Homework 3: Table Manipulation and Visualization

Reading:

- Visualization (https://www.inferentialthinking.com/chapters/07/visualization.html)

Please complete this notebook by filling in the cells provided. Before you begin, execute the following cell to load the provided tests. Each time you start your server, you will need to execute this cell again to load the tests.

Homework 3 is due Thursday, 2/13 at 11:59pm. Start early so that you can come to office hours if you're stuck. Check the website for the office hours schedule. You will receive an early submission bonus point if you turn in your final submission by Wednesday, 2/12 at 11:59pm. Late work will not be accepted as per the policies (http://data8.org/sp20/policies.html) of this course.

Throughout this homework and all future ones, please be sure to not re-assign variables throughout the notebook! For example, if you use `max_temperature` in your answer to one question, do not reassign it later on. Moreover, please be sure to only put your written answers in the provided cells.

In [1]:
```
# Don't change this cell; just run it.

import numpy as np
from datascience import *


# These lines do some fancy plotting magic.\n",
import matplotlib
%matplotlib inline
import matplotlib.pyplot as plots
plots.style.use('fivethirtyeight')

from client.api.notebook import Notebook
ok = Notebook('hw03.ok')
_ = ok.auth(inline=True)
```

```
=====================================================================
Assignment: Homework 3: Table Manipulation and Visualization
OK, version v1.14.19
=====================================================================
---------------------------------------------------------------------------
LoadingException                          Traceback (most recent call last)
<ipython-input-1-a965a5591d49> in <module>
     12 
     13 from client.api.notebook import Notebook
---> 14 ok = Notebook('hw03.ok')
     15 _ = ok.auth(inline=True)

/opt/anaconda3/lib/python3.7/site-packages/client/api/notebook.py in __init__(self, filepath, cmd_args, debug, mode)
     13         ok_logger = logging.getLogger('client')   # Get top-level ok logger
     14         ok_logger.setLevel(logging.DEBUG if debug else logging.ERROR)
---> 15         self.assignment = load_assignment(filepath, cmd_args)
     16         # Attempt a login with enviornment based tokens
     17         login_with_env(self.assignment)

/opt/anaconda3/lib/python3.7/site-packages/client/api/assignment.py in load_assignment(filepath, cmd_args)
     22     if cmd_args is None:
     23         cmd_args = Settings()
---> 24     return Assignment(cmd_args, **config)
     25 
     26 def _get_config(config):

/opt/anaconda3/lib/python3.7/site-packages/client/sources/common/core.py in __call__(cls, *args, **kargs)
    185                 raise ex.SerializeException('__init__() missing expected '
    186                                 'argument {}'.format(attr))
--> 187         obj.post_instantiation()
    188         return obj
    189 

/opt/anaconda3/lib/python3.7/site-packages/client/api/assignment.py in post_instantiation(self)
    151     def post_instantiation(self):
    152         self._print_header()
--> 153         self._load_tests()
    154         self._load_protocols()
    155         self.specified_tests = self._resolve_specified_tests(

/opt/anaconda3/lib/python3.7/site-packages/client/api/assignment.py in _load_tests(self)
    205 
    206         if not self.test_map:
--> 207             raise ex.LoadingException('No tests loaded')
    208 
    209     def dump_tests(self):

LoadingException: No tests loaded
```

Before continuing the assignment, select "Save and Checkpoint" in the File menu and then execute the submit cell below. The result will contain a link that you can use to check that your assignment has been submitted successfully. If you submit more than once before the deadline, we will only grade your final submission. If you mistakenly submit the wrong one, you can head to okpy.org and flag the correct version. There will be another submit cell at the end of the assignment when you finish!

```
In [2]: _ = ok.submit()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-2-cc46ca874451> in <module>
----> 1 _ = ok.submit()

NameError: name 'ok' is not defined
```

1. Unemployment

The Federal Reserve Bank of St. Louis publishes data about jobs in the US. Below, we've loaded data on unemployment in the United States. There are many ways of defining unemployment, and our dataset includes two notions of the unemployment rate:

1. Among people who are able to work and are looking for a full-time job, the percentage who can't find a job. This is called the Non-Employment Index, or NEI.
2. Among people who are able to work and are looking for a full-time job, the percentage who can't find any job *or* are only working at a part-time job. The latter group is called "Part-Time for Economic Reasons", so the acronym for this index is NEI-PTER. (Economists are great at marketing.)

