**Exploring Data with SQL**

Like most organizations, Codecademy uses SQL (Structured Query Language) to access its database.

A database is a set of data stored in a computer. This data is usually structured into *tables*. Tables can grow large and have a multitude of columns and records.

Spreadsheets, like Microsoft Excel and Google Sheets, allow you to view and manipulate data directly: with selecting, filtering, sorting, etc. By applying a number of these operations you can obtain the subset of data you are seeking.

SQL (pronounced “S-Q-L” or “sequel”) allows you to write queries which define the subset of data you are seeking. Unlike Excel and Sheets, your computer and SQL will handle how to get the data; you can focus on what data you would like. You can save these queries, refine them, share them, and run them on different databases.

It is a great way to access data and a great entry point to programming because its syntax (the specific vocabulary that gives instructions to the computer) is very human-readable. Without knowing any SQL, you might still be able to guess what each command will do.

On her first day at Codecademy, Catherine wants to become familiar with the company’s data, so she connects to the database and uses SQL to explore the database.

1. One of the tables in Codecademy’s database is called browse. It contains information on each time someone visited the Codecademy’s website. Paste the following code into the code editor (middle panel) and click Run.
   * SELECT \*  
     FROM browse  
     LIMIT 10;

This code will select all (\*) columns from browse table for the first 10 records.

Once you correctly enter the code and click Run, this instruction will turn green, letting you know that you completed this checkpoint.

**SELECT \***

**FROM browse**

**LIMIT 10;**

**Exploring Data with SQL - Continued**

Next, Catherine wants to take a look at the churn rate.

*Churn rate* is the percent of subscribers to a monthly service who have canceled. For example, in January, let’s say Codecademy had 1,000 learners. In February, 200 learners sign up, and 250 cancel.

The churn rate for February would be:

������������������ �����������=2501000+200=20.8%*total* *subscriberscancellations*​=1000+200250​=20.8%

Catherine wants to analyze the churn rates for Codecademy for the past few months so she writes another SQL query.

1. Click Run, to see Catherine’s analysis for the churn rate in March 2017. What recommendations would you make to Codecademy based on Catherine’s analysis? (This query might take some time to load because the pro\_users table has 118,135 rows!)

**SELECT COUNT(DISTINCT user\_id) AS 'enrollments',**

**COUNT(CASE**

**WHEN strftime("%m", cancel\_date) = '03'**

**THEN user\_id**

**END) AS 'march\_cancellations',**

**ROUND(100.0 \* COUNT(CASE**

**WHEN strftime("%m", cancel\_date) = '03'**

**THEN user\_id**

**END) / COUNT(DISTINCT user\_id)) AS 'churn\_rate'**

**FROM pro\_users**

**WHERE signup\_date < '2017-04-01'**

**AND (**

**(cancel\_date IS NULL) OR**

**(cancel\_date > '2017-03-01')**

**);**

**Programming with Python**

After interacting with the database, it is time to analyze the data further and eventually visualize the data. And SQL cannot get us there.

Python is a general-purpose programming language. It can do almost all of what other languages can do with comparable, or faster, speed. It is often chosen by Data Analysts and Data Scientists for prototyping, visualization, and execution of data analyses on datasets.

There’s an important question here. Plenty of other programming languages, like R, can be useful in the field of data science. Why are so many people choosing Python?

One major factor is Python’s versatility. [There are over 125,000 third-party Python libraries.](https://pypi.org/) These libraries make Python more useful for specific purposes, from the traditional (e.g. web development, text processing) to the cutting edge (e.g. AI and machine learning). For example, a biologist might use the [Biopython library](https://biopython.org/" \t "_blank) to aid their work in genetic sequencing.

Additionally, Python has become a go-to language for data analysis. With data-focused libraries like pandas, NumPy, and Matplotlib, anyone familiar with Python’s syntax and rules can use it as a powerful tool to process, manipulate, and visualize data.

1. Catherine just downloaded Python 3 onto her office laptop. Let’s test out a simple piece of code. Run the script.py program.

**libraries = ["NumPy", "SciPy", "Pandas", "Matplotlib", "Seaborn"]**

**completion = [100, 100, 96, 0, 0]**

**libraries.append("scikit-learn")**

**completion.append(0)**

**gradebook = list(zip(libraries, completion))**

**print("Lesson Completion Rates:")**

**print(gradebook)**

**print("\n")**

**# What's next?**

**# gradebook.append(("BeautifulSoup", 0))**

**# gradebook.append(("Tensorflow", 0))**

**Visualizing Data with Matplotlib and Seaborn**

Catherine wants to *visualize* her analysis and share it with her boss. For this, she will use Matplotlib, another Python module.

