Part I

Chapter 1

An Introduction to Streamlit

1.1

st.pyplot() is a function that lets us use all the power of the popular matplotlib library and push our matplotlib graph to Streamlit. Once we create a figure in matplotlib, we can explicitly tell Streamlit to write that to our app with the st.pyplot() function.

1.2 Finishing touches – adding text to Streamlit

Other than st.write(), we also can utilize other built-in functions that format our text for us, such as st.title(), st.header(), st.markdown(), and st.subheader(). Using these five functions helps to format text in our Streamlit apps easily and keeps sizing consistent for bigger apps.

More specifically, st.title() will place a large block of text in our app, st.header() uses a slightly smaller font than st.title(), and st.subheader() uses an even smaller one. Other than those three, st.markdown() will allow anyone already familiar with Markdown to use the popular markup language in our Streamlit apps.

Chapter 2

Uploading, Downloading, and Manipulating Data

2.1 An introduction to caching

A good analogy for an app's cache is a human's short-term memory, where we keep bits of information close at hand that we think might be useful. When something is in our short-term memory, we don't have to think very hard to get access to that piece of information. In the same way, when we cache a piece of information in Streamlit, we are making a bet that we'll use that information often.

The way Streamlit caching works more specifically is by storing the results of a function in our app, and if that function is called with the same parameters by another user (or by us if we rerun the app), Streamlit does not run the same function but instead loads the result of the function from memory.

There are two Streamlit caching functions, one for data (`st.cache_data`) and one for resources like database connections or machine learning models (`st.cache_resource`).