My First Data Project

```
Team Member(s): Ryann Alvarez
         Class: CISD-41
     Professor: Dr. Sohair Zaki
  Last Updated: 05/31/2025
```

In this Data Project, I will be using a student performance dataset from Kaggle (https://www.kaggle.com/datasets/devansodariya/student-performance-data/data).

This project explores the factors (e.g., demographic, behavioral, academic) that influence student performance using descriptive statistics, data visualizatons, and hypothesis testing learned in class.

Questions

1. What is the demographic breakdown of the students?
2. Are there differences in academic performance based on demographic variables?
3. How are students' first and second period grades related to their final grade?
4. How does parental education level relate to final grades?
5. Does access to family educational support, extra educational support, or paid classes improve student performance?
6. How much do students study each week, and how is that related to their grades?
7. Do social behaviors like going out and alcohol consumption impact final grades?
8. Are students who have internet access or want higher education performing better academically?
9. What are the most important predictors of student success?
10. What are the differences between high-performing and low-performing students in terms of study time, support, social habits, and family background?

Data and Setup

```python
print('Done by Ryann Alvarez')

# Import required libraries
import numpy as np
import pandas as pd

# Hide warnings, if needed
import warnings
warnings.filterwarnings('ignore')

# Set up dataset as a Pandas DataFrame
```

```python
student_df = pd.read_csv('data/student_data.csv')

# Preview of the data (head)
student_df.head(10)
```

```
Done by Ryann Alvarez

  school sex  age address famsize Pstatus  Medu  Fedu      Mjob
Fjob  \
0     GP   F   18       U     GT3       A     4     4   at_home
teacher
1     GP   F   17       U     GT3       T     1     1   at_home
other
2     GP   F   15       U     LE3       T     1     1   at_home
other
3     GP   F   15       U     GT3       T     4     2    health
services
4     GP   F   16       U     GT3       T     3     3     other
other
5     GP   M   16       U     LE3       T     4     3  services
other
6     GP   M   16       U     LE3       T     2     2     other
other
7     GP   F   17       U     GT3       A     4     4     other
teacher
8     GP   M   15       U     LE3       A     3     2  services
other
9     GP   M   15       U     GT3       T     3     4     other
other

   ... famrel freetime  goout  Dalc  Walc health absences  G1  G2  G3

0  ...      4        3      4     1     1      3        6   5   6   6

1  ...      5        3      3     1     1      3        4   5   5   6

2  ...      4        3      2     2     3      3       10   7   8  10

3  ...      3        2      2     1     1      5        2  15  14  15

4  ...      4        3      2     1     2      5        4   6  10  10

5  ...      5        4      2     1     2      5       10  15  15  15

6  ...      4        4      4     1     1      3        0  12  12  11

7  ...      4        1      4     1     1      1        6   6   5   6

8  ...      4        2      2     1     1      1        0  16  18  19

9  ...      5        5      1     1     1      5        0  14  15  15
```

```
[10 rows x 33 columns]

print('Done by Ryann Alvarez')

# Preview of the data (tail)
student_df.tail(10)

Done by Ryann Alvarez

    school sex  age address famsize Pstatus  Medu  Fedu      Mjob
Fjob  \
385     MS   F   18       R     GT3       T     2     2   at_home
other
386     MS   F   18       R     GT3       T     4     4   teacher
at_home
387     MS   F   19       R     GT3       T     2     3  services
other
388     MS   F   18       U     LE3       T     3     1   teacher
services
389     MS   F   18       U     GT3       T     1     1     other
other
390     MS   M   20       U     LE3       A     2     2  services
services
391     MS   M   17       U     LE3       T     3     1  services
services
392     MS   M   21       R     GT3       T     1     1     other
other
393     MS   M   18       R     LE3       T     3     2  services
other
394     MS   M   19       U     LE3       T     1     1     other
at_home

     ... famrel freetime  goout  Dalc  Walc health absences  G1  G2
G3
385  ...      5        3      3     1     3      4        2  10   9
10
386  ...      4        4      3     2     2      5        7   6   5
6
387  ...      5        4      2     1     2      5        0   7   5
0
388  ...      4        3      4     1     1      1        0   7   9
8
389  ...      1        1      1     1     1      5        0   6   5
0
390  ...      5        5      4     4     5      4       11   9   9
9
391  ...      2        4      5     3     4      2        3  14  16
16
```

```
392  ...       5         5       3      3      3       3         3  10   8
7
393  ...       4         4       1      3      4       5         0  11  12
10
394  ...       3         2       3      3      3       5         5   8   9
9

[10 rows x 33 columns]
```

Since there are a lot of columns, let's list them out in a way that is easy to read.

```
print('Done by Ryann Alvarez')

# Let's iterate over the columns for using a for loop
for col in student_df.columns:
    print(col)
```

```
Done by Ryann Alvarez
school
sex
age
address
famsize
Pstatus
Medu
Fedu
Mjob
Fjob
reason
guardian
traveltime
studytime
failures
schoolsup
famsup
paid
activities
nursery
higher
internet
romantic
famrel
freetime
goout
Dalc
Walc
health
absences
G1
```

```
G2
G3
```

```python
print('Done by Ryann Alvarez')

# Use .shape to look at the number of rows and columns
student_df.shape
```

```
Done by Ryann Alvarez
```

```
(395, 33)
```

```python
print('Done by Ryann Alvarez')

# Let's get overall info for the dataset
student_df.info()
```

```
Done by Ryann Alvarez
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 395 entries, 0 to 394
Data columns (total 33 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   school      395 non-null    object
 1   sex         395 non-null    object
 2   age         395 non-null    int64
 3   address     395 non-null    object
 4   famsize     395 non-null    object
 5   Pstatus     395 non-null    object
 6   Medu        395 non-null    int64
 7   Fedu        395 non-null    int64
 8   Mjob        395 non-null    object
 9   Fjob        395 non-null    object
 10  reason      395 non-null    object
 11  guardian    395 non-null    object
 12  traveltime  395 non-null    int64
 13  studytime   395 non-null    int64
 14  failures    395 non-null    int64
 15  schoolsup   395 non-null    object
 16  famsup      395 non-null    object
 17  paid        395 non-null    object
 18  activities  395 non-null    object
 19  nursery     395 non-null    object
 20  higher      395 non-null    object
 21  internet    395 non-null    object
 22  romantic    395 non-null    object
 23  famrel      395 non-null    int64
 24  freetime    395 non-null    int64
 25  goout       395 non-null    int64
 26  Dalc        395 non-null    int64
 27  Walc        395 non-null    int64
```

```
 28  health      395 non-null    int64
 29  absences    395 non-null    int64
 30  G1          395 non-null    int64
 31  G2          395 non-null    int64
 32  G3          395 non-null    int64
dtypes: int64(16), object(17)
memory usage: 102.0+ KB
```

Great! There's no missing data, so we do not need to worry about that when data cleaning.

Data Dictionary

Now that we've inspected our data a bit, let's create a data dictionary for our variables. This will explain what each variable means and the values they hold.

- **school** - student's school ('GP' - Gabriel Pereira, 'MS' - Mousinho da Silveira)
- **sex** - student's sex ('F' - female, 'M' - male)
- **age** - student's age
- **address** - student's home address type ('U' - urban, 'R' - rural)
- **famsize** - family size ('LE3' - less than or equal to 3, 'GT3' - greater than 3)
- **Pstatus** - parent's cohabitation status ('T' - living together, 'A' - apart)
- **Medu** - mother's education (1 - primary education (up to 4th grade), 2 - 5th to 9th grade, 3 - secondary education, 4 - higher education)
- **Fedu** - father's education (1 - primary education (up to 4th grade), 2 - 5th to 9th grade, 3 - secondary education, 4 - higher education)
- **Mjob** - mother's job
- **Fjob** - father's job
- **reason** - reason to choose this school
- **guardian** - student's guardian
- **traveltime** - home to school travel time
- **studytime** - weekly study time
- **failures** - number of past class failures
- **schoolsup** - extra educational support (yes or no)
- **famsup** - family educational support (yes or no)
- **paid** - extra paid classes within the course subject (i.e., math or portuguese) (yes or no)
- **activities** - extracurricular activities (yes or no)
- **nursery** - attended nursery school (yes or no)
- **higher** - wants to take higher education (yes or no)
- **internet** - internet access at home (yes or no)
- **romantic** - in a romantic relationship (yes or no)
- **famrel** - quality of family relationships (1 - very bad to 5 - excellent)
- **freetime** - free time after school (1 - very low to 5 - very high)
- **goout** - going out with friends (1 - very low to 5 - very high)
- **Dalc** - workday alcohol consumption (1 - very low to 5 - very high)

- **Walc** - weekend alcohol consumption (1 - very low to 5 - very high)
- **health** - current health status (1 - very bad to 5 - very good)
- **absences** - number of school absences
- **G1** - first period grade
- **G2** - second period grade
- **G3** - final period grade

I will analyze this data to answer some questions, but first let's clean the data.

Data Cleaning

Duplicates

Let's quickly check if there are any duplicates in the dataset.

```
print('Done by Ryann Alvarez')

# Use .duplicated() to check for duplicates
duplicates = student_df.duplicated()

# Print the value counts for duplicated items
duplicates.value_counts()
```

```
Done by Ryann Alvarez

False    395
Name: count, dtype: int64
```

There are only 'False' values, meaning there are no duplicates in this dataset. We can move forward with data cleaning.

Data Types

First, I'll focus on the data types for each variable, specifically any object types.

```
print('Done by Ryann Alvarez')

# Use .dtypes to display only the column and the data type
print(student_df.dtypes)
```

```
Done by Ryann Alvarez
school        object
sex           object
age            int64
address       object
famsize       object
Pstatus       object
```

```
Medu             int64
Fedu             int64
Mjob            object
Fjob            object
reason          object
guardian        object
traveltime       int64
studytime        int64
failures         int64
schoolsup       object
famsup          object
paid            object
activities      object
nursery         object
higher          object
internet        object
romantic        object
famrel           int64
freetime         int64
goout            int64
Dalc             int64
Walc             int64
health           int64
absences         int64
G1               int64
G2               int64
G3               int64
dtype: object
```

I would like to change any variables with an 'object' data type to a 'category' data type. Since there are a lot of variables with an 'object' data type, let's use a function for this conversion. Using a function will be more efficient than using a line of code for each and every variable.

```python
print('Done by Ryann Alvarez')

# Define a function that converts a variable to a 'category' data type
if their current data type is 'object'

# Function takes a DataFrame and a list of variables
def dtype_to_category(df, columns):
    """
    Converts specified columns in the DataFrame to 'category' dtype
from 'object' dtype
    """
    # For every variable in the list, change the dtype to 'category'
    for col in columns:
        df[col] = df[col].astype('category')
```

```
    # Return the DataFrame
    return df
```

```
Done by Ryann Alvarez
```

```
print('Done by Ryann Alvarez')

# Apply the function to convert any 'object' data types to 'category'.
# I will pass only the variables that are of 'object' data type!
student_df = dtype_to_category(student_df, ['school', 'sex',
'address', 'famsize', 'Pstatus',
                                             'Mjob', 'Fjob', 'reason',
'guardian', 'schoolsup',
                                             'famsup', 'paid',
'activities', 'nursery', 'higher',
                                             'internet', 'romantic'])

# Check to see if the data types have changed and are ready to use
print(student_df.dtypes)
```

```
Done by Ryann Alvarez
school        category
sex           category
age              int64
address       category
famsize       category
Pstatus       category
Medu             int64
Fedu             int64
Mjob          category
Fjob          category
reason        category
guardian      category
traveltime       int64
studytime        int64
failures         int64
schoolsup     category
famsup        category
paid          category
activities    category
nursery       category
higher        category
internet      category
romantic      category
famrel           int64
freetime         int64
goout            int64
Dalc             int64
Walc             int64
health           int64
```

```
absences          int64
G1                int64
G2                int64
G3                int64
dtype: object
```

Great! All of our variables are in their proper data types.

Renaming Variables

Some variable names do not make much sense. Let's change that by renaming the necessary variables.

```
print('Done by Ryann Alvarez')

# Rename the variables using the .rename() method
# Use a dictionary to rename multiple variables at once
student_df = student_df.rename(columns={'school': 'schoolName', 'sex':
'studentGender', 'age': 'studentAge',
                                        'address': 'homeLocation',
'famsize': 'familySize', 'Pstatus': 'parentStatus',
                                        'Medu': 'motherEdu', 'Fedu':
'fatherEdu', 'Mjob': 'motherJob',
                                        'Fjob': 'fatherJob', 'reason':
'schoolChoiceReason', 'guardian': 'studentGuardian',
                                        'traveltime': 'commuteTime',
'studytime': 'studyTime', 'failures': 'numFailures',
                                        'schoolsup': 'schoolSupport',
'famsup': 'familySupport', 'paid': 'paidClasses',
                                        'activities':
'inExtracurriculars', 'nursery': 'attendedNursery', 'higher':
'wantsHigherEdu',
                                        'internet': 'hasInternet',
'romantic': 'inRelationship', 'famrel': 'familyRel',
                                        'freetime': 'freeTime',
'goout': 'goOutFreq', 'Dalc': 'weekdayAlc',
                                        'Walc': 'weekendAlc',
'health': 'healthStatus', 'absences': 'numAbsences',
                                        'G1': 'firstGrade', 'G2':
'secondGrade', 'G3': 'finalGrade'})

# Iterate over the columns using a for loop
for col in student_df.columns:
    print(col)
```

```
Done by Ryann Alvarez
schoolName
studentGender
studentAge
```

```
homeLocation
familySize
parentStatus
motherEdu
fatherEdu
motherJob
fatherJob
schoolChoiceReason
studentGuardian
commuteTime
studyTime
numFailures
schoolSupport
familySupport
paidClasses
inExtracurriculars
attendedNursery
wantsHigherEdu
hasInternet
inRelationship
familyRel
freeTime
goOutFreq
weekdayAlc
weekendAlc
healthStatus
numAbsences
firstGrade
secondGrade
finalGrade
```

Now, we can see that our variables are renamed to something that is more appropriate/accurate.

Functions for Variable Recoding

Now, let's recode some of our variables so they will be easier to work with.

Begin by creating all the necessary functions...

```
print('Done by Ryann Alvarez')

# Define a function that recodes the values of the variable 'school'
def recode_school(school):
    """
    Converts values of the 'school' variable from 'GP' and 'MS' to
'Gabriel Pereira' and 'Mousinho da Silveira', respectively
    """
    # Return 'Gabriel Pereira' if school is 'GP'
```

```python
    if school == 'GP':
        return 'Gabriel Pereira'

    # Return 'Mousinho da Silveira' if school is 'MS'
    elif school == 'MS':
        return 'Mousinho da Silveira'
```

```
Done by Ryann Alvarez
```

```python
print('Done by Ryann Alvarez')

# Define a function that recodes the values of the variable 'gender'
def recode_gender(gender):
    """
    Converts values of the 'gender' variable from 'F' and 'M' to
'female' and 'male', respectively
    """
    # Return 'Female' if gender is 'F'
    if gender == 'F':
        return 'Female'

    # Return 'Male' if gender is 'M'
    elif gender == 'M':
        return 'Male'
```

```
Done by Ryann Alvarez
```

```python
print('Done by Ryann Alvarez')

# Define a function that recodes the values of the variable 'location'
def recode_location(location):
    """
    Converts values of the 'location' variable from 'U' and 'R' to
'Urban' and 'Rural', respectively
    """
    # Return 'Urban' if location is 'U'
    if location == 'U':
        return 'Urban'

    # Return 'Rural' if location is 'R'
    elif location == 'R':
        return 'Rural'
```

```
Done by Ryann Alvarez
```

```python
print('Done by Ryann Alvarez')

# Define a function that recodes the values of the variable
'parentStatus'
def recode_parentStatus(status):
    """
```

```
    Converts values of the 'parentStatus' variable from 'T' and 'A' to
'Together' and 'Apart', respectively
    """
    # Return 'together' if status is 'T'
    if status == 'T':
        return 'Together'

    # Return 'apart' if status is 'A'
    elif status == 'A':
        return 'Apart'

Done by Ryann Alvarez
```

Continue with applying all the necessary functions...

```
print('Done by Ryann Alvarez')

# Apply the recode_school function to the schoolName column
student_df['schoolName'] = student_df.schoolName.apply(recode_school)

# Apply the recode_gender function to the studentGender column
student_df['studentGender'] =
student_df.studentGender.apply(recode_gender)

# Apply the recode_location function to the homeLocation column
student_df['homeLocation'] =
student_df.homeLocation.apply(recode_location)

# Apply the recode_parentStatus function to the parentStatus column
student_df['parentStatus'] =
student_df.parentStatus.apply(recode_parentStatus)

# Preview of the data
student_df.head()

Done by Ryann Alvarez

        schoolName studentGender  studentAge homeLocation
familySize  \
0  Gabriel Pereira        Female          18        Urban        GT3

1  Gabriel Pereira        Female          17        Urban        GT3

2  Gabriel Pereira        Female          15        Urban        LE3

3  Gabriel Pereira        Female          15        Urban        GT3

4  Gabriel Pereira        Female          16        Urban        GT3


  parentStatus  motherEdu  fatherEdu motherJob fatherJob  ...
```

```
familyRel  \
0        Apart          4          4  at_home   teacher  ...
4
1     Together          1          1  at_home     other  ...
5
2     Together          1          1  at_home     other  ...
4
3     Together          4          2   health  services  ...
3
4     Together          3          3    other     other  ...
4

  freeTime  goOutFreq  weekdayAlc  weekendAlc healthStatus numAbsences
\
0        3          4           1           1            3           6

1        3          3           1           1            3           4

2        3          2           2           3            3          10

3        2          2           1           1            5           2

4        3          2           1           2            5           4


  firstGrade secondGrade finalGrade
0          5           6          6
1          5           5          6
2          7           8         10
3         15          14         15
4          6          10         10

[5 rows x 33 columns]
```

Awesome! We have recoded successfully!

Now, our data is clean and ready to work with!

Updated Data Dictionary

Let's update our data dictionary to reflect all the changes made during the cleaning process. We will use this one moving forward.

- **schoolName** - student's school ('Gabriel Pereira' or 'Mousinho da Silveira')
- **studentGender** - student's gender ('Female' or 'Male')
- **studentAage** - student's age
- **homeLocation** - student's home address type ('Urban' or 'Rural')
- **familySize** - family size ('LE3' - less than or equal to 3, 'GT3' - greater than 3)

- **parentStatus** - parent's cohabitation status ('Together' or 'Apart')
- **motherEdu** - mother's education (1 - primary education (up to 4th grade), 2 - 5th to 9th grade, 3 - secondary education, 4 - higher education)
- **fatherEdu** - father's education (1 - primary education (up to 4th grade), 2 - 5th to 9th grade, 3 - secondary education, 4 - higher education)
- **motherJob** - mother's job
- **fatherJob** - father's job
- **schoolChoiceReason** - reason to choose this school
- **studentGuardian** - student's guardian
- **commuteTime** - home to school travel time
- **studyTime** - weekly study time
- **numFailures** - number of past class failures
- **schoolSupport** - extra educational support (yes or no)
- **familySupport** - family educational support (yes or no)
- **paidClasses** - extra paid classes within the course subject (i.e., math or portuguese) (yes or no)
- **inExtracurriculars** - extracurricular activities (yes or no)
- **attendedNursery** - attended nursery school (yes or no)
- **wantsHigherEdu** - wants to take higher education (yes or no)
- **hasInternet** - internet access at home (yes or no)
- **inRelationship** - in a romantic relationship (yes or no)
- **familyRel** - quality of family relationships (1 - very bad to 5 - excellent)
- **freeTime** - free time after school (1 - very low to 5 - very high)
- **goOutFreq** - going out with friends (1 - very low to 5 - very high)
- **weekdayAlc** - workday alcohol consumption (1 - very low to 5 - very high)
- **weekendAlc** - weekend alcohol consumption (1 - very low to 5 - very high)
- **healthStatus** - current health status (1 - very bad to 5 - very good)
- **numAbsences** - number of school absences
- **firstGrade** - first period grade
- **secondGrade** - second period grade
- **finalGrade** - final period grade

Data Analysis

```python
print('Done by Ryann Alvarez')

# Import necessary libraries for analysis
import numpy as np
import pandas as pd
from scipy import stats

# Import necessary libraries for data visualization
import seaborn as sns
import matplotlib.pyplot as plt
```

```
# Import necessary libraries for statistics
import scipy.stats as stats

# Display visualizations in Jupyter notebook
%matplotlib inline

# Let's ignore any warnings as well
import warnings
warnings.filterwarnings('ignore')
```

```
Done by Ryann Alvarez
```

Question #1

Starting with the first question: What is the demographic breakdown of the students?

School

I'll start by checking the demographic of students that attend a particular school.

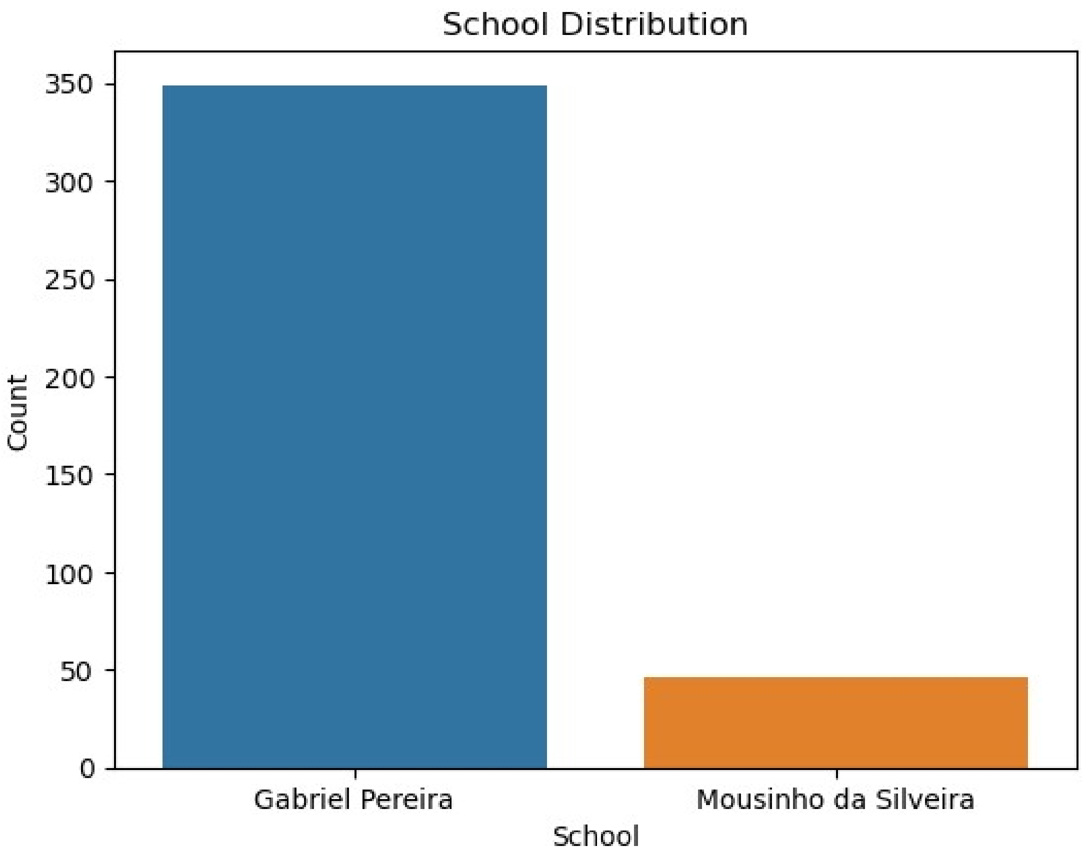
```
print('Done by Ryann Alvarez')

# Since schoolName is a categorical variable, use .value_counts() to
get a number
print(student_df['schoolName'].value_counts())
```

```
Done by Ryann Alvarez
schoolName
Gabriel Pereira         349
Mousinho da Silveira     46
Name: count, dtype: int64
```

We can see that way more students attend Gabriel Pereira (i.e., 349 students) compared to Mousinho da Silveira (i.e., 46 students). Let's visualize this...

```
print('Done by Ryann Alvarez')

# Use .countplot to create visualization for schoolName
sns.countplot(x='schoolName', data=student_df, palette='tab10')

# Plot title
plt.title("School Distribution")

# Plot axis labels
plt.xlabel("School")
plt.ylabel("Count")
```

```
# Show plot
plt.show()

Done by Ryann Alvarez
```



Wow, we can see a big difference! Way more students in this dataset attend Gabriel Pereira compared to Mousinho da Silveira.

Gender

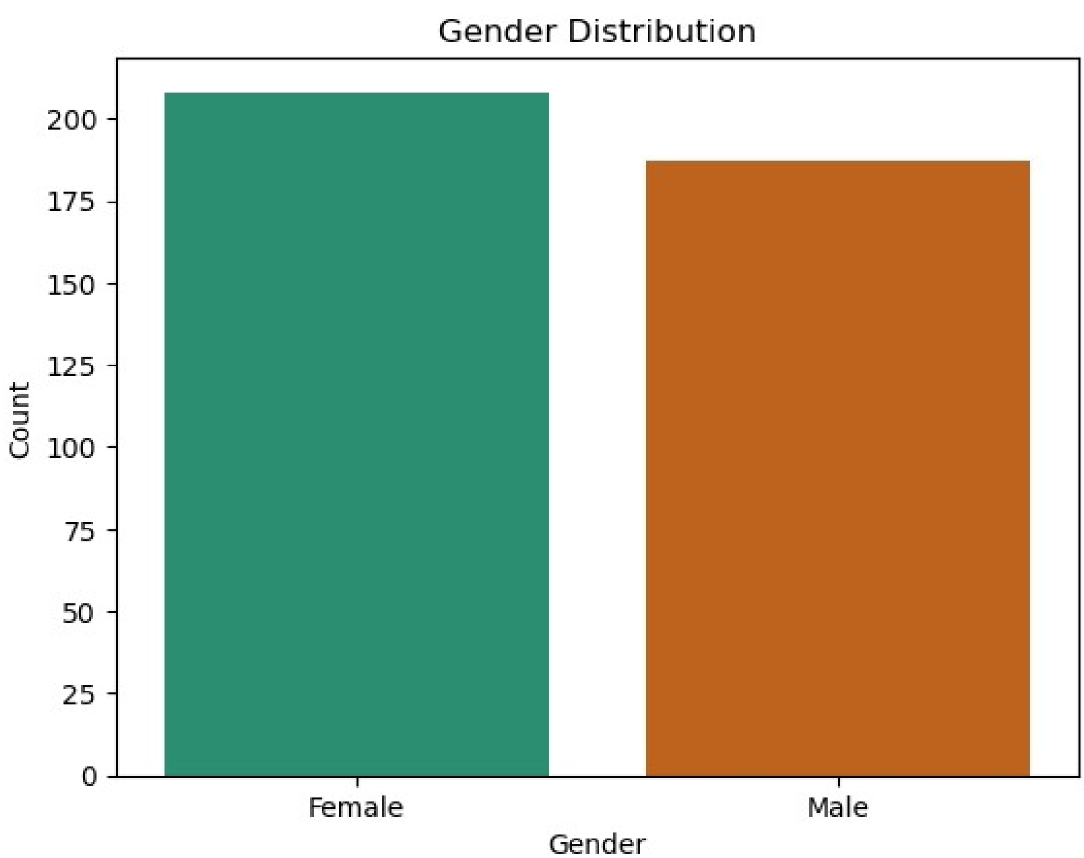
I'll continue by checking the gender demographic of students.

```
print('Done by Ryann Alvarez')

# Since studentGender is a categorical variable, use .value_counts()
to get a number
print(student_df['studentGender'].value_counts())

Done by Ryann Alvarez
studentGender
Female    208
Male      187
Name: count, dtype: int64
```

There are slightly more female students (i.e., 208 students) compared to male students (i.e., 187 students). Let's visualize this...

```
print('Done by Ryann Alvarez')

# Use .countplot() to create a visualization for gender
sns.countplot(x='studentGender', data=student_df, palette='Dark2')

# Plot title
plt.title("Gender Distribution")

# Plot axis labels
plt.xlabel("Gender")
plt.ylabel("Count")

# Show plot
plt.show()
```

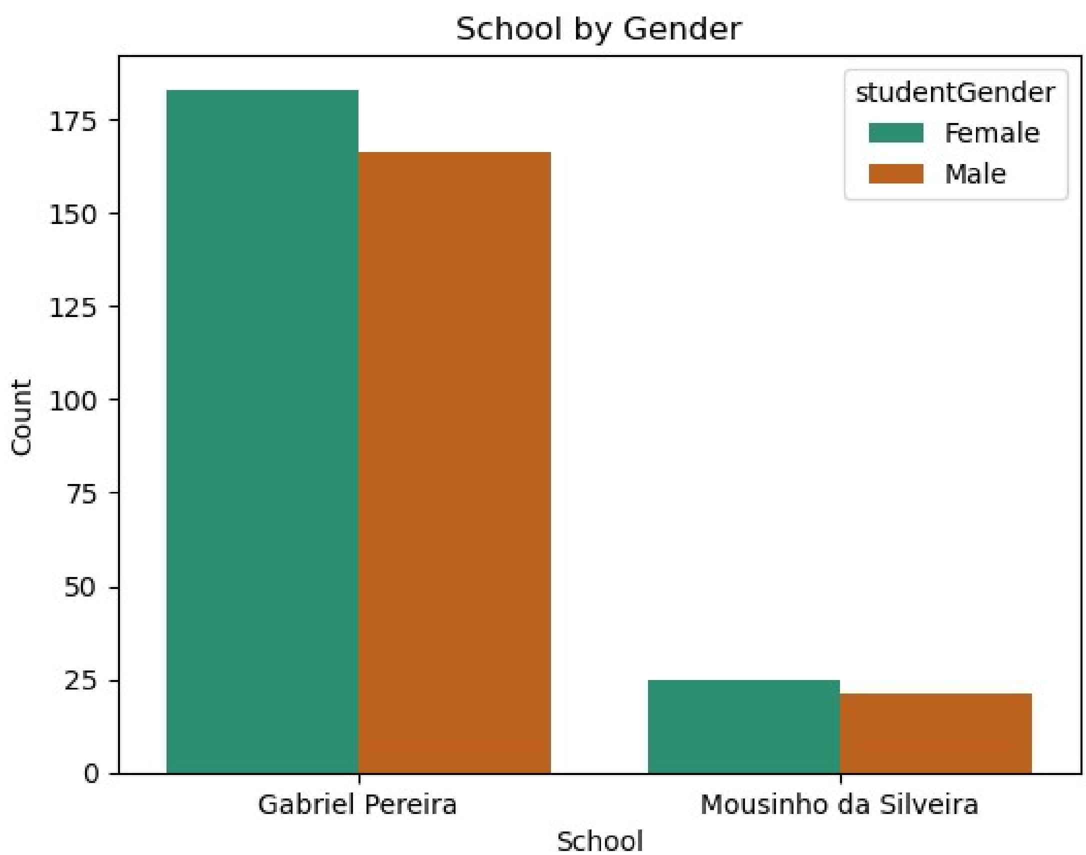
```
Done by Ryann Alvarez
```



We see that there are more females than males in this dataset, though the difference is not drastic.

What does the gender distribution look like at each school?

```
print('Done by Ryann Alvarez')

# Let's separate school by gender using the 'hue' argument
sns.countplot(x='schoolName', data=student_df, hue='studentGender',
palette='Dark2')

# Plot title
plt.title("School by Gender")

# Plot axis labels
plt.xlabel("School")
plt.ylabel("Count")

# Show plot
plt.show()

Done by Ryann Alvarez
```



We see that slighty more females attend both Gabriel Pereira and Mousinho da Silveira, which is expected considering there are more females in this dataset.

