The <thead> element is the <table> equivalent of <header> and defines a context for heading information. Like the <thead>, the <tbody> element defines the area where the tabular content will reside. In both cases they define an area, but on their own they do not show content. While not used in this example, the <tfoot> element exists to provide summary information, such as totals.

A table in HTML is built by rows, not columns. Each cell of the table is contained within a <tr> element. These elements are usually descendants of <thead>, <tbody>, and <tfoot>, but can also be a direct descendant of <table> if the area elements are not used.

Return to the index.html in your text editor to add a single heading row and three rows of content in the body, as highlighted in the following code block:

index.html

...

<table>

<caption>2019 Fourth Quarter Report</caption>

<thead>

<tr></tr>

</thead>

<tbody>

<tr></tr>

<tr></tr>

<tr></tr>

</tbody>

</table>

...

Copy

The last two elements are functionally similar and are both the final element in the <table> HTML structure, meaning unlike the previous elements, these can contain non-table elements.

<td> elements contain the individual table data points. <th> defines content as a heading for a row or a column. Table elements are unique to HTML as their markup structure correlates directly with a visual structure. When considering a table as a spreadsheet, the <th> and <td> elements behave as the cells. In order to have four columns in this table, each <tr> will need to have no more and no less than four <td> or <th> elements. Depending on the content of the data, this may mean there is a blank <th> or <td> element. It can be helpful to use an HTML comment to explain when an element is intentionally left blank.

Add the highlighted HTML from the following code block to your index.html file:

index.html

...

<table>

<caption>2019 Fourth Quarter Report</caption>

<thead>

<tr>

<th><!-- Intentionally Blank --></th>

<th>October</th>

<th>November</th>

<th>December</th>

<tr>

</thead>

<tbody>

<tr>

<th>Projected</th>

<td>$820,180</td>

<td>$841,640</td>

<td>$732,270</td>

</tr>

<tr>

<th>Actual</th>

<td>$850,730</td>

<td>$892,580</td>

<td>$801,240</td>

</tr>

<tr>

<th>Utilization</th>

<td>83%</td>

<td>90%</td>

<td>75%</td>

</tr>

</tbody>

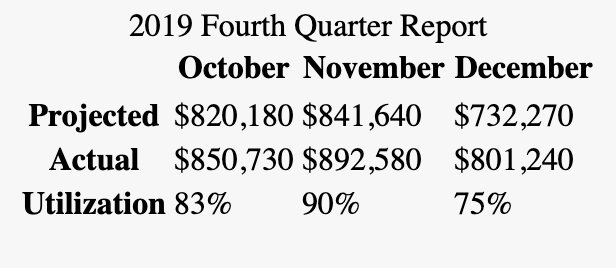
</table>

...

Copy

**Note:** Like in spreadsheet software, there are times that it may be necessary to merge cells, such as when a single cell takes up two columns. This is possible to do, but only with the use of HTML attributes on the cells, and not with CSS. It is important to keep this in mind when dealing with more complex tables.

Now that you have written out your table, save the file. Then, open index.html in your web browser. The following image depicts how the browser default styles for this table will look when loaded in the [Firefox web browser](https://www.mozilla.org/):



In this section, you set up the HTML for the tabular data. You learned about how a table is made of a series of elements combined in a distinct order to create an accessible dataset. Next, you will use the border and border-collapse properties to begin applying styles to the table.

**Using border and border-collapse to Create the Initial Table Style**

The first step to styling a table is understanding some of the browser default styles and behaviors. This section will cover the border property and the border-collapse property, and show how to create a boundary line between cells.

To begin styling your table, create and open a file named styles.css in your text editor, in the same folder as index.html. Add a selector group consisting of a th element selector and a td element selector. Then, in the selector block, add a border property with a value of 1px solid black, as shown in the following code block:

styles.css

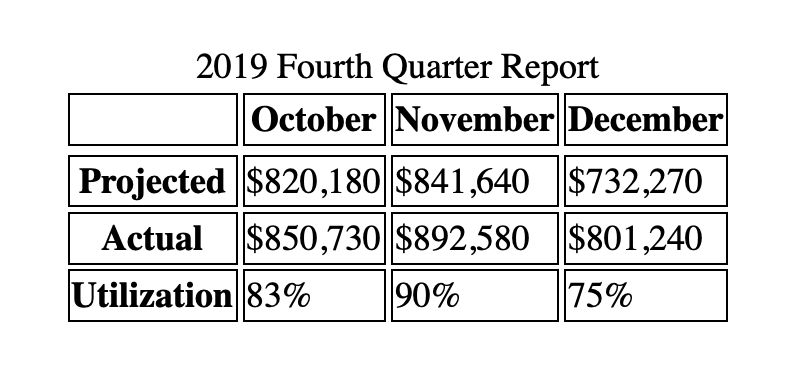
th, td {

border: 1px solid black;

}

Copy

Save your changes to styles.css and then open index.html in your web browser. Instead of a cohesive grid, there will be several boxes with their own borders. The following image depicts how the table will appear in a web browser:



To change this default, return to styles.css in your text editor and add a table element selector to the top of the file. Then, in the selector block, apply the border-collapse property. This property defaults to separate, but here you will change it to a value of collapse. This removes the spacing between the table cells and causes the borders to overlap. The highlighted CSS in the following code block indicates what to add to your styles.css file:

styles.css

table {

border-collapse: collapse;

}

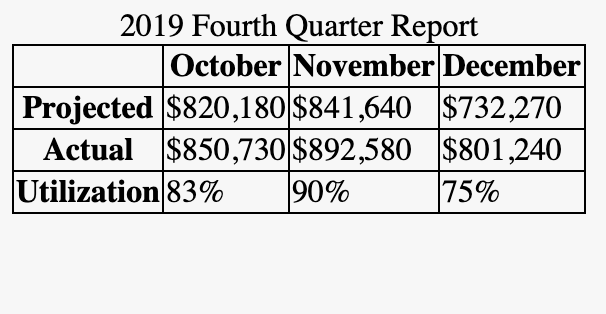
th, td {

border: 1px solid black;

}

Copy

Open your web browser and refresh index.html. The table will now have a grid defined by multiple crossing black lines. The following image depicts how the borders will appear in your browser:



In this section you used the border property to apply a border on each table cell with the th and td element selectors. You also learned that by default table cells are separated by a space. You used the border-collapse property applied to the table element selector with the collapse property to remove the space between table cells. In the next section, you will use the padding and width properties to define the size of the table.

**Setting the Size of the Table**

Next, you will add some spacing to the table cells to make the data more readable. To address adding in white space to the table cells and making the table more balanced, this section will focus on the width and padding properties.

As of now, the content of each cell is bunched up, with the borders right on top of the content. You may have also noticed that the table is only as wide as its content. <table> has its own display property: display: table. To make the table go the full width of the parent container, a width: 100% can be added to the table selector.

As this is a relatively small table, adding a width property to the <table> element isn’t necessary. Instead, open styles.css in your text editor and add a combinator selector consisting of thead th, which will scope styles to <th> elements inside of a <thead> element. Then, add a width property with a value of 25%, as indicted in the highlighted portion of the following code block:

styles.css

...

th, td {

border: 1px solid black;

}

thead th {

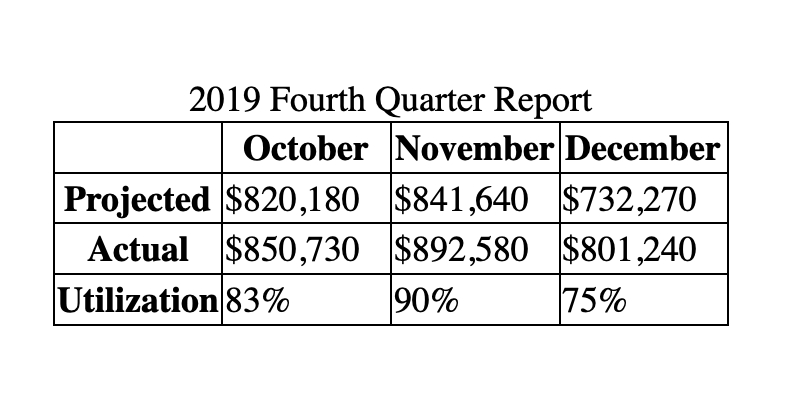
width: 25%;

}

Copy

Since there are four columns in this table, you can give each column an equal width by applying a width: 25%. Only the first cell of each column needs to be set, hence the thead th selector. The width of one cell determines the width of all cells in that column.

After saving your changes to styles.css, return to your browser and refresh index.html. The table will now have four columns of the same width, as depicted in the following image:



**Note:** If you wish to have varying widths for each column, apply a specific class to each th in the column. Then, using those classes, set your desired width.

Now that the columns are equal widths, the content of each cell could use some more space inside by using the padding property. Unlike the width property, applying space inside the cell requires targeting all the th and td cell elements.

Open styles.css in your text editor and add a padding property to the group selector for th, td, then give it a value of 8px. The highlighted line in the following code block indicates the necessary change:

styles.css

...

th, td {

border: 1px solid black;

padding: 8px;

