Where are we?
Recap: Tables and their parts
Lots of real-world data is naturally represented as tables, e.g.,

<table>
<thead>
<tr>
<th>Name</th>
<th>NRO</th>
<th>Asmt 1</th>
<th>Asmt 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alfie</td>
<td>FALSE</td>
<td>85%</td>
<td>90%</td>
</tr>
<tr>
<td>Carl</td>
<td>FALSE</td>
<td>75%</td>
<td>60%</td>
</tr>
<tr>
<td>Elan</td>
<td>TRUE</td>
<td>95%</td>
<td>63%</td>
</tr>
<tr>
<td>Levant</td>
<td>FALSE</td>
<td>67%</td>
<td>88%</td>
</tr>
<tr>
<td>Nana</td>
<td>TRUE</td>
<td>70%</td>
<td>0%</td>
</tr>
</tbody>
</table>
Lots of real-world data is naturally represented as tables, e.g.,

```plaintext
table gradebook =
  row: "Allie", false, 85, 90
  row: "Carl",  false, 75, 60
  row: "Elan",  true, 95, 63
  row: "Lavon", false, 87, 88
  row: "Nunu",  true, 70, 0
end
```
Lots of real-world data is naturally represented as tables, e.g.,

```plaintext
gradebook

<table>
<thead>
<tr>
<th>name</th>
<th>NRO</th>
<th>asmt1</th>
<th>asmt2</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Allie&quot;</td>
<td>false</td>
<td>85</td>
<td>90</td>
</tr>
<tr>
<td>&quot;Carl&quot;</td>
<td>false</td>
<td>75</td>
<td>60</td>
</tr>
<tr>
<td>&quot;Elan&quot;</td>
<td>true</td>
<td>95</td>
<td>63</td>
</tr>
<tr>
<td>&quot;Lavon&quot;</td>
<td>false</td>
<td>87</td>
<td>88</td>
</tr>
<tr>
<td>&quot;Nunu&quot;</td>
<td>true</td>
<td>70</td>
<td>0</td>
</tr>
</tbody>
</table>
```
To get a particular row from a table, we use its numeric index $n$, counting from 0:

\[
\langle \text{table} \rangle . \text{row-} n(0)
\]
```python
>>> gradebook

<table>
<thead>
<tr>
<th>name</th>
<th>NRO</th>
<th>asmt1</th>
<th>asmt2</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Allie&quot;</td>
<td>false</td>
<td>85</td>
<td>90</td>
</tr>
<tr>
<td>&quot;Carl&quot;</td>
<td>false</td>
<td>75</td>
<td>60</td>
</tr>
<tr>
<td>&quot;Elan&quot;</td>
<td>true</td>
<td>95</td>
<td>63</td>
</tr>
<tr>
<td>&quot;Lavon&quot;</td>
<td>false</td>
<td>87</td>
<td>88</td>
</tr>
<tr>
<td>&quot;Nunu&quot;</td>
<td>true</td>
<td>70</td>
<td>0</td>
</tr>
</tbody>
</table>

>>> gradebook.row-n(0)

| "name" | "Allie" | "NRO" | false | "asmt1" | 85 | "asmt2" | 90 |
```
<table>
<thead>
<tr>
<th>name</th>
<th>NRO</th>
<th>asmt1</th>
<th>asmt2</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Allie&quot;</td>
<td>false</td>
<td>85</td>
<td>90</td>
</tr>
<tr>
<td>&quot;Carl&quot;</td>
<td>false</td>
<td>75</td>
<td>60</td>
</tr>
<tr>
<td>&quot;Elan&quot;</td>
<td>true</td>
<td>95</td>
<td>63</td>
</tr>
<tr>
<td>&quot;Lavon&quot;</td>
<td>false</td>
<td>87</td>
<td>88</td>
</tr>
<tr>
<td>&quot;Nunu&quot;</td>
<td>true</td>
<td>70</td>
<td>0</td>
</tr>
</tbody>
</table>

```python
>>> gradebook.row_n(1)

<table>
<thead>
<tr>
<th>&quot;name&quot;</th>
<th>&quot;Carl&quot;</th>
<th>&quot;NRO&quot;</th>
<th>false</th>
<th>&quot;asmt1&quot;</th>
<th>75</th>
<th>&quot;asmt2&quot;</th>
<th>60</th>
</tr>
</thead>
</table>
```
```plaintext
>>> gradebook

<table>
<thead>
<tr>
<th>name</th>
<th>NRO</th>
<th>asmt1</th>
<th>asmt2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allie</td>
<td>false</td>
<td>85</td>
<td>90</td>
</tr>
<tr>
<td>Carl</td>
<td>false</td>
<td>75</td>
<td>60</td>
</tr>
<tr>
<td>Elan</td>
<td>true</td>
<td>95</td>
<td>63</td>
</tr>
<tr>
<td>Lavon</td>
<td>false</td>
<td>87</td>
<td>88</td>
</tr>
<tr>
<td>Nunu</td>
<td>true</td>
<td>70</td>
<td>0</td>
</tr>
</tbody>
</table>

>>> gradebook.row-n(2)

<table>
<thead>
<tr>
<th>name</th>
<th>Elan</th>
<th>NRO</th>
<th>true</th>
<th>asmt1</th>
<th>asmt2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allie</td>
<td></td>
<td>false</td>
<td></td>
<td>85</td>
<td>90</td>
</tr>
<tr>
<td>Carl</td>
<td></td>
<td>false</td>
<td></td>
<td>75</td>
<td>60</td>
</tr>
<tr>
<td>Elan</td>
<td>true</td>
<td>95</td>
<td>63</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lavon</td>
<td></td>
<td>false</td>
<td></td>
<td>87</td>
<td>88</td>
</tr>
<tr>
<td>Nunu</td>
<td></td>
<td>true</td>
<td>70</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
```
To get a particular column’s value from a row, we specify the column name using square brackets:

\[ \langle row \rangle["asmt1"] \]
```python
>>> gradebook.row-n(0)
```

```
| name  | Allie  | NRO   | false | asmt1 | 85 | asmt2 | 90 |
```
```python
>>> gradebook.row-n(0)

| name     | Allie | NRO | False | asmt1 | 85 | asmt2 | 90 |

>>> gradebook.row-n(0)['name']
Allie

>>> gradebook.row-n(0)['asmt1']
85
```
Recap: Ordering tables
To do more with tabular data, first include the textbook library:

```plaintext
include shared-gdrive("dcic-2021", "1wyQZj_L0qqV9Ekgr9au6RX2iqt2Ga8Ep")
```
We can transform tabular data to get a particular view. E.g., to order the rows from the highest Assignment 1 grades to the lowest:

```python
>>> order-by(gradebook, "asmt1", false)
```

<table>
<thead>
<tr>
<th>name</th>
<th>NRO</th>
<th>asmt1</th>
<th>asmt2</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Elan&quot;</td>
<td>true</td>
<td>95</td>
<td>63</td>
</tr>
<tr>
<td>&quot;Lavon&quot;</td>
<td>false</td>
<td>87</td>
<td>88</td>
</tr>
<tr>
<td>&quot;Allie&quot;</td>
<td>false</td>
<td>85</td>
<td>90</td>
</tr>
<tr>
<td>&quot;Carl&quot;</td>
<td>false</td>
<td>75</td>
<td>60</td>
</tr>
<tr>
<td>&quot;Nunu&quot;</td>
<td>true</td>
<td>70</td>
<td>0</td>
</tr>
</tbody>
</table>
We can transform tabular data to get a particular view. E.g., to order the rows from the lowest Assignment 1 grades to the highest:

```python
>>> order-by(gradebook, "asmt1", true)
```

<table>
<thead>
<tr>
<th>name</th>
<th>NRO</th>
<th>asmt1</th>
<th>asmt2</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Nunu&quot;</td>
<td>true</td>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>&quot;Carl&quot;</td>
<td>false</td>
<td>75</td>
<td>60</td>
</tr>
<tr>
<td>&quot;Allie&quot;</td>
<td>false</td>
<td>85</td>
<td>90</td>
</tr>
<tr>
<td>&quot;Lavon&quot;</td>
<td>false</td>
<td>87</td>
<td>88</td>
</tr>
<tr>
<td>&quot;Elan&quot;</td>
<td>true</td>
<td>95</td>
<td>63</td>
</tr>
</tbody>
</table>
```python
>>> gradebook.row-n(0)

| name | Allie | NRO | false | asmt1 | 85 | asmt2 | 90 |