Age

What is the age demographic of the students?

```python
print('Done by Ryann Alvarez')

# Since studentAge is a numeric variable, .describe() can give us a quick summary for the age of our students
student_df['studentAge'].describe()
```

```
Done by Ryann Alvarez

count    395.000000
mean      16.696203
std        1.276043
min       15.000000
25%       16.000000
50%       17.000000
75%       18.000000
max       22.000000
Name: studentAge, dtype: float64
```

Using .describe() gives us key descriptives regarding the age of our students. Let's highlight some important findings.

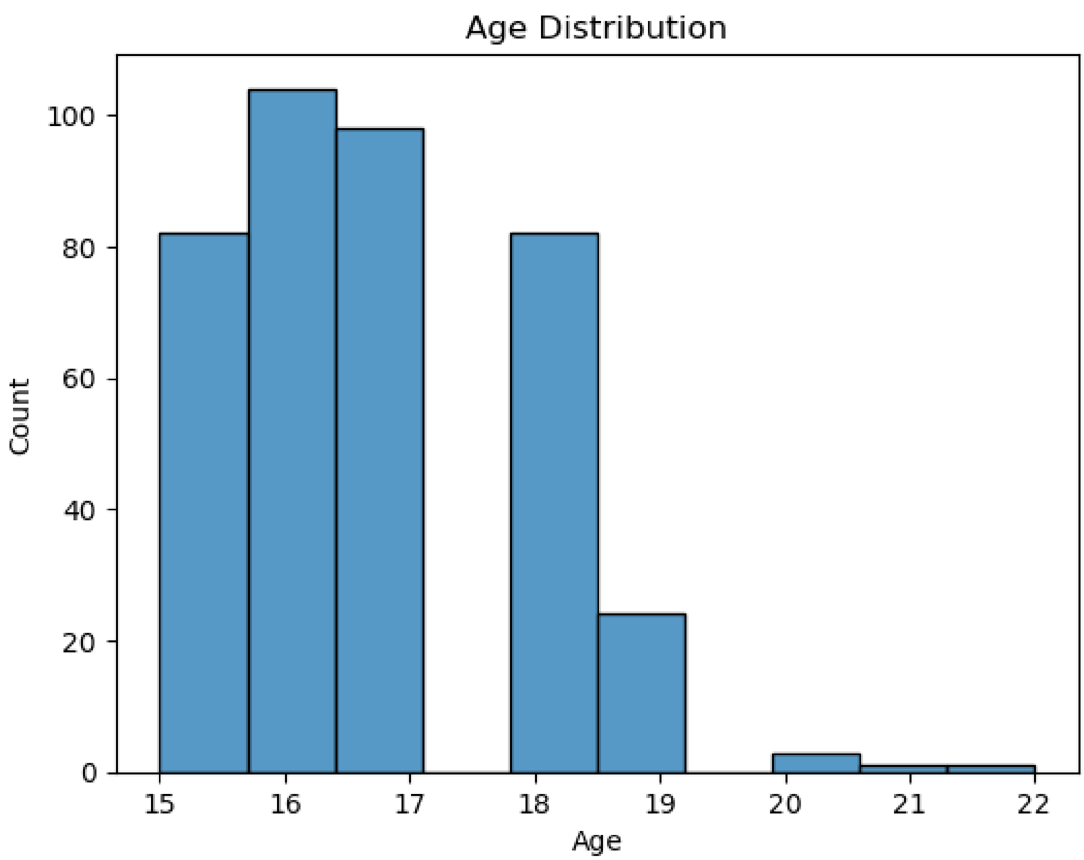
```python
print('Done by Ryann Alvarez')

# Average age of the students
avg_age = student_df['studentAge'].mean()

# Median age of the students
med_age = student_df['studentAge'].median()

# Minimum age of the students
min_age = student_df['studentAge'].min()

# Maximum age of the students
max_age = student_df['studentAge'].max()

# Display results for age
print(f'The average age is {avg_age:.4f}')
print(f'The median age is {med_age:.2f}')
print(f'The minimum age is {min_age:.2f}')
print(f'The maximum age is {max_age:.2f}')
```

```
Done by Ryann Alvarez
The average age is 16.6962
The median age is 17.00
The minimum age is 15.00
The maximum age is 22.00
```

Based off that quick line of code, the average age is about 17 years old, median age is 17, minimum age is 15, and maximum age is 22. Let's see a visual distribution of ages.

```
print('Done by Ryann Alvarez')

# Distribution for age
sns.histplot(x='studentAge', data=student_df, bins=10)

# Plot title
plt.title("Age Distribution")

# Plot axis labels
plt.xlabel("Age")
plt.ylabel("Count")

# Show plot
plt.show()
```

```
Done by Ryann Alvarez
```

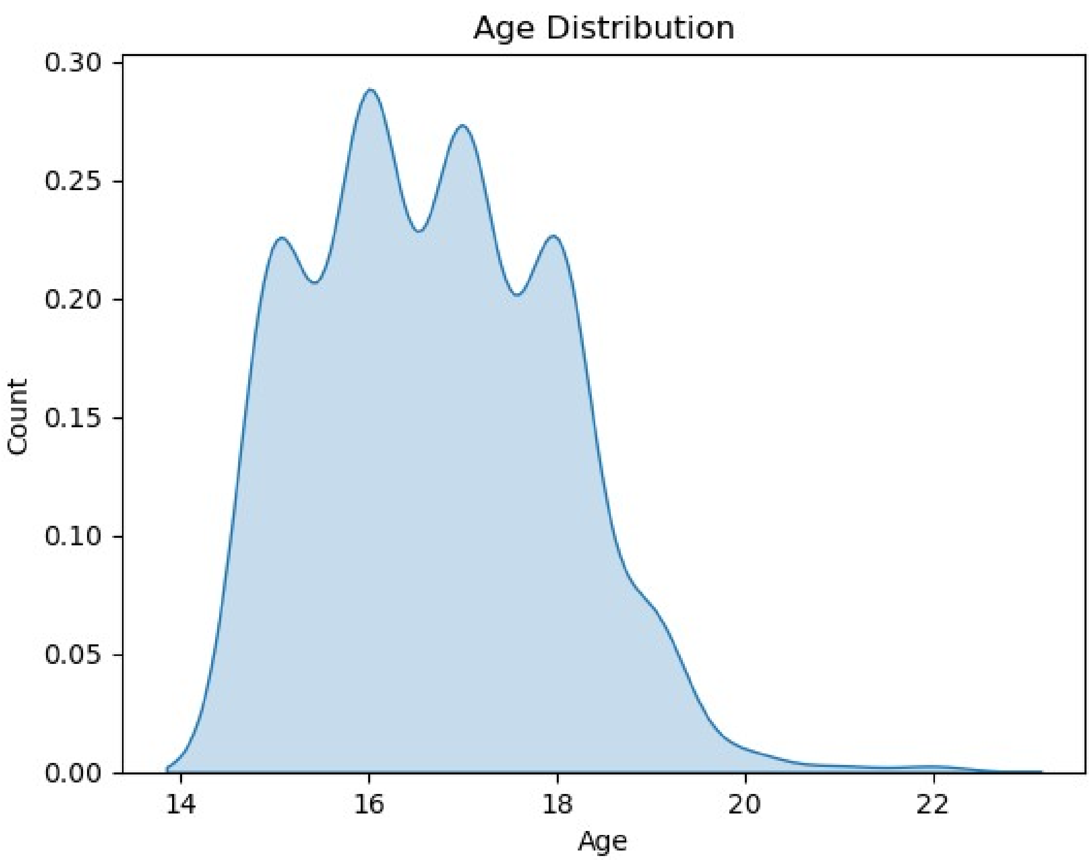


The distribution agrees with the average age in that most students are 16 to 17 years old. The distribution also shows some outliers that are 20 years old or older.

Let's see what the distribution looks like as a kde plot.

```python
print('Done by Ryann Alvarez')

# Distribution for age
sns.kdeplot(x='studentAge', data=student_df, fill=True)

# Plot title
plt.title("Age Distribution")

# Plot axis labels
plt.xlabel("Age")
plt.ylabel("Count")

# Show plot
plt.show()
```

```
Done by Ryann Alvarez
```



The kde plot does not really resemble a normal distribution, as it has several peaks. Let's test this to see if this sample differs from a normal distribution using a normal distribution test.

```python
print('Done by Ryann Alvarez')

# Import required libraries for a normal distribution test
```

```
from scipy.stats import normaltest
import scipy.stats as stats

# Test if student age is normally distributed

# H0: Age is normally distributed
# H1: Age is not normally distributed

# Use .normaltest which returns a tuple of the test statistic and p-value
stat, p = stats.normaltest(student_df['studentAge'])

# Print result
print(f"Normality test on age: statistic = {stat:.2f}, p = {p:.4f}")
```

```
Done by Ryann Alvarez
Normality test on age: statistic = 13.44, p = 0.0012
```

Since p = 0.0012 < 0.05, we can reject the null hypothesis. This suggests that there is strong evidence that the student's ages are not normally distributed. I will be mindful of this as we move forward in the project.

Let's use a z-test as well, which is appropriate given that our sample size is larger than 30.

```
print('Done by Ryann Alvarez')

# Import required libraries
from statsmodels.stats.weightstats import ztest
import scipy.stats as stats

# Use a two-tailed ztest to test that the mean age is 17
# H0: The population mean is equal to 17
# H1: The population mean is not equal to 17

# Specify the value being tested = 17
# Pass the alternative='two-sided' because this is a two-tailed test
# Specify degrees of freedom as 1, ddof = 1
# Returns the test statistic and p-value
stat, p = ztest(student_df['studentAge'], value=17, alternative='two-sided', ddof=1.0)

# Print result
print(f"Z-test for mean of age = 17: statistic = {stat:.2f}, p = {p:.4f}")
```

```
Done by Ryann Alvarez
Z-test for mean of age = 17: statistic = -4.73, p = 0.0000
```

Since p < 0.0001, we can reject the null hypothesis. This suggests there is strong evidence that the mean age of students is not equal to 17.

Location

Lastly, I'll check the demographic location of the students.

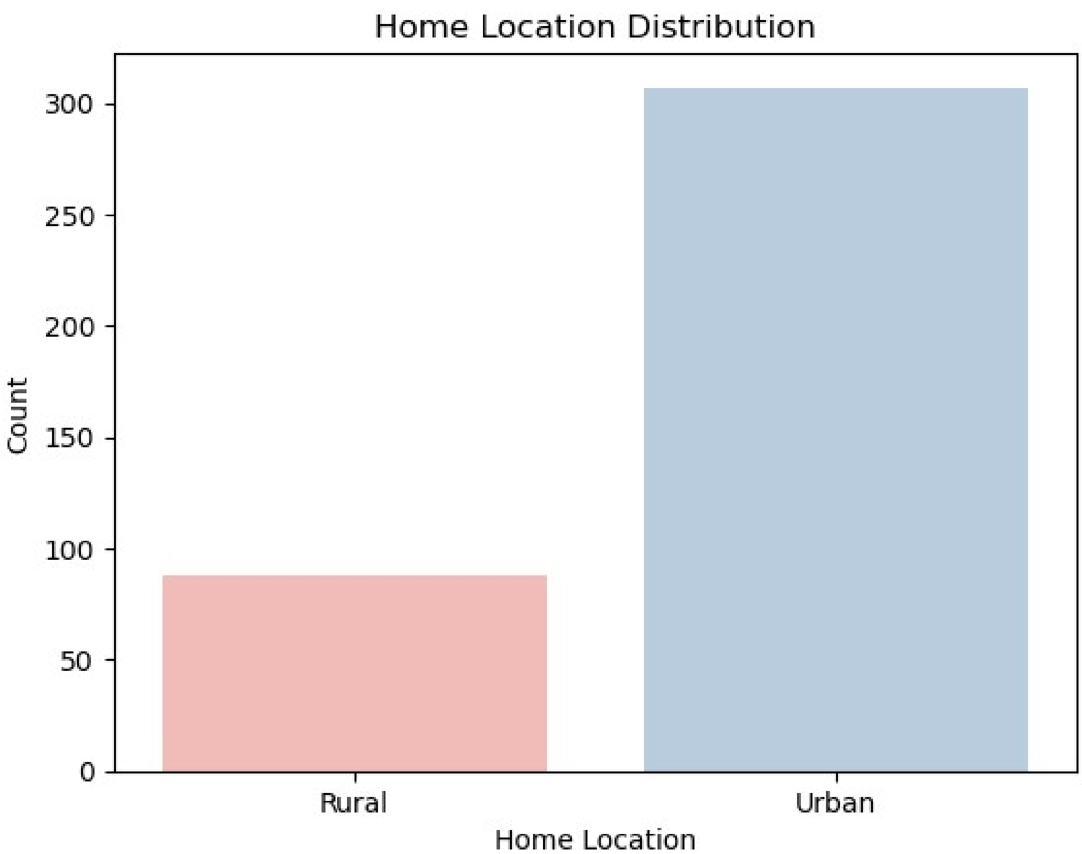
```
print('Done by Ryann Alvarez')

# Since homeLocation is a categorical variable, using .value_counts()
will give us a number
print(student_df['homeLocation'].value_counts())
```

```
Done by Ryann Alvarez
homeLocation
Urban    307
Rural     88
Name: count, dtype: int64
```

Based off the table, way more students live in an urban area (i.e., 307 students) compared to a rural area (i.e., 88 students).

```
print('Done by Ryann Alvarez')

# Use .countplot to create visualization for location
sns.countplot(x='homeLocation', data=student_df, palette='Pastel1')

# Plot title
plt.title("Home Location Distribution")

# Plot axis labels
plt.xlabel("Home Location")
plt.ylabel("Count")

# Show plot
plt.show()
```

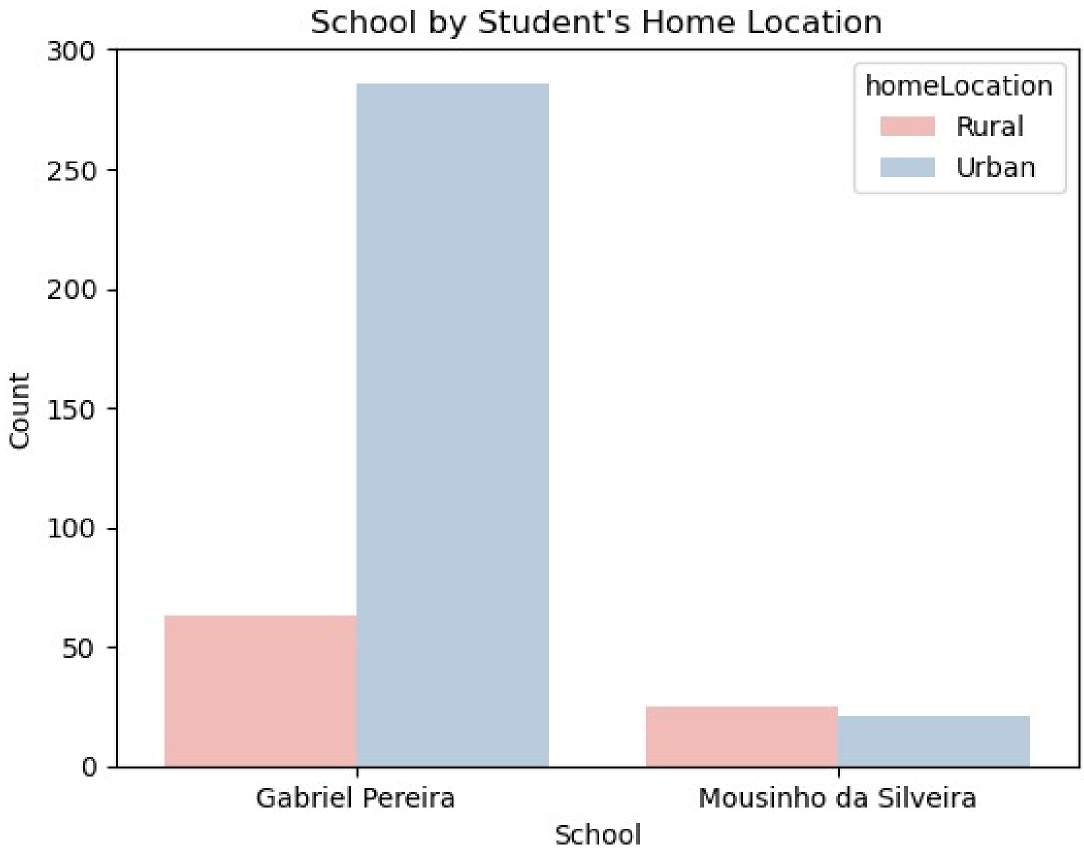
```
Done by Ryann Alvarez
```



We see that most students in this dataset live in an urban area compared to a rural area.

I'm curious to know if student's that attend a particular school live in more of a rural or urban area.

```
print('Done by Ryann Alvarez')

# Let's separate school by location using the 'hue' argument
sns.countplot(x='schoolName', data=student_df, hue='homeLocation',
palette='Pastel1')

# Plot title
plt.title("School by Student's Home Location")

# Plot axis labels
plt.xlabel("School")
plt.ylabel("Count")

# Show plot
plt.show()

Done by Ryann Alvarez
```



Here, we see that most students enrolled at Gabriel Pereira live in an urban area. There is a more even split between students living in a rural or urban area for those enrolled at Mousinho da Silveira.

With such a large difference in rural and urban living for students enrolled at Gabriel Pereira, I'm curious to know whether these categorical variables are related or independent. For this, I will use a chi-square test.

```
print('Done by Ryann Alvarez')

# Import required libraries for a chi-square test
from scipy.stats import chi2_contingency

# H0: There is no association between school and student's home
location (i.e., school and student's home location are independent)
# H1: There is an association between school and student's home
location

# Create a crosstab for observed frequencies
contingency_table = pd.crosstab(student_df['schoolName'],
student_df['homeLocation'])

# Returns the calculated statistic, p-value, degrees of freedom, and
table of expected frequncies
```

```
stat, p, dof, expected = chi2_contingency(contingency_table)

# Print results
print(f"Chi-Square Test: statistic = {stat:.2f}, p = {p:.4f}, dof =
{dof}")
```

```
Done by Ryann Alvarez
Chi-Square Test: statistic = 28.86, p = 0.0000, dof = 1
```

Since p < 0.0001, we can reject the null hypothesis. This suggests there is strong evidence that there is a statistically significant association between school and student's home location.

Question #2

Are there differences in academic performance based on demographic variables?

Let's begin by establishing and understanding the distribution of first, second and final grades.

```
print('Done by Ryann Alvarez')

# Use describe() to get summary of descriptive statistics for first,
second, and final grade
student_df[['firstGrade', 'secondGrade', 'finalGrade']].describe()
```

```
Done by Ryann Alvarez
```

	firstGrade	secondGrade	finalGrade
count	395.000000	395.000000	395.000000
mean	10.908861	10.713924	10.415190
std	3.319195	3.761505	4.581443
min	3.000000	0.000000	0.000000
25%	8.000000	9.000000	8.000000
50%	11.000000	11.000000	11.000000
75%	13.000000	13.000000	14.000000
max	19.000000	19.000000	20.000000

We get lot's of information from this (e.g., average, median, min, max, etc.)! Let's see a visualization for these descriptive statistics.

Let's reshape the data using **.melt()** into a DataFrame that will be easier to work with for creating visualizations.

```
print('Done by Ryann Alvarez')

# Need to use .melt to reshape the data
student_grades_melted = pd.melt(student_df, id_vars=['studentGender',
'studentAge', 'homeLocation'],
                          value_vars=['firstGrade', 'secondGrade',
'finalGrade'],
                          var_name='gradeType', value_name='grade')
```

```python
# Show result (easier to work with for visualizations)
student_grades_melted.head()
```

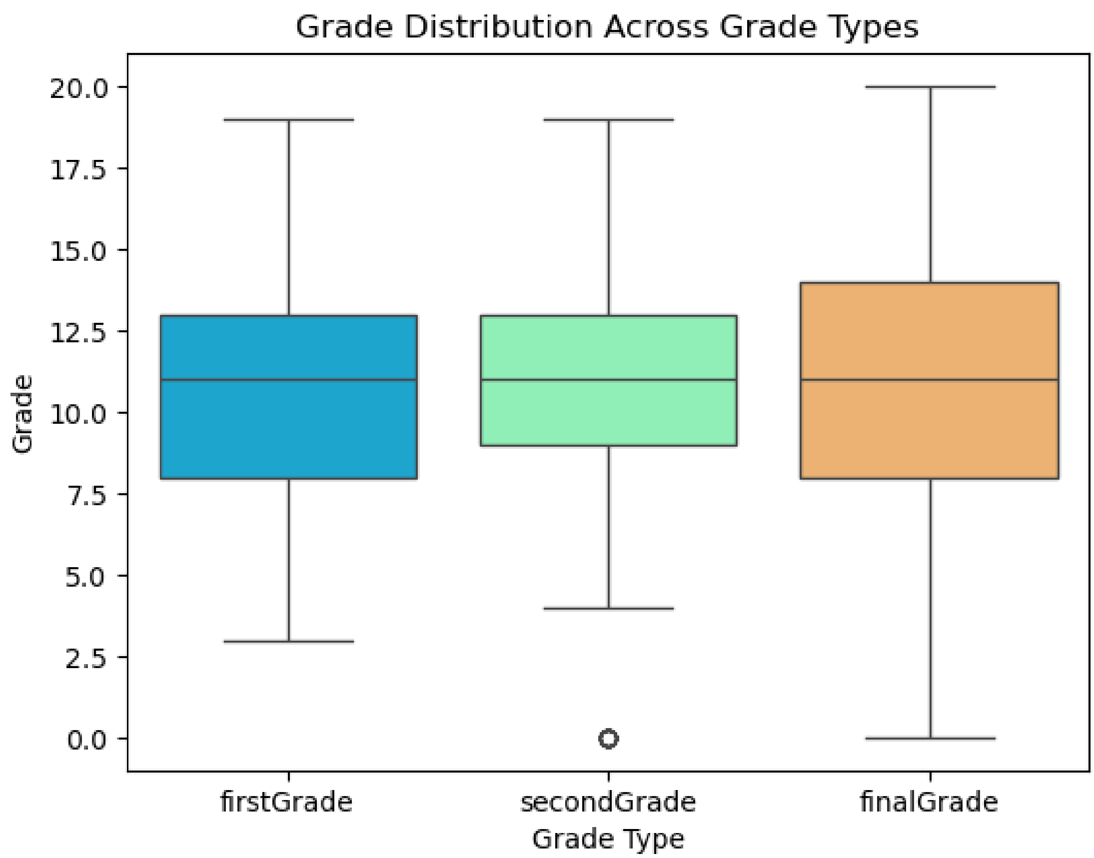
```
Done by Ryann Alvarez
```

```
  studentGender  studentAge homeLocation   gradeType  grade
0        Female          18        Urban  firstGrade      5
1        Female          17        Urban  firstGrade      5
2        Female          15        Urban  firstGrade      7
3        Female          15        Urban  firstGrade     15
4        Female          16        Urban  firstGrade      6
```

Now that the data is the proper format, let's create some visualization...

```python
print('Done by Ryann Alvarez')

# Use a boxplot to display the descriptive statistics for first,
second, and final grade
sns.boxplot(x='gradeType', y='grade', data=student_grades_melted,
palette='rainbow')

# Plot title
plt.title("Grade Distribution Across Grade Types")

# Plot axis labels
plt.xlabel("Grade Type")
plt.ylabel("Grade")

# Show plot
plt.show()
```

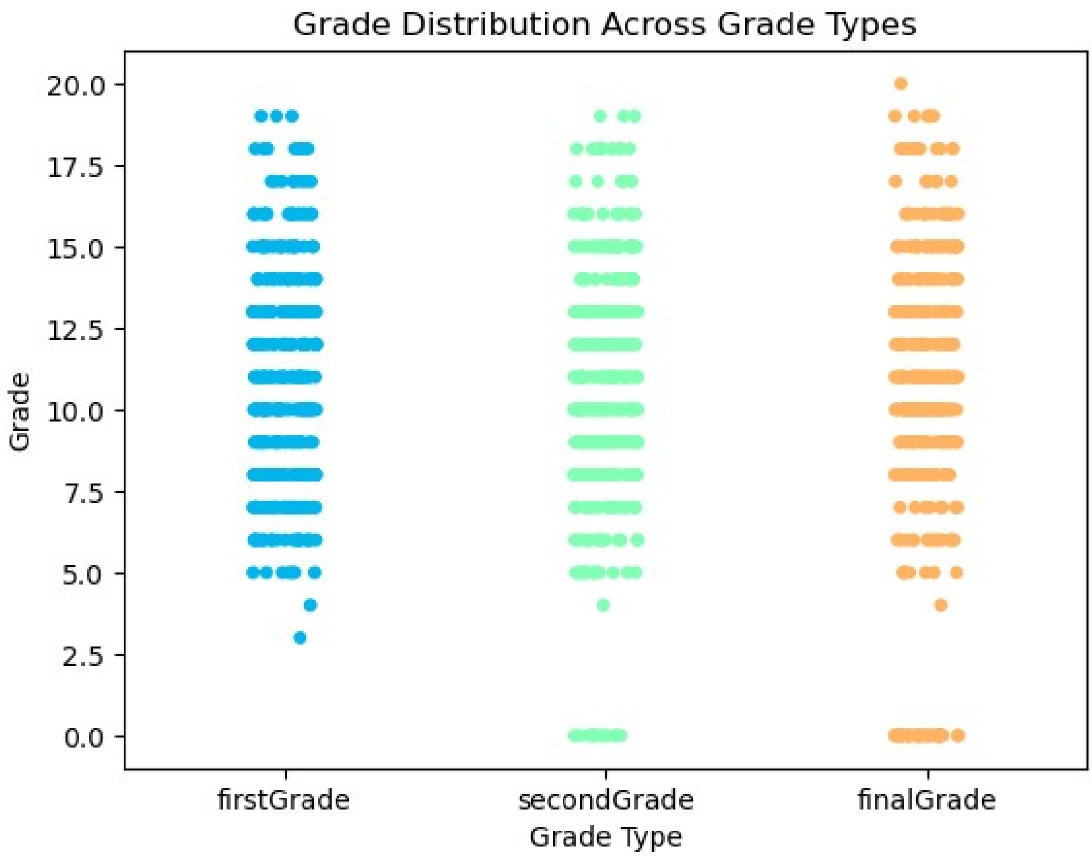
```
Done by Ryann Alvarez
```



Across each grade type (first, second, and final), the median is about the same - roughly a score of 11. Notably, the second grade has the least amount of dispersion (i.e., smallest interquartile range) while the final grade has the most amount of dispersion (i.e., largest interquartile range). Also note the second grade has outliers at 0, and the whiskers for final grade extend from the very bottom (i.e., 0) to the very top (i.e., 20). It seems that as the academic year progresses, student's scores start to vary or disperse more.

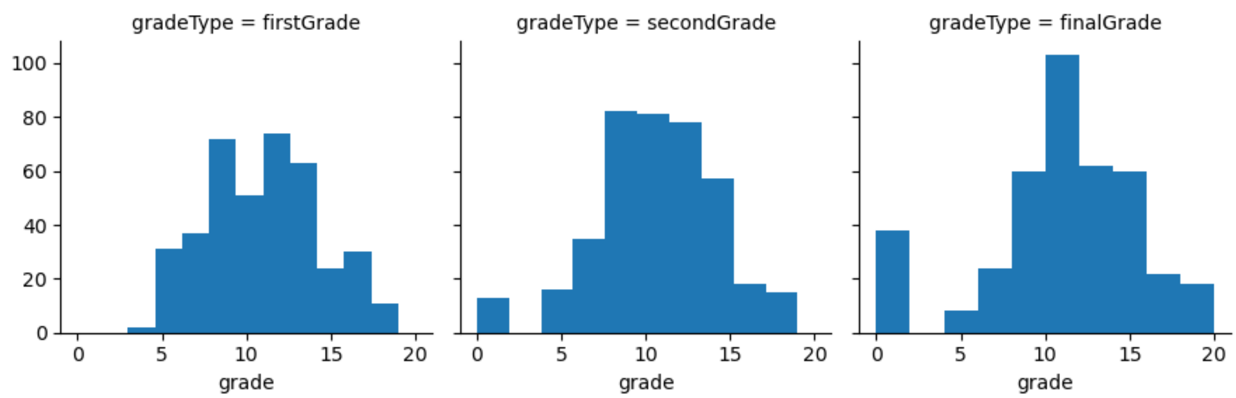
```
print('Done by Ryann Alvarez')

# Use a stripplot to display the spread of scores for first, second,
and final grade
# Not using a swarmplot because swarmplots are not recommended for
large datasets
sns.stripplot(x='gradeType', y='grade', data=student_grades_melted,
jitter=True, palette='rainbow')

# Plot title
plt.title('Grade Distribution Across Grade Types')

# Plot axis labels
plt.xlabel('Grade Type')
plt.ylabel('Grade')
```

```
# Show plot
plt.show()

Done by Ryann Alvarez
```



Similar to the boxplots, the stripplot emphasizes the scores at 0 seen in the second and final grade.

```
print('Done by Ryann Alvarez')

# Use FacetGrid to display the distribution of scores for first, second, and final grade
g = sns.FacetGrid(data=student_grades_melted, col='gradeType')
g.map(plt.hist, 'grade')

# Show plot
plt.show()

Done by Ryann Alvarez
```

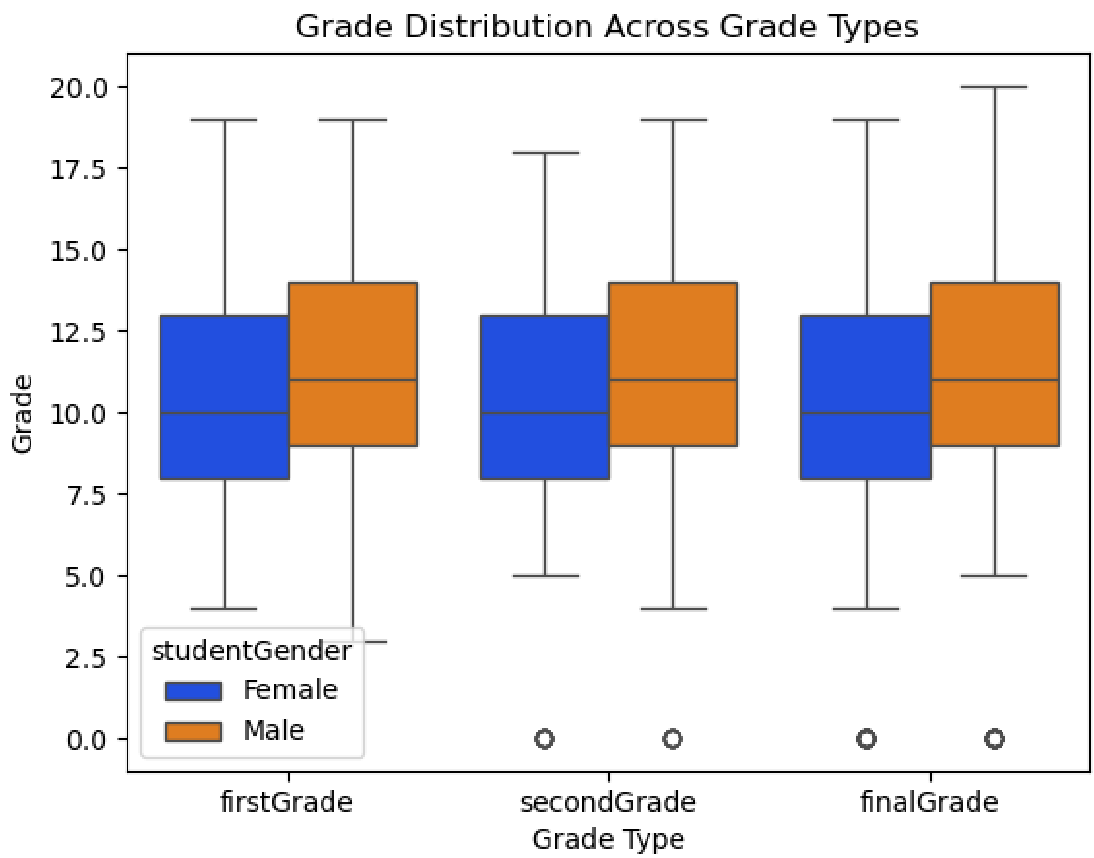


As seen earlier, most grades have a score around 11. Notably though, there are many grades that have 0. This would surely skew the distribution to be negatively skewed.