}

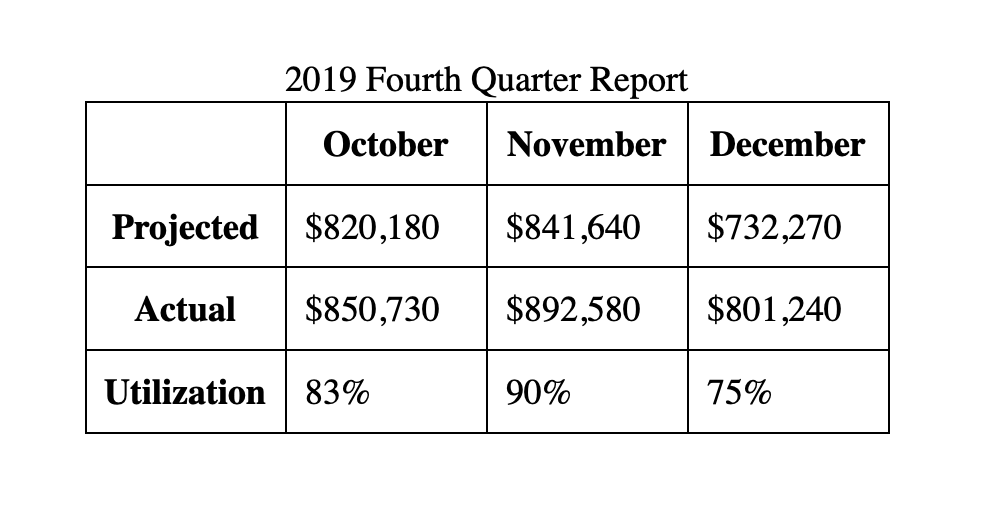
thead th {

width: 25%;

}

Copy

Save your changes to styles.css, then refresh index.html in the browser. The 8px padding is added to each side of each cell, providing space to make the tabular data more legible. The following image depicts how this will appear in the browser:



**Note:** The box model for table cells is an outlier from the usual model, and does not recognize the margin property.

In this section, you set the width property of each column to be equal and you added spacing to each cell using the padding property to make the data easier to read. In the next section, you will use a class to target and style a specific table cell.

**Targeting a Specific Table Cell**

In this step, the goal is to visually highlight one of the cells in the table. You will apply a class name in the HTML, then use a class selector and background-color property to create the highlight effect.

To begin, open index.html in your text editor and add a class attribute to the <td> element with the text **90%** inside. Give the class attribute a value of cell-highlight, as indicated in the highlighted HTML of the following code block:

index.html

<table>

...

<tr>

<th>Utilization</th>

<td>83%</td>

<td class="cell-highlight">90%</td>

<td>75%</td>

</tr>

...

</table>

Copy

Save your changes to index.html, then open styles.css in your text editor. Append to the end of the file a class selector of .cell-hightlight. Inside the selector block, add a background-color property with a value of gold. Next, add a font-weight property with the value set to bold. The highlighted CSS in the following code block demonstrates how this is formatted:

styles.css

...

thead th {

width: 25%;

}

.cell-highlight {

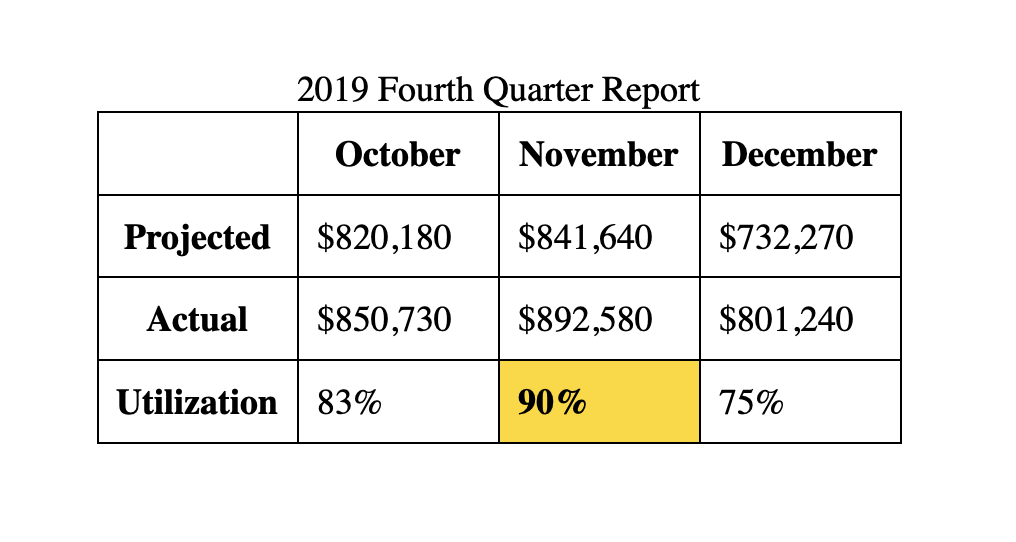
background-color: gold;

font-weight: bold;

}

Copy

Save your changes to styles.css, then return to the web browser and refresh index.html. As the following image showcases, the table cell with the content of **90%** now has a deep yellow background and a bold font weight:



You’ve now used a class selector on a specific table cell to apply a highlight style using the background-color and font-weight properties. Next, you will change the border placement, font, and text alignment to move the styles toward the final look of the table.

**Setting the Font Family for the Table**

To begin moving toward the final style of the table, you will move the border to be around the whole table, instead of the individual cells. Then, you will set a new default font-family for the page and adjust the default text alignment for the individual cells.

