4.11.1 Establish the Spending Ranges per Student

You meet again with Maria and her supervisor and a question arises: how does school spending per student affect the school's average scores and passing percentages? Maria tasks you with finding an answer to this question. This information will help the school board make decisions about the budget for the upcoming school year. Maria would like to see this data organized by spending ranges for the schools. You will need to create four spending bins, or ranges, to group the school spending per student.

For this analysis, we will need to sort the school spending per student into four spending "bins," or ranges. The four bins will be dollar amounts that range from the lowest amount ($578) to the highest amount ($655) a school spends on a student.

REWIND

We determined the budget per student in the school summary when we retrieved the `per_school_capita` Series:

```
per_school_capita

Bailey High School       628.0
Cabrera High School      582.0
Figueroa High School     639.0
Ford High School         644.0
Griffin High School      625.0
Hernandez High School    652.0
Holden High School       581.0
Huang High School        655.0
Johnson High School      650.0
Pena High School         609.0
Rodriguez High School    637.0
Shelton High School      600.0
Thomas High School       638.0
Wilson High School       578.0
Wright High School       583.0
dtype: float64
```

We will need to arbitrarily determine the spending ranges in order to group the schools fairly. Each bin must include the average math and reading scores, the percentage passing math and reading, and the overall passing percentage for the spending bins, not for the schools.

The final DataFrame will look like the following image, with the spending bins included in place of the box that says "Spending Bins." (We'll determine the four spending bins later in this section.) The columns will contain the average math score, average reading score, percent passing math, percent passing reading, and percent overall passing for the schools in each spending bin.

Spending Ranges (Per Student)	Average Math Score	Average Reading Score	% Passing Math	% Passing Reading	% Overall Passing
Spending Bins	83.5	83.9	93	97	90
	81.9	83.2	87	93	81
	78.5	81.6	73	84	63
	77.0	81.0	66	81	54

To create this DataFrame, we need to do the following:

1. Group the schools into four bins, or ranges, based on the budget per student.
2. For each spending range, get the following data:
    - Average math and reading scores
    - The percentage of students passing math and reading
    - The overall passing percentage, which is the average of the percentage of students passing math and reading

Before we aggregate the school spending per student into four bins, we must establish the spending ranges. We can determine the spending ranges by using the `per_school_capita` Series that was used to get the budget per student.

Get the Spending Ranges

Before we create the bins, we need to determine the distribution of spending per student. We can find the distribution for school spending per student by using the `describe()` method.

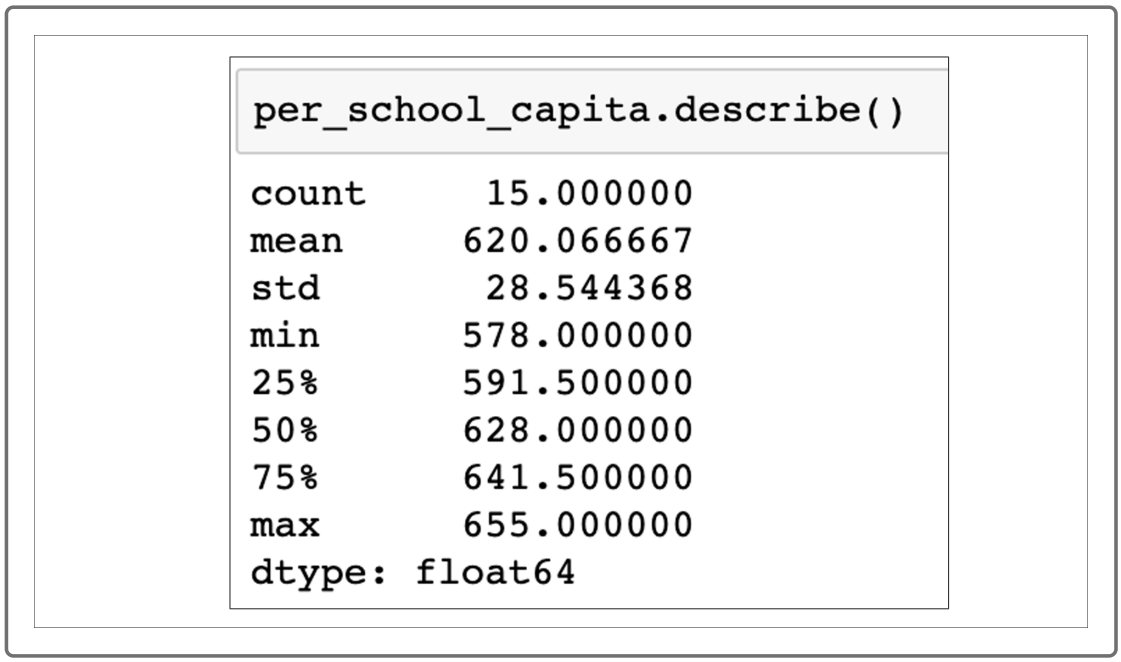
When we apply the `describe()` method to a DataFrame or Series, it will return the following descriptive statistics for the DataFrame or Series:

