13 Final project

Congratulations! You've finished all the exercises in this book. If you've gone through each one—and especially if you've gone through the "Beyond the exercise" questions—I'm sure you have improved your pandas skills a lot.

But before you go, I want to give you a final project. We'll explore the "college scorecard," a data set assembled by the US Department of Education about post-secondary (i.e., after high school) educational programs. The college scorecard allows us to see what programs schools offer, how many students they admit, what those students pay in tuition and fees, how many students graduate, and how much they can expect to earn after graduation. From looking at this data, we can better understand many different aspects of university

education in the United States. I should add that with a data set this large and rich, there are many different questions that you could ask. After you finish answering the questions I pose here, I strongly suggest that you also explore the data set on your own, asking (and answering) questions that you think are interesting and relevant.

roblem

Here is what I'd like you to do:

1. Create a data frame (`institutions_df`) from the college scorecard cohorts-institutions CSV file (**Most-Recent-Cohorts-Institution.csv.gz**). You only need to load the following columns: `OPEID6`, `INSTNM`, `CITY`, `STABBR`, `FTFTPCTPELL`, `TUITIONFEE_IN`, `TUITIONFEE_OUT`, `ADM_RATE`, `NPT4_PUB`, `NPT4_PRIV`, `NPT41_ PUB`, `NPT41_PRIV`, `NPT45_PUB`, `NPT45_PRIV`, `MD_EARN_WNE_P10`, and `C100_4`.

2. Load the CSV file for fields of study (**FieldOfStudyData1718_1819_PP.csv.gz**) into another data frame (`fields_of_study_df`). Here, load the columns `OPEID6`, `INSTNM`, `CREDDESC`, `CIPDESC`, and `CONTROL`.

3. Answer the following questions:
    1. What state has the greatest number of universities in this database?
    2. What city, in which state, has the greatest number of universities in this database?
    3. How much memory can you save if you set the `CITY` and `STABBR` columns in `institutions_df` to be categories?
    4. Create a histogram showing how many bachelor programs universities offer.
    5. Determine which university offers the greatest number of bachelor programs.
    6. Create a histogram showing how many graduate (master's and doctoral) programs universities offer.
    7. Determine which university offers the greatest number of different graduate (master + doctoral) programs.

4. Answer these questions:
    1. How many universities offer bachelor's degrees but not master's or doctorates?
    2. How many universities offer master's and doctoral degrees but not bachelor's?
    3. How many institutions offer bachelor's degrees whose name contains the term "Computer Science"?
    4. The `CONTROL` field describes the types of institutions in the database. How many of each type offer a computer science program?
5. Create a pie chart showing the different types of institutions that offer CS (short for "computer science") degrees.
6. Determine the minimum, median, mean, and maximum tuitions for an undergrad CS degree. (We define this as a bachelor's program with the phrase "Computer Science" in the name.) When comparing tuition, use `TUITIONFEE_OUT` for all schools.

7. Describe the tuition again, but grouped by the different types of universities (`CONTROL`).
8. Determine the correlation between admission rate and tuition cost. How would you interpret this?
9. Create a scatter plot with tuition on the *x* axis, admission rate on the *y* axis, and median earnings after 10 years are used for colorizing. Use the "Spectral" colormap. Where do the lowest-paid graduates show up on the graph?
10. Determine which universities are in the top 25% of tuition and also the top 25% with Pell grants (i.e., government assistance to lower-income students). Print only the institution name, city, and state, ordered by institution name.

11. `NPT4_PUB` indicates the average net price for public institutions (in-state tuition) and `NPT4_PRIV` for private institutions. `NPT41_PUB` and `NPT45_PUB` show the average price paid by people in the lowest income bracket (1) versus the highest income bracket (5) at public institutions. `NPT41_PRIV` and `NPT45_PRIV` show the average price paid by people in the lowest income bracket (1) versus the highest income bracket (5) at private institutions. At how many institutions does the bottom quintile receive money (i.e., the value is negative)?
12. Determine the average proportion that the bottom quintile pays versus the top quintile at public universities.
13. Determine the average proportion that the bottom quintile pays versus the top quintile at private universities?

14. Let's try to figure out which universities offer the best overall return on investment (ROI) (across all disciplines):
    1. For which schools in the cheapest 25% do their students have the top 25% of salaries 10 years after graduation?
    2. How about private institutions?
    3. Is there a correlation between admission rates and completion rates? That is: If a school is highly selective, are students more likely to graduate?
    4. Ten years after graduating, from what kinds of schools (private, for-profit, private nonprofit, or public) do people earn, on average, the greatest amount?
    5. Do people who graduate from "Ivy Plus" schools (the Ivy League as well as MIT, Stanford, and the University of Chicago) earn more than the average private-school university graduate? If so, how much more?

6. Do people studying at universities in particular states earn, on average, more after 10 years?

15. Create a bar plot for the average amount earned, per state, sorted by ascending pay.
16. Create a boxplot for the earnings by state.

Column names and meanings

The column names in the two CSV files we're examining in this chapter are terse, as shown in tables 13.1 and 13.2.

Table 13.1 Column names in the “Cohorts and institutions”

file

Column name	Explanation	Sample value
`OPEID6`	Unique ID (integer) for each educational institution	1002
`INSTNM`	Institution name	`"Alabama A & M University"`
`CITY`	Institution's city	`"Normal"`
`STABBR`	Institution's state name (abbreviated)	`"AL"`
`FTFTPCTPELL`	Percentage of Pell-grant recipients	0.6925
`TUITIONFEE_IN`	In-state tuition	10024.0
`TUITIONFEE_OUT`	Out-of-state tuition	18634.0
`ADM_RATE`	Admission rate	0.8965

