Snowpark & Streamlit Basic Hands-on Lab

With Snowpark for Python, teams can now code with Python's familiar syntax and execute with the superior performance, security and near-zero maintenance of the Snowflake processing engine. Snowpark for Python allows data scientists to write our familiar Python code and translate Python back to SQL in Snowflake. With its partnership with Anaconda, we can use all the secure and well-curated Python packages for Snowpark

Please see this link for more information about Snowpark
https://www.snowflake.com/snowpark/

Snowpark Documentation Link -
https://docs.snowflake.com/en/developer-guide/snowpark/index.html

Streamlit offers a faster way to build and share data apps. Streamlit turns data scripts into shareable web apps in minutes. All in pure Python. No front-end experience required. Snowflake acquired Streamlit in March 2022 to

For more information about Streamlit, check out https://streamlit.io/. To see awesome templates, and community apps curated from our forums or Twitter. Try them out, browse their source code, share with the world, and get inspired for your own projects. Check out https://streamlit.io/gallery .

Snowflake set up

```
-- Create file format, stage and table to copy data from s3 into Snowflake
CREATE or REPLACE file format csvformat
  skip_header = 1
  type = 'CSV';

CREATE or REPLACE stage campaign_data_stage
  file_format = csvformat
  url = 's3://sfquickstarts/Summit 2022 Keynote Demo/campaign_spend/';

CREATE or REPLACE TABLE CAMPAIGN_SPEND (
  CAMPAIGN VARCHAR(60),
  CHANNEL VARCHAR(60),
  DATE DATE,
  TOTAL_CLICKS NUMBER(38,0),
  TOTAL_COST NUMBER(38,0),
  ADS_SERVED NUMBER(38,0)
);
```

```
COPY into CAMPAIGN_SPEND
  from @campaign_data_stage;

CREATE or REPLACE stage monthly_revenue_data_stage
  file_format = csvformat
  url = 's3://sfquickstarts/Summit 2022 Keynote Demo/monthly_revenue/';

CREATE or REPLACE TABLE MONTHLY_REVENUE (
  YEAR NUMBER(38,0),
  MONTH NUMBER(38,0),
  REVENUE FLOAT
);

COPY into MONTHLY_REVENUE
  from @monthly_revenue_data_stage;

CREATE or REPLACE TABLE BUDGET_ALLOCATIONS_AND_ROI (
  MONTH varchar(30),
  SEARCHENGINE integer,
  SOCIALMEDIA integer,
  VIDEO integer,
  EMAIL integer,
  ROI float
);

INSERT INTO BUDGET_ALLOCATIONS_AND_ROI (MONTH, SEARCHENGINE, SOCIALMEDIA, VIDEO,
EMAIL, ROI)
VALUES
('January',35,50,35,85,8.22),
('February',75,50,35,85,13.90),
('March',15,50,35,15,7.34),
('April',25,80,40,90,13.23),
('May',95,95,10,95,6.246),
('June',35,50,35,85,8.22);

CREATE OR REPLACE STAGE dash_sprocs;
CREATE OR REPLACE STAGE dash_models;
CREATE OR REPLACE STAGE dash_udfs;
```

Python Environment – Jupyter Text Editor

Note: You can also use Visual Studio Code. For Visual Studio code, make sure you have the Python interpreter add-in installed. It will prompt you to install it when you create the streamlit_lab.py file.

```
-- create streamlit_lab.py file
```