The source of the data is here (https://fred.stlouisfed.org/categories/33509).

Question 1. The data are in a CSV file called `unemployment.csv` . Load that file into a table called `unemployment` .

```
BEGIN QUESTION
name: q1_1
```

In [3]:
```
unemployment = Table().read_table("unemployment.csv") #SOLUTION
unemployment
```

Out[3]:

Date	NEI	NEI-PTER
1994-01-01	10.0974	11.172
1994-04-01	9.6239	10.7883
1994-07-01	9.3276	10.4831
1994-10-01	9.1071	10.2361
1995-01-01	8.9693	10.1832
1995-04-01	9.0314	10.1071
1995-07-01	8.9802	10.1084
1995-10-01	8.9932	10.1046
1996-01-01	9.0002	10.0531
1996-04-01	8.9038	9.9782

... (80 rows omitted)

In [4]:
```
# TEST
unemployment.select('Date', 'NEI', 'NEI-PTER').take(0)
```

Out[4]:

Date	NEI	NEI-PTER
1994-01-01	10.0974	11.172

Question 2. Sort the data in descending order by NEI, naming the sorted table `by_nei`. Create another table called `by_nei_pter` that's sorted in descending order by NEI-PTER instead.

```
BEGIN QUESTION
name: q1_2
```

In [5]:
```
by_nei = unemployment.sort("NEI", descending=True) #SOLUTION
by_nei_pter = unemployment.sort("NEI-PTER", descending=True) #SOLUTION
```

In [6]:
```
# TEST
by_nei.take(0)
```

Out[6]:

Date	NEI	NEI-PTER
2009-10-01	10.9698	12.8557

In [7]:
```
# TEST
by_nei_pter.take(0)
```

Out[7]:

Date	NEI	NEI-PTER
2009-10-01	10.9698	12.8557

Question 3. Use `take` to make a table containing the data for the 10 quarters when NEI was greatest. Call that table `greatest_nei`.

`greatest_nei` should be sorted in descending order of `NEI`. Note that each row of `unemployment` represents a quarter.

```
BEGIN QUESTION
name: q1_3
```

In [8]:
```
greatest_nei = by_nei.take(np.arange(10)) #SOLUTION
greatest_nei
```

Out[8]:

Date	NEI	NEI-PTER
2009-10-01	10.9698	12.8557
2010-01-01	10.9054	12.7311
2009-07-01	10.8089	12.7404
2009-04-01	10.7082	12.5497
2010-04-01	10.6597	12.5664
2010-10-01	10.5856	12.4329
2010-07-01	10.5521	12.3897
2011-01-01	10.5024	12.3017
2011-07-01	10.4856	12.2507
2011-04-01	10.4409	12.247

In [9]:
```
# TEST
greatest_nei.take(0)
```

Out[9]:

Date	NEI	NEI-PTER
2009-10-01	10.9698	12.8557

Question 4. It's believed that many people became PTER (recall: "Part-Time for Economic Reasons") in the "Great Recession" of 2008-2009. NEI-PTER is the percentage of people who are unemployed (and counted in the NEI) plus the percentage of people who are PTER. Compute an array containing the percentage of people who were PTER in each quarter. (The first element of the array should correspond to the first row of `unemployment`, and so on.)