Matplotlib lets Catherine create line charts, bar charts, pie charts, and more. It gives her precise control over colors and labels so that she can create the perfect chart to communicate her findings.

Catherine has written some code using Matplotlib that visualizes hours of usage on Codecademy!

1. Catherine has written some code in script.py, but it won’t display her new graph until you add the following code to the very end of the file:

plt.show()

This tells Matplotlib to create and display the plot!

**import codecademylib3\_seaborn**

**from matplotlib import pyplot as plt**

**import numpy as np**

**import pandas as pd**

**hour = range(24)**

**viewers\_hour = [30, 17, 34, 29, 19, 14, 3, 2, 4, 9, 5, 48, 62, 58, 40, 51, 69, 55, 76, 81, 102, 120, 71, 63]**

**plt.title("Codecademy Learners Time Series")**

**plt.xlabel("Hour")**

**plt.ylabel("Viewers")**

**plt.plot(hour, viewers\_hour)**

**plt.legend(['2015-01-01'])**

**ax = plt.subplot()**

**ax.set\_facecolor('seashell')**

**ax.set\_xticks(hour)**

**ax.set\_yticks([0, 20, 40, 60, 80, 100, 120])**

**y\_upper = [i + (i\*0.15) for i in viewers\_hour]**

**y\_lower = [i - (i\*0.15) for i in viewers\_hour]**

**plt.fill\_between(hour, y\_lower, y\_upper, alpha=0.2)**

**# Add the code here:**

**plt.show()**

**Probability**

Catherine is going to visit the Inference and Machine Learning Teams this week. She knows that both of these specialists work with Probability, and wants to brush up on her skills before she goes, so she’s going to explore a famous problem – about birthdays.

Calculating the probability of an event is sometimes dependent on external factors. For instance, in the birthday problem “What is the probability that two people in a room have the same birthday?” the probability is dependent on the number of people in the room.

Other times, the probability of something is constant. For instance, the probability of flipping a coin and it landing heads will always be 50%.

1. In data science, probability is often used to simulate scenarios.

The code on the right simulates the birthday problem. Right now the code simulates a room with only 2 people that get random birthdays, and the probability that those 2 people have the same birthday is really low.

Change the number 2 to a higher number of your choosing where it says #Change This Number and run the code.

Is there a match in the simulation? What’s the probability that there would be a match?

Keep changing the number to test out different simulations.

Note that if you make your number too big, the program will throw an error due to the way we have implemented some of the math. This is a great example of needing to be mindful of possible inputs to your program!

**# We have hidden code in another file. If you're curious, open the folder to the left and inspect the simulate.py file**

**from simulate import simulate**

**num\_people\_in\_room = 100 #Change This Number (keep it smaller than 100 to save processing power)**

**simulate(num\_people\_in\_room)**

**A Day with the Machine Learning Team**

Next, Catherine is headed over to the Machine Learning and Algorithms Team. They use data to make predictions and create new products using data (like recommendation systems). Today they are trying to find trends among millions of learners according to their behavior on the site. They will use a cluster analysis.

Catherine is playing around with some sample data about penguins to learn about cluster analyses.

The dataset she’s looking at is a collection of flipper and bill measurements for three different species of penguins [collected by Dr. Kristen Gorman and the Palmer Station in Antarctica](https://allisonhorst.github.io/palmerpenguins/).

*When you opened this exercise, some code running behind the scenes loaded the data and created a visualization of the flipper and bill measurements for three penguin species.*

Take a look at the visualization in the learning environment. You might notice that there isn’t much overlap between species on the plot. For example, Chinstrap penguins seem to usually have longer bills than Adelie penguins, and so Chinstrap penguins appear above Adelie penguins in the visualization. Regions like these that mostly feature one species over another are called **clusters**.

With learner data, you might find clusters centered around Data Science or Web Development.

We can use this to make predictions. For example, if we find a new penguin that has 180mm long flippers and a 35mm long bill, we might conclude (based on these clusters) that our penguin is more likely to be an Adelie penguin. In our code running behind the scenes, we’ve built a computer model to do this kind of prediction automatically.

1. Change the flipper and bill measurements and run the code. An algorithm we’ve loaded for you will attempt to predict which species of penguin matches the measurements you chose. How well do you think it does? What happens if you enter values that are between two regions, or values outside of the existing regions altogether?

In future Codecademy courses, you will learn to do all of this yourself! If you are curious, you can explore the code by clicking on the file icon in the center pane.

**# Change these numbers and run to predict species!**

**mystery\_penguin\_flipper = 200**

**mystery\_penguin\_bill = 41**

**# The model will use those values and the dataset to predict a species**

**from code import predict**

**predict(mystery\_penguin\_flipper,mystery\_penguin\_bill)**