- The number of rows in the DataFrame or Series as `count`
- The average of the rows as `mean`
- The standard deviation of the rows as `std`
- The minimum value of the rows as `min`
- The 25th percentile as `25%`
- The 50th percentile as `50%`
- The 75th percentile as `75%`
- The minimum value of the rows as `max`

Using the `describe()` method will help us determine the spending bins we should use, based on the descriptive statistics. To get the descriptive statistics on the `per_school_capita`, add the following code to a new cell:

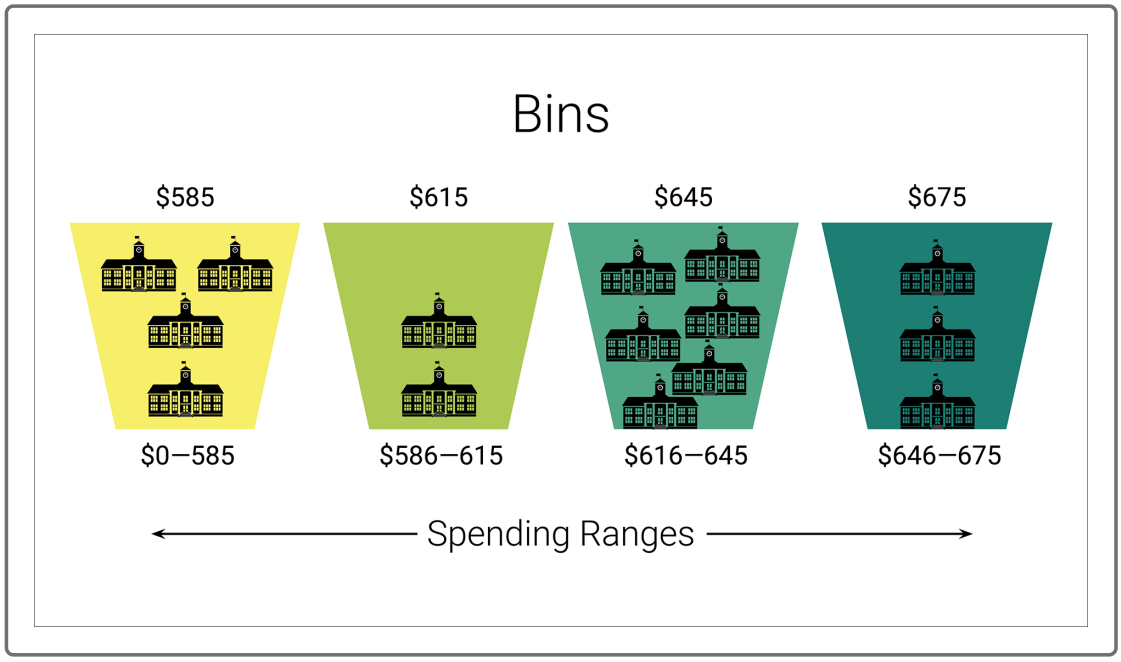
```
# Get the descriptive statistics for the per_school_capita.
per_school_capita.describe()
```

Your results should look like this:



As you can see, the minimum is 578 and the maximum is 655. The standard deviation is 28.5, or approximately 30. We don't want the lowest bin to be $578 because there is only one school at or below $578, which is Wilson High School. However, there are four schools that spend less than $585 per student, so $585 will be our lowest bin. Also, because the standard deviation is about 30, we will increase the bins by $30, up to $675. Therefore, the four bins will be: $585, $615, $645, and $675.

