`

FACULTY OF ENGINEERING

SCHOOL OF COMPUTING

SEMESTER 1/20202021

SCSP3223-02 PENGATURCARAAN DATA ANALITIK (DATA ANALYTICS PROGRAMMING)

LECTURER: DR. CHAN WENG HOWE

SECTION: 02

Group Project :

STUDENTS' PERFORMANCE IN EXAMS

GROUP MEMBERS:

NAME	MATRIC NO
CHUA HUN HO	A18CS0050
JASMINE CHAN YUAN QI	A18CS0083
KHOO JIE XUAN	A18CS0091

A. Find a dataset which contains enough data to practice data preparation and analysis (at least 1000 rows)

The dataset chosen for this project is about 'Student Performance in Exam' and it can be found on the Kaggle website. This dataset recorded marks secured by the students in high school students from the United States. The aim of this dataset is to understand the factors that influences students' performance in their exam based on the given variables:

Column name	Description
Gender	Gender of each students
Race/ethnicity	5 different ethnic group of students (A,B,C,D,E)
Parental level of education	Type of education level for each student's parents
Lunch	Amount of food taken during lunch for each students before exam started
Test preparation course	Degree of course preparation before test
Math score	Students' score in Math subject
Reading score	Students' score in reading subject
Writing score	Students' score in writing subject

This dataset is suitable for data analysis and machine learning as it contains the right amount of data needed for this project which are 1000 rows.

B. Formulate research question(s) from the dataset. What do you want to present?

The following are the research questions that we focused on:

- Which major factors contribute to test outcomes?
- What is the distribution of the average score of students?
- Is there any correlation between scores in each subject?
- What would be the best way to improve student scores on each test?
- Will economic background have any impact on a student's performance?

We will be using the techniques of data visualization and machine learning to help us to explore the research questions mentioned above.

C. Data Cleaning, Preparation and Wrangling

The libraries such as pandas, numpy and matplotlib are imported into this python notebook. The data source of 'StudentPerformance.csv' is read by using the function of read_csv() and stored into a dataframe named as 'record'. The first 10 rows of records are shown by using the function of head().

```
[1]: import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
```

Data preparation and cleaning

```
[2]: record = pd.read_csv('StudentsPerformance.csv')
     record.head(10)
```

[2]:

	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

The process of data cleaning and preparation is started with changing the column name of the dataframe. This is because there are some spaces between words in the column name and there is a special character '/' in the column of race/ethnicity. After going through this process, the column names are standardized.

```
In [3]: # Change column name
record = record.rename(
    columns =
    {   "gender":"Gender",
        "race/ethnicity":"Race",
        "parental level of education":"Parental_education_level",
        "lunch":"Lunch",
        "test preparation course":"Preparation_Course",
        "math score":"Math_score",
        "reading score":"Reading_score",
        "writing score":"Writing_score"}).copy()
record.head()
```

Out[3]:

	Gender	Race	Parental_education_level	Lunch	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

The process is continued by formatting the value of gender and race. This is implemented by using the replace function to change the values, i.e. from "female" to "F" and from "male" to "M". For the column of race, the value of "Group A" is being replaced with "A" same goes to the rest of the group.

```
In [4]: #Replace value of gender and race
record["Gender"].replace({"female":"F","male":"M"},inplace=True)
record["Race"].replace({"group A":"A","group B":"B","group C":"C","group D":"D","group E":"E"},inplace=True)
record.head()
```

Out[4]:

	Gender	Race	Parental_education_level	Lunch	Preparation_Course	Math_score	Reading_score	Writing_score
0	F	B	bachelor's degree	standard	none	72	72	74
1	F	C	some college	standard	completed	69	90	88
2	F	B	master's degree	standard	none	90	95	93
3	M	A	associate's degree	free/reduced	none	47	57	44
4	M	C	some college	standard	none	76	78	75

The function of value_counts is used to count the frequency of each column value. This is used to see if there is any incorrect value in the columns. From the result, we can say that all the values are correct.

```
In [5]: #List value of each column
        for i in list(record.columns[:5]):
            print("{} Column \n".format(i),record[i].value_counts(),end="\n\n",sep="")

        Gender Column
        F    518
        M    482
        Name: Gender, dtype: int64

        Race Column
        C    319
        D    262
        B    190
        E    140
        A     89
        Name: Race, dtype: int64

        Parental_education_level Column
        some college          226
        associate's degree    222
        high school           196
        some high school      179
        bachelor's degree     118
        master's degree        59
        Name: Parental_education_level, dtype: int64

        Lunch Column
        standard        645
        free/reduced    355
        Name: Lunch, dtype: int64

        Preparation_Course Column
        none         642
        completed    358
        Name: Preparation_Course, dtype: int64
```