Now, let's see what the grades look like based on student's gender.

```
print('Done by Ryann Alvarez')

# Use a boxplot to display the descriptive statistics for first,
second, and final grade
sns.boxplot(x='gradeType', y='grade', data=student_grades_melted,
hue='studentGender', palette='bright')

# Plot title
plt.title("Grade Distribution Across Grade Types")

# Plot axis labels
plt.xlabel("Grade Type")
plt.ylabel("Grade")

# Show plot
plt.show()
```

```
Done by Ryann Alvarez
```



Based on the boxplot, we see that males performed slightly better than females for the first, second, and final grade.

Let's see if this difference is statistically significant using a t-test.

```
print('Done by Ryann Alvarez')

# Import required libraries for a t-test
from scipy import stats
from scipy.stats import ttest_ind

# Identify male and female grades
male_grades =
student_grades_melted[student_grades_melted['studentGender'] ==
'Male']['grade']
female_grades =
student_grades_melted[student_grades_melted['studentGender'] ==
'Female']['grade']

# H0: There is no significant difference between the means (i.e., mean
scores are equal)
# H1: There is a significant difference between the means (i.e., mean
scores are not equal)
```

```
# Perform independent samples t-test
# Returns t-test statistic and p-value
stat, p = stats.ttest_ind(male_grades, female_grades)

# Print result
print(f"Gender t-test: t = {stat:.3f}, p = {p:.4f}")
```

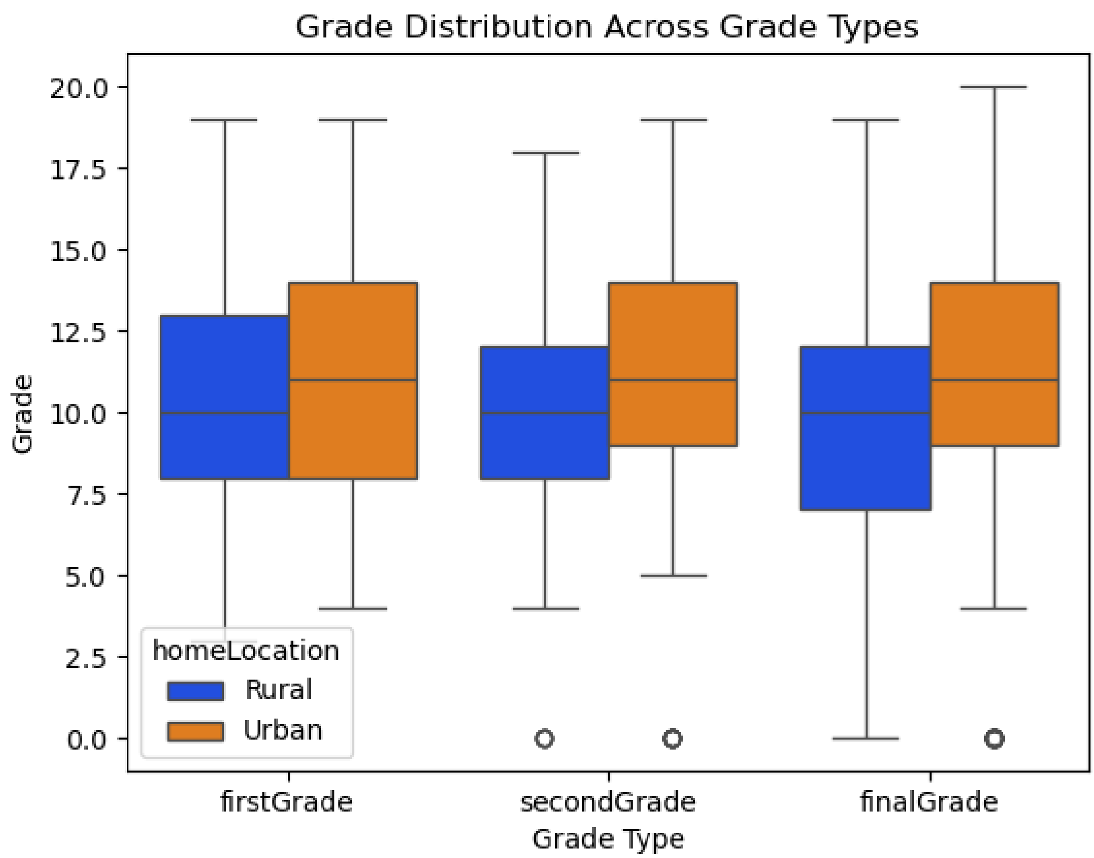
```
Done by Ryann Alvarez
Gender t-test: t = 3.289, p = 0.0010
```

Our p-value of = 0.001 is less than p = 0.05, so we can reject the null hypothesis and say there is a significant difference between the mean scores of males and females (i.e., mean scores are not equal). Since the t-test statistic is positive, that means the mean of the first group (i.e., males) is larger than the mean of the second group (i.e., females).

Let's see the scores based on student's home location.

```
print('Done by Ryann Alvarez')

# Use a boxplot to display the descriptive statistics for first,
second, and final grade
sns.boxplot(x='gradeType', y='grade', data=student_grades_melted,
hue='homeLocation', palette='bright')

# Plot title
plt.title("Grade Distribution Across Grade Types")

# Plot axis labels
plt.xlabel("Grade Type")
plt.ylabel("Grade")

# Show plot
plt.show()
```

```
Done by Ryann Alvarez
```



Students located in urban areas performed better than students located in rural areas for first, second, and final grades.

Let's see if this difference is statistically significant using a t-test.

```
print('Done by Ryann Alvarez')

# Import required libraries for a t-test
from scipy import stats
from scipy.stats import ttest_ind

# Identify urban and rural grades
urban_grades =
student_grades_melted[student_grades_melted['homeLocation'] ==
'Urban']['grade']
rural_grades =
student_grades_melted[student_grades_melted['homeLocation'] ==
'Rural']['grade']

# H0: There is no significant difference between the means (i.e., mean
scores are equal)
# H1: There is a significant difference between the means (i.e., mean
scores are not equal)
```

```
# Perform independent samples t-test
# Returns t-test statistic and p-value
stat, p = stats.ttest_ind(urban_grades, rural_grades)

# Print result
print(f"Home location t-test: t = {stat:.3f}, p = {p:.4f}")
```

```
Done by Ryann Alvarez
Home location t-test: t = 3.491, p = 0.0005
```

Our p-value of < 0.001 is less than p = 0.05, so we can reject the null hypothesis and say there is a significant difference between the mean scores of students who live in urban versus rural locations (i.e., mean scores are not equal). Since the t-test statistic is positive, that means the mean of the first group (i.e., urban) is larger than the mean of the second group (i.e., rural).

Question #3

How are students' first and second period grades related to their final grade?

Let's begin similar to the previous question by establishing and understanding the distribution of first, second and final grades.

```
print('Done by Ryann Alvarez')

# Use .describe() to get summary of descriptive statistics for first,
second, and final grade
student_df[['firstGrade', 'secondGrade', 'finalGrade']].describe()
```

```
Done by Ryann Alvarez
```

	firstGrade	secondGrade	finalGrade
count	395.000000	395.000000	395.000000
mean	10.908861	10.713924	10.415190
std	3.319195	3.761505	4.581443
min	3.000000	0.000000	0.000000
25%	8.000000	9.000000	8.000000
50%	11.000000	11.000000	11.000000
75%	13.000000	13.000000	14.000000
max	19.000000	19.000000	20.000000

The describe function gives us a table of useful descriptive statistics, like mean, min, max, and percentiles. One interesting trend I see is that the mean score is decreasing as the academic year progresses - not a trend we would expect or like to see from students.

Let's create some visualizations...

```
print('Done by Ryann Alvarez')

# Need to use .melt to reshape the data
student_grades_melted = pd.melt(student_df, value_vars=['firstGrade',
```

```
'secondGrade', 'finalGrade'],
                         var_name='gradeType', value_name='grade')

# Show result (easier to work with for visualizations)
student_grades_melted.head()
```

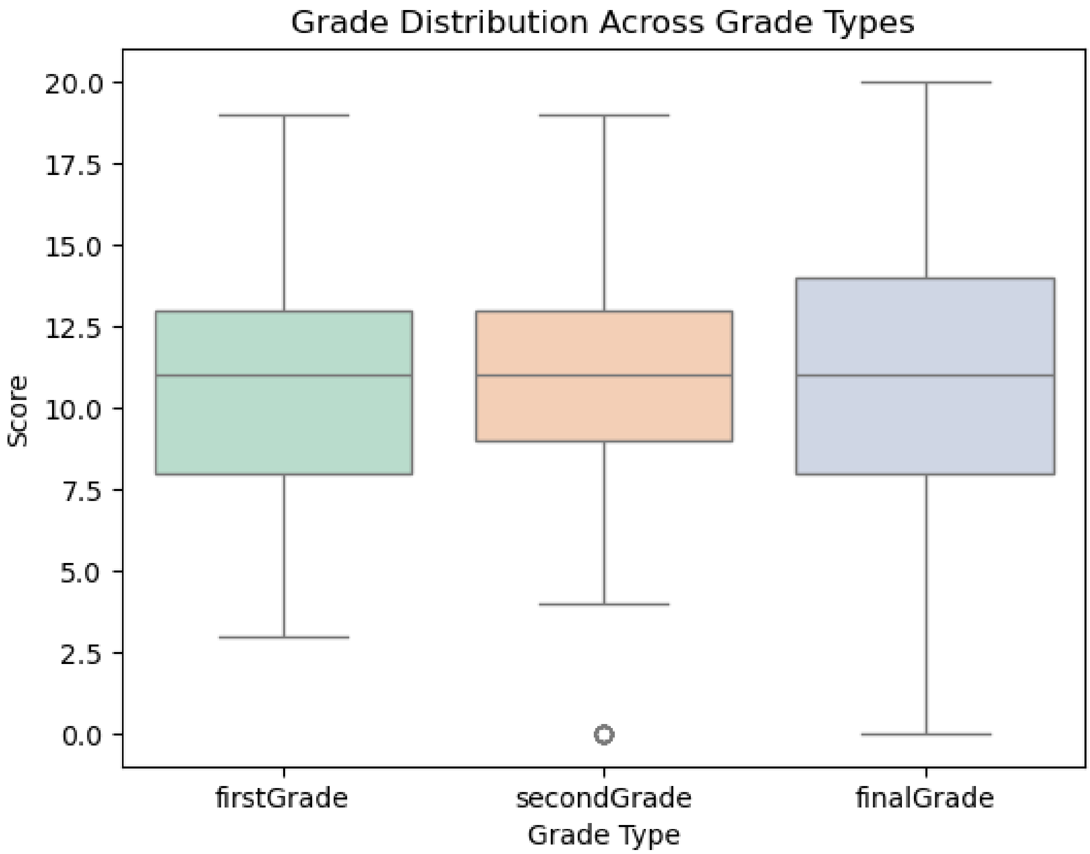
```
Done by Ryann Alvarez
```

```
    gradeType  grade
0  firstGrade      5
1  firstGrade      5
2  firstGrade      7
3  firstGrade     15
4  firstGrade      6
```

Now that the data is the proper format, let's create some visualization...

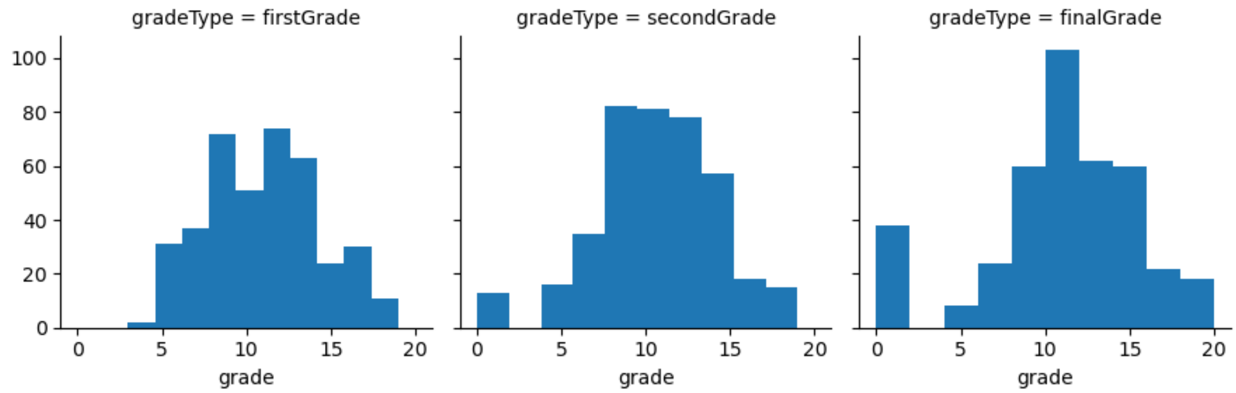
```
print('Done by Ryann Alvarez')

# Use a boxplot to display the descriptive statistics for first,
second, and final grade
sns.boxplot(x='gradeType', y='grade', data=student_grades_melted,
palette='Pastel2')

# Plot title
plt.title("Grade Distribution Across Grade Types")

# Plot axis labels
plt.xlabel("Grade Type")
plt.ylabel("Score")

# Show plot
plt.show()
```

```
Done by Ryann Alvarez
```



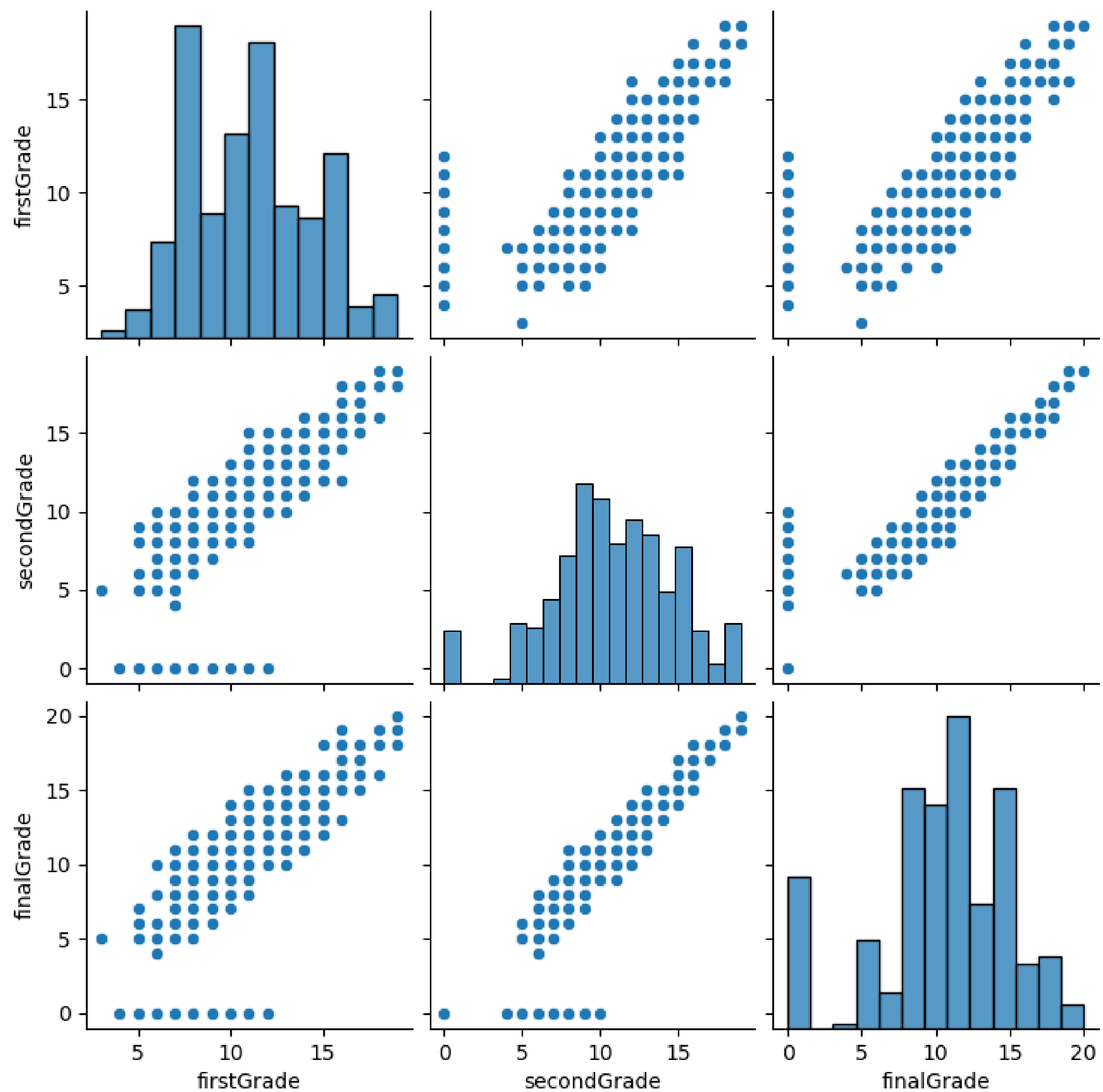
The boxplot shows that the second grade has outliers at 0, and the whiskers for final grade extend from the very bottom (i.e., 0). These low scores could explain why the mean score is decreasing as the academic year progresses.

```
print('Done by Ryann Alvarez')

# Use FacetGrid to display the distribution of scores for first,
second, and final grade
g=sns.FacetGrid(data=student_grades_melted, col='gradeType')
g.map(plt.hist, 'grade')

# Show plot
plt.show()

Done by Ryann Alvarez
```



Once again, these histograms emphasize the growing number of low scores (i.e., 0) for the second and final grade.

```
print('Done by Ryann Alvarez')

# Use a pairplot to see relationships across the first, second, and
final grade
sns.pairplot(student_df[['firstGrade', 'secondGrade', 'finalGrade']])

# Show plot
plt.show()

Done by Ryann Alvarez
```

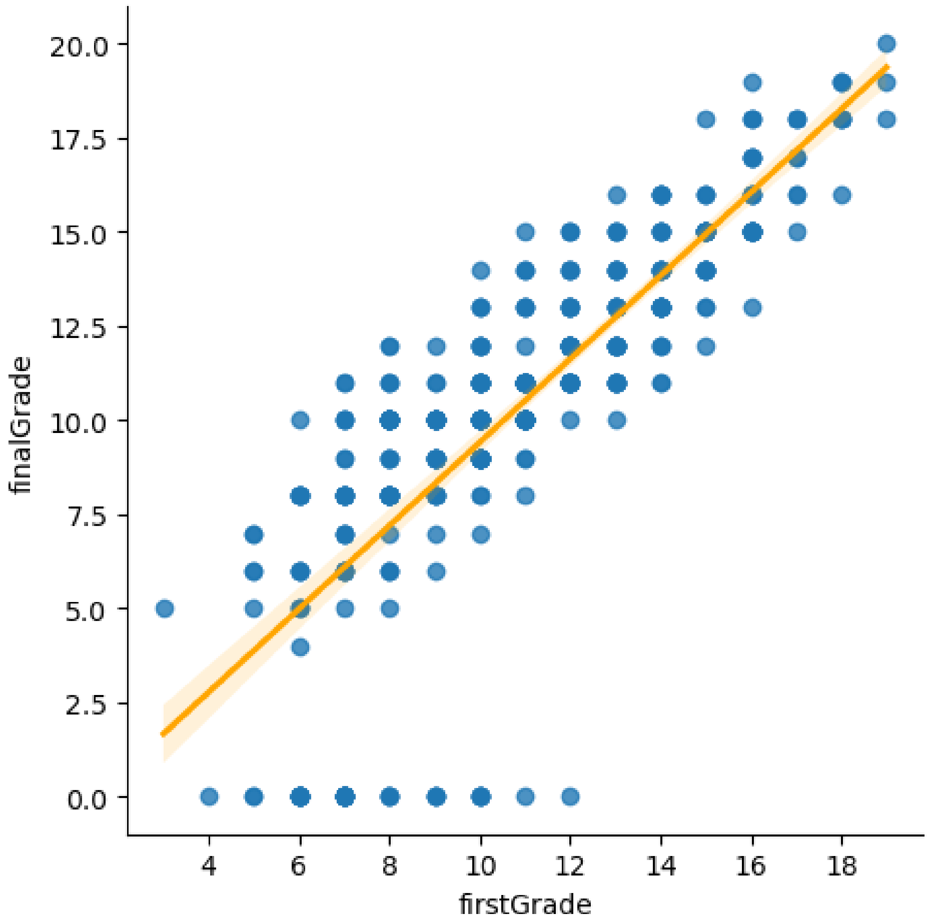


This pairplot gives us a rough idea of what to expect. We see that every scatterplot shows a positive correlation. Let's explore these relationships in more depth...

How are students' first grades related to their final grade?

```
print('Done by Ryann Alvarez')

# Use .lmplot()
sns.lmplot(x='firstGrade', y='finalGrade', data=student_df,
line_kws={'color': 'orange'})

# Show plot
plt.show()
```

```
Done by Ryann Alvarez
```

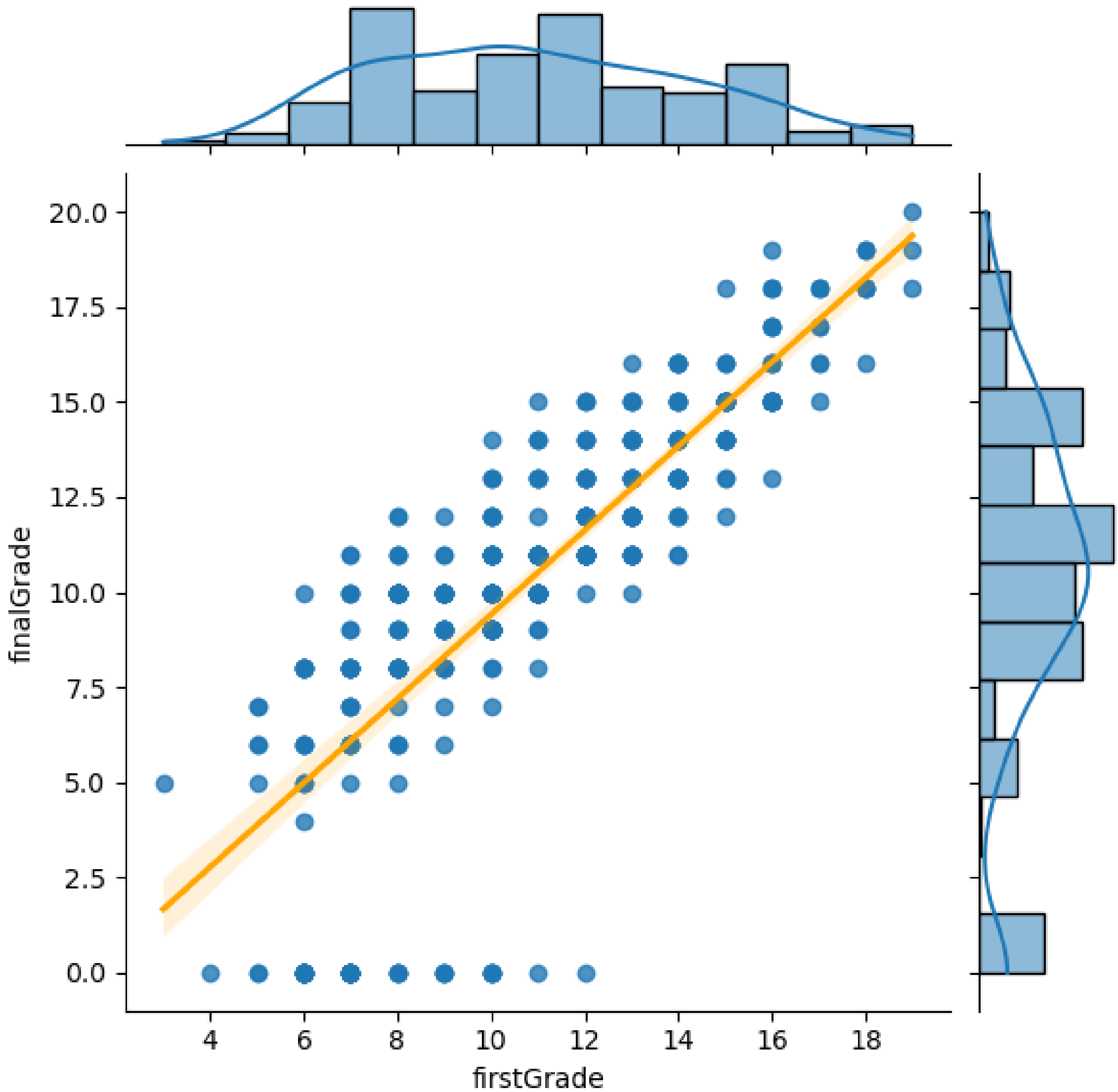


This lmplot shows us the linear relationship between student's first grade and final grade, which is a positive linear relationship. This means that as one variable increases (e.g., first grade), the other variable also increases (e.g., final grade).

Let's get a fuller picture using a jointplot.

```
print('Done by Ryann Alvarez')

# Use .jointplot()
sns.jointplot(x='firstGrade', y='finalGrade', data=student_df,
kind='reg', line_kws={'color': 'orange'})

# Show plot
plt.show()
```

```
Done by Ryann Alvarez
```



This jointplot shows a histogram of the distribution of scores as well as a scatterplot.

What is the correlation coefficient exactly?

```
print('Done by Ryann Alvarez')

# Import required libraries for calculating the Pearson correlation
coefficient
import numpy as np

# Define a function used to calculate the Pearson correlation
coefficient
def pearson_r(x, y):
    '''Compute Pearson correlation corricient between two
variables.'''
    # Compute correlation matrix: corr_mat
```

```
    corr_mat = np.corrcoef(x, y)

    # Return entry
    return corr_mat[0,1]

# Compute Pearson correlation coefficient for first and final grade
r = pearson_r(student_df['firstGrade'], student_df['finalGrade'])

# Print the result
print(f"The Pearson correlation coefficient is {r:.4f}")
```

```
Done by Ryann Alvarez
The Pearson correlation coefficient is 0.8015
```

Cool! We know that there is a strong positive linear relationship between student's first and final grades.

Now, we want to know if this is significant. If so, how significant?

```
print('Done by Ryann Alvarez')

# Import libraries for Pearson's Correlation test
from scipy.stats import pearsonr

# H0: There is no relationship between first and final grade
# H1: There is a relationship between first and final grade

# Tests the relationship between two variables (i.e., first and final
grade)
stat, p = pearsonr(student_df['firstGrade'], student_df['finalGrade'])

# Print stat (correlation coeffecient) and p-value for Pearson's
Correlation test
print('stat=%.3f, p=%.3f' % (stat, p))
```

```
Done by Ryann Alvarez
stat=0.801, p=0.000
```

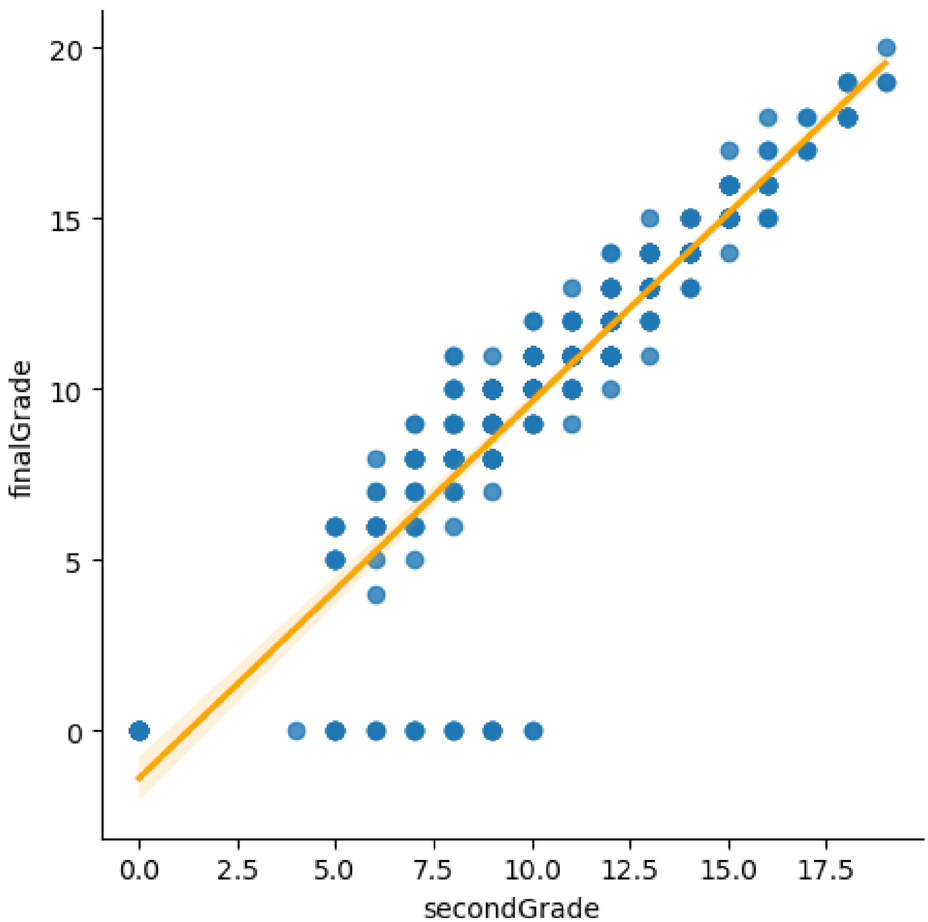
With a p-value that is less than 0.001, we can reject the null hypothesis. This suggests there is strong evidence that that there is a relationship between first and final grade.