The bins will define a range of spending. For the $585 bin, the range is all values at or below $585, the range for the $615 bin is $586–615, and so on. Here's what our bins look like:



As shown in this image, each bin has a lower "edge" and an upper "edge" value. Schools that spend $0–585 will be placed in the first bin. Schools that spend $586–615 will be placed in the second bin, and so on.

In Pandas, we can write the ranges for the bins as follows: `spending_bins = [0, 585, 615, 645, 675]`.

Let's go over why these spending bins have five numbers instead of four, which is the number of bins we want. [When Pandas creates the lower edge, it needs a value lower than the lowest value in the column of the `per_school_capita` Series, which happens to be 578. In our DataFrame, we make the lower edge equal to 0. If we don't include the 0 for the spending bins, the lowest bin becomes $585–614, which means that the schools that spend less than $585 are not considered. First we'll use the correct spending bins for our ranges, and then we'll leave out the 0 to see how that affects the bins.

Let's address one more item before moving on. If you look at the `spending_bins` list, you'll notice that the dollar sign ($) isn't included with the ranges. This is because the numbers in the `per_school_capita` Series are the `float64` data type, so adding a dollar sign would cause an error.

Now let's take a look at how to use the spending bins.

Group the Series on the Spending Ranges

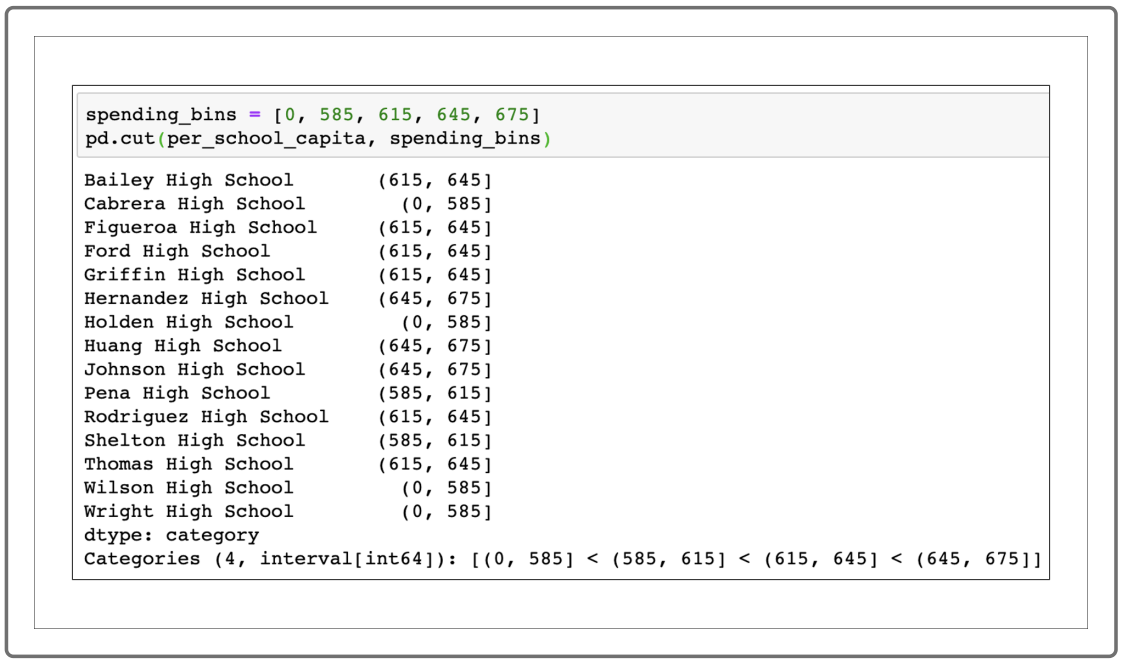
To group the spending ranges, we use the Pandas `cut()` function. The `cut()` function segments and sorts data values into bins. We'll use the `cut()` function to create our spending bins.

To apply the `cut()` function to a DataFrame or Series, we use `pd.cut(df, ranges)`. This creates a new DataFrame or Series on the given ranges.

