3/multipage.py

```
import streamlit as st

st.set_page_config(page_title="Multipage Demo")


def intro():
    import streamlit as st

    st.write("# Welcome to Streamlit! ")
    st.sidebar.success("Select a demo above.")

    st.markdown(
        """
        Streamlit is an open-source app framework built specifically for
        Machine Learning and Data Science projects.

        ** Select a demo from the dropdown on the left** to see some examples
        of what Streamlit can do!

        ### Want to learn more?

        - Check out [streamlit.io](https://streamlit.io)
        - Jump into our [documentation](https://docs.streamlit.io)
        - Ask a question in our [community
          forums](https://discuss.streamlit.io)

        ### See more complex demos

        - Explore a [New York City rideshare dataset](https://github.com/streamlit/demo-uber-nyc-pickups)
    """
    )

def mapping_demo():
    import streamlit as st
    import pandas as pd
    import pydeck as pdk

    from urllib.error import URLError

    st.markdown("# Mapping Demo")
    st.write(
        """
        This demo shows how to use
[`st.pydeck_chart`](https://docs.streamlit.io/develop/api-reference/charts/st.pydeck_chart)
to display geospatial data.
"""
    )

    @st.cache_data
    def from_data_file(filename):
        url = (
            "http://raw.githubusercontent.com/streamlit/"
            "example-data/master/hello/v1/%s" % filename
        )
        return pd.read_json(url)

    try:
        ALL_LAYERS = {
            "Bike Rentals": pdk.Layer(
                "HexagonLayer",
                data=from_data_file("bike_rental_stats.json"),
                get_position=["lon", "lat"],
                radius=200,
```

```
            elevation_scale=4,
            elevation_range=[0, 1000],
            extruded=True,
        ),
        "Bart Stop Exits": pdk.Layer(
            "ScatterplotLayer",
            data=from_data_file("bart_stop_stats.json"),
            get_position=["lon", "lat"],
            get_color=[200, 30, 0, 160],
            get_radius="[exits]",
            radius_scale=0.05,
        ),
        "Bart Stop Names": pdk.Layer(
            "TextLayer",
            data=from_data_file("bart_stop_stats.json"),
            get_position=["lon", "lat"],
            get_text="name",
            get_color=[0, 0, 0, 200],
            get_size=15,
            get_alignment_baseline="'bottom'",
        ),
        "Outbound Flow": pdk.Layer(
            "ArcLayer",
            data=from_data_file("bart_path_stats.json"),
            get_source_position=["lon", "lat"],
            get_target_position=["lon2", "lat2"],
            get_source_color=[200, 30, 0, 160],
            get_target_color=[200, 30, 0, 160],
            auto_highlight=True,
            width_scale=0.0001,
            get_width="outbound",
            width_min_pixels=3,
            width_max_pixels=30,
        ),
    }
    st.sidebar.markdown("### Map Layers")
    selected_layers = [
        layer
        for layer_name, layer in ALL_LAYERS.items()
        if st.sidebar.checkbox(layer_name, True)
    ]
    if selected_layers:
        st.pydeck_chart(
            pdk.Deck(
                map_style="mapbox://styles/mapbox/light-v9",
                initial_view_state={
                    "latitude": 37.76,
                    "longitude": -122.4,
                    "zoom": 11,
                    "pitch": 50,
                },
                layers=selected_layers,
            )
        )
    else:
        st.error("Please choose at least one layer above.")
except URLError as e:
    st.error(
        """
        **This demo requires internet access.**

        Connection error: %s
    """
        % e.reason
    )
```

```
def plotting_demo():
    import streamlit as st
    import time
    import numpy as np

    st.markdown("# Plotting Demo")
    st.write(
        """
        This demo illustrates a combination of plotting and animation with
Streamlit. We're generating a bunch of random numbers in a loop for around
5 seconds. Enjoy!
"""
    )

    progress_bar = st.sidebar.progress(0)
    status_text = st.sidebar.empty()
    last_rows = np.random.randn(1, 1)
    chart = st.line_chart(last_rows)

    for i in range(1, 101):
        new_rows = last_rows[-1, :] + np.random.randn(5, 1).cumsum(axis=0)
        status_text.text("%i%% Complete" % i)
        chart.add_rows(new_rows)
        progress_bar.progress(i)
        last_rows = new_rows
        time.sleep(0.05)

    progress_bar.empty()

    # Streamlit widgets automatically run the script from top to bottom. Since
    # this button is not connected to any other logic, it just causes a plain
    # rerun.
    st.button("Re-run")


def data_frame_demo():
    import streamlit as st
    import pandas as pd
    import altair as alt

    from urllib.error import URLError

    st.markdown("# DataFrame Demo")
    st.write(
        """
        This demo shows how to use `st.write` to visualize Pandas DataFrames.