What about student's second grades? How are students’ second grades related to their final grade?

```
print('Done by Ryann Alvarez')

# Use .lmplot()
sns.lmplot(x='secondGrade', y='finalGrade', data=student_df,
line_kws={'color': 'orange'})
```

```
# Show plot
plt.show()
```

```
Done by Ryann Alvarez
```

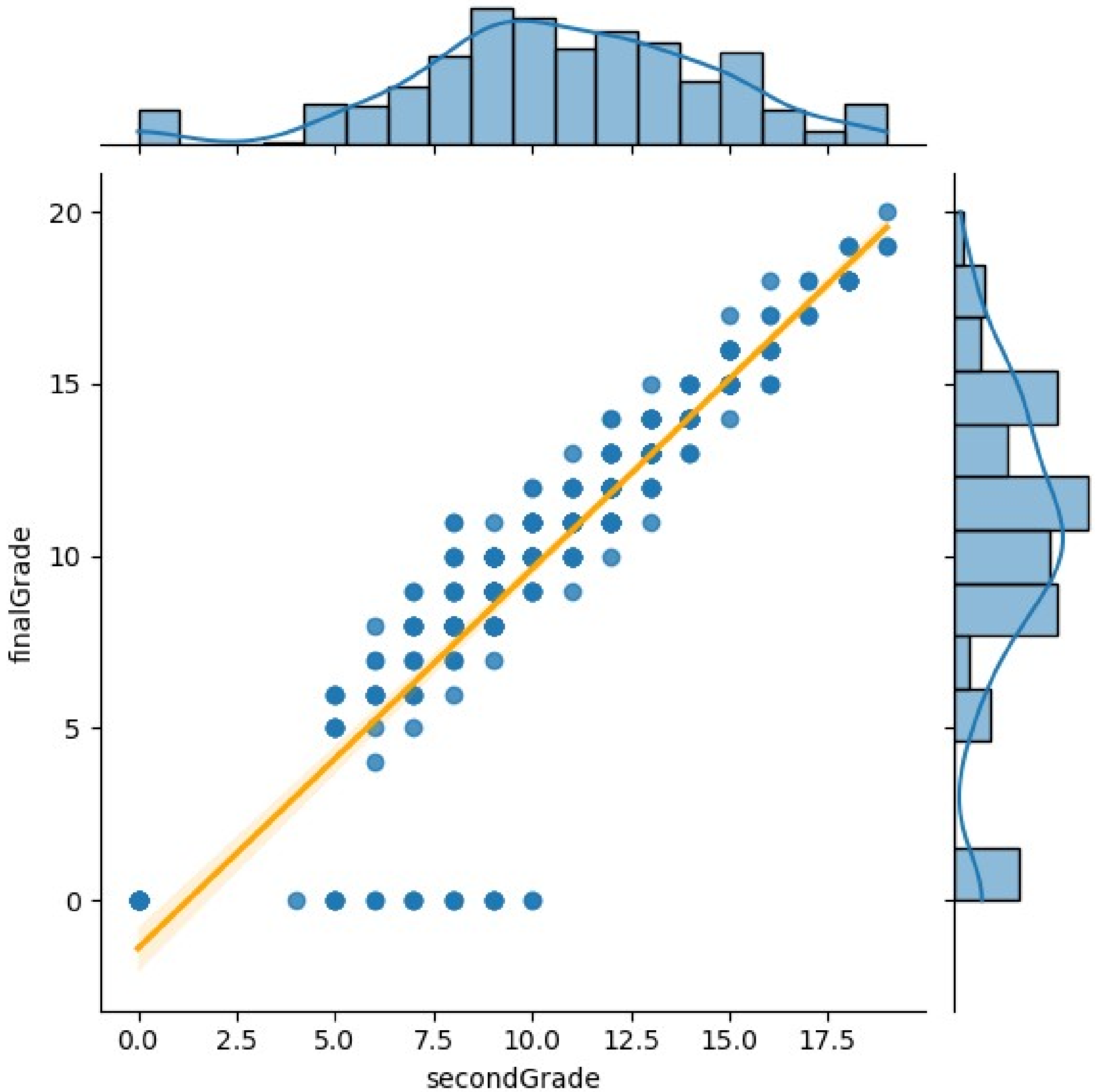


Similar to before, there is a positive linear relationship between student's second grade and final grade. So, as one variable increases (e.g., second grade), the other variable also increases (e.g., final grade).

Let's visualize it another way using jointplot.

```
print('Done by Ryann Alvarez')

# Use .jointplot()
sns.jointplot(x='secondGrade', y='finalGrade', data=student_df,
kind='reg', line_kws={'color': 'orange'})

# Show plot
plt.show()
```

```
Done by Ryann Alvarez
```



This jointplot shows a histogram of the distribution of scores as well as a scatterplot.

Let's identify the exact correlation coefficient.

```
print('Done by Ryann Alvarez')

# Use function defined previously to compute Pearson correlation
coefficient for second and final grade
r = pearson_r(student_df['secondGrade'], student_df['finalGrade'])

# Print the result
print(f"The Pearson correlation coefficient is {r:.4f}")
```

```
Done by Ryann Alvarez
The Pearson correlation coefficient is 0.9049
```

This shows a strong positive linear relationship between student's second and final grades. Notice that this correlation coefficient is larger than the correlation coefficient for first and final grades.

Let's find out if this is significant. If so, how significant?

```
print('Done by Ryann Alvarez')

# Import libraries for Pearson's Correlation test
from scipy.stats import pearsonr

# H0: There is no relationship between second and final grade
# H1: There is a relationship between second and final grade

# Tests the relationship between two variables (i.e., first and final
grade)
stat, p = pearsonr(student_df['secondGrade'],
student_df['finalGrade'])

# Print stat and p-value for Pearson's Correlation test
print('stat=%.3f, p=%.3f' % (stat, p))

Done by Ryann Alvarez
stat=0.905, p=0.000
```

With a p-value that is less than 0.001, we can reject the null hypothesis. This suggests there is strong evidence that that there is a relationship between second and final grade.

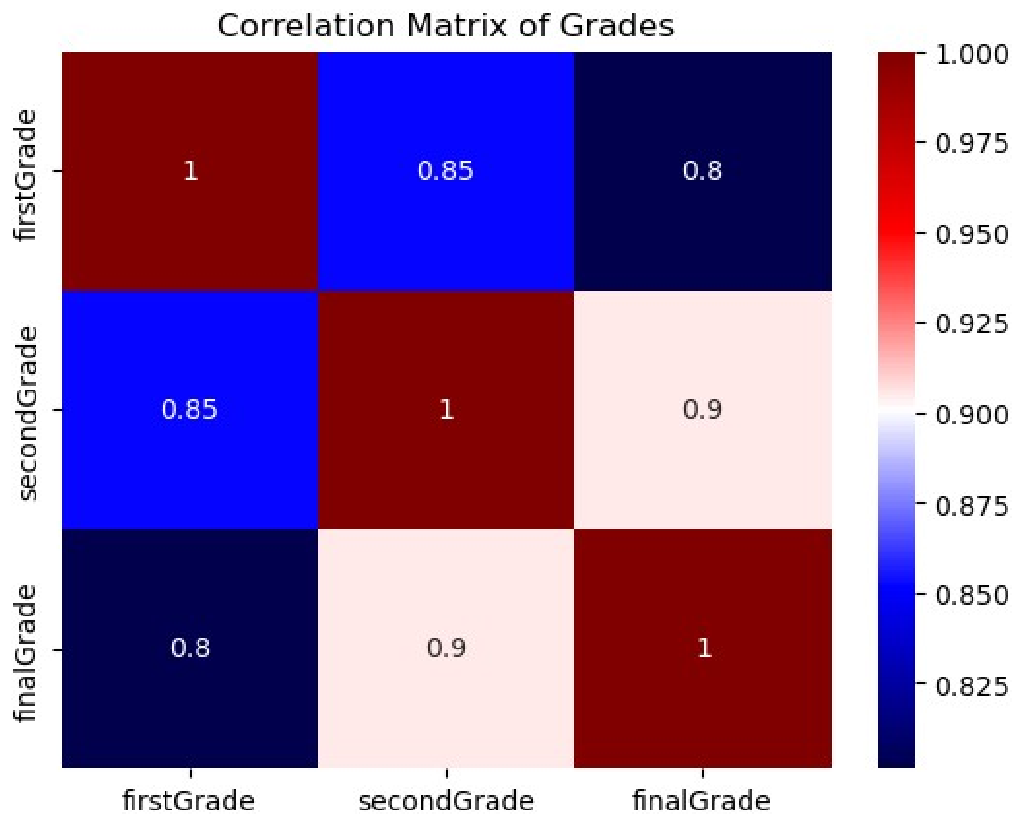
To sum it all up, let's put together a correlation matrix for all the grade correlations.

```
print('Done by Ryann Alvarez')

# Create correlation matrix use .corr
corr_matrix = student_df[['firstGrade', 'secondGrade',
'finalGrade']].corr(method='pearson')

# Show result
print(corr_matrix)

Done by Ryann Alvarez
             firstGrade  secondGrade  finalGrade
firstGrade     1.000000     0.852118    0.801468
secondGrade    0.852118     1.000000    0.904868
finalGrade     0.801468     0.904868    1.000000
```

...And let's visualize this correlation matrix using a heatmap.

```
print('Done by Ryann Alvarez')

# Create heatmap using .heatmap
```

```
sns.heatmap(corr_matrix, annot=True, cmap='seismic')

# Plot title
plt.title("Correlation Matrix of Grades")

# Show plot
plt.show()

Done by Ryann Alvarez
```



Question #4

How does parental education level relate to final grades?

For this, I'm going to start by pulling all the necessary variables into a separate DataFrame.

```
print('Done by Ryann Alvarez')

# Pull only necessary variables into new DataFrame
parent_education_df = pd.DataFrame(student_df[['motherEdu',
'fatherEdu', 'finalGrade']])

# Preview the new DataFrame
parent_education_df.head()
```

```
Done by Ryann Alvarez

   motherEdu  fatherEdu  finalGrade
0          4          4           6
1          1          1           6
2          1          1          10
3          4          2          15
4          3          3          10
```

Currently, motherEdu and fatherEdu variables have integer values that represent categories. Let's remind ourselves what each value represents and create a function that will recode the values:

- 1 - Primary education (up to 4th grade)
- 2 - 5th to 9th grade
- 3 - Secondary education
- 4 - Higher education

```
print('Done by Ryann Alvarez')

# Define a function that recodes the values of the variables
'motherEdu' and 'fatherEdu' into a new variable
def recode_parentEdu(edu):
    """
    Converts values of the 'motherEdu' or 'fatherEdu' variables from:
    1 to 'Primary education (up to 4th grade)'
    2 to '5th to 9th grade'
    3 to 'Secondary education'
    4 to 'Higher education'
    """
    # Return 'Primary education (up to 4th grade)' if edu is 1
    if edu == 1:
        return 'Primary'

    # Return '5th to 9th grade' if edu is 2
    elif edu == 2:
        return '5th-9th grade'

    # Return 'Secondary education' if edu is 3
    elif edu == 3:
        return 'Secondary'

    # Return 'Higher education' if edu is 4
    elif edu == 4:
        return 'Higher'
```

```
Done by Ryann Alvarez
```

```
print('Done by Ryann Alvarez')
```

```
# Apply the recode_parentEdu function to motherEdu
parent_education_df['motherEdu_recode'] =
parent_education_df.motherEdu.apply(recode_parentEdu)

# Apply the recode_parentEdu function to fatherEdu
parent_education_df['fatherEdu_recode'] =
parent_education_df.fatherEdu.apply(recode_parentEdu)

# Preview of the data
parent_education_df.head()
```

```
Done by Ryann Alvarez

   motherEdu  fatherEdu  finalGrade motherEdu_recode fatherEdu_recode
0          4          4           6           Higher           Higher
1          1          1           6          Primary          Primary
2          1          1          10          Primary          Primary
3          4          2          15           Higher    5th-9th grade
4          3          3          10        Secondary        Secondary
```

Perfect, now let's get to work!

Let's get some descriptive statistics going for mother and father education level...

```
print('Done by Ryann Alvarez')

# Use groupby to group by mother's education level
by_motherEdu = parent_education_df.groupby('motherEdu_recode')
['finalGrade']

# Use .describe() to get descriptive statistics
by_motherEdu.describe()
```

```
Done by Ryann Alvarez

                  count       mean       std  min  25%   50%   75%
max
motherEdu_recode

5th-9th grade     103.0   9.728155  4.636163  0.0  8.0  11.0  13.0
19.0
Higher            131.0  11.763359  4.267646  0.0  9.5  12.0  15.0
20.0
Primary            59.0   8.677966  4.364594  0.0  7.5  10.0  11.0
16.0
Secondary          99.0  10.303030  4.623486  0.0  8.0  10.0  13.0
19.0
```

We can also use a pivot table to look more closely at the mean for final grade by mother's education level.

```python
print('Done by Ryann Alvarez')

# Use a pivot table to summarize the results for mean
pivot_motherEdu = parent_education_df.pivot_table(values='finalGrade',
index='motherEdu_recode')

# Display the pivot table
print(pivot_motherEdu)
```

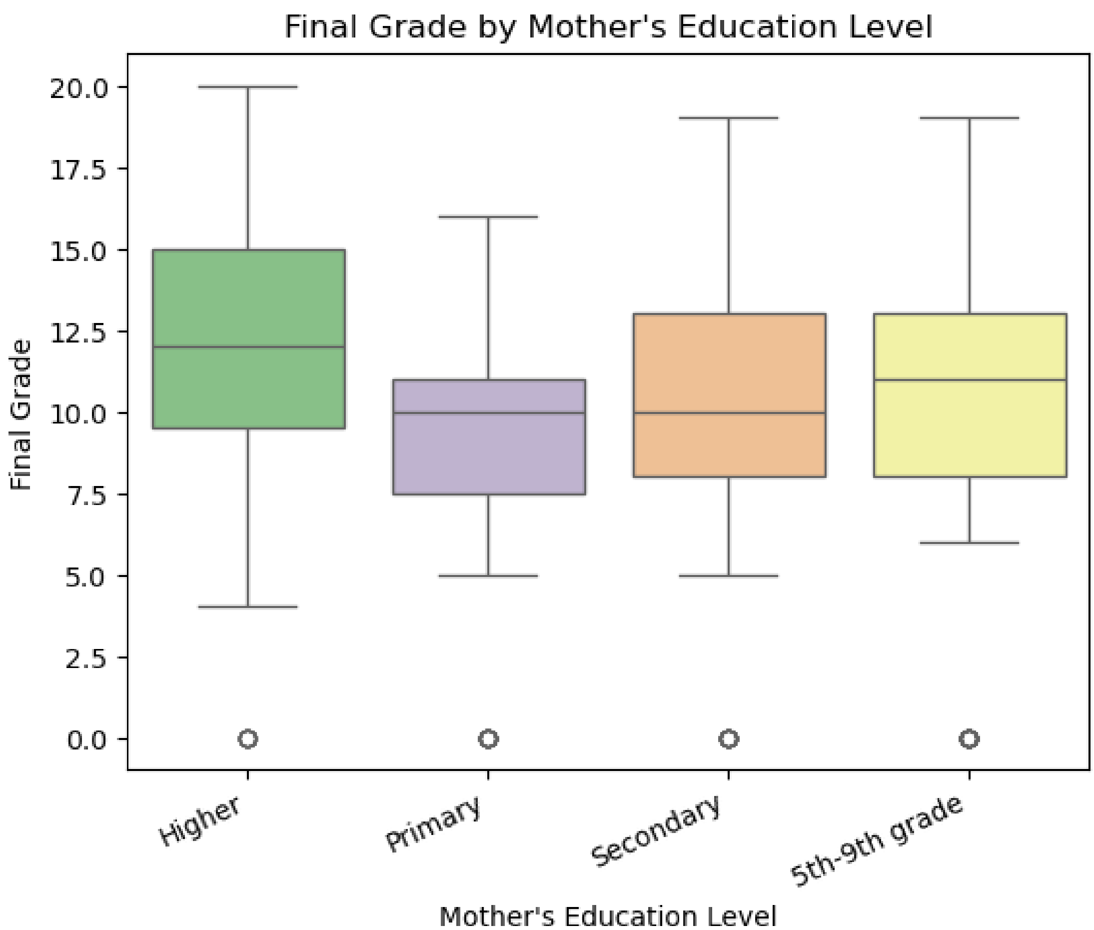
```
Done by Ryann Alvarez
                  finalGrade
motherEdu_recode
5th-9th grade       9.728155
Higher             11.763359
Primary             8.677966
Secondary          10.303030
```

Off this, we can see that the average final grade increases as mother's education level increases.

What do the descriptive statistics look like in a visualization? Let's use a boxplot for this.

```python
print('Done by Ryann Alvarez')

# Create a boxplot showing the distribution of final grades across
mother's educational levels
sns.boxplot(x='motherEdu_recode', y='finalGrade',
data=parent_education_df, palette='Accent')

# Adjust lavels
plt.xticks(rotation=25, ha='right')

# Plot title
plt.title("Final Grade by Mother's Education Level")

# Plot axes
plt.xlabel("Mother's Education Level")
plt.ylabel("Final Grade")

# Show plot
plt.show()
```

```
Done by Ryann Alvarez
```



Here, we can see very clearly that students whose mother's education level is primary score the lowest out of the other categories. We see that students whose mother's education level is higher score the highest out of the other categories - and have the widest dispersion of scores.

To get a better understanding of the distribution of scores by mother's education level, let's look at an ECDF function.

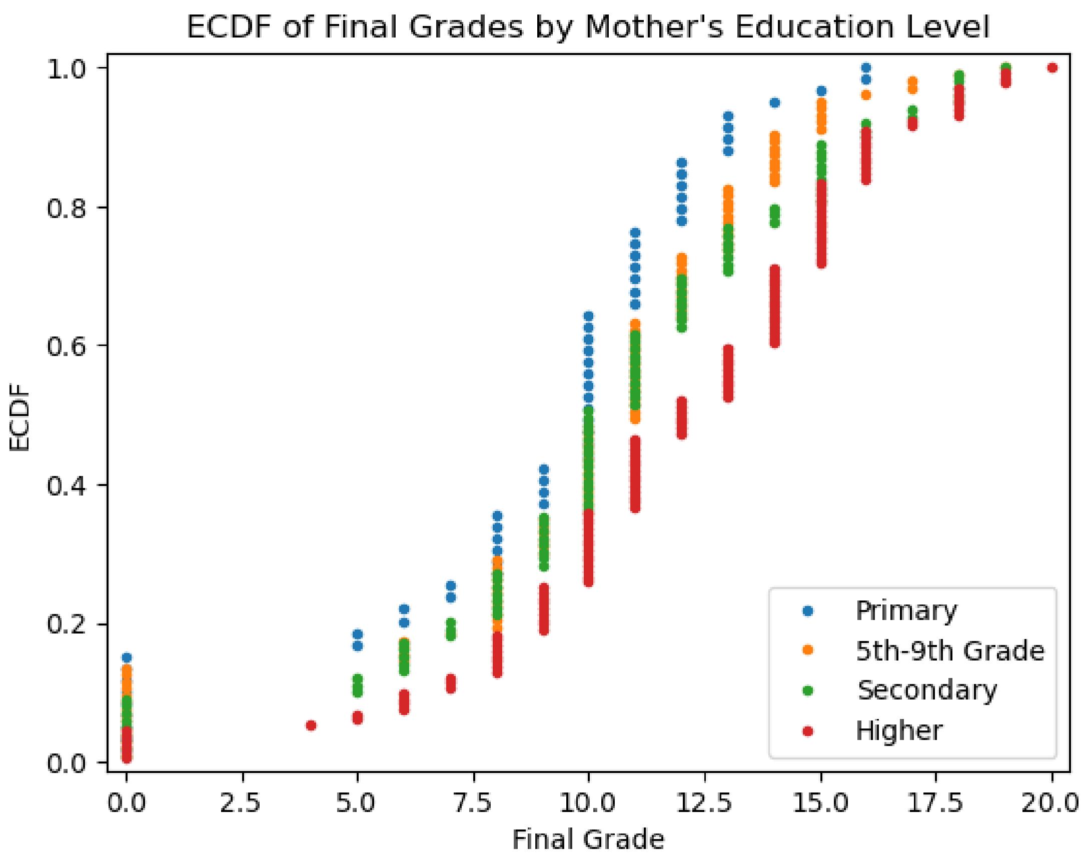
```
print('Done by Ryann Alvarez')

# Define ecdf function that takes one argument
def ecdf(data):
    """Compute ECDF for a one-dimensional array of measurements."""

    # Number of data points: n
    n = len(data)

    # x-data for the ECDF: x
    x = np.sort(data)

    # y-data for the ECDF: y
    y = np.arange(1, n+1) / n
```

```
    return x, y
```

Done by Ryann Alvarez

```
print('Done by Ryann Alvarez')

# Compute arrays for ECDF
motherEdu_1 = student_df[student_df['motherEdu'] == 1]['finalGrade']
motherEdu_2 = student_df[student_df['motherEdu'] == 2]['finalGrade']
motherEdu_3 = student_df[student_df['motherEdu'] == 3]['finalGrade']
motherEdu_4 = student_df[student_df['motherEdu'] == 4]['finalGrade']

# Compute ECDFs
x_motherEdu_1, y_motherEdu_1 = ecdf(motherEdu_1)
x_motherEdu_2, y_motherEdu_2 = ecdf(motherEdu_2)
x_motherEdu_3, y_motherEdu_3 = ecdf(motherEdu_3)
x_motherEdu_4, y_motherEdu_4 = ecdf(motherEdu_4)

# Plot all ECDFs on the same plot
_ = plt.plot(x_motherEdu_1, y_motherEdu_1, marker = '.', linestyle =
'none', label = "Primary")
_ = plt.plot(x_motherEdu_2, y_motherEdu_2, marker = '.', linestyle =
'none', label = "5th-9th Grade")
_ = plt.plot(x_motherEdu_3, y_motherEdu_3, marker = '.', linestyle =
'none', label = "Secondary")
_ = plt.plot(x_motherEdu_4, y_motherEdu_4, marker = '.', linestyle =
'none', label = "Higher")

# Make nice margins
plt.margins(0.02)

# Annotate the plot add a legend locate it at the lower right
_ = plt.title("ECDF of Final Grades by Mother's Education Level")
_ = plt.xlabel('Final Grade')
_ = plt.ylabel('ECDF')
plt.legend(loc='lower right')

# Display the plot
plt.show()
```

Done by Ryann Alvarez



The ECDF shows us that students whose mother's education level is higher education have a higher probability of scoring higher on the final compared to all other groups. Also, students whose mother's education level is primary education have a lower probability of scoring lower on the final compared to all other groups.

I'm curious to know if these categories are statistically significant. Let's conduct an ANOVA.

```
print('Done by Ryann Alvarez')

# Import libraries for one-way ANOVA
from scipy.stats import f_oneway

# Create a function that will conduct one way ANOVA test

# Function takes one argument, which is the name of a categorical
column
def anova_test(col):

    # parent_education_df.groupby(col) splits the data into groups
based on unique values
    # For each group, it pulls the finalGrade values using
group['finalGrade'].values
    groups = [group['finalGrade'].values for name, group in
parent_education_df.groupby(col)]
```

```
    # *groups unpacks the list so each array becomes a separate
argument
    # Returns the F-statistic and p-value
    stat, p = f_oneway(*groups)

    # Print result
    print(f"{col}: F = {stat:.2f}, p = {p:.4f}")

# H0: Mean final grade is the same across all four mother education
groups (i.e., Primary, 5th-9th Grade, Secondary, Higher)
# H1: At least one group has a different mean final grade

# Conduct ANOVA
anova_test('motherEdu_recode')
```

```
Done by Ryann Alvarez
motherEdu_recode: F = 7.76, p = 0.0000
```

Since our p < 0.001, we can reject the null hypothesis. This suggests strong evidence that final grades differ depending on their mother's education level.

Let's repeat this with father's education level.

```
print('Done by Ryann Alvarez')

# Use groupby to group by father's education level
by_fatherEdu = parent_education_df.groupby('fatherEdu_recode')
['finalGrade']

# Use .describe() to get descriptive statistics
by_fatherEdu.describe()
```

```
Done by Ryann Alvarez
```

	count	mean	std	min	25%	50%	75%	max
fatherEdu_recode								
5th-9th grade	115.0	10.260870	4.733396	0.0	8.50	11.0	13.50	19.0
Higher	96.0	11.364583	4.665934	0.0	9.75	12.0	14.25	19.0
Primary	82.0	9.158537	4.563596	0.0	7.00	10.0	12.00	18.0
Secondary	100.0	10.660000	4.149285	0.0	9.00	10.0	13.00	20.0

Use a pivot table to look more closely at the mean for final grade by father's education level.

```python
print('Done by Ryann Alvarez')

# Use a pivot table to summarize the results for mean
pivot_fatherEdu = parent_education_df.pivot_table(values='finalGrade',
index='fatherEdu_recode')

# Display the pivot table
print(pivot_fatherEdu)
```

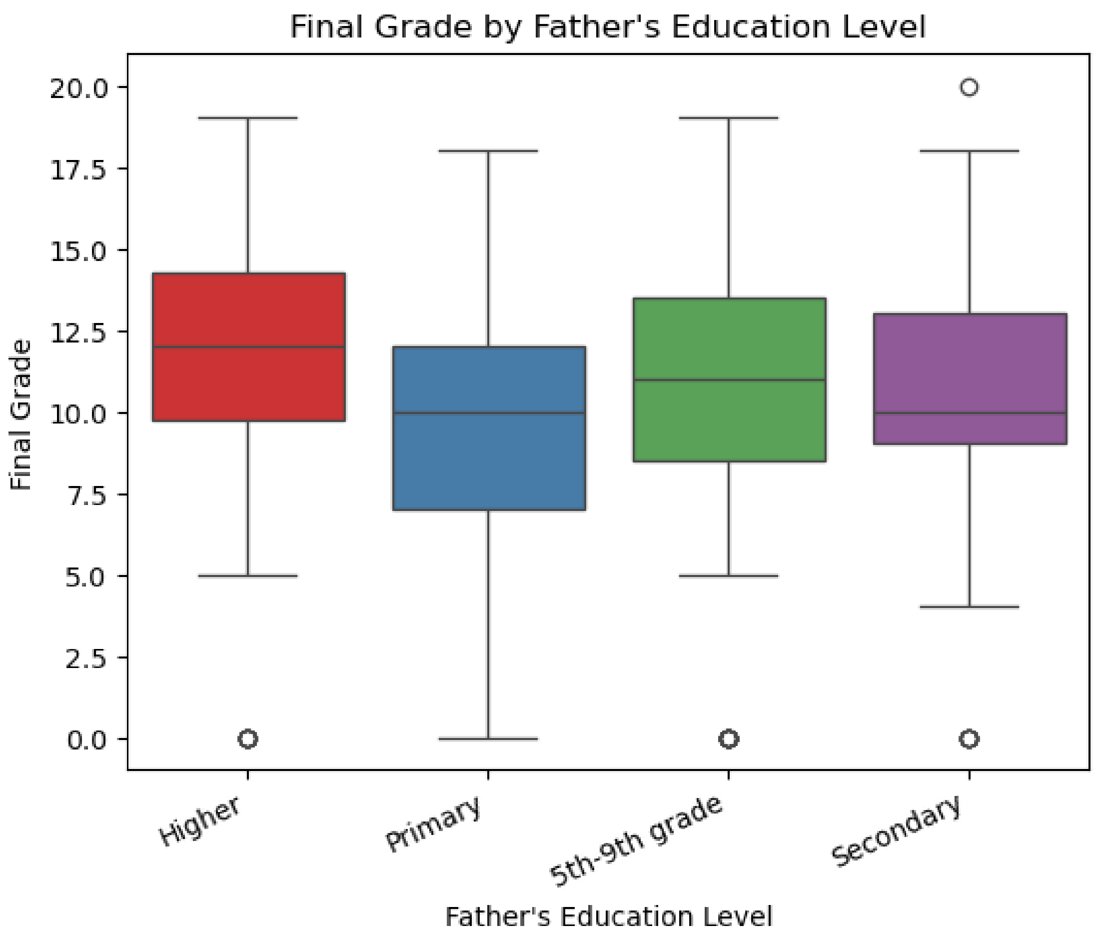
```
Done by Ryann Alvarez
                  finalGrade
fatherEdu_recode
5th-9th grade      10.260870
Higher             11.364583
Primary             9.158537
Secondary          10.660000
```

Similar to mother's education level, we can see that the average final grade increases as father's education level increases.

What do the descriptive statistics look like in a visualization? Let's use a boxplot.

```python
print('Done by Ryann Alvarez')

# Create a boxplot showing the distribution of final grades across
mother's educational levels
sns.boxplot(x='fatherEdu_recode', y='finalGrade',
data=parent_education_df, palette='Set1')

# Adjust lavels
plt.xticks(rotation=25, ha='right')

# Plot title
plt.title("Final Grade by Father's Education Level")

# Plot axes
plt.xlabel("Father's Education Level")
plt.ylabel("Final Grade")

# Show plot
plt.show()
```

```
Done by Ryann Alvarez
```



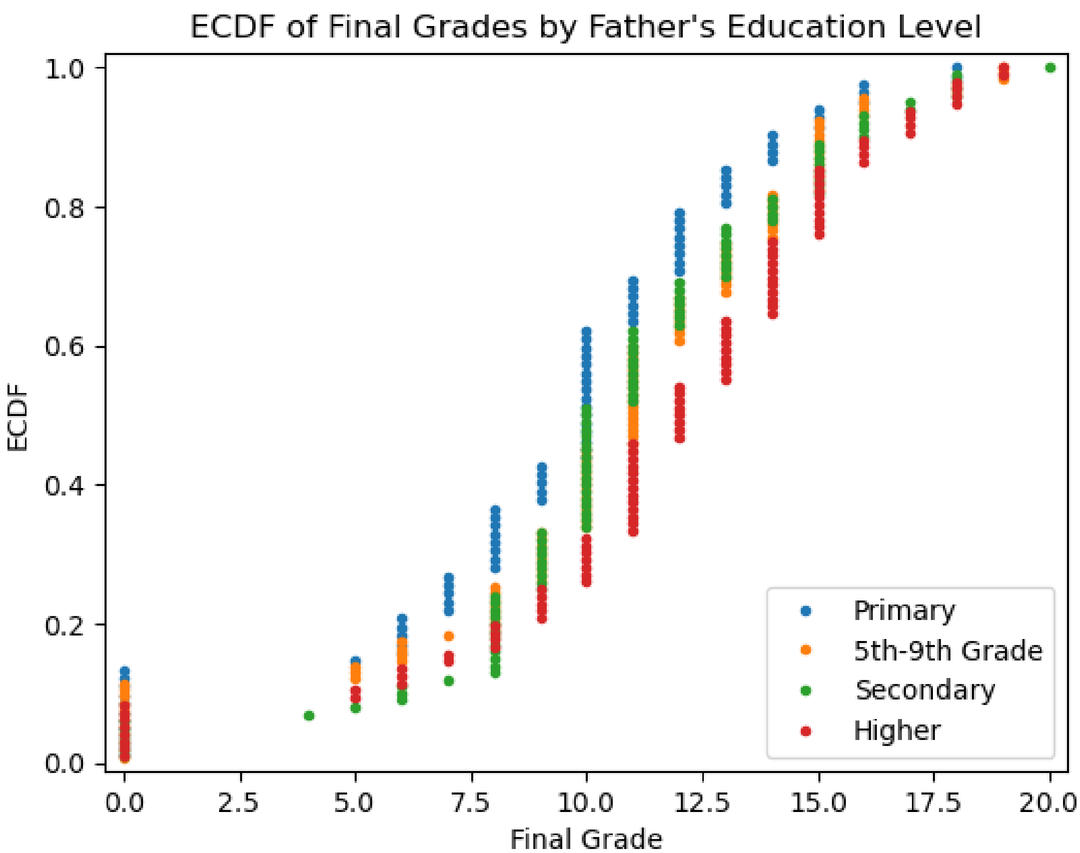
Similar to the mother's education level, we see that students whose father's education level is primary education score the lowest out of the other categories. They also have a wide dispersion of scores - wider than the other categories. Interestingly, students whose father's education level is secondary have outliers on the low and high ends. We see that students whose father's education level is higher education score the highest out of the other categories.