The function of dropna is used to filter axis labels based on whether values for each label have missing data. From the output, we can say that there is no null or empty data in our dataset.

```
In [6]: #Drop if any empty row exists
        df = record.dropna()
        df
```

Out[6]:

	Gender	Race	Parental_education_level	Lunch	Preparation_Course	Math_score	Reading_score	Writing_score
0	F	B	bachelor's degree	standard	none	72	72	74
1	F	C	some college	standard	completed	69	90	88
2	F	B	master's degree	standard	none	90	95	93
3	M	A	associate's degree	free/reduced	none	47	57	44
4	M	C	some college	standard	none	76	78	75
...	...	...	...	...	...	...	...	...
995	F	E	master's degree	standard	completed	88	99	95
996	M	C	high school	free/reduced	none	62	55	55
997	F	C	high school	free/reduced	completed	59	71	65
998	F	D	some college	standard	completed	68	78	77
999	F	D	some college	free/reduced	none	77	86	86

1000 rows × 8 columns

The function of isnull is used for data preparation to ensure that there is no null or empty data. From the output, we can clearly see that all the sum of missing values for each column is zero, which means there is no missing data.

```
In [7]: #Check for null/empty row
        record.isnull().sum()

Out[7]: Gender                     0
        Race                       0
        Parental_education_level   0
        Lunch                      0
        Preparation_Course         0
        Math_score                 0
        Reading_score              0
        Writing_score              0
        dtype: int64
```

For the process of data preparation, a new column is added which is being named as "Average_score". This column is created by using the mean of three scores which are "Math_score", "Reading_score" and "Writing_score".

```
In [8]: df['Average_score'] = df[['Math_score', 'Reading_score', 'Writing_score']].mean(axis=1)
        df.head()

Out[8]:
```

	Gender	Race	Parental_education_level	Lunch	Preparation_Course	Math_score	Reading_score	Writing_score	Average_score
0	F	B	bachelor's degree	standard	none	72	72	74	72.666667
1	F	C	some college	standard	completed	69	90	88	82.333333
2	F	B	master's degree	standard	none	90	95	93	92.666667
3	M	A	associate's degree	free/reduced	none	47	57	44	49.333333
4	M	C	some college	standard	none	76	78	75	76.333333

D. Data Aggregation and Group Operations

For data aggregation and group operations, we decided to group the average of each score by the factors that might affect a student's score which are gender, race, lunch, parental education level and their completeness of the preparation course. We have called GroupBy's mean method and grouped the score list by each factor.

Data Aggregation and Group Operations

Score list are grouped by each factor

In [9]:
```
# Factor: Gender
Score_list = ['Math_score','Reading_score','Writing_score','Average_score']

factor_gender = df[Score_list].groupby(df['Gender']).mean()
factor_gender
```

Out[9]:

Gender	Math_score	Reading_score	Writing_score	Average_score
F	63.633205	72.608108	72.467181	69.569498
M	68.728216	65.473029	63.311203	65.837483

In [10]:
```
# Factor: Race

factor_race = df[Score_list].groupby(df['Race']).mean()
factor_race
```

Out[10]:

Race	Math_score	Reading_score	Writing_score	Average_score
A	61.629213	64.674157	62.674157	62.992509
B	63.452632	67.352632	65.600000	65.468421
C	64.463950	69.103448	67.827586	67.131661
D	67.362595	70.030534	70.145038	69.179389
E	73.821429	73.028571	71.407143	72.752381

In [11]:
```
# Factor: Parental_education_level

factor_edu = df[Score_list].groupby(df['Parental_education_level']).mean()
factor_edu
```

Out[11]:

Parental_education_level	Math_score	Reading_score	Writing_score	Average_score
associate's degree	67.882883	70.927928	69.896396	69.569069
bachelor's degree	69.389831	73.000000	73.381356	71.923729
high school	62.137755	64.704082	62.448980	63.096939
master's degree	69.745763	75.372881	75.677966	73.598870
some college	67.128319	69.460177	68.840708	68.476401
some high school	63.497207	66.938547	64.888268	65.108007

In [12]:
```
# Factor: Lunch

factor_lunch = df[Score_list].groupby(df['Lunch']).mean()
factor_lunch
```

Out[12]:

Lunch	Math_score	Reading_score	Writing_score	Average_score
free/reduced	58.921127	64.653521	63.022535	62.199061
standard	70.034109	71.654264	70.823256	70.837209

In [13]:
```
# Factor: Preparation_Course

factor_pre = df[Score_list].groupby(df['Preparation_Course']).mean()
factor_pre
```

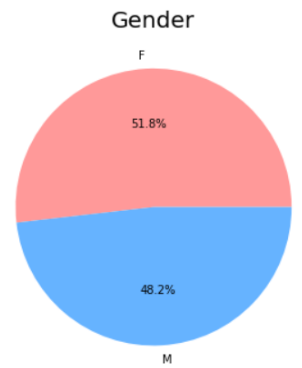
Out[13]:

Preparation_Course	Math_score	Reading_score	Writing_score	Average_score
completed	69.695531	73.893855	74.418994	72.669460
none	64.077882	66.534268	64.504673	65.038941

E. Visualize your analysis using appropriate visualization.

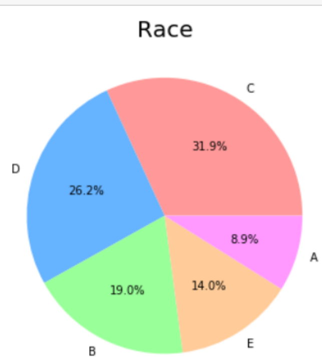
The pie chart is plotted for data visualization purposes. We have used Matplotlib API which has a pie() function in its pyplot module which creates a pie chart representing the data in an array. From the chart below, we can see that the majority of the students are female which are 51.8% of all students.

```
In [14]: colors = ['#ff9999','#66b3ff','#99ff99','#ffcc99','#ff99ff','#ffb266']

         plt.figure(figsize=(30,10))
         plt.subplots_adjust(left=0.125, bottom=0.1, right=0.9, top=0.9,
                             wspace=0.5, hspace=0.2)
         plt.subplot(141)
         plt.axis('off')
         plt.title('Gender',fontsize = 20)
         df['Gender'].value_counts().plot.pie(autopct="%1.1f%%",colors=colors)
         plt.show()
```



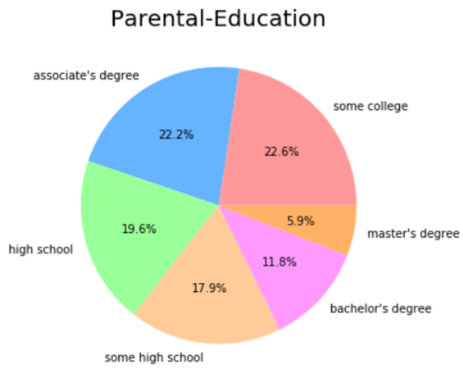
From the chart below, we can conclude that the majority of the students belong to group C's race and that consists of 31.9%. However, the minority race among the students is group A which is 8.9%.

```
In [15]: plt.figure(figsize=(30,10))
         plt.subplots_adjust(left=0.125, bottom=0.1, right=0.9, top=0.9,
                             wspace=0.5, hspace=0.2)
         plt.subplot(142)
         plt.axis('off')
         plt.title('Race',fontsize = 20)
         df['Race'].value_counts().plot.pie(autopct="%1.1f%%",colors=colors)
         plt.show()
```



