绩的平均值是：", mean\_C5)  
print("C5成绩的协方差是：", ss\_C5)  
print("C5成绩的标准差是：", std\_C5)  
C5\_Score1 = []  
for m in range(len(C5\_Score)): # C5成绩数组C5`  
 C5\_Score1.append((C5\_Score[m] - mean\_C5)/std\_C5)  
  
# 体能 成绩数组和C5成绩数组的相关性  
correlation\_Con\_C5 = 0.0  
for m in range(len(Constitution\_SocreB1)):  
 correlation\_Con\_C5 += Constitution\_SocreB1[m] \* C5\_Score[m]  
print("C5和体能测试成绩的相关性是（C5\*B)：", correlation\_Con\_C5)  
print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  
  
# 计算C6  
C6\_Score = []  
for i in range(len(d)):  
 if d[i][10] != '' and d[i][15] != '': # 体能成绩有两处为空，要舍去  
 C6\_Score.append(float(d[i][10]))  
# 求协方差  
ss\_C6 = 0.0  
mean\_C6 = mean(C6\_Score) # C6成绩的平均值  
for i in range(len(C6\_Score)): # 协方差公式，先计算和  
 ss\_C6 += (C6\_Score[i] - mean\_C6) \* (C6\_Score[i] - mean\_C6)  
ss\_C6 = ss\_C6/(len(C6\_Score) - 1)  
std\_C6 = ss\_C6 \*\* 0.5  
print("C6成绩的平均值是：", mean\_C6)  
print("C6成绩的协方差是：", ss\_C6)  
print("C6成绩的标准差是：", std\_C6)  
C6\_Score1 = []  
for m in range(len(C6\_Score)): # C6成绩数组C6`  
 C6\_Score1.append((C6\_Score[m] - mean\_C6)/std\_C6)  
  
# 体能 成绩数组和C6成绩数组的相关性  
correlation\_Con\_C6 = 0.0  
for m in range(len(Constitution\_Socre1)):  
 correlation\_Con\_C6 += Constitution\_Socre1[m] \* C6\_Score[m]  
print("C6和体能测试成绩的相关性是（C6\*B)：", correlation\_Con\_C6)  
print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  
  
# 计算C7  
C7\_Score = []  
for i in range(len(d)):  
 if d[i][11] != '' and d[i][15] != '': # 体能成绩有两处为空，要舍去  
 C7\_Score.append(float(d[i][11]))  
# 求协方差  
ss\_C7 = 0.0  
mean\_C7 = mean(C7\_Score) # C7成绩的平均值  
for i in range(len(C7\_Score)): # 协方差公式，先计算和  
 ss\_C7 += (C7\_Score[i] - mean\_C7) \* (C7\_Score[i] - mean\_C7)  
ss\_C7 = ss\_C7/(len(C7\_Score) - 1)  
std\_C7 = ss\_C7 \*\* 0.5  
print("C7成绩的平均值是：", mean\_C7)  
print("C7成绩的协方差是：", ss\_C7)  
print("C7成绩的标准差是：", std\_C7)  
C7\_Score1 = []  
for m in range(len(C7\_Score)): # C7成绩数组C7`  
 C7\_Score1.append((C7\_Score[m] - mean\_C7)/std\_C7)  
  
# 体能 成绩数组和C7成绩数组的相关性  
correlation\_Con\_C7 = 0.0  
for m in range(len(Constitution\_Socre1)):  
 correlation\_Con\_C7 += Constitution\_Socre1[m] \* C7\_Score[m]  
print("C7和体能测试成绩的相关性是（C7\*B)：", correlation\_Con\_C7)  
print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  
  
# 计算C8  
C8\_Score = []  
for i in range(len(d)):  
 if d[i][12] != '' and d[i][15] != '': # 体能成绩有两处为空，要舍去  
 C8\_Score.append(float(d[i][12]))  
# 求协方差  
ss\_C8 = 0.0  
mean\_C8 = mean(C8\_Score) # C8成绩的平均值  
for i in range(len(C8\_Score)): # 协方差公式，先计算和  
 ss\_C8 += (C8\_Score[i] - mean\_C8) \* (C8\_Score[i] - mean\_C8)  
ss\_C8 = ss\_C8/(len(C8\_Score) - 1)  
std\_C8 = ss\_C8 \*\* 0.5  
print("C8成绩的平均值是：", mean\_C8)  
print("C8成绩的协方差是：", ss\_C8)  
print("C8成绩的标准差是：", std\_C8)  
C8\_Score1 = []  
for m in range(len(C8\_Score)): # C8成绩数组C8`  
 C8\_Score1.append((C8\_Score[m] - mean\_C8)/std\_C8)  
  
# 体能 成绩数组和C8成绩数组的相关性  
correlation\_Con\_C8 = 0.0  
for m in range(len(Constitution\_Socre1)):  
 correlation\_Con\_C8 += Constitution\_Socre1[m] \* C8\_Score[m]  
print("C8和体能测试成绩的相关性是（C8\*B)：", correlation\_Con\_C8)  
print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  
  