Let's see the distribution of scores by father's education level using an ECDF.

```
print('Done by Ryann Alvarez')

# Compute arrays for ECDF
fatherEdu_1 = student_df[student_df['fatherEdu'] == 1]['finalGrade']
fatherEdu_2 = student_df[student_df['fatherEdu'] == 2]['finalGrade']
fatherEdu_3 = student_df[student_df['fatherEdu'] == 3]['finalGrade']
fatherEdu_4 = student_df[student_df['fatherEdu'] == 4]['finalGrade']

# Compute ECDFs
x_fatherEdu_1, y_fatherEdu_1 = ecdf(fatherEdu_1)
x_fatherEdu_2, y_fatherEdu_2 = ecdf(fatherEdu_2)
x_fatherEdu_3, y_fatherEdu_3 = ecdf(fatherEdu_3)
x_fatherEdu_4, y_fatherEdu_4 = ecdf(fatherEdu_4)
```

```
# Plot all ECDFs on the same plot
_ = plt.plot(x_fatherEdu_1, y_fatherEdu_1, marker = '.', linestyle =
'none', label = "Primary")
_ = plt.plot(x_fatherEdu_2, y_fatherEdu_2, marker = '.', linestyle =
'none', label = "5th-9th Grade")
_ = plt.plot(x_fatherEdu_3, y_fatherEdu_3, marker = '.', linestyle =
'none', label = "Secondary")
_ = plt.plot(x_fatherEdu_4, y_fatherEdu_4, marker = '.', linestyle =
'none', label = "Higher")

# Make nice margins
plt.margins(0.02)

# Annotate the plot add a legend locate it at the lower right
_ = plt.title("ECDF of Final Grades by Father's Education Level")
_ = plt.xlabel('Final Grade')
_ = plt.ylabel('ECDF')
plt.legend(loc='lower right')

# Display the plot
plt.show()
```

Done by Ryann Alvarez



Similar to the ECDF for mother's educaiton level, the ECDF shows us that students whose father's education level is higher education have a higher probability of scoring higher on the final compared to all other groups. Also, students whose father's education level is primary education have a lower probability of scoring lower on the final compared to all other groups.

Let's run an ANOVA for father's education level.

```
print('Done by Ryann Alvarez')

# Import libraries for one-way ANOVA
from scipy.stats import f_oneway

# Use function that was defined previously

# H0: Mean final grade is the same across all four father education
groups (i.e., Primary, 5th-9th Grade, Secondary, Higher)
# H1: At least one group has a different mean final grade

# Conduct ANOVA
anova_test('fatherEdu_recode')

Done by Ryann Alvarez
fatherEdu_recode: F = 3.64, p = 0.0130
```

Since our p = 0.0130, it is less than p = 0.05 - we can reject the null hypothesis. This suggests evidence that final grades differ depending on their father's education level. Note that this evidence is not nearly as strong as the evidence for mother's education level.

Question #5

Does access to family educational support, extra educational support, or paid classes improve student performance?

Family Support

Let's start by looking at family support.

```
print('Done by Ryann Alvarez')

# Use .value_counts() to identify how many students do and do not receive family support
student_df['familySupport'].value_counts()
```

```
Done by Ryann Alvarez

familySupport
yes    242
no     153
Name: count, dtype: int64
```

More students do receive family support than not.

```
print('Done by Ryann Alvarez')

# Use .groupby to get descriptive statistics
student_df.groupby('familySupport')['finalGrade'].describe()
```

```
Done by Ryann Alvarez
```

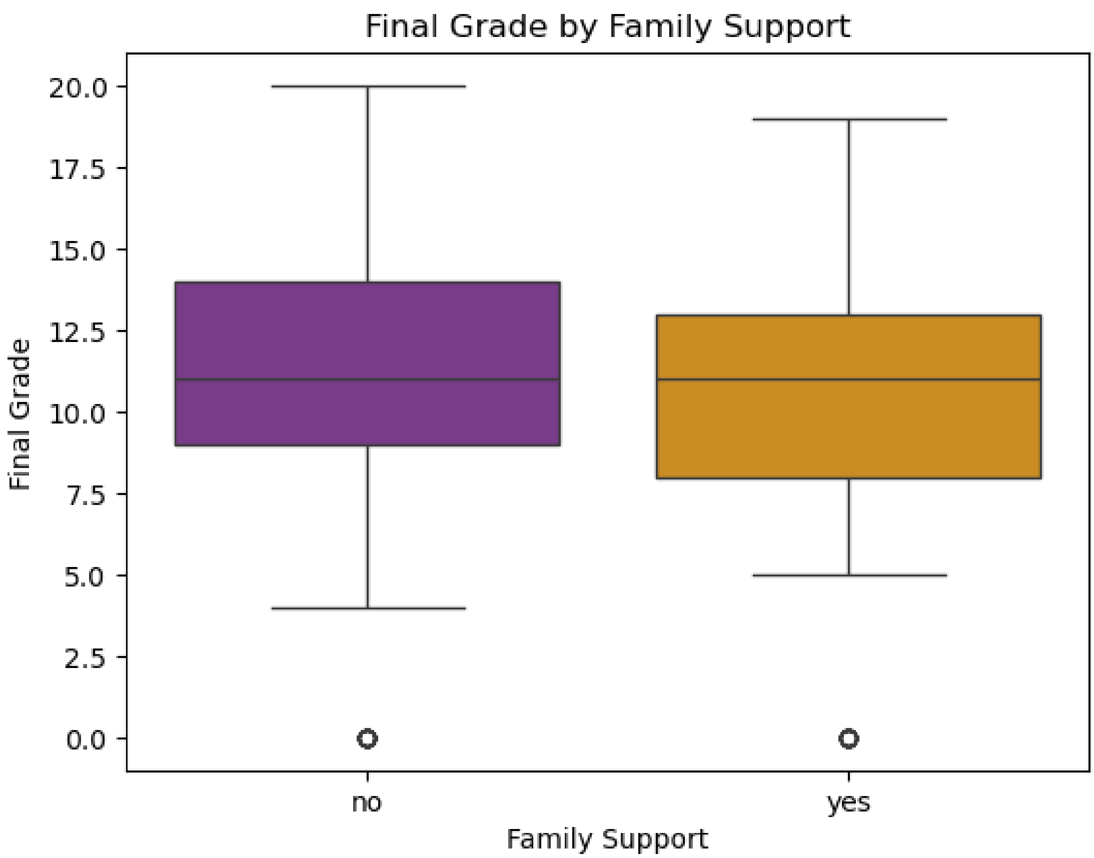
	count	mean	std	min	25%	50%	75%	max
familySupport								
no	153.0	10.640523	4.636262	0.0	9.0	11.0	14.0	20.0
yes	242.0	10.272727	4.550318	0.0	8.0	11.0	13.0	19.0

Based off the table of descriptives, it seems that students that do not receive family support perform better academically compared to students that do receive family support, which is not what I would expect.

```
print('Done by Ryann Alvarez')

# Create boxplot
sns.boxplot(x='familySupport', y='finalGrade', data=student_df, palette='CMRmap')
```

```
# Plot title
plt.title("Final Grade by Family Support")

# Plot axes
plt.xlabel("Family Support")
plt.ylabel("Final Grade")

# Show plot
plt.show()
```

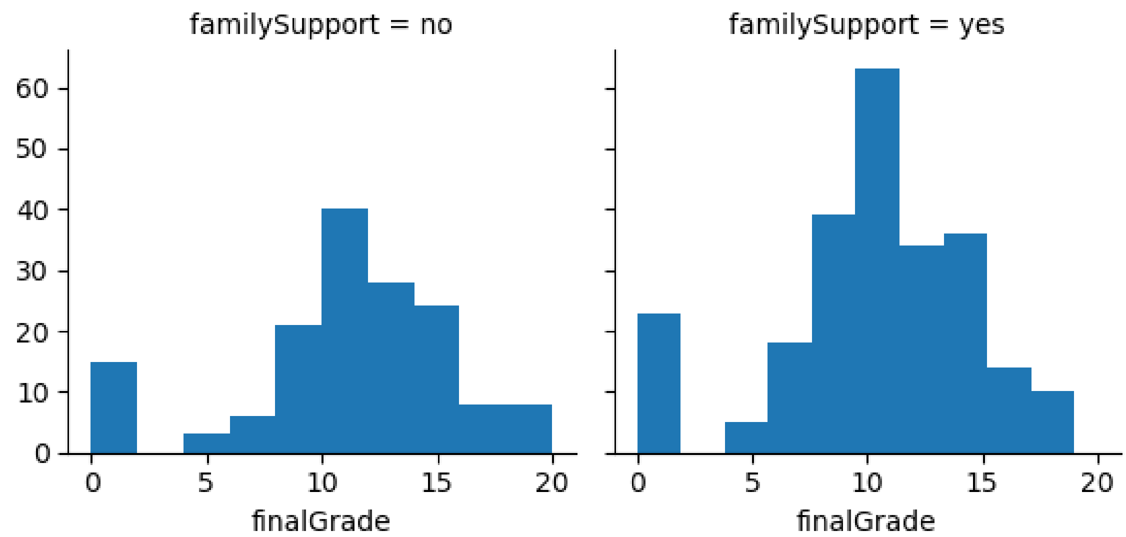
```
Done by Ryann Alvarez
```



The boxplot visualizes just that - students that do not receive family support perform better academically compared to students that do receive family support.

```
print('Done by Ryann Alvarez')

# Use FacetGrid to see how the distribution of final grades compare
g = sns.FacetGrid(data=student_df, col='familySupport')
g.map(plt.hist, 'finalGrade')

# Show plot
plt.show()
```

```
Done by Ryann Alvarez
```



These side-by-side histograms show a similar distribution of grades for both groups, but there are more students that receive family support. This aligns with what was found using .value_counts() previously.

Let's see if the differences between family support are significant using a t-test.

```
print('Done by Ryann Alvarez')

# Import required libraries for a t-test
from scipy import stats
from scipy.stats import ttest_ind

# Identify family support yes and no
yes_familySupport = student_df[student_df['familySupport'] == 'yes']
['finalGrade']
no_familySupport = student_df[student_df['familySupport'] == 'no']
['finalGrade']

# H0: There is no significant difference between the means
# H1: There is a significant difference between the means

# Perform independent samples t-test
# Returns t-test statistic and p-value
stat, p = stats.ttest_ind(yes_familySupport, no_familySupport)

# Print result
print(f"Family support t-test: t = {stat:.3f}, p = {p:.4f}")

Done by Ryann Alvarez
Family support t-test: t = -0.777, p = 0.4377
```

Our p-value of = 0.4377 is not less than p = 0.05, so we fail to reject the null hypothesis and say there is no significant difference between the the means of those that do and do not receive family support. Since the t-test statistic is negative, that means the mean of the first group (i.e., those that receive family support) is smaller than the mean of the second group (i.e., those that do not receive family support).

School Support

Now, let's look at school support.

```python
print('Done by Ryann Alvarez')

# Use .value_counts() to identify how many students do and do not receive school support
student_df['schoolSupport'].value_counts()
```

```
Done by Ryann Alvarez

schoolSupport
no     344
yes     51
Name: count, dtype: int64
```

Wow, plenty more students do not receive school support compared to those that do.

```python
print('Done by Ryann Alvarez')

# Use .groupby to get descriptive statistics
student_df.groupby('schoolSupport')['finalGrade'].describe()
```

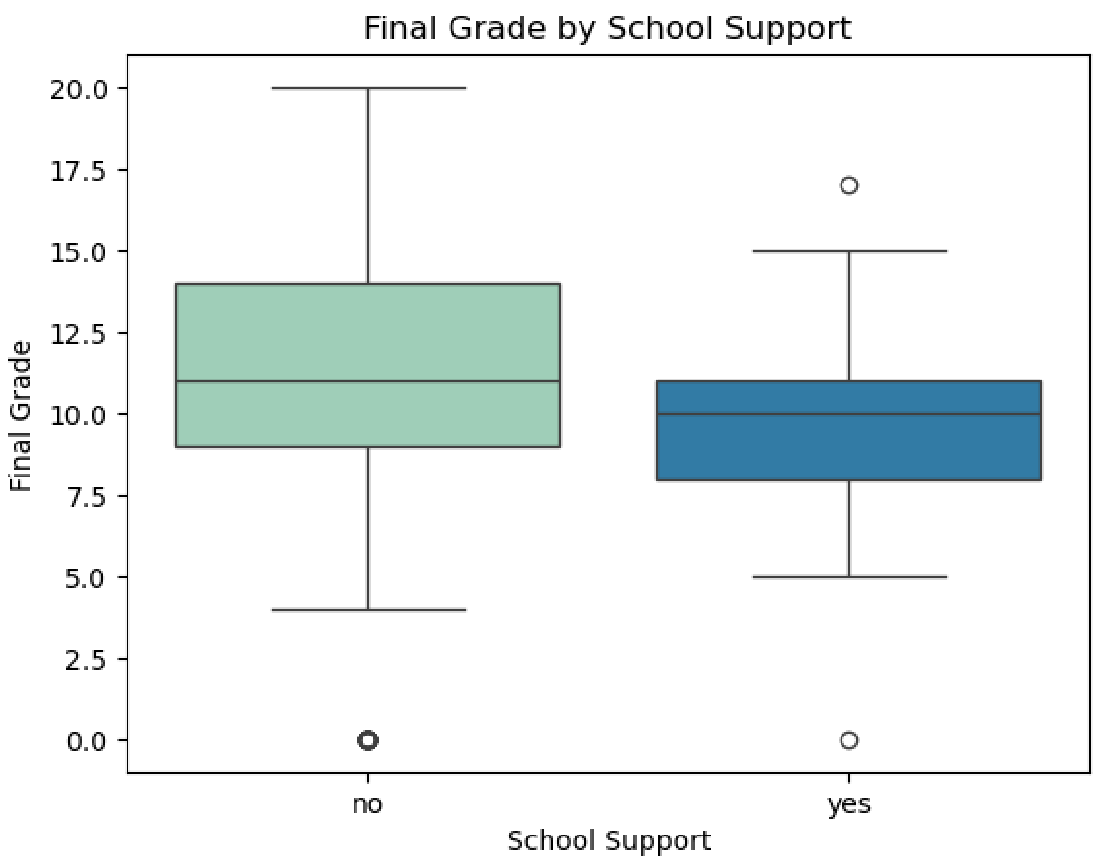
```
Done by Ryann Alvarez
```

	count	mean	std	min	25%	50%	75%	max
schoolSupport								
no	344.0	10.561047	4.769533	0.0	9.0	11.0	14.0	20.0
yes	51.0	9.431373	2.865344	0.0	8.0	10.0	11.0	17.0

Based off the table of descriptives, it seems that students that do not receive school support perform better academically compared to students that do receive school support. This is not what I would expect.

```python
print('Done by Ryann Alvarez')

# Create boxplot
sns.boxplot(x='schoolSupport', y='finalGrade', data=student_df, palette='YlGnBu')

# Plot title
plt.title("Final Grade by School Support")
```

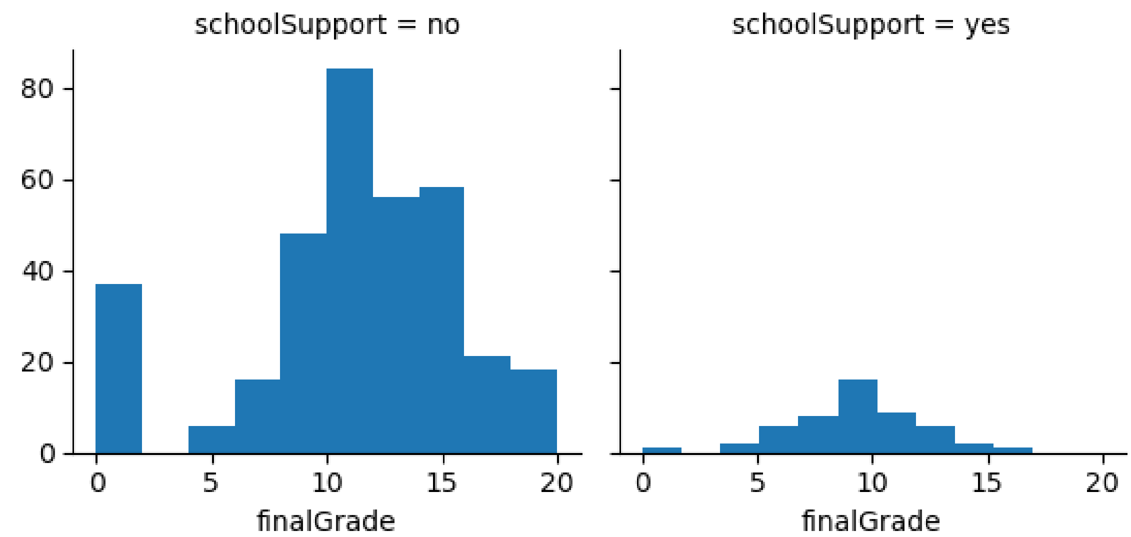
```
# Plot axes
plt.xlabel("School Support")
plt.ylabel("Final Grade")

# Show plot
plt.show()

Done by Ryann Alvarez
```



The boxplot visualizes what was found in the table of descriptives. Also, we see less dispersion of grades for students that do receive school support. It is important to note that the whisker for 'No' to school support reaches the maximum score of 20, while the whisker for 'Yes' to school support does not.

```
print('Done by Ryann Alvarez')

# Use FacetGrid to see how the distribution of final grades compare
g = sns.FacetGrid(data=student_df, col='schoolSupport')
g.map(plt.hist, 'finalGrade')

# Show plot
plt.show()
```

```
Done by Ryann Alvarez
```



These side-by-side histograms show a similar distribution of grades, but there are signficantly more students that do not receive school support. This aligns with what was found using .value_counts() previously. With such drasticly different sample sizes, it is possible that our results are not completely representative or accurate.

Let's use a t-test to see if the differences between school support are significant.

```
print('Done by Ryann Alvarez')

# Import required libraries for a t-test
from scipy import stats
from scipy.stats import ttest_ind

# Identify school support yes and no
yes_schoolSupport = student_df[student_df['schoolSupport'] == 'yes']
['finalGrade']
no_schoolSupport = student_df[student_df['schoolSupport'] == 'no']
['finalGrade']

# H0: There is no significant difference between the means
# H1: There is a significant difference between the means

# Perform independent samples t-test
# Returns t-test statistic and p-value
stat, p = stats.ttest_ind(yes_schoolSupport, no_schoolSupport)

# Print result
print(f"School support t-test: t = {stat:.3f}, p = {p:.4f}")
```

```
Done by Ryann Alvarez
School support t-test: t = -1.647, p = 0.1004
```

Our p-value of = 0.1004 is not less than p = 0.05, so we fail to reject the null hypothesis and say there is no significant difference between the the means of those that do and do not receive school support. Since the t-test statistic is negative, that means the mean of the first group (i.e., those that receive school support) is smaller than the mean of the second group (i.e., those that do not receive school support).

Paid Classes

Lastly, let's explore paid classes.

```python
print('Done by Ryann Alvarez')

# Use .value_counts() to identify how many students do and do not receive family support
student_df['paidClasses'].value_counts()
```

```
Done by Ryann Alvarez

paidClasses
no     214
yes    181
Name: count, dtype: int64
```

More students do not attend paid classes compared to those that do - thought the difference is not as extreme as the difference for school support.

```python
print('Done by Ryann Alvarez')

# Use .groupby to get descriptive statistics
student_df.groupby('paidClasses')['finalGrade'].describe()
```

```
Done by Ryann Alvarez
```

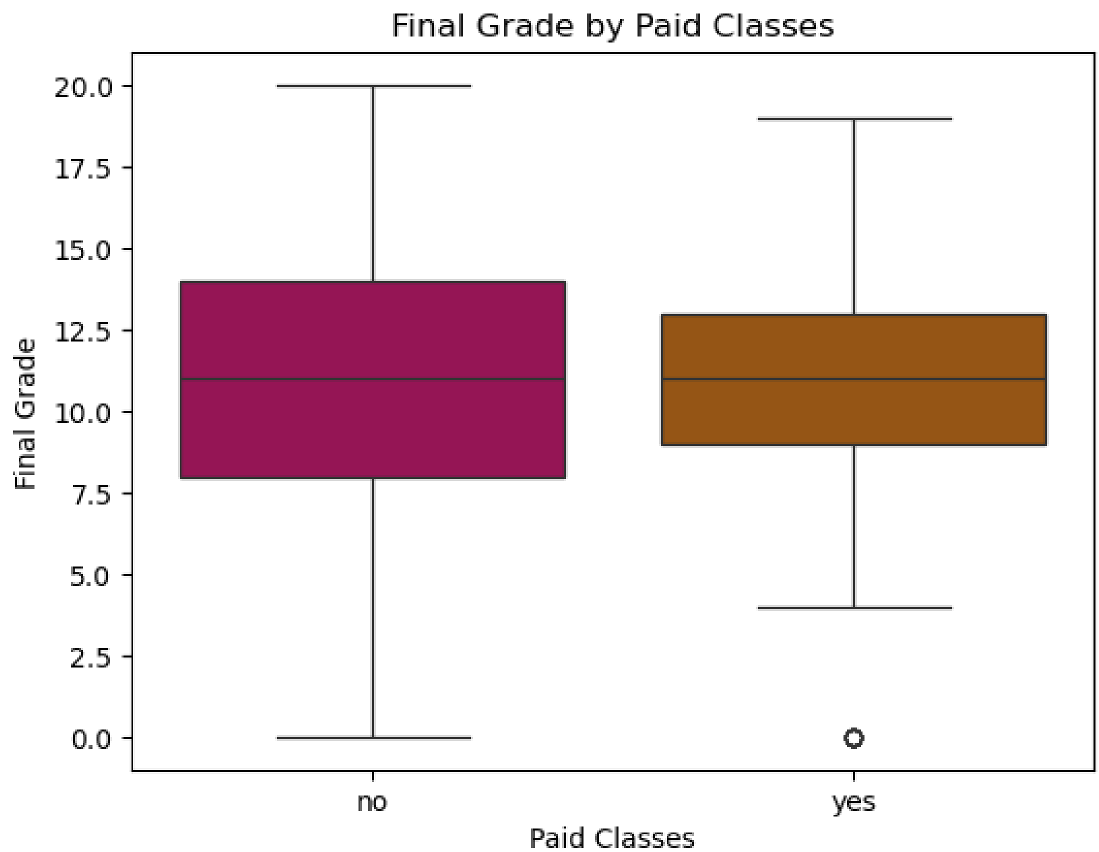
	count	mean	std	min	25%	50%	75%	max
paidClasses								
no	214.0	9.985981	5.126090	0.0	8.0	11.0	14.0	20.0
yes	181.0	10.922652	3.791011	0.0	9.0	11.0	13.0	19.0

Based off the table, it seems that students that do attend paid classes school support perform better academically compared to students that do not attend paid classes. This aligns with what I would expect.

```python
print('Done by Ryann Alvarez')

# Create boxplot
sns.boxplot(x='paidClasses', y='finalGrade', data=student_df, palette='brg')

# Plot title
plt.title("Final Grade by Paid Classes")
```

```
# Plot axes
plt.xlabel("Paid Classes")
plt.ylabel("Final Grade")

# Show plot
plt.show()
```

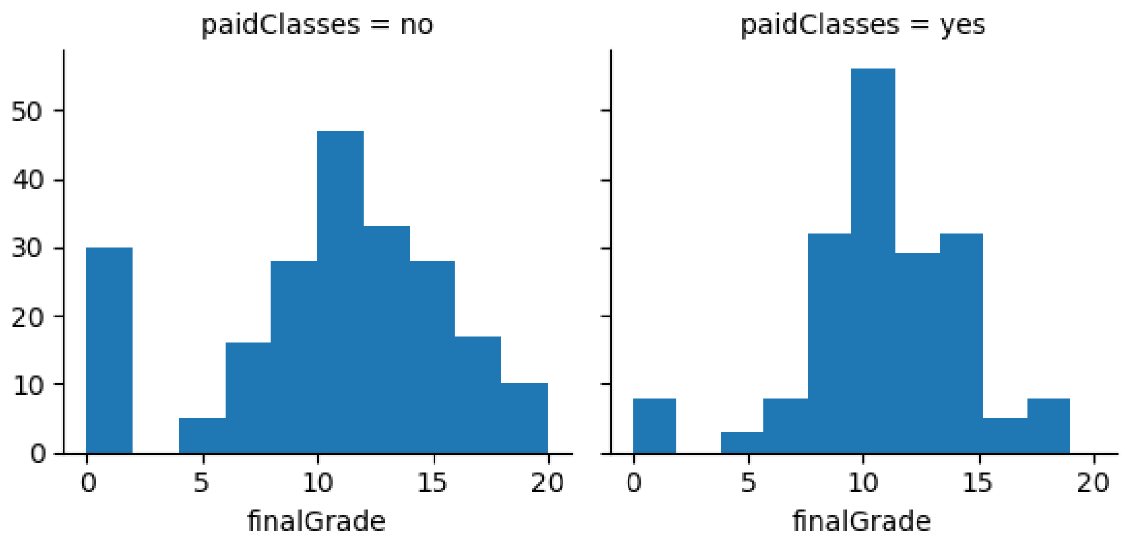
```
Done by Ryann Alvarez
```



The boxplot shows less dispersion of grades for students that do attend paid classes. It is important to note that the whiskers for 'No' to paid classes reach both the maximum score of 20 and minimum score of 0, while the whisker for 'Yes' to school support does not reach either.

```
print('Done by Ryann Alvarez')

# Use FacetGrid to see how the distribution of final grades compare
g = sns.FacetGrid(data=student_df, col='paidClasses')
g.map(plt.hist, 'finalGrade')

# Show plot
plt.show()
```

```
Done by Ryann Alvarez
```



The distribution of grades between groups is similar, but the graph for students that attend paid classes demostrates more positive kurtosis. The graph for students that do not attend paid classes highlights the large amount of low scores (e.g., 0).

And finally, let's determine if the differences between paid classes are significant using a t-test.

```
print('Done by Ryann Alvarez')

# Import required libraries for a t-test
from scipy import stats
from scipy.stats import ttest_ind

# Identify school support yes and no
yes_paidClasses = student_df[student_df['paidClasses'] == 'yes']
['finalGrade']
no_paidClasses = student_df[student_df['paidClasses'] == 'no']
['finalGrade']

# H0: There is no significant difference between the means
# H1: There is a significant difference between the means

# Perform independent samples t-test
# Returns t-test statistic and p-value
stat, p = stats.ttest_ind(yes_paidClasses, no_paidClasses)

# Print result
print(f"Paid classes t-test: t = {stat:.3f}, p = {p:.4f}")

Done by Ryann Alvarez
Paid classes t-test: t = 2.033, p = 0.0428
```

Our p-value of = 0.0428 is less than p = 0.05, so we can reject the null hypothesis and say there is a significant difference between the the means of those that do and do not attend paid classes. Since the t-test statistic is positive, that means the mean of the first group (i.e., those that attend paid classes) is larger than the mean of the second group (i.e., those that attend paid classes).

Question #6

How much do students study each week, and how is that related to their grades?

Let's first begin by understanding how much do students study?

```
print('Done by Ryann Alvarez')

# Use .describe() to get descriptive statistics for studyTime
student_df['studyTime'].describe()
```

```
Done by Ryann Alvarez
```

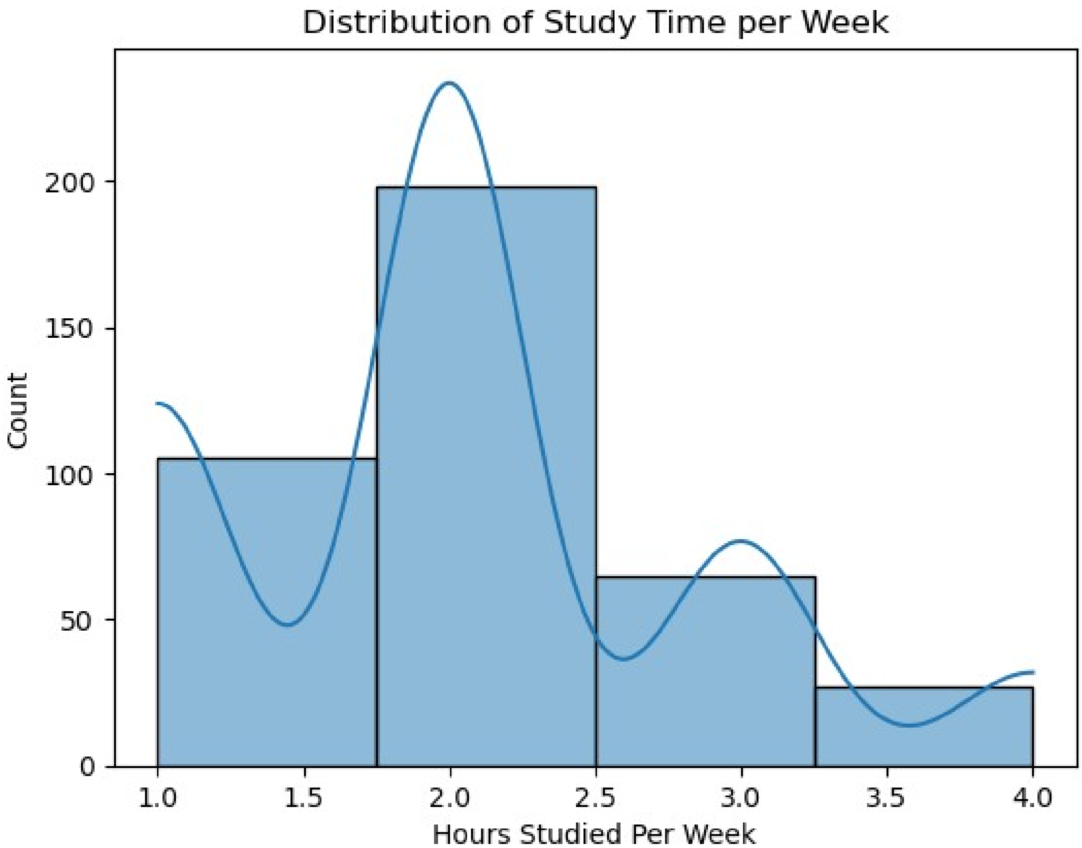
```
count    395.000000
mean       2.035443
std        0.839240
min        1.000000
25%        1.000000
50%        2.000000
75%        2.000000
max        4.000000
Name: studyTime, dtype: float64
```

Based off of the table of descriptives, most students study for about 2 hours per week.

Let's use a histogram to show the distribution of hours studied per week.

```
print('Done by Ryann Alvarez')

# Create histogram to show the distribution of study time per week
sns.histplot(student_df['studyTime'], bins = 4, kde=True)

# Plot title
plt.title("Distribution of Study Time per Week")

# Plot axes
plt.xlabel("Hours Studied Per Week")
plt.ylabel("Count")

# Show plot
plt.show()
```

```
Done by Ryann Alvarez
```



The histogram shows that most students study 2 hours or less each week. We also see peaks at around 1 hour and 3 hours studied per week through the KDE line.

Now, is study time related to grades? Let's begin with some descriptives.

```
print('Done by Ryann Alvarez')

# Organize grades by student's study time
by_studyTime = student_df.groupby('studyTime')[['firstGrade',
'secondGrade', 'finalGrade']]

# Display mean
by_studyTime.mean()
```

```
Done by Ryann Alvarez
```

	firstGrade	secondGrade	finalGrade
studyTime			
1	10.438095	10.276190	10.047619
2	10.651515	10.505051	10.171717
3	12.046154	11.507692	11.400000
4	11.888889	12.037037	11.259259

This table gives us a quick look at the average grades broken down by hours studied. We can see that for the first and final grades, students performed (on average) best when studying for 3 hours per week. For the second grade, students performed best when studying for 4 hours per week - again, on average.

