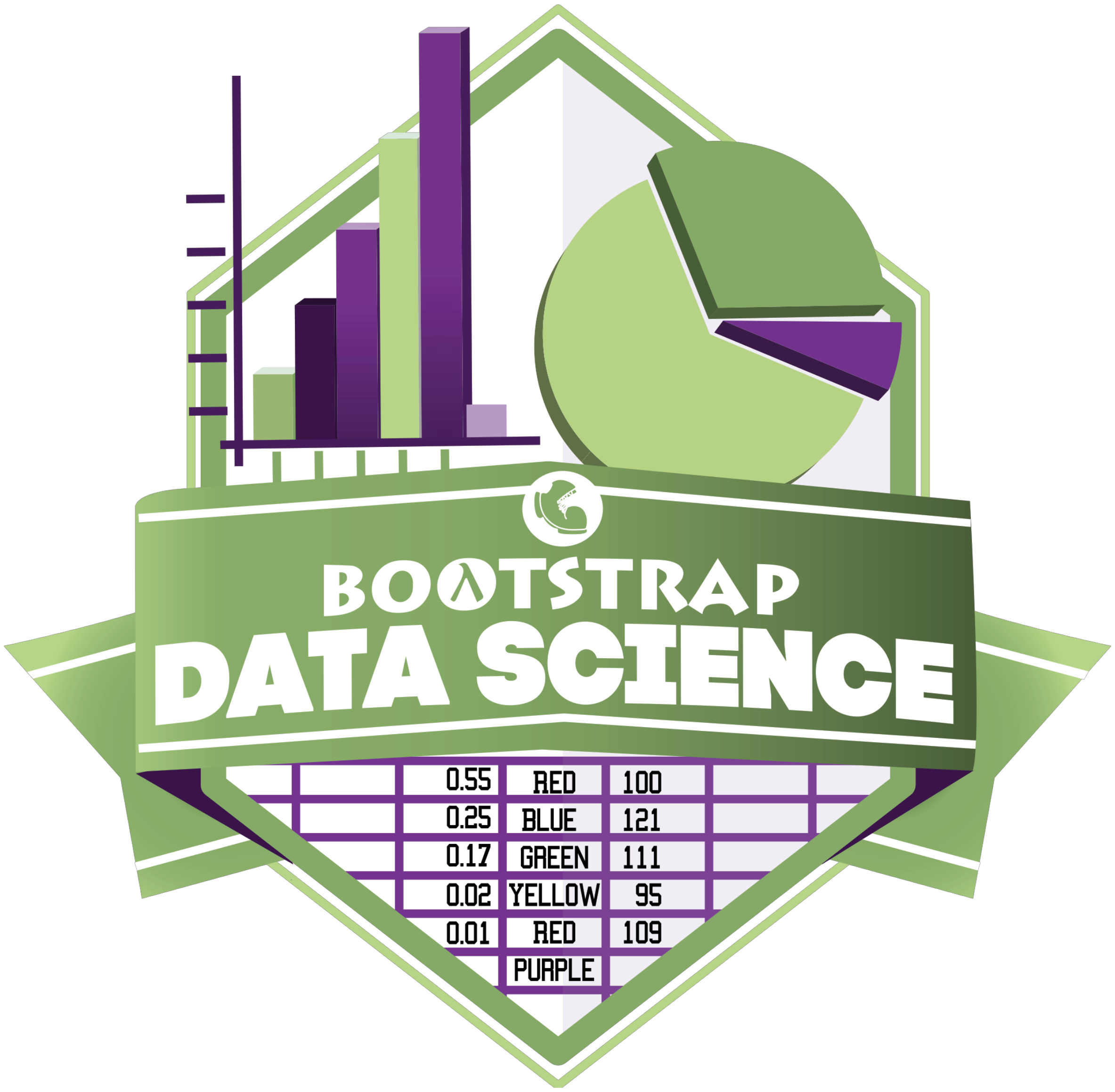


BOOTSTRAP

www.bootstrapworld.org

Workbook v1.2

Brought to you by the Bootstrap team:

- Emmanuel Schanzer
- Kathi Fisler
- Shriram Krishnamurthi
- Ed Campos
- Emma Youndtsmith
- Sam Dooman

Unit 1

Many important questions ("what's the best restaurant in town?", "is this law good for citizens?", etc.) are answered with data. Data Scientists try and answer these questions, by writing *programs that ask questions of data.*

Data of all types can be organized into **Tables**

- Every Table has a **header row**, and some number of **data rows**
- **Quantitative data** is data - usually numeric - that measures *quantity*, such as a person's height, a score on test, a measure of distance, etc. A list of quantitative data can be ordered from smallest to largest.
- **Categorical data** is data that specifies *categories*, such as eye color, country of origin, etc. A list of categorical data has no notion of "smallest" or "largest", and cannot be ordered.

Programming languages involves different *datatypes*, such as Numbers, Strings, Booleans and Images.

- **Operators** (like +, -, *, <, etc.) are written between values. For example: `4 + 2`
- We can use **functions** (like triangle, star, string-repeat, etc.) by writing the function name first, followed by a list of **arguments** in parentheses. For example: `star(50, "solid", "red")`
- **Methods** are special functions that are attached to pieces of data. We use them to manipulate Tables. They are different from functions in several ways:
    - Their names can't be used alone: they can only be used as part of data, separated by a dot. (For example, `shapes.row-n(2)`)
    - Their contracts are different: they include the type of the data as part of their names. (eg, `<table>.row-n :: (index :: Number) → Row`)
    - They have a "secret" argument, which is the data they are attached to
- In this course, we will use three **Table Methods** to manipulate our datasets:
    - `<Table>.order-by` – order the rows of a table based on a column
    - `<Table>.filter` – create a **subset** of the data, with only certain rows
    - `<Table>.build-column` – use the columns of a table to make a new one

Numbers and Strings

Make sure you've loaded the Unit 1 Starter File, and clicked "Run".

1. Try typing `42` into the Interactions Area and hitting "Enter". What happens?

2. Try typing in other Numbers. What happens if you try a decimal like `0.5`? A fraction like `1/3`? Try really big Numbers, and really small ones.

3. String values are always in quotes. Try typing your name (in quotes!). What happens when you hit "Enter"?

4. Try typing your name with the opening quote, but *without* the closing quote. What happens? Now try typing it without *any* quotes.

5. Is `42` the same as `"42"`? Why or why not? Write your answer below:

They are different data types: 42 (without quotes) is a Number, and "42" (with quotes) is a string.

Operators

6. Just like in math, Pyret has *operators* like `+` and `-`. Try typing in `4 + 2`, and then `4+2` (without the spaces). What can you conclude from this? Write your answer below:

Operators (like +) need whitespace separating them from their operands.

7. Typing in the following expressions, one at a time: `4 + 2 + 6`, `4 + 2 * 6`, and `4 + (2 * 6)`. What do you notice? Write your answer below:

You can use the same operator multiple times without parentheses, but you need parentheses to group order of operations if using different operators (like + and *) together.

8. Try typing in `4 + "cat"`, and then `"dog" + "cat"`. What can you conclude from this? Write your answer below:

The + operator can only be used with Numbers, not Strings.

Booleans

Boolean expressions are yes-or-no questions, and will always evaluate to either `true` ("yes") or `false` ("no"). What will each of the expressions below evaluate to? Write down the result in the blanks provided, and type them into Pyret if you're not sure.

`3 <= 4`	True	`"a" > "b"`	False
`3 == 2`	False	`"a" <> "b"`	True
`2 <> 4`	True	`"a" == "b"`	False
`3 <> 3`	True	`"a" <> "a"`	False

Boolean Operators

Pyret also has operators that work on *Booleans*. For each expression below, *write down your guess* about what it will evaluate to. Then type them in and see if you were right!