```

```python
>>> order-by(gradebook, "asmt1", false).row-n(0)

| name | Elan | NRO | true | asmt1 | 95 | asmt2 | 63 |
```
```python
>>> gradebook.row-n(0)
```

```
<table>
<thead>
<tr>
<th>name</th>
<th>Allie</th>
<th>NRO</th>
<th>false</th>
<th>asmt1</th>
<th>85</th>
<th>asmt2</th>
<th>90</th>
</tr>
</thead>
</table>
```

```python
>>> order-by(gradebook, "asmt1", false).row-n(0)
```

```
<table>
<thead>
<tr>
<th>name</th>
<th>Elan</th>
<th>NRO</th>
<th>true</th>
<th>asmt1</th>
<th>95</th>
<th>asmt2</th>
<th>63</th>
</tr>
</thead>
</table>
```

```python
>>> ordered = order-by(gradebook, "asmt1", false)
>>> ordered.row-n(0)
```

```
<table>
<thead>
<tr>
<th>name</th>
<th>Elan</th>
<th>NRO</th>
<th>true</th>
<th>asmt1</th>
<th>95</th>
<th>asmt2</th>
<th>63</th>
</tr>
</thead>
</table>
```
>>> ordered = order-by(gradebook, "asmt1", false)
>>> best-student = row-n(0)
>>> best-student["asmt1"]
95
Recap: Filtering tables
Make a table keeping only those students who have NROed:

```plaintext
fun nro-ed(r :: Row) -> Boolean:
    r["NR0"]
end

>>> filter-with(gradebook, nro-ed)
```

<table>
<thead>
<tr>
<th>name</th>
<th>NRO</th>
<th>asmt1</th>
<th>asmt2</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Elan&quot;</td>
<td>true</td>
<td>95</td>
<td>63</td>
</tr>
<tr>
<td>&quot;Nunu&quot;</td>
<td>true</td>
<td>70</td>
<td>0</td>
</tr>
</tbody>
</table>
Make a table keeping only those students whose grades dropped from Assignment 1 to Assignment 2:

```fun asmt2-lower(r :: Row) -> Boolean:
    r["asmt1"] > r["asmt2"]
end

>>> filter-with(gradebook, asmt2-lower)
```

<table>
<thead>
<tr>
<th>name</th>
<th>NRO</th>
<th>asmt1</th>
<th>asmt2</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Carl&quot;</td>
<td>false</td>
<td>75</td>
<td>60</td>
</tr>
<tr>
<td>&quot;Elan&quot;</td>
<td>true</td>
<td>95</td>
<td>63</td>
</tr>
<tr>
<td>&quot;Nunu&quot;</td>
<td>true</td>
<td>70</td>
<td>0</td>
</tr>
</tbody>
</table>
Example: Grade look-up revisited
fun look-up-grade(student :: String, asmt :: String) -> Number:
    doc: "Return grade of a given student on a given assignment"
    if student == "Allie":
        if asmt == "asmt1":
            85
        else if asmt == "asmt2":
            90
        else:
            raise("No such assignment")
    end
    else if student == "Carl":
        if asmt == "asmt1":
            75
        else if asmt == "asmt2":
            60
        else:
            raise("No such assignment")
    end
    else:
        raise("No such student")
    end
end
Break the problem down:

Given a student name and an assignment ID,

Filter the table to just the row for the student
Select the column for the assignment.
fun look-up-grade(student :: String, asmt :: String) -> Number:
    doc: "Return grade of a given student on a given assignment"
    ...
end
fun look-up-grade(student :: String, asmt :: String) -> Number:
    doc: "Return grade of a given student on a given assignment"
    filter-with(gradebook, some-function)
end
fun **look-up-grade**(student :: String, asmt :: String) -> Number:
  doc: "Return grade of a given student on a given assignment"
  filter-with(gradebook, **some-function**) ...
fun look-up-grade(student :: String, asmt :: String) -> Number:
  doc: "Return grade of a given student on a given assignment"
  filter-with(gradebook, some-function).get-row(0) ...
end
fun look-up-grade(student :: String, asmt :: String) -> Number:
    doc: "Return grade of a given student on a given assignment"
    filter-with(gradebook, some-function).get-row(0)[asmt]
end
fun look-up-grade(student :: String, asmt :: String) -> Number:
  
doc: "Return grade of a given student on a given assignment"
  filter-with(gradebook, some-function).get-row(0)[asmt]
end
fun **look-up-grade**(student :: String, asmt :: String) -> Number:
    doc: "Return grade of a given student on a given assignment"
    filter-with(gradebook, **some-function**).get-row(0)[asmt]
end