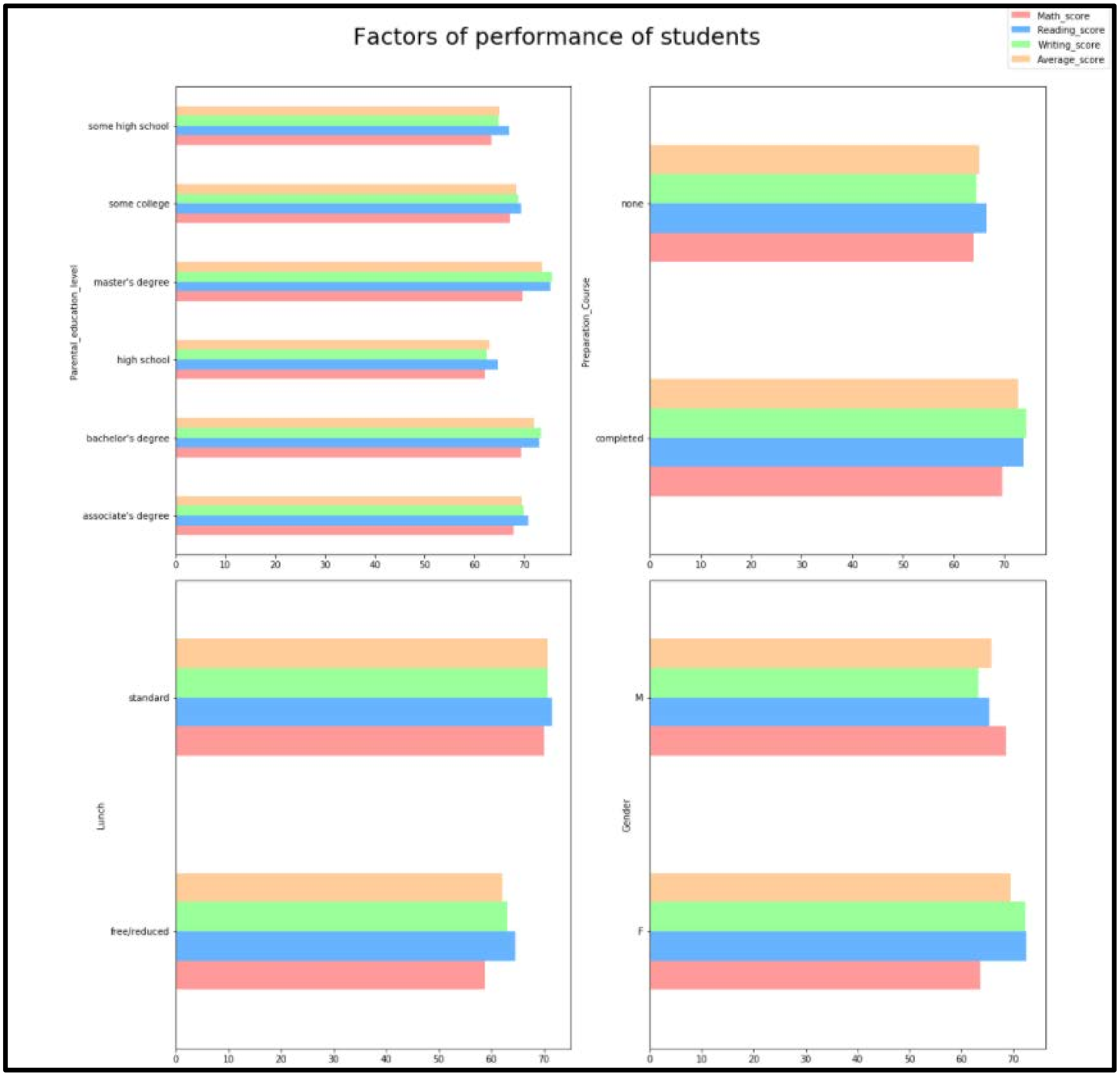
From the chart below, we can say that the largest portion of this pie chart shows that students that their parental education level is some college (22.6%), followed by those whose parental education level is associate's degree (22.2%). The least amount of students' parental education level is master's degree (5.9%).

```
In [16]: plt.figure(figsize=(30,10))
         plt.subplots_adjust(left=0.125, bottom=0.1, right=0.9, top=0.9,
                             wspace=0.5, hspace=0.2)
         plt.subplot(141)
         plt.axis('off')
         plt.title('Parental-Education',fontsize = 20)
         df['Parental_education_level'].value_counts().plot.pie(autopct="%1.1f%%",colors=colors)
         plt.show()
```



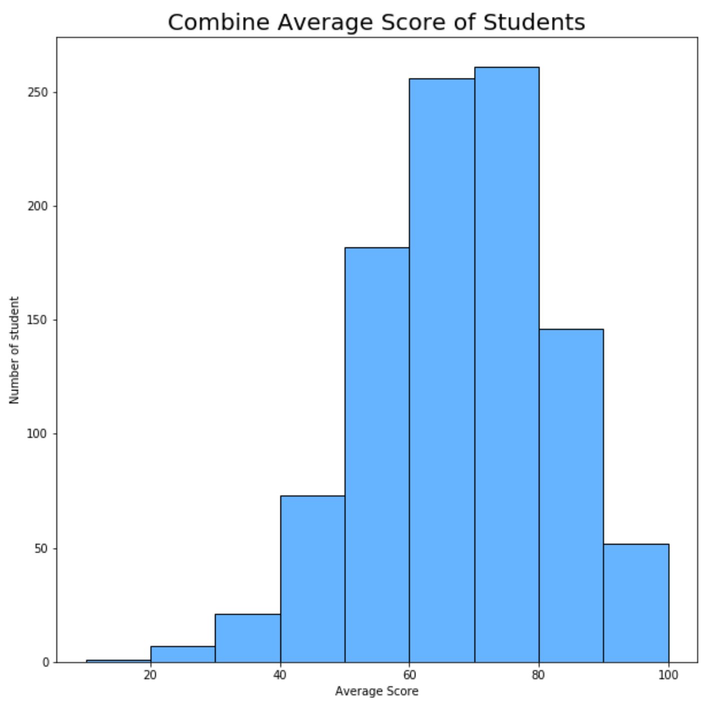
We have used the data we get from the function of groupBy to plot the following horizontal bar charts. These charts show the average score of math, reading, writing and mean score of these three topics with other variables which are gender, lunch, parental educational level and preparation course. By plotting these bar charts, we can find out how these factors will affect the student's performance. In short, we can say that the female students with parents who have a master's degree, completed their preparation course and have standard lunch usually have better performance compared to other students. Thus, we can say that these might be the factors that affect student's performance. The difference is more obvious on the factor lunch which students with standard lunch have a higher average score compared to those who have free or reduced lunch. This is because lunch might be representing their family's financial status.

```
In [62]: fig = plt.figure(figsize=(15,15))
         axes = fig.subplots(nrows=2, ncols=2)
         factor_edu.plot(kind = 'barh',ax=axes[0,0],color=colors,legend=False)
         factor_pre.plot(kind = 'barh',ax=axes[0,1],color=colors,legend=False)
         factor_lunch.plot(kind = 'barh',ax=axes[1,0],color=colors,legend=False)
         factor_gender.plot(kind = 'barh',ax=axes[1,1],color=colors,legend=False)

         fig.suptitle('Factors of performance of students', y=1.05,fontsize=25)
         lines, labels = fig.axes[-1].get_legend_handles_labels()
         fig.legend(lines, labels, loc = 'lower center',bbox_to_anchor=(1, 1))
         fig.tight_layout()
         plt.show()
```