# 计算C9  
C9\_Score = []  
for i in range(len(d)):  
 if d[i][13] != '' and d[i][15] != '': # 体能成绩有两处为空，要舍去  
 C9\_Score.append(float(d[i][13]))  
# 求协方差  
ss\_C9 = 0.0  
mean\_C9 = mean(C9\_Score) # C9成绩的平均值  
for i in range(len(C9\_Score)): # 协方差公式，先计算和  
 ss\_C9 += (C9\_Score[i] - mean\_C9) \* (C9\_Score[i] - mean\_C9)  
ss\_C9 = ss\_C9/(len(C9\_Score) - 1)  
std\_C9 = ss\_C9 \*\* 0.5  
print("C9成绩的平均值是：", mean\_C9)  
print("C9成绩的协方差是：", ss\_C9)  
print("C9成绩的标准差是：", std\_C9)  
C9\_Score1 = []  
for m in range(len(C9\_Score)): # C9成绩数组C9`  
 C9\_Score1.append((C9\_Score[m] - mean\_C9)/std\_C9)  
  
# 体能 成绩数组和C9成绩数组的相关性  
correlation\_Con\_C9 = 0.0  
for m in range(len(Constitution\_Socre1)):  
 correlation\_Con\_C9 += Constitution\_Socre1[m] \* C9\_Score[m]  
print("C9和体能测试成绩的相关性是（C9\*B)：", correlation\_Con\_C9)  
print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

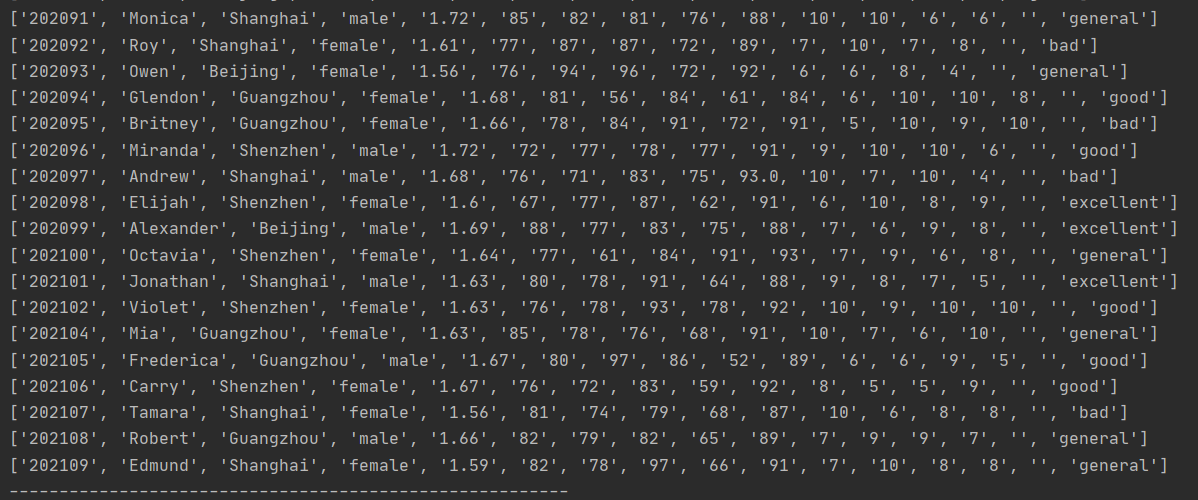
运行结果截图：

整合后的数据（也保存在Final\_Data.xlsx中）

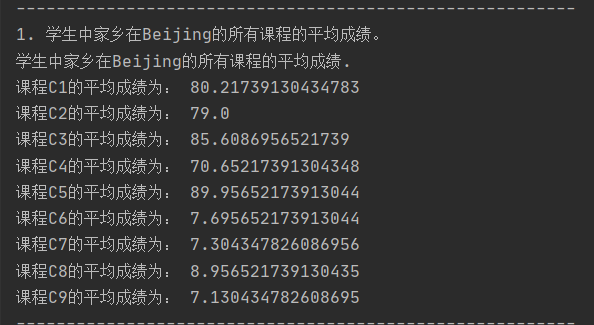




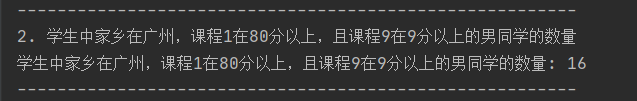




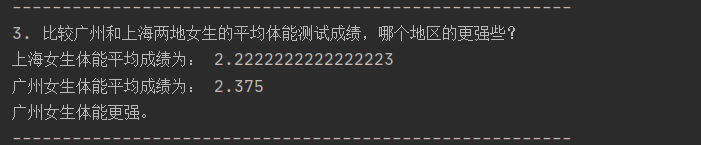
第一题：



第二题：



第三题：



第四题：

