8.2.3

Extract the Crowdfunding Data

To get started on the ETL project, you need to write code to extract the data from the Excel worksheets and assemble that data in DataFrames. Then, you'll want to explore that data. Britta happens to mention that using Pandas with Jupyter Notebook is fantastic for exploring data. Although you recall how to import and read files by using Python, you're not completely sure how to do so by using Pandas. So, you do an internet search for "how to read Excel worksheets using Pandas" to get some help.

But before writing the code, you need to activate your coding environment. If you're running macOS, go to the next subsection, "Activate Your Coding Environment on macOS," for instructions. If you're running Windows, skip to the "Activate Your Coding Environment on Windows" section.

Activate Your Coding Environment on macOS

To activate your coding environment on macOS, complete the following steps:

1. Open a new terminal window.
2. Navigate to your class folder.
3. Activate the PythonData conda environment.
4. Start the Jupyter Notebook server.

REWIND

The command to activate the PythonData environment is `conda activate PythonData`. The command to start the Jupyter Notebook server is `jupyter notebook`.

Activate Your Coding Environment on Windows

To activate your coding environment on Windows, complete the following steps:

1. Open the Anaconda Prompt for the PythonData conda environment.
2. Navigate to your class folder.
3. Start the Jupyter Notebook server.

REWIND

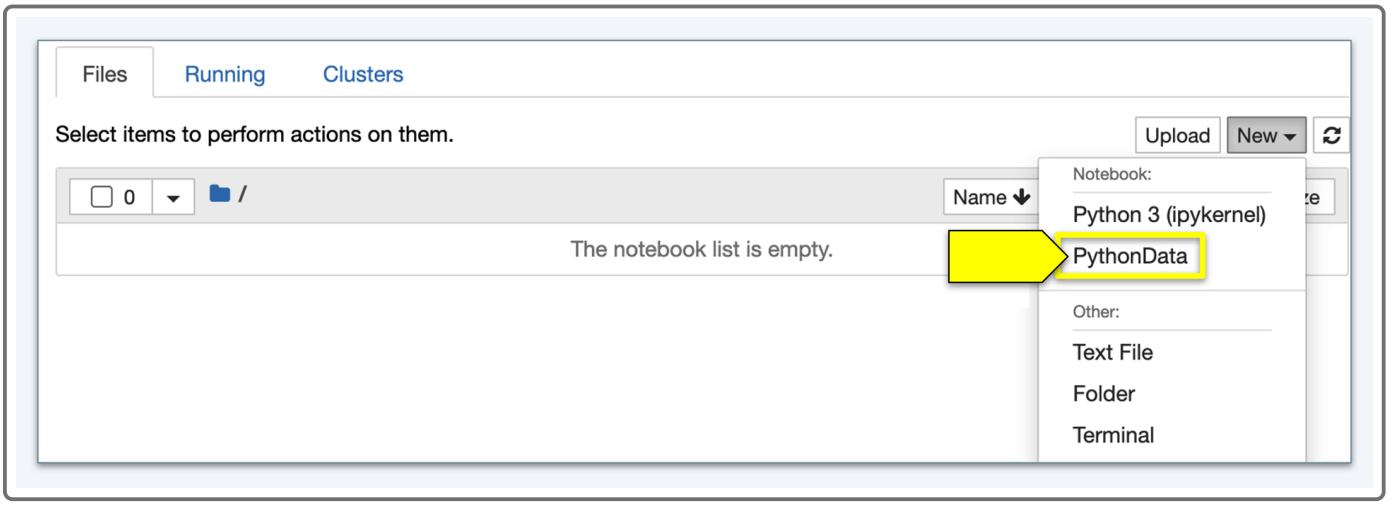
The command to start the Jupyter Notebook server is `jupyter notebook`.

Create a New PythonData Notebook

Once Jupyter Notebook is up and running, create a new PythonData notebook.

REWIND

To create a new PythonData notebook, on the New drop-down menu on the Jupyter page, select PythonData (which appears under "Notebook").



Import the Dependencies

Now that your notebook is up and running and you've downloaded the crowdfunding data, you can start writing the code. The first thing to do is import the dependencies.

REWIND

If you'll use any Python dependencies, it's best to import them all at the beginning. If you learn that you need more dependencies as you're writing your code, you should then add them to the `import` statements at the beginning of the code.

First, import the Pandas dependency, by adding the following code in the first cell:

```
import pandas as pd
```

Extract the Worksheet Data into DataFrames

Because the crowdfunding data already exists in a flat-file format, you just need to read each worksheet from the Excel file into its own Pandas DataFrame.