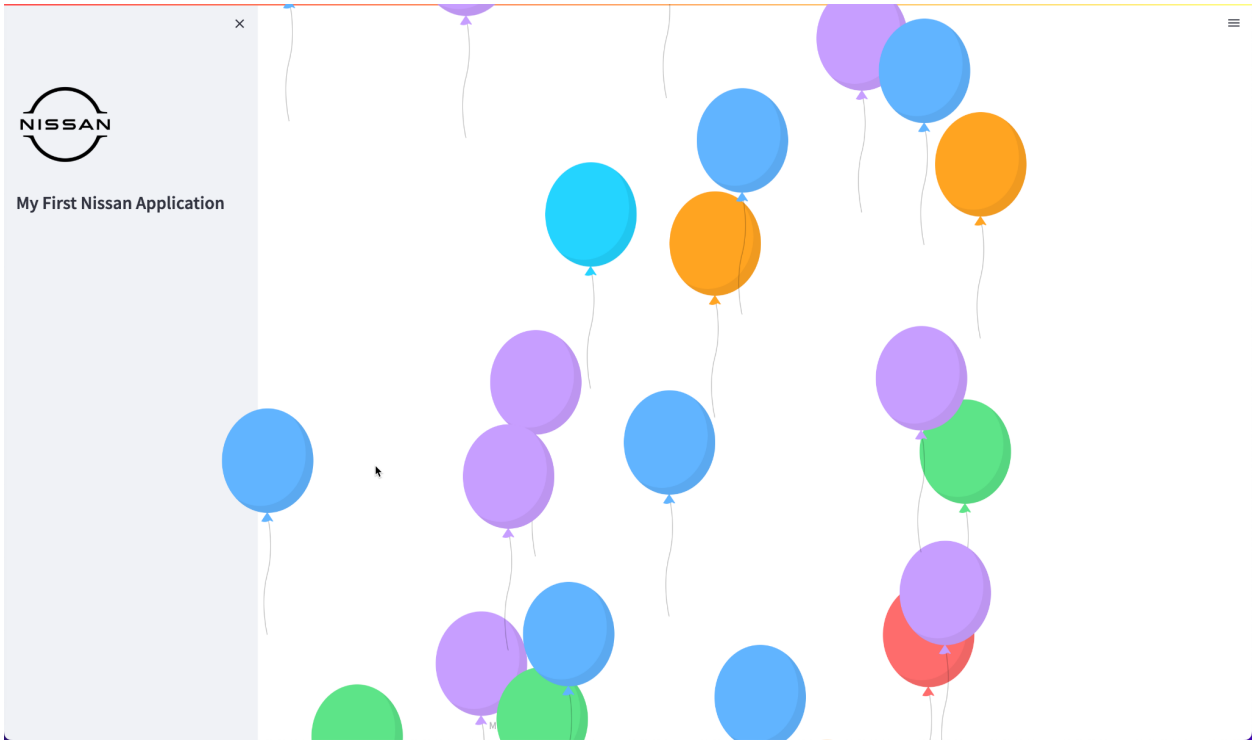
```
# Snowpark for Python API reference:
https://docs.snowflake.com/en/developer-guide/snowpark/reference/python/index.html
# Snowpark for Python Developer Guide:
https://docs.snowflake.com/en/developer-guide/snowpark/python/index.html
# Streamlit docs: https://docs.streamlit.io/

import json
import altair as alt
import pandas as pd
from snowflake.snowpark.session import Session
from snowflake.snowpark.functions import col, sum, call_builtin
from snowflake.snowpark import functions as F

import streamlit as st
from datetime import datetime, date

APP_ICON_URL = "https://www.carlogos.org/car-logos/nissan-logo-2020-black.png"

# Streamlit config
st.set_page_config("Nissan", APP_ICON_URL, "centered")
st.write("<style>[data-testid='stMetricLabel'] {min-height: 0.5rem !important}</style>",
unsafe_allow_html=True)
st.sidebar.image(APP_ICON_URL, width=130)
st.sidebar.title("My First Nissan Application")

#Just for some fun - check out Streamlit's Status elements
https://docs.streamlit.io/library/api-reference/status
st.balloons()
```

Save the streamlit_lab.py file and open a terminal to test your Streamlit app.