`NPT4_PUB`	Net price (for public institutions; `NaN` if a private institution)	15529.0
`NPT4_PRIV`	Net price (for private institutions; `NaN` if a public institution)	`NaN`
`NPT41_PUB`	Average price paid by people in the lowest income bracket (for public institutions; `NaN` if a private institution)	14694.0
`NPT41_PRIV`	Average price paid by people in the lowest income bracket (for private institutions; `NaN` if a	`NaN`

	public institution)	
`NPT45_PUB`	Average price paid by people in the highest income bracket (for public institutions; `NaN` if a private institution)	20483.0
`NPT45_PRIV`	Average price paid by people in the highest income bracket (for private institutions; `NaN` if a public institution)	`NaN`
`MD_EARN_WNE_P10`	Median income for graduates 10 years following graduation	36339.0

`C100_4`	Completion rates after four years	0.1052

Table 13.2 Column names in the “Fields of study” file

Column name	Explanation	Sample value
`OPEID6`	Unique ID (integer) for each educational institution	1002
`INSTNM`	Institution name	`"Alabama A & M University"`
`CREDDESC`	Degree being offered	`"Bachelors Degree"`
`CIPDESC`	Education program	`"Agriculture, General."`
`CONTROL`	What type of institution is this?	`"Public"`

Working it out

As I said, the college scorecard data set includes a large

number of facts and figures about American higher education, describing both the institutions and the students who learn there. To answer this set of questions, we only need to look at two CSV files: (1) information about the most recent cohorts of students who enrolled at and graduated from these institutions and (2) the fields of study that each institution offers. Some of our questions require just one of these data sources, and others require that we combine them into a single data frame.

Create a data frame from the college scorecard cohorts-institutions CSV file

To start, I asked you to load each of the CSV files into a data frame. You may have noticed that the files I've provided have a .csv.gz suffix. This means they are compressed with gzip—but you don't need to uncompress them before loading because pandas is smart enough to automatically do so when we run `read_csv`. Load the first data frame as follows, defining `institutions_df`:

```
institutions_filename = '../data/Most-Recent-Cohorts-Institution.csv.gz'
institutions_df = pd.read_csv(institutions_filename,
                              usecols=['OPEID6',
                                       'INSTNM', 'CITY', 'STABBR',
                                       'FTFTPCTPELL', 'TUITIONFEE_IN',
                                       'TUITIONFEE_OUT', 'ADM_RATE',
                                       'NPT4_PUB', 'NPT4_PRIV',
                                       'NPT41_PUB', 'NPT41_PRIV',
                                       'NPT45_PUB', 'NPT45_PRIV',
                                       'MD_EARN_WNE_P10', 'C100_4'])
```

Load the CSV file for fields of study into another data frame

We load the other CSV file with information about fields of study for the last few years as follows, assigning it to `fields_df`:

```
fields_filename = '../data/FieldOfStudyData1718_1819_PP.csv.gz'
fields_of_study_df = pd.read_csv(fields_filename,
                                 usecols=['OPEID6', 'INSTNM',
                                 'CREDDESC', 'CIPDESC', 'CONTROL'])
```

With these two data frames defined and in memory, we can start performing some queries.

What state has the greatest number of universities in this database?

First, I wanted to know which state has the greatest number of universities in this

database. This is a classic example of when to use grouping. We can group on the `STABBR` (state abbreviation) column, running the `count` method. This will tell us how often each state appears in the data set. We also have to provide a second column, which is where the count is reported. The choice doesn't matter, so we go with `OPEID6`, the unique ID used for each institution:

```
(
    institutions_df
    .groupby('STABBR')['OPEID6'].count()
)
```

This query tells us how often each state appears in the data set. But we're interested in finding which states have the greatest number of universities. To find that, we sort the series by the values we get back in descending order. The first row in this series is, by definition, the state with the largest number —which we retrieve using `head(1)`:

```
(
    institutions_df
    .groupby('STABBR')['OPEID6'].count()
```

```
    .sort_values(ascending=False)
    .head(1)
)
```

According to this data, California has the greatest number of universities—a large number, at 705.

What city, in which state, has the greatest number of universities in this database?

I then decided to ask a slightly different question: which city has the greatest number of universities? At first glance, it may seem that this query is identical to the previous one, grouping by the `CITY` column rather than `STABBR`. But that would combine cities of the same name in different states, combining Springfield, Illinois with Springfield, Massachusetts. The solution requires that we group by two columns: first `STABBR` and then `CITY`. The combination allows us to find which city, in which state, has the greatest number of universities:

```
(
    institutions_df
    .groupby(['STABBR', 'CITY'])['OPEID6'].count()
    .sort_values(ascending=False)
```

```
    .head(1)
)
```

Once again, we ask for the `count` method to be run on `OPEID6` because we need to count on a nongrouping column. And again we sort in descending order and grab the top value. The answer is New York City, with 81 institutions of higher learning.

How much memory can we save if we set the CITY and STABBR columns in institutions_df to be categories?

Considering that both state and city names are text data and that they repeat so often, it makes sense to consider how much memory we may save by turning the `STABBR` and `CITY` columns into categories. But as always when trying to optimize, we should measure before and after taking such an action, to know whether our efforts were worthwhile.

I thus asked you to determine how much memory our data frame was already using. The easiest way to find this is to run `memory_usage` on a data frame. Don't forget to pass the

`deep=True` keyword argument. This returns the total memory usage of each column, including the objects to which it refers. (As we saw in chapter 12, that argument can make a huge difference!) Here's how we can calculate that and then print it:

```
pre_category_memory = (
    institutions_df
    .memory_usage(deep=True)
    .sum()
)
print(f'{pre_category_memory:,}')
```

First, we calculate the total memory usage and assign it to `pre_category_memory`. Then, to print the number with commas between the digits—and yes, we're showing off here—we print it in an f-string, using a single comma (`,`) as the format specifier after the colon (`:`).

