Table Methods

Students learn about *table methods*, which allow them to order, filter, and build columns to extend the animals table.

Prerequisites	Defining Functions
Relevant Standards OK CSTA NGSS	*Select one or more standards from the menu on the left (⌘-click on Mac, Ctrl-click elsewhere).*
Lesson Goals	Students will be able to... • order the Animals Dataset by a number of criteria • filter the Animals Dataset by species, fixed status, and age
Student-facing Lesson Goals	• Let's learn how to start with one table and transform it into another.
Materials	• Lesson Slides (Google Slides) • "Function Purpose cards" which describe simple boolean functions to apply to students • Computer for each student (or pair), with access to the internet • Student workbook, and something to write with • All students should log into CPO and open the Table Methods Starter File • One copy of Function Cards printed and cut.
Preparation	• Make sure all materials have been gathered • Decide how students will be grouped in pairs
Supplemental Resources	
Language Table	See table below

Types	Functions	Values
Number	`num-sqrt, num-sqr`	`4, -1.2, 2/3`
String	`string-repeat, string-contains`	`"hello", "91"`
Boolean	`==, <, <=, >=, string-equal`	`true, false`
Image	`triangle, circle, star, rectangle, ellipse, square, text, overlay`	

Review Function Definitions

15 minutes

Overview

Students get some practice reading function definitions, and in the process they build knowledge that's needed later on in the lesson.

Launch

Let's see how much you remember about function definitions! Load the Table Methods Starter File, go to the File menu, and click "Save a Copy".

Investigate

Students complete Reading Function Definitions (Page 27) in their student workbooks.

Synthesize

Can students explain what each function does?

Ordering Tables

10 minutes

Overview

Students learn a second table method, which allows them to sort rows in ascending or descending order, according to one column.

Launch

Have students find the contract for `.order-by` in their contracts pages. The `.order-by` method consumes a String (the name of the column by which we want to order) and a Boolean (true for ascending, false for descending). But what does it produce?

Investigate

- Type `animals-table.order-by("name", true)` into the Interactions Area. What do you get?
- Type `animals-table.order-by("age", false)` into the Interactions Area. What do you get?
- Sort the animals table from heaviest-to-lightest.
- Sort the animals table alphabetically by species.
- Sort the animals table by how long it took for each animal to be adopted, in ascending order.

Synthesize

Answer any questions students may have. Class discussion: what do `.order-by` and `.row-n` have in common? How are they different?

Filtering Tables

20 minutes

Overview

Students learn how to *filter* tables, by removing rows.

Launch

Explain to students that you have "Function Cards", which describe the purpose statement of a function that consumes a Row from a table of students, and produces a Boolean (e.g. - "this student is wearing glasses"). Select a volunteer to be the "filter method", and have them *randomly choose* a Function Card, and make sure they read it without showing it to anyone else.

Have ~10 students line up in front of the classroom, and have the filter method go to each student and say "stay" or "sit" depending on whether their function would return true or false for that student. If they say "sit", the student sits down. If they say true, the student stays standing.

Ask the class: based on who sat and who stayed, *what function was on the card?*

> The `.filter` method takes a function, and produces a *new table* containing only rows for which the function returns `true`.

Suppose we want to get a table of only animals that have been fixed? Have students find the contract for `.filter` in their contracts pages. The `.filter` method is taking in a *function*. What is the contract for that function? Where have

we seen functions-taking-functions before?

Investigate

- In the Interactions Area, type `animals-table.filter(is-fixed)`. What did you get?
- What do you expect `animals-table` to produce, and why? Try it out. What happened?
- In the Interactions Area, type `animals-table.filter(is-old)`. What did you get?
- In the Interactions Area, type `animals-table.filter(is-dog)`. What did you get?
- In the Interactions Area, type `animals-table.filter(lookup-name)`. What did you get?

The `.filter` method walks through the table, applying whatever function it was given to each row, and producing a new table containing all the rows for which the function returned `true`. Notice that the Domain for `.filter` says that test must be a function (that's the arrow), which consumes a `Row` and produces a `Boolean`. If it consumes anything besides a single `Row`, or if it produces anything else besides a `Boolean`, we'll get an error.

Possible Misconceptions

Students often think that filtering a table *changes* the table. In Pyret, all table methods produce a *brand new table*. If we want to save that table, we need to define it. For example: `cats = animals-table.filter(is-cat)`.

Synthesize

Debrief with students. Some guiding questions on filtering:

- Suppose we wanted to determine whether cats or dogs get adopted faster. How might using the `.filter` method help?
- If the shelter is purchasing food for older cats, what filter would we write to determine how many cats to buy for?
- Can you think of a situation where filtering fixed animals would be helpful?

Building Columns

10 minutes

Overview

Students learn how to *build columns*, using the `.build-column` table method.

Launch

Suppose we want to *transform* our table, converting `pounds` to `kilograms` or `weeks` to `days`. Or perhaps we want to add a "cute" column that just identifies the puppies and kittens? Have students find the contract for `.build-column` in their contracts pages. The `.build-column` method is taking in a *function* and a *string*. What is the contract for that function?

Investigate

- Try typing `animals-table.build-column("old", is-old)` into the Interactions Area.
- Try typing `animals-table.build-column("sticker", label)` into the Interactions Area.
- What do you get? What do you think is going on?

The `.build-column` method walks through the table, applying whatever function it was given to each row. Whatever the function produces for that row becomes the value of our new column, which is named based on the string it was given. In the first example, we gave it the `is-old` function, so the new table had an extra Boolean column for every animal, indicating whether or not it was young. Notice that the Domain for `.build-column` says that the builder must be a function which consumes a `Row` and produces some other value. If it consumes anything besides a single `Row`, we'll get an error.

Synthesize

Debrief with students. Ask them if they think of a situation where they would want to use this. Some ideas:

- A dataset about school might include columns for how many students are in the school and how many pass the state exam. But when comparing schools of different sizes, what we really want is a column showing what *percentage* passed the exam. We could use `.build-column` to compute that for every row in the table.
- The animals shelter might want to print nametags for every animal. They could build a column using the `text` function to have every animal's name in big, purple letters.
- A dataset from Europe might list everything in metric (centimeters, kilograms, etc), so we could build a column to convert that to imperial units (inches, pounds, etc).

Additional Exercises:

What Table Do We Get?