`(3 <= 4) and (3 == 2)`	False
`("a" == "b") and (3 <> 4)`	False
`(3 <= 4) or (3 == 2)`	True
`("a" == "b") or (3 <> 4)`	True

1. How many different Number values are there in Pyret? Infinite
2. How many different String values are there in Pyret? Infinite
3. How many different Boolean values are there in Pyret? Two

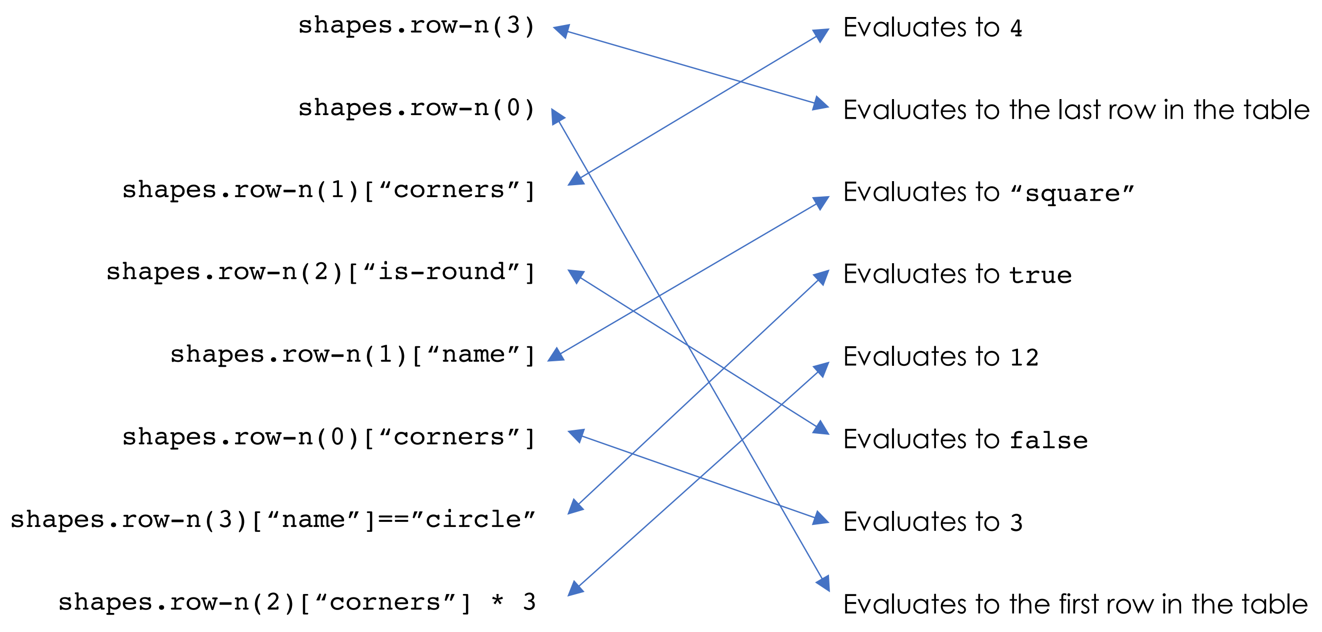
Lookups

The table below represents four shapes in a table:

shapes

name	corners	is-round
"triangle"	3	false
"square"	4	false
"rectangle"	4	false
"circle"	0	true

1. ***Match*** each Pyret expression (left) to the description of what it looks up(right).



`shapes.row-n(3)`	Evaluates to `4`
`shapes.row-n(0)`	Evaluates to the last row in the table
`shapes.row-n(1)["corners"]`	Evaluates to `"square"`
`shapes.row-n(2)["is-round"]`	Evaluates to `true`
`shapes.row-n(1)["name"]`	Evaluates to `12`
`shapes.row-n(0)["corners"]`	Evaluates to `false`
`shapes.row-n(3)["name"]=="circle"`	Evaluates to `3`
`shapes.row-n(2)["corners"] * 3`	Evaluates to the first row in the table

2. Fill in the blanks (left) with the Pyret lookup code that will produce the value (right).

a. *shapes.row-n(2)["name"]*	`"rectangle"`
b. *shapes.row-n(0)["name"]*	`"triangle"`
c. *shapes.row-n(1)["corners"]*	`4`
d. *shapes.row-n(3)["corners"]*	`0`
e. *shapes.row-n(3)["is-round"]*	`true`

Unit 2

Answering Questions from Data can take many forms. Here are a few types of questions, each requiring a different kind of analysis:

- **Lookup Questions** can be answered just by finding the right row and column a table. (e.g. – “How old is Toggle?”)
- **Compute Questions** can be answered by computing over a single row or column. (e.g. – “What is the heaviest animal at the shelter?”)
- **Analyze Questions** require looking for trends across multiple rows or columns. (e.g. – “Do cats tend to be adopted sooner than dogs?”)

We can **define our own functions**, using a technique called the **Design Recipe**.

- We use the Design Recipe to help us define functions **without making mistakes**.
- The first step is to write a **Contract** and **Purpose Statement** for the function, which specify the Name, Domain and Range of the function and give a summary of what it does.
- The second step is to **write at least two examples**, which show how the function should work for specific inputs. These examples help us see patterns, and we express those patterns by **circling and labeling** what changes.
- The final step is to **define the function**, which generalizes our examples.

The `Animals` Dataset

1. This dataset is Animals from an animal shelter, which contains 31 data rows.
2. Some of the columns are:
   i. name, which contains categorical data, and is of type String. Some example values from this column are: "Toggle", "Fritz", and "Nori".
   ii. species, which contains categorical data, and is of type String. Some example values from this column are: "cat", "dog".
   iii. age, which contains quantitative data, and is of type Number. Some example values from this column are: 1, 2, 6.
   iv. pounds, which contains quantitative data, and is of type Number. Some example values from this column are: 6.5, 35.3, 6.1.
3. Some questions I have about this dataset:

My question is…	Lookup, Compute or Analyze?

Practicing Lookups

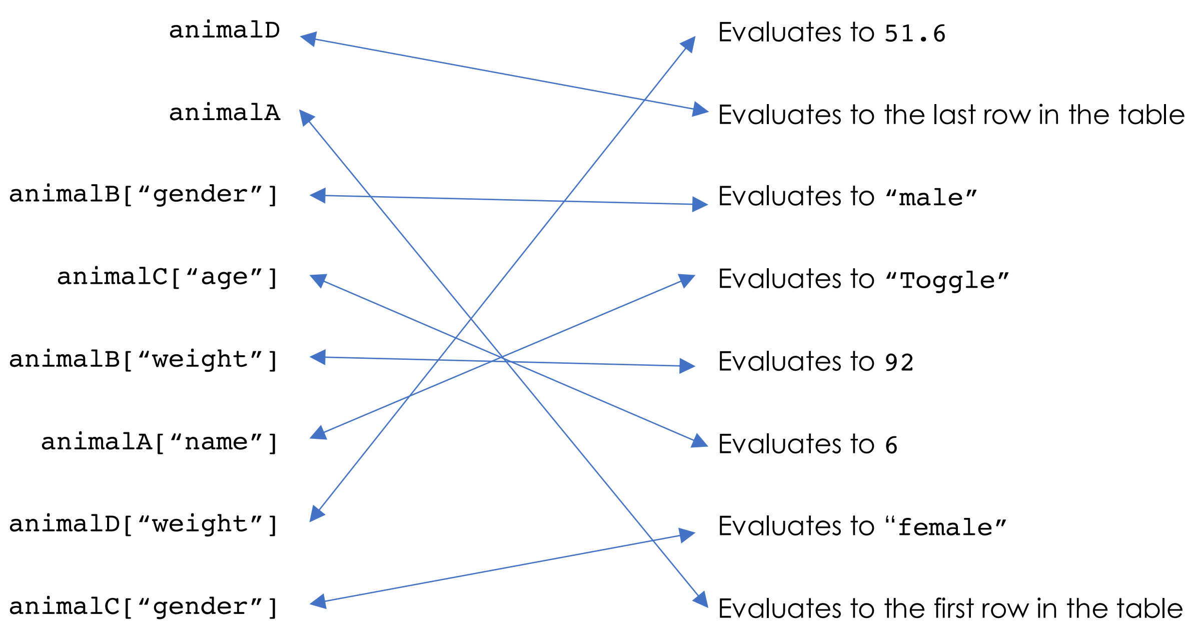
The table below represents four pets at an animal shelter, and four value definitions for rows in that table:

animals-table

name	gender	age	Weight
"Toggle"	"female"	3	48
"Fritz"	"male"	4	92
"Nori"	"female"	6	35.3
"Maple"	"female"	3	51.6

```
animalA = animals-table.row-n(0)
animalB = animals-table.row-n(1)
animalC = animals-table.row-n(2)
animalD = animals-table.row-n(3)
```

v. Match each Pyret expression (left) to the description of what it looks up(right).



`animalD`	Evaluates to `51.6`
`animalA`	Evaluates to the last row in the table
`animalB["gender"]`	Evaluates to `"male"`
`animalC["age"]`	Evaluates to `"Toggle"`
`animalB["weight"]`	Evaluates to `92`
`animalA["name"]`	Evaluates to `6`
`animalD["weight"]`	Evaluates to `"female"`
`animalC["gender"]`	Evaluates to the first row in the table

vi. Fill in the blanks (left) with the Pyret lookup code that will produce the value (right).

animalD["name"]	`"Maple"`
`animalB["gender"]`	`"male"`
`animalB["age"]`	`4`
`animalA["weight"]`	`48`
`animalC["name"]`	`"Nori"`

The Design Recipe

For the word problems below, assume you have `animalA` and `animalB` defined in your code.

Define a function called `is-fixed`, which looks up whether or not an animal is fixed

```
# is-fixed :: (animal :: Row) → Boolean
  name        domain              range
# Consumes an animal, and looks up the value in the fixed column
examples:
    is-fixed ( animalA ) is animalA["fixed"]
    is-fixed ( animalB ) is animalB["fixed"]
end
fun is-fixed ( animal ) : animal["fixed"]
end
```

Define a function called `gender`, which consumes a Row of the animals table and looks up the gender of that animal

```
# gender :: (animal :: Row) → String
  name      domain              range
# Consumes an animal, and produces the value in the gender column
examples:
    gender ( animalA ) is animalA["gender"]
    gender ( animalB ) is animalB["gender"]
end
fun gender ( animal ) : animal["gender"]
end
```

The Design Recipe

For the word problems below, assume you have `animalA` and `animalB` defined in your code.