For proper visualizations, we need to "melt" the data.

```
print('Done by Ryann Alvarez')

# Use .melt to reshape the data
studyTime_melted = pd.melt(student_df, id_vars='studyTime',
                           value_vars=['firstGrade', 'secondGrade',
'finalGrade'],
                           var_name='gradeType', value_name='grade')

# Show result (easier to work with for visualizations)
studyTime_melted.head()
```

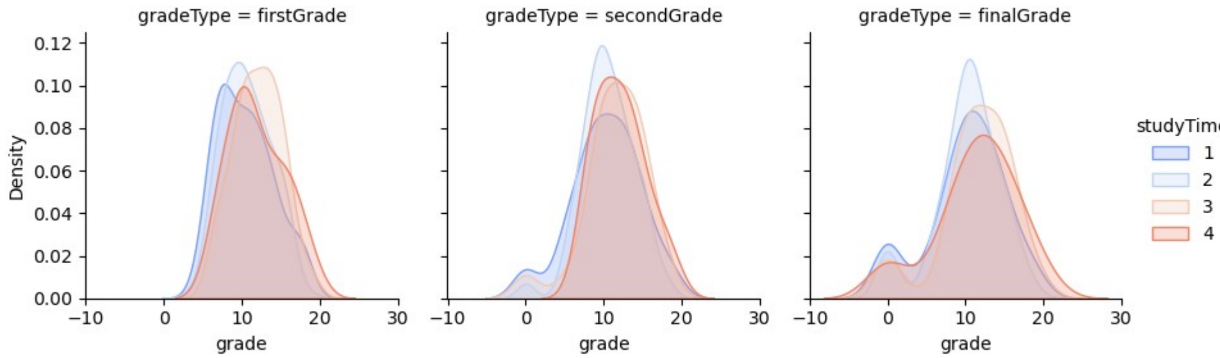
```
Done by Ryann Alvarez
```

	studyTime	gradeType	grade
0	2	firstGrade	5
1	2	firstGrade	5
2	2	firstGrade	7
3	3	firstGrade	15
4	2	firstGrade	6

Great, this is what I want, so let's get into some visualizations.

```
print('Done by Ryann Alvarez')

# Use FaceGrid to show the distributions (i.e., first, second, and
final grade) side-by-side
# Add hue='studyTime' to distinguish by the number of hours studied
per week
g = sns.FacetGrid(data=studyTime_melted, col='gradeType',
hue='studyTime', palette='coolwarm')
g.map(sns.kdeplot, 'grade', fill=True).add_legend()

# Show plot
plt.show()
```

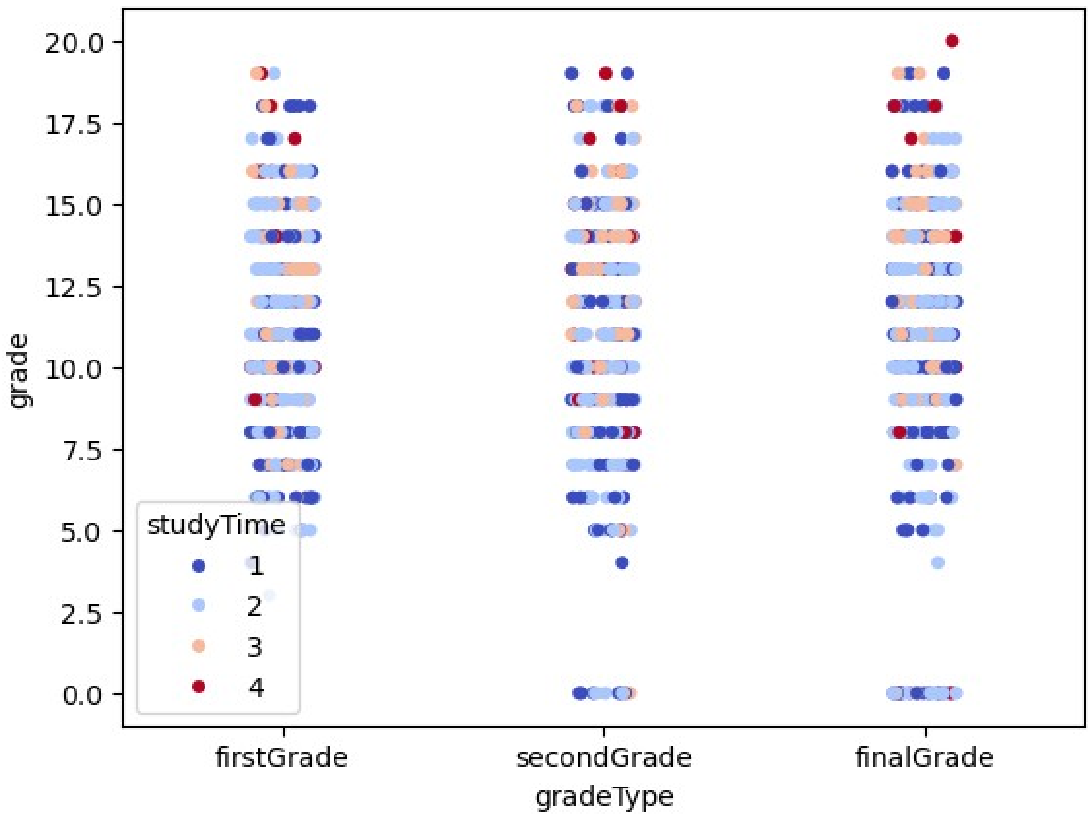
```
Done by Ryann Alvarez
```



I used a kdeplot because it will be easier to identify trends compared to a histplot. We can see that students who studied for 2 hours have strong peaks for the second and final grades, demostrating positive kurtosis. An example of negative kurtosis would be the students who studied for 4 hours for the final grade. We can also see that the spread of scores is the smallest for the first grade and largest for the final grade, and there are more and more outliers as the grades progress (i.e., from first to final).

```
print('Done by Ryann Alvarez')

# Use stripplot to show the distributions (i.e., first, second, and
final grade) side-by-side
# Not using a swarmplot because swarmplots are not recommended for
large datasets

# Add hue='studyTime' to distinguish by the number of hours studied
per week
sns.stripplot(x='gradeType', y='grade', data=studyTime_melted,
hue='studyTime', palette='coolwarm')

# Show plot
plt.show()
```

```
Done by Ryann Alvarez
```



The stripplot shows more cooler-toned dots (i.e., blue and purple) toward the lower ends of the grades and more warmer-toned dots (i.e., pink and red) toward the higher end. Since the cooler-toned dots represent students that studied for less time (i.e., 1-2 hours), then that means students who studied for less time did not perform as well academically compared to the students that studied for more time (i.e., 3-4 hours), which are represented by the warmer-toned dots.

Moving onto the analyses...

```
print('Done by Ryann Alvarez')

# Import libraries for Pearson's Correlation test
from scipy.stats import pearsonr

# H0: There is no relationship between first grade and study time
# H1: There is a relationship between first grade and study time

# Tests the relationship between two variables (i.e., first and final
grade)
stat, p = pearsonr(student_df['firstGrade'], student_df['studyTime'])

# Print stat (correlation coeffecient) and p-value for Pearson's
Correlation test
print('stat=%.3f, p=%.3f' % (stat, p))
```

```
Done by Ryann Alvarez
stat=0.161, p=0.001
```

The Pearson Correlation test reveals a weak, positive, linear relationship for the relationship between first grade and study time (r = 0.161). Since p = 0.001, it is less than p = 0.05, so we can reject the null hypothesis. This suggests strong evidence that first grade and study time are related.

How about the relationship between second grade and study time?

```
print('Done by Ryann Alvarez')

# Import libraries for Pearson's Correlation test
from scipy.stats import pearsonr

# H0: There is no relationship between second grade and study time
# H1: There is a relationship between second grade and study time

# Tests the relationship between two variables (i.e., first and final
grade)
stat, p = pearsonr(student_df['secondGrade'], student_df['studyTime'])

# Print stat (correlation coeffecient) and p-value for Pearson's
Correlation test
print('stat=%.3f, p=%.3f' % (stat, p))
```

```
Done by Ryann Alvarez
stat=0.136, p=0.007
```

The Pearson Correlation test also reveals a weak, positive, linear relationship for the relationship between second grade and study time (r = 0.136). Since p = 0.007, it is less than p = 0.05, so we can reject the null hypothesis. This suggests strong evidence that second grade and study time are related.

How about the relationship between final grade and study time?

```
print('Done by Ryann Alvarez')

# Import libraries for Pearson's Correlation test
from scipy.stats import pearsonr

# H0: There is no relationship between final grade and study time
# H1: There is a relationship between final grade and study time

# Tests the relationship between two variables (i.e., first and final
grade)
stat, p = pearsonr(student_df['finalGrade'], student_df['studyTime'])

# Print stat (correlation coeffecient) and p-value for Pearson's
```

```
Correlation test
print('stat=%.3f, p=%.3f' % (stat, p))

Done by Ryann Alvarez
stat=0.098, p=0.052
```

The Pearson Correlation test also reveals a weak, positive, linear relationship for the relationship between final grade and study time (r = 0.098). The p-value of = 0.052 is greater than p = 0.05, so we fail to reject the null hypothesis. There is not enough evidence to suggest that this relationship is significant.

Question #7

Do social behaviors like going out and alcohol consumption impact final grades?

First, I'm going to begin by pulling at the necessary variables into a separate DataFrame.

```
print('Done by Ryann Alvarez')

# Pull only necessary variables into new DataFrame
social_behaviors_df = pd.DataFrame(student_df[['goOutFreq',
'weekdayAlc', 'weekendAlc', 'finalGrade']])

# Preview the new DataFrame
social_behaviors_df.head()

Done by Ryann Alvarez
```

	goOutFreq	weekdayAlc	weekendAlc	finalGrade
0	4	1	1	6
1	3	1	1	6
2	2	2	3	10
3	2	1	1	15
4	2	1	2	10

Recall that the variables 'goOutFreq', 'weekdayAlc', and 'weekendAlc' are collected on a Likert scale from 1 to 5, with 1 being very low and 5 being very high. Let's create a function to recode the values:

```
print('Done by Ryann Alvarez')

# Define a function that recodes the values of the variables
'goOutFreq', 'weekdayAlc', and 'weekendAlc' into a new variable
def recode_socialBehaviors(behavior):
    """
    Converts values of the 'goOutFreq', 'weekdayAlc', and 'weekendAlc'
variables from:
    1 to 'Very Low'
    2 to 'Low'
```

```
    3 to 'Moderate'
    4 to 'High'
    5 to 'Very High'
    """
    # Return 'Very Low' if behavior is 1
    if behavior == 1:
        return 'Very Low'

    # Return 'Low' if behavior is 2
    elif behavior == 2:
        return 'Low'

    # Return 'Moderate' if behavior is 3
    elif behavior == 3:
        return 'Moderate'

    # Return 'High' if behavior is 4
    elif behavior == 4:
        return 'High'

    # Return 'Very High' if behavior is 4
    elif behavior == 5:
        return 'Very High'
```

```
Done by Ryann Alvarez
```

```
print('Done by Ryann Alvarez')

# Apply the recode_socialBehaviors function to goOutFreq
social_behaviors_df['goOutFreq_recode'] =
social_behaviors_df.goOutFreq.apply(recode_socialBehaviors)

# Apply the recode_socialBehaviors function to weekendAlc
social_behaviors_df['weekendAlc_recode'] =
social_behaviors_df.weekendAlc.apply(recode_socialBehaviors)

# Apply the recode_socialBehaviors function to weekdayAlc
social_behaviors_df['weekdayAlc_recode'] =
social_behaviors_df.weekdayAlc.apply(recode_socialBehaviors)

# Preview of the data
social_behaviors_df.head()
```

```
Done by Ryann Alvarez
```

```
   goOutFreq  weekdayAlc  weekendAlc  finalGrade goOutFreq_recode  \
0          4           1           1           6             High
1          3           1           1           6         Moderate
2          2           2           3          10              Low
3          2           1           1          15              Low
4          2           1           2          10              Low
```

```
  weekendAlc_recode weekdayAlc_recode
0          Very Low          Very Low
1          Very Low          Very Low
2          Moderate               Low
3          Very Low          Very Low
4               Low          Very Low
```

Perfect, now let's get to work!

Let's establish just how often student's go out with friends and consume alcohol on the weekdays and weekends.

```
print('Done by Ryann Alvarez')

# Use .describe() to see how often students go out and consume alcohol
social_behaviors_df[['goOutFreq', 'weekdayAlc',
'weekendAlc']].describe()
```

```
Done by Ryann Alvarez
```

	goOutFreq	weekdayAlc	weekendAlc
count	395.000000	395.000000	395.000000
mean	3.108861	1.481013	2.291139
std	1.113278	0.890741	1.287897
min	1.000000	1.000000	1.000000
25%	2.000000	1.000000	1.000000
50%	3.000000	1.000000	2.000000
75%	4.000000	2.000000	3.000000
max	5.000000	5.000000	5.000000

We can see that on average, students go out a moderate amount. On average, students do not consume alcohol frequently, but there is increase in alcohol consumption on the weekends.

Going Out Frequency

Let's look more closely at going out frequency.

```
print('Done by Ryann Alvarez')

# Group final grade by going out frequency
by_goOutFreq = social_behaviors_df.groupby('goOutFreq')['finalGrade']

# Show table of descriptives
by_goOutFreq.describe()
```

```
Done by Ryann Alvarez
```

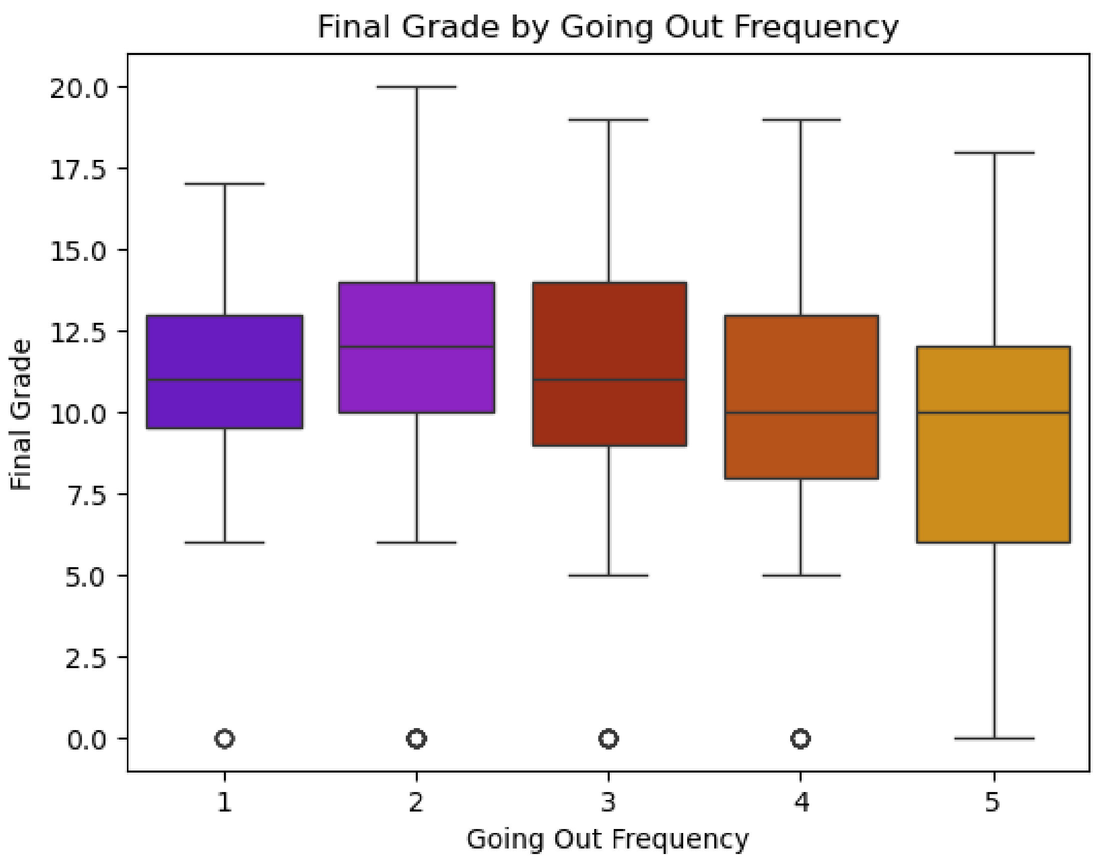
	count	mean	std	min	25%	50%	75%	max
goOutFreq								

```
1            23.0   9.869565  5.336873  0.0   9.5  11.0  13.0  17.0
2           103.0  11.194175  4.535391  0.0  10.0  12.0  14.0  20.0
3           130.0  10.961538  4.210367  0.0   9.0  11.0  14.0  19.0
4            86.0   9.651163  4.421252  0.0   8.0  10.0  13.0  19.0
5            53.0   9.037736  5.072408  0.0   6.0  10.0  12.0  18.0
```

The table gives us lots of great descriptive information, but let's create a visualization for this and then discuss.

```python
print('Done by Ryann Alvarez')

# Create boxplot to look closer at going out frequency
sns.boxplot(x='goOutFreq', y='finalGrade', data=social_behaviors_df,
palette='gnuplot')

# Plot title
plt.title("Final Grade by Going Out Frequency")

# Plot axes labels
plt.xlabel('Going Out Frequency')
plt.ylabel('Final Grade')

# Show plot
plt.show()
```

```
Done by Ryann Alvarez
```



Using a boxplot, we can see that students that go out a 'low' amount (i.e., 2) scored the highest on the final grade compared to their peers. What is interesting is that these students scored higher than students that go out a 'very low' amount (i.e., 1). This suggests that some social time could be good! Life is all about balance!

Also, students that go out a 'very high' amount (i.e., 5) scored the lowest on the final grade compared to their peers. This makes sense as we can understand that going out more can result in less time to study.

Let's see how an ECDF will represent the data...

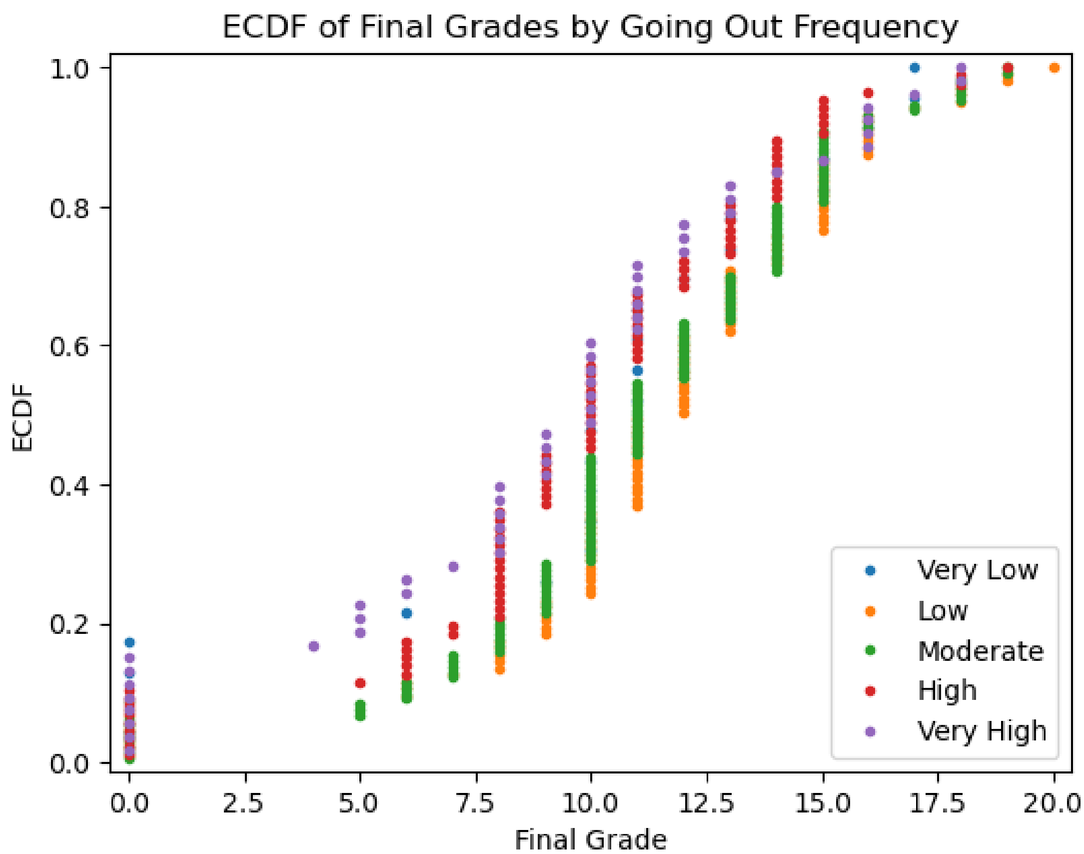
```
print('Done by Ryann Alvarez')

def ecdf(data):
    """Compute ECDF for a one-dimensional array of measurements."""

    # Number of data points: n
    n = len(data)

    # x-data for the ECDF: x
    x = np.sort(data)

    # y-data for the ECDF: y
    y = np.arange(1, n+1) / n
```

```
    return x, y
```

Done by Ryann Alvarez

```
print('Done by Ryann Alvarez')

# Compute arrays for ECDF
goOutFreq_1 = social_behaviors_df[social_behaviors_df['goOutFreq'] ==
1]['finalGrade']
goOutFreq_2 = social_behaviors_df[social_behaviors_df['goOutFreq'] ==
2]['finalGrade']
goOutFreq_3 = social_behaviors_df[social_behaviors_df['goOutFreq'] ==
3]['finalGrade']
goOutFreq_4 = social_behaviors_df[social_behaviors_df['goOutFreq'] ==
4]['finalGrade']
goOutFreq_5 = social_behaviors_df[social_behaviors_df['goOutFreq'] ==
5]['finalGrade']

# Compute ECDFs
x_goOutFreq_1, y_goOutFreq_1 = ecdf(goOutFreq_1)
x_goOutFreq_2, y_goOutFreq_2 = ecdf(goOutFreq_2)
x_goOutFreq_3, y_goOutFreq_3 = ecdf(goOutFreq_3)
x_goOutFreq_4, y_goOutFreq_4 = ecdf(goOutFreq_4)
x_goOutFreq_5, y_goOutFreq_5 = ecdf(goOutFreq_5)

# Plot all ECDFs on the same plot
_ = plt.plot(x_goOutFreq_1, y_goOutFreq_1, marker = '.', linestyle =
'none', label = "Very Low")
_ = plt.plot(x_goOutFreq_2, y_goOutFreq_2, marker = '.', linestyle =
'none', label = "Low")
_ = plt.plot(x_goOutFreq_3, y_goOutFreq_3, marker = '.', linestyle =
'none', label = "Moderate")
_ = plt.plot(x_goOutFreq_4, y_goOutFreq_4, marker = '.', linestyle =
'none', label = "High")
_ = plt.plot(x_goOutFreq_5, y_goOutFreq_5, marker = '.', linestyle =
'none', label = "Very High")

# Make nice margins
plt.margins(0.02)

# Annotate the plot add a legend locate it at the lower right
_ = plt.title("ECDF of Final Grades by Going Out Frequency")
_ = plt.xlabel('Final Grade')
_ = plt.ylabel('ECDF')
plt.legend(loc='lower right')

# Display the plot
plt.show()
```

Done by Ryann Alvarez



The ECDF shows us that students who go out a 'low' amount have a higher probability of scoring higher on the final compared to all other groups. Also, students who go out a 'very high' amount have a lower probability of scoring lower on the final compared to all other groups.

I'm curious to know if these categories are statistically significant. Let's conduct an ANOVA.

```
print('Done by Ryann Alvarez')

# Import libraries for one-way ANOVA
from scipy.stats import f_oneway

# Create a function that will conduct one way ANOVA test

# Function takes one argument, which is the name of a categorical
column
def anova_test(col):

    # social_behaviors_df.groupby(col) splits the data into groups
based on unique values
    # For each group, it pulls the finalGrade values using
group['finalGrade'].values
    groups = [group['finalGrade'].values for name, group in
social_behaviors_df.groupby(col)]
```

```
    # *groups unpacks the list so each array becomes a separate
argument
    # Returns the F-statistic and p-value
    stat, p = f_oneway(*groups)

    # Print result
    print(f"{col}: F = {stat:.2f}, p = {p:.4f}")

# H0: Mean final grade is the same across all going out frequencies
(i.e., Very Low, Low, Moderate, High, Very High)
# H1: At least one group has a different mean final grade

# Conduct ANOVA
anova_test('goOutFreq_recode')
```

```
Done by Ryann Alvarez
goOutFreq_recode: F = 3.15, p = 0.0144
```

Since our p = 0.0144, it is less than p = 0.05, and we can reject the null hypothesis. This suggests strong evidence that final grades differ depending on their going out frequency.

Alcohol Consumption

Let's repeat this with weekend and weekday alcohol consumption.

```
print('Done by Ryann Alvarez')

# Group final grade by weekday alcohol consumption
by_weekdayAlc = social_behaviors_df.groupby('weekdayAlc')
['finalGrade']

# Show table of descriptives
by_weekdayAlc.describe()
```

```
Done by Ryann Alvarez
```

	count	mean	std	min	25%	50%	75%	max
weekdayAlc								
1	276.0	10.731884	4.676502	0.0	9.0	11.0	14.00	20.0
2	75.0	9.253333	4.812970	0.0	8.0	10.0	12.00	18.0
3	26.0	10.500000	3.443835	0.0	9.0	10.0	12.75	17.0
4	9.0	9.888889	2.619372	5.0	9.0	9.0	12.00	13.0
5	9.0	10.666667	2.692582	5.0	10.0	11.0	13.00	13.0

```
print('Done by Ryann Alvarez')

# Group final grade by weekend alcohol consumption
by_weekendAlc = social_behaviors_df.groupby('weekendAlc')
['finalGrade']
```

```
# Show table of descriptives
by_weekendAlc.describe()
```

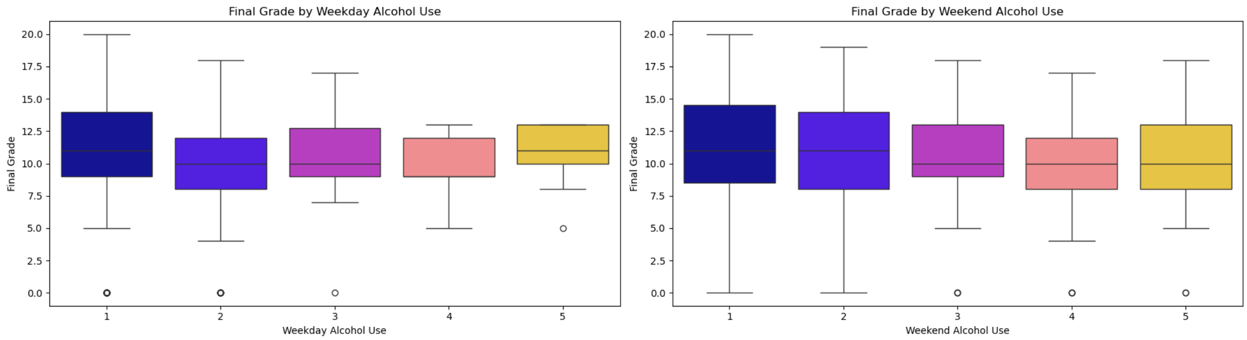
```
Done by Ryann Alvarez
```

	count	mean	std	min	25%	50%	75%	max
weekendAlc								
1	151.0	10.735099	5.133812	0.0	8.5	11.0	14.5	20.0
2	85.0	10.082353	4.950257	0.0	8.0	11.0	14.0	19.0
3	80.0	10.725000	3.700753	0.0	9.0	10.0	13.0	18.0
4	51.0	9.686275	3.619338	0.0	8.0	10.0	12.0	17.0
5	28.0	10.142857	4.125030	0.0	8.0	10.0	13.0	18.0

The table gives us lots of great descriptive information. For example, we can see that most students consume a 'very low' amount of alcohol on the weekdays and weekend. We also see an increase in alcohol consumption on the weekends. Let's create a visualization to see this better...

```
print('Done by Ryann Alvarez')

# Use matplotlib to create subplots
# Use figsize= to change the figure size
fig, axes = plt.subplots(1, 2, figsize=(18, 5))

# Create first boxplot
sns.boxplot(x='weekdayAlc', y='finalGrade', data=social_behaviors_df,
ax=axes[0], palette='gnuplot2')
axes[0].set_title("Final Grade by Weekday Alcohol Use")
axes[0].set_xlabel("Weekday Alcohol Use")
axes[0].set_ylabel("Final Grade")

# Create second boxplot
sns.boxplot(x='weekendAlc', y='finalGrade', data=social_behaviors_df,
ax=axes[1], palette='gnuplot2')
axes[1].set_title("Final Grade by Weekend Alcohol Use")
axes[1].set_xlabel("Weekend Alcohol Use")
axes[1].set_ylabel("Final Grade")

# Fix any formatting using .tight_layout()
plt.tight_layout()

# Show plot
plt.show()
```

```
Done by Ryann Alvarez
```



Using subplots allows us to see boxplots for weekday and weekend alcohol use side-by-side. Easier visual comparison!

For weekday alcohol consumption, students that consume a 'very low' amount of alcohol scored the highest on the final grade compared to their peers (on average). What is interesting is students that consume a 'very high' amount of alcohol scored the second highest on the final grade (again, on average). It is important to point out, however, the very small sample of students that consume a 'very high' alcohol of alcohol - only 9 students based off of the table of descriptives. With this being said, we should take this information cautiously.

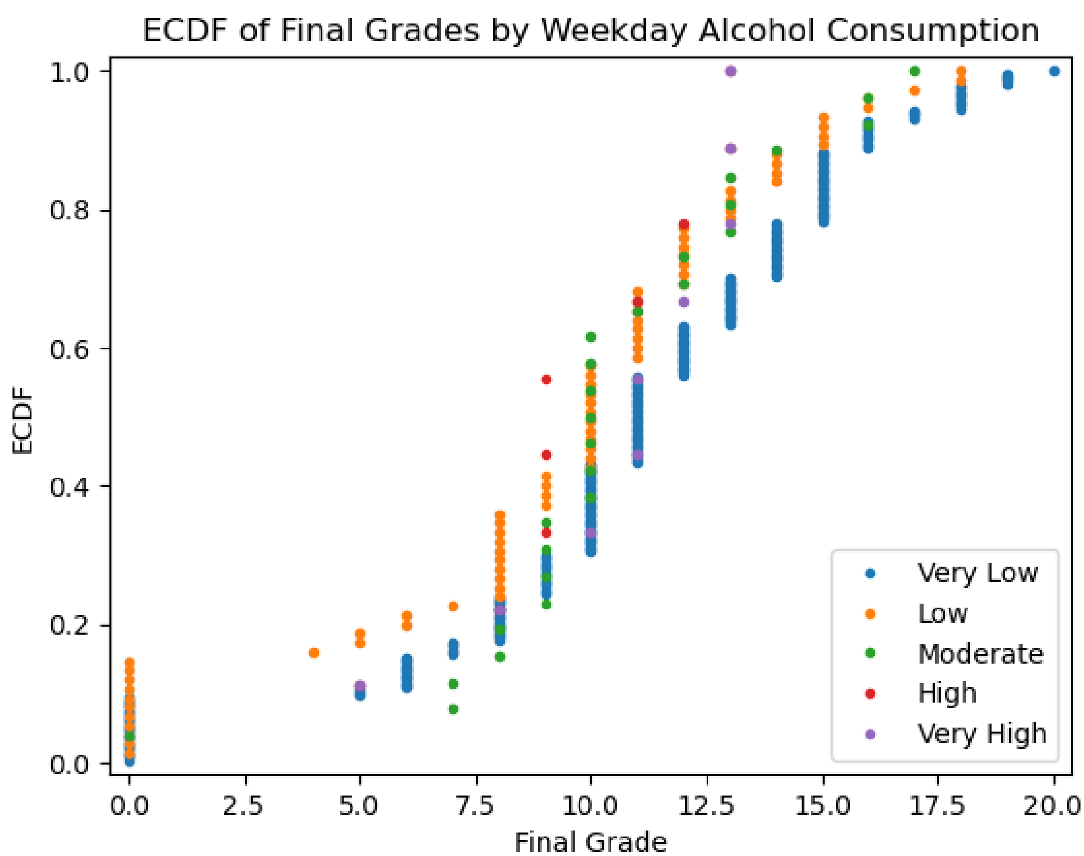
For weekend alcohol consumption, students that consume a 'very low' amount of alcohol scored the highest on the final grade compared to their peers (on average). This is followed by students that consume a 'low' amount of alcohol.