We then turn both the `STABBR` and `CITY` columns into categories:

```
institutions_df['CITY'] = (
    institutions_df['CITY']
    .astype('category')
)
```

```
institutions_df['STABBR'] = (
    institutions_df['STABBR']
    .astype('category')
)
```

Now that this has been done, how much memory did we save?

```
post_category_memory = (
    institutions_df
    .memory_usage(deep=True)
    .sum()
)

savings = pre_category_memory - post_category_memory
print(f'{savings:,}')

savings = pre_category_memory - post_category_memory
print(f'{savings:,}')
```

On my computer, the savings is calculated as 579,371 bytes—meaning we reduced memory usage by approximately one-third by turning these two columns into categories. Not a bad gain for a few seconds of coding, I'd say.

Create a histogram showing how many bachelor's programs universities offer.

Next, I asked you to create a histogram indicating how many programs are offered by

each university. That is, we'd like to know how many universities offer 10 programs, how many offer 20, how many offer 30, and so forth.

To create such a histogram, we first need to count the number of different bachelor's programs each university offers. We start by looking at `fields_of_study_df` and retrieving only those rows for which the `CREDDESC` value is `'Bachelors Degree'`:

```
(
    fields_of_study_df
    .loc[fields_of_study_df['CREDDESC'] == 'Bachelors Degree']
```

With that in hand, we can run `groupby` on the `INSTNM` (institution name) column. This means our aggregation method (`count`, in this case) runs once for each distinct value of `INSTNM`. To avoid getting a result for each column, we restrict our output to `CIPDESC`:

```
(
    fields_of_study_df
    .loc[fields_of_study_df['CREDDESC'] == 'Bachelors Degree']
    .groupby('INSTNM')['CIPDESC'].count()
    .plot.hist()
)
```

This returns a series in which the index contains the institution name and the value contains the number of bachelor-level degrees offered by each institution. Finally, we can feed that into the histogram-plotting method (figure 13.1):

```
(
    fields_of_study_df
    .loc[fields_of_study_df['CREDDESC'] == 'Bachelors Degree']
    .groupby('INSTNM')['CIPDESC'].count()
    .plot.hist()
)
```

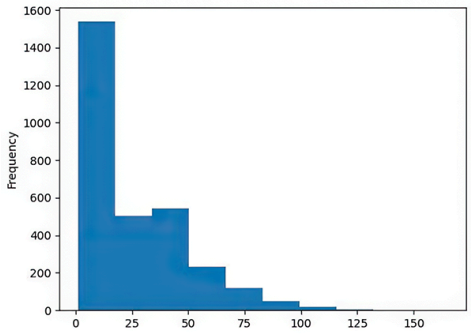

Figure 13.1 Histogram showing how many schools offer different numbers of bachelor's programs

The result shows that a very large number of institutions (more than 1,400!) offer fewer than 20 bachelor-level programs, fewer than 600 institutions offer between 20 and 50 programs, and 200 or fewer institutions offer more than 50 programs.

Which university offers the greatest number of bachelor's programs?

Now that we've counted the number of programs offered by each institution in this data set, we can ask which universities offer the greatest number of programs. We already have their counts, thanks to the `groupby` we ran before. We can thus rerun that query, sorting the resulting values in descending order and keeping only the top 10 results:

```
(
    fields_of_study_df
    .loc[fields_of_study_df['CREDDESC'] == 'Bachelors Degree']
    .groupby('INSTNM')['CIPDESC'].count()
    .sort_values(ascending=False)
    .head(10)
)
```

When I ran this, I found that the institution with the greatest number of programs was Westminster College (with 165 bachelor-level programs), followed by Pennsylvania State University's main campus (141) and the University of Washington's Seattle campus (137).

Create a histogram showing how many graduate (master's and doctoral) programs universities offer.

Now that we've counted bachelor's programs, how about graduate programs offering either a master's or doctoral degree? That query is trickier because we can no longer compare `CREDDESC` with a single string. Rather, we need to check if the value is one of two different strings. For that, we use the `isin` method, which takes a list of strings and returns `True` if the value in that row matches one or more of the values in the list.

To start, we can get all schools that offer master's and doctoral degrees:

```
(
    fields_of_study_df
    .loc[fields_of_study_df['CREDDESC']
          .isin(["Master's Degree", "Doctoral Degree"])]
)
```

With that in hand, we can repeat our `groupby` query, using `count` as our aggregation method:

```
(
    fields_of_study_df
    .loc[fields_of_study_df['CREDDESC']
         .isin(["Master's Degree", "Doctoral Degree"])]
    .groupby('INSTNM')['CIPDESC'].count()
)
```

Finally, having grouped by `INSTNM` using `count` and knowing how many programs each institution offers, we can create the histogram (figure 13.2):

```
(
    fields_of_study_df
    .loc[fields_of_study_df['CREDDESC']
         .isin(["Master's Degree", "Doctoral Degree"])]
    .groupby('INSTNM')['CIPDESC'].count()
    .plot.hist()
)
```

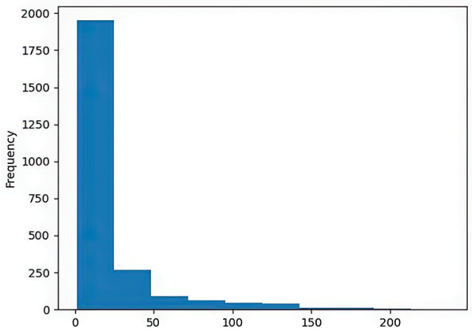

Figure 13.2 Histogram showing how many schools offer different numbers of graduate programs

Here we see that the vast majority of schools offer fewer than 25 different graduate programs, with more offering fewer than 50. The number of

schools offering more than 50 master's and doctoral degrees declines even more precipitously, although a handful offer more than 200.

Which university offers the greatest number of different graduate (master + doctoral) programs?

I next asked you to find just which schools offer the greatest number of different graduate programs. As before, this means sorting the results from our `groupby` and `count`:

```
(
    fields_of_study_df
    .loc[fields_of_study_df['CREDDESC']
        .isin(["Master's Degree", "Doctoral Degree"])]
    .groupby('INSTNM')['CIPDESC'].count()
    .sort_values(ascending=False)
    .head(10)
)
```

The University of Washington's Seattle campus has the most programs (237), followed by Penn State's main campus (230) and New York University (226).