Factors of performance of students
Math_score
Reading_score
Writing_score
Average_score
Parental_education_level
some high school
some college
master's degree
high school
bachelor's degree
associate's degree
Preparation_Course
none
completed
Lunch
standard
free/reduced
Gender
M
F
0
10
20
30
40
50
60
70

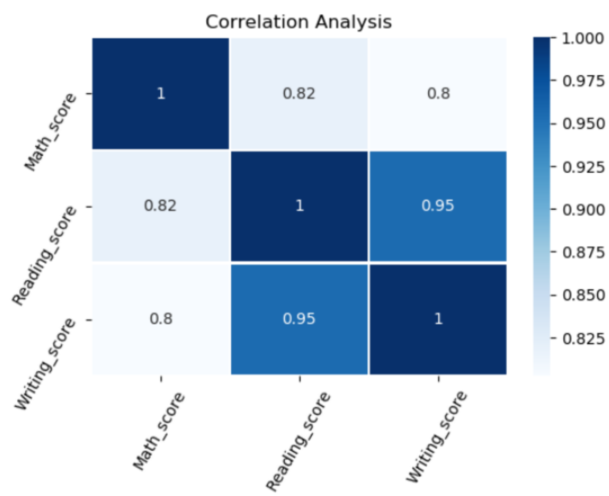
A histogram chart is plotted by using the function of matplotlib.hist(). For this dataset, the lowest range of average score is about 10-20 whereas the highest is 90 - 100. More than 25% of students are able to score an average score in a range of 70-80. Average scores between 60 and 80 were quite frequent. The distribution of this histogram is left-skewed because most of the sample values are clustered on the left side of histogram.

```
In [48]: plt.figure(figsize=(10,10))
         data = df['Average_score']
         bins = [10,20,30,40,50,60,70,80,90,100]
         plt.hist(data, color = '#66b3ff', edgecolor = 'black',bins=bins)

         plt.title('Combine Average Score of Students',fontsize="20")
         plt.xlabel('Average Score')
         plt.ylabel('Number of student')
         plt.show()
```



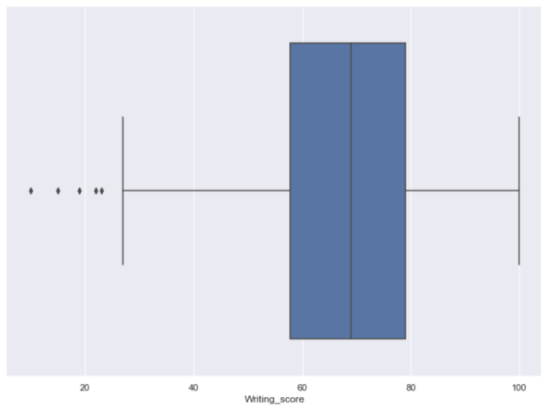
A heat map is designed by using the function of seaborn.heatmap(). From this analysis, the correlation coefficient between Reading_score and Writing_score is the highest which is up to 0.95. This shows that the relationship between reading and writing has the strongest relationship. Thus, we believe that the more a student reads, the better his/her writing. Although the correlation coefficient between Math_score and Writing_score is the lowest (0.80) in this heatmap, the value of correlation coefficient shows the relationship is significant because the value of coefficient is close to 1.

In [19]:
```
plt.figure(dpi=100)
plt.title('Correlation Analysis')
sns.heatmap(record.corr(),annot=True,lw=1,linecolor='white',cmap='Blues')
plt.xticks(rotation=60)
plt.yticks(rotation = 60)
plt.show()
```



There are three different themes of box plots. The first box plot analyses the score of writing among 1000 students. From this boxplot, there are about 25% of students score lower than 57.75 and about 75% of students score higher than 79 in this dataset. The highest score of writing is 100 whereas the lowest score of that is 10.