(Data courtesy of the [UN Data Explorer](http://data.un.org/Explorer.aspx).)
"""
    )

    @st.cache_data
    def get_UN_data():
        AWS_BUCKET_URL = "http://streamlit-demo-data.s3-us-west-2.amazonaws.com"
        df = pd.read_csv(AWS_BUCKET_URL + "/agri.csv.gz")
        return df.set_index("Region")

    try:
        df = get_UN_data()
        countries = st.multiselect(
            "Choose countries", list(df.index), ["China", "United States of America"]
        )
        if not countries:
            st.error("Please select at least one country.")
        else:
            data = df.loc[countries]
```

```
            data /= 1000000.0
            st.write("### Gross Agricultural Production ($B)")

            #print pandas dataframe as a nicely formatted table (kable)
            st.write(data.sort_index())

            data = data.T.reset_index()
            data = pd.melt(data, id_vars=["index"]).rename(
                columns={"index": "year", "value": "Gross Agricultural Product ($B)"}
            )
            chart = (
                alt.Chart(data)
                .mark_area(opacity=0.3)
                .encode(
                    x="year:T",
                    y=alt.Y("Gross Agricultural Product ($B):Q", stack=None),
                    color="Region:N",
                )
            )
            st.altair_chart(chart, use_container_width=True)
    except URLError as e:
        st.error(
            """
            **This demo requires internet access.**

            Connection error: %s
        """
            % e.reason
        )

page_names_to_funcs = {
    "-": intro,
    "Plotting Demo": plotting_demo,
    "Mapping Demo": mapping_demo,
    "DataFrame Demo": data_frame_demo
}

demo_name = st.sidebar.selectbox("Choose a demo", page_names_to_funcs.keys())
page_names_to_funcs[demo_name]()
```

4/uber_pickups.py

```
"""An example of showing geographic data."""

import os

import altair as alt
import numpy as np
import pandas as pd
import pydeck as pdk
import streamlit as st

# SETTING PAGE CONFIG TO WIDE MODE AND ADDING A TITLE AND FAVICON
st.set_page_config(layout="wide", page_title="NYC Ridesharing Demo", page_icon=":taxi:")


# LOAD DATA ONCE
@st.cache_data
def load_data():
    path = "uber-raw-data-sep14.csv.gz"
    if not os.path.isfile(path):
        path = f"https://github.com/streamlit/demo-uber-nyc-pickups/raw/main/{path}"
```

```
    data = pd.read_csv(
        path,
        nrows=100000,  # approx. 10% of data
        names=[
            "date/time",
            "lat",
            "lon",
        ],  # specify names directly since they don't change
        skiprows=1,  # don't read header since names specified directly
        usecols=[0, 1, 2],  # doesn't load last column, constant value "B02512"
        parse_dates=[
            "date/time"
        ],  # set as datetime instead of converting after the fact
    )

    return data


# FUNCTION FOR AIRPORT MAPS
def map(data, lat, lon, zoom):
    st.write(
        pdk.Deck(
            map_style="mapbox://styles/mapbox/light-v9",
            initial_view_state={
                "latitude": lat,
                "longitude": lon,
                "zoom": zoom,
                "pitch": 50,
            },
            layers=[
                pdk.Layer(
                    "HexagonLayer",
                    data=data,
                    get_position=["lon", "lat"],
                    radius=100,
                    elevation_scale=4,
                    elevation_range=[0, 1000],
                    pickable=True,
                    extruded=True,
                ),
            ],
        )
    )


# FILTER DATA FOR A SPECIFIC HOUR, CACHE
@st.cache_data
def filterdata(df, hour_selected):
    return df[df["date/time"].dt.hour == hour_selected]


# CALCULATE MIDPOINT FOR GIVEN SET OF DATA
@st.cache_data
def mpoint(lat, lon):
    return (np.average(lat), np.average(lon))


# PLOT FILTERED DATA AS HISTOGRAM
@st.cache_data
def histdata(df, hr):
    filtered = data[
        (df["date/time"].dt.hour >= hr) & (df["date/time"].dt.hour < (hr + 1))
    ]

    hist = np.histogram(filtered["date/time"].dt.minute, bins=60, range=(0, 60))[0]
```

```
    return pd.DataFrame({"minute": range(60), "pickups": hist})


# STREAMLIT APP LAYOUT
data = load_data()

# LAYING OUT THE TOP SECTION OF THE APP
row1_1, row1_2 = st.columns((2, 3))

# SEE IF THERE'S A QUERY PARAM IN THE URL (e.g. ?pickup_hour=2)
# THIS ALLOWS YOU TO PASS A STATEFUL URL TO SOMEONE WITH A SPECIFIC HOUR SELECTED,
# E.G. https://share.streamlit.io/streamlit/demo-uber-nyc-pickups/main?pickup_hour=2
if not st.session_state.get("url_synced", False):
    try:
        pickup_hour = int(st.query_params["pickup_hour"])
        st.session_state["pickup_hour"] = pickup_hour
        st.session_state["url_synced"] = True
    except KeyError:
        pass


# IF THE SLIDER CHANGES, UPDATE THE QUERY PARAM
def update_query_params():
    hour_selected = st.session_state["pickup_hour"]
    st.query_params["pickup_hour"] = hour_selected


with row1_1:
    st.title("NYC Uber Ridesharing Data")
    hour_selected = st.slider(
        "Select hour of pickup", 0, 23, key="pickup_hour", on_change=update_query_params
    )


with row1_2:
    st.write(
        """
    ##
    Examining how Uber pickups vary over time in New York City's and at its major regional airports.
