



Girls Who Code At Home

Data Playground
Data Manipulation in Python

Activity Overview

Did you know that about 70% of the data generated by the internet is not used? Data can be incredibly useful to companies, that might - for instance - use data to suggest restaurants on your favorite food delivery app. But datasets are really only as valuable as the diversity of the people creating them! If we don't have women and minorities developing the tech defining our future, then the systems running our lives will be dictated by the biases of the people (largely, white men) who created them.

The field of [data science](#) is vast, connecting concepts of artificial intelligence, data mining, big data, and machine learning. According to [Glassdoor](#), data scientists are one of the most sought after jobs with an average base salary of **\$110,000!** In this activity you will learn how to draw information from a dataset in order to observe possible trends. By using **Python**, we will learn how to do basic data manipulation by filtering, modifying, removing, and performing calculations on a given dataset. In this activity we will specifically look at data around [Kickstarter](#) projects, a crowdfunding platform, to determine which category of projects are the most successful.

Learning Goals

By the end of this activity you will be able to...

- ◆ use decomposition to determine how to utilize data to solve a problem.
- ◆ store, search, delete, and modify data using Python.
- ◆ understand how to navigate Kaggle and Jupyter Notebook to explore a dataset.

Materials

- ◆ [Kaggle Kickstarter Data](#)
- ◆ [Sample Project](#)

Prior Knowledge

Before embarking on this project, we recommend that you:

- ◆ can explain what a [variable](#) is in your own words and describe how they can be used in a program.
- ◆ can explain what a [conditional statement](#) is in your own words and describe how they can be used in a program.
- ◆ can explain what a [method/function](#) is in your own words and describe how they are used in a program.
- ◆ have experience using a text-based language like JavaScript, Python, Swift, etc.

If you want a quick refresher on Python we highly suggest you check out our activity

[Can I Help You?](#)

Women in Tech Spotlight: Theresa Johnson



Image Source: [Medium](#)

A resume is a document that presents a person's skills and accomplishments. They are an opportunity to boast about your skills and let any future employer know what an amazing person you are! But, have you ever heard of a **failure resume**? On top of maintaining and updating her regular resume, Theresa Johnson documents all of her failures in a separate resume. She believes that mistakes are valuable experiences that she can learn from! You can learn more about what is a failure resume [here](#) and how she got this idea from [Tina Seelig](#) at Stanford.

Dr. Theresa Johnson is a Product Designer at [Airbnb](#), where she also worked as a data scientist. Before working at Airbnb she received her PhD in aeronautics and astronautics from Stanford University. You may be thinking, how do all of these roles and fields relate? You may not initially connect aeronautics to data at Airbnb, but the way we utilize and manipulate data at its core is the same. During her time as a data scientist, Theresa worked on analyzing metrics to help optimize review for homes in various markets. Now as a product manager, she utilizes her experience with Machine Learning and AI to understand payment data and coordinate and communicate this information across multiple teams.

As a woman of color in the tech industry, Theresa speaks out for the importance of diversity within the industry. Theresa also serves as a Chairman Of the Board at [StreetCode Academy](#), a non-profit organization that supplies free technology resources for communities of color throughout Silicon Valley. They connect students to laptops and classes in coding, entrepreneurship, and design.

Watch this [video](#) to learn more about Theresa and her work at Airbnb.

Reflect

Being a computer scientist is more than just being great at coding. Take some time to reflect on how Theresa and her work relates to the strengths that great computer scientists focus on building - bravery, resilience, creativity, and purpose.



RESILIENCE

Theresa highlights the importance of failure with her failure resume. Think of a time that you faced failure and were disappointed. How can you rewrite this experience to be more positive? What was something that you learned about yourself that you can apply to your next challenge?

Share your responses with a family member or friend. Encourage others to read more about Theresa to join in the discussion!

Step 1: What is Big Data? (5-10 mins)

About 70% of the data generated by the internet goes unused, but why? In this section we will discuss what is Big Data and why it is valuable to companies in every industry.

Big Data (2 mins)

Big Data is just as it is named, a lot of data! We distinguish big data from just normal data because big data is generated at a rate that is difficult to maintain and often requires a lot of efforts to get it “clean” or ready to use and interpret. Think about how you might text your friend a simple phrase like “ok”. You might text “okay”, “ok”, “k”, “kay”, and many more! This might also vary with how we use uppercase or lowercase letters and special characters as well. All of these possible choices created data and we need to train the computer to recognize that all of these words all mean the same thing.



Despite the sheer amount of data available, most of it may be considered unusable due to the existence of bias. What is bias? Bias in data occurs when a particular result might be more favorable to certain outcomes. This can happen for a variety of reasons, one being that the information gathered is not representative of the population as a whole. Let's say you were surveying your people at school. If you only asked the first 20 people you see, do you think your data is representative of the whole school? Not really. Not only is it important *who* or *what* you survey to represent the whole population, but also what kind of information you gather. To obtain a diverse dataset, it is important to have diversity in your population and the team who will be conducting the analysis. If you want to learn more about data bias, check out this [article](#) by Elder Research or this [video](#) by Google.

Getting Started with Kaggle (5-8 mins)



Kaggle is a website that hosts real data that is contributed by an online community. Data hosted on this website ranges from statistics on [COVID-19](#), [YouTube videos](#), [Apps on Google Play](#), [breast cancer](#), [avocado prices](#), and [even Pokémon](#)! It is a great place to explore real data that affects the decisions we make today. Within Kaggle, you can create a **notebook** associated with a particular dataset to write code within the website! This is a great way to organize your projects as well as submit work to real live projects!

- **Create a Kaggle Account.** Click this [link](#) to create an account on Kaggle. Alternatively, you can click the **Register** button at the top-right of the Kaggle website to create an account. *If you are under 13, you'll need your guardian's permission and email address to sign up.*



- **Open the [Kickstarter dataset](#).** At the top of this dataset there is a main header containing the title of the dataset, creator, and when it was last updated. Right below this several options are presented: Data, Tasks, Notebooks, Discussion, Activity, and Metadata. You can learn more about features Kaggle provides for each dataset [here](#).

Step 2: Explore the dataset (10-15 mins)

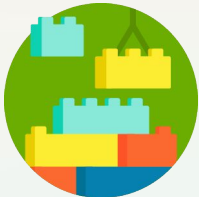
Before we get started on manipulating the data, we need to take the time to walk through the dataset we chose for this activity.

Kickstarter Projects (1 min)

[Kickstarter](#) is a crowdfunding platform where small businesses can open a project on the website to collect small investment into their project from the online community. Those looking to invest in a project make a pledge of a certain amount. In return for their pledge they are awarded gifts at the completion of their projects. You can explore some examples of projects hosted on Kickstarter [here](#).

You may be thinking, *why did we choose this dataset?* In this activity we will walk you through a basic example of some of the things you can do to manipulate data. This Kickstarter data contains values that are both [categorical](#), or data that can be grouped into categories, and numerical giving you a snapshot of how other datasets might be represented.

Breaking Down the Dataset (5-8 mins)

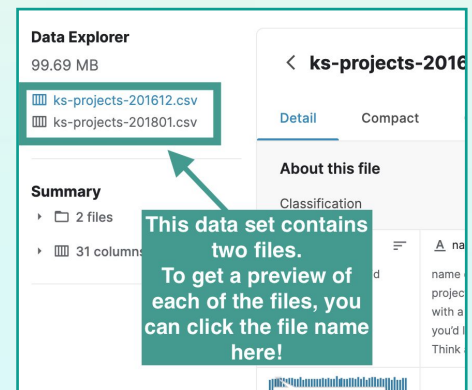


Let's dive into our data! First, let's reopen the [Kaggle Kickstarter dataset](#). Make sure that you are on the **Data** tab, you should notice that the word "Data" at the top header bar is [blue](#) and underlined. Next, scroll down to the **Data Explorer** part.