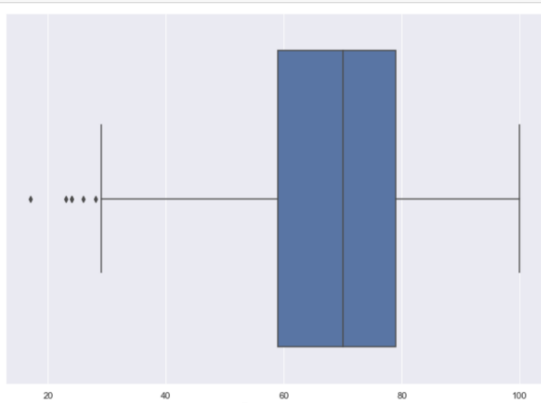
In [70]:
```
#Plotting boxplot of writing score
sns.boxplot(x='Writing_score',data=record)
plt.show()
df['Writing_score'].describe()
```



Out[70]:
```
count    1000.000000
mean       68.054000
std        15.195657
min        10.000000
25%        57.750000
50%        69.000000
75%        79.000000
max       100.000000
Name: Writing_score, dtype: float64
```

Next, the second box plot is about the score of reading among 1000 students. There are about 25% of students (250 students) score lower than 59.0 and about 75% of students (750 students) score higher than 79.0 in this dataset. The highest average score of reading is 100 whereas the lowest average score of that is 17.

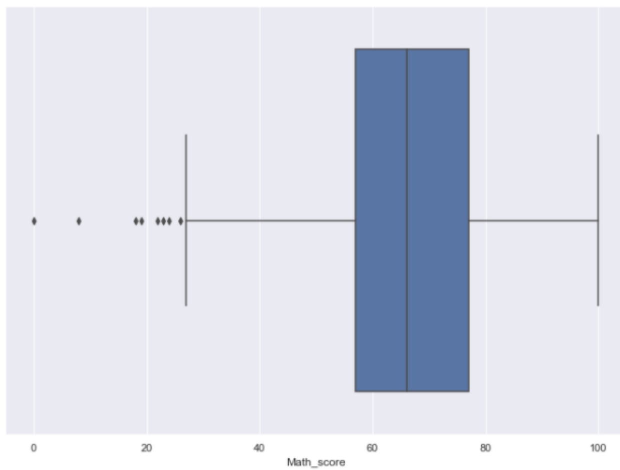
In [71]:
```
#Plotting boxplot of reading score
sns.boxplot(x='Reading_score',data=record)
plt.show()
df['Reading_score'].describe()
```



Out[71]:
```
count    1000.000000
mean       69.169000
std        14.600192
min        17.000000
25%        59.000000
50%        70.000000
75%        79.000000
max       100.000000
Name: Reading_score, dtype: float64
```

For the third boxplot, the score of mathematics among 1000 students is analyzed. About 25% of students score below than 57 marks and about 75% of students are able to score higher than 77 marks. Among 1000 students, the lowest score is 0 whereas the highest score is 100 for ‘Math_score’.

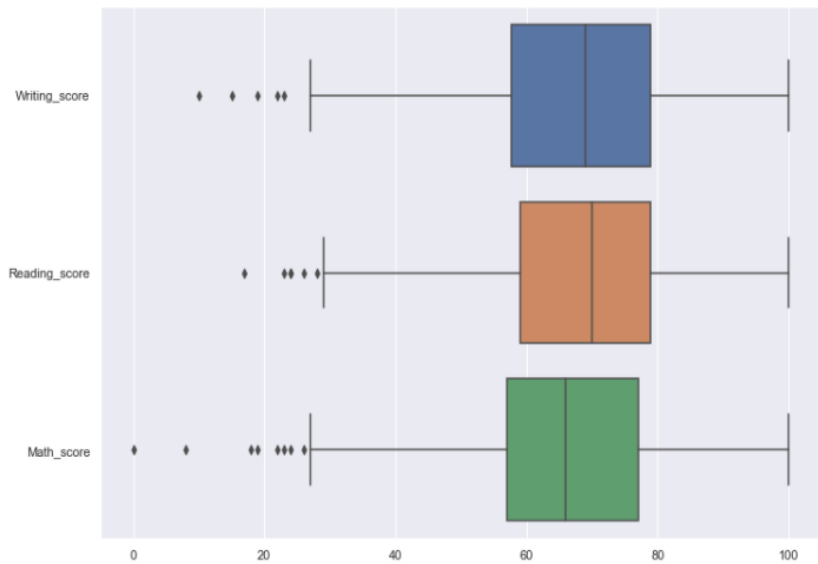
```
In [72]: #Plotting boxplot of math score
         sns.boxplot(x='Math_score',data=record)
         plt.show()
         df['Math_score'].describe()
```




```
Out[72]: count    1000.00000
         mean       66.08900
         std        15.16308
         min         0.00000
         25%        57.00000
         50%        66.00000
         75%        77.00000
         max       100.00000
         Name: Math_score, dtype: float64
```

