Import required dependencies

In [30]:
```
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 [31]:
```
# Create the path and import the data
full_student_data = os.path.join('../School District Analysis/Resources/new_full_st
student_df = pd.read_csv(full_student_data)
```

In [32]:
```
# Verify that the data was properly imported
student_df.head()
```

Out[32]:

	student_id	student_name	grade	school_name	reading_score	math_score	school_type	school_
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 [33]:
```
# Check for null values
student_df.isnull().sum()
```

Out[33]:
```
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 [34]:
```
# Drop rows with null values and verify removal
student_df = student_df.dropna()
student_df.isna().sum()
```

Out[34]:
```
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 [35]:
```
# Check for duplicated rows
student_df.duplicated().sum()
```

Out[35]:
```
1836
```

In [36]:
```
# Drop duplicated rows and verify removal
student_df = student_df.drop_duplicates()
student_df.duplicated().sum()
```

Out[36]:
```
0
```

In [37]:
```
# Check data types
student_df.dtypes
```

Out[37]:
```
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 [38]:
```
# Examine the grade column to understand why it is not an int
student_df['grade']
```

Out[38]:
```
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 [39]:
```
# Remove the non-numeric characters and verify the contents of the column
student_df['grade'] = student_df['grade'].str.replace('th', '')
student_df['grade']
```

Out[39]:
```
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 [40]:
```
# Change the grade column to the int type and verify column types
student_df['grade'] = student_df['grade'].astype(int)
student_df.dtypes
```

Out[40]:
```
student_id         int64
student_name      object
grade              int32
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 [41]:
```
# Display summary statistics for the DataFrame
student_df.describe()
```

Out[41]:

	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 [42]:
```
# Display the mean math score using the mean function
mean_math_score = student_df["math_score"].mean()
mean_math_score
```

Out[42]:
```
64.67573326141189
```

In [43]:
```
# Store the minimum reading score as min_reading_score
min_reading_score = student_df["reading_score"].min()
min_reading_score
```

Out[43]:
```
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 [44]:
```
# Use loc to display the grade column
student_df.loc[:, ["grade"]]
```

Out[44]:

	grade
0	9
1	9
2	9
3	9
5	9
...	...
19508	10
19509	12
19511	11
19512	11
19513	12

14831 rows × 1 columns

In [45]:
```
# Use `iloc` to display the first 3 rows and columns 3, 4, and 5.
student_df.iloc[0:3, [3,4,5]]
```

Out[45]:

	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 [46]:
```
# Select the rows for grade nine and display their summary statistics using `loc` a
student_df.loc[student_df["grade"] == 9].describe()
```

Out[46]:

	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 [47]:
```
# Store the row with the minimum overall reading score as `min_reading_row`
# using `loc` and the `min_reading_score` found in Deliverable 3.
min_reading_score = student_df["reading_score"].min()
min_reading_row = student_df.loc[student_df["reading_score"] == min_reading_score]
min_reading_row
```

Out[47]:

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

In [48]:
```
# Use loc with conditionals to select all reading scores from 10th graders at Dixon
dixon_df = student_df.loc[(student_df["school_name"] == "Dixon High School")
                & (student_df["grade"] == 10), "reading_score"]
dixon_df
```

Out[48]:
```
45       71.1
60       59.5
69       88.6
94       81.5
100      95.3
         ...
19283    52.9
19306    58.0
19344    38.0
19368    84.4
19445    43.9
Name: reading_score, Length: 569, dtype: float64
```

In [49]:
```
# Find the mean reading score for all students in grades 11 and 12 combined.
mean_reading_score = student_df.loc[(student_df["grade"] == 11) | (student_df["grad
mean_reading_score
```

Out[49]:
```
74.90038089192188
```

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 [50]:
```
# Use groupby and mean to find the average school budget for each school type.
avg_school_budget_by_school_type = student_df.groupby(by='school_type').mean()
avg_school_budget_by_school_type.loc[:, ["school_budget"]]
```

Out[50]:

	school_budget
school_type	
Charter	872625.656236
Public	911195.558251

In [135…
```
# Use the `groupby`, `count`, and `sort_values` functions to find the
# total number of students at each school and sort from most students to least stua
total_student_count_by_school_name = student_df.groupby(by="school_name").count()
total_student_count_by_school_name.loc[:, ["student_id"]].sort_values("student_id",
```