Note: Use the original `unemployment` table for this.

```
BEGIN QUESTION
name: q1_4
```

```
In [10]: pter = unemployment.column("NEI-PTER") - unemployment.column("NEI") #SOL
         UTION
         pter
```

```
Out[10]: array([1.0746, 1.1644, 1.1555, 1.129 , 1.2139, 1.0757, 1.1282, 1.1114,
                1.0529, 1.0744, 1.1004, 1.0747, 1.0705, 1.0455, 1.008 , 0.9734,
                0.9753, 0.8931, 0.9451, 0.8367, 0.8208, 0.8105, 0.8248, 0.7578,
                0.7251, 0.7445, 0.7543, 0.7423, 0.7399, 0.7687, 0.8418, 0.9923,
                0.9181, 0.9629, 0.9703, 0.9575, 1.0333, 1.0781, 1.0675, 1.0354,
                1.0601, 1.01  , 1.0042, 1.0368, 0.9704, 0.923 , 0.9759, 0.93  ,
                0.889 , 0.821 , 0.9409, 0.955 , 0.898 , 0.8948, 0.9523, 0.9579,
                1.0149, 1.0762, 1.2873, 1.4335, 1.7446, 1.8415, 1.9315, 1.8859,
                1.8257, 1.9067, 1.8376, 1.8473, 1.7993, 1.8061, 1.7651, 1.7927,
                1.7286, 1.6387, 1.6808, 1.6805, 1.6629, 1.6253, 1.6477, 1.6298,
                1.4796, 1.5131, 1.4866, 1.4345, 1.3675, 1.3097, 1.2319, 1.1735,
                1.1844, 1.1746])
```

```
In [11]: # TEST
         # It looks like you subtracted in the wrong order.
         round(pter.item(6), 4) != -1.1282
```

```
Out[11]: True
```

```
In [12]: # TEST
         round(pter.item(6), 4)
```

```
Out[12]: 1.1282
```

Question 5. Add `pter` as a column to `unemployment` (named "PTER") and sort the resulting table by that column in descending order. Call the table `by_pter`.

Try to do this with a single line of code, if you can.

```
BEGIN QUESTION
name: q1_5
```

In [13]:
```
by_pter = unemployment.with_column("PTER", pter).sort("PTER", descending
=True) #SOLUTION
by_pter
```

Out[13]:

Date	NEI	NEI-PTER	PTER
2009-07-01	10.8089	12.7404	1.9315
2010-04-01	10.6597	12.5664	1.9067
2009-10-01	10.9698	12.8557	1.8859
2010-10-01	10.5856	12.4329	1.8473
2009-04-01	10.7082	12.5497	1.8415
2010-07-01	10.5521	12.3897	1.8376
2010-01-01	10.9054	12.7311	1.8257
2011-04-01	10.4409	12.247	1.8061
2011-01-01	10.5024	12.3017	1.7993
2011-10-01	10.3287	12.1214	1.7927

... (80 rows omitted)

In [14]:
```
# TEST
by_pter.take(0)
```

Out[14]:

Date	NEI	NEI-PTER	PTER
2009-07-01	10.8089	12.7404	1.9315

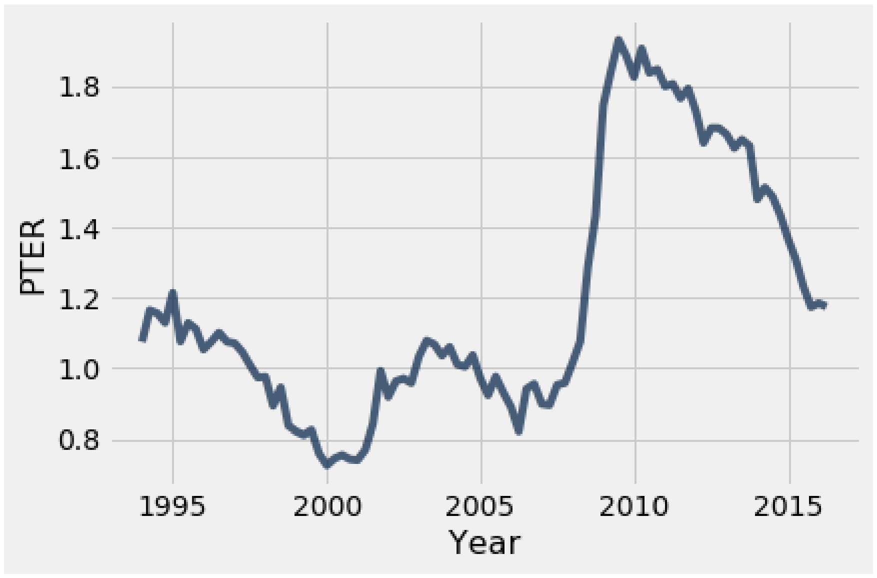
Question 6.

Create a line plot of the PTER over time.