Python Environment – Jupyter Terminal

-- change directory to where your Streamlit .py file is located. Hit enter. Then run your streamlit_lab.py file entering "streamlit run streamlit_lab.py" and press run. A new web browser should open with your Streamlit app running.

```
(base) XXXX@C02G5045MD6R ~ % cd Documents/Streamlit
(base) XXXX@C02G5045MD6R Streamlit % streamlit run streamlit_lab.py
```

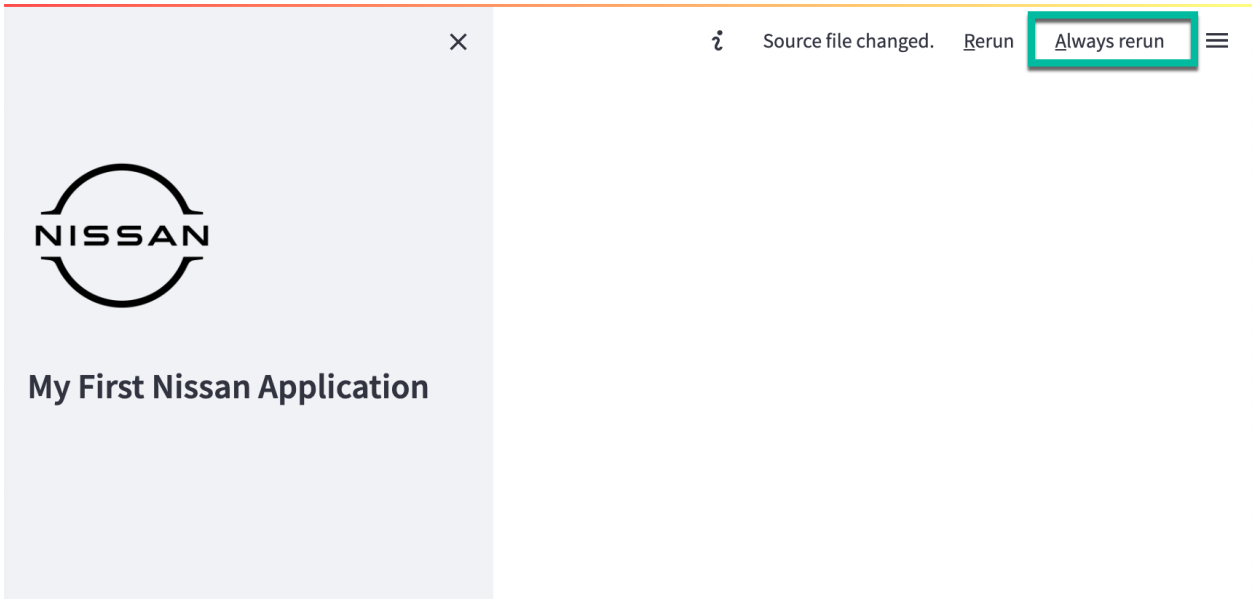


Python Environment – Update Jupyter Text Editor

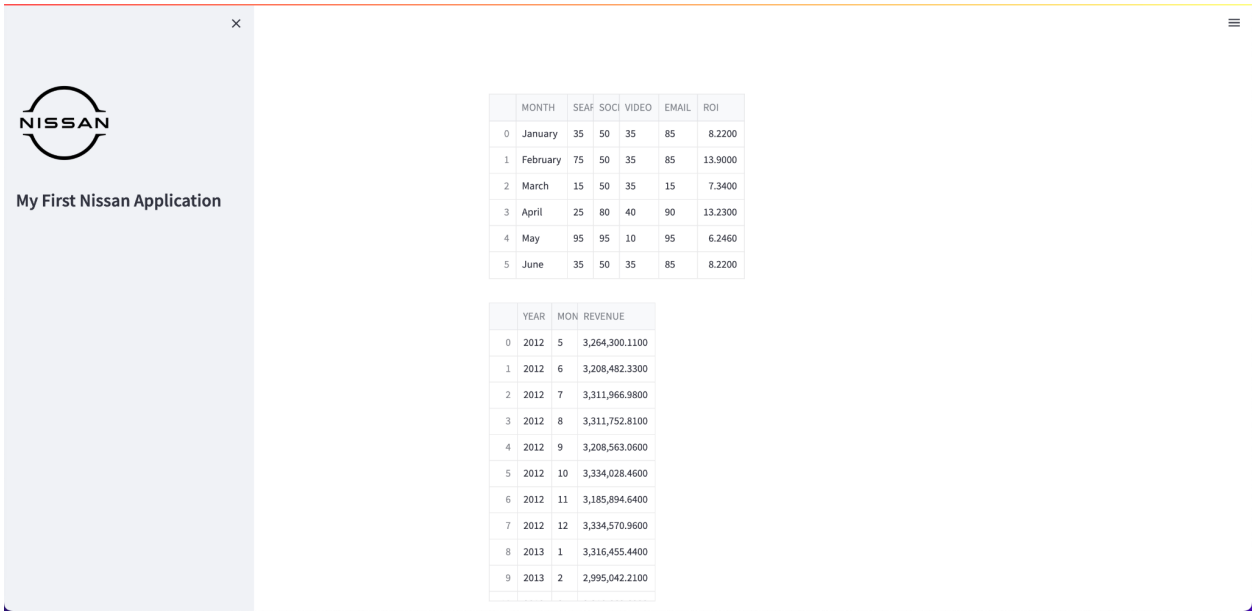
-- update streamlit_lab.py file to ingest data from Snowflake via Snowpark Python API calls

```
# Snowpark for Python API reference:
https://docs.snowflake.com/en/developer-guide/snowpark/reference/python/index.html
# Snowpark for Python Developer Guide:
https://docs.snowflake.com/en/developer-guide/snowpark/python/index.html
# Streamlit docs: https://docs.streamlit.io/

# Part 2 - let's add Snowflake data
vCreds = '/Users/dshaw/Documents/Creds/cred.json'