*What does some-function need to take as input?*

*What does it need to return?*
fun `find-student` (r :: Row) -> Boolean:
  ...
end

fun `look-up-grade` (student :: String, asmt :: String) -> Number:
  doc: "Return grade of a given student on a given assignment"
  filter-with(gradebook, find-student).get-row(0)[asmt]
end
fun find-student(r :: Row) -> Boolean:
   ...
end

fun look-up-grade(student :: String, asmt :: String)
   -> Number:
   doc: "Return grade of a given student on a given assignment"
   filter-with(gradebook, find-student).get-row(0)[asmt]
end

How does find-student know if this is the row we want?
fun find-student(r :: Row) -> Boolean:
    r["name"] == student
end

fun look-up-grade(student :: String, asmt :: String)
    -> Number:
    doc: "Return grade of a given student on a given assignment"
    filter-with(gradebook, find-student).get-row(0)[asmt]
end
fun find-student(r :: Row) -> Boolean:
    r["name"] == student
end

fun look-up-grade(student :: String, asmt :: String)
    -> Number:
    doc: "Return grade of a given student on a given assignment"
    filter-with(gradebook, find-student).get-row(0)[asmt]
end
fun `look-up-grade`(student :: String, asmt :: String) -> Number:

doc: "Return grade of a given student on a given assignment"

fun `find-student`(r :: Row) -> Boolean:
    r["name"] == student
end

filter-with(gradebook, find-student).get-row(0)[asmt]
end
A helper function can be used by one or more functions.

If only one function uses the helper, it can be defined *inside* the main function!

In this case, the helper depends on the input student that’s given to `look-up-grade`. This only has a value inside `look-up-grade`, so `find-student` needs to be defined inside that function.
This issue – where names are visible – is called *scope*.

We’ll return to it again!
Adding columns
We can also transform a table by adding new values.

For instance, we might want to add a column to our grade book with the average of Assignment 1 and Assignment 2 for each student.
To add this column to the table, we need a function that computes the value for each row:

```plaintext
fun asmt-avg(r :: Row) -> Number:
    ...
end
```
To add this column to the table, we need a function that computes the value for each row:

```plaintext
fun asmt-avg(r :: Row) -> Number:
  doc: "Add a column with the average of the assignment grades"
  ...
end
```
To add this column to the table, we need a function that computes the value for each row:

```haxe
fun asmt-avg(r : Row) -> Number:
    doc: "Add a column with the average of the assignment grades"
    (r["asmt1"] + r["asmt2"]) / 2
end
```
To add this column to the table, we need a function that computes the value for each row:

```pyret
fun asmt-avg(r :: Row) -> Number:
    doc: "Add a column with the average of the assignment grades"
    (r["asmt1"] + r["asmt2"]) / 2
end
```

And we tell Pyret to use this function to build a new column:

```pyret
build-column(gradebook, "avg", asmt-avg)
```
build-column ::
(t :: Table,
  colname :: String,
  builder :: (Row -> A))
-> Table
This is the second time we’ve seen a function that takes a function as one of its inputs!

Both \texttt{filter-with} and \texttt{build-column} need a helper function that tells them \textit{how} to do what we want.