NOTE The number of programs a university offers at

any level shouldn't be taken as an indication of how good the university is or whether the program is appropriate for you. Especially when it comes to graduate studies, the important thing is whether the specific program is good for you and (perhaps even more importantly) whether your advisor is someone you can trust to help you through the program. So don't take these questions as anything other than a numeric exercise; I'm certainly not trying to imply that the more programs a university offers, the better it is.

How many universities offer bachelor's degrees but not master's or doctorates?

Although the universities I attended all offered degree programs at all levels, some focus exclusively on either undergraduate or graduate education. I asked you to find how many universities offer bachelor's degrees but not master's or doctorates, followed by the reverse—how many offer master's or doctoral degrees but not bachelor's.

To answer these questions, we first find all schools offering bachelor's programs and those offering master's and doctoral programs. These queries are identical to what we did before. However, here we store them in two separate variables so we can make calculations based on them:

```
ug_schools = (
    fields_of_study_df
    .loc[fields_of_study_df['CREDDESC'] == 'Bachelors Degree',
    'INSTNM']
)

grad_schools = (
    fields_of_study_df
    .loc[fields_of_study_df['CREDDESC']
        .isin(["Master's Degree", "Doctoral Degree"]),
    'INSTNM']
)
```

Both `ug_schools` and `grad_schools` are pandas series with the values containing the names of the universities. However, because we retrieved the university names from `fields_of_study_df`, there are plenty of repeats, with one row for each program offered by the institution. We will leave things as they are rather than apply the `unique` method because `apply`

returns a NumPy array and we want to use some additional pandas functionality.

Now that we have defined these two series, how can we determine which schools offer bachelor's degrees but not master's or doctoral degrees? We can again rely on `isin`. That is, to find all undergraduate institutions that are also graduate schools, we can say

```
ug_schools.isin(grad_schools)
```

This code returns a boolean series. But we want the opposite of this: the undergraduate schools that are *not* graduate schools. So we use `~` to flip the logic:

```
~ug_schools.isin(grad_schools)
```

This gives the opposite boolean series from what we had before. If we apply that boolean series to `ug_schools`, we get the rows corresponding to undergraduate schools that aren't graduate schools:

```
ug_schools[~ug_schools.isin(grad_schools)]
```

However, there is a problem with this result: the school names are repeated. This is where we can use the `drop_duplicates` method, getting distinct values back:

```
ug_schools[~ug_schools.isin(grad_schools)].drop_duplicates()
```

We can retrieve `size` from the result:

```
ug_schools[~ug_schools.isin(grad_schools)].drop_duplicates().size
```

The database has 923 undergraduate schools that don't offer graduate degrees.

How many universities offer master's and doctoral degrees but not bachelor's?

We can apply similar logic to this to flip the question around:

```
grad_schools[~grad_schools.isin(ug_schools)].drop_duplicates().size
```

The result is 404 institutions that offer master's and doctoral degrees but don't offer bachelor's degrees.

How many institutions offer bachelor's degrees whose name

contains the term "Computer Science"?

Next, I thought it would be interesting to determine how many institutions offer bachelor's degrees in computer science. Every institution calls its department and degree something slightly different, which means we'll likely miss many possibilities. But if we look for programs containing the term `'Computer Science'`, how many will we find?

First, we need to find all those rows in which `CIPDESC` contains the string `'Computer Science'`:

```
fields_of_study_df['CIPDESC'].str.contains('Computer Science')
```

But that isn't enough because we're specifically looking for bachelor's programs in computer science. We thus need to have two conditions joined with `&`:

```
fields_of_study_df['CIPDESC'].str.contains('Computer Science') &
    fields_of_study_df['CREDDESC'] == 'Bachelors Degree'
```

This combined query returns a boolean series. We can then

apply that boolean series to `fields_of_study_df` with `.loc`:

```
(
    fields_of_study_df
    .loc[(fields_of_study_df['CIPDESC']
          .str.contains('Computer Science')) &
           (fields_of_study_df['CREDDESC'] == 'Bachelors Degree')]
)
```

The thing is, we're not interested in all the columns. We just want to see the institution names so we can count them. We can do this by adding a column selector to `.loc`, indicating that we want to see `INSTNM`:

```
(
    fields_of_study_df
    .loc[(fields_of_study_df['CIPDESC']
          .str.contains('Computer Science')) &
          (fields_of_study_df['CREDDESC'] == 'Bachelors Degree'),
         'INSTNM']
)
```

This returns a series of 824 institution names. But, as before, the names aren't necessarily unique, given that there may be more than one degree program with "Computer Science" in its name. For this reason, we take the results, invoke `unique()`

on them, and then get the size of the resulting array:

```
(
    fields_of_study_df
    .loc[(fields_of_study_df['CIPDESC']
          .str.contains('Computer Science')) &
          (fields_of_study_df['CREDDESC'] == 'Bachelors Degree'),
          'INSTNM']
    .unique()
    .size
)
```

The result, on my system, is 762.

How many types of institutions in the database offer a computer-science program?

The college scorecard data set puts each university into one of four categories listed in the `CONTROL` column: public, private and nonprofit, private and for-profit, or foreign. In my next question, I asked you to show how many institutions of each type offer computer science as a bachelor-level degree.