To do this, create a new table called `pter_over_time` that adds the `year` array and the `pter` array to the `unemployment` table. Label these columns `Year` and `PTER`. Then, generate a line plot using one of the table methods you've learned in class.

```
BEGIN QUESTION
name: q1_6
```

In [15]:
```
year = 1994 + np.arange(by_pter.num_rows)/4
pter_over_time = unemployment.with_columns("Year", year, 'PTER', pter) # SOLUTION
pter_over_time.plot("Year", "PTER") # SOLUTION
```



In [16]:
```
# TEST
pter_over_time.take(0)
```

Out[16]:

Date	NEI	NEI-PTER	Year	PTER
1994-01-01	10.0974	11.172	1994	1.0746

In [17]:
```
# HIDDEN TEST
np.all(pter_over_time.column("PTER") == pter)
```

Out[17]:
```
True
```

In [18]:
```
# HIDDEN TEST
np.all(pter_over_time.column("Year") == year)
```

Out[18]:
```
True
```

In [19]:
```
# HIDDEN TEST
pter_over_time.labels
```

Out[19]:
```
('Date', 'NEI', 'NEI-PTER', 'Year', 'PTER')
```

Question 7. Were PTER rates high during the Great Recession (that is to say, were PTER rates particularly high in the years 2008 through 2011)? Assign highPTER to `True` if you think PTER rates were high in this period, and `False` if you think they weren't.

```
BEGIN QUESTION
name: q1_7
```

```
In [20]: highPTER = True #SOLUTION
```

```
In [21]: # TEST
         highPTER == True or highPTER == False

Out[21]: True
```

```
In [22]: # HIDDEN TEST
         highPTER

Out[22]: True
```

2. Birth Rates

The following table gives census-based population estimates for each state on both July 1, 2015 and July 1, 2016. The last four columns describe the components of the estimated change in population during this time interval. **For all questions below, assume that the word "states" refers to all 52 rows including Puerto Rico & the District of Columbia.**

The data was taken from here (http://www2.census.gov/programs-surveys/popest/datasets/2010-2016/national/totals/nst-est2016-alldata.csv).

If you want to read more about the different column descriptions, click here (http://www2.census.gov/programs-surveys/popest/datasets/2010-2015/national/totals/nst-est2015-alldata.pdf)!

The raw data is a bit messy - run the cell below to clean the table and make it easier to work with.

```
In [23]: # Don't change this cell; just run it.
         pop = Table.read_table('nst-est2016-alldata.csv').where('SUMLEV', 40).se
         lect([1, 4, 12, 13, 27, 34, 62, 69])
         pop = pop.relabeled('POPESTIMATE2015', '2015').relabeled('POPESTIMATE201
         6', '2016')
         pop = pop.relabeled('BIRTHS2016', 'BIRTHS').relabeled('DEATHS2016', 'DEA
         THS')
         pop = pop.relabeled('NETMIG2016', 'MIGRATION').relabeled('RESIDUAL2016',
         'OTHER')
         pop = pop.with_columns("REGION", np.array([int(region) if region != "X"
         else 0 for region in pop.column("REGION")]))
         pop.set_format([2, 3, 4, 5, 6, 7], NumberFormatter(decimals=0)).show(5)
```

REGION	NAME	2015	2016	BIRTHS	DEATHS	MIGRATION	OTHER
3	Alabama	4,853,875	4,863,300	58,556	52,405	3,874	-600
4	Alaska	737,709	741,894	11,255	4,511	-2,557	-2
4	Arizona	6,817,565	6,931,071	87,204	56,564	76,405	6,461
3	Arkansas	2,977,853	2,988,248	37,936	30,581	3,530	-490
4	California	38,993,940	39,250,017	502,848	273,850	33,530	-6,451

... (47 rows omitted)

Question 1. Assign `us_birth_rate` to the total US annual birth rate during this time interval. The annual birth rate for a year-long period is the total number of births in that period as a proportion of the population size at the start of the time period.