To do so, you first need to read the whole Excel file into a temporary DataFrame so that you can get the two sheet names, as the following code shows:

```
# Read the data into a Pandas DataFrame
crowdfunding = pd.ExcelFile("crowdfunding.xlsx")
# Get the sheet names.
crowdfunding.sheet_names
```

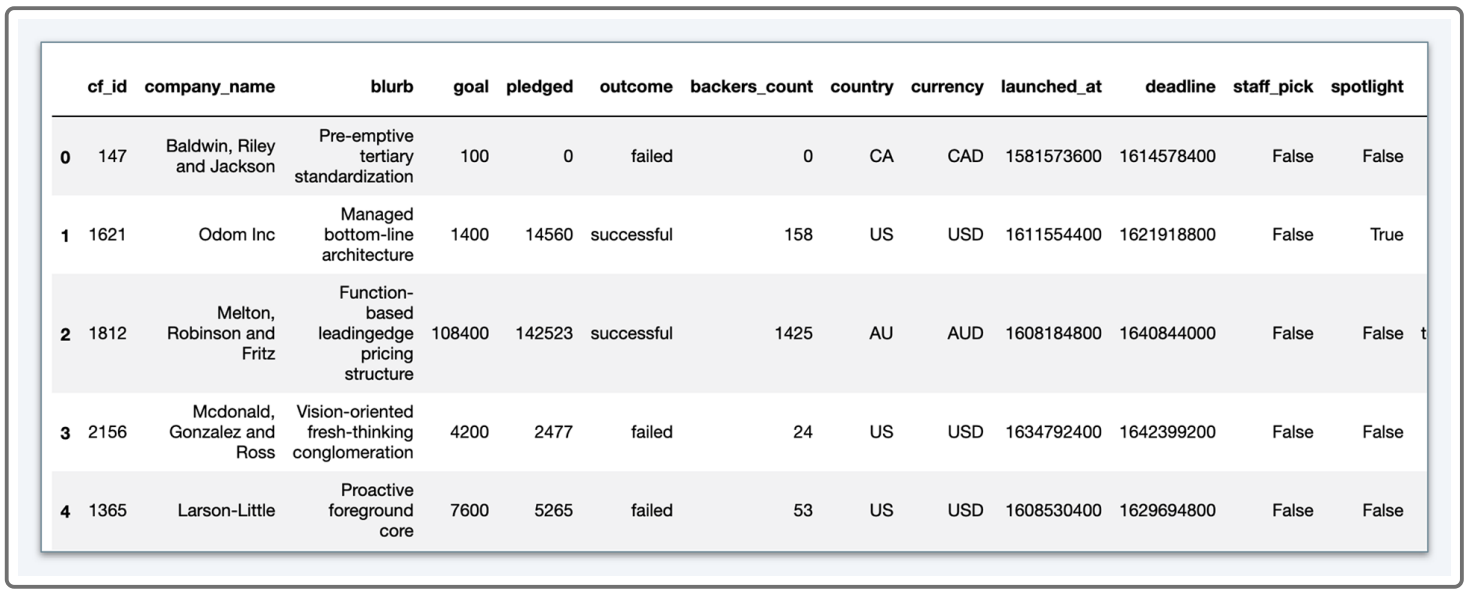
Running the preceding code produces the following output:

```
['crowdfunding_info', 'contact_info']
```

Now that you have the sheet names, you can read each into its own DataFrame. So, you first read the `crowdfunding_info` sheet into a new DataFrame, as the following code shows:

```
rowdfunding_info from the crowdfunding_info worksheet.
g_info_df = pd.read_excel(crowdfunding_data, sheet_name='crowdfunding_info')
g_info_df.head()
```

The following image shows the output from running the preceding code:



	cf_id	company_name	blurb	goal	pledged	outcome	backers_count	country	currency	launched_at	deadline	staff_pick	spotlight	
0	147	Baldwin, Riley and Jackson	Pre-emptive tertiary standardization	100	0	failed	0	CA	CAD	1581573600	1614578400	False	False	
1	1621	Odom Inc	Managed bottom-line architecture	1400	14560	successful	158	US	USD	1611554400	1621918800	False	True	
2	1812	Melton, Robinson and Fritz	Function-based leadingedge pricing structure	108400	142523	successful	1425	AU	AUD	1608184800	1640844000	False	False	t
3	2156	Mcdonald, Gonzalez and Ross	Vision-oriented fresh-thinking conglomeration	4200	2477	failed	24	US	USD	1634792400	1642399200	False	False	
4	1365	Larson-Little	Proactive foreground core	7600	5265	failed	53	US	USD	1608530400	1629694800	False	False	

In the preceding image, notice that each column name and each row matches the `crowdfunding_info` worksheet from the `crowdfunding.xlsx` file.