# Snowflake snowpark connection
with open(vCreds) as f:data = json.load(f)
connection_parameters = { "account": data['account'],
                          "user": data['user'],
                          "password": data['password']}
session = Session.builder.configs(connection_parameters).create()

session.sql("use database streamlit").collect()
df_budget = session.table("aaa.public.BUDGET_ALLOCATIONS_AND_ROI").toPandas()
df_revenue = session.table("aaa.public.MONTHLY_REVENUE").toPandas()

st.dataframe(df_budget)
st.dataframe(df_revenue)
```

Save the updated streamlit_lab.py file and return to the web browser tab running your Streamlit app. In the top right hand corner, select the Always rerun. See how your Streamlit app automatically updates as you adjust and save changes to your streamlit_lab.py file.



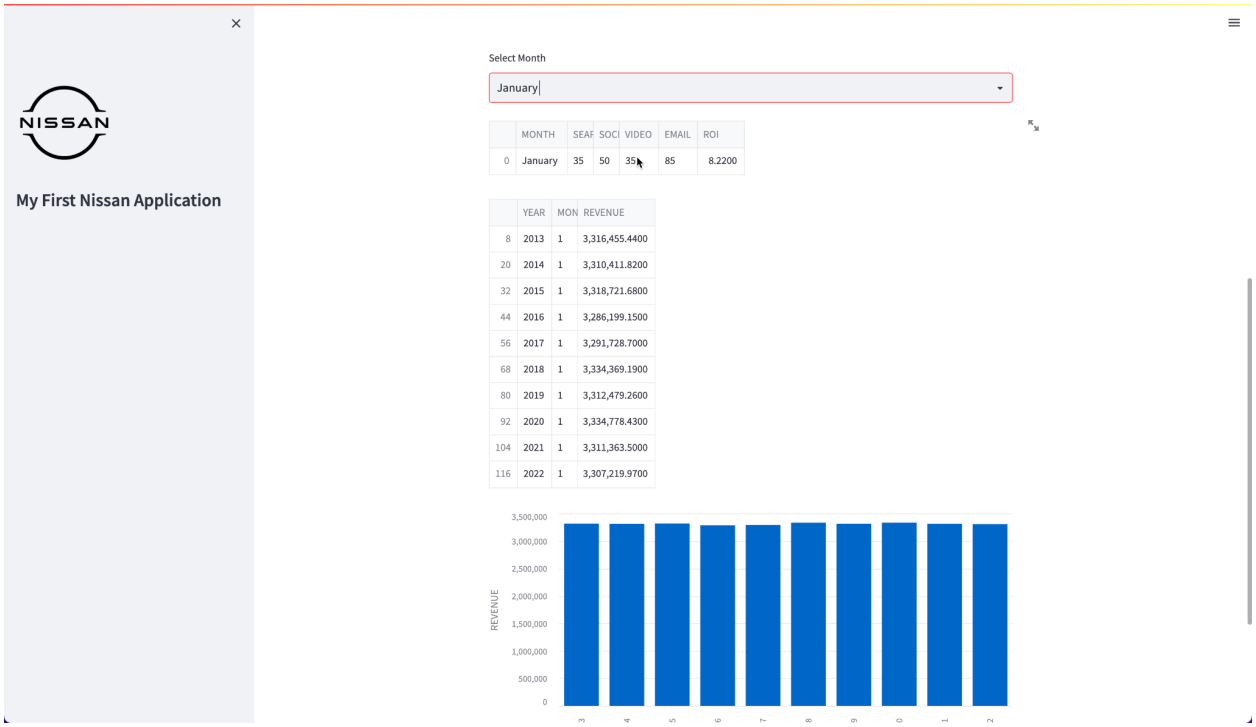
You should now see the following:



Python Environment – Update Jupyter Text Editor

-- update streamlit_lab.py file to ingest data from Snowflake via Snowpark Python API calls

```
# Snowpark for Python API reference:
https://docs.snowflake.com/en/developer-guide/snowpark/reference/python/index.html
# Snowpark for Python Developer Guide:
https://docs.snowflake.com/en/developer-guide/snowpark/python/index.html
# Streamlit docs: https://docs.streamlit.io/

# Part 3 - let's add a filter and chart the Snowflake data within Streamlit using Streamlit Chart Elements
https://docs.streamlit.io/library/api-reference/charts

months = ['']
months.extend(list(df_budget['MONTH'].unique()))
selected_month = st.selectbox("Select Month", options=months, index=0)
if selected_month == '':
    st.dataframe(df_budget)
    st.dataframe(df_revenue)
    st.bar_chart(df_revenue[['YEAR','REVENUE']],x='YEAR', y='REVENUE')
else:
    st.dataframe(df_budget.loc[df_budget['MONTH'] == selected_month])
    st.dataframe(df_revenue.loc[df_revenue['MONTH'] == datetime.strptime(selected_month, '%B').month])
    filtered_df =df_revenue.loc[df_revenue['MONTH'] == datetime.strptime(selected_month, '%B').month]
    st.bar_chart(filtered_df[['YEAR','REVENUE']],x='YEAR', y='REVENUE')
```



```
# Part 4 - let's make this even better!

col1, col2 = st.sidebar.columns(2)
with st.container():
    with col1:
        input_dim = st.selectbox("Dimension",('CAMPAIGN','CHANNEL'))
    with col2:
        input_measure = st.selectbox("Measure",('TOTAL_CLICKS','TOTAL_COST','ADS_SERVED'))

input_date = st.sidebar.date_input("Filter to dates greater than or equal to",date(2019, 1,
1)).strftime("%Y-%m")

df_spend = session.table('aaa.public.CAMPAIGN_SPEND') \
            .withColumn('DATE',call_builtin("to_char", col('DATE'),'YYYY-MM')) \
            .filter(col('DATE') >= input_date) \
            .groupBy('DATE', input_dim) \
            .agg(sum(input_measure).alias(input_measure)) \
            .collect()

st.vega_lite_chart(df_spend, {
        'width': 565,
        'height': 400,
        'mark': {'type': 'area', 'tooltip': True},
        'encoding': {
            'x': {'field': 'DATE'},
            'y': {'field': input_measure.upper(), 'type': 'quantitative'},
            'color': {'field': input_dim.upper()},
        },
    })
```