Import required dependencies

In [1]:
```
import pandas as pd
import os
```

Deliverable 1: Collect the Data

To collect the data that you'll need, complete the following steps:

1. Using the Pandas `read_csv` function and the `os` module, import the data from the `new_full_student_data.csv` file, and create a DataFrame called student_df.
2. Use the head function to confirm that Pandas properly imported the data.

In [2]:
```
# Create the path and import the data
full_student_data = os.path.join('../Resources/new_full_student_data.csv')
student_df = pd.read_csv(full_student_data)
```

In [3]:
```
# Verify that the data was properly imported
```

Out[3]:

	student_id	student_name	grade	school_name	reading_score	math_score	school_type	schoo
0	103880842	Travis Martin	9th	Sullivan High School	59.0	88.2	Public	
1	45069750	Michael Brown	9th	Dixon High School	94.7	73.5	Charter	
2	45024902	Gabriela Lucero	9th	Wagner High School	89.0	70.4	Public	
3	62582498	Susan Richardson	9th	Silva High School	69.7	80.3	Public	
4	16437227	Sherry Davis	11th	Bowers High School	NaN	27.5	Public	

Deliverable 2: Prepare the Data

To prepare and clean your data for analysis, complete the following steps:

1. Check for and remove all rows with `NaN`, or missing, values in the student DataFrame.
2. Check for and remove all duplicate rows in the student DataFrame.
3. Use the `str.replace` function to remove the "th" from the grade levels in the grade column.
4. Check data types using the `dtypes` property.
5. Remove the "th" suffix from every value in the grade column using `str` and `replace`.
6. Change the grade colum to the `int` type and verify column types.
7. Use the head (and/or the tail) function to preview the DataFrame.

In [4]:
```
# Check for null values
```

Out[4]:
```
student_id          0
student_name        0
grade               0
school_name         0
reading_score    1968
math_score        982
school_type         0
school_budget       0
dtype: int64
```

In [5]:
```
# Drop rows with null values and verify removal
```

Out[5]:
```
student_id       0
student_name     0
grade            0
school_name      0
reading_score    0
math_score       0
school_type      0
school_budget    0
dtype: int64
```

In [6]:
```
# Check for duplicated rows
```

Out[6]:
```
1836
```

In [7]:
```
# Drop duplicated rows and verify removal
```

Out[7]:
```
0
```

In [8]:
```
# Check data types
```

Out[8]:
```
student_id         int64
student_name      object
grade             object
school_name       object
reading_score    float64
math_score       float64
school_type       object
school_budget      int64
dtype: object
```

In [9]:
```
# Examine the grade column to understand why it is not an int
```

Out[9]:
```
0         9th
1         9th
2         9th
3         9th
5         9th
         ...
19508    10th
19509    12th
19511    11th
19512    11th
19513    12th
Name: grade, Length: 14831, dtype: object
```

In [10]:
```
# Remove the non-numeric characters and verify the contents of the column
```

Out[10]:
```
0         9
1         9
2         9
3         9
5         9
         ..
19508    10
19509    12
19511    11
19512    11
19513    12
Name: grade, Length: 14831, dtype: object
```

In [11]:
```
# Change the grade column to the int type and verify column types
```

Out[11]:
```
student_id         int64
student_name      object
grade              int64
school_name       object
reading_score    float64
math_score       float64
school_type       object
school_budget      int64
dtype: object
```

Deliverable 3: Summarize the Data

Describe the data using summary statistics on the data as a whole and on individual columns.

1. Generate the summary statistics for each DataFrame by using the `describe` function.
2. Display the mean math score using the `mean` function.
3. Store the minimum reading score as `min_reading_score` .

In [12]:
```
# Display summary statistics for the DataFrame
```

Out[12]:

	student_id	grade	reading_score	math_score	school_budget
count	1.483100e+04	14831.000000	14831.000000	14831.000000	14831.000000
mean	6.975296e+07	10.355539	72.357865	64.675733	893742.749107
std	3.452909e+07	1.097728	15.224590	15.844093	53938.066467
min	1.000906e+07	9.000000	10.500000	3.700000	817615.000000
25%	3.984433e+07	9.000000	62.200000	54.500000	846745.000000
50%	6.965978e+07	10.000000	73.800000	65.300000	893368.000000
75%	9.927449e+07	11.000000	84.000000	76.000000	956438.000000
max	1.299997e+08	12.000000	100.000000	100.000000	991918.000000

In [13]:
```
# Display the mean math score using the mean function
```

Out[13]:
```
64.67573326141189
```

In [14]:
```
# Store the minimum reading score as min_reading_score
```

Out[14]:
```
10.5
```

Deliverable 4: Drill Down into the Data

Drill down to specific rows, columns, and subsets of the data.