We can "cut" the `per_school_capita` into the `spending_ranges` using the following code. Add this code to a new cell and run the cell.

```
# Cut the per_school_capita into the spending ranges.
spending_bins = [0, 585, 615, 645, 675]
pd.cut(per_school_capita, spending_bins)
```

The output shows that each school has a range for the spending per student instead of a specific amount:



```
spending_bins = [0, 585, 615, 645, 675]
pd.cut(per_school_capita, spending_bins)
```

```
Bailey High School       (615, 645]
Cabrera High School        (0, 585]
Figueroa High School     (615, 645]
Ford High School         (615, 645]
Griffin High School      (615, 645]
Hernandez High School    (645, 675]
Holden High School         (0, 585]
Huang High School        (645, 675]
Johnson High School      (645, 675]
Pena High School         (585, 615]
Rodriguez High School    (615, 645]
Shelton High School      (585, 615]
Thomas High School       (615, 645]
Wilson High School         (0, 585]
Wright High School         (0, 585]
dtype: category
Categories (4, interval[int64]): [(0, 585] < (585, 615] < (615, 645] < (645, 675]]
```

The `cut()` function places each value of the `per_school_capita` Series into one of four bins. The bins are created by a list of five numbers given as the `spending_bins` that define the edges.

What happens if we don't add 0 as the lower edge for the first bin? Copy the following code in a new cell and run the cell.

```
# Cut the per_school_capita into the spending ranges.
spending_bins = [585, 615, 645, 675]
pd.cut(per_school_capita, spending_bins)
```

The output shows that four high schools—Cabrera, Holden, Wilson, and Wright—are not in a bin and therefore would not be in the final DataFrame.

```
spending_bins = [585, 615, 645, 675]
pd.cut(per_school_capita, spending_bins)
```

```
Bailey High School       (615.0, 645.0]
Cabrera High School                 NaN
Figueroa High School     (615.0, 645.0]
Ford High School         (615.0, 645.0]
Griffin High School      (615.0, 645.0]
Hernandez High School    (645.0, 675.0]
Holden High School                  NaN
Huang High School        (645.0, 675.0]
Johnson High School      (645.0, 675.0]
Pena High School         (585.0, 615.0]
Rodriguez High School    (615.0, 645.0]
Shelton High School      (585.0, 615.0]
Thomas High School       (615.0, 645.0]
Wilson High School                  NaN
Wright High School                  NaN
dtype: category
Categories (3, interval[int64]): [(585, 615] < (615, 645] < (645, 675]]
```

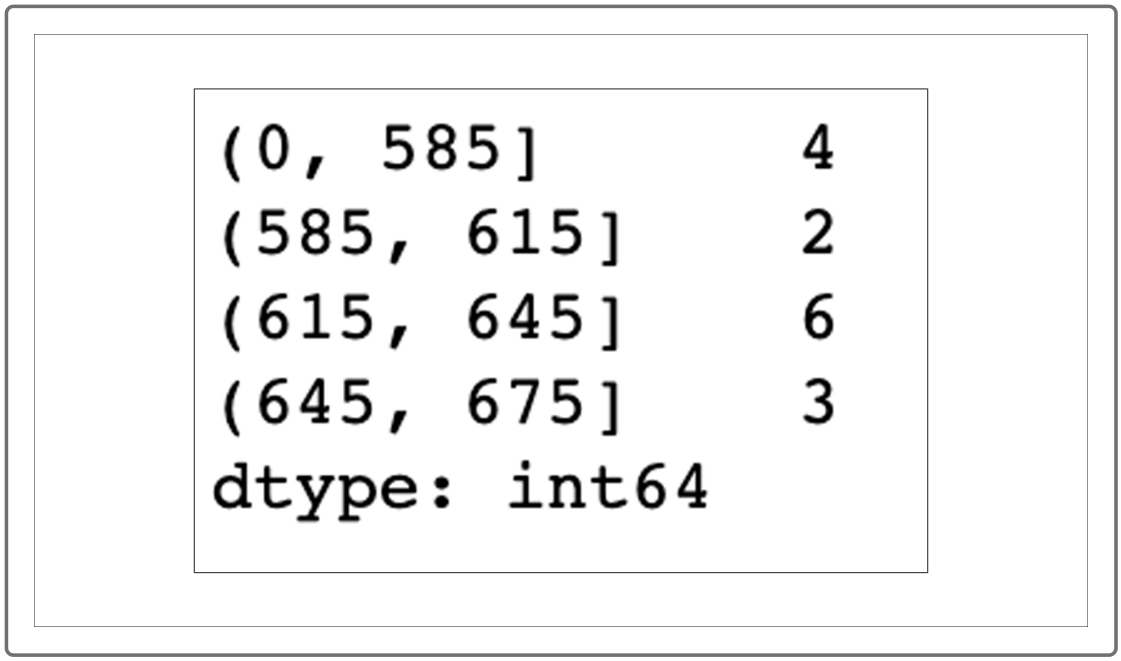
Next, we'll determine how many schools are in each range by grouping the spending bins as the index using the `groupby()` function on the `per_school_capita` Series. Remember that we need to perform a mathematical operation when we use the `groupby()` function.

