In Snowsight, create a Streamlit app by going to the Projects tab, clicking on Streamlit, and then clicking on “+ Streamlit App” on the top right of the screen. Name the app whatever you’d like. Choose “TASTY\_BYTES” as the app location, and pick “PUBLIC” as the schema. Use “COMPUTE\_WH” as your warehouse.

Note: If you ever find that you’re just seeing the app and not the code, you can click the small “Open Editor” icon on the bottom left of the screen.

Okay, now delete the code that’s in the app, and paste in the following:

# Import Python Packages

import pandas as pd

import streamlit as st

from snowflake.snowpark.context import get\_active\_session

import altair as alt

# Get the Current Credentials

session = get\_active\_session()

# Streamlit App

st.title(":snowflake: Tasty Bytes Streamlit App :snowflake:")

st.write(

"""Tasty Bytes is a fictitious, global food truck network, that is on a mission to serve unique food options with high quality items in a safe, convenient and cost effective way. In order to drive

forward on their mission, Tasty Bytes is beginning to leverage the Snowflake Data Cloud.

"""

)

st.divider()

@st.cache\_data

def get\_city\_sales\_data(city\_names: list, start\_year: int = 2020, end\_year: int = 2023):

sql = f"""

SELECT

date,

primary\_city,

SUM(order\_total) AS sum\_orders

FROM tasty\_bytes.analytics.orders\_v

WHERE primary\_city in ({city\_names})

and year(date) between {start\_year} and {end\_year}

GROUP BY date, primary\_city

ORDER BY date DESC

"""

sales\_data = session.sql(sql).to\_pandas()

return sales\_data, sql

@st.cache\_data

def get\_unique\_cities():

sql = """

SELECT DISTINCT primary\_city

FROM tasty\_bytes.analytics.orders\_v

ORDER BY primary\_city

"""

city\_data = session.sql(sql).to\_pandas()

return city\_data

def get\_city\_sales\_chart(sales\_data: pd.DataFrame):

sales\_data["SUM\_ORDERS"] = pd.to\_numeric(sales\_data["SUM\_ORDERS"])

sales\_data["DATE"] = pd.to\_datetime(sales\_data["DATE"])

# Create an Altair chart object

chart = (

alt.Chart(sales\_data)

.mark\_line(point=False, tooltip=True)

.encode(

alt.X("DATE", title="Date"),

alt.Y("SUM\_ORDERS", title="Total Orders Sum USD"),

color="PRIMARY\_CITY",

)

)

return chart

def format\_sql(sql):

# Remove padded space for visual purposes

return sql.replace("\n ", "\n")

first\_col, second\_col = st.columns(2, gap="large")

with first\_col:

start\_year, end\_year = st.select\_slider(

"Select date range you want to filter the chart on below:",

options=range(2020, 2024),

value=(2020, 2023),

)

with second\_col:

selected\_city = st.multiselect(

label="Select cities below that you want added to the chart below:",

options=get\_unique\_cities()["PRIMARY\_CITY"].tolist(),

default="Cairo",

)

if len(selected\_city) == 0:

city\_selection = ""

else:

city\_selection = selected\_city

city\_selection\_list = ("'" + "','".join(city\_selection) + "'") if city\_selection else ""

sales\_data, sales\_sql = get\_city\_sales\_data(city\_selection\_list, start\_year, end\_year)

sales\_fig = get\_city\_sales\_chart(sales\_data)

chart\_tab, dataframe\_tab, query\_tab = st.tabs(["Chart", "Raw Data", "SQL Query"])

chart\_tab.altair\_chart(sales\_fig, use\_container\_width=True)

dataframe\_tab.dataframe(sales\_data, use\_container\_width=True)

query\_tab.code(format\_sql(sales\_sql), "sql")