Next, you read the `contact_info` worksheet into a new DataFrame, as the following code shows:

```
# Get the contact_info from the contact_info worksheet.
# Increase the width of the column.
pd.set_option('max_colwidth', 400)
contact_info_df = pd.read_excel(crowdfunding, sheet_name='contact_info', hea
contact_info_df.head()
```

Let's go over the preceding code.

- The `pd.set_option('max_colwidth', 400)` line sets the width of each column to 400 pixels. That's to help us view all the data in the column.
- The next line includes the `header=2` parameter. Why are we using this? Recall that on the `contact_info` worksheet, the first two rows have information about the data on the sheet.
- Then come two blank rows, a header row that's labeled "contact_info," and six rows of data. The `header` parameter uses list indexing, where the index of the first row is 0, the index of the second row is 1, and so on. This means that the row that's labeled "contact_info" is the fourth row, which gives us `header=3`.

The following image shows the output from running the preceding code:

	contact_info
0	{"contact_id": 4661, "name": "Cecilia Velasco", "email": "cecilia.velasco@rodrigues.fr"}
1	{"contact_id": 3765, "name": "Mariana Ellis", "email": "mariana.ellis@rossi.org"}
2	{"contact_id": 4187, "name": "Sofie Woods", "email": "sofie.woods@riviere.com"}
3	{"contact_id": 4941, "name": "Jeanette Iannotti", "email": "jeanette.iannotti@yahoo.com"}
4	{"contact_id": 2199, "name": "Samuel Sorgatz", "email": "samuel.sorgatz@gmail.com"}

In the preceding image, notice that there is one column, "contact_info", and each row contains a Python dictionary with three keys—`contact_id`, `name`, and `email`—and values for each key.

Congratulations! You just completed the extract phase of the ETL project.

Next, we'll explore the two crowdfunding DataFrames to determine what data types we have and how to proceed with the transform phase.

Explore the Crowdfunding DataFrames

Now that you've extracted both the crowdfunding data and the contact info data, Britta wants to know what the data types of the columns in each DataFrame are, whether any null values exist. You'll both need this information for the upcoming transform phase.

You've already inspected the two DataFrames by using the `head()` method to verify that the data was correctly loaded into those DataFrames.

Before moving on, practice further verifying that the data was correctly loaded by completing the following Skill Drill:

SKILL DRILL

Use the `tail()` method to verify that the data at the end of each DataFrame was correctly loaded. Even if it was, errors might still have occurred in the middle. So, a best practice is to randomly sample a few rows by using the `sample()` method. For a DataFrame named `df`, `df.sample(n=5)` will show five random rows from the dataset. Use the `sample()` method on each of the two DataFrames to further verify that the data was correctly loaded.

To get the information that Britta wants, let's get some basic information about each DataFrame. Using the `info()` method on a DataFrame returns a brief summary of the DataFrame, including the index, the data types of each column, the non-null values, and the memory usage.

First, you get a brief summary of the `crowdfunding_info_df` DataFrame, as the following code shows:

```
# Get a brief summary of the crowdfunding_info DataFrame.
crowdfunding_info_df.info()
```

The following image shows the output from running the preceding code:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 14 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   cf_id                     1000 non-null   int64 
 1   company_name              1000 non-null   object
 2   blurb                     1000 non-null   object
 3   goal                      1000 non-null   int64 
 4   pledged                   1000 non-null   int64 
 5   outcome                   1000 non-null   object
 6   backers_count             1000 non-null   int64 
 7   country                   1000 non-null   object
 8   currency                  1000 non-null   object
 9   launched_at               1000 non-null   int64 
 10  deadline                  1000 non-null   int64 
 11  staff_pick                1000 non-null   bool  
 12  spotlight                 1000 non-null   bool  
 13  category & sub-category   1000 non-null   object
dtypes: bool(2), int64(6), object(6)
memory usage: 95.8+ KB
```

In the preceding image, notice that the `crowdfunding_info` DataFrame has 1,000 non-null rows. And, it has two columns with the data type, `bool`, six columns of data type, `int64`, and six columns of data type, `object`.

Next, you get a brief summary of the `contact_info_df` DataFrame, as the following code shows:

```
# Get a brief summary of the contact_info DataFrame.
contact_info_df.info()
```

The following image shows the output from running the preceding code:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 1 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   contact_info  1000 non-null   object
dtypes: object(1)
memory usage: 7.9+ KB
```

In the preceding image, notice that the `contact_info` DataFrame has 1000 non-null rows. And, it has one column of type `object`.

Based on your exploration of the crowdfunding DataFrames, you have the data types of the columns, you know that no null values exist. Next, you'll help Britta transform the data to create new DataFrames.