Define a function called `is-cat`, which consumes a Row of the animals table and computes whether the animal is a cat.

```
# is-cat :: (animal :: Row) → Boolean
  name      domain            range
# Consumes an animal, look up the species column, and computer if species = "cat"
examples:
    is-cat ( animalA ) is animalA["species"] == "cat"
    is-cat ( animalB ) is animalB["species"] == "cat"
end
fun is-cat ( animal ) : animal["species"] == "cat"
end
```

Define a function called `is-young`, which consumes a Row of the animals table and computers whether it is less than four years old.

```
# is-young :: (animal :: Row) → Boolean
  name        domain            range
# Consumes an animal, returns true if the animal is less than 4 years old
examples:
    is-young ( animalA ) is animalA["age"] < 4
    is-young ( animalB ) is animalB["age"] < 4
end
fun is-young ( animal ) : animal["age"] < 4
end
```

Unit 3

Functions can contain value definitions

We use **Table Plans** to help us use table methods correctly, without making mistakes:

- Like functions, we start with a Contract and Purpose Statement
- But instead of writing *programmed examples*, we sketch out **Sample Tables** and **Results**, based on the Contract and Purpose.
- Then we define the function based on our Sample Table and Result. Every function includes both the table definition (using methods) and a table expression.

Design Recipe

For the word problems below, assume you have `animalA` and `animalB` defined in your code.

Define a function called `birth-year`, which consumes a Row of the animals table and produces the year that animal was born.

# *birth-year* :: *(animal :: Row)* → *Number*

name — domain — range

#*Consumes an animal, and produces the year that they were born, subtracting age from the current year*

examples:

birth-year (animalA) **is** 2019 - animalA["age"]

birth-year (animalB) **is** 2019 - animalB["age"]

end

fun birth-year (animal) **:** 2019 - animal["age"]

end

Define a function called `nametag`, prints out each animal's name in big red letters.

# *nametag* :: *(animal :: Row)* → *Image*

name — domain — range

# *Consumes an animal, and produces an image of their name in big, red letters*

examples:

nametag (animalA) **is** text(animalA["name"], 50, "red")

nametag (animalB) **is** text(animalB["name"], 50, "red")

end

fun nametag (animal) **:** text(animal["name"], 50, "red")

end

Playing with Methods

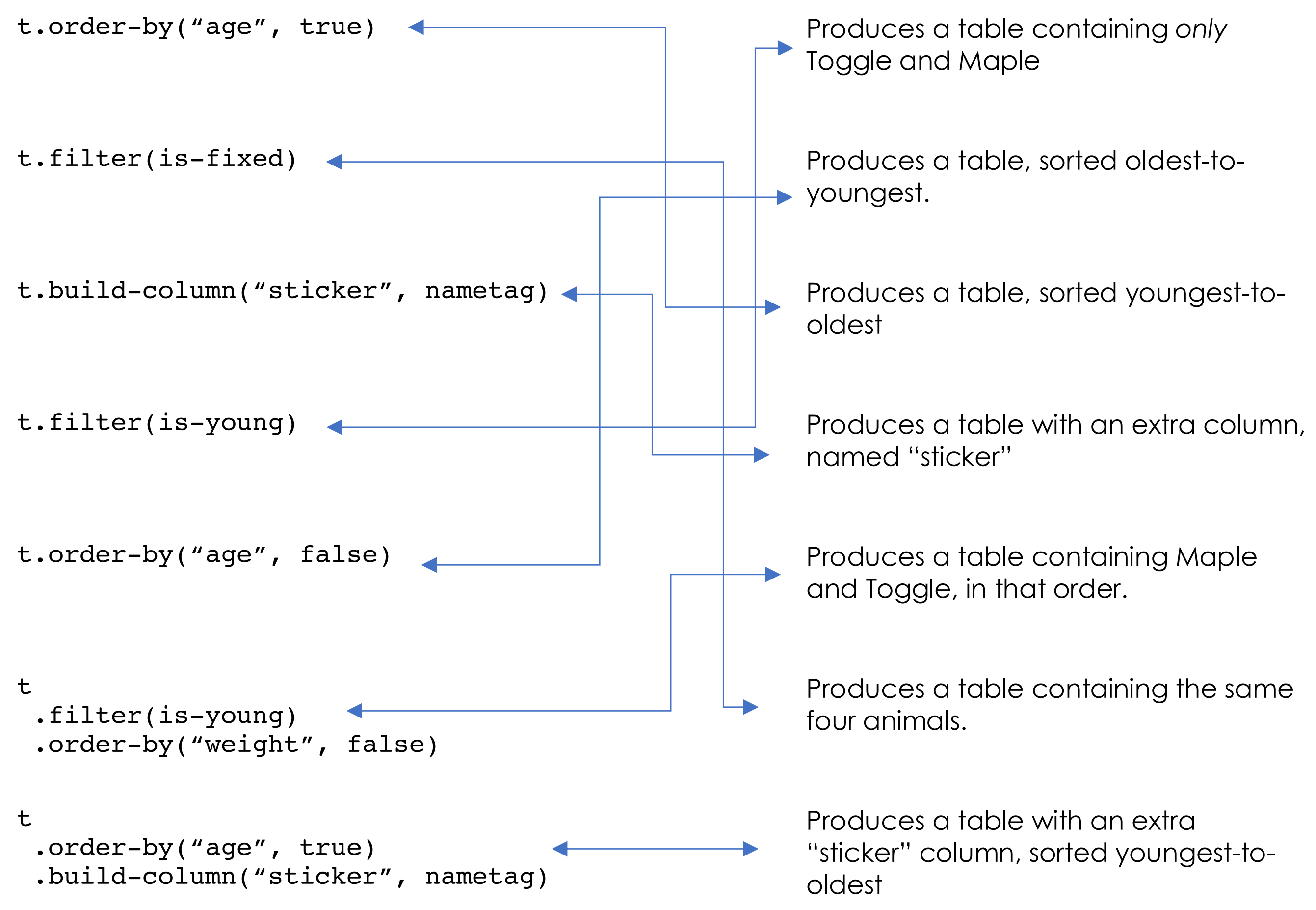
You have the following functions defined below (read them *carefully*!):

```
fun is-fixed(animal): animal[“fixed”]                    end
fun is-young(animal): animal[“age”] < 4                  end
fun nametag(animal):  text(animal["name"], 20, "red") end
```

The table **t** below represents four animals at the shelter:

name	gender	age	fixed	weight
“Toggle”	“female”	3	true	48
“Fritz”	“male”	4	true	92
“Nori”	“female”	6	true	35.3
“Maple”	“female”	3	true	51.6

Match each Pyret expression (left) to the description of what it does (right).



```
t.order-by(“age”, true)

t.filter(is-fixed)

t.build-column(“sticker”, nametag)

t.filter(is-young)

t.order-by(“age”, false)

t
 .filter(is-young)
 .order-by(“weight”, false)

t
 .order-by(“age”, true)
 .build-column(“sticker”, nametag)
```

Produces a table containing *only* Toggle and Maple

Produces a table, sorted oldest-to-youngest.

Produces a table, sorted youngest-to-oldest

Produces a table with an extra column, named “sticker”

Produces a table containing Maple and Toggle, in that order.

Produces a table containing the same four animals.

Produces a table with an extra “sticker” column, sorted youngest-to-oldest

Table Plan

The shelter wants to print up bar charts showing young animal's ages, in alphabetical order. Sometimes they want to do this for *every* animal, but sometimes they just need it for the cats, or for animals that are fixed.
Define a function `sorted-age-bar`, which takes in a table of animals and computes a bar-chart showing their ages (in alphabetical order), *for only the young animals*.

Contract and Purpose

# *sorted-age-bar* **::** *(animals :: Table)* → *Image*

# *Consume a table of animals, and compute a bar chart showing their ages, in alphabetical order*

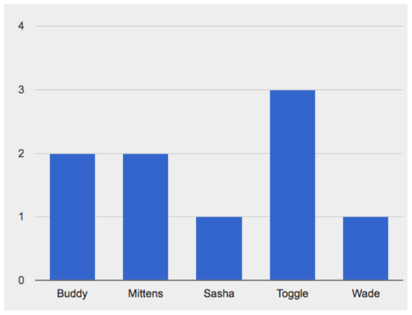
Where I start, what I type, and what I get back

An example table to start with: *To use the function, I would type:*

`example-table` `sorted-age-bar(example-table)`

name	...	age
Sasha		1
Toggle		3
Buddy		2
Wade		1
Mittens		2

→



Define the function

Use the relevant methods (circle your helper functions!), then produce a result with the new table.

```
fun  sorted-age-bar ( animals ) :
    t = animals
     .build-column(                    )
     .filter(                          )
     .order-by(   "age", true          )
    bar-chart( t, "name", "age" )
end
```

Define the table
Are there more columns?
Are there fewer rows?
Are the rows ordered?
Produce the result

Table Plan

The shelter wants to see if there's a relationship between how old an animal is, and how long it takes them to be adopted. Sometimes they want to do this for *every* animal, but sometimes they just need it for the cats, or for animals that are young. Define a function `age-adopted-scatter`, which takes in a table of animals and computes a scatter-plot showing only the fixed animals, with their ages on the x-axis and weeks to be adopted on the y-axis.