Taking a look on the left side, you may notice that there are two files, [ks-projects-201612.csv](#) and [ks-projects-201801.csv](#).


- ◆ [ks-projects-201612.csv](#): This dataset contains all Kickstarter projects that were launched before December 2016.
- ◆ [ks-projects-201801.csv](#): This dataset contains all Kickstarter projects that were launched before January 2018.

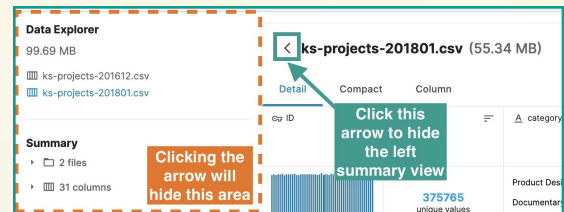
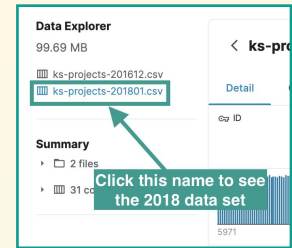
Since we want to work with the most recent data, we will only be working with [ks-projects-201801.csv](#), or for short we will call this the 2018 dataset.



Since the [ks-projects-201801.csv](#) contains all projects before January 2018, this dataset contains ALL of the data in [ks-projects-201612.csv](#) and more!

Step 2: Explore the Dataset (cont.)

- ◆ **Open the 2018 Kickstarter Dataset.** In the **Data Explorer**, click on the name **ks-projects-201801.csv** on the left side. You should then see this name highlighted in blue.
- ◆ **Expand the View.** Click the box icon  on the top-right corner of the data explorer window.
- ◆ **Hide the Summary Information.** Click the arrow to the left of the dataset name. This should close the left hand side giving even more space for us to look at the data!



This view gives us a quick snapshot of the data and some of the values inside. Note, this does not display the full dataset because it is very large! In fact, the entire dataset contains information for 375,765 Kickstarter projects! The *columns* of this dataset are called **features** of the data. **Features** let us know what type of information is captured for each project. Each row of the dataset represents a single Kickstarter project, or an **entity**. For example, if we had a dataset on humans, a person would be an *entity* in the data set and some *features* we might include in this data set would be a person's name, eye color, hair color, date of birth, etc.

Example

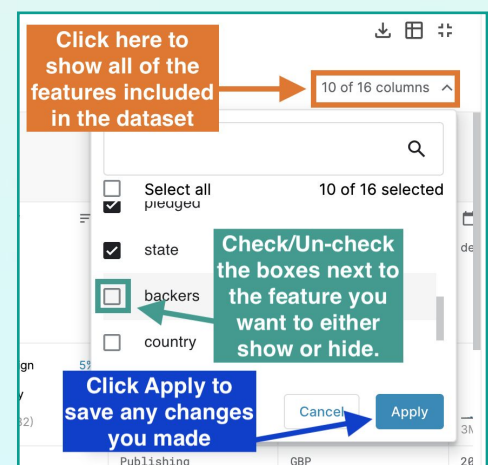
Name	Eye Color	Hair Color	Date of Birth
Reshma Saujani	Brown	Brown	November 18, 1975

Example of a single **entity**, Reshma Saujani, and some of the **features** that define her.

Kickstarter Project Features (1 min)

Let's breakdown the **features**, or columns, in this Kickstarter data! This dataset has a total of 15 features: **ID, name, category, main_category, currency, deadline, goal, launched, pledged, state, backers, country, usd pledged, usd_pledged_real, usd_goal_real**. The first row tells us the name of the feature and a brief description while the second row provides a visual snapshot of the some of the values for each feature. Take a minute to review each of the features and their short description.

Note: In the data explorer you can only see 10 of the 15 columns, there is a drop-down menu on the top-right corner of the dataset that allows you to view additional columns. You will need to scroll down in the menu to find the additional 5 columns that are not shown in the default view.



Decomposing the Problem (5-10 mins)

Now that we understand a little bit about the dataset and the information given to us, what do we do now? From here, data scientists brainstorm about questions they want to answer with this dataset. Some possible questions we might consider are:



- ◆ *Is the relationship between the length of a project and if it will be a success?*
- ◆ *Is there a relationship between the number of backers (people who pledge) and if it will be a success?*
- ◆ *Can we figure out if a project will be a success based on it's current pledges?*

There are many more questions that we can answer with this dataset. If you want to explore what others have done with this dataset, feel free to explore the [notebooks](#) for this dataset on Kaggle. The question we will be exploring in this activity is:

WHICH MAIN CATEGORY OF PROJECTS HAS THE HIGHEST PERCENTAGE OF SUCCESS?



In this next part we will walk you through how to break down this question by answering a series of questions. This process of breaking big problems into smaller parts is called **problem decomposition**. Computer scientists often use this method to help understand the problem better and figure out in smaller sections how to go about solving it.

QUESTION 1

Which features will we need to use to answer our question?

QUESTION 2

How should we calculate the percentage of success?

QUESTION 3

What tasks will we need to do in order to answer this question?

Feel free to **pause** here to brainstorm some ideas on how you would answer these questions before taking a look at our solutions on the next page.

Step 2: Explore the Dataset (cont.)

Question 1: Which features will we need to use to answer our question?

Features included in the dataset: ID, name, category, main_category, currency, deadline, goal, launched, pledged, state, backers, country, usd pledged, usd_pledged_real, usd_goal_real.

Since our question focuses around finding the percentage of success of projects based on its main category, the features we want to use are **state** and **main_category**.

- ◆ We want to use **main_category** instead of **category** because the **main_category** feature contains only a set number of choices and more general. Since category is more specific we might get categories with only a few projects that are associated, giving us inaccurate results.
- ◆ We use the **state** feature because this is the only feature that informs us whether or not a project was successful.

Question 2: How should we calculate the percentage of success?

In order to calculate the percentage, we must compare the number of successes to something. Since we are concerned about the success per main category, we need to calculate success rate as follows:

$$\% \text{ of success for category}_{\text{film+video}} = \frac{\# \text{ of successful projects in category}_{\text{film+video}}}{\# \text{ of total projects in category}_{\text{film+video}}} \times 100\%$$

In the example above we show how to calculate the % of success for projects in Film & Video. We will want to calculate this percentage for **each category** in our dataset. While the number of projects in each category may differ, the equation should remain the same.

Question 3: What tasks will we need to do in order to answer this question?

When breaking down a problem, you want to make sure that your tasks/steps are small and manageable. It is okay if you do not know how to solve some of your sub steps yet, but the point is to have a starting point to being to solve your larger problem. Here are some of the smaller steps that we want to solve in order to answer our guiding question.



Step 3: Intro to Python & Pandas (20-25 mins)

Data scientists use a number of different tools and languages to process and analyze data. Some programming languages include [R](#), [Structured Query Language \(or SQL\)](#), and [Python](#). In this activity we will focus on using the Python programming language to help us manipulate our data with the help of a Python library, [Pandas](#).

Python (2 mins)



Python is a text-based programming language which means that all commands will need to be typed! Many programmers choose to use Python because it is easy to learn and understand. Python is an **open source** language, meaning that it is freely available for the public to use and modify as necessary. There are strict guidelines on how updates are accepted and implemented to the language, but essentially anyone can contribute to the evolution of the language!

Since Python is an open source language, this has led to the community to develop additional libraries. A **library** is a collection of commands and variables. A library can make writing code easier since we can just use the commands in a library to do an action instead of writing several lines of code to do the same thing. In this activity, we will be using the **pandas** library. This special library has been made specifically for data scientists to analyze datasets easily without having to write a lot of lines of code to do simple actions like search, filter, compare, modify, and remove information from a dataset. Before we start diving into how we will use **pandas** specifically to do actions on our data, let's set up our programming environment in Kaggle by creating a new notebook.