We start with our previous query before the call to `unique`:

```
(
    fields_of_study_df
```

```
    .loc[(fields_of_study_df['CIPDESC']
            .str.contains('Computer Science')) &
         (fields_of_study_df['CREDDESC'] == 'Bachelors Degree'),
    ['CONTROL','INSTNM']].groupby('CONTROL').count()
)
```

We then run a `groupby` on `CONTROL` because we want to know how many institutions of each type offer undergraduate CS programs. For this to work, our column selector needs to include not just `INSTNM`, as before, but also `CONTROL`:

```
fields_of_study_df.loc[(fields_of_study_df[
        'CIPDESC'].str.contains('Computer Science')) &
        (fields_of_study_df['CREDDESC'] ==
            'Bachelors Degree'), ['CONTROL',
            'INSTNM']].groupby('CONTROL')
```

This query gives us a `groupby` object on which we can then invoke `count`:

```
fields_of_study_df.loc[(fields_of_study_df[
        'CIPDESC'].str.contains('Computer Science')) &
        (fields_of_study_df['CREDDESC'] ==
            'Bachelors Degree'),
            ['CONTROL', 'INSTNM']].groupby('CONTROL').count()
```

With this query, I find 32 foreign universities, 18 private for-profit universities, 501 private nonprofit universities, and 273 public universities, all

offering undergraduate CS programs.

Create a pie chart showing the different types of institutions that offer CS degrees.

Seeing this information in a table, however accurate, isn't as striking as a graphical display would be. I thus asked you to take these results and put them into a pie chart. Fortunately, that's easy. We start with this query and then retrieve only the `INSTNM` column:

```
(
    fields_of_study_df
    .loc[(fields_of_study_df['CIPDESC']
              .str.contains('Computer Science')) &
         (fields_of_study_df['CREDDESC'] == 'Bachelors Degree'),
    ['CONTROL','INSTNM']]
    .groupby('CONTROL').count()['INSTNM']
)
```

That returns a single series of values along with the index (i.e., the different institution categories). We can turn that into a pie chart by invoking `.plot.pie()` at the end (figure 13.3):

```
(
    fields_of_study_df
```

```
    .loc[(fields_of_study_df['CIPDESC']
              .str.contains('Computer Science')) &
         (fields_of_study_df['CREDDESC'] == 'Bachelors Degree'),
    ['CONTROL','INSTNM']]
    .groupby('CONTROL').count()['INSTNM']
    .plot.pie()
)
```

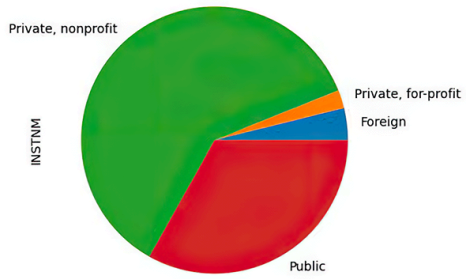

Figure 13.3 Pie chart comparing the types of institutions that offer computer science

Next, we want to start looking at the cost of getting a computer science degree from an American university. To do this, we first need to find all universities at which computer science is taught at the undergraduate level. This query is identical to one we've already seen, except that we're looking for three different columns: `OPEID6` (a unique ID number for each university in the system), `CONTROL` (the category of institution we've already seen), and `INSTNM` (the name of the institution):

```
comp_sci_universities = (
    fields_of_study_df
```

```
    .loc[(fields_of_study_df['CIPDESC']
            .str.contains('Computer Science')) &
         (fields_of_study_df['CREDDESC'] == 'Bachelors Degree'),
    ['OPEID6','CONTROL','INSTNM']]
)
```

The good news is that we now have these rows and have put them into a new data frame, `comp_sci_universities`. However, the index contains the same values we had in `fields_of_study_df`. This isn't inherently bad, except that to answer our questions, we need to join this data frame with `institutions_df`. Joining requires that the indexes match up. For that reason, we modify our creation of `comp_sci_universities` so it sets the index to be `OPEID6`:

```
comp_sci_universities = (
    fields_of_study_df
    .loc[(fields_of_study_df['CIPDESC']
            .str.contains('Computer Science')) &
         (fields_of_study_df['CREDDESC'] == 'Bachelors Degree'),
    ['OPEID6','CONTROL','INSTNM']]
    .set_index('OPEID6')
)
```

Now let's make sure `institutions_df` has the index we need to join them:

```
institutions_df[['OPEID6', 'TUITIONFEE_OUT']].set_index('OPEID6')
```

Note that this doesn't change `institutions_df`; rather, it returns a new data frame with `OPEID6` as its index.

Now that we have two data frames with a common index, we can join them:

```
(
    comp_sci_universities
    .join(institutions_df[['OPEID6', 'TUITIONFEE_OUT']]
          .set_index('OPEID6'))
)
```

But this query gives us the entire new data frame. We don't really want that; we only need the `TUITIONFEE_OUT` column:

```
(
    comp_sci_universities
    .join(institutions_df[['OPEID6', 'TUITIONFEE_OUT']]
          .set_index('OPEID6'))
    ['TUITIONFEE_OUT']
)
```

The result of this query, short as it is, packs a real punch: we retrieve the tuition at each university with an undergraduate CS program in the data set.

What are the minimum, median, mean, and maximum tuitions for an undergrad CS degree?

I asked you to find the minimum, median, mean, and maximum values for tuition. We could, of course, calculate each of these individually. But when we want to perform a number of aggregate calculations, the easiest thing to do is invoke `describe`, which gives them all:

```
(
    comp_sci_universities
    .join(institutions_df[['OPEID6', 'TUITIONFEE_OUT']]
          .set_index('OPEID6'))
    ['TUITIONFEE_OUT']
    .describe()
)
```

Describe the tuition again, but grouped by the different types of universities

Next, I asked you to describe the tuition again, but grouped by the different types of universities (i.e., the `CONTROL` column). We can accomplish this by invoking `groupby('CONTROL')` on the result of the join, retrieving `TUITIONFEE_OUT`, and then

invoking `describe` on the result:

```
comp_sci_universities.join(institutions_df[
    ['OPEID6', 'TUITIONFEE_OUT']].set_index('OPEID6')).groupby(
        'CONTROL')['TUITIONFEE_OUT'].describe()
```

However, I find two problems with this result. First, foreign-owned universities give results of 0 or `NaN` for each column. Second, it's weird to have the university types in the index and the results from `describe` in the columns. Both of these are problems of aesthetics, but if we're already playing with the data, let's see how we can clean it up.

