# Get the 'student' column from the DataFrame. What is the data type of the returned object?

grades\_df['student']

# Get the 'grades' column from the DataFrame and store in a variable. What is the data type of the returned object?

temp = grades\_df['grade']

type(temp)

# Get the count of non-null values using the function 'count()'

grades\_df.count()

# # check which rows are duplicated, then count them up

duplicates = grades\_df.duplicated()

duplicates\_sum = duplicates.sum()

# drop the duplicated rows then count the rows to make sure they've been deleted.

grades\_df = grades\_df.drop\_duplicates()

grades\_df.count()

# Does any of the columns contain null values(use the isna() function)? If yes:

# 1) isolate the column and check its entries

grades\_df.isnull().sum()

grades\_df['subject']

# 2) use the loc attribute to isolate the grades of the student with the missing value

grades\_df.loc[grades\_df['student'] == 'David']

# 3) can you check what the value of null cell should have been? If yes, fill that cell with the correct value using the loc attribute, and then verify if it was filled correctly.

# grades\_df['subject'].unique()

grades\_df.loc[21, 'subject'] = 'art'

grades\_df.loc[grades\_df['student'] == 'David']

# check to see if any grades are wrong, i.e. less than 0 or greater than 100. Correct the cell values if possible

grades\_df.loc[(grades\_df['grade'] <0) | (grades\_df['grade'] >100), 'grade']

grades\_df.loc[5, 'grade'] = 85

grades\_df.loc[(grades\_df['grade'] <0) | (grades\_df['grade'] >100), 'grade'].count()

# print out the student names without replication.

students = grades\_df['student'].unique()

students\_count = len(students)

students\_count

# print out the student subjects without replication.

subjects = grades\_df['subject'].unique()

subjects\_count = len(subjects)

subjects\_count

# group the grades by subject and get the lowest, highest, and the average grade.

min\_grades\_by\_subject= grades\_df.groupby('subject').min()['grade']

max\_grades\_by\_subject= grades\_df.groupby('subject').max()['grade']

average\_grades\_by\_subject =grades\_df.groupby('subject').mean()['grade']

subject\_aggregates = pd.DataFrame({'lowest\_grade':min\_grades\_by\_subject, 'highest\_grade':max\_grades\_by\_subject, "average\_grade":average\_grades\_by\_subject})

subject\_aggregates

# group the grades by student and get the lowest grade, highest grade, and the GPA.

min\_grades\_by\_student= grades\_df.groupby('student').min()['grade']

max\_gradesby\_student= grades\_df.groupby('student').max()['grade']

GPAs =grades\_df.groupby('student').mean()['grade']

student\_aggregates = pd.DataFrame({'lowest\_grade':min\_grades\_by\_student, 'highest\_grade':max\_gradesby\_student, "GPA":GPAs})

student\_aggregates

# create a chart using the 'subject\_aggregates' DataFrame, and set the chart kind to 'bar'

subject\_aggregates.plot(kind="bar")

# study the grade correlation between each pair of subjects

math\_grades = grades\_df[grades\_df['subject'] == 'math']['grade'].reset\_index(drop=True)

biology\_grades = grades\_df[grades\_df['subject'] == 'biology']['grade'].reset\_index(drop=True)

art\_grades = grades\_df[grades\_df['subject'] == 'art']['grade'].reset\_index(drop=True)

math\_and\_biology = math\_grades.corr(biology\_grades)

math\_and\_art = math\_grades.corr(art\_grades)

biology\_and\_art = biology\_grades.corr(art\_grades)

correlation\_df = pd.DataFrame([{'math\_and\_biology':math\_and\_biology,'math\_and\_art':math\_and\_art, 'biology\_and\_art':biology\_and\_art}])

correlation\_df