Hint: Which year corresponds to the start of the time period?

```
BEGIN QUESTION
name: q2_1
```

```
In [24]: us_birth_rate = sum(pop.column('BIRTHS'))/sum(pop.column('2015')) # SOLU
         TION
         us_birth_rate

Out[24]: 0.012358536498646102

In [25]: # TEST
         0 < us_birth_rate < 1

Out[25]: True

In [26]: # HIDDEN TEST
         us_birth_rate == sum(pop.column('BIRTHS'))/sum(pop.column('2015'))

Out[26]: True
```

Question 2. Assign `movers` to the number of states for which the **absolute value** of the **annual rate of migration** was higher than 1%. The annual rate of migration for a year-long period is the net number of migrations (in and out) as a proportion of the population size at the start of the period. The `MIGRATION` column contains estimated annual net migration counts by state.

```
BEGIN QUESTION
name: q2_2
```

```
In [27]: migration_rates = pop.with_column('Migration Rate', np.abs(pop.column('M
         IGRATION')/pop.column('2015'))) # SOLUTION
         movers = migration_rates.where('Migration Rate', are.above(0.01)).num_ro
         ws # SOLUTION
         movers
```

```
Out[27]: 9
```

```
In [28]: # TEST
         0 < movers <= 52
```

```
Out[28]: True
```

```
In [29]: # HIDDEN TEST
         movers == 9
```

```
Out[29]: True
```

Question 3. Assign `west_births` to the total number of births that occurred in region 4 (the Western US).

Hint: Make sure you double check the type of the values in the region column, and appropriately filter (i.e. the types must match!).

```
BEGIN QUESTION
name: q2_3
```

```
In [30]: west_births = sum(pop.where('REGION', are.equal_to(4)).column('BIRTHS'))
         # SOLUTION
         west_births
```

```
Out[30]: 979657
```

```
In [31]: # TEST
         5e5 < west_births < 1e6
```

```
Out[31]: True
```

```
In [32]: # HIDDEN TEST
         west_births == 979657
Out[32]: True
```

Question 4. Assign `less_than_west_births` to the number of states that had a total population in 2016 that was smaller than the *total number of births in region 4 (the Western US)* during this time interval.

```
BEGIN QUESTION
name: q2_4
```

```
In [33]: less_than_west_births = pop.where('2016', are.below(west_births)).num_ro
         ws # SOLUTION
         less_than_west_births
Out[33]: 7
```

```
In [34]: # TEST
         0 <= less_than_west_births <= 52
Out[34]: True
```

```
In [35]: # HIDDEN TEST
         less_than_west_births == 7
Out[35]: True
```

Question 5.

In the next question, you will be creating a visualization to understand the relationship between birth and death rates. The annual death rate for a year-long period is the total number of deaths in that period as a proportion of the population size at the start of the time period.

What visualization is most appropriate to see if there is an association between birth and death rates during a given time interval?

1. Line Graph
2. Scatter Plot
3. Bar Chart

Assign `visualization` below to the number corresponding to the correct visualization.

```
BEGIN QUESTION
name: q2_5
```

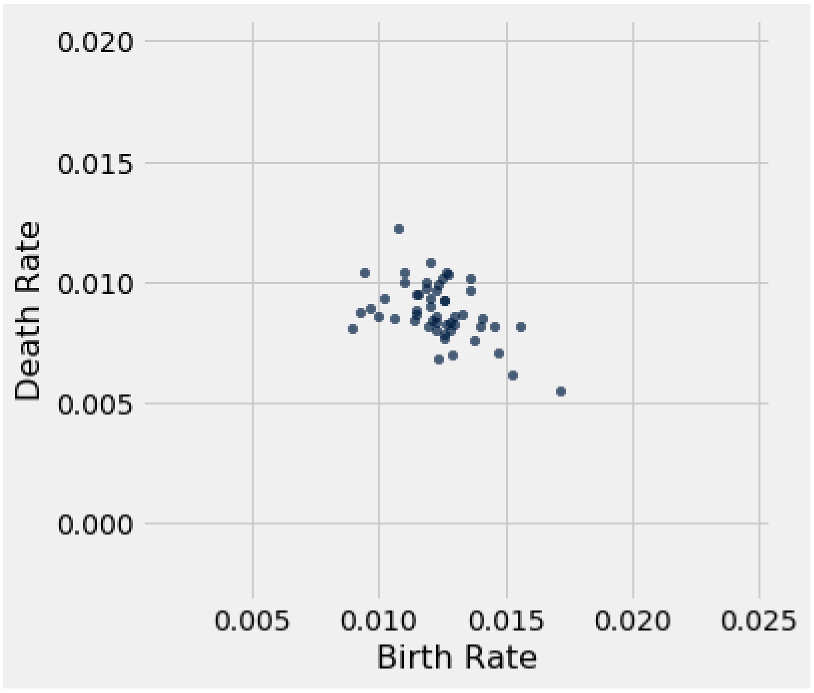
```
In [36]: visualization = 2 #SOLUTION
In [37]: # TEST
         type(visualization) == int

Out[37]: True

In [38]: # TEST
         1 <= visualization <= 3

Out[38]: True

In [39]: # HIDDEN TEST
         visualization == 2

Out[39]: True
```

Question 6.