To find out how many schools are in those bins, use the `count()` method on the `groupby()` function. Edit your previous code so that it looks like the

following:

```
# Cut the per_school_capita into the spending ranges.
spending_bins = [0, 585, 615, 645, 675]
per_school_capita.groupby(pd.cut(per_school_capita, spending_bins)).count()
```

When we run the cell, the output is the number of schools in each range:



Looking at this output, it seems that we didn't group the schools fairly. We need to adjust our ranges so that we have three or four schools in each range. Let's increase the `[585, 615]` range to `[585, 630]` and change the third range to `[630, 645]`. The `spending_bins` will look like this:

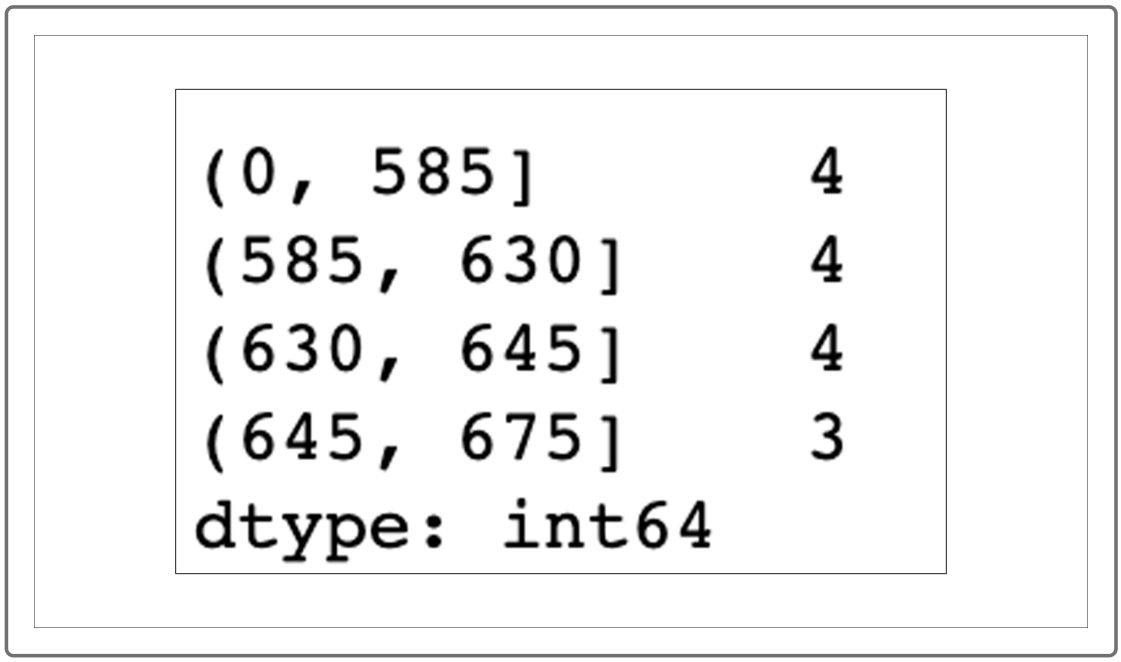
`[0, 585, 630, 645, 675]`

Edit your previous code and run the cell again. The code should look like this:

```
# Cut the per_school_capita into the spending ranges.
spending_bins = [0, 585, 630, 645, 675]
```

```
per_school_capita.groupby(pd.cut(per_school_capita, spending_bins)).count()
```

Now the output shows that there is a more fair distribution of schools in each range:



These ranges look better! Now let's name each range. We can label the ranges using a list of string values. Add the following code to a new cell and run the cell.

```
# Establish the spending bins and group names.
spending_bins = [0, 585, 630, 645, 675]
group_names = ["<$584", "$585-629", "$630-644", "$645-675"]
```

Now that we have our correct spending bins and group names, we can "cut" the `per_school_capita` Series to get the four spending bins and add them to the `per_school_summary_df` DataFrame.