Just as a function is an abstraction over specific computations, \texttt{filter-with} and \texttt{build-column} are abstractions over more specific functions. They provide the common functionality and the arguments we give provide the specifics.
Interlude: Functional programming
We can

sort the rows a table with `order-with`,
select certain rows using `filter-with`, and
add a new column of values with `build-column`

but none of these functions change the original table!
<table>
<thead>
<tr>
<th>name</th>
<th>NRO</th>
<th>asmt1</th>
<th>asmt2</th>
<th>avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Allie&quot;</td>
<td>false</td>
<td>85</td>
<td>90</td>
<td>87.5</td>
</tr>
<tr>
<td>&quot;Carl&quot;</td>
<td>false</td>
<td>75</td>
<td>60</td>
<td>67.5</td>
</tr>
<tr>
<td>&quot;Elan&quot;</td>
<td>true</td>
<td>95</td>
<td>63</td>
<td>79</td>
</tr>
<tr>
<td>&quot;Lavon&quot;</td>
<td>false</td>
<td>87</td>
<td>88</td>
<td>87.5</td>
</tr>
<tr>
<td>&quot;Nunu&quot;</td>
<td>true</td>
<td>70</td>
<td>0</td>
<td>35</td>
</tr>
</tbody>
</table>
Just as the expression $2 + 3$ doesn’t change the value of $2$ or of $3$, functions that take a table as input don’t change the original table.

Instead, they return a new table.
This is a paradigm called *functional programming*.

If you have experience working in other languages, this may seem strange, but it can be extremely useful!

We'll explore the idea of functional programming more in the coming weeks.
Example: Population growth
Let’s pivot to a very similar but more interesting data set.

Consider representing data about municipalities as a table:

```plaintext
municipalities = table:
  name, city, population-2000, population-2010
row: "Providence", true, 173618, 178042
row: "Cranston", true, 79269, 80387
row: "Coventry", false, 33668, 35014
row: "Warwick", true, 85808, 82672
row: "North Providence", false, 32411, 32078
end
```
These are just a few of the municipalities in Rhode Island.

I don’t want to type all of them into Pyret like this. Instead, let’s read this data from a Google Sheets spreadsheet.
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Barrington</td>
<td>FALSE</td>
<td>16,819</td>
<td>16,310</td>
</tr>
<tr>
<td>2</td>
<td>Bristol</td>
<td>FALSE</td>
<td>22,469</td>
<td>22,954</td>
</tr>
<tr>
<td>3</td>
<td>Burrillville</td>
<td>FALSE</td>
<td>15,796</td>
<td>15,965</td>
</tr>
<tr>
<td>4</td>
<td>Central Falls</td>
<td>TRUE</td>
<td>18,928</td>
<td>19,376</td>
</tr>
<tr>
<td>5</td>
<td>Charlestown</td>
<td>FALSE</td>
<td>7,859</td>
<td>7,827</td>
</tr>
<tr>
<td>6</td>
<td>Coventry</td>
<td>FALSE</td>
<td>33,668</td>
<td>35,014</td>
</tr>
<tr>
<td>7</td>
<td>Cranston</td>
<td>TRUE</td>
<td>79,269</td>
<td>80,387</td>
</tr>
<tr>
<td>8</td>
<td>Cumberland</td>
<td>FALSE</td>
<td>31,840</td>
<td>33,506</td>
</tr>
<tr>
<td>9</td>
<td>East Greenwich</td>
<td>FALSE</td>
<td>12,948</td>
<td>13,146</td>
</tr>
<tr>
<td>10</td>
<td>East Providence</td>
<td>TRUE</td>
<td>48,688</td>
<td>47,037</td>
</tr>
<tr>
<td>11</td>
<td>Exeter</td>
<td>FALSE</td>
<td>6,045</td>
<td>6,425</td>
</tr>
<tr>
<td>12</td>
<td>Foster</td>
<td>FALSE</td>
<td>4,274</td>
<td>4,606</td>
</tr>
<tr>
<td>13</td>
<td>Glocester</td>
<td>FALSE</td>
<td>9,948</td>
<td>9,746</td>
</tr>
<tr>
<td>14</td>
<td>Hopkinton</td>
<td>FALSE</td>
<td>7,836</td>
<td>8,188</td>
</tr>
<tr>
<td>15</td>
<td>Jamestown</td>
<td>FALSE</td>
<td>5,622</td>
<td>5,465</td>
</tr>
<tr>
<td>16</td>
<td>Johnston</td>
<td>FALSE</td>
<td>28,195</td>
<td>28,769</td>
</tr>
<tr>
<td>17</td>
<td>Lincoln</td>
<td>FALSE</td>
<td>20,868</td>
<td>21,105</td>
</tr>
<tr>
<td>18</td>
<td>Little Compton</td>
<td>FALSE</td>
<td>3,593</td>
<td>3,492</td>
</tr>
<tr>
<td>19</td>
<td>Middletown</td>
<td>FALSE</td>
<td>17,334</td>
<td>16,150</td>
</tr>
<tr>
<td>20</td>
<td>Narragansett</td>
<td>FALSE</td>
<td>16,361</td>
<td>15,688</td>
</tr>
<tr>
<td>21</td>
<td>Newport</td>
<td>TRUE</td>
<td>28,475</td>
<td>24,672</td>
</tr>
<tr>
<td>22</td>
<td>New Shoreham</td>
<td>FALSE</td>
<td>1,010</td>
<td>1,051</td>
</tr>
<tr>
<td>23</td>
<td>North Kingstown</td>
<td>FALSE</td>
<td>26,326</td>
<td>26,486</td>
</tr>
<tr>
<td>24</td>
<td>North Providence</td>
<td>FALSE</td>
<td>32,411</td>
<td>32,078</td>
</tr>
<tr>
<td>25</td>
<td>North Smithfield</td>
<td>FALSE</td>
<td>10,618</td>
<td>11,967</td>
</tr>
</tbody>
</table>
# Textbook library
include shared-gdrive("dcic-2021", "1wyQZj_L0qqV9Ekgr9au6RX2iqt2Ga8Ep")