To drill down into the data, complete the following steps:

1. Use `loc` to display the grade column.
2. Use `iloc` to display the first 3 rows and columns 3, 4, and 5.
3. Show the rows for grade nine using `loc`.
4. Store the row with the minimum overall reading score as `min_reading_row` using `loc` and the `min_reading_score` found in Deliverable 3.
5. Find the reading scores for the school and grade from the output of step three using `loc` with multiple conditional statements.
6. Using conditional statements and `loc` or `iloc`, find the mean reading score for all students in grades 11 and 12 combined.

In [15]:
```
# Use loc to display the grade column
```

Out[15]:
```
0         9
1         9
2         9
3         9
5         9
         ..
19508    10
19509    12
19511    11
19512    11
19513    12
Name: grade, Length: 14831, dtype: int64
```

In [26]:
```
# Use `iloc` to display the first 3 rows and columns 3, 4, and 5.
```

Out[26]:

	school_name	reading_score	math_score
0	Sullivan High School	59.0	88.2
1	Dixon High School	94.7	73.5
2	Wagner High School	89.0	70.4

In [17]:
```
# Select the rows for grade nine and display their summary statistics using
```

Out[17]:

	student_id	grade	reading_score	math_score	school_budget
count	4.132000e+03	4132.0	4132.000000	4132.000000	4132.000000
mean	6.979441e+07	9.0	69.236713	66.585624	898692.606002
std	3.470565e+07	0.0	15.277354	16.661533	54891.596611
min	1.000906e+07	9.0	17.900000	5.300000	817615.000000
25%	3.953848e+07	9.0	59.000000	56.000000	846745.000000
50%	6.984037e+07	9.0	70.050000	67.800000	893368.000000
75%	9.939504e+07	9.0	80.500000	78.500000	957299.000000
max	1.299997e+08	9.0	99.900000	100.000000	991918.000000

In [18]:
```
# Store the row with the minimum overall reading score as `min_reading_row`
# using `loc` and the `min_reading_score` found in Deliverable 3.
```

Out[18]:

	student_id	student_name	grade	school_name	reading_score	math_score	school_type	scl
3706	81758630	Matthew Thomas	10	Dixon High School	10.5	58.4	Charter	

In [19]:
```
# Use loc with conditionals to select all reading scores from 10th graders a
```

Out[19]:

	school_name	reading_score
45	Dixon High School	71.1
60	Dixon High School	59.5
69	Dixon High School	88.6
94	Dixon High School	81.5
100	Dixon High School	95.3
...	...	...
19283	Dixon High School	52.9
19306	Dixon High School	58.0
19344	Dixon High School	38.0
19368	Dixon High School	84.4
19445	Dixon High School	43.9

569 rows × 2 columns

In [20]:
```
# Find the mean reading score for all students in grades 11 and 12 combined.
```

Out[20]:
```
63.25853039200117
```

Deliverable 5: Make Comparisons Between District and Charter Schools

Compare district vs charter schools for budget, size, and scores.

Make comparisons within your data by completing the following steps:

1. Using the `groupby` and `mean` functions, look at the average reading and math scores per school type.
2. Using the `groupby` and `count` functions, find the total number of students at each school.
3. Using the `groupby` and `mean` functions, find the average budget per grade for each school type.

In [31]:
```
# Use groupby and mean to find the average reading and math scores for each
```

Out[31]:

	school_budget
school_type	
Charter	872625.656236
Public	911195.558251

In [22]:
```
# Use the `groupby`, `count`, and `sort_values` functions to find the
# total number of students at each school and sort from most students to lea
```

Out[22]:

school_name	student_count
Montgomery High School	2038
Green High School	1961
Dixon High School	1583
Wagner High School	1541
Silva High School	1109
Woods High School	1052
Sullivan High School	971
Turner High School	846
Bowers High School	803
Fisher High School	798
Richard High School	551
Campos High School	541
Odonnell High School	459
Campbell High School	407
Chang High School	171

In [32]:
```

```

Out[32]:

school_type	grade	math_score
Charter	9	70.0
	10	66.0
	11	68.0
	12	60.0
Public	9	64.0
	10	64.0
	11	59.0
	12	64.0

Deliverable 6: Summarize Your Findings

In the cell below, write a few sentences to describe any discoveries you made while performing your analysis along with any additional analysis you believe would be worthwhile.

your summary here