Contract and Purpose

# *age-adopted-scatter* **::** *(animals :: Table)* → *Image*

# *Consume a table of animals, and compute a scatterplot showing their ages on the x-axis, and weeks be adopted on the y-axis*

Where I start, what I type, and what I get back

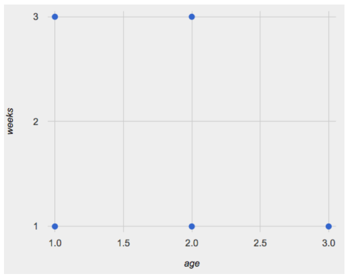
A sample table to start with:

name	...	age	weeks
Sasha		1	3
Toggle		3	1
Buddy		2	3
Wade		1	1
Mittens		2	1

→

To use the function, I would type:

`age-adopted-scatter(sample)`



Define the function

Use the relevant methods (circle your helper functions!), then produce a result with the new table.

```
fun age-adopted-scatter ( animals ) :
    t = animals                                        Define the table
      .build-column(                            )      Are there more columns?
      .filter(                                  )      Are there fewer rows?
      .order-by(                                )      Are the rows ordered?
    scatter-plot( t, "name", "age", "weeks" )          Produce the result
end
```

Unit 4

Bar charts show the *absolute* quantity of each row in a dataset. The larger the quantity, the longer the bar. Bar charts provide a visual representation of values in a dataset.

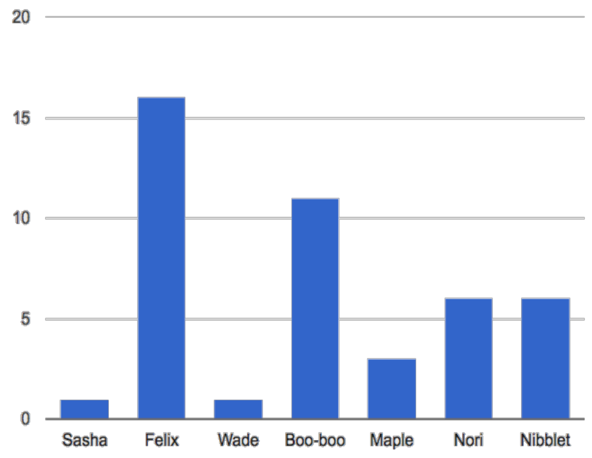
Pie charts show the *relative* quantity of each row in a dataset. The greater the percentage, the larger the pie slice. Pie charts provide a visual representation of proportions in a dataset.

Choosing a Sample Table is important when coming up with small examples for Table Plans. A good sample table has:

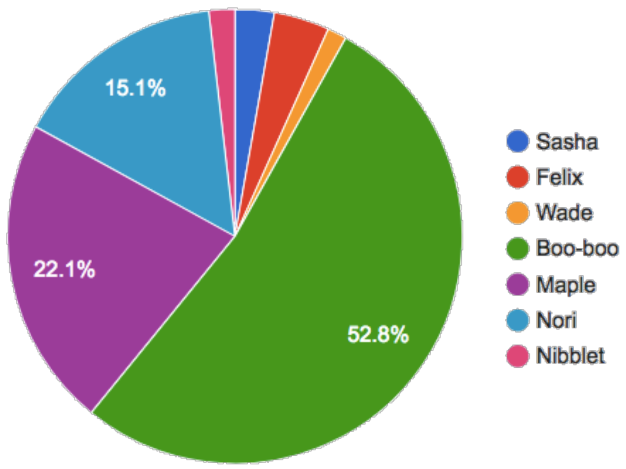
- At least all the relevant columns
- Enough rows to accurately represent the dataset
- Rows that are randomly-ordered

Quantity Charts in the `Animals` Dataset

Below are two **quantity charts** made from subsets of the animals table



Animals Ages (yrs)



Animals Weights (lbs)

What do you NOTICE about these charts?	What do you WONDER about these charts?

Why are some questions easier to answer with one kind of chart or another?

Bad Sample Tables!

For each word problem, a Sample Table must have (1) all the columns that matter, (2) a representative sample of the rows, and be in (3) random order. For each problem below, check the boxes if the Sample Table meets those criteria.

1. The shelter wants to a scatter plot showing the age of the cats v. their weight

name	species	age	fixed	legs	pounds	weeks
Sasha	cat	1	FALSE	4	6.5	3
Mittens	cat	2	TRUE	4	7.4	5
Sunflower	cat	5	TRUE	4	8.1	10

- ✓ Relevant columns
- ✓ Representative sample of rows
- ✓ Random order

2. The shelter wants a pie chart showing all the dogs' weight

name	species	age
Fritz	dog	4
Wade	cat	2
Nibblet	rabbit	6
Daisy	dog	5

- ☐ Relevant columns
- ☐ Representative sample of rows
- ✓ Random order

3. Sort all the animals alphabetically by name

name	species	age	fixed	legs	pounds	weeks
Ada	dog	2	TRUE	4	32	3
Bo	dog	4	TRUE	4	76.1	10
Boo-boo	dog	11	TRUE	4	123	10

- ✓ Relevant columns
- ☐ Representative sample of rows
- ☐ Random order

4. Make a bar chart for all the fixed animals

name	species	age	fixed	legs	pounds	weeks
Sasha	cat	1	FALSE	4	6.5	3

- ✓ Relevant columns
- ☐ Representative sample of rows
- ☐ Random order

Table Plan

Define a function `pie-pounds-young`, which takes in a Table of animals and creates a pie chart of the animals' weight, but only for animals that are young.

Contract and Purpose

# *pie-pounds-young* :: *(animals :: Table)* → *Image*

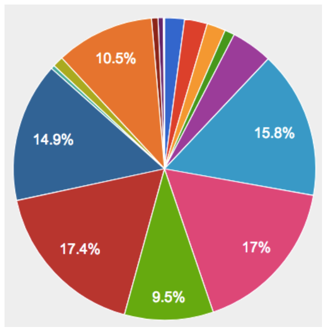
# *Consumes a table of animals, filters to show only young animals, and produces a pie chart of their weight*

Where I start, what I type, and what I get back

A sample table to start with: *To use the function, I would type:*

sample-table → *pie-pounds-young(sample-table)*

name	age	pounds
Snowcone	…	6.1
Lucky	…	45.4
Hercules	…	13.4
Toggle	…	48
Snuggles	…	0.1



Define the function

Use the relevant methods (circle your helper functions!), then produce a result with the new table.

```
fun  pie-pounds-young  ( animals ) :
  t = animals

   .filter(is-young)

  pie-chart(t, "name", "pounds")
end
```

Define the table

Are there more columns?

Are there fewer rows?

Are the rows ordered?

Produce the result

My Dataset

1. This dataset is ______________________________, which contains _________ data rows.
2. Some of the columns are:

   i. ______________________________, which contains ___________________________ data, and is of type ________________. Some example values from this column are: _______________________________.

   ii. ______________________________, which contains ___________________________ data, and is of type ________________. Some example values from this column are: _______________________________.

   iii. ______________________________, which contains ___________________________ data, and is of type ________________. Some example values from this column are: _______________________________.

   iv. ______________________________, which contains ___________________________ data, and is of type ________________. Some example values from this column are: _______________________________.

3. Some questions I have about this dataset:

My question is…	**Lookup, Compute or Analyze?**

My Dataset

What are two ways you might want to *order* this dataset?

1) ____

2) ____

What are two subsets into which you might *filter* this dataset?

1) ____

2) ____

What are two new columns you might want to *build* from this dataset?

1) ____

2) ____

Design Recipes – Filtering Rows

What are two criteria you might want to *filter* by? Write your own word problems below, and solve them using the Design Recipe.

Define a function called ______________________________ , which consumes a Row of the ____________________________ table and ____ ____

```
# ______________________::_______________________________ → _____________________
          name                         domain                        range
# _________________________________________________________________________________
examples:

      _____________(__________) is ________________________________

      _____________(__________) is ________________________________
end
fun _____________ (________) : ________________________________

end
```

```
# ______________________::_______________________________ → _____________________
          name                         domain                        range
# _________________________________________________________________________________
examples:

      _____________(__________) is ________________________________

      _____________(__________) is ________________________________
end
fun _____________ (________) : ________________________________

end
```

Design Recipes – Building Columns

What are two columns you might want to *build* for your dataset? Write your own word problems below, and solve them using the Design Recipe.

```
# _________________::________________________ → _________________
        name                  domain                   range
# ______________________________________________________________
examples:

     _____________(__________) is ________________________________

     _____________(__________) is ________________________________
end
fun  _____________  (________) :  ________________________________

end
```

```
# _________________::________________________ → _________________
        name                  domain                   range
# ______________________________________________________________
examples:

     _____________(__________) is ________________________________

     _____________(__________) is ________________________________
end
fun  _____________  (________) :  ________________________________

end
```

Quantity Charts in My Dataset

Describe two of the pie or bar charts you made from your dataset.