In the code cell below, create a visualization that will help us determine if there is an association between birth rate and death rate during this time interval. It may be helpful to create an intermediate table here.

```
BEGIN QUESTION
name: q2_6
manual: true
```

```
In [40]: # Generate your chart in this cell
         pop.with_columns("Birth Rate", pop.column('BIRTHS')/pop.column('2015'),
         "Death Rate", pop.column('DEATHS')/pop.column('2015')).scatter(8, 9) # S
         OLUTION
```



Question 7. `True` or `False` : There is an association between birth rate and death rate during this time interval.

Assign `assoc` to `True` or `False` in the cell below.

```
BEGIN QUESTION
name: q2_7
```

In [41]:
```
assoc = True #SOLUTION
```

In [42]:
```
# TEST
type(assoc) is bool
```

Out[42]:
```
True
```

In [43]:
```
# HIDDEN TEST
assoc
```

Out[43]:
```
True
```

3. Marginal Histograms

Consider the following scatter plot:

The axes of the plot represent values of two variables: x and y.

Suppose we have a table called `t` that has two columns in it:

- `x` : a column containing the x-values of the points in the scatter plot
- `y` : a column containing the y-values of the points in the scatter plot

Below, you are given two histograms, each of which corresponds to either column `x` or column `y` .

Histogram A:

Histogram B:

Question 1. Suppose we run `t.hist('x')`. Which histogram does this code produce? Assign `histogram_column_x` to either 1 or 2.

1. Histogram A
2. Histogram B

```
BEGIN QUESTION
name: q3_1
manual: false
```

```
In [44]: histogram_column_x = 2 #SOLUTION
```

```
In [45]: # TEST
         # Make sure you assign histogram_column_x to either 1 or 2!
         type(histogram_column_x) == int

Out[45]: True
```

```
In [46]: # TEST
         histogram_column_x == 1 or histogram_column_x == 2

Out[46]: True
```

```
In [47]: # HIDDEN TEST
         histogram_column_x == 2

Out[47]: True
```

Question 2. Explain why you chose the histogram from Question 1. Make sure to indicate which histogram you selected (ex: "I chose histogram A because ...").

```
BEGIN QUESTION
name: q3_2
manual: true
```

SOLUTION: Because there are no gaps in the X-variable, we would expect the histogram for X to have no gaps in it. Also, because the two masses on the scatter plot overlap in the area between -1 and 0, we would expect there to be more mass in the -1 to 0 area of the histogram, since each vertical slice in this range contains more points. Also, the values of the X-variable range from -2 to 2, which fits the range of values in histogram B.

Question 3. Suppose we run `t.hist('y')`. Which histogram does this code produce? `Assign histogram_column_y` to either 1 or 2.

1. Histogram A
2. Histogram B

```
BEGIN QUESTION
name: q3_3
manual: false
```

```
In [48]: histogram_column_y = 1 #SOLUTION
```

```
In [49]: # TEST
         # Make sure you assign histogram_column_y to either 1 or 2!
         type(histogram_column_y) == int

Out[49]: True
```

```
In [50]: # TEST
         histogram_column_y == 1 or histogram_column_y == 2

Out[50]: True
```

```
In [51]: # HIDDEN TEST
         histogram_column_y == 1

Out[51]: True
```

Question 4. Explain why you chose the histogram from Question 3. Make sure to indicate which histogram you selected (ex: "I chose histogram A because ...").

```
BEGIN QUESTION
name: q3_4
manual: true
```

SOLUTION: There is a gap in the points in the Y-direction, so we would expect a gap in the histogram of those values. Also, the range of values covered by the Y-variable range from -1.5 to 1.5, which fits the range of values in histogram A.