Creating a New Notebook (5-8 mins)

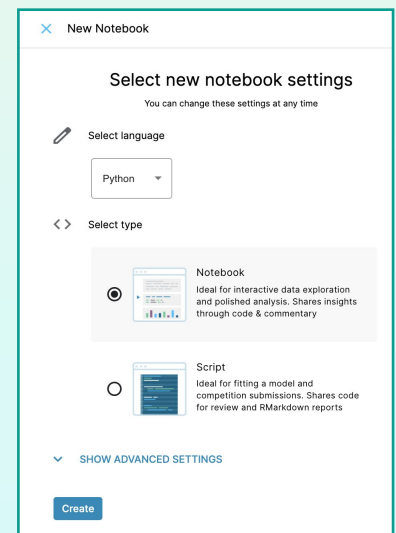
Let's go back to our [Kickstarter dataset](#).

- **Create a New Notebook.** Click the **New Notebook** button under the main header image to the right. This should bring you to a screen to select new notebook settings. Be sure that your notebook has the following:

- ◆ **Language:** Python
- ◆ **Type:** Notebook

Your dataset may still be displaying in full-view. To **minimize** your dataset, click the **box icon** on the top-right corner.

Confirm Settings and Click **Create**.



If you have programmed in Python before, this “notebook” might look different from programming in Trinket or other Python editors. Kaggle uses a tool called [Jupyter Notebook](#) to program in Python. Jupyter Notebook is a tool used application that can be used to show both code, text, and visualization all in one place. It is easy to run blocks of code without having to run the entire program.

- **Rename Your Notebook.** To do this, click the default name at the top-left of your screen and replace the text with your new title. Your title should reflect something similar to the question you want to explore. For example, you might choose Success Rate and Categories for Kickstarter Projects. You may also choose to include your name in the title as well, be sure to only display either initials or your first name and last initial.

Step 3: Intro to Python & Pandas (cont.)

Exploring the Starter Code (3-5 mins)

Let's take a moment to take a look at some of the starter code that is included in your notebook. You may notice a few lines of code starting with the `#` symbol throughout the starter code. These lines of code are **code comments**. These are used by programmers to help organize their code making it more readable. In Python, any text written after a `#` symbol is considered a code comment and it is then colored in teal.



Next, let's look at the first lines of code, starting with the keyword `import`.

PYTHON	DESCRIPTION
<pre>import numpy as np import pandas as pd</pre>	<ul style="list-style-type: none">◆ import: This keyword lets the computer know that we are using a Python library.◆ as: This keyword gives a nickname to the package. This is an optional step but can make programming easier.◆ numpy/np: This Python library is used to do mathematical operations on the dataset. We have given this package a nickname of np.◆ pandas/pd: This Python library is used to convert the dataset into a more usable format for analysis. We have given this package a nickname of pd.

We will not be using the `numpy` library in this activity. As you continue to explore data analytics on your own, feel free to learn more about how data scientists use the `numpy` library to enhance their analysis of a dataset.

Finally, let's take at the last lines of code.

PYTHON	DESCRIPTION
<pre>import os for dirname, _, filenames in os.walk('/kaggle/input'): for filename in filenames: print(os.path.join(dirname, filename))</pre>	<p>One of the best features of the notebook in Kaggle is that you can use the dataset easily without having to do additional steps. How? Kaggle already associates the dataset you are interested in with your notebook! These optional lines of code helps let us know what the filename is for our dataset.</p>

Step 3: Intro to Python & Pandas (cont.)

Running Code (2 mins)

First, click anywhere inside the code block. Click the **blue play button** ► that appears to the left of the window. This should run all of the lines of code inside the code block and display any output right below the window. When you run these lines of code you should get the following output:

```
/kaggle/input/kickstarter-projects/ks-projects-201801.csv  
/kaggle/input/kickstarter-projects/ks-projects-201612.csv
```

These are the file names of our dataset! Now that we understand a little bit about what is included in the starter code and how to run code in our notebook, let's start coding!

Importing the Kickstarter Dataset (10-15 mins)

Now that we know the file names of our dataset, we need to actually import the data into our program. Right now it lives as a separate file, but we need to connect this information with our Python program. To do this, we will use the `read_csv()` method from the `pandas` library to help us. Remember that a method/function is a set of instructions (lines of code) that performs a specific task. Let's break down the syntax of this method.

CSV, short for comma separated values, is a type of file used to hold data in plain text. The `pandas` library can read CSV files easily and convert it into a table-like format for us to easily read and manipulate.

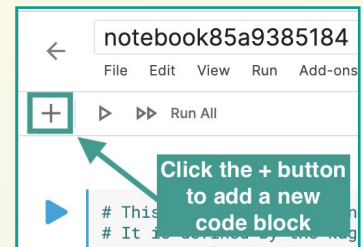
PYTHON	DESCRIPTION
<code>pd.read_csv("filename")</code>	<ul style="list-style-type: none">◆ pd: This keyword lets the computer know that we are using a method from the <code>pandas</code> library. We use <code>pd</code> instead of <code>pandas</code> because this is the nickname we gave the library when importing the library at the start of our program.◆ <code>.</code>: This symbol lets the computer know that we are using a method.◆ <code>read_csv()</code>: This method in <code>pandas</code> reads in a CSV file and converts it to a table-like format so that it is easier to use in Python.◆ "filename": We need to tell the computer which file to open! Add the filename to the CSV file here. We include the name in quotation marks (can be single or double quotes) since it is a name.

Step 3: Intro to Python & Pandas (cont.)

- **Add a New Code Block.** Code blocks are a great way to organize your code to map to the subproblems we defined when we decomposed our problem. You may already have an empty code block in your notebook, indicated with a light-gray box with the symbol [] to the left of the box.

There are two options to add a new code block:

- ◆ Click the **+** button at the top menu of the notebook.
- ◆ Hover your mouse below your last code block. You should notice an option that reads **"+ Code"**. Select this button to create a new code block below.



- **Import the Kickstarter dataset.** Now that we have a new code block, it's time to use the `read_csv()` method to import our dataset. But wait, we need the file name of our dataset! Remember that after running the first block, this outputs the file names of our dataset. In fact, this outputted two names, the 2016 and 2018 dataset. We will only use the 2018 Kickstarter dataset since this is the most recent information. In your new code block, use the `read_csv()` method to import the 2018 dataset. **Copy and paste** the 2018 dataset file name from the output of your previous code block.

```
pd.read_csv("/kaggle/input/kickstarter-projects/ks-projects-201801.csv")
```

- **Run your code block.** Click the blue play button to the left of the code block to run your code. You should see a small snapshot (similar to what you saw in the Data Explorer) of your dataset.

RESULTS

	ID	name	category	main_category	currency
0	1000002330	The Songs of Adelaide & Abullah	Poetry	Publishing	GBP
1	1000003930	Greeting From Earth: ZGAC Arts Capsule For ET	Narrative Film	Film & Video	USD
2	1000004038	Where is Hank?	Narrative Film	Film & Video	USD
3	1000007540	Tosh/Capital Rekordz Needs Help to Complete Album	Music	Music	USD
4	1000011046	Community Film Project: The Art of Neighborhood...	Film & Video	Film & Video	USD
...
378656	999976400	ChknTruk Nationwide Charity Drive 2014 (Canceled)	Documentary	Film & Video	USD
378657	999977640	The Tribe	Narrative Film	Film & Video	USD
378658	999986353	Walls of Remedy: New Lesbian Romantic Comedy I...	Narrative Film	Film & Video	USD
378659	999987933	BioDefense Education Kit	Technology	Technology	USD
378660	999988282	Nou Renmen Ayiti! We Love Haiti!	Performance Art	Art	USD
378661 rows x 6 columns					

DEBUGGING TIPS

- ◆ Check that you included the correct 2018 file name: `/kaggle/input/kickstarter-projects/ks-projects-201801.csv`
- ◆ Make sure that there are quotation marks around the file name
- ◆ Remember that in Python, spelling counts! Check that not only everything is spelled correctly but also in the correct case.
- ◆ Check that you include a period when calling a method of **pandas**.
- ◆ Check that you do not have extra parentheses `()`. You may notice that when you type a `(` symbol the notebook automatically adds a closing bracket `)`. This can cause you to add additional brackets by accident.