1) I made a ____________________ chart, showing the ________________________ for
pie / bar column in your dataset

____.
your subset (for example, "fixed dogs at the shelter")

2) I made a ____________________ chart, showing the ________________________ for

____.

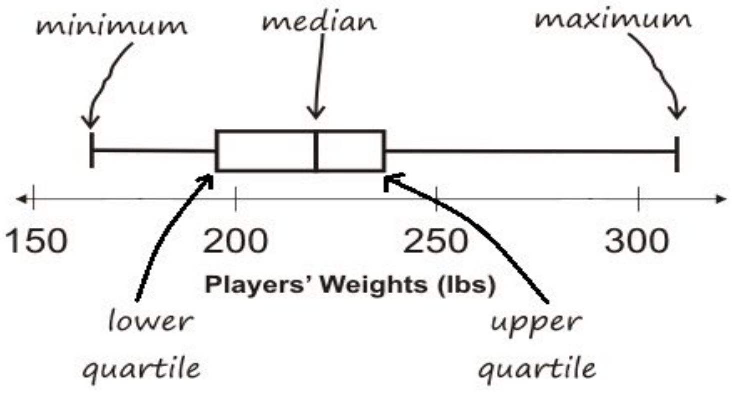
What do you NOTICE about these charts?	What do you WONDER about these charts?

Unit 5

- There are three ways to measure the "center" of a dataset, to talk about a whole column of data using just one number:
    - The **mean** of a dataset is the average of all the numbers
    - The **median** of a dataset is a value that is smaller than half the dataset, and larger than the other half
    - The **modes** of a dataset are the numbers that appear the most often.

- Data Scientists can also measure the "variation" of a dataset using a **five number summary:**
    - The **minimum** – the smallest value in the dataset
    - The **first, or "lower" quartile (Q1)** – the median value that separates the first quarter of the values in the dataset from the second quarter
    - The **second quartile (Q2)** – the median value which separates the entire dataset into "top" and "bottom" halves.
    - The **third, or "upper" quartile (Q3)** – the median value that separates the third quarter of the values in the dataset from the fourth quarter
    - The **maximum** – the largest value in the dataset

- The **five number summary** can be used to draw a **box-and-whisker plot.**



Summarizing Columns in `Animals`

1) The column I choose to measure is weeks

Measures of Center

The three measures for this column are:

Mean (Average)	Median	Mode(s)
6.0689	4	1

2) Since the mean is higher than the median, this suggests that there may
[higher/lower]

be outliers repesenting a few animals who took a long time to be adopted.
[explain your outliers!]

Measures of Variation

My five-number summary is:

Minimum	Q1	Q2 (Median)	Q3	Maximum
1	2.5	4	8	30

A box plot can be drawn from this summary on the number line below:

From this summary and box-plot, I conclude:

The vast majority of animals are adopted before 8 weeks in the shelter, but there are a number of outliers (such as the maximum of 30).

Interpreting Variation

Consider the following list dataset, representing the annual income of ten people:

```
$65k,  $12k,  $14k,  $280k,  $15k,  $22k,  $45k,  $34k,  $45k,  $175k
```

1. In the space below, rewrite this dataset in **sorted order**.

```
$12k,  $14k,  $15k,  $22k,  $34k,  $45k,  $45k,  $65k,  $175k,  $280k
```

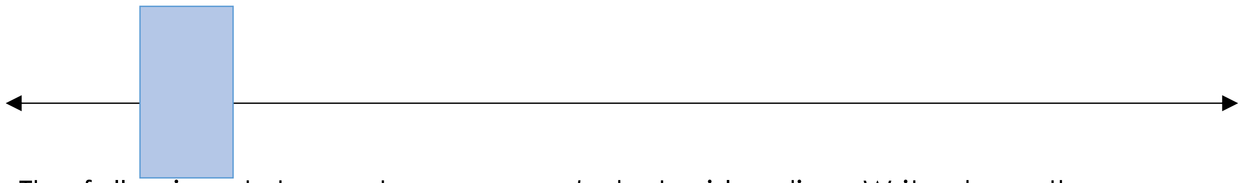
2. In the table below, compute the **measures of center** for this dataset.

Mean (Average)	Median	Mode(s)
70,700	39,500	45,000

3. In the table below, compute the **five number summary** of this dataset.

Minimum	Q1	Q2 (Median)	Q3	Maximum
12,000	15,000	39,500	65,000	280,000

4. On the number line below, draw a **box plot** for this dataset.



5. The following statements are *correct*…but misleading. Write down the reason why.

Statement	Why it's misleading
"They're rich! The average person makes more than $70k dollars!"	While the mean is close to $70k, there are some very high earning outliers pushing the average up.
"It's a middle-income list: the most common salary is $45k/yr!"	In the full dataset, more than half of the entries are people making less than $45k, making the mode misleading.
"This group is really diverse, with people making as little as 12k and as much as $280k!"	While the spread of incomes is large, the vast majority are still making less than $65k, with very high earning outliers.

Table Plan

The Animal Shelter Bureau would like to study the distribution of weeks-until-adoption for fixed animals housed at shelters around the country. They need a function that consumes a table of animals, filters to show only the fixed animals, and produces a box-plot for the `weeks` column. Define a function called `fixed-weeks-box` below.

Contract and Purpose

# *fixed-weeks-box* **::** *(animals :: Table)* → *Image*

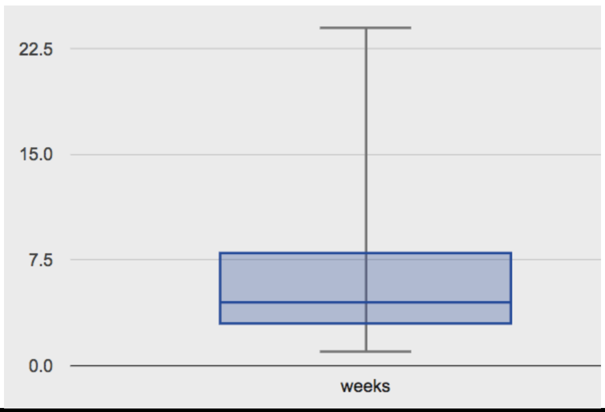
# *Consumes a table of animals, filters only the fixed animals, and produces a box plot of their weeks until adoption*

Where I start, what I type, and what I get back

A sample table to start with: *To use the function, I would type:*

sample table → *fixed-weeks-box(sample table)*

name	species	age	fixed	legs	weight	weeks
Snowcone	cat	2	TRUE	4	6.1	5
Lucky	dog	3	TRUE	3	45.4	9
Hercules	cat	3	FALSE	4	13.4	7
Toggle	dog	3	TRUE	4	48	3
Snuggles	tarantula	2	FALSE	8	0.1	1



Define the function

Use the relevant methods (circle your helper functions!), then produce a result with the new table.

```
fun  fixed-weeks-box  ( animals ) :
  t = animals-table
  
  .filter(  is-fixed  )
  
  box-plot( t, "weeks" )
end
```

Define the table

Are there more columns?

Are there fewer rows?

Are the rows ordered?

Produce the result

Summarizing a Column in My Dataset

The column I choose to measure is ______________________________

Measures of Center

The three measures for this column are:

Mean (Average)	Median	Mode(s)

3) Since the mean is ________________ than the median, this suggests that there may
`[higher/lower]`

be outliers repesenting____.
`[explain your outliers!]`

Measures of Variation

My five-number summary is:

Minimum	Q1	Q2 (Median)	Q3	Maximum

A box plot can be drawn from this summary on the number line below:

From this summary and box-plot, I conclude:

Unit 6

Frequency Bar charts show the number of rows belonging to a given category. The more rows in each category, the longer the bar.

- *Frequency bar charts provide a visual representation of the frequency of values in a* ***categorical*** *column.*

- Since categorical data cannot be ordered, there is no strict ordering of bars in a frequency bar chart.

Histograms show the number of rows that fall within certain ranges, or "bins" of a dataset. The more rows that that fall within a particular "bin", the longer the bar.

- *Histograms provide a visual representation of the frequency of values in a* ***quantitative*** *column.*

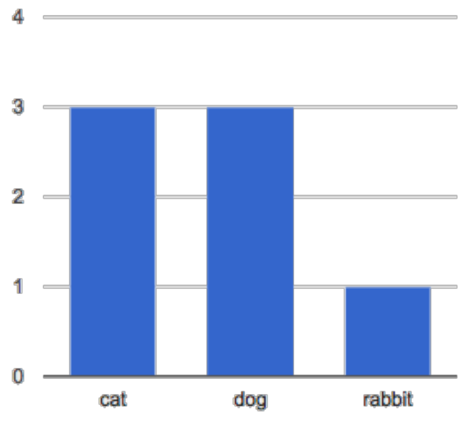
- Quantitative data can be ordered, so the bars of a histogram are always sorted.

- When dealing with histograms, it's important to select a good **bin size**. If the bins are too small or too large, it is difficult to see the distribution in the dataset.