4. Uber

Below we load tables containing 200,000 weekday Uber rides in the Manila, Philippines, and Boston, Massachusetts metropolitan areas from the Uber Movement (https://movement.uber.com) project. The `sourceid` and `dstid` columns contain codes corresponding to start and end locations of each ride. The `hod` column contains codes corresponding to the hour of the day the ride took place. The `ride time` column contains the length of the ride, in minutes.

In [52]:
```
boston = Table.read_table("boston.csv")
manila = Table.read_table("manila.csv")
print("Boston Table")
boston.show(4)
print("Manila Table")
manila.show(4)
```

```
Boston Table
```

sourceid	dstid	hod	ride time
584	33	7	11.866
1013	1116	13	17.7993
884	1190	22	19.3488
211	364	1	1.7235

... (199996 rows omitted)

```
Manila Table
```

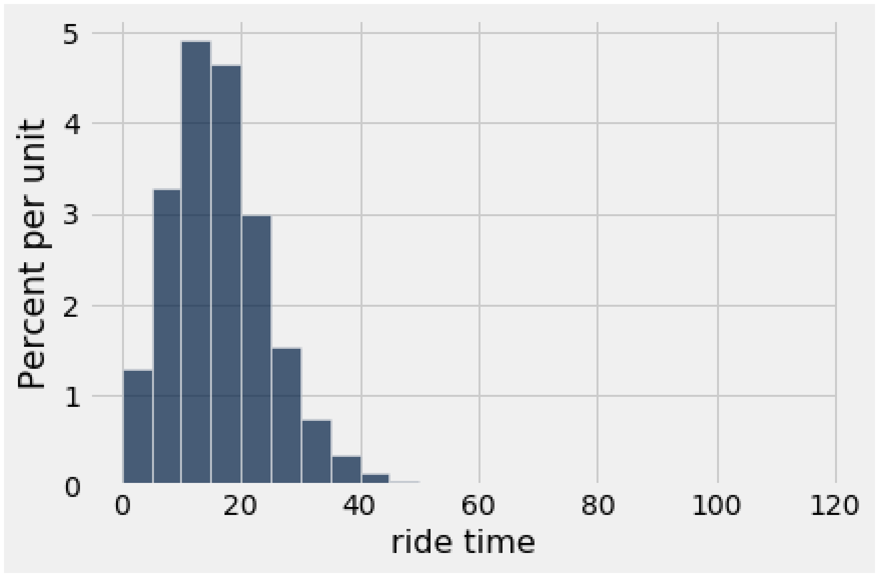
sourceid	dstid	hod	ride time
544	5	22	22.8115
302	240	21	7.02267
278	99	16	21.6437
720	775	18	13.0597

... (199996 rows omitted)

Question 1. Produce histograms of all ride times in Boston using the given bins.

```
BEGIN QUESTION
name: q4_1
manual: true
```

```
In [53]: equal_bins = np.arange(0, 120, 5)
         boston.select("ride time").hist(bins = equal_bins) #SOLUTION
```