Step 3: Intro to Python & Pandas (cont.)

- **Store your dataset in a variable named `ds`.** We are almost done! Now that we got our dataset linked to our Python program, we need to store this in a variable. Remember that [variables](#) are used to store information (data) in a computer program. Store your dataset in a variable named `ds` (an abbreviation of dataset).

```
ds = pd.read_csv("/kaggle/input/kickstarter-projects/ks-projects-201801.csv")
```

- **Use the `info()` method to print out information about your dataset.** We can use the `info()` method to get a quick snapshot of the features and number of rows this dataset.

PYTHON	DESCRIPTION
<code>ds.info()</code>	<ul style="list-style-type: none">◆ ds: We use the <code>info()</code> method on our variable, <code>ds</code>, NOT the whole <code>pandas</code> library. This is because we want to know information about our specific dataset.◆ info(): This method in <code>pandas</code> gets information about the dataset, including features, rows, and data type.

- **Run your code block.** Click the blue play button to the left of the code block to run your code. This should output some information about your dataset, including the number of entries (or number of Kickstarter projects) and number of features. This also includes the [type of values](#) of each feature (number or words) and the number of entries that are “**non-null**” or not empty.

RESULTS

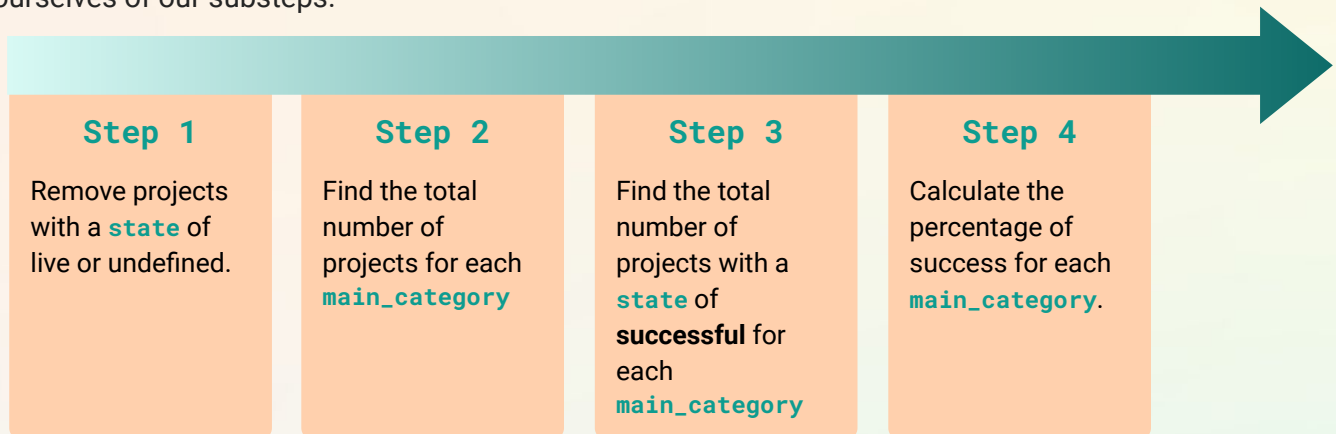
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 378661 entries, 0 to 378660
Data columns (total 15 columns):
 #   Column                Non-Null Count  Dtype  
---  -
 0   ID                    378661 non-null  int64  
 1   name                  378657 non-null  object  
 2   category              378661 non-null  object  
 3   main_category         378661 non-null  object  
 4   currency              378661 non-null  object  
 5   deadline              378661 non-null  object  
 6   goal                  378661 non-null  float64 
 7   launched              378661 non-null  object  
 8   pledged               378661 non-null  float64 
 9   state                 378661 non-null  object  
10  backers               378661 non-null  int64  
11  country               378661 non-null  object  
12  usd pledged           374864 non-null  float64 
13  usd_pledged_real      378661 non-null  float64 
14  usd_goal_real         378661 non-null  float64 
dtypes: float64(5), int64(2), object(8)
memory usage: 43.3+ MB
```

DEBUGGING TIPS

- ◆ **Name 'ds' is not defined:** Each code block in your notebook runs separately, sometimes your notebook might lose its place in the program and require you to **run all** of your code blocks. To do this, click the double arrow button `⏮` at the top of your notebook to run all code blocks.
- ◆ Remember that in Python, spelling counts! Check that not only everything is spelled correctly but also in the correct case.
- ◆ Check that you include a period when calling a method.

Step 3: Modifying the Data (25-35 mins)

Before we start calculating the success rate on our data set, we need to first make sure that our data is cleaned. Data scientists must first check for values that might cause errors in their analysis like duplicate values, spelling errors, or empty values. This process is called [data cleaning](#). Before we dive in, let's remind ourselves of our substeps.

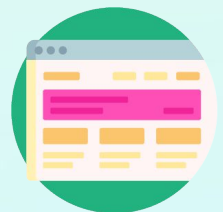


You may have noticed when you first opened the Kickstarter dataset on Kaggle, there is a **usability** score. The usability score lets us know how *clean* the data is. This dataset has a usability score of 7.9 which is very high! We still need to do additional cleaning of our dataset before we start calculating the percentage of success for our data.

Recall that in this dataset there are projects with a **state** of either success, cancelled, undefined, suspended, failed, or live. In calculating the success rate of projects, we want to focus on the projects that have already concluded. This means that we do not want to include projects that are considered **live** or **undefined** since we don't know its final outcome yet. Before we dive into how to do this, let's learn a little bit more about the structure of our dataset in [pandas](#).

Pandas DataFrame (10-15 mins)

In [pandas](#), data is stored in a table-like object called a [DataFrame](#). It stores the data similar to how we viewed it in the Data Explorer on Kaggle. A **DataFrame** mimics the data structure's rows and columns. To access the data, we use `[]` symbols. This is very similar to [Arrays](#) in JavaScript and [Lists](#) in Python.



With a total of **15 features**, or columns, we might not want to view all of this data. Since we will be focusing on just two of the features, `main_category` and `state`, let's focus our dataset to display only this information. Let's build up some of the components to *select* and display these columns.

- ➔ **Add a new code block at the bottom of your notebook.** Either hover your mouse below your last code block and press the **+Code** button or press the **+** button at the top-left of your notebook

Step 4: Modifying the Data (cont.)

- **Create a `List` with the names of the desired features.**
Using the `[]` symbols, we will add the names of the features inside the square brackets that are separated by commas.

In Python, a `List` is an ordered data structure that holds information. A lists assigns a number, or **index**, making it easy to access, delete, or replace values.

PYTHON	DESCRIPTION
<code>["main_category", "state"]</code>	<ul style="list-style-type: none">◆ <code>[]</code>: The square brackets indicate that we are creating a <code>List</code> in Python.◆ <code>"main_category", "state"</code>: We include the name of the features we are interested in. Since these are names we include quotation marks around each feature name and separate the values with a comma.

- **Using our dataset variable, `ds`, call the list of feature names.** Remember that we stored the reference to our dataset in a variable named `ds`. Here we use the `[]` symbols to indicate that we want to select information from the dataset and then also include our `List` of features *inside*.


PYTHON	DESCRIPTION
<code>ds[["main_category", "state"]]</code>	<ul style="list-style-type: none">◆ <code>ds[]</code>: We call the variable <code>ds</code> which holds the reference to our dataset and use the <code>[]</code> symbols to indicate we want to access information from our dataset.◆ <code>["main_category", "state"]</code>: We include this <code>List</code> of features to let the computer know which columns we want from our dataset. This information goes <i>inside</i> of the square brackets that appear after <code>ds</code>, the variable storing our dataset.

- **Run your code block.** Click the blue play button to the left of the code block to run your code. You should get a snapshot of all of the data in your dataset but only displaying the `main_category` and `state` features. If your code doesn't run properly, check the following:

RESULTS

	main_category	state
0	Publishing	failed
1	Film & Video	failed
2	Film & Video	failed
3	Music	failed
4	Film & Video	canceled
...
378656	Film & Video	canceled
378657	Film & Video	failed
378658	Film & Video	failed
378659	Technology	failed
378660	Art	failed
378661 rows x 2 columns		

DEBUGGING TIPS