# Google Sheets library
include gdrive-sheets

# The ID of the Google Sheets file, which appears in the URL
ssid = "1jHvn5CPE6RkTTQRIXQbY5n5p4ai0H7fZsnwK2s6s6tc"

spreadsheet = load-spreadsheet(ssid)
A spreadsheet consists of multiple sheets:

```python
>>> spreadsheet
spreadsheet("municipalities", "municipalities-counties", "municipalities-area", "municipalities-full")
```
To load a table from a spreadsheet, we need to tell Pyret which sheet to load it from and what we want the columns to be called (which can be different from what is in the spreadsheet):

```pyret
municipalities = load-table:
    name :: String,
    city :: Boolean,
    population-2000 :: Number,
    population-2010 :: Number
# true because the sheet has a "header" row
source: spreadsheet.sheet-by-name("municipalities", true)
end
```
Now we can work with this table the same as if we’d entered it manually:

<table>
<thead>
<tr>
<th>name</th>
<th>city</th>
<th>population-2000</th>
<th>population-2010</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Barrington&quot;</td>
<td>false</td>
<td>16819</td>
<td>16310</td>
</tr>
<tr>
<td>&quot;Bristol&quot;</td>
<td>false</td>
<td>22469</td>
<td>22954</td>
</tr>
<tr>
<td>&quot;Burrillville&quot;</td>
<td>false</td>
<td>15796</td>
<td>15955</td>
</tr>
<tr>
<td>&quot;Central Falls&quot;</td>
<td>true</td>
<td>18928</td>
<td>19376</td>
</tr>
<tr>
<td>&quot;Charlestown&quot;</td>
<td>false</td>
<td>7859</td>
<td>7827</td>
</tr>
<tr>
<td>&quot;Coventry&quot;</td>
<td>false</td>
<td>33668</td>
<td>35014</td>
</tr>
<tr>
<td>&quot;Cranston&quot;</td>
<td>true</td>
<td>79269</td>
<td>80387</td>
</tr>
<tr>
<td>&quot;Cumberland&quot;</td>
<td>false</td>
<td>31840</td>
<td>33506</td>
</tr>
<tr>
<td>&quot;East Greenwich&quot;</td>
<td>false</td>
<td>12948</td>
<td>13146</td>
</tr>
<tr>
<td>&quot;East Providence&quot;</td>
<td>true</td>
<td>48688</td>
<td>47937</td>
</tr>
</tbody>
</table>

Click to show the remaining 29 rows...
Given this data set, we might want to figure out which are the fastest-growing towns in Rhode Island.
Filter out cities – we only want towns!
Calculate the percentage change in population
Build a column for percentage change
Sort on that column in descending order
Try implementing this yourself and we’ll go over it on Monday.
Acknowledgments

This lecture incorporates material from:

Kathi Fisler, Brown University