We can use `dropna` to remove the `Foreign` row, the only one in which we have any `NaN` values:

```
(
    comp_sci_universities
    .join(institutions_df[['OPEID6', 'TUITIONFEE_OUT']]
          .set_index('OPEID6'))
    .groupby('CONTROL')['TUITIONFEE_OUT'].describe()
    .dropna()
)
```

What about my preference that the values of `describe` be in the rows rather than the columns? We can transpose

rows and columns in a pandas data frame with the `transpose` method:

```
(
    comp_sci_universities
    .join(institutions_df[['OPEID6', 'TUITIONFEE_OUT']]
          .set_index('OPEID6'))
    .groupby('CONTROL')['TUITIONFEE_OUT'].describe()
    .dropna()
    .transpose()
)
```

However, because this is used so often, we can instead invoke it with `T`:

```
(
    comp_sci_universities
    .join(institutions_df[['OPEID6', 'TUITIONFEE_OUT']]
          .set_index('OPEID6'))
    .groupby('CONTROL')['TUITIONFEE_OUT'].describe()
    .dropna()
    .T
)
```

NOTE Whereas `transpose` is a method and needs to be invoked with parentheses after its name, `T` is a Python property and should *not* have parentheses. Using `T()` will result in an error. They are otherwise equivalent to one another.

What is the correlation between admission rate and tuition cost? How would you interpret this?

We often hear that the most expensive universities are also the hardest to get into. Is this true? Do we see a correlation in the data? To find out, we can invoke `corr` on `institutions_df`, looking at the `ADM_RATE` and `TUITIONFEE_OUT` columns:

```
institutions_df[['ADM_RATE', 'TUITIONFEE_OUT']].corr()
```

As always, a correlation of 0 means there's no correlation between the two values, 1 means they're perfectly aligned, and –1 means they're completely opposite. In this case, we see a correlation of –0.3: slightly negative. This means as the tuition fee goes up, the admission rate goes (slightly) down—which does indeed describe many American universities. Another way to say this is that the universities that are hardest to get into are, in general, also more expensive.

Create a scatter plot with tuition on the x axis and admission rate on

the y axis. Where do the lowest-paid graduates show up on the graph?

I asked you to create a scatter plot with the admission rate on the *y* axis and the tuition fee on the *x* axis:

```
institutions_df.plot.scatter(x='TUITIONFEE_OUT', y='ADM_RATE')
```

We can see that the plot (overall) starts in the top left and moves toward the bottom right. This aligns with our numeric correlation finding that higher admission rates are associated with lower tuition and vice versa.

However, I was intrigued by the fact that the college scorecard includes a column `MD_EARN_WNE_P10`, which shows the median income for graduates from each school 10 years following graduation. This allows us to ask and answer a number of different questions. For example, we have now seen that more-expensive schools are also harder to get into. However, is there a tangible benefit for that additional cost? Specifically, if you attend a

more exclusive school, can you expect to earn more after graduation?

I thus asked you to modify this scatter plot, colorizing it using the Spectral colormap and drawing on the values in the `MD_EARN_WNE_P10` column (figure 13.4):

```
(
    institutions_df
    .plot.scatter(x='TUITIONFEE_OUT',
                  y='ADM_RATE',
                  c='MD_EARN_WNE_P10',
                  colormap='Spectral')
)
```

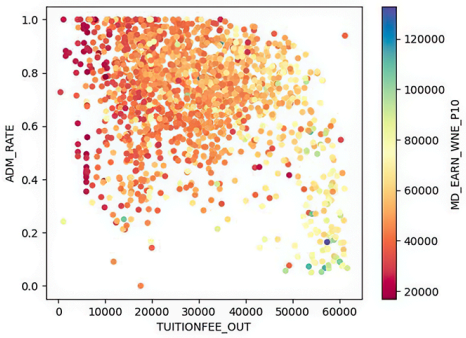

Figure 13.4 Scatter plot comparing tuition fees and admission rates

The Spectral colormap puts earnings of $20,000/year in red, $120,000/year in blue, and everything else in between. The closer to blue (darkest gray in bottom-right corner) the dots are colored, the higher the income. It's not a

huge surprise that we see a great deal of red (darkest gray) in the top-left corner (i.e., less expensive, lower-admission schools with lower earnings), whereas yellows, greens, and blues are in the lower-right corner (i.e., more expensive, higher-admission schools with higher earnings). On average, it would seem, graduates from more exclusive schools do earn more.

Which universities are in the top 25% of tuition and also the top 25% with Pell grants?

I decided to probe expensive, exclusive schools further. First, I asked you to find schools that charge in the top 25% of tuition (i.e., the most expensive universities) that are also in the top 25% of schools offering Pell grants. Pell grants are awarded to students on the basis of financial need and provide a rough estimate of how many less-wealthy students are studying somewhere. We're thus looking to find, in simple terms, expensive schools that have a relatively high proportion of nonwealthy students.

To do that, we need to use `quantile(0.75)` on the `TUITIONFEE_OUT` column to determine the top quartile of tuition. We similarly need to run `quantile(0.75)` on the `FTFTPCTPELL` column, which contains the percentage of Pell-grant recipients at each school. We can then compare each institution's value for `TUITIONFEE_OUT` and `FTFTPCTPELL` against that 0.75 quantile, retrieving institutions that are above those thresholds in both:

```
(
    institutions_df
    .loc[(institutions_df['TUITIONFEE_OUT'] >
          institutions_df['TUITIONFEE_OUT'].quantile(0.75)) &
         (institutions_df['FTFTPCTPELL'] >
          institutions_df['FTFTPCTPELL'].quantile(0.75))]
)
```

This returns all rows in `institutions_df` where both `TUITIONFEE_OUT` and `FTFTPCTPELL` are above the 75th percentile. But we aren't really interested in all the columns; I asked you to show the institution name along with its city and state. For that, we need to include a column selector in our call to `.loc`:

```
(
    institutions_df
    .loc[(institutions_df['TUITIONFEE_OUT'] >
          institutions_df['TUITIONFEE_OUT'].quantile(0.75)) &
         (institutions_df['FTFTPCTPELL'] >
          institutions_df['FTFTPCTPELL'].quantile(0.75)),
        ['INSTNM', 'CITY', 'STABBR']]
)
```

Finally, I asked you to sort the output by institution name, which we can do by calling `sort_values` and specifying the `INSTNM` column:

```
(
    institutions_df
    .loc[(institutions_df['TUITIONFEE_OUT'] >
          institutions_df['TUITIONFEE_OUT'].quantile(0.75)) &
         (institutions_df['FTFTPCTPELL'] >
          institutions_df['FTFTPCTPELL'].quantile(0.75)),
        ['INSTNM', 'CITY', 'STABBR']]
    .sort_values(by='INSTNM')
)
```

In how many institutions does the bottom quintile receive money?