Let's see how an ECDF will represent the data for both weekday and weekend alcohol consumption.

```
print('Done by Ryann Alvarez')

# Compute arrays for ECDF
weekdayAlc_1 = social_behaviors_df[social_behaviors_df['weekdayAlc']
== 1]['finalGrade']
weekdayAlc_2 = social_behaviors_df[social_behaviors_df['weekdayAlc']
== 2]['finalGrade']
weekdayAlc_3 = social_behaviors_df[social_behaviors_df['weekdayAlc']
== 3]['finalGrade']
weekdayAlc_4 = social_behaviors_df[social_behaviors_df['weekdayAlc']
== 4]['finalGrade']
weekdayAlc_5 = social_behaviors_df[social_behaviors_df['weekdayAlc']
== 5]['finalGrade']

# Compute ECDFs
x_weekdayAlc_1, y_weekdayAlc_1 = ecdf(weekdayAlc_1)
x_weekdayAlc_2, y_weekdayAlc_2 = ecdf(weekdayAlc_2)
x_weekdayAlc_3, y_weekdayAlc_3 = ecdf(weekdayAlc_3)
x_weekdayAlc_4, y_weekdayAlc_4 = ecdf(weekdayAlc_4)
x_weekdayAlc_5, y_weekdayAlc_5 = ecdf(weekdayAlc_5)

# Plot all ECDFs on the same plot
_ = plt.plot(x_weekdayAlc_1, y_weekdayAlc_1, marker = '.', linestyle =
```

```
'none', label = "Very Low")
_ = plt.plot(x_weekdayAlc_2, y_weekdayAlc_2, marker = '.', linestyle =
'none', label = "Low")
_ = plt.plot(x_weekdayAlc_3, y_weekdayAlc_3, marker = '.', linestyle =
'none', label = "Moderate")
_ = plt.plot(x_weekdayAlc_4, y_weekdayAlc_4, marker = '.', linestyle =
'none', label = "High")
_ = plt.plot(x_weekdayAlc_5, y_weekdayAlc_5, marker = '.', linestyle =
'none', label = "Very High")

# Make nice margins
plt.margins(0.02)

# Annotate the plot add a legend locate it at the lower right
_ = plt.title("ECDF of Final Grades by Weekday Alcohol Consumption")
_ = plt.xlabel('Final Grade')
_ = plt.ylabel('ECDF')
plt.legend(loc='lower right')

# Display the plot
plt.show()
```

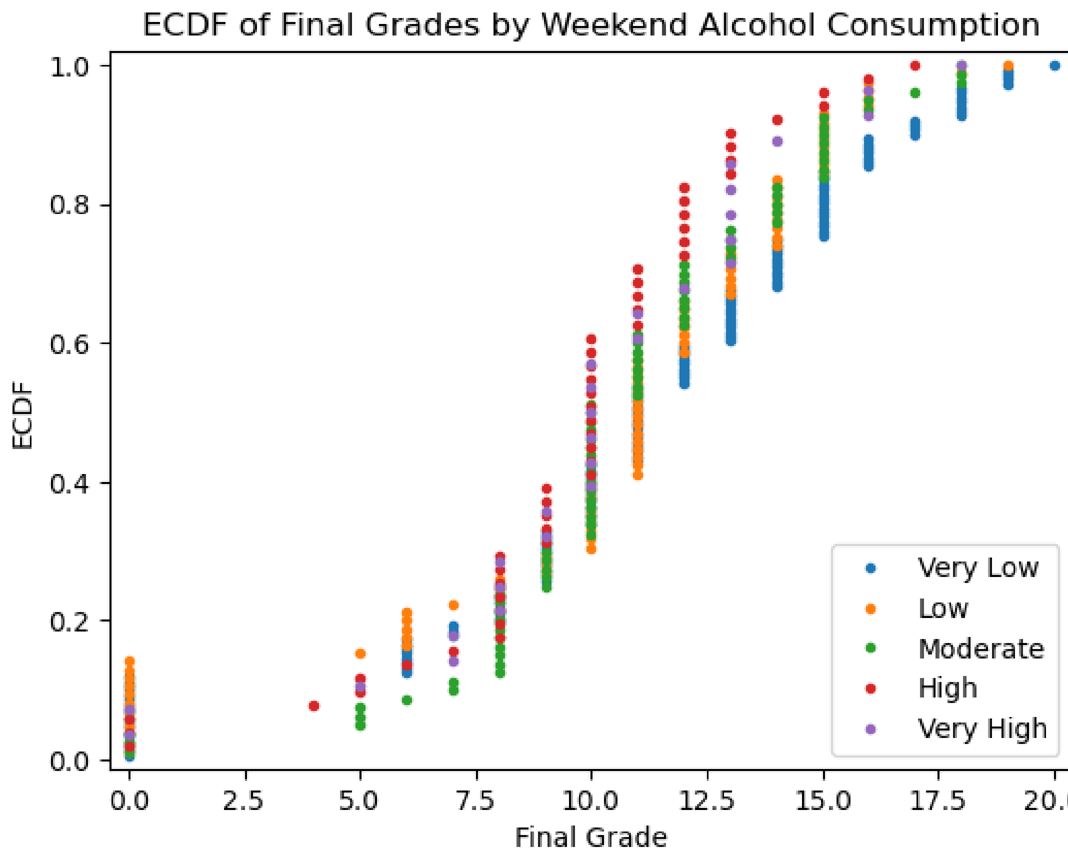
Done by Ryann Alvarez



The ECDF shows us that students who consume a 'very low' amount of alcohol have a higher probability of scoring higher on the final compared to all other groups. It is unclear when determining which group of students have the lowest probability of scoring higher on the final because the dots are overlapping with each other.

```
print('Done by Ryann Alvarez')

# Compute arrays for ECDF
weekendAlc_1 = social_behaviors_df[social_behaviors_df['weekendAlc']
== 1]['finalGrade']
weekendAlc_2 = social_behaviors_df[social_behaviors_df['weekendAlc']
== 2]['finalGrade']
weekendAlc_3 = social_behaviors_df[social_behaviors_df['weekendAlc']
== 3]['finalGrade']
weekendAlc_4 = social_behaviors_df[social_behaviors_df['weekendAlc']
== 4]['finalGrade']
weekendAlc_5 = social_behaviors_df[social_behaviors_df['weekendAlc']
== 5]['finalGrade']

# Compute ECDFs
x_weekendAlc_1, y_weekendAlc_1 = ecdf(weekendAlc_1)
x_weekendAlc_2, y_weekendAlc_2 = ecdf(weekendAlc_2)
x_weekendAlc_3, y_weekendAlc_3 = ecdf(weekendAlc_3)
x_weekendAlc_4, y_weekendAlc_4 = ecdf(weekendAlc_4)
x_weekendAlc_5, y_weekendAlc_5 = ecdf(weekendAlc_5)

# Plot all ECDFs on the same plot
_ = plt.plot(x_weekendAlc_1, y_weekendAlc_1, marker = '.', linestyle =
'none', label = "Very Low")
_ = plt.plot(x_weekendAlc_2, y_weekendAlc_2, marker = '.', linestyle =
'none', label = "Low")
_ = plt.plot(x_weekendAlc_3, y_weekendAlc_3, marker = '.', linestyle =
'none', label = "Moderate")
_ = plt.plot(x_weekendAlc_4, y_weekendAlc_4, marker = '.', linestyle =
'none', label = "High")
_ = plt.plot(x_weekendAlc_5, y_weekendAlc_5, marker = '.', linestyle =
'none', label = "Very High")

# Make nice margins
plt.margins(0.02)

# Annotate the plot add a legend locate it at the lower right
_ = plt.title("ECDF of Final Grades by Weekend Alcohol Consumption")
_ = plt.xlabel('Final Grade')
_ = plt.ylabel('ECDF')
plt.legend(loc='lower right')

# Display the plot
plt.show()
```

```
Done by Ryann Alvarez
```



The ECDF shows us that students who consume a 'very low' amount of alcohol have a higher probability of scoring higher on the final compared to all other groups. It seems as though students who consume a 'high' amount of alcohol have the lowest probability of scoring higher on the final compared to all other groups.

I'm curious to know if these categories are statistically significant. Let's conduct an ANOVA.

```
print('Done by Ryann Alvarez')

# Import libraries for one-way ANOVA
from scipy.stats import f_oneway

# Create a function that will conduct one way ANOVA test

# Function takes one argument, which is the name of a categorical column
def anova_test(col):

    # social_behaviors_df.groupby(col) splits the data into groups based on unique values
    # For each group, it pulls the finalGrade values using
```

```
group['finalGrade'].values
    groups = [group['finalGrade'].values for name, group in
social_behaviors_df.groupby(col)]

    # *groups unpacks the list so each array becomes a separate
argument
    # Returns the F-statistic and p-value
    stat, p = f_oneway(*groups)

    # Print result
    print(f"{col}: F = {stat:.2f}, p = {p:.4f}")

# H0: Mean final grade is the same across all alcohol consumption
frequencies (i.e., Very Low, Low, Moderate, High, Very High)
# H1: At least one group has a different mean final grade

# Conduct ANOVA
anova_test('weekdayAlc_recode')
anova_test('weekendAlc_recode')
```

```
Done by Ryann Alvarez
weekdayAlc_recode: F = 1.58, p = 0.1779
weekendAlc_recode: F = 0.73, p = 0.5698
```

For weekday alcohol consumption, our p-value is p = 0.1779. It is not less than p = 0.05, so we fail to reject the null hypothesis. This suggests there is not enough evidence to say that final grades differ depending on student's weekday alcohol consumption.

For weekend alcohol consumption, the results are similar. The p-value is p = 0.5698, so it is not less than p = 0.05, so we fail to reject the null hypothesis. This suggests there is not enough evidence to say that final grades differ depending on student's weekend alcohol consumption.

Overall Correlations

```
print('Done by Ryann Alvarez')

# Import libraries for Pearson's Correlation test
from scipy.stats import pearsonr

# Tests the relationship between two variables (i.e., final grade
and ...)

# We can do this more efficient using a for loop
for var in ['goOutFreq', 'weekdayAlc', 'weekendAlc']:
    # Conduct Pearson correlation
    stat, p = pearsonr(social_behaviors_df[var],
social_behaviors_df['finalGrade'])
    # Print result
    print(f"{var} vs finalGrade: stat = {stat:.3f}, p = {p:.3f}")
```

```
Done by Ryann Alvarez
goOutFreq vs finalGrade: stat = -0.133, p = 0.008
weekdayAlc vs finalGrade: stat = -0.055, p = 0.278
weekendAlc vs finalGrade: stat = -0.052, p = 0.303
```

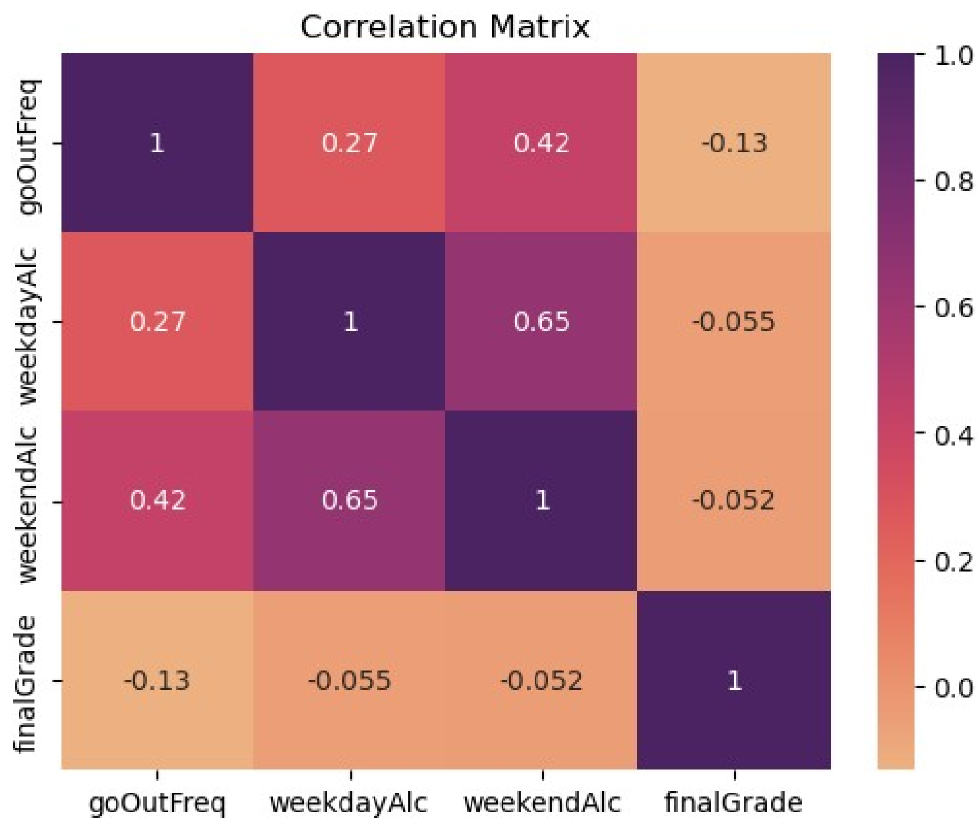
We can see that going out frequency and final grade have a negative, weak, linear relationship. This suggests an inverse relationship between the two variables - as one variable increases, the other tends to decrease, and vice versa. For example, as going out frequency increases, final grade decreases. This relationship has a p-value of = 0.008, which is less than p = 0.05, so we can reject the null hypothesis and say there is a significant relationship between going out frequency and final grade.

For weekday alcohol consumption and final grade and weekend alcohol consumption and final grade, the p-values are not less than 0.05, and so their relationships are not significant.

Let's visualize everything using a heatmap.

```python
print('Done by Ryann Alvarez')

# Visualize correlations using a correlation matrix
corr_matrix = social_behaviors_df[['goOutFreq', 'weekdayAlc',
'weekendAlc', 'finalGrade']].corr(method='pearson')

# Create heatmap using .heatmap
sns.heatmap(corr_matrix, annot=True, cmap='flare')

# Plot title
plt.title("Correlation Matrix")

# Show plot
plt.show()
```

```
Done by Ryann Alvarez
```



Something new we can gather from the heatmap is weekday and weekend alcohol are strongly correlated and going out frequency and weekend alcohol consumption are moderately correlated.

Question #8

Are students who have internet access or want higher education performing better academically?

Internet Access

First, let's establish how many students have internet access at home.

```
print('Done by Ryann Alvarez')

# Use .value_counts() to show how many students do and do not have internet access at home
student_df['hasInternet'].value_counts()
```

```
Done by Ryann Alvarez

hasInternet
yes    329
no      66
Name: count, dtype: int64
```

We can see that the majority of students have internet access at home.

How are these students performing academically?

```
print('Done by Ryann Alvarez')

# Group final grade by internet access
by_hasInternet = student_df.groupby('hasInternet')['finalGrade']

# Show table of descriptives
by_hasInternet.describe()
```

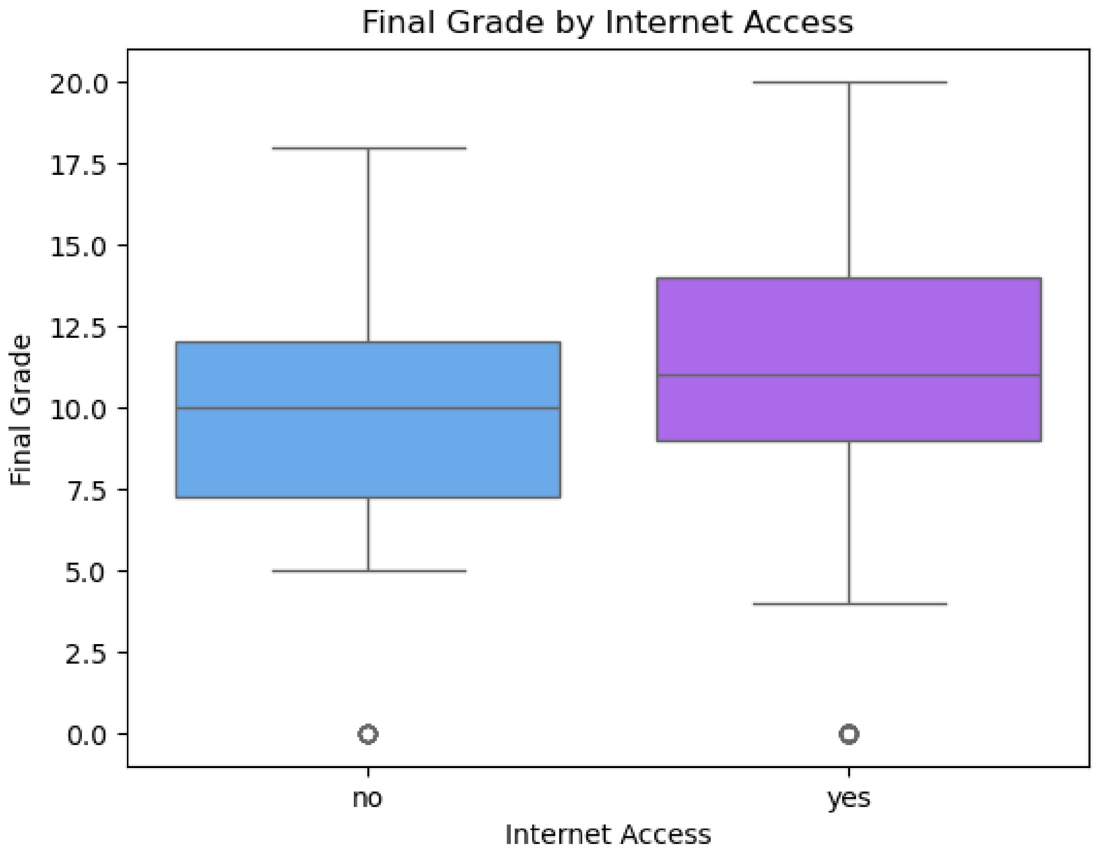
```
Done by Ryann Alvarez
```

	count	mean	std	min	25%	50%	75%	max
hasInternet								
no	66.0	9.409091	4.485797	0.0	7.25	10.0	12.0	18.0
yes	329.0	10.617021	4.580494	0.0	9.00	11.0	14.0	20.0

Based off of the table of descriptive statistics, we can see that students who have Internet access perform better on the final compared to students who do not have Internet access. Let's create a visualization for the descriptive statistics.

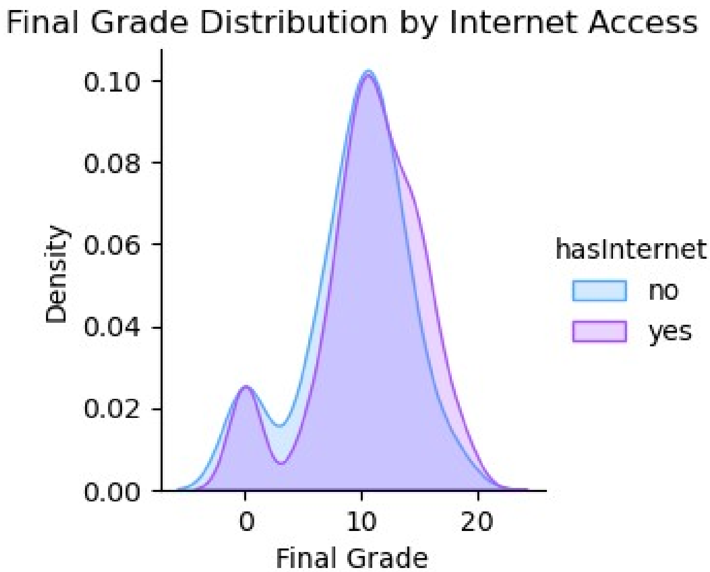
```
print('Done by Ryann Alvarez')

# Create boxplot to look closer at going out frequency
sns.boxplot(x='hasInternet', y='finalGrade', data=student_df,
palette='cool')

# Plot title
plt.title("Final Grade by Internet Access")

# Plot axes labels
plt.xlabel("Internet Access")
plt.ylabel("Final Grade")

# Show plot
plt.show()
```

```
Done by Ryann Alvarez
```



The boxplot shows that students who have internet access perform better as a whole compared to students who do not have internet access.

```
print('Done by Ryann Alvarez')

# Use FaceGrid to show the distributions of final grade
# Add hue='hasInternet' to distinguish by the number of hours studied
per week
g = sns.FacetGrid(data=student_df, hue='hasInternet', palette='cool')
g.map(sns.kdeplot, 'finalGrade', fill=True).add_legend()

# Plot title
plt.title("Final Grade Distribution by Internet Access")

# Plot axes
plt.xlabel("Final Grade")
plt.ylabel("Density")

# Show plot
plt.show()

Done by Ryann Alvarez
```



The KDE plot shows the distribution of final grades by internet access. We can see that both groups have similar peaks, but those that have internet access have fewer low scores and more high scores. This could be a reason as to why their average score is higher than students who do not have internet access.

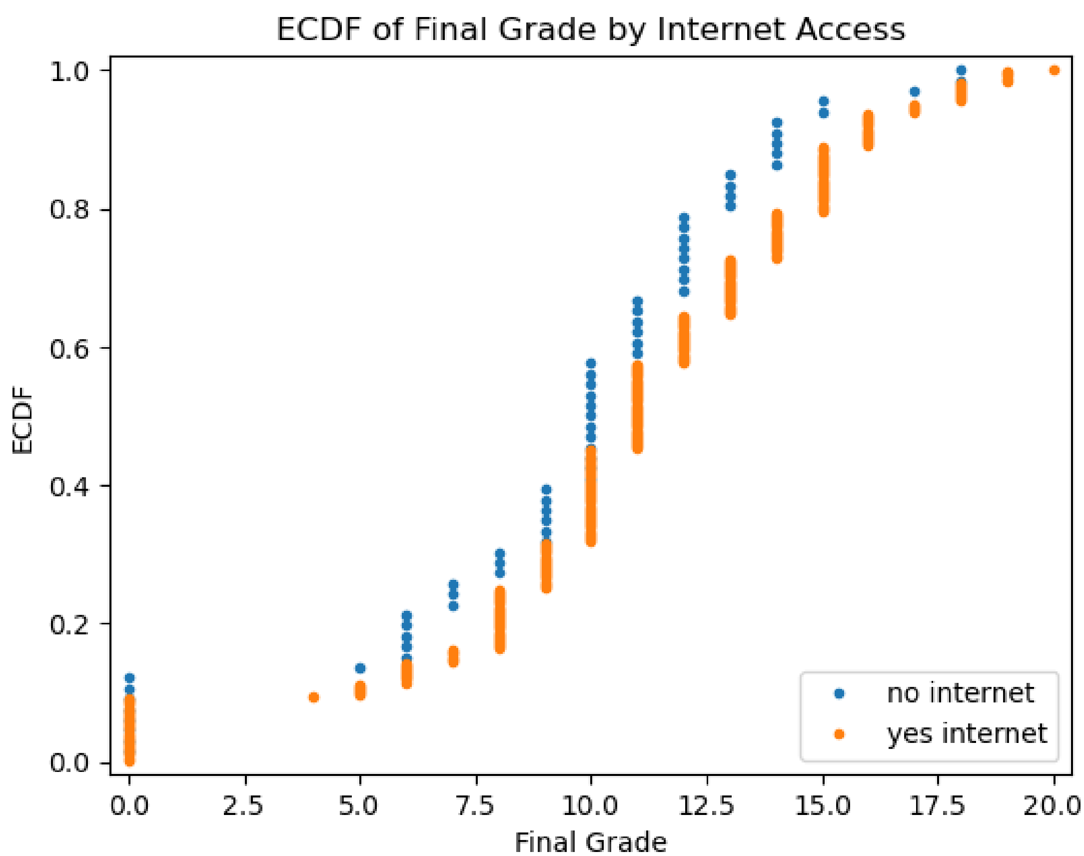
What does an ECDF reveal?

```
print('Done by Ryann Alvarez')

def ecdf(data):
    """Compute ECDF for a one-dimensional array of measurements."""

    # Number of data points: n
    n = len(data)

    # x-data for the ECDF: x
    x = np.sort(data)

    # y-data for the ECDF: y
    y = np.arange(1, n+1) / n

    return x, y
```

```
Done by Ryann Alvarez
```

```
print('Done by Ryann Alvarez')

# Another way to do an ECDF is to use a for loop for this to capture
every individual student
for group, df in student_df.groupby('hasInternet'):
    # Compute x, y by calling the ECDF function
    x, y = ecdf(df['finalGrade'])
```

```
    # Plot the ECDF
    plt.plot(x, y, marker='.', linestyle='none', label = f"{group}
internet")

# Adjust margins
plt.margins(0.02)

# Annotate the plot
plt.title("ECDF of Final Grade by Internet Access")
plt.xlabel("Final Grade")
plt.ylabel("ECDF")
plt.legend(loc='lower right')

# Show plot
plt.show()

Done by Ryann Alvarez
```



The ECDF shows us that students who have internet access have a higher probability of scoring higher on the final compared to students who do not have internet access.

Let's see if this difference is statistically significant using a t-test.

```python
print('Done by Ryann Alvarez')

# Import required libraries for a t-test
from scipy import stats
from scipy.stats import ttest_ind

# Identify male and female grades
yes_Internet = student_df[student_df['hasInternet'] == 'yes']
['finalGrade']
no_Internet = student_df[student_df['hasInternet'] == 'no']
['finalGrade']

# H0: There is no significant difference between the means
# H1: There is a significant difference between the means

# Perform independent samples t-test
# Returns t-test statistic and p-value
stat, p = stats.ttest_ind(yes_Internet, no_Internet)

# Print result
print(f"Internet access t-test: t = {stat:.3f}, p = {p:.4f}")
```

```
Done by Ryann Alvarez
Internet access t-test: t = 1.962, p = 0.0505
```

Our p-value of = 0.0505 is not less than p = 0.05, so we fail to reject the null hypothesis and say there is no significant difference between the the means of having internet access or not. Since the t-test statistic is positive, that means the final mean grade of the first group (i.e., has internet access) is larger than the final mean grade of the second group (i.e., does not have internet access).

Higher Education

How many students want to pursue higher education? How many do not?

```python
print('Done by Ryann Alvarez')

# Use .value_counts() to show how many students do and do not want to
pursue higher education
student_df['wantsHigherEdu'].value_counts()
```

```
Done by Ryann Alvarez

wantsHigherEdu
yes    375
no      20
Name: count, dtype: int64
```

Awesome! The majority of students want to puruse higher education!

Now, how are these students performing academically?

```
print('Done by Ryann Alvarez')

# Group final grade by aspirations of higher education
by_wantsHigherEdu = student_df.groupby('wantsHigherEdu')['finalGrade']

# Show table of descriptives
by_wantsHigherEdu.describe()
```

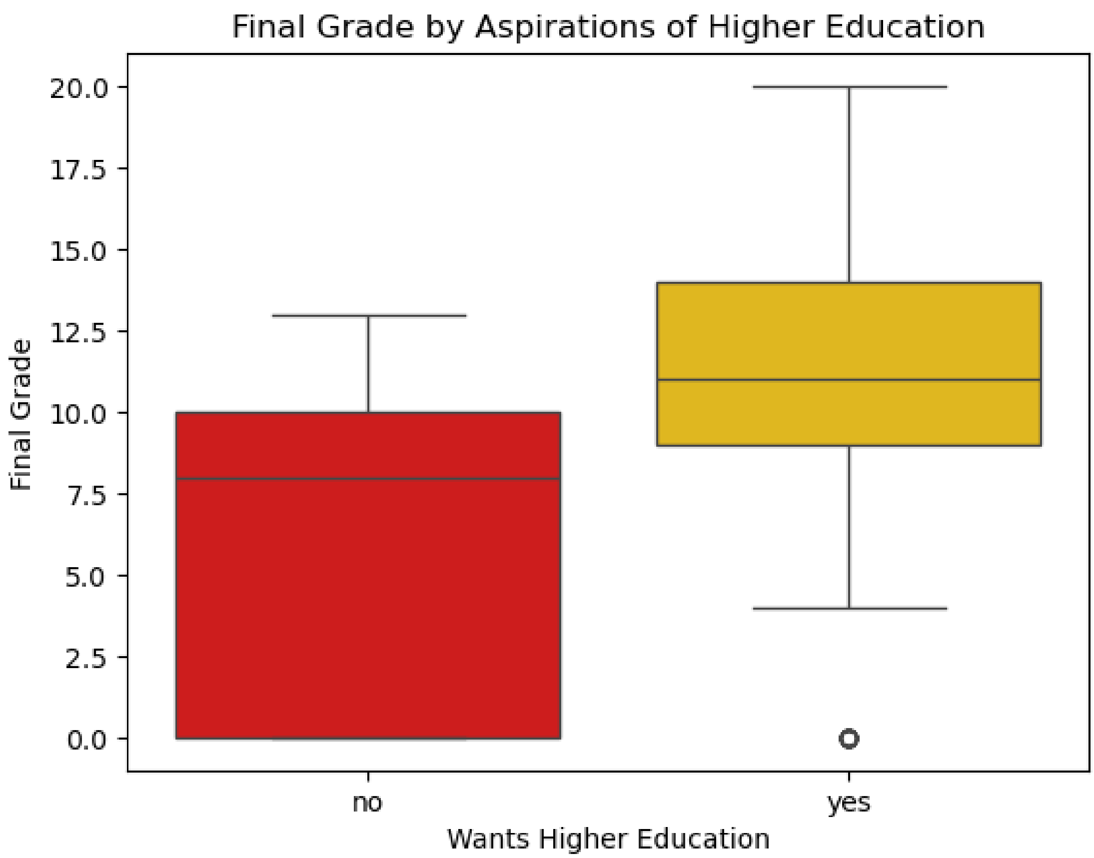
```
Done by Ryann Alvarez
```

	count	mean	std	min	25%	50%	75%	max
wantsHigherEdu								
no	20.0	6.800	4.829732	0.0	0.0	8.0	10.0	13.0
yes	375.0	10.608	4.493422	0.0	9.0	11.0	14.0	20.0

Based off of the table of descriptive statistics, we can see that students who want higher education perform noticeably better than students who do not want higher education. One thing to note is the sample sizes of these groups are vary different from one another, so this may skew our data and results.

```
print('Done by Ryann Alvarez')

# Create boxplot to look closer at going out frequency
sns.boxplot(x='wantsHigherEdu', y='finalGrade', data=student_df,
palette='hot')

# Plot title
plt.title("Final Grade by Aspirations of Higher Education")

# Plot axes labels
plt.xlabel("Wants Higher Education")
plt.ylabel("Final Grade")

# Show plot
plt.show()
```

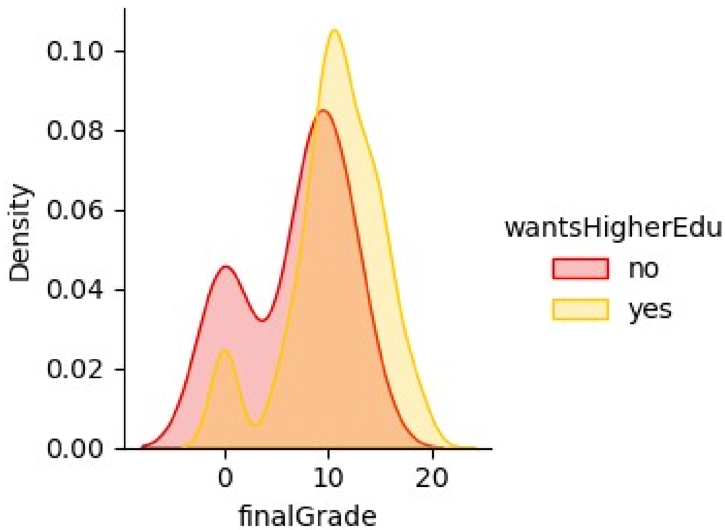
```
Done by Ryann Alvarez
```



The boxplot shows that students who do not want higher education perform very poorly compared to students who do want higher education. Such students interquartile range remains on the low end, with their max score being 13. Compared to those who want higher education, such students interquartile range is smaller (i.e., less variability), the max score is a 20, and there are few outliers on the low end.

```
print('Done by Ryann Alvarez')

# Use FaceGrid to show the distributions of final grade
# Add hue='hasInternet' to distinguish by the number of hours studied
per week
g = sns.FacetGrid(data=student_df, hue='wantsHigherEdu',
palette='hot')
g.map(sns.kdeplot, 'finalGrade', fill=True).add_legend()

# Show plot
plt.show()
```

```
Done by Ryann Alvarez
```

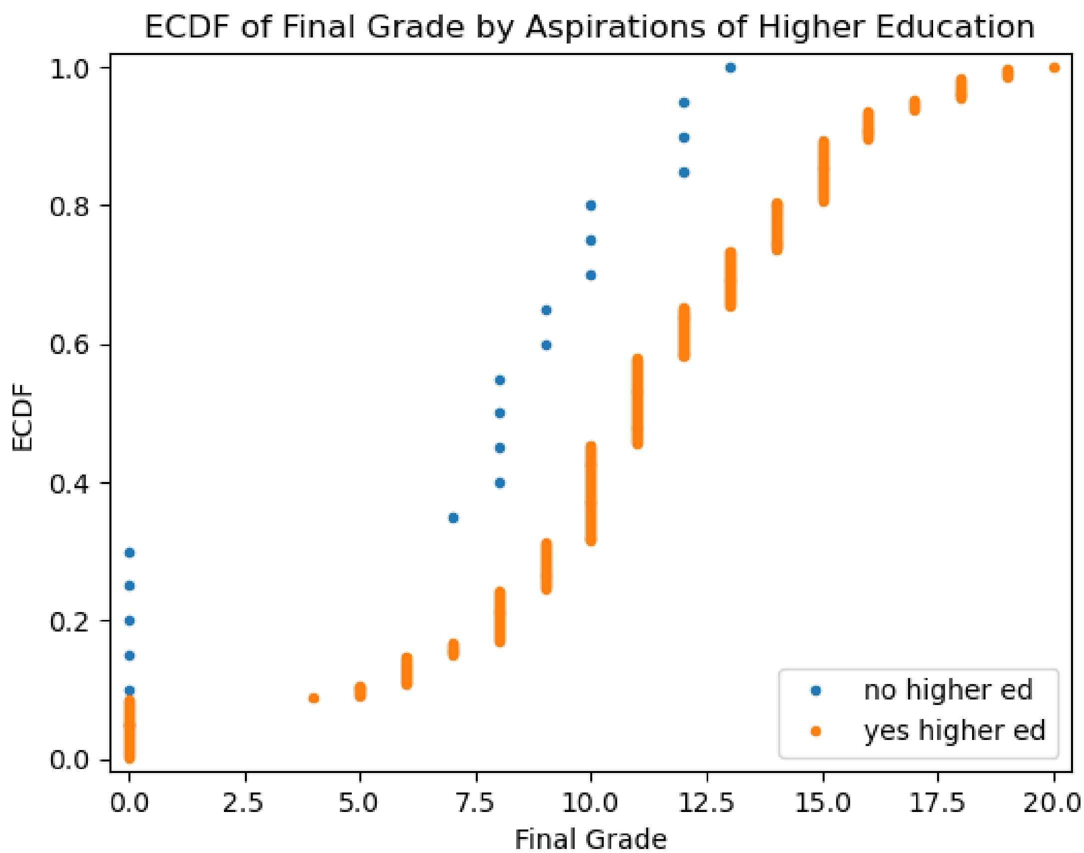


The KDE plot shows the distribution of final grades by wanting higher education. We can see that for students that want higher education, they have a higher peak at a higher score. They also have less scores on the low end and more scores on the high end compared to students who do not want higher education.

