Descripción: Segundo elemento del modelo LIVE y es contenido desarrollado por el experto en formato electrónico Word. La finalidad es profundizar en los temas definidos en este bloque. Los elementos del modelo INSPIRA que se aplican son: **Nutre y Significa el contenido y Practica.**

Instrucciones : Se realizará como una secuencia de contenidos que se integrará y producirá en diferentes recursos formativos a publicar en la plataforma digital CANVAS. Por lo que se solicita que se desarrollen 18-20 cuartillas de contenido de autoría de experto en Arial 12 (incluyendo actividades de práctica)

Duración: 2h + 30 min de evaluación

Es información tomada del Mapa de Contenidos

Nombre del experto	Adolfo Centeno Tellez
Nombre del programa	Data Science A&I
Nombre del módulo	Aplicación Web de Ciencia de Datos
Objetivos del elemento PROFUNDIZAR) **(trabajo asincrónico)**	**Profundizaras en las características y atributos de la plataforma Streamlit para la construcción de una aplicación Web.**
Contenido	TEMA 1: Sidebar TEMA 2: Interacción con otros componentes (slider, radio, checkbox, date_input) TEMA 3: Gráficas en Streamlit TEMA 4: Visualización de mapas
Take away	Conocimientos par construir una aplicación Web de analítica de datos

1. **Bienvenida (video inicial)**

Las aplicaciones Web son una de las formas más convenientes de mostrar el trabajo que se realiza en la ciencia de datos. La creación de aplicaciones Web puede resultar abrumadora para muchos científicos de datos si no tienen experiencia en desarrollo Web. Sin embargo, con el marco optimizado, las aplicaciones Web ya no son difíciles para los científicos de datos; si se conoce el lenguaje Python, se pueden crear aplicaciones Web interactivas por ejemplo con Streamlit; este marco increíble hace el trabajo difícil para hacer y diseñar elementos web y dejar que el científico de datos simplemente se centre en la parte de los datos y en el análisis.

En este módulo aprenderemos a crear aplicaciones Web utilizando Streamlit para poder generar tableros interactivos que permitan visualizar datos de una forma sencilla pero poderosa.

Se incluye el contexto de los contenidos a revisar en este bloque asincrónico (objetivos) y el resultado esperado, se incluyen preguntas de reflexión y se puede hacer una conexión entre lo revisado en la sesión sincrónica y lo que se verá en este bloque.

2. **Desarrollo del contenido por temas (esto se desarrolla por cada tema incluido en este bloque)**

TEMA 1: Sidebar
TEMA 2: Interacción con otros componentes (slider, radio, checkbox, date_input)
TEMA 3: Gráficas en Streamlit
TEMA 4: Visualización de mapas

Hacer un colab dividido por tema y subtemas en donde se incluya todos los códigos **del nutre**. Este colab se pondrá al final del profundiza, en material adicional para que los participantes puedan consultarlo y ejecutarlo en caso de ellos no consigan replicar las indicaciones del nutre. **(INCLUIR COLAB GUIA NUTRE que va a hacer Grettel)**

GUIÁ PARA EL DESARROLLO DE CONTENIDOS:

TEMA 1: Sidebar

Hay varios marcos de trabajo de Python versátiles para el desarrollo Web, incluidos Flask y Django. Sin embargo, cuando se trata del mundo de la ciencia de datos, los científicos de datos o los ingenieros de aprendizaje automático no son desarrolladores

Web y no están interesados en pasar semanas aprendiendo a usar estos marcos para crear aplicaciones Web. En cambio, quieren una herramienta que sea más fácil de aprender y usar, siempre que pueda mostrar datos y recopilar los parámetros necesarios para el modelado.
Aquí es donde entra en juego Streamlit: un marco Web fácil de usar que nos permite crear aplicaciones web interactivas basadas en datos. Si ya se conoce bien el lenguaje de programación Python, tal vez sea necesario unas pocas horas antes de que se pueda crear una aplicación Web significativa. Ciertamente, aprender a usarlo bien requiere algunas pruebas y errores. En esta sección empezaremos a revisar una de las formas básicas para que vayas construyendo una aplicación Web usando Streamlit.

NUTRE

Si la aplicación Web debe mostrar datos y cifras relacionadas según la entrada del usuario, es una buena idea colocar widgets relevantes en la barra lateral (sidebar). Esta separación deja en claro la intención de la interactividad para los usuarios: la barra lateral (sidebar) es para proporcionar parámetros (es decir, la entrada) mientras que la pantalla principal es para proporcionar comentarios (es decir, la salida) en respuesta a la entrada de los usuarios.

El uso de la barra lateral (sidebar) puede mejorar la limpieza de la aplicación. Sin embargo, debes tener en cuenta que el ancho de la barra lateral es limitado, así que hay que evitar los textos largos y los controles grandes.

Vas a construir el archivo en Google Colab que contendrá una barra lateral. La primera instrucción que tendrá nuestro archivo es la importación de la librería Streamlit. La segunda instrucción es crear un título para nuestra aplicación.

Posteriormente vamos a crear una sección del lado derecho que se denomina barra lateral o **sidebar**. Ya que tenemos creada la barra lateral, lo que tenemos que agregar es un título y un párrafo que explique esta sección.
Ahora bien, vamos a generar unos subtítulos que nos ayuden a dar más contexto sobre nuestra aplicación y para hacerlo vamos a utilizar la instrucción **st.header**.
Finalmente vamos a poner un pequeño párrafo que nos ayude a mostrar una breve descripción de nuestra aplicación Web con **st.write**. A continuación se muestra el código completo instrucción:

```
# Importamos la librería Streamlit
import streamlit as st

# Crear el título para la aplicación web
st.title("Mi Primera App con Streamlit")

# Creamos el sidebar
sidebar = st.sidebar

# Agregamos un titulo y texto al sidebar
sidebar.title("Esta es la barra lateral.")
```

```
sidebar.write("Aquí van los elementos de entrada.")

# Agregamos headers a la seccion principal
st.header("Información sobre el Conjunto de Datos")
st.header("Descripción de los datos ")

# Agregamos texto a la seccion principal
st.write("""
Este es un simple ejemplo de una app para predecir

¡Esta app predice mis datos!

""")
```

La figura 1 muestra la salida de la aplicación con Sidebar.