Now let's look at university tuition from another perspective. The college scorecard tracks the net price for four-year public and private institutions (`NPT4_PUB` and `NPT4_PRIV`, respectively). It then breaks the tuition payments down even further

in additional columns, showing (for example) the average price paid by people in the lowest income bracket at public (`NPT41_PUB`) and private (`NPT41_PRIV`) universities.

At how many institutions, both public and private, does the average lowest-income-bracket student receive money rather than spend money? If we were merely interested in public institutions, we could find all those where the value of `NPT41_PUB` is less than 0 and then show their names:

```
(
    institutions_df
    .loc[((institutions_df['NPT41_PUB'] < 0) |
          (institutions_df['NPT41_PRIV'] < 0)),
    'INSTNM']
    .count()
)
```

Or if we were interested in private institutions, we would do the same for `NPT41_PRIV`:

```
institutions_df.loc[institutions_df['NPT41_PRIV'] < 0,
    'INSTNM'].count()
```

We could use `|` for an “or” condition, thus getting the

values where either of these is less than 0:

```
institutions_df.loc[((institutions_df['NPT41_PUB'] < 0) |
                     (institutions_df['NPT41_PRIV'] < 0)),
                     'INSTNM'].count()
```

This gave me an answer of 12. However, there's another way to do this: we can add the values in `NPT41_PRIV` to those in `NPT41_PUB` with a `fill_value` of 0. Then we can simply check to see where `NPT41_PUB` is < 0:

```
institutions_df.loc[institutions_df['NPT41_PUB'].add(
    institutions_df['NPT41_PRIV'], fill_value=0) < 0,
    'INSTNM'].count()
```

This gives the same answer. Although I'm not convinced it's a better way to solve the problem, it shows that in pandas, there's always more than one option, and they often look different from one another.

What is the average proportion that the bottom quintile pays versus the top quintile at public universities?

I then asked you to show, for public universities, the

average proportion that the bottom quintile pays versus the top quintile. To calculate this, we divide `NPT41_PUB` (the bottom quintile) into `NPT45_PUB` (the top quintile) and then take the mean:

```
(institutions_df['NPT41_PUB'] / institutions_df['NPT45_PUB']).mean()
```

We get a result of about 52%.

What is the average proportion that the bottom quintile pays versus the top quintile at private universities?

We can then repeat this for private universities:

```
(institutions_df['NPT41_PRIV'] / institutions_df['NPT45_PRIV']).mean()
```

It turns out that people in the bottom quintile at private universities pay about 71% of what the top quintile do. So not only do students pay more to attend private universities, but the poorest of them pay a higher percentage of their tuition fees than their public-university counterparts.

In looking at this data, we've seen that, overall, the schools with the highest-paid alumni

are also the most expensive and the hardest to get into. But of course, that's only overall, in the aggregate.

For which schools in the cheapest 25% do their students have the top 25% of salaries 10 years after graduation?

To try to figure out which universities offer the best overall ROI, I asked you to find the schools whose tuitions are in the lowest 25% but whose 10-year alumni are in the highest 25% of salaries. First, let's look at public institutions:

```
(
    institutions_df
    .loc[(institutions_df['NPT4_PUB']
          <= institutions_df['NPT4_PUB'].quantile(0.25)) &
         (institutions_df['MD_EARN_WNE_P10']
          >= institutions_df['MD_EARN_WNE_P10'].quantile(0.75)),
        ['INSTNM', 'STABBR', 'CITY']]
    .sort_values(by=['STABBR', 'CITY'])
)
```

This query is a variation on what we've already done, looking for those public universities whose tuition is in the lowest quartile but whose 10-year alumni are earning in the highest quartile. In our column selector, we ask for

only three columns: institution name, state, and city. That allows us to sort the results first by state and then by city. The result is a data frame with 22 rows whose universities are in California, Florida, New York, Texas, and New Mexico.

How about private institutions?

What about private universities? We can run a similar query but using `NPT4_PRIV` rather than `NPT4_PUB`:

```
(
    institutions_df
    .loc[(institutions_df['NPT4_PRIV']
          <= institutions_df['NPT4_PRIV'].quantile(0.25)) &
         (institutions_df['MD_EARN_WNE_P10']
          >= institutions_df['MD_EARN_WNE_P10'].quantile(0.75)),
    ['INSTNM', 'STABBR', 'CITY']]
    .sort_values(by=['STABBR', 'CITY'])
)
```

This query returns 30 universities spread across a variety of states. Some well-known universities (e.g., Harvard, Stanford, and Princeton) are in there, along with smaller and lesser-known ones.

Is there a correlation between admission rates and completion

rates?

Next, I wanted to know if we could find any correlation between admission rates and completion rates. That is, if a school is highly selective, are its students more likely to graduate? We run the following query:

```
institutions_df[['C100_4', 'ADM_RATE']].corr()
```

Sure enough, we see a moderate negative correlation. That is, a school that accepts more people has a lower graduation rate. That shouldn't be a huge surprise; after all, for a school to accept more people, it likely has to take people who are bigger risks in terms of not finishing.

Ten years after graduating, from what kinds of schools do people earn, on average, the greatest amount?