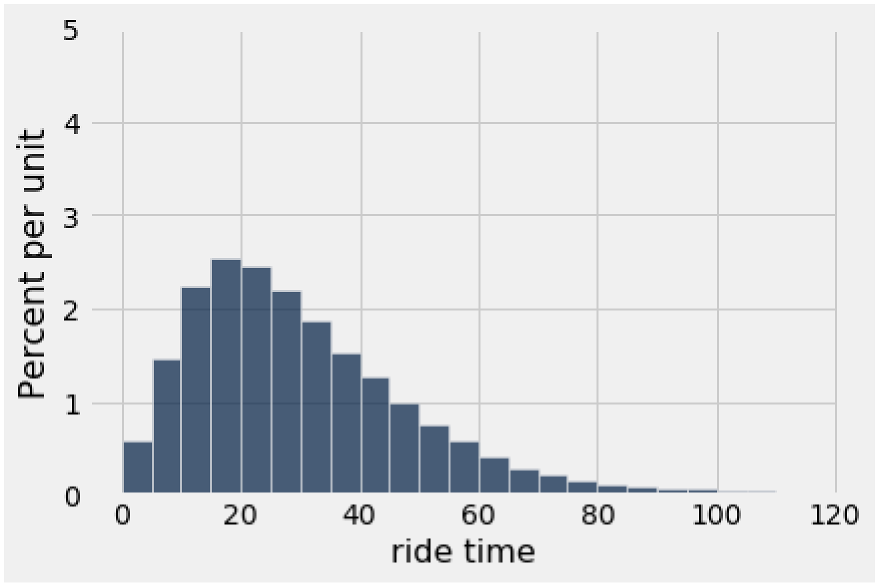


Question 2. Now, produce histograms of all ride times in Manila using the given bins.

```
BEGIN QUESTION
name: q4_2
manual: true
```

```
In [54]: manila.select("ride time").hist(bins = equal_bins) #SOLUTION

         # Don't delete the following line!
         plots.ylim(0, 0.05)
```

```
Out[54]: (0, 0.05)
```



Question 3. Assign `boston_under_10` and `manila_under_10` to the percentage of rides that are less than 10 minutes in their respective metropolitan areas. Use the height variables provided below in order to compute the percentages. Your solution should only use height variables, numbers, and mathematical operations. You should not access the tables boston and manila in any way.

```
BEGIN QUESTION
name: q4_3
manual: false
```

In [55]:
```
boston_under_5_height = 1.2
manila_under_5_height = 0.6
boston_5_to_under_10_height = 3.2
manila_5_to_under_10_height = 1.4

boston_under_10 = 5*boston_under_5_height + 5*boston_5_to_under_10_heigh
t # SOLUTION
manila_under_10 = 5*manila_under_5_height + 5*manila_5_to_under_10_heigh
t # SOLUTION
```

In [56]:
```
# TEST
boston_under_10 >= 0 and boston_under_10 <= 100
```

Out[56]:
```
True
```

In [57]:
```
# TEST
manila_under_10 >= 0 and manila_under_10 <= 100
```

Out[57]:
```
True
```

In [58]:
```
# HIDDEN TEST
boston_under_10 == 5 * 1.2 + 5 * 3.2
```

Out[58]:
```
True
```

In [59]:
```
# HIDDEN TEST
manila_under_10 == 5 * 0.6 + 5 * 1.4
```

Out[59]:
```
True
```

Question 4. Let's take a closer look at the distribution of ride times in Manila. Assign `manila_median_bin` to an integer (1, 2, 3, or 4) that corresponds to the bin that contains the median time

1: 0-15 minutes
2: 15-40 minutes
3: 40-60 minutes
4: 60-80 minutes

Hint: The median of a sorted list has half of the list elements to its left, and half to its right

```
BEGIN QUESTION
name: q4_4
manual: false
```

```
In [60]: manila_median_bin = 2 #SOLUTION
         manila_median_bin
```

```
Out[60]: 2
```

```
In [61]: # TEST
         1 <= manila_median_bin <= 4
```

```
Out[61]: True
```

```
In [62]: # HIDDEN TEST
         manila_median_bin == 2
```

```
Out[62]: True
```

Question 5. What is the main difference between the two histograms. What might be causing this?

Hint: Try thinking about external factors that may be causing the difference!

```
BEGIN QUESTION
name: q4_5
manual: true
```

SOLUTION: Long rides make up a greater proportion of all Uber rides in Manila than in Boston. This could be because there's more traffic in Manila, or because the weather is not pleasant in Boston in the winter, so people may choose to take a car for short trips rather than walk.

5. Submission

Once you're finished, select "Save and Checkpoint" in the File menu and then execute the `submit` cell below. The result will contain a link that you can use to check that your assignment has been submitted successfully. If you submit more than once before the deadline, we will only grade your final submission. If you mistakenly submit the wrong one, you can head to okpy.org (https://okpy.org/) and flag the correct version. To do so, go to the website, click on this assignment, and find the version you would like to have graded. There should be an option to flag that submission for grading!

```
In [63]: _ = ok.submit()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-63-cc46ca874451> in <module>
----> 1 _ = ok.submit()

NameError: name 'ok' is not defined
```

```
In [64]: # For your convenience, you can run this cell to run all the tests at once!
         import os
         print("Running all tests...")
         _ = [ok.grade(q[:-3]) for q in os.listdir("tests") if q.startswith('q') and len(q) <= 10]
         print("Finished running all tests.")
```

```
Running all tests...
Finished running all tests.
```