What does an ECDF reveal?

```
print('Done by Ryann Alvarez')

# Another way to do an ECDF is to use a for loop for this to capture every individual student
for group, df in student_df.groupby('wantsHigherEdu'):
    # Compute x, y by calling the ECDF function
    x, y = ecdf(df['finalGrade'])
    # Plot the ECDF
    plt.plot(x, y, marker='.', linestyle='none', label=f"{group} higher ed")

# Adjust margins
plt.margins(0.02)

# Annotate the plot
plt.title("ECDF of Final Grade by Aspirations of Higher Education")
plt.xlabel("Final Grade")
plt.ylabel("ECDF")
plt.legend(loc='lower right')

# Show plot
plt.show()
```

```
Done by Ryann Alvarez
```



The ECDF shows us that students who want higher education have a higher probability of scoring higher on the final compared to students who do not want higher education.

Let's see if this difference is statistically significant using a t-test.

```
print('Done by Ryann Alvarez')

# Import required libraries for a t-test
from scipy import stats
from scipy.stats import ttest_ind

# Identify male and female grades
yes_higherEdu = student_df[student_df['wantsHigherEdu'] == 'yes']
['finalGrade']
no_higherEdu = student_df[student_df['wantsHigherEdu'] == 'no']
['finalGrade']

# H0: There is no significant difference between the means
# H1: There is a significant difference between the means

# Perform independent samples t-test
# Returns t-test statistic and p-value
stat, p = stats.ttest_ind(yes_higherEdu, no_higherEdu)
```

```
# Print result
print(f"Wants higher education t-test: t = {stat:.3f}, p = {p:.4f}")

Done by Ryann Alvarez
Wants higher education t-test: t = 3.679, p = 0.0003
```

Our p-value of = 0.0003 is less than p = 0.05, so we can reject the null hypothesis. This suggests strong evidence that there is a significant difference between the the final grade means of wanting to pursue higher education or not. Since the t-test statistic is positive, that means the mean of the first group (i.e., wants higher education) is larger than the mean of the second group (i.e., does not want higher education).

Question #9

What are the most important predictors of student success?

I will start by putting all the relevant variables in a separate DataFrame. These are the variables I think will be important predictors of student success.

```
print('Done by Ryann Alvarez')

# Put all the relevant variables in a separate DataFrame
student_success_df = student_df[['studyTime', 'numAbsences',
'schoolSupport', 'familySupport',
                                  'paidClasses', 'weekdayAlc',
'weekendAlc', 'wantsHigherEdu',
                                  'motherEdu', 'fatherEdu',
'finalGrade']]

# Preview of DataFrame
student_success_df.head()

Done by Ryann Alvarez

   studyTime  numAbsences schoolSupport familySupport paidClasses
weekdayAlc  \
0          2            6           yes            no          no
1
1          2            4            no           yes          no
1
2          2           10           yes            no         yes
2
3          3            2            no           yes         yes
1
4          2            4            no           yes         yes
1

   weekendAlc wantsHigherEdu  motherEdu  fatherEdu  finalGrade
0           1            yes          4          4           6
1           1            yes          1          1           6
```

2	3	yes	1	1	10
3	1	yes	4	2	15
4	2	yes	3	3	10

To determine if the variables are significant predictors, I will run correlations for the numeric predictors and t-tests for binary predictors.

Correlations

For numeric predictors

```
# Import libraries for Pearson's Correlation test
from scipy.stats import pearsonr

# Create a list for the numeric variables
numeric_vars = ['studyTime', 'numAbsences', 'weekdayAlc',
'weekendAlc', 'motherEdu', 'fatherEdu']

# Conduct correlations using a for loop
# This way it is more efficient instead of doing individual
correlations
for var in numeric_vars:
    stat, p = pearsonr(student_success_df[var],
student_success_df['finalGrade'])
    print(f"{var} vs. finalGrade: stat = {stat:.3f}, p-value =
{p:.3f}")
```

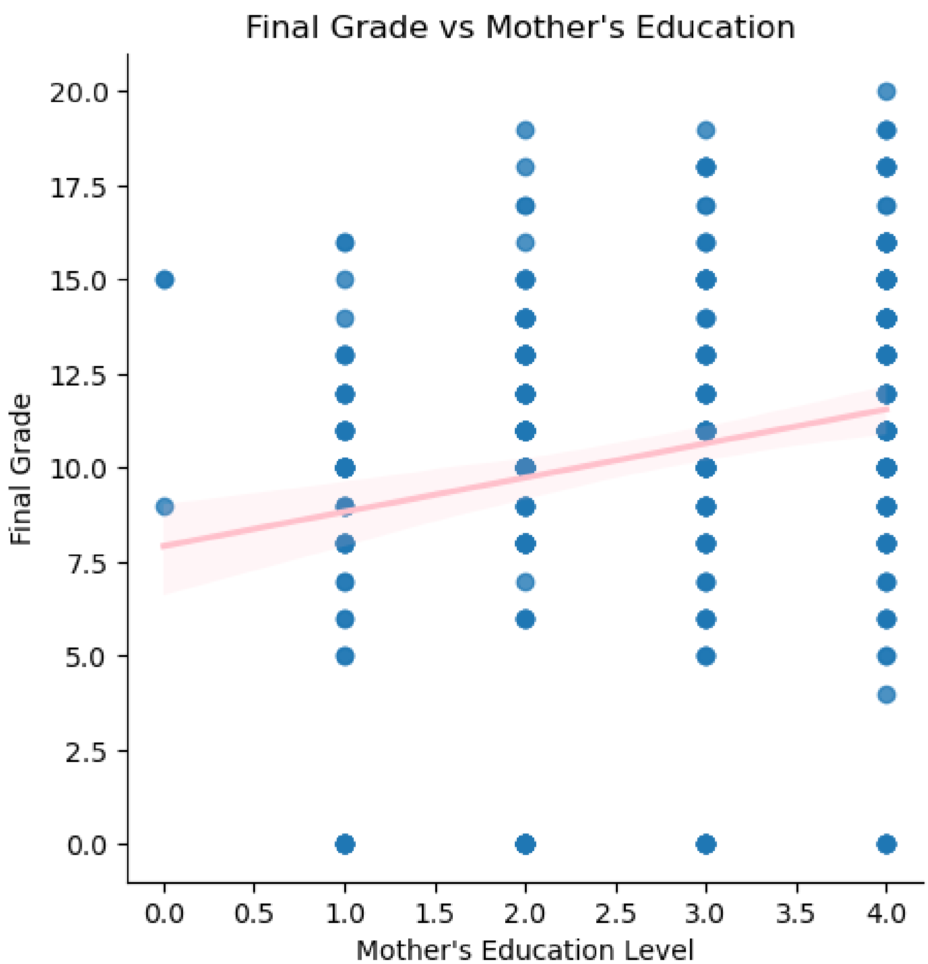
```
studyTime vs. finalGrade: stat = 0.098, p-value = 0.052
numAbsences vs. finalGrade: stat = 0.034, p-value = 0.497
weekdayAlc vs. finalGrade: stat = -0.055, p-value = 0.278
weekendAlc vs. finalGrade: stat = -0.052, p-value = 0.303
motherEdu vs. finalGrade: stat = 0.217, p-value = 0.000
fatherEdu vs. finalGrade: stat = 0.152, p-value = 0.002
```

For the numeric predictors, the significant relationships are between mother's education level and final grade ($r = 0.217$, $p < 0.001$), and father's education level and final grade ($r = 0.152$, $p = 0.002$).

Let's see a lmplot for the correlations that are significant.

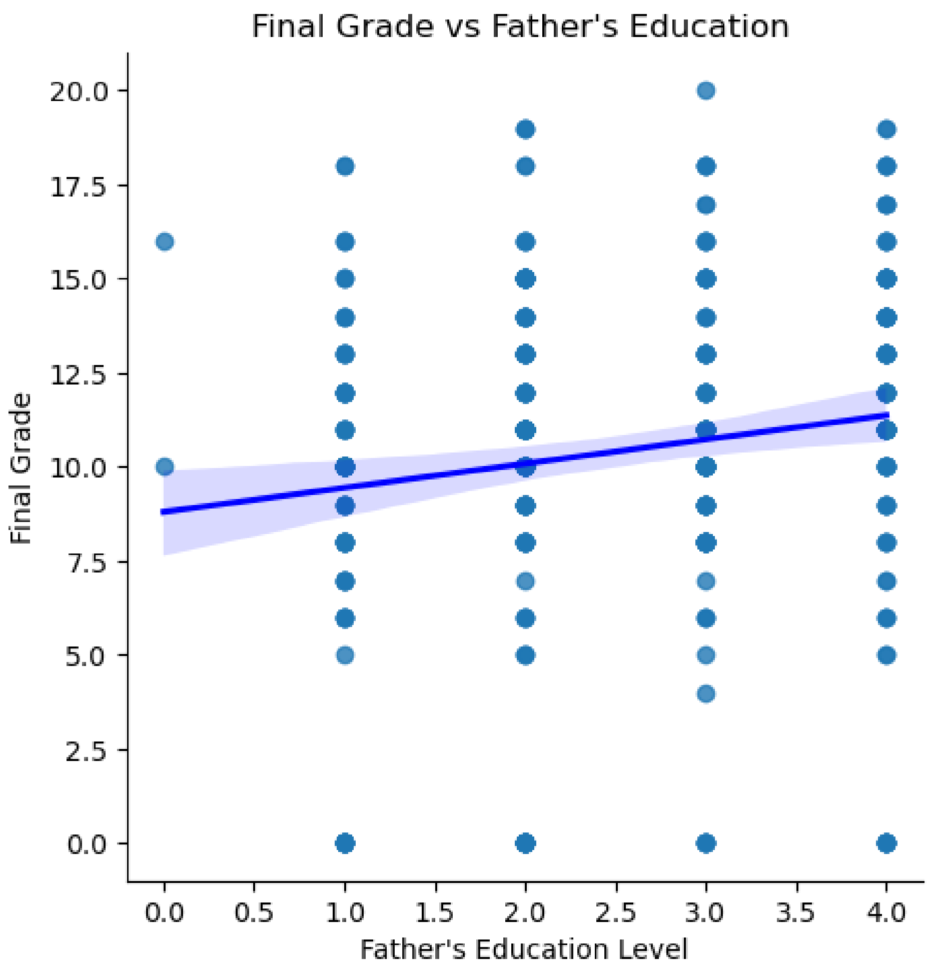
```
print('Done by Ryann Alvarez')

# Use lmplot()
sns.lmplot(x='motherEdu', y='finalGrade', data=student_success_df,
line_kws={'color': 'pink'})

# Plot title
plt.title("Final Grade vs Mother's Education ")

# Plot axes
```

```
plt.xlabel("Mother's Education Level")
plt.ylabel("Final Grade")

# Show plot
plt.show()

Done by Ryann Alvarez
```




```
print('Done by Ryann Alvarez')

# Use lmplot()
sns.lmplot(x='fatherEdu', y='finalGrade', data=student_success_df,
line_kws={'color': 'blue'})

# Plot title
plt.title("Final Grade vs Father's Education ")

# Plot axes
plt.xlabel("Father's Education Level")
plt.ylabel("Final Grade")
```

```
# Show plot
plt.show()

Done by Ryann Alvarez
```



Both of these graphs show a weak, positive, linear relationship between the two variables.

T-Test

For binary predictors

```
# Import libraries for t-test
from scipy import stats
from scipy.stats import ttest_ind

# Create a list for the binary (yes/no) variables
binary_vars = ['schoolSupport', 'familySupport', 'paidClasses',
'wantsHigherEdu']
```

```
# H0: There is no significant difference between the means
# H1: There is a significant difference between the means

# Use a for loop to conduct multiple t-tests at once, rather than one
at a time

# Loop through each variable
for var in binary_vars:
    # Create two groups based on performance level
    yes_group = student_df[student_df[var] == 'yes']['finalGrade']
    no_group = student_df[student_df[var] == 'no']['finalGrade']

    # Perform independent samples t-test
    stat, p = stats.ttest_ind(yes_group, no_group)

    # Print result
    print(f"{var} t-test: t = {stat:.3f}, p = {p:.4f}")
```

```
schoolSupport t-test: t = -1.647, p = 0.1004
familySupport t-test: t = -0.777, p = 0.4377
paidClasses t-test: t = 2.033, p = 0.0428
wantsHigherEdu t-test: t = 3.679, p = 0.0003
```

For the binary predictors, the significant differences are between receiving paid classes and final grade ($t = 2.033$, $p = 0.0428$), and wanting higher education and final grade ($t = 3.679$, $p = 0.0003$).

Question #10

What are the differences between high-performing and low-performing students in terms of study time, support, social habits, and family background?

To begin, I will identify the important variables:

- Study habits: studyTime
- Support: schoolSupport, familySupport
- Social habits: weekdayAlc, weekendAlc, goOutFreq
- Family background: motherEdu, fatherEdu, parentStatus

Next, I will define performance groups to separate students into 'high-performing' and 'low-performing.'

```
print('Done by Ryann Alvarez')

# Use median as the split
median_grade = student_df['finalGrade'].median()

# Display result
print(median_grade)
```

```
Done by Ryann Alvarez
11.0
```

```
print('Done by Ryann Alvarez')

# Define a function that creates a performance group
def performance_group(grade):
    """
    Separates students into high- and low-performing groups based on
their final grade
    """
    # Return 'High' is grade is greater than or equal to 11
    if grade >= 11:
        return 'High'
    # Return 'Low' if grade is less than 11
    else:
        return 'Low'
```

```
Done by Ryann Alvarez
```

```
print('Done by Ryann Alvarez')

# Apply the performance_group function to the DataFrame
student_df['performance_group'] =
student_df.finalGrade.apply(performance_group)

# Display result along with final grade to make sure it recoded
properly
student_df[['finalGrade', 'performance_group']].head()
```

```
Done by Ryann Alvarez
```

	finalGrade	performance_group
0	6	Low
1	6	Low
2	10	Low
3	15	High
4	10	Low

For the numeric variables, I will group by performance group and get overall descriptive statistics.

```
print('Done by Ryann Alvarez')

# Create a list for the numeric variables
numeric_vars = ['studyTime', 'weekdayAlc', 'weekendAlc', 'goOutFreq',
'motherEdu', 'fatherEdu']

# Use a for loop to get many descriptives at once, rather than one at
a time
```

```
for var in numeric_vars:

    print(f"\nDescriptives for {var} grouped by performance group:")

    # Group by low- and high-performance group
    # Use .describe() to get descriptives for numeric variables
    print(student_df.groupby('performance_group')[var].describe())
```

```
Done by Ryann Alvarez

Descriptives for studyTime grouped by performance group:
                   count      mean       std  min  25%  50%  75%  max
performance_group
High               209.0  2.114833  0.869472  1.0  2.0  2.0  3.0  4.0
Low                186.0  1.946237  0.796826  1.0  1.0  2.0  2.0  4.0

Descriptives for weekdayAlc grouped by performance group:
                   count      mean       std  min  25%  50%  75%  max
performance_group
High               209.0  1.421053  0.895897  1.0  1.0  1.0  1.0  5.0
Low                186.0  1.548387  0.882455  1.0  1.0  1.0  2.0  5.0

Descriptives for weekendAlc grouped by performance group:
                   count      mean       std  min  25%  50%   75%  max
performance_group
High               209.0  2.133971  1.221303  1.0  1.0  2.0  3.00  5.0
Low                186.0  2.467742  1.340243  1.0  1.0  2.0  3.75  5.0

Descriptives for goOutFreq grouped by performance group:
                   count      mean       std  min  25%  50%  75%  max
performance_group
High               209.0  2.947368  1.061614  1.0  2.0  3.0  4.0  5.0
Low                186.0  3.290323  1.144488  1.0  2.0  3.0  4.0  5.0

Descriptives for motherEdu grouped by performance group:
                   count     mean       std  min  25%  50%   75%  max
performance_group
High               209.0  2.91866  1.068827  0.0  2.0  3.0  4.00  4.0
Low                186.0  2.55914  1.095074  0.0  2.0  3.0  3.75  4.0

Descriptives for fatherEdu grouped by performance group:
                   count      mean       std  min  25%  50%  75%  max
performance_group
High               209.0  2.698565  1.078660  0.0  2.0  3.0  4.0  4.0
Low                186.0  2.322581  1.067063  0.0  1.0  2.0  3.0  4.0
```

This is a lot of information all at once, so let's narrow this down into just looking at the mean. Let's use a pivot_table for this.

```python
print('Dony by Ryann Alvarez')

# Use a for loop to get many descriptives at once, rather than one at
a time
for var in numeric_vars:

    print(f"\nDescriptives for {var} grouped by performance group:")

    # Group by low- and high-performance group
    # Use .pivot_table() to show mean for numeric variables
    print(student_df.pivot_table(values={var},
index='performance_group'))
```

```
Dony by Ryann Alvarez

Descriptives for studyTime grouped by performance group:
                   studyTime
performance_group
High                2.114833
Low                 1.946237

Descriptives for weekdayAlc grouped by performance group:
                   weekdayAlc
performance_group
High                 1.421053
Low                  1.548387

Descriptives for weekendAlc grouped by performance group:
                   weekendAlc
performance_group
High                 2.133971
Low                  2.467742

Descriptives for goOutFreq grouped by performance group:
                   goOutFreq
performance_group
High                2.947368
Low                 3.290323

Descriptives for motherEdu grouped by performance group:
                   motherEdu
performance_group
High                 2.91866
Low                  2.55914

Descriptives for fatherEdu grouped by performance group:
                   fatherEdu
performance_group
High                2.698565
Low                 2.322581
```

We can see important differences between each group, such as the high-performing group studies more, consumes less alcohol (on both the weekdays and weekends), goes out less frequently, and both their mother and father have higher education level compared to the low-performing group.

Are any of these differences statistically significant? Let's use a t-test.

```
print('Done by Ryann Alvarez')

# Import required libraries for a t-test
from scipy import stats
from scipy.stats import ttest_ind

# H0: There is no significant difference between the means
# H1: There is a significant difference between the means

# Use a for loop to conduct multiple t-tests at once, rather than one
at a time

# Loop through each variable
for var in numeric_vars:
    # Create two groups based on performance level
    group_high = student_df[student_df['performance_group'] == 'High']
[var]
    group_low = student_df[student_df['performance_group'] == 'Low']
[var]

    # Perform independent samples t-test
    stat, p = stats.ttest_ind(group_high, group_low)

    # Print result
    print(f"{var} t-test: t = {stat:.3f}, p = {p:.4f}")
```

```
Done by Ryann Alvarez
studyTime t-test: t = 2.001, p = 0.0461
weekdayAlc t-test: t = -1.420, p = 0.1564
weekendAlc t-test: t = -2.590, p = 0.0100
goOutFreq t-test: t = -3.089, p = 0.0022
motherEdu t-test: t = 3.299, p = 0.0011
fatherEdu t-test: t = 3.475, p = 0.0006
```

For the numeric predictors, the significant differences are between the high- and low-performing groups are for study time (t = 2.001, p = 0.0461), weekend alcohol consumption (t = -2.590, p = 0.0100), going out frequency (t = -3.089, p = 0.0022), mother's education level (t = 3.299, p = 0.0011), and father's education level (t = 3.475, p = 0.0006).

Now, for the categorical variables, I will use value counts to get a exact number.

```
print('Done by Ryann Alvarez')
```

```
# Create a list for the categorical variables
categorical_vars = ['schoolSupport', 'familySupport', 'parentStatus']

# Use a for loop to get many value counts at once, rather than one at
a time
for var in categorical_vars:

    print(f"\nFrequency of {var} by performance group:")

    # Group by low- and high-performance group
    # Use .value_counts() to get a distribution for categorical
variables
    print(pd.crosstab(student_df['performance_group'],
student_df[var]))
```

```
Done by Ryann Alvarez

Frequency of schoolSupport by performance group:
schoolSupport       no  yes
performance_group
High               191   18
Low                153   33

Frequency of familySupport by performance group:
familySupport      no  yes
performance_group
High               84  125
Low                69  117

Frequency of parentStatus by performance group:
parentStatus       Apart  Together
performance_group
High                  25       184
Low                   16       170
```

To determine if any of these differences are important, let's conduct a chi-square test to assess the difference between observed and expected values.

```
print('Done by Ryann Alvarez')

# Import required libraries for a chi-square test
from scipy.stats import chi2_contingency

# H0: There is no association between the variables (i.e., performance
group and ...)
# H1: There is an association between the varaibles


# Create a list for the categorical variables
categorical_vars = ['schoolSupport', 'familySupport', 'parentStatus']
```

```
# Use a for loop to get many value counts at once, rather than one at
a time
for var in categorical_vars:

    # Create a crosstab for observed frequencies for each variable
    contingency_table = pd.crosstab(student_df['performance_group'],
student_df[var])

    # Returns the calculated statistic, p-value, degrees of freedom,
and table of expected frequncies
    stat, p, dof, expected = chi2_contingency(contingency_table)

    # Print results
    print(f"Chi-Square Test ({var} and performance_group): stat =
{stat:.2f}, p = {p:.4f}, dof = {dof}")
```

```
Done by Ryann Alvarez
Chi-Square Test (schoolSupport and performance_group): stat = 6.51, p
= 0.0108, dof = 1
Chi-Square Test (familySupport and performance_group): stat = 0.28, p
= 0.5984, dof = 1
Chi-Square Test (parentStatus and performance_group): stat = 0.86, p =
0.3537, dof = 1
```

For the categorical predictors, the significant differences in the expected and observed frequencies for school support and performance group (stat = 6.51, p = 0.0108). We can reject the null hypothesis, suggesting there is strong evidence that there is a statistically significant association between school support and performance.

Summary

1. What is the demographic breakdown of the students?

- More students attend Gabriel Pereira (i.e., 349 students) compared to Mousinho da Silveira (i.e., 46 students).
- There are slightly more female students (i.e., 208 students) compared to male students (i.e., 187 students). More specifically, slightly more females attend both Gabriel Pereira and Mousinho da Silveira.
- The average age of students is about 17 years old. The minimum and maximun age is 15 and 22, respectively. A normal distribution test revealed strong evidence that the student's ages are not normally distributed.
- More students live in an urban area (i.e., 307 students) compared to a rural area (i.e., 88 students). More specifically, most students enrolled at Gabriel Pereira live in an urban area, but there is a more even split between students living in a rural or urban area for those enrolled at Mousinho da Silveira. A chi-square test revealed strong evidence that there is a statistically significant association between school and student's home location.

2. Are there differences in academic performance based on demographic variables?

- Males performed slightly better than females for the first, second, and final grade. An independent-samples t-test revealed strong evidence that there is a significant difference between the mean scores of males and females (i.e., mean scores are not equal).
- Students located in urban areas performed better than students located in rural areas for first, second, and final grades. An independent-samples t-test revealed strong evidence that there is a significant difference between the mean scores of students who live in urban versus rural locations (i.e., mean scores are not equal).

3. How are students' first and second period grades related to their final grade?

- For first and final grades, there is a strong, positive, linear relationship between the two variables. A Pearson's Correlation revealed strong evidence that there is a significant relationship between first and final grades.
- For second and final grades, there is also a strong, positive, linear relationship between the two variables. A Pearson's Correlation revealed strong evidence that there is a significant relationship between second and final grades,,

4. How does parental education level relate to final grades?

- The average final grade increases as mother's education level increases. For instance, students whose mother's education level is primary education score the lowest out of the other categories and students whose mother's education level is higher education score the highest out of the other categories. A One-way ANOVA revealed strong evidence that final grades differ depending on their mother's education level.
- Similar to mother's education level, the average final grade increases as father's education level increases. For instance, students whose father's education level is primary education score the lowest out of the other categories and students whose father's education level is higher education score the highest out of the other categories. A One-way ANOVA revealed strong evidence that final grades differ depending on their father's education level. This evidence was not nearly as strong as the evidence for mother's education level.

5. Does access to family educational support, extra educational support, or paid classes improve student performance?

- Unexpectedly, students that do not receive family support perform better academically compared to students that do receive family support. An independent-samples t-test revealed not enough evidence to suggest a significant difference between the means for final grade for students that do and do not receive family support.
- Unexpectedly, students that do not receive school support perform better academically compared to students that do receive school support. An independent-samples t-test revealed not enough evidence to suggest a significant difference between the means for final grade for students that do and do not receive school support.
- Students that do attend paid classes perform better academically compared to students that do not attend paid classes. An independent-samples t-test revealed evidence for a significant difference between the means for final grade for students that do and do not attend paid classes.

6. How much do students study each week, and how is that related to their grades?

- On average, most students study for about 2 hours per week.
- For first and final grades, students performed (on average) best when studying for 3 hours per week.
- For the second grade, students performed best when studying for 4 hours per week - again, on average.
- There is a weak, positive, linear relationship between first grade and study time. A Pearson's Correlation revealed strong evidence that there is a significant relationship between first grade and study time.
- There is also a weak, positive, linear relationship between second grade and study time. A Pearson's Correlation revealed strong evidence that there is a significant relationship between second grade and study time.
- There is also a weak, positive, linear relationship between final grade and study time. A Pearson's Correlation revealed not enough evidence to suggest that there is a significant relationship between final grade and study time.

7. Do social behaviors like going out and alcohol consumption impact final grades?

- Students that go out a 'low' amount scored the highest on the final grade compared to their peers. Students that go out a 'very high' amount scored the lowest on the final grade compared to their peers. A One-way ANOVA revealed strong evidence that final grades differ depending on their going out frequency.
- For weekday alcohol consumption, students that consume a 'very low' amount of alcohol scored the highest on the final grade compared to their peers. Suprisingly, students that consume a 'low' amount of alcohol scored the lowest on the final grade compared to their peers. A One-way ANOVA revealed not enough evidence to suggest that final grades differ depending on student's weekday alcohol consumption.
- For weekend alcohol consumption, students that consume a 'very low' amount of alcohol scored the highest on the final grade compared to their peers. Students that consume a 'high' amount of alcohol scored the lowest on the final grade compared to their peers. A One-way ANOVA revealed not enough evidence to suggest that final grades differ depending on student's weekend alcohol consumption.

8. Are students who have internet access or want higher education performing better academically?

- Students who have internet access perform better as a whole compared to students who do not have internet access. An independent-samples t-test revealed not enough evidence to suggest that there is a significant difference between the the mean for final for students who have internet access or not.
- Students who want higher education perform noticeably better than students who do not want higher education. An independent-samples t-test revealed strong evidence to suggest that there is a significant difference between the the mean for final for students who want higher education or not.

9. What are the most important predictors of student success?

- Pearson's Correlations revealed significant relationships between mother's education level and final grade and father's education level and final grade.

- An independent-samples t-test revealed significant differences in the means for final grades for receiving paid classes and wanting higher education.

10. What are the differences between high-performing and low-performing students in terms of study time, support, social habits, and family background?

- Important differences between the high- and low-performing groups are the high-performing group studies more, consumes less alcohol (on both the weekdays and weekends), goes out less frequently, and both their mother and father have a higher education level compared to the low-performing group. An independent-samples t-test revealed significant differences between the high- and low-performing groups for study time, weekend alcohol consumption, going out frequency, mother's education level, and father's education level.
- A chi-square test revealed significant differences in the expected and observed frequencies for school support and performance group, suggesting strong evidence that there is a statistically significant association between the two variables.