By comparing these three boxplots, we observe that potential outliers exist in all of these boxplots. Moreover, we found out that most of the students are able to score better marks in reading by comparing the median. The length of the box for all of the subjects are roughly similar. The median of ‘Math_score’ is closer to the bottom of the box, this shows that the distribution of ‘Math_score’ is slightly right-skewed whereas that of ‘Reading_score’ is closer to the top of the box and this shows that the distribution of ‘Reading_score’ is slightly left-skewed.

```
In [63]: df1 = df['Writing_score']
         df2 = df['Reading_score']
         df3 = df['Math_score']

         cdf = pd.concat([df1,df2,df3],axis=1)
         sns.set(rc={'figure.figsize':(11.7,8.27)})
         sns.boxplot(orient='h',data=cdf)
         df['Math_score'].describe()

Out[63]: count    1000.00000
         mean       66.08900
         std        15.16308
         min         0.00000
         25%        57.00000
         50%        66.00000
         75%        77.00000
         max       100.00000
         Name: Math_score, dtype: float64
```



F. Machine Learning

For machine learning, we decided to proceed with logistic regression which is a machine learning classification algorithm. It uses a different method for estimating the parameters, which gives better results–better meaning unbiased, with lower variances. The logistic regression is implemented to find out the relationship between a student's economic background with their academic performance.

The function of pandas.DataFrame.iloc() is used to allocate and extract the column we want. In this case, the values of the entered variable 'x' that are used are ‘Math_score’, ‘Reading_score’ and‘Writing_score’.

```
In [21]: x = record.iloc[:, -3:]
         x
```

Out[21]:

	Math_score	Reading_score	Writing_score
0	72	72	74
1	69	90	88
2	90	95	93
3	47	57	44
4	76	78	75
...	...	...	...
995	88	99	95
996	62	55	55
997	59	71	65
998	68	78	77
999	77	86	86

1000 rows × 3 columns

For the variable 'y', we use the column 'Lunch' as the resulting variable.

```
In [22]: y = record.iloc[:, 3:4]
         y
```

Out[22]:

	Lunch
0	standard
1	standard
2	standard
3	free/reduced
4	standard
...	...
995	standard
996	free/reduced
997	free/reduced
998	standard
999	free/reduced

1000 rows × 1 columns

The function of train_test_split is used to split the data into two parts which are for the use of training data and testing data respectively. The size of testing data is being set to 0.33 which is 33.33% of the dataset. We also did a scaling of data by training and transforming from x_train and x_test. After that, the logistic regression classifier is implemented by building the model with the function of LogisticRegression().

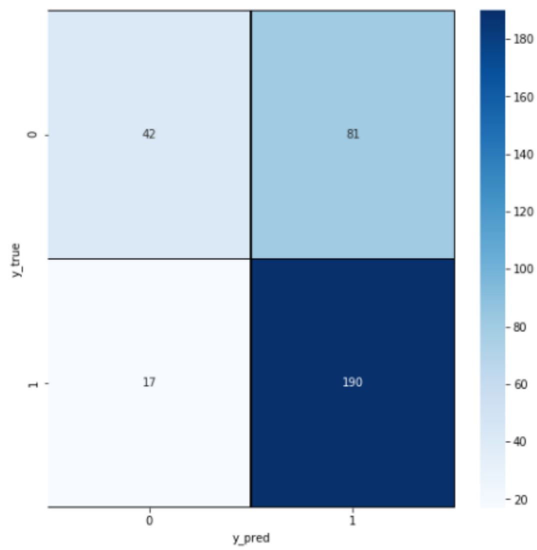
```
In [23]: from sklearn.model_selection import train_test_split
         x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.33, random_state=0)

In [24]: from sklearn.preprocessing import StandardScaler
         sc = StandardScaler()
         X_train = sc.fit_transform(x_train)  # training and transforming from x_train
         X_test = sc.transform(x_test)    # only transforming from x_test

In [25]: from sklearn.linear_model import LogisticRegression
         log_reg = LogisticRegression(random_state=0)
         log_reg.fit(X_train, y_train)

         C:\Users\MR.COOL\anaconda3\lib\site-packages\sklearn\utils\validation.py:760: DataConversionWarning: A column-vector y was pass
         ed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
           y = column_or_1d(y, warn=True)

Out[25]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                            intercept_scaling=1, l1_ratio=None, max_iter=100,
                            multi_class='auto', n_jobs=None, penalty='l2',
                            random_state=0, solver='lbfgs', tol=0.0001, verbose=0,
                            warm_start=False)
```