Next, I asked whether, on average, people earn more after graduating from a public or private university. That is, on average, how much do people earn for each value of the `CONTROL` column? Once

again, we join `institutions_df` with `fields_of_ study_df`—but only after doing a `groupby` on `fields_of_study_df`:

```
(
    institutions_df[['OPEID6', 'MD_EARN_WNE_P10']]
    .set_index('OPEID6')
    .join(fields_of_study_df
          .groupby('OPEID6')['CONTROL'].min())
    .groupby('CONTROL')
    .mean()
)
```

The result aligns with my expectations: that people who attend private for-profit universities end up earning less than those who attend public universities, who in turn end up earning less than those who attend private universities. Obviously, this is an aggregate measure—and I definitely know high earners who attended public universities and low earners who attended private ones. But data analytics is all about making generalizations, drawing conclusions that are incorrect for any individual but correct for the overall population.

Do people who graduate from "Ivy Plus" schools earn more than the average private-school university graduate?

Let's take this question of private universities to an extreme. People often want to get into the best-known universities on the assumption that they'll be able to earn more later on in life. Is this true? Does going to a famous, exclusive university mean you'll have a more lucrative career? I asked you to check the mean salary for graduates from what are sometimes known as "Ivy Plus" schools: the Ivy League as well as MIT, Stanford, and the University of Chicago.

To do this, we use `isin` in the column selector. Note that the universities' formal names are tricky to figure out, especially for "Columbia University in the City of New York." But here's the final query:

```
ivy_plus = ['Harvard University',
            'Massachusetts Institute of Technology',
            'Yale University',
            'Columbia University in the City of New York',
            'Brown University',
            'Stanford University',
            'University of Chicago',
```

```
              'Dartmouth College',
              'University of Pennsylvania',
              'Cornell University',
              'Princeton University']

(
    institutions_df
    .loc[institutions_df['INSTNM'].isin(ivy_plus),
         'MD_EARN_WNE_P10']
    .mean()
)
```

The answer to this query is just over $91,806/year, more than twice the average salary earned by all graduates of private universities—which was, as we saw, greater still than the amount earned by graduates of public or for-profit institutions.

Do people studying at universities in particular states earn, on average, more after 10 years?

Finally, we want to compare post-graduation salaries, 10 years out, by state. That is, do your future earnings depend in part on the state in which you studied? For starters, we perform a `groupby` on the states (`STABBR`), looking at the mean salary of 10-year graduates:

```
institutions_df.groupby('STABBR')['MD_EARN_WNE_P10'].mean()
```

This gives the overall answer we want, but understanding such data is always easier when it's sorted. I thus asked you to sort the values in descending order:

```
(
    institutions_df
    .groupby('STABBR', observed=True)
    ['MD_EARN_WNE_P10'].mean()
    .sort_values(ascending=False)
)
```

Create a bar plot for the average amount earned, per state, sorted by ascending pay

Now I asked you to create a bar plot from the per-state salary averages (figure 13.5):

```
(
    institutions_df
    .groupby('STABBR', observed=True)
    ['MD_EARN_WNE_P10'].mean()
    .sort_values()
    .plot.bar(figsize=(20,10))
)
```

By sorting the values, we get (I believe) a more aesthetically pleasing, easy-to-read plot than would otherwise be the

case. We can easily see that there is a big difference between how much people earn after graduating from schools in Massachusetts and Rhode Island as opposed to Arkansas and Mississippi. However, before we make a claim regarding the quality of universities in these respective states, we have to determine how many people still live in the states where they studied. After all, the cost of living in New England is significantly higher than in Arkansas and Mississippi, so it stands to reason that people living there will earn more—regardless of what university they attended.

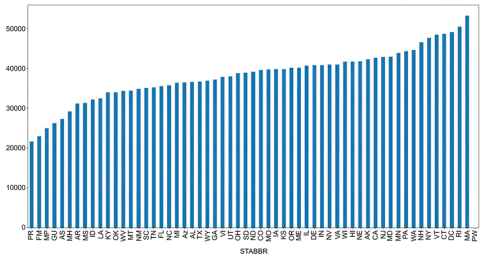

Figure 13.5 Bar chart showing the average amount earned per state

Create a boxplot for the earnings by state.

Finally, I asked you to create a box plot based on the per-state salary data so we can easily see the spread in visual form (figure 13.6):

```
(
    institutions_df
    .groupby('STABBR', observed=True)
    ['MD_EARN_WNE_P10'].mean()
    .plot.box()
)
```

The plot shows that most annual salaries are between $25,000 and $50,000, with the median being just under $40,000.

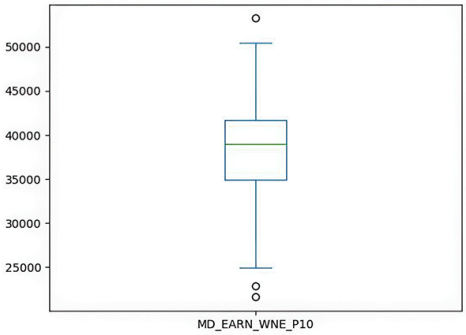

Figure 13.6 Boxplot showing the average salaries earned by graduates

ummary

You've now come to the true and actual end of the book. Thanks for joining me on this journey. I hope the exercises in this book, including all the extra "Beyond the exercise" questions, have helped improve your understanding of pandas and how to load, clean, and analyze data in a variety of ways.

Beyond the specific techniques I've covered in this book, I hope you've also begun to internalize the pandas perspective on data analysis. Pandas is a huge (and constantly growing) library, and there's no way for someone to know all of it. Understanding how pandas works means when you're faced with a new problem, you can guess how to solve it, even predicting what methods pandas will provide to do so.

I wish you the best of success in your use of pandas to analyze data in whatever you're doing. And I hope this book helped you to advance your skills in that area. If it did, please drop me a line at reuven@lerner.co.il! I'm always delighted to hear from people who have read my books.