- ◆ **Name 'ds' is not defined:** Each code block in your notebook runs separately, sometimes your notebook might lose its place in the program and require you to **run all** of your code blocks. To do this, click the double arrow button  at the top of your notebook to run all code blocks.
- ◆ Check that you use quotation marks around the names of your features.
- ◆ Check that your feature names are spelled correctly. Remember that Python is also case sensitive.
- ◆ Check that you do not have extra square brackets `[]`. You may notice that when you type a `[` symbol the notebook automatically adds a closing bracket `]`. This can cause you to add additional brackets by accident. You should have a total of two sets of brackets: one around the feature names, and the other surrounding the list of features.

Step 4: Modifying the Data (cont.)

Removing live or undefined Projects (15-20 mins)

In order to remove the projects that we don't want to include in our analysis, we will need to use [conditionals](#) in Python to let the computer know which data values we want! To prepare our dataset for our analysis, we want to remove the projects that have not been completed. This means removing the projects with a **state** of either **undefined** or **live**. Before we go about how to remove these projects, let's preview the number of projects for each state. To do this, we will use the method [value_counts\(\)](#).

→ Use the `value_counts()` method on the `state` feature.

PYTHON	DESCRIPTION
<code>ds["state"].value_counts()</code>	<ul style="list-style-type: none">◆ <code>ds["state"]</code>: We want to know the breakdown of projects based on <code>state</code>. We use the <code>[]</code> symbols to select only the <code>state</code> feature in our dataset and use double quotation marks around the feature to indicate that it is a name.◆ <code>.</code>: This symbol lets the computer know that we are using a method.◆ <code>value_counts()</code>: This method returns the number of entities (or rows) associated with each unique feature value.

→ **Run your code block.** Click the blue play button to the left of the code block to run your code. You should have a result identical to the breakdown below:

RESULTS	DEBUGGING TIPS												
<table><tr><td>failed</td><td>197719</td></tr><tr><td>successful</td><td>133956</td></tr><tr><td>canceled</td><td>38779</td></tr><tr><td>undefined</td><td>3562</td></tr><tr><td>live</td><td>2799</td></tr><tr><td>suspended</td><td>1846</td></tr></table>	failed	197719	successful	133956	canceled	38779	undefined	3562	live	2799	suspended	1846	<ul style="list-style-type: none">◆ Name 'ds' is not defined: Each code block in your notebook runs separately, sometimes your notebook might lose its place in the program and require you to run all of your code blocks. To do this, click the double arrow button ⇨ at the top of your notebook to run all code blocks.◆ Check that you use quotation marks around the names of your features.◆ Check that your feature names are spelled correctly. Remember that Python is also case sensitive.◆ Check that you do not have extra square brackets <code>[]</code>.
failed	197719												
successful	133956												
canceled	38779												
undefined	3562												
live	2799												
suspended	1846												

Step 4: Modifying the Data (cont.)

As we try to remove the **live** and **undefined** projects from our dataset, we can compare the breakdown of projects with these original values. In order to filter out the data that we don't want to include, we will need to use a combination of conditionals. Recall that [conditional statements](#) execute code **if** a condition, or a set of rules, is met. Take a moment to think about how we can phrase what we want into a **true** conditional statement. Let's break down how we would write our rule to a Python conditional statement.

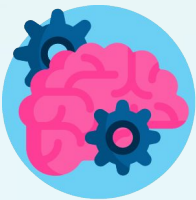
Example

RULE	PYTHON CONDITION
Ex: Projects with a successful state	<code>ds["state"] == "successful"</code>

- `ds["state"]`: We want to apply filtering to our dataset, we use our dataset variable, `ds`, and then use the square brackets `[]` to specify filtering on the `"state"` feature.
- `== "successful"`: Since we want only projects that were successful, we use the `==` symbol which searches for when the state equals `"successful"`

Take a moment to try to rewrite the remaining rules to conditional statements in Python in the table below. Need a refresher on conditional statements and/or comparison operators? Check out this [W3 School's tutorial](#) on conditionals in Python. Since we want to apply filtering to our dataset, we use our dataset variable, `ds`, and then use the square brackets `[]` to specify filtering on the `"state"` feature.

RULE	PYTHON CONDITION
Projects that do not have a state of "undefined"	<code>ds["state"]</code>
Projects that do not have a state of "live"	<code>ds["state"]</code>



Pause here before revealing our solutions on the next page. There are many ways that you can write a conditional statement to represent the rules we want. Our solution offers only one way but there are multiple solutions.

Rule #1: Projects that do not have a state of "undefined"

RULE	PYTHON CONDITION
Projects that do not have a state of "undefined"	<code>ds["state"] != "undefined"</code>

Since we want projects that do not have a state of "undefined" we use the `!=` operator. Alternatively you could have written a conditional statement that checks if the projects equals either failed, successful, cancelled, or suspended.

Rule #2: Projects that do not have a state of "live"

RULE	PYTHON CONDITION
Projects that do not have a state of "undefined"	<code>ds["state"] != "undefined"</code>

Similar to the example, we use the `!=` operator to find projects with a state that is not "live". Now that we know which conditionals to include, let's program this in our notebook!

- **Add a new code block at the bottom of your notebook.** Either hover your mouse below your last code block and press the `+Code` button or press the `+` button at the top-left of your notebook.
- **Write a condition to remove projects with a state of "undefined" and "live".** We already reviewed the conditional to filter projects without the state of "undefined" and "live" but in order to *combine* these rules in **one** conditional statement we need to use the **and** operator. We simply use the symbol `&` to represent and in pandas.

The `and` operator represented differently in pandas than in a regular Python conditional statement. Since we are using an and operator in our DataFrame we use the symbol `&`. If we were writing a conditional statement separate from the DataFrame we would use the keyword `and` instead

```
(ds["state"] != "undefined") & (ds["state"] != "live")
```

- ◆ `&`: This keyword represents "and" and helps us combine two conditions.
- ◆ `()`: We use parentheses around each of the conditionals to distinguish between the two conditions we want to check for.

Step 4: Modifying the Data (cont.)

- **Apply the conditional statement to filter the dataset.** Here we will use the variable `ds` and the symbols `[]` to apply the conditional statements from the previous step. Remember that the conditional statement must be *inside* the square brackets `[]`.

```
ds[(ds["state"] != "undefined") & (ds["state"] != "live")]
```


- **Store this new filtered dataset in a new variable named `compProj`.** We want to store this filtered dataset in a new variable so that we don't modify the original dataset. We named our variable `compProj`, short for completed projects.

