**Import required dependencies**

In [2]:



**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 [3]:



*# 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 [4]:



*# Verify that the data was properly imported*

student\_df.head()

Out[4]:

|  | **student\_id** | **student\_name** | **grade** | **school\_name** | **reading\_score** | **math\_score** | **school\_type** | **school\_budget** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 103880842 | Travis Martin | 9th | Sullivan High School | 59.0 | 88.2 | Public | 961125 |
| **1** | 45069750 | Michael Brown | 9th | Dixon High School | 94.7 | 73.5 | Charter | 870334 |
| **2** | 45024902 | Gabriela Lucero | 9th | Wagner High School | 89.0 | 70.4 | Public | 846745 |
| **3** | 62582498 | Susan Richardson | 9th | Silva High School | 69.7 | 80.3 | Public | 991918 |
| **4** | 16437227 | Sherry Davis | 11th | Bowers High School | NaN | 27.5 | Public | 848324 |

**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 [16]:



​

Out[16]:

|  | **student\_id** | **student\_name** | **grade** | **school\_name** | **reading\_score** | **math\_score** | **school\_type** | **school\_budget** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | False | False | False | False | False | False | False | False |
| **1** | False | False | False | False | False | False | False | False |
| **2** | False | False | False | False | False | False | False | False |
| **3** | False | False | False | False | False | False | False | False |
| **4** | False | False | False | False | True | False | False | False |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **19509** | False | False | False | False | False | False | False | False |
| **19510** | False | False | False | False | False | False | False | False |
| **19511** | False | False | False | False | False | False | False | False |
| **19512** | False | False | False | False | False | False | False | False |
| **19513** | False | False | False | False | False | False | False | False |

19514 rows × 8 columns

In [5]:



*# Check for null values*

student\_df.isnull().sum()

Out[5]:

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 [6]:



*# Drop rows with null values and verify removal*

student\_df.dropna(inplace**=True**)

student\_df.isnull().sum()

Out[6]:

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 [7]:



*# Check for duplicated rows*

student\_df.duplicated().sum()

Out[7]:

1836

In [8]:



*# Drop duplicated rows and verify removal*

student\_df.drop\_duplicates(inplace**=True**)

student\_df.duplicated().sum()

Out[8]:

0

In [9]:



*# Check data types*

student\_df.dtypes

Out[9]:

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 [10]:



*# Check data types*

student\_df.dtypes

Out[10]:

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 [11]:



*# Check data types*

student\_df.dtypes

Out[11]:

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 [12]:



*# Examine the grade column to understand why it is not an int*

student\_df['grade']

Out[12]:

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 [14]:



*# Remove the non-numeric characters and verify the contents of the column*

student\_df['grade'].str.replace('th', '')

Out[14]:

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 [18]:



*# Change the grade column to the int type and verify column types*

student\_df['grade'] **=** student\_df['grade'].astype(int)

student\_df.dtypes

Out[18]:

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 [19]:



*# Display summary statistics for the DataFrame*

student\_df.describe()

Out[19]:

|  | **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 [20]:



*# Display the mean math score using the mean function*

student\_df["math\_score"].mean()

Out[20]:

64.67573326141189

In [22]:



*# Store the minimum reading score as min\_reading\_score*

min\_reading\_score **=** student\_df["reading\_score"].min()

min\_reading\_score

Out[22]:

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 [23]:



*# Use loc to display the grade column*

student\_df.loc[:, [ "grade"]]

Out[23]:

|  | **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 [24]:



*# Use `iloc` to display the first 3 rows and columns 3, 4, and 5.*

student\_df.iloc[0:3, 3:6]

Out[24]:

|  | **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 [25]:



*# Select the rows for grade nine and display their summary statistics using `loc` and `describe`.*

student\_df.loc[student\_df["grade"] **==** 9].describe()

Out[25]:

|  | **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 [26]:



*# Store the row with the minimum overall reading score as `min\_reading\_row`*

*# using `loc` and the `min\_reading\_score` found in Deliverable 3.*

student\_df.loc[student\_df["reading\_score"] **==** min\_reading\_score ]

Out[26]:

|  | **student\_id** | **student\_name** | **grade** | **school\_name** | **reading\_score** | **math\_score** | **school\_type** | **school\_budget** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **3706** | 81758630 | Matthew Thomas | 10 | Dixon High School | 10.5 | 58.4 | Charter | 870334 |

In [27]:



*# Use loc with conditionals to select all reading scores from 10th graders at Dixon High School.*

student\_df.loc[(student\_df["grade"] **==** 10) **&** (student\_df["school\_name"] **==** "Dixon High School"),["school\_name","reading\_score"]]

Out[27]:

|  | **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 [28]:



*# Find the mean reading score for all students in grades 11 and 12 combined.*

student\_df.loc[(student\_df["grade"] **==** 11) **|** (student\_df["grade"] **==** 12),"reading\_score"].mean()

Out[28]:

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 [29]:



*# Use groupby and mean to find the average reading and math scores for each school type.*

student\_df.groupby("school\_type").mean().loc[:,"school\_budget"]

Out[29]:

school\_type

Charter 872625.656236

Public 911195.558251

Name: school\_budget, dtype: float64

In [30]:



*# 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 students.*

student\_df.groupby("school\_name")["school\_name"].count().sort\_values(ascending**=False**)

Out[30]:

school\_name

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

Name: school\_name, dtype: int64

In [31]:



student\_df.groupby(["school\_type", "grade"])["math\_score"].mean().round(0)

Out[31]:

school\_type grade

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

Name: math\_score, dtype: float64

**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.

\*By removing grade 9th test score did not affect the scores as a whole. the data set used for 39100 total students is too great for the removal of only 461 score to see a dent. The Thomas school individual % scores within the school but not by a large margin to affect the scores. We a little bit the dishonesty and a great deal of fraud show that someone was artificially temper with data for Thomas high school.