    By sliding the slider on the left you can view different slices of time and explore different transportation
↪   trends.
    """
    )

# LAYING OUT THE MIDDLE SECTION OF THE APP WITH THE MAPS
row2_1, row2_2, row2_3, row2_4 = st.columns((2, 1, 1, 1))

# SETTING THE ZOOM LOCATIONS FOR THE AIRPORTS
la_guardia = [40.7900, -73.8700]
jfk = [40.6650, -73.7821]
newark = [40.7090, -74.1805]
zoom_level = 12
midpoint = mpoint(data["lat"], data["lon"])

with row2_1:
    st.write(
        f"""**All New York City from {hour_selected}:00 and {(hour_selected + 1) % 24}:00**"""
    )
    map(filterdata(data, hour_selected), midpoint[0], midpoint[1], 11)

with row2_2:
    st.write("**La Guardia Airport**")
    map(filterdata(data, hour_selected), la_guardia[0], la_guardia[1], zoom_level)

with row2_3:
```

```
    st.write("**JFK Airport**")
    map(filterdata(data, hour_selected), jfk[0], jfk[1], zoom_level)

with row2_4:
    st.write("**Newark Airport**")
    map(filterdata(data, hour_selected), newark[0], newark[1], zoom_level)

# CALCULATING DATA FOR THE HISTOGRAM
chart_data = histdata(data, hour_selected)

# LAYING OUT THE HISTOGRAM SECTION
st.write(
    f"""**Breakdown of rides per minute between {hour_selected}:00 and {(hour_selected + 1) % 24}:00**"""
)

st.altair_chart(
    alt.Chart(chart_data)
    .mark_area(
        interpolate="step-after",
    )
    .encode(
        x=alt.X("minute:Q", scale=alt.Scale(nice=False)),
        y=alt.Y("pickups:Q"),
        tooltip=["minute", "pickups"],
    )
    .configure_mark(opacity=0.2, color="red"),
    use_container_width=True,
)
```

5/session_state_example.py

```
import streamlit as st
import time

st.set_page_config(page_title="Expensive Calculation")

st.title("Expensive Calculation Demo")

# user inputs
route = st.selectbox("CTA route to cut", ["Red Line", "Blue Line", "Green Line"])
cut_pct = st.slider("What fraction of service to cut?", min_value = 0.0, max_value = 50.0, value = 10.0, step=5.0)

if "counter" not in st.session_state:
    st.session_state.counter = 0

# session_state key
key = f"n_pass_{route}_{cut_pct}"

# run the heavy job only if we haven't already stored it
if key not in st.session_state:
    with st.spinner("Crunching numbers..."):
        time.sleep(3)  # stand-in for a slow computation
        st.session_state[key] = (100 - cut_pct) * 1.2345  # fake result
    st.session_state.counter += 1

st.metric("Expected number of passengers per trip", f"{st.session_state[key]:.2f}")

st.write(f"This page has run calculations {st.session_state.counter} times. Session state keys (programmer tool,
↪  would not normally be shown to users):", list(st.session_state.keys()))
```

6/session_state_cache.py

```python
import streamlit as st
import time


st.set_page_config(page_title="@cache_data")
st.title("Expensive Calculation Demo")

# user inputs
route = st.selectbox("CTA route to cut", ["Red Line", "Blue Line", "Green Line"])
cut_pct = st.slider("What fraction of service to cut?", 0.0, 50.0, 10.0, step=5.0)

if "counter" not in st.session_state:
    st.session_state.counter = 0

@st.cache_data()
def compute_passengers(route, cut_pct):
    with st.spinner("Crunching numbers..."):
        time.sleep(3)  # stand-in for a slow computation
    st.session_state.counter += 1
    return (100 - cut_pct) * 1.2345  # fake result
expected_passengers = compute_passengers(route, cut_pct)

st.metric("Expected number of passengers per trip", expected_passengers)

st.write(f"This page has run calculations {st.session_state.counter} times.")
```