```
compProj = ds[(ds["state"] != "undefined") & (ds["state"] != "live")]
```

- **Use the `value_counts()` method to see the breakdown of the projects in our filtered data.**

```
compProj["state"].value_counts()
```

- **Run your code block.** Click the blue play button to the left of the code block to run your code. You should get the following results:

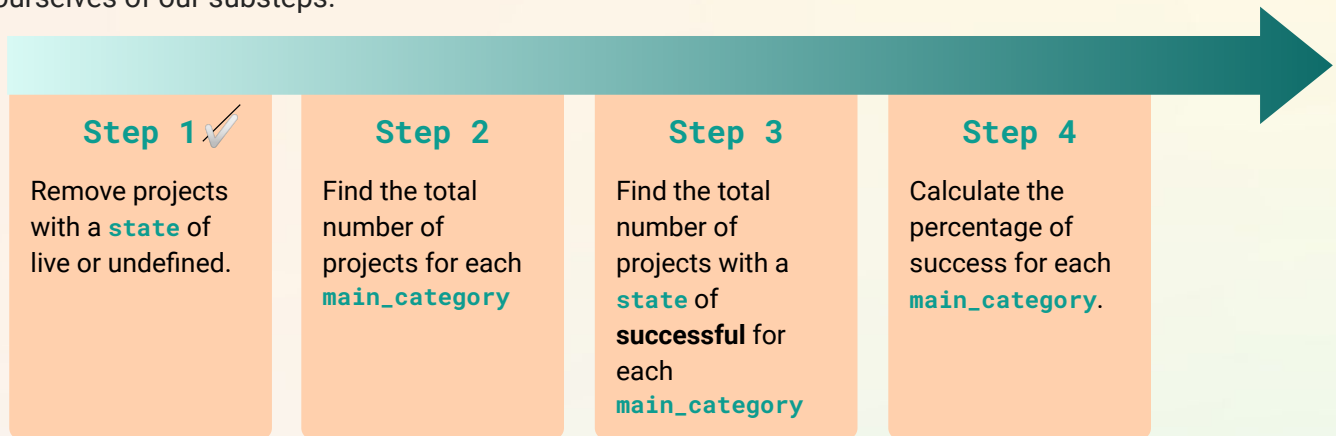
RESULTS		DEBUGGING TIPS	
failed	197719	◆	Name 'ds' is not defined: Each code block in your notebook runs separately, sometimes your notebook might lose its place in the program and require you to run all of your code blocks. To do this, click the double arrow button  at the top of your notebook to run all code blocks.
successful	133956	◆	Check that you use quotation marks around the names of your features.
canceled	38779	◆	Check that your feature names are spelled correctly. Remember that Python is also case sensitive.
suspended	1846	◆	Check that you do not have extra square brackets <code>[]</code> .
		◆	Check that each conditional is surrounded by parentheses <code>()</code> .

Notice that there are no longer projects that are listed as "undefined" or "live" in our filtered dataset. You should also notice that the number of projects with a state of "failed", "successful", "canceled", and "suspended" remained the same. If you received an error, check for the following things

Whew, we have accomplished a lot already! Now that we have cleaned up our data, it is time to put on our analytical hat and start calculating the success rate!

Step 5: Calculating Percentage of Success (15-20 mins)

Now that our data has been cleaned, we can start calculating the percentage of success. Let's remind ourselves of our substeps.



Believe it or not, you already know how to code the remaining steps! We will be using a combination of the **value_counts()** method to get the number of projects in each category and **conditionals** to filter for only successful projects. Don't worry, we will walk you through all of the steps!

Remember, to calculate the percentage of success we need to compare the number of successful projects to the total number of projects for each category.

$$\% \text{ of success for category}_{\text{film+video}} = \frac{\# \text{ of successful projects in category}_{\text{film+video}}}{\# \text{ of total projects in category}_{\text{film+video}}} \times 100\%$$

In the example above we show how to calculate the % of success for projects in Film & Video. We will want to calculate this percentage for **each category** in our dataset. While the number of projects in each category may differ, the equation should remain the same.

First, we will find the number of successful projects for each category. Then, find the total number of projects for each category. Finally, use those two values to calculate the percentage of success.

Step 5: Calculating Percentage of Success (cont.)

Total Number of Projects Per Category (3-5 mins)

- **Add a new code block at the bottom of your notebook.** Either hover your mouse below your last code block and press the **+Code** button or press the **+** button at the top-left of your notebook.
- **Store the total number of projects per `main_category` in a variable named `totProjCount`.** We will use the `value_counts()` method on the `"main_category"` feature on our filtered dataset, `compProj`. The `value_counts()` method returns a Series object, which is like a List of the values. This will make it easier for us to find the percentage of success for each category with just one line of code. We named our variable `totProjCount`, short for total number of projects.
- **Print out `totProjCount` and run your code block.** This is an optional step, but it is good to check that we are able to get the correct number of total projects before moving on. Use the `print()` method to print out `totProjCount` and then click the blue play button to the left of the code block to run your code.

We previously use the `value_counts()` method to find the number of projects for each type of `state`. Here we want to break down our dataset into categories, so we add the `"main_category"` feature instead.

```
print(totProjCount)
```

RESULTS

Film & Video	62399
Music	49403
Publishing	39113
Games	34943
Technology	32189
Design	29763
Art	27959
Food	24418
Fashion	22563
Theater	10871
Comics	10743
Photography	10730
Crafts	8733
Journalism	4724
Dance	3749

DEBUGGING TIPS

- ◆ You may need to rerun all of your code blocks. Click the double arrow button at the top of your notebook.
- ◆ Check that you are using our filtered dataset, `compProj`.
- ◆ Check that you use quotation marks around the names of your features.
- ◆ Check that your names (features, table, and methods) are spelled correctly. Remember that Python is also case sensitive.
- ◆ Check that you do not have extra square brackets `[]`.
- ◆ Check that you include parentheses after using the `value_counts` method.

Step 5: Calculating Percentage of Success (cont.)

Number of Successful Projects Per Category (3-5 mins)

- **Add a new code block at the bottom of your notebook.** Either hover your mouse below your last code block and press the **+Code** button or press the **+** button at the top-left of your notebook.
- **Store a new reference to the dataset with only successful projects in a variable named `susProj`.** Let's *filter* our dataset on the `compProj` dataset to contain only projects that have a state of "successful".

```
susProj = compProj[compProj["state"] == "successful"]
```

- **Store the total number of successful projects per `main_category` in a variable named `susProjCount`.** We want to make sure that we use the `value_counts()` method on the "`main_category`" feature on our filtered dataset with only successful projects, `susProj`.

```
susProjCount = susProj["main_category"].value_counts()
```


- **Print out `susProjCount` and Run your code block.** This is an optional step, but it is good to check that we are able to get the correct number of total projects before moving on. Use the `print()` method to print out `susProjCount` and then click the blue play button to the left of the code block to run your code.