Out[135]:

school_name	student_id
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 [9]:
```
#Find the average math score by grade for each school type
#by using the groupby and mean functions.
avg_math_scores_by_grade_by_school_type = student_df.groupby(['school_name', 'grade
avg_math_scores_by_grade_by_school_type.loc[:, ["math_score"]]
```

Out[9]:

school_name	grade	math_score
Bowers High School	11th	50.0
	12th	58.0
	9th	49.0
Campbell High School	10th	74.0
	11th	53.0
	12th	59.0
	9th	52.0
Campos High School	10th	63.0
	11th	77.0
	12th	74.0
	9th	52.0
Chang High School	10th	79.0
	11th	70.0
	12th	67.0
	9th	57.0
Dixon High School	10th	73.0
	11th	77.0
	12th	53.0
	9th	70.0
Fisher High School	10th	62.0
	11th	54.0
	12th	73.0
	9th	77.0
Green High School	10th	67.0
	11th	60.0
	12th	58.0
	9th	72.0
Montgomery High School	10th	53.0
	11th	72.0
	12th	64.0
	9th	75.0
Odonnell High School	10th	52.0

		math_score
school_name	grade	
	11th	65.0
	12th	75.0
	9th	60.0
Richard High School	10th	67.0
	11th	77.0
	12th	53.0
	9th	65.0
Silva High School	10th	52.0
	11th	67.0
	12th	59.0
	9th	75.0
Sullivan High School	10th	60.0
	11th	65.0
	12th	60.0
	9th	61.0
Turner High School	10th	71.0
	11th	53.0
	12th	66.0
	9th	65.0
Wagner High School	10th	69.0
	11th	64.0
	12th	61.0
	9th	63.0
Woods High School	10th	68.0
	11th	58.0
	12th	61.0
	9th	71.0

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.

The objective of this analysis was to assist the chief data scientist, Maria, for a city school disctrict. Maria provided standardized test data for analysis, reporting, and presentation to provide insights about school performance trends and patterns. These insights could be used to inform discussions and strategic decisions at the school and district level. Specifically, she was interested in analyzing data on school funding and students' standardized test scores. Data source for this analysis contained every student's math and reading scores, as well as various information on the schools they attend. The major task was to aggregate the data and showcase trends in school performance.

As a tool, Pandas library along with Jupiter Notebook were used to automate and simplify the data analysis.

First, collected data were imported from a *.csv file to pandas library to be able to work with them. Second, data were prepared for analysis by inspecting and dealing with missing, duplicate, and other messy data. Third, data were analyzed using a three-way approach,i.e., (i) summerization, (ii) targeted analysis, and (iii) aggregated analysis.

Preparation for the analysis show that data source contains 1968 empty cells in reading_score column and 982 empty cells in math_score column. There are also 1836 duplicate cells in the data source. In addition, grade column in data source is present as object(string) rather than an integer. After fixing these issues in the initial part of the code "Deliverable 2", the prepared data set contains 14831 row entries and nine columns.

Summerization of the data set show that average value for reading_score, math_score, and school_budget is 72.357865, 64.675733, and 893742.749107, respectively. The min and max values for reading_score is 10.500000 and 100.000000, respectively. The min and max values for math_score is 3.700000 and 100.000000, respectively. Both reading_score and math_score have mean values similar to median values suggesting that the data set has a symmetrical distribution. The same is observed for school_budget. The min and max values for budget_school is 817615.000000 and 991918.000000, repsectively.

Targeted analysis examining the min reading_score (10.5) show the results are from the charter school Dixon High School. Interestingly, students at this school scored much better in math (58.4), which is within the 25th percentile of the school disctrict math results. This school received a below average school funding of 870334 (average: 893742.749107). These results suggest that there may be another factor affecting the test_scores at the school than the school_funding. It would be worth examining further the relationship between fund distribution at a school level and test_score.

Aggregated analysis comparing school funding based on a school type show that public schools on average received more funding than the charter schools, 911195.558251 versus 872625.656236, respectively. Comparing schools based on average math_score show difference among grades and schools. The best mean math_scores are observed at the Fisher High School and Richard High School (both scored 77). Interstingly, both schools belong to schools with the least amount of students attending them. It would be thus worth examining

the relationship between fund amount per student and test_score.