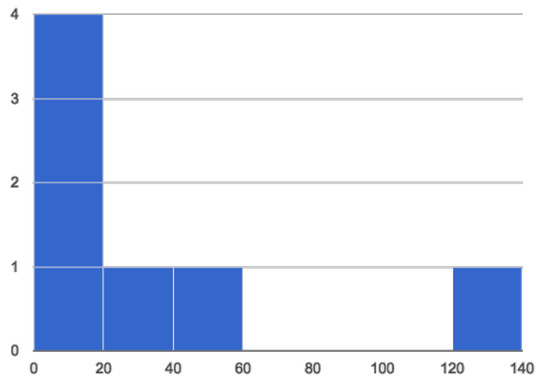
Frequency Charts in the `Animals` Dataset

name	species	age	pounds
"Sasha"	"cat"	1	6.5
"Boo-boo"	"dog"	11	123
"Felix"	"cat"	16	9.2
"Nori"	"dog"	6	35.3
"Wade"	"cat"	1	3.2
"Nibblet"	"rabbit"	6	4.3
"Maple"	"dog"	3	51.6

1. How many cats are there? 3
2. How many dogs are there? 3
3. How many animals are between 3-6 years old? 3
4. How many weigh between 0-5 pounds? 2
5. Are there more animals weighing 0-5 than 6-10 pounds? Yes
6. The charts below are based on the Sample Table above. What is each one measuring? Write down your guess underneath each one.



Amount of each species



Frequency of animal weights

Table Plan

Define a function `freq-bar-gender`, which takes in a Table of animals and creates a frequency bar chart showing how many _fixed_ animals are male v. female.

Contract and Purpose

freq-bar-gender :: (animals :: Table) → Image

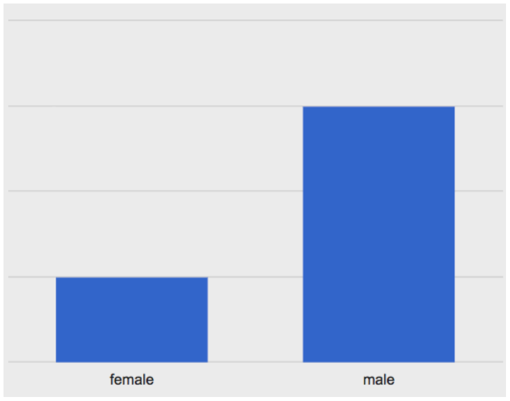
Consumes a table of animals and produces a frequency bar chart of their genders, for *fixed* animals

Examples

Make a Start Table and a result based on that table.

animals-table → freq-bar-gender(animals-table)

name	species	age	gender
Fritz	dog	4	male
Wade	cat	2	male
Nibblet	rabbit	6	male
Daisy	dog	5	female



Define the function

Use the relevant methods (circle your helper functions!), then produce a result with the new table.

```
fun freq-bar-gender ( animal ) :
    t = animals
    
    .filter(is-fixed)
    freq-bar-chart(t, "gender")
    fre
end
```

Define the table

Are there more columns?

Are there fewer rows?

Are the rows ordered?

Produce the result

Table Plan

Define a function `histogram-cats-adoption`, which takes in a Table of animals and creates a histogram showing how long it took for cats in the dataset to get adopted

Contract and Purpose

histogram-adoption :: (animals :: Table) → Image

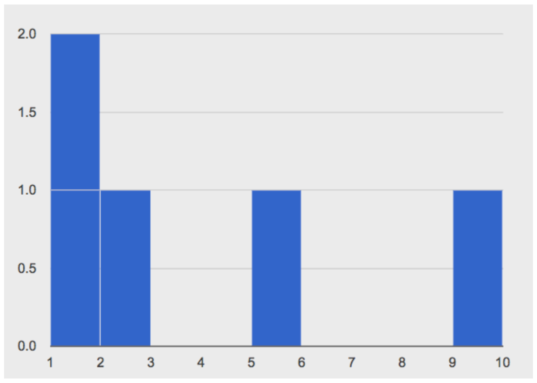
Consumes a table of animals and produces a histogram showing how long it took for the cats to get adopted

Examples

Make a Start Table and a result based on that table.

animals-table → histogram-adoption(animals-table)

name	species	age	fixed	legs	weight	weeks
Snowcone	cat	2	TRUE	4	6.1	5
Lucky	dog	3	TRUE	3	45.4	9
Hercules	cat	3	FALSE	4	13.4	7
Toggle	dog	3	TRUE	4	48	3
Snuggles	tarantula	2	FALSE	8	0.1	1



Define the function

Use the relevant methods (circle your helper functions!), then produce a result with the new table.

```
fun histogram-adoption ( animals ) :
    t = animals
    
    .filter(is-cat)
    
    histogram(t, "weeks", 1)
end
```

Define the table

Are there more columns?

Are there fewer rows?

Are the rows ordered?

Produce the result

Visualizing My Dataset

Describe two of the histograms or frequency bar charts you made from your dataset.

1) I made a ______________________________, showing the ______________________ for
histogram / frequency bar chart — column in your dataset

____.
your subset (for example, "fixed dogs at the shelter"

2) I made a ______________________________, showing the ______________________ for

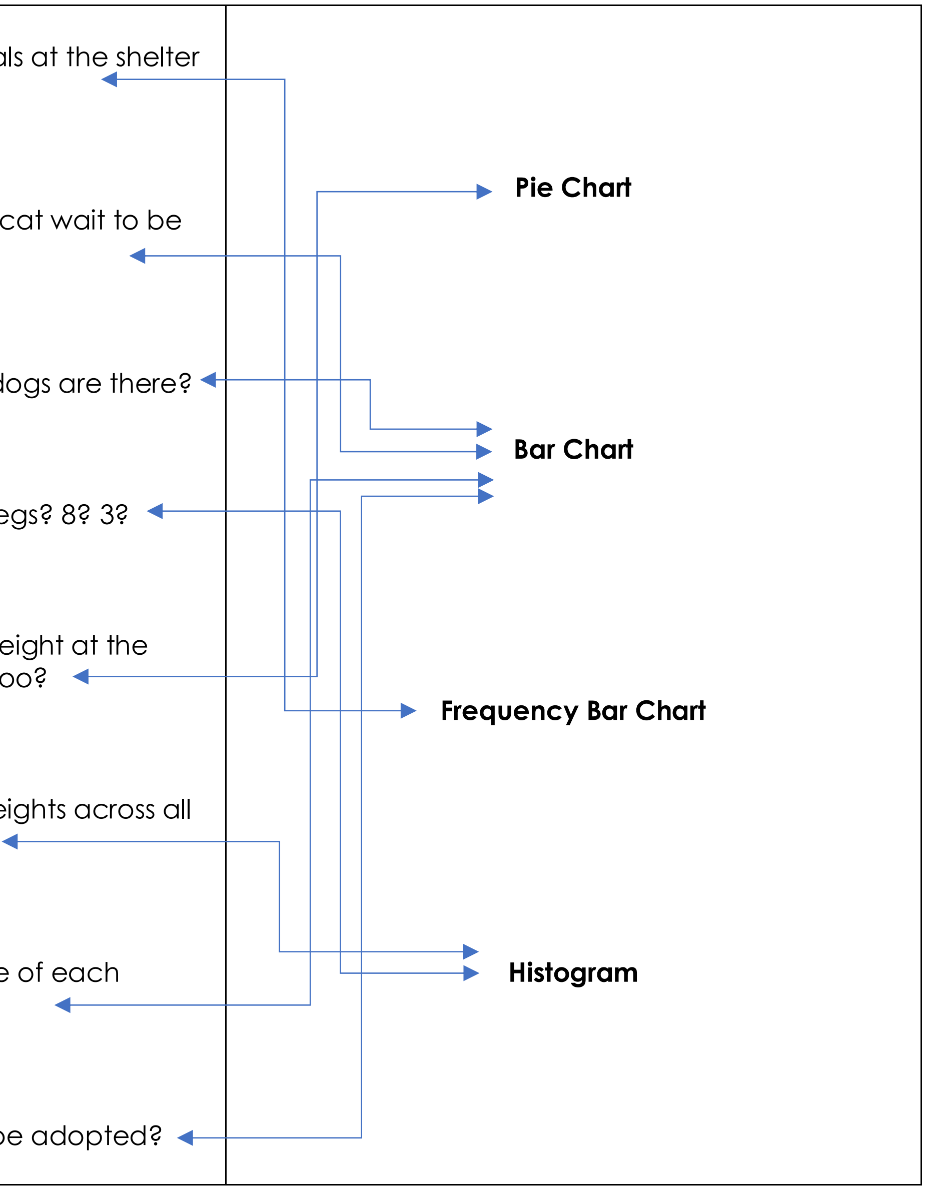
____.

What do you NOTICE about these charts?	What do you WONDER about these charts?

Matching Charts to Questions

For each of the questions below, draw a line to the chart that will best answer it. (You may find that more than one question is best answered by the same chart!)