```
print(susProjCount)
```

RESULTS

Music	24197
Film & Video	23623
Games	12518
Publishing	12300
Art	11510
Design	10550
Theater	6534
Technology	6434
Food	6085
Comics	5842
Fashion	5593
Photography	3305
Dance	2338
Crafts	2115
Journalism	1012

DEBUGGING TIPS

- ◆ You may need to rerun all of your code blocks. Click the double arrow button  at the top of your notebook.
- ◆ Check that you are using our filtered dataset, `susProj`.
- ◆ Check that you use quotation marks around the names of your features.
- ◆ Check that your names (features, table, and methods) are spelled correctly. Remember that Python is also case sensitive.
- ◆ Check that you do not have extra square brackets `[]`.
- ◆ Check that you include parentheses after using the `value_counts` method.

Step 5: Calculating Percentage of Success (cont.)

Calculating Percentage of Success (3-5 mins)

- Add a new code block at the bottom of your notebook.
- Using the `susProjCount` and `totProjCount` variables, calculate the percentage of success. One of the best features of `pandas` is that you can perform the same calculation on multiple features with just one line of code! We can calculate the percentage of success for each category simply by just dividing the `susProjCount` by `totProjCount`. Don't forget to multiply by 100 to get our percentage value.


`susProjCount/totProjCount * 100`

- Run your code block. Click the blue play button to the left of the code block to run your code.

RESULTS

Art	41.167424
Comics	54.379596
Crafts	24.218482
Dance	62.363297
Design	35.446696
Fashion	24.788370
Film & Video	37.857978
Food	24.920141
Games	35.824056
Journalism	21.422523
Music	48.978807
Photography	30.801491
Publishing	31.447345
Technology	19.988195
Theater	60.104866

DEBUGGING TIPS

- ◆ You may need to rerun all of your code blocks. Click the double arrow button  at the top of your notebook.
- ◆ Check that your names are spelled correctly. Remember that Python is also case sensitive.
- ◆ Don't forget to multiple to **100** to get your answer in percentage form.

Step 5: Calculating Percentage of Success (cont.)

Sorting Your Results (3-5 mins)

You may have noticed that your final results are sorted alphabetically by the category name. You can sort your results by using the `sort_values()` method. Let's make updates to our previous code block to output the sorted values instead.

- Store the percentage of success in a variable named `percentSuccess`.

```
percentSuccess = susProjCount/totProjCount * 100
```

- Use the `sort_values()` method to sort the percentages.

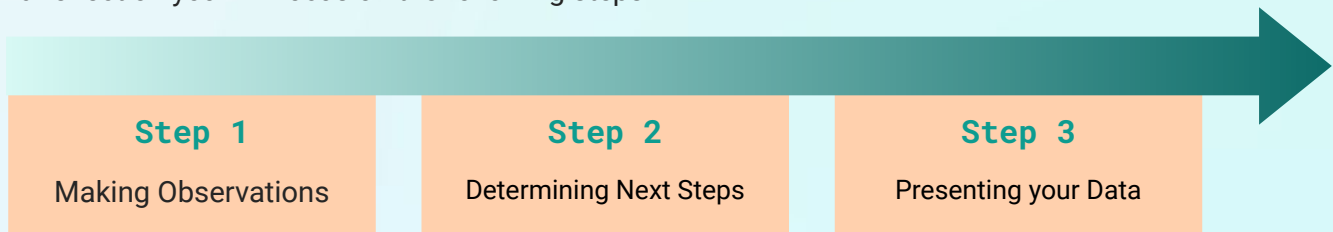
```
percentSuccess.sort_values()
```

- Run your code block. Click the blue play button to the left of the code block to run your code.

RESULTS	DEBUGGING TIPS
Technology 19.988195 Journalism 21.422523 Crafts 24.218482 Fashion 24.788370 Food 24.920141 Photography 30.801491 Publishing 31.447345 Design 35.446696 Games 35.824056 Film & Video 37.857978 Art 41.167424 Music 48.978807 Comics 54.379596 Theater 60.104866 Dance 62.363297	<ul style="list-style-type: none">◆ You may need to rerun all of your code blocks. Click the double arrow button <code>⏮</code> at the top of your notebook.◆ Check that your names are spelled correctly. Remember that Python is also case sensitive.

Step 5: Reflecting on Final Results (15-20 mins)

A detailed reflection on your analysis is extremely important when making inferences about future trends. It is just as important as every other step in your analysis process! From a reflection you will focus on the following steps:



In this section we will walk you through a snapshot of the reflection process that data scientists conduct after performing their analysis on big data. We will only be walking you through steps 1 and 2 of the process in this activity. Stay tuned for next week when we walk through how to use different visualization techniques in Python to present your data!

Step 6: Calculating Percentage of Success (cont.)

Making Observations (6-10 mins)

Take **3-5 minutes** to look at the final results and document your observations. We are using time as a constraint here so that you can focus on the data that stands out to you the most - you can always come back to these later if you like! During this time you should write down any observations you notice about your data. Focus only on the facts/data and try not to make any observations.

OBSERVATION <i>What data stood out to you? Stick to the facts in the data and do not make conclusions at this time.</i>	DATA SOURCE <i>Where did you find this information? Include the name of the dataset and/or the line of code.</i>
<i>Ex. The Film & Video category has the most number of projects, totaling to 62,399.</i>	<i>Ex: In totProjCount. Count total from filtered dataset removing the "undefined" and "live" projects.</i>

Take a moment to review the observations you wrote down in your table. In this section, we will answer our main question: **Which main category of projects has the highest success rate?**

Now set the timer for **3 minutes** to answer our main question and identify anything else you found interesting and why. You might want to consider what you think your data says about Kickstarter projects in general and their categories. Think about in the point of view of a backer, if you were to pledge to a project which category of project would you consider to have the potential of success?

Determining Next Steps (3 mins)

Now that you have made some observations about your data, what is next for your work? In this section take some time to think about some of the constraints you had in your analysis and some other things you would want to explore further with this dataset. What additional information would be helpful in your research? Take **3 minutes** to reflect on the following questions:

QUESTION 1

What are some challenges you faced when analyzing your data?

QUESTION 2

What additional questions would you want to know about the data?

QUESTION 3

What additional information would you need to take your analysis further?

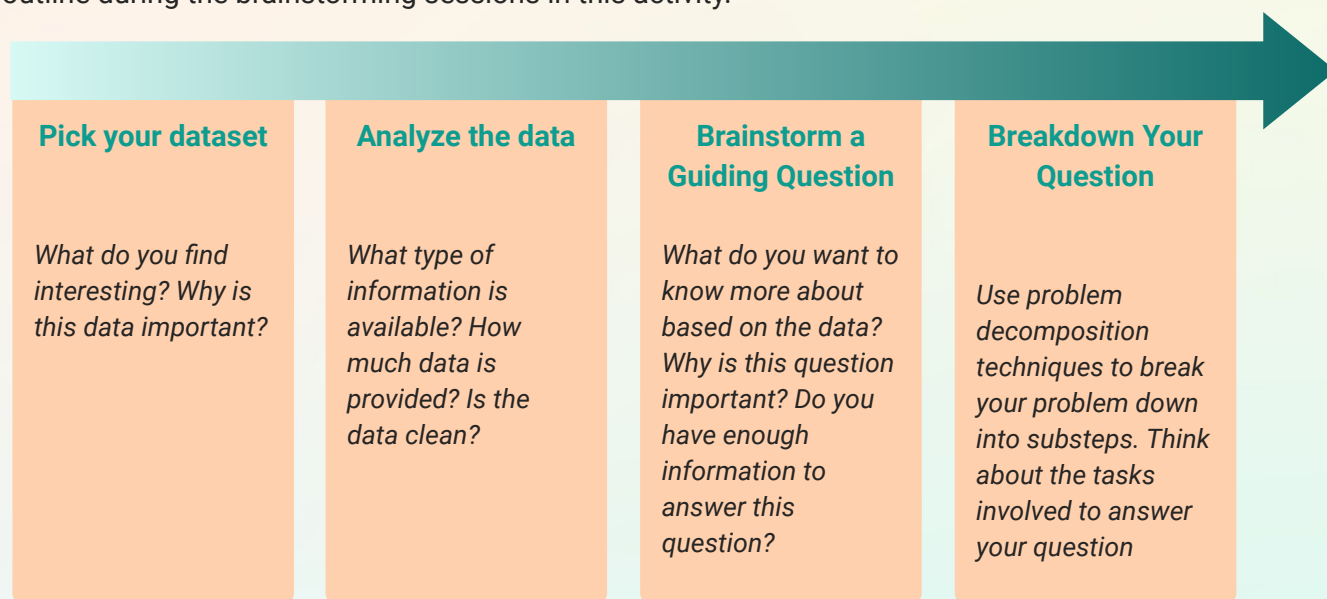
Often when working with data you will want to present your findings with different types of visualizations, including charts and graphs. In this activity we explored how to manipulate data, but stay tuned for our next activity when we explore data visualization techniques in Python. For now, take a break and pat yourselves on the back for being data scientists for a day!

Step 7: Extensions (5-40 mins)

Extension 1: Exploring other Kaggle Datasets (30-40 mins)

There are many datasets available on Kaggle. Take a moment to search through some datasets [here](#). You might want to consider sorting by **Usability**, recall that this score helps you determine how “clean” a dataset is.

When performing your own analysis on your chosen dataset, you want to follow a similar process as we outline during the brainstorming sessions in this activity.



Once you have picked your dataset and brainstormed ways you can analyze the data, it's time to start your research! Some research may take hours, days, or even months. Don't be discouraged. In this activity we only scratched the surface of some of the things that **pandas** can help with analyzing data, but **pandas** can do so much more! Take a look at these resources to learn more about this powerful library!

- [DataCamp's Pandas Tutorial: DataFrames in Python](#)
- [LearnDataSci's Python Pandas Tutorial](#)
- [Tutorials Point's Python Pandas Tutorial](#)
- [DataCamp's Pandas Cheat Sheet](#)

Extension 2: Finding Percentage of Failure (10-15 mins)

Now that we found the percentage of success, what if we wanted the percentage of failure? We can do this in a couple different ways.

- If we only view failure as the *opposite* of success and that there are only two final states, we can just subtract the percentage of success from 100%.