Here is the process of prediction of the testing data. In this case, the function *predict()* is used to undergo the prediction on the 'X_test' which is the testing set. Then, this function returns an array of predicted values of the column 'Lunch'.

```
In [26]: y_pred = log_reg.predict(X_test)
         y_pred

Out[26]: array(['standard', 'standard', 'free/reduced', 'standard', 'standard',
                'standard', 'standard', 'free/reduced', 'standard', 'standard',
                'standard', 'free/reduced', 'standard', 'standard', 'free/reduced',
                'free/reduced', 'standard', 'standard', 'standard', 'standard',
                'standard', 'free/reduced', 'standard', 'standard', 'standard',
                'standard', 'standard', 'standard', 'free/reduced', 'standard',
                'standard', 'standard', 'standard', 'free/reduced', 'free/reduced',
                'standard', 'standard', 'standard', 'standard', 'standard',
                'standard', 'standard', 'standard', 'free/reduced', 'standard',
                'standard', 'standard', 'standard', 'standard', 'free/reduced',
                'free/reduced', 'standard', 'free/reduced', 'standard',
                'free/reduced', 'standard', 'free/reduced', 'standard', 'standard',
                'standard', 'standard', 'standard', 'free/reduced', 'standard',
                'standard', 'standard', 'standard', 'free/reduced', 'standard',
                'standard', 'standard', 'standard', 'standard', 'standard',
                'standard', 'standard', 'standard', 'standard', 'free/reduced',
                'standard', 'standard', 'standard', 'standard', 'standard',
                'standard', 'standard', 'standard', 'standard', 'standard',
                'free/reduced', 'standard', 'standard', 'free/reduced', 'standard',
                'standard', 'standard', 'standard', 'free/reduced', 'standard',
                'standard', 'free/reduced', 'standard', 'standard', 'free/reduced',
                'standard', 'standard', 'standard', 'free/reduced', 'free/reduced',
                'standard', 'standard', 'standard', 'free/reduced', 'free/reduced',
                'standard', 'standard', 'standard', 'standard', 'standard',
                'standard', 'free/reduced', 'standard', 'standard', 'standard',
                'standard', 'free/reduced', 'standard', 'standard', 'standard',
                'standard', 'standard', 'standard', 'free/reduced', 'standard',
                'standard', 'standard', 'standard', 'standard', 'standard',
                'standard', 'standard', 'standard', 'standard', 'standard',
                'standard', 'free/reduced', 'free/reduced', 'free/reduced',
                'standard', 'standard', 'standard', 'standard', 'standard',
                'standard', 'standard', 'standard', 'standard', 'free/reduced',
                'free/reduced', 'standard', 'standard', 'free/reduced',
```

Confusion matrix is used to describe the performance of the classification model. True negative (tn) and true positive (tp) means prediction is negative and actual is negative, prediction is positive and actual is positive respectively while false negative (fn) and false positive (fp) means prediction is negative and actual is positive, prediction is positive and actual is negative respectively. Therefore, the accuracy is calculated by dividing the sum of tn and tp (correct prediction) with length of y_test. We managed to get the accuracy of 0.70 which means that the percentage of correct classification of the model is around 70%. Moreover, we also plotted a heat map to show the result of the confusion matrix.

```
In [27]: from sklearn.metrics import confusion_matrix
         result = confusion_matrix(y_test, y_pred)
         tn, fp, fn, tp = confusion_matrix(y_test, y_pred).ravel()
         (tn, fp, fn, tp)

Out[27]: (42, 81, 17, 190)

In [28]: accuracy = (tn+tp)/len(y_test)
         accuracy

Out[28]: 0.703030303030303

In [29]: f, ax = plt.subplots(figsize =(8,8))
         sns.heatmap(result,annot = True,linewidths=0.5,linecolor="black",fmt = ".0f",ax=ax,cmap='Blues')
         plt.xlabel("y_pred")
         plt.ylabel("y_true")
         plt.show()
```



Conclusion

In conclusion, we found that the major factors that contribute to the performance of students are lunch and their completeness on preparation courses. This is said because we can observe that from the four horizontal bar charts and it shows that there are significant differences between students who completed the preparation course and took full lunch compared to those students who only reduced lunches and haven’t completed their preparation course. The distribution of the average score of students is left-skewed due to more sample data located at the left-side of histogram. There is a correlation between scores in each subject but the most significant correlation is between writing and reading. The best ways to improve their performance are having a standard lunch and also getting some preparation of course before taking the test. From the horizontal bar charts, we can also observe that overall the students who are able to have a standard lunch can perform better in the test compared with students who have reduced or free lunch. From the part of machine learning, we obtained up to 70% of accuracy on predicting the types of lunch students have based on their results. This result further confirms the statement of the economic background of students will affect their performance.