To update the borders, open styles.css in your text editor. Then, edit the existing group selector th, tr by removing the border: 1px solid black; property and value. This will remove the cell borders from the table; the padding will remain the same for the second stage of the table styles. Then, on the table type selector, add a border property with a value of 1px solid black. The following code block demonstrates how this will appear in your code:

styles.css

table {

border-collapse: collapse;

border: 1px solid black;

}

th, td {

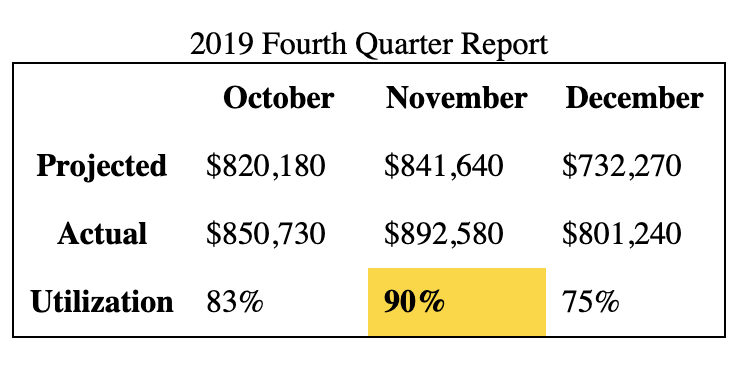
padding: 8px;

}

...

Copy

Save your changes to styles.css and return to your browser to refresh index.html. The border will now surround the whole table instead of the individual table cells, as illustrated in the following image:



To change the font for the whole document, return to styles.css in your text editor. Before the table selector block, add a body type selector. Within the body selector block, add the font-family property with a value of sans-serif. This will set the font for the page to the browser’s default sans-serif font, such as Helvetica or Arial. The highlighted CSS in the following code block indicates the changes to styles.css:

styles.css

body {

font-family: sans-serif;

}

table {

border-collapse: collapse;

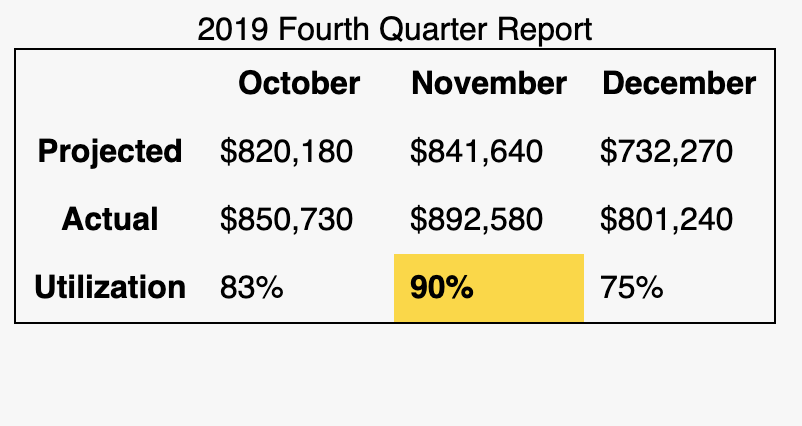
border: 1px solid black;

}

...

Copy

Save these changes to styles.css, then reload index.html in the browser. The font for the whole table will now have the browser’s default sans-serif font, as shown in the following image:



Finally, to adjust the alignment of the table’s contents, return to styles.css in your text editor. Browsers typically default the content alignment to a top left position. Similar to aligning content in a spreadsheet application, tables can have the content aligned to the middle of a table cell regardless of row height.

To set the horizontal middle alignment, go to the table type selector and add the text-align property with a value of center. Then, to set the vertical middle alignment, add the vertical-align property with a value of middle. The highlighted section of the following code block demonstrates how to add this to styles.css:

styles.css

body {

font-family: sans-serif;

}

table {

border-collapse: collapse;

border: 1px solid black;

text-align: center;

vertical-align: middle;

}

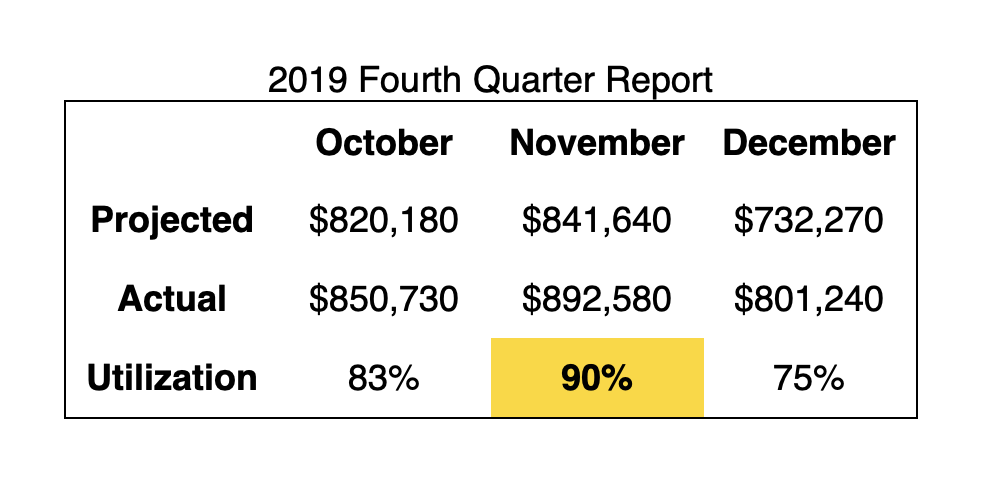
...