```
percentFail = 1 - percentSuccess
```

Remember that we originally filtered out the projects with a state of "undefined" and "live"

Recall that there are several different states in our dataset: successful, failed, canceled, undefined, live, and suspended. By just subtracting the percentage of success from 100% this assumes that we treat the state "canceled" and "suspended" the same as "failed".

- Let's say we want the percentage of failed to only consider the projects with a state of "failed". We can do this simply by using the `value_counts()` method on our dataset.

```
failProj = compProj[compProj["state"]=="failed"]
failProjCount = failProj["main_category"].value_counts()
percentFail = failProjCount/totProjCount * 100
```

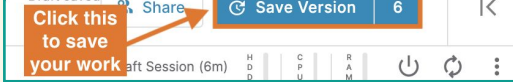
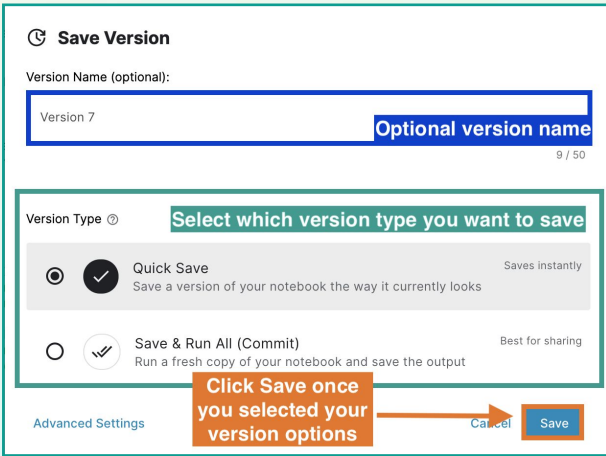
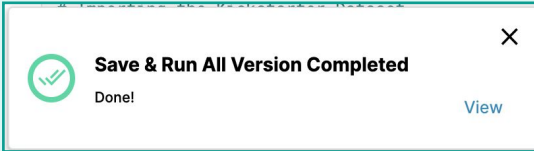
Extension 3: Replacing Values in Your Dataset (5-10 mins)

Sometimes you might want to replace values within your dataset. For example, you might want to combine some categories or maybe you want to make sure that certain values are read the same like "Film & Video" is the same as "Film and Video". We can do this easily with `pandas` by using the `replace()` method. Take a look at this [resource](#) to learn more about the different situations you may use the replace method.

Step 8: Share Your Girls Who Code at Home Project! (5-10 mins)

We would love to see your work and we know others would as well. Share your final project with us. Don't forget to tag [@girlswhocode](#) [#codefromhome](#) and we might even feature you on our account!

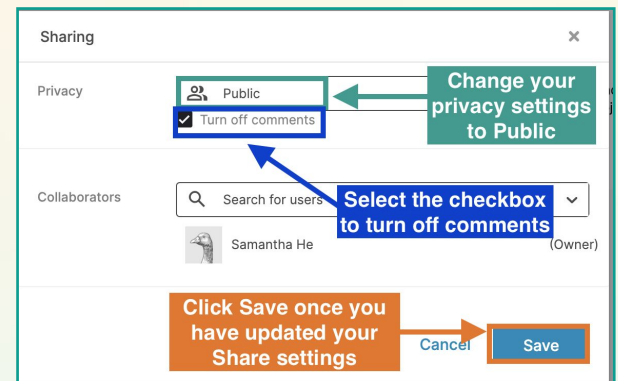
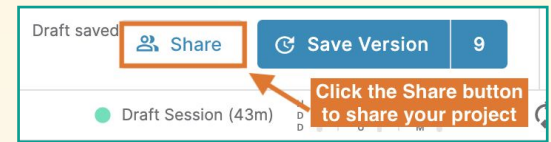
Saving Your Work (2-5 mins)

- **Click the Save Version Button.** On the top right corner of your notebook, you may have noticed the **Save Version** button. This should open a new window.
- **Add a Version Name.** This optional field is a great way for you to document what you did in this new version that may have been different than previous versions. Kaggle will automatically number your versions so it is easy to view old versions.
- **Select the Version Type.** We recommend selecting the **Save & Run All** option.
 - ◆ **Quick Save:** This is a great way to save very quickly your work. This version will save everything exactly as it is displayed in your notebook. This version of save may be problematic if you made edits to your notebook but did not rerun all code blocks.
 - ◆ **Save & Run All:** This saves a fresh copy of your notebook by running all code blocks and then saving this version. This is always the best way to save your notebooks if you have the time.
- **Click the Save button and Wait.** Once you have confirmed your save version options, click the **Save** button. You should see a pop-up window letting you know the status of your save. You might need to wait a minute for your version to be saved completely.

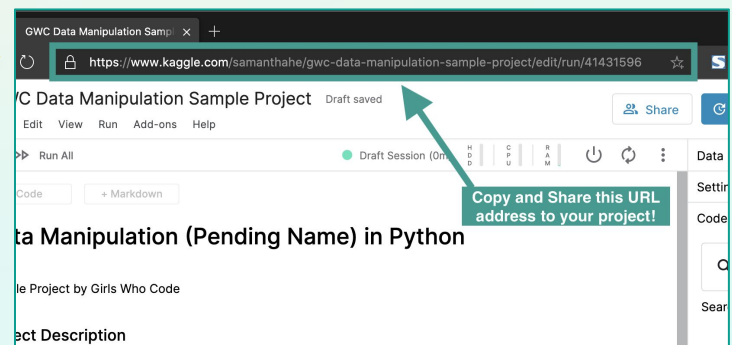
Step 8: Share Your Girls Who Code at Home Project (cont.)

Sharing Your Work (3-5 mins)

- **Click the Share button.** On the top right of your notebook, next to the Save Version button is the Share button.
- **Change the Privacy to Public.** Select the drop down column to the right of the Privacy field and select **Public**. This will pop up a warning message make sure you are aware that other users will be able to view your project. Select **Ok, make public**.
- **Check the turn off comments box.** Click the checkbox to the left of the Turn off comments option.
- **Click Save.** Once you have confirmed that your settings are correct, click the **Save** button.
- **Share the URL address to your notebook.** Finally, just copy and paste the URL address to your project with us!



Project Link



Stay tuned for more Girls Who Code at Home projects!