1. Are there more of the animals at the shelter fixed or unfixed?
2. How many weeks did each cat wait to be adopted?
3. How many male v. female dogs are there?
4. How many animals have 4 legs? 8? 3?
5. What percent of the total weight at the shelter is made up by Boo-boo?
6. What is the distribution of weights across all the animals older than 3?
7. How many animals are there of each species?
8. Who waited the longest to be adopted?



that younger animals *will* get adopted faster, possibly because they are considered cuter, but there may be other factors causing them to get adopted faster.

I would look at both the ages and number of weeks until adoption for each animal to see if there was a correlation. I would also want to collect more data, such as conduct a survey of adopters.

name	species	age	weeks
"Sasha"	"cat"	1	3
"Boo-boo"	"dog"	11	5
"Felix"	"cat"	16	4
"Buddy"	"lizard"	2	24
"Nori"	"dog"	6	9
"Wade"	"cat"	1	2
"Nibblet"	"rabbit"	6	12
"Maple"	"dog"	3	2

A few points are close to the line, but as ages increase the points get much farther apart.

Contract and Purpose

cats-age-weeks :: *(animals :: Table)* → *Image*

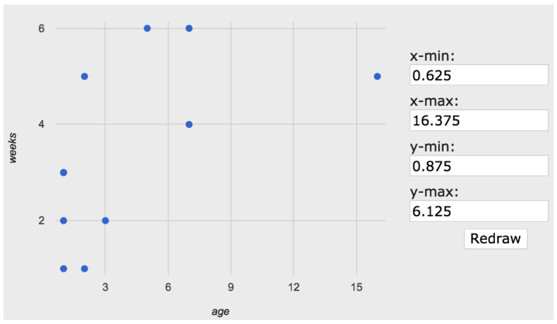
Consumes a table of animals, creates a scatter plot of only the cat's ages and their weeks to adoption

Examples Where I start, what I type, and what I get back

A sample table to start with:	*To use the function, I would type:*
animals-table	cats-age-weeks(animals-table)

→

name	species	age	fixed	legs	weight	weeks
Snowcone	cat	2	TRUE	4	6.1	5
Lucky	dog	3	TRUE	3	45.4	9
Hercules	cat	3	FALSE	4	13.4	7
Toggle	dog	3	TRUE	4	48	3
Snuggles	tarantula	2	FALSE	8	0.1	1

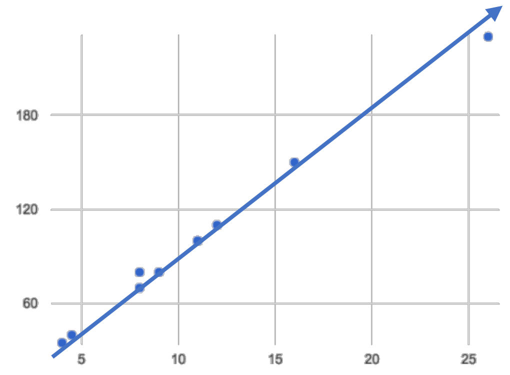


Define the function

Use the relevant methods (circle your helper functions!), then produce a result with the new table.

```
fun cats-age-weeks ( animals ) :
    t = animals-table
                                                   Define the table
                                                   Are there more columns?
    .filter( is-cat )                              Are there fewer rows?
                                                   Are the rows ordered?
    scatter-plot(t, "name", "age", "weeks" )       Produce the result
end
```

A

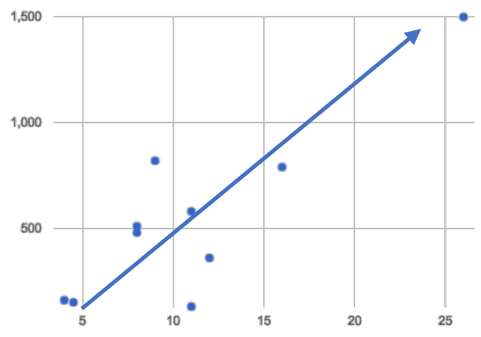


fat (g) v. calories-from-fat in common menu items

Direction: [✓ Positive] Negative None

Strength: [✓ Strong] Weak

B

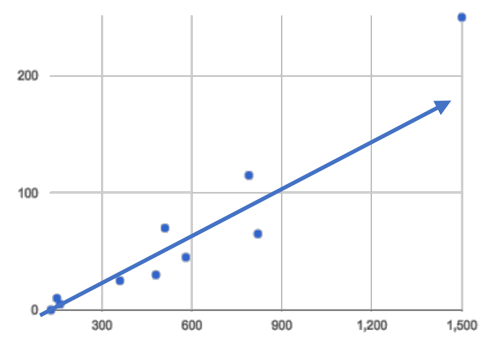


fat (g) v. sodium (g) in common menu items

Direction: [✓ Positive] Negative None

Strength: [✓ Strong] Weak

C

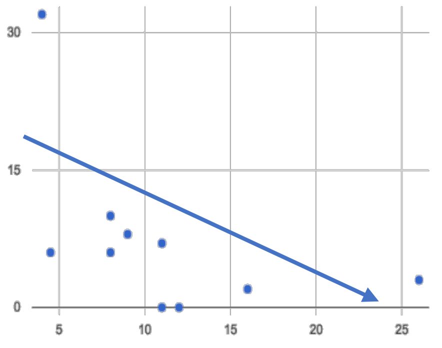


sodium (g) v. cholesterol (mg) in common menu items

Direction: [✓ Positive] Negative None

Strength: [✓ Strong] Weak

D



fat (g) v. sugar (g) in common menu items

Direction: Positive [✓ Negative] None

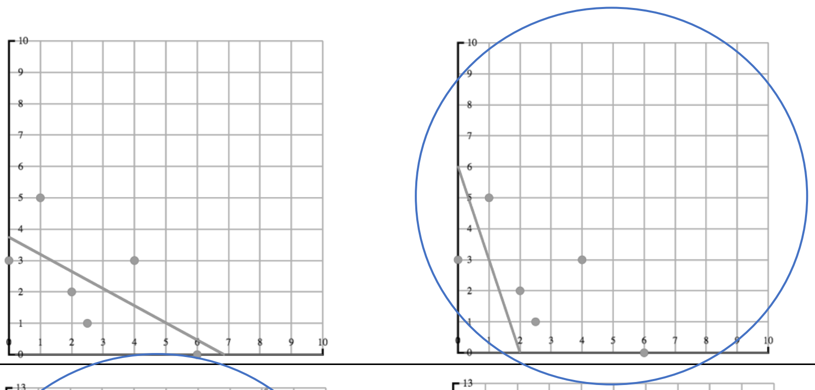
Strength: Strong [✓ Weak]

a subset or extension of my data strong / weak positive / negative

a subset or extension of my data

a subset or extension of my data

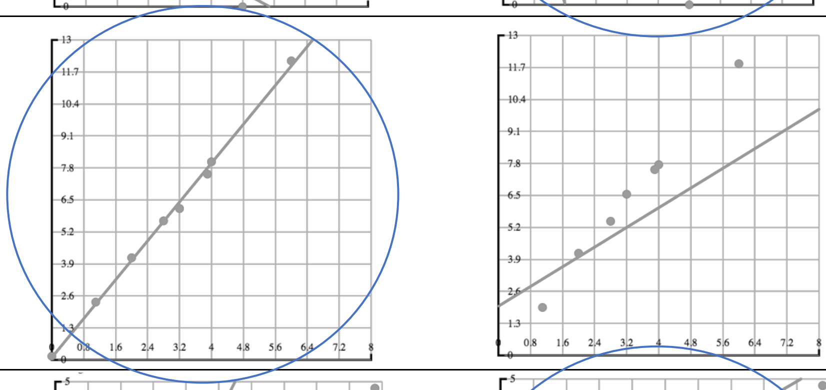
A



Strength of Correlation:

0.2

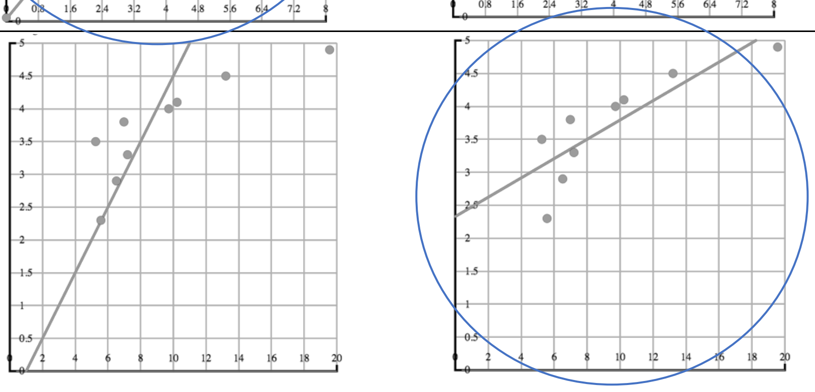
B



Strength of Correlation:

0.95

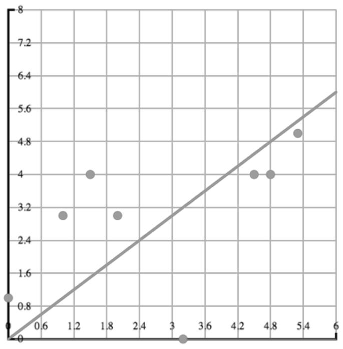
C

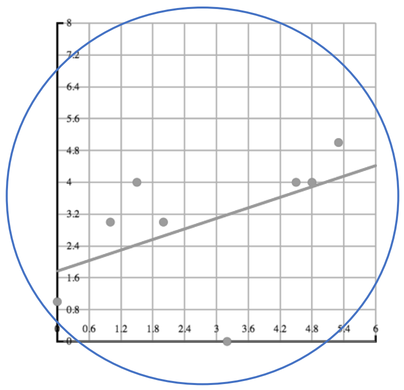


Strength of Correlation:

0.65

D





Strength of Correlation:

0.4

dataset or subset

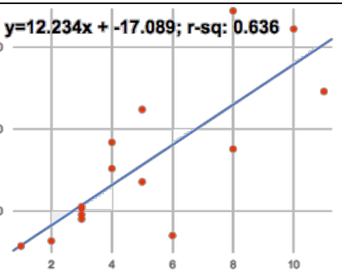
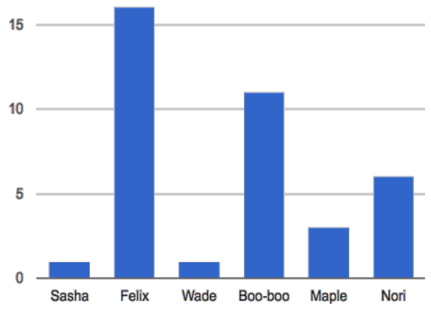
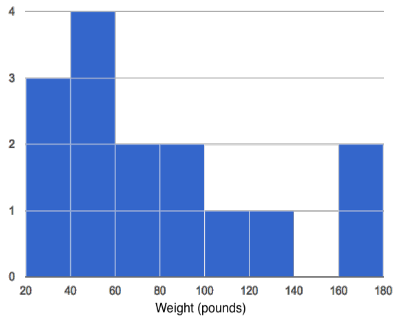
cats at the shelter

0.23 week increase adoption time

	Data	Claim	Why it's *wrong*

[illegible] evening. He finished, the day's [illegible] later, which is likely to make the dogs very excited!

group is wrong!

1	The average player on a basketball team is 6'1".	*"Most of the players on the team are taller than 6'."*	The average is based on all the players, and there may be outliers pushing the average height up-average tells you nothing about the majority of the players.
2	After performing linear regression on census data, a positive correlation (r^2=0.18) was found between people's height and salary.	*"Taller people get paid more."*	Only 18% of the variation in salary is based on height, which is not a large enough r-squared value to say that taller people get paid more.
3		*"According to the predictor function indicated here, the value on the x-axis is will predict the value on the y-axis 63.6% of the time."*	The r-squared value of 0.636 does not mean how often the y-value will be predicted, rather what percent of variation in the y-value is based on the x-value.
4	 Bar Chart of Pet Ages	*"According to this bar chart, Felix makes up a little more than 15% of the total ages of all the animals in the dataset."*	Bar charts are not the most appropriate image for showing the percentage of each measurement based on the total- pie charts should be used for that info. This bar chart shows that Felix is a little more than 15 years old.
5		*"According to this histogram, most animals weigh between 40 and 60 pounds."*	More animals fit into the histogram bin between 40-60 pounds than any other bin, but that doesn't mean that most animals weigh between 40-60 pounds.
6	After performing linear regression, a negative correlation (r^2=0.91) was found between the number of hairs on a person's head and their likelihood of owning a wig.	*"Owning wigs causes people to go bald."*	

________________ : : ________________________ → ________________

name domain range

```
# ____________________________________________________

examples:
    __________(________) is ______________________
    __________(________) is ______________________
end

fun __________ (________) : ______________________

end
```

```
# ______________::____________________ → ______________
        name              domain                range
# ____________________________________________________
examples:
    __________(________) is ______________________
    __________(________) is ______________________
end

fun __________ (________) : ______________________

end
```

Design Recipes

```
# ______________::____________________ → ______________
        name              domain                range
# ____________________________________________________
```

```
examples:

    ______________(__________) is _________________________________

    ______________(__________) is _________________________________
end

fun ______________ (_________) :  _________________________________

end
```

```
# __________________::_________________________ → __________________
         name                    domain                  range
# ___________________________________________________________________
examples:

    ______________(__________) is _________________________________

    ______________(__________) is _________________________________
end

fun ______________ (_________) :  _________________________________

end
```

Design Recipes

```
# __________________::_________________________ → __________________
         name                    domain                  range
```

____

examples:

______________(__________) **is** ________________________________

______________(__________) **is** ________________________________

end

fun ______________ (_________) **:** ________________________________

end

__________________**::**_________________________ → _________________

name domain range

____

examples:

______________(__________) **is** ________________________________

______________(__________) **is** ________________________________

end

fun ______________ (_________) **:** ________________________________

end

Contract and Purpose

__________________**::**_________________________ → _________________

____

Examples

Make a Start Table and a result based on that table.

______________________ → ______________________

Define the function

Use the relevant methods (circle your helper functions!), then produce a result with the new table.

```
fun ____________________ (________) :
    t = ______________________________________________
    ______________________________________________
    ______________________________________________
    ______________________________________________
    ______________________________________________
end
```

Define the table

Are there more columns?

Are there fewer rows?

Are the rows ordered?

Produce the result

Contract and Purpose

________________ :: ______________________ → ________________

____

Examples

Make a Start Table and a result based on that table.

______________________ → ______________________

Define the function

Use the relevant methods (circle your helper functions!), then produce a result with the new table.

```
fun ________________________ (__________) :
    t = _______________________________________________________
        _______________________________________________________
        _______________________________________________________
        _______________________________________________________
        _______________________________________________________
end
```

Define the table

Are there more columns?

Are there fewer rows?

Are the rows ordered?

Produce the result

Contract and Purpose

____________________ :: ____________________________ → ____________________

____

Examples

Make a Start Table and a result based on that table.

_________________________________ → _________________________________

Define the function

Use the relevant methods (circle your helper functions!), then produce a result with the new table.

```
fun ___________________________ (__________) :
  t = ____________________________________________________________
  ________________________________________________________________
  ________________________________________________________________
  ________________________________________________________________
  ________________________________________________________________
end
```

Define the table

Are there more columns?

Are there fewer rows?

Are the rows ordered?

Produce the result

Name	Domain		Range
triangle	:: (side-length :: *Number*, style :: *String*, color :: *String*)	→	*Image*
circle	:: (radius :: *Number*, style :: *String*, color :: *String*)	→	*Image*
star	:: (radius :: *Number*, style :: *String*, color :: *String*)	→	*Image*
rectangle	:: (width :: *Num*, height :: *Num*, style :: *Str*, color :: *Str*)	→	*Image*
ellipse	:: (width :: *Num*, height :: *Num*, style :: *Str*, color :: *Str*)	→	*Image*
square	:: (size-length :: *Number*, style :: *String*, color :: *String*)	→	*Image*
text	:: (str :: *String*, size :: *Number*, color :: *String*)	→	*Image*
overlay	:: (img1 :: *Image*, img2 :: *Image*)	→	*Image*
rotate	:: (degree :: *Number*, img :: *Image*)	→	*Image*
scale	:: (factor :: *Number*, img :: *Image*)	→	*Image*
string-repeat	:: (text :: *String*, repeat :: *Number*)	→	*String*
string-contains	:: (text :: *String*, search-for :: *String*)	→	*Boolean*
num-sqr	:: (n :: *Number*)	→	*Number*
num-sqrt	:: (n :: *Number*)	→	*Number*
num-min	:: (a :: *Number*, b:: *Number*)	→	*Number*
num-max	:: (a :: *Number*, b:: *Number*)	→	*Number*

<*Table*>.row-n	:: (index :: *Number*)	→	*Row*
<*Table*>.order-by	:: (col :: *String*, *increasing* :: *Boolean*)	→	*Table*
<*Table*>.filter	:: (test :: (*Row* → *Boolean*))	→	*Table*
<*Table*>.build-column	:: (col :: *String*, builder :: (*Row* → *Value*))	→	*Table*
mean	:: (t :: *Table*, col :: *String*)	→	*Number*
median	:: (t :: *Table*, col :: *String*)	→	*Number*
modes	:: (t :: *Table*, col :: *String*)	→	*List<Number>*
bar-chart	:: (t :: *Table*, labels :: *String*, values :: *String*)	→	*Image*
pie-chart	:: (t :: *Table*, labels :: *String*, values :: *String*)	→	*Image*
box-plot	:: (t :: *Table*, col:: *String*)	→	*Image*
freq-bar-chart	:: (t :: *Table*, values :: *String*)	→	*Image*
histogram	:: (t :: *Table*, values :: *String*, bin-width :: *Number*)	→	*Image*
scatter-plot	:: (t :: *Table*, labels :: *String*, xs :: *String*, ys :: *String*)	→	*Image*
lr-plot	:: (t :: *Table*, labels :: *String*, xs :: *String*, ys :: *String*)	→	*Image*