Copy

Save your changes to styles.css and then return to the web browser to reload index.html. The cell contents will now be horizontally and vertically centered within the cell. Note that the <th> cells have not changed their spacing. This is because table headers have centered text as a default.

The vertical centering will not be immediately evident with the content as it is, but should the content of one cell wrap to a second line, the remaining cells in the row will vertically align their contents.

The following image shows how this will appear in the browser:



In this section, you moved the border property from the table cells to the whole table. You also set a new font family for the page and changed the default alignment for the contents of the table cells. In the next section, you will add styles to the table’s <caption> element and learn more about its purpose.

**Styling the Table Caption**

The <caption> element provides context for both sighted and non-sighted readers of the table and displays above the table, regardless of where the <caption> is within the <table> element. For screen reader and braille users, a <caption> provides clear context of the purpose of a table, especially when there are multiple tables on a page.

Since <caption> is an element that only occurs inside of a <table> element, it can be styled with the caption type selector. The default for caption is centered text, with an inherited size, family, and a regular weight.

To begin changing the styles of the <caption> element, open styles.css in your text editor. Add a caption selector after the table selector in order to keep your CSS in a logical flow order. Then, using the font-weight, font-size, text-align, and color properties, create a caption that is large, bold, left-aligned, and dark gray. The highlighted CSS in the following code block demonstrates how this will be formatted:

styles.css

table {

border-collapse: collapse;

}

caption {

font-weight: bold;

font-size: 24px;

text-align: left;

color: #333;

}

th, td {

border: 1px solid black;

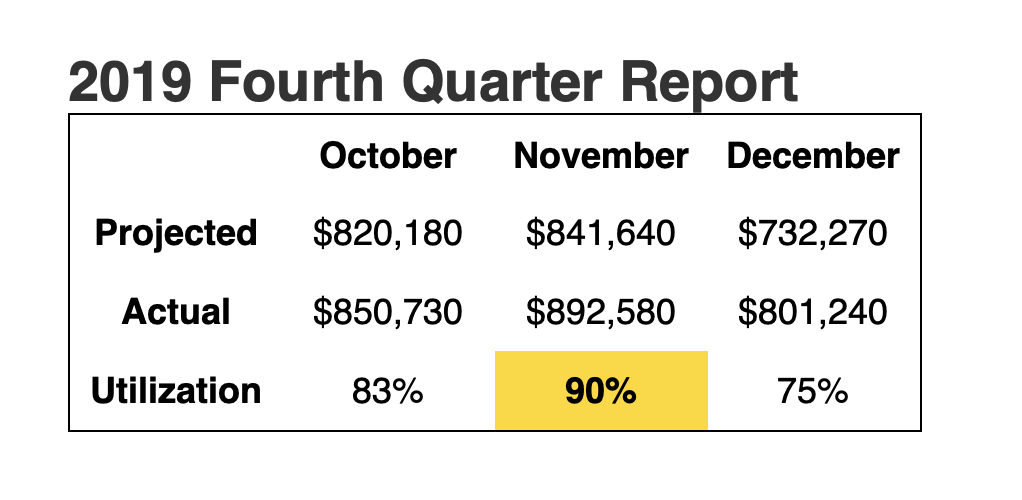
padding: 8px;

}

...

Copy

Save your changes to styles.css and reload index.html in your browser. As shown in the following image, the caption contents are now much larger and bold, creating a heading for the table:



Next, some space is needed between the caption and the visual part of the table. Return to styles.css in your text editor to add additional spacing to the caption.

The caption can accept margin and padding spacing properties. Since the spacing is only needed beneath the caption, add a margin-bottom property to the selector block with a value of 16px. The highlighted line of the following code block indicates how to apply this:

styles.css

caption {

font-weight: bold;

font-size: 24px;

text-align: left;

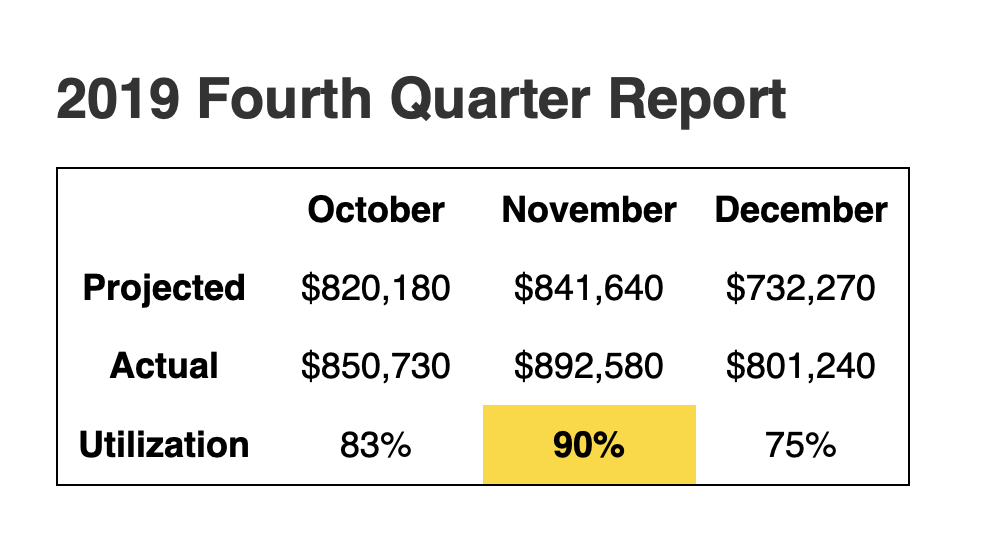
color: #333;

margin-bottom: 16px;

}

Copy

Save your changes to styles.css and refresh index.html in the web browser. The caption now has more space between the text and the table, as shown in the following image:



In this section, you created custom styles for the table’s <caption> element. You also learned that the <caption> is an important element to provide informational context to those using assistive technologies to read the table. In the next section, you will apply styles to the top heading table row.

**Styling the Top Row Header Cells**

Next, you will apply styles to the top row headings. The <thead> element will contain the top row, so all styles for this can be applied to that element directly. The intent will be to create a dark gray background with white, all uppercase text.

To start, open styles.css in your text editor. Create a new thead type selector. In the selector block, add a background-color property with a value of #333, which will create the dark gray color. Then, add a color property with a value of white:

styles.css

...

caption {

font-weight: bold;

font-size: 24px;

text-align: left;

color: #333;

margin-bottom: 16px;

}

thead {

background-color: #333;

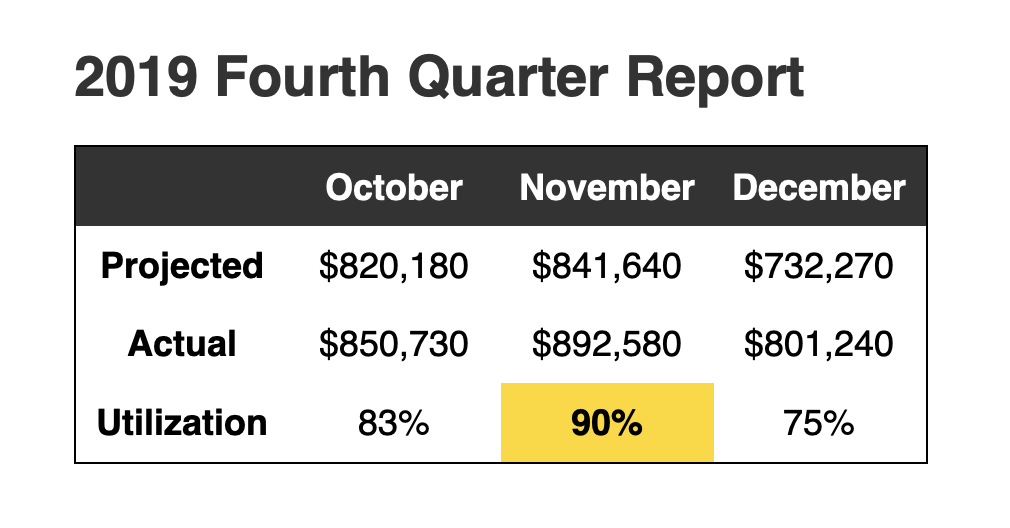
color: white;

}

...

Copy

Save you changes to styles.css and refresh index.html in your browser. The top heading row is now visually distinctive, with a solid black background and bold white text. The following image shows how this will appear in the browser:



Next, to add a bit more to the aesthetic of the top header, return to styles.css in the text editor. Change the size of the text by adding a font-size property with a value of 0.875rem, which will bring the font-size down a little. Then, to make all the letters capitals, add a text-transform property with uppercase as the value. Finally, to give some space between the letters, use the letter-spacing property and set the value to 2%. This will create a sufficient space between the uppercase characters so they aren’t bunched up, making them easier to read.

The highlighted CSS in the following code block demonstrates how to format these styles:

styles.css

thead {

background-color: #333;

color: white;

font-size: 0.875rem;

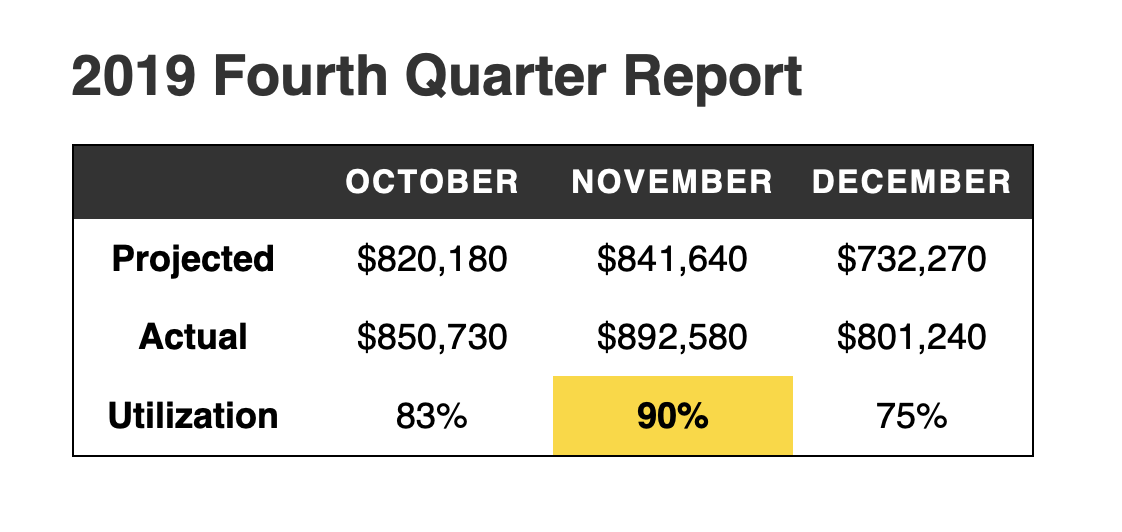
text-transform: uppercase;

letter-spacing: 2%;

}

Copy

Save your changes to styles.css and then return to your browser to refresh index.html. As shown in the following image, the text is now uppercase, a bit smaller in size than the cell contents, but hierarchically clear as a heading:



In this step, you used several properties to provide a discernible style to the top heading row. The HTML for this portion of the table was already accessible for non-sighted and assertive technology users. Now, the visual styles provide more contextual information. Next, you will continue to work with visual aids by adding alternating row colors.

**Adding Striped Row Styles to the Table**

Next, to create the alternating stripe colors, you will need to use what is called a [pseudo-class selector](https://www.digitalocean.com/community/tutorials/how-to-use-links-and-buttons-with-state-pseudo-classes-in-css). There are various kinds of pseudo-classes, and in this case you will use the :nth-child() pseudo-class. The parentheses after :nth-child can take various numbers and word values to create an alternating style that will include the odd and even values.

To begin, open styles.css in your text editor. The :nth-child() pseudo-class works by applying it to sibling elements. In this case, that will be <tr> elements inside of <tbody>. To create the first value, write a tbody tr combinator selector, immediately followed by the :nth-child(odd) pseudo-class. In this selector block, set the background-color property to #fff, the hexadecimal shorthand for white. Then create another selector in the same format, but use even instead of odd and set that background-color property to the light gray #eee value.

The highlighted CSS in the following code block showcases how this will appear in your text editor:

styles.css

...

.cell-highlight {

background-color: gold;

font-weight: bold;

}

tbody tr:nth-child(odd) {

background-color: #fff;

}

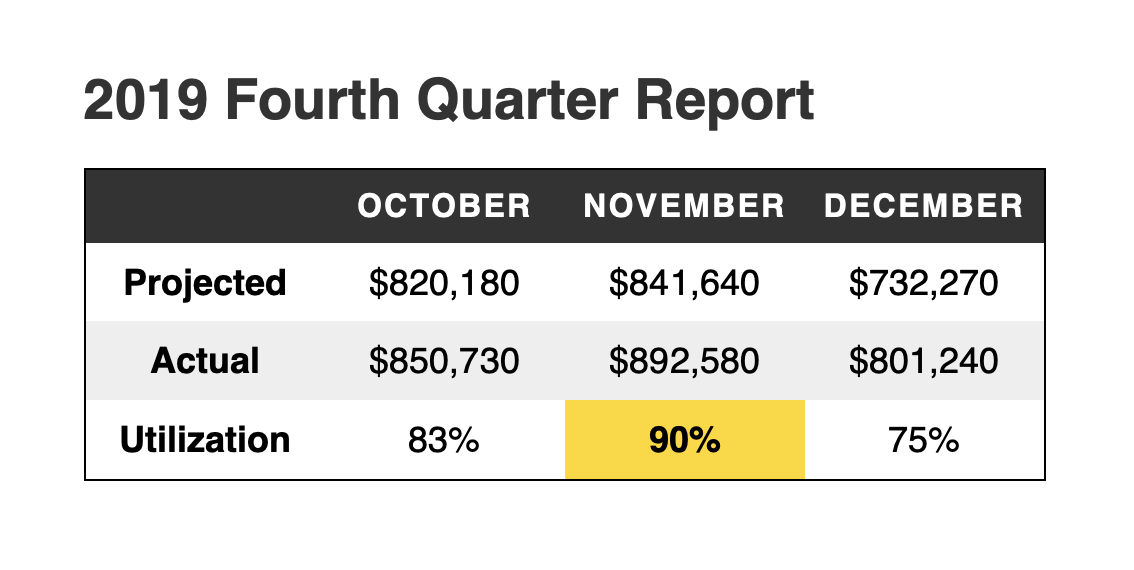
tbody tr:nth-child(even) {

background-color: #eee;

}

Copy

Save your changes to styles.css then return to index.html in your browser and refresh the page. The second row will now have a light gray background, and though it won’t look different, the odd rows now have a defined white background instead of its default transparent. As you add rows, these styles will alternate from white to light gray. The following image demonstrates how this will appear in the browser:



In this section, you used the :nth-child() pseudo-class to create alternating row colors in the body section of the table. In the last section of this tutorial, you will pair together what you learned in the previous two sections to create custom styles for the row headings on the left side of the table.

**Styling the Left Side Header Cells**

The last styling for this table is to add a blue background to the y-axis headers on the left side of the table. This will occur in two parts: the first will be similar to the section styling the top heading row by targeting the th cells in each row. Then you will create a color shift using the same :nth-child() pseudo-class approach from the previous section.

To apply the primary blue background, open your styles.css file in a text editor. You need to target the <th> elements in the <tbody> so that the <th> elements in the <thead> don’t get these styles. Create a combinator selector of tbody th, then give it a background-color property and value of #36c. Apply a color property with a value of #fff or white. Finally, to set the text to be left-aligned, add a text-align property set to the value of left:

styles.css

...

tbody tr:nth-child(even) {

background-color: #eee;

}

tbody th {

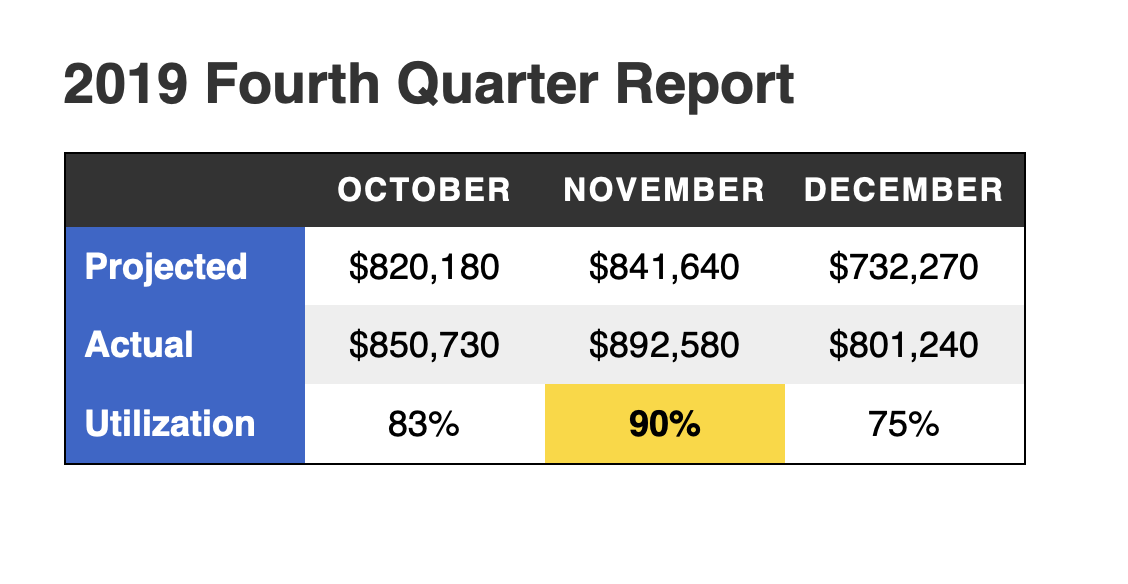
background-color: #36c;

color: #fff;

text-align: left;

}

Save your change to styles.css, then refresh index.html in your browser. As shown in the following image, the row headings are now a distinctive blue color with white text:



Lastly, to carry over the alternating row colors into the row header, return to styles.css in your text editor. To accomplish the same effect as the data row, you will need an :nth-child() pseudo-class selector. Since a blue background is already set on the tbody th combinator selector, you only need :nth-child(even) to adjust to a darker blue. However, due to how the :nth-child() pseudo-class selector works, you will still need to apply it to the <tr> element and not the <th> element, since the row (tr) counting is how the effect is achieved. This will require a more complex combinator selector of tbody tr:nth-child(even) th with a background-color property set to #25c.

The following code block highlights how this CSS is formatted:

styles.css

...

tbody th {

background-color: #36c;

color: #fff;

text-align: left;

}

tbody tr:nth-child(even) th {

background-color: #25c;

}

Copy

Save your changes to styles.css then return to your browser one last time and refresh index.html. The styling is now complete with alternating row colors across the row headers and the data, as depicted in the following image:

In this section, you scoped styles to row headings and carried over what had been learned in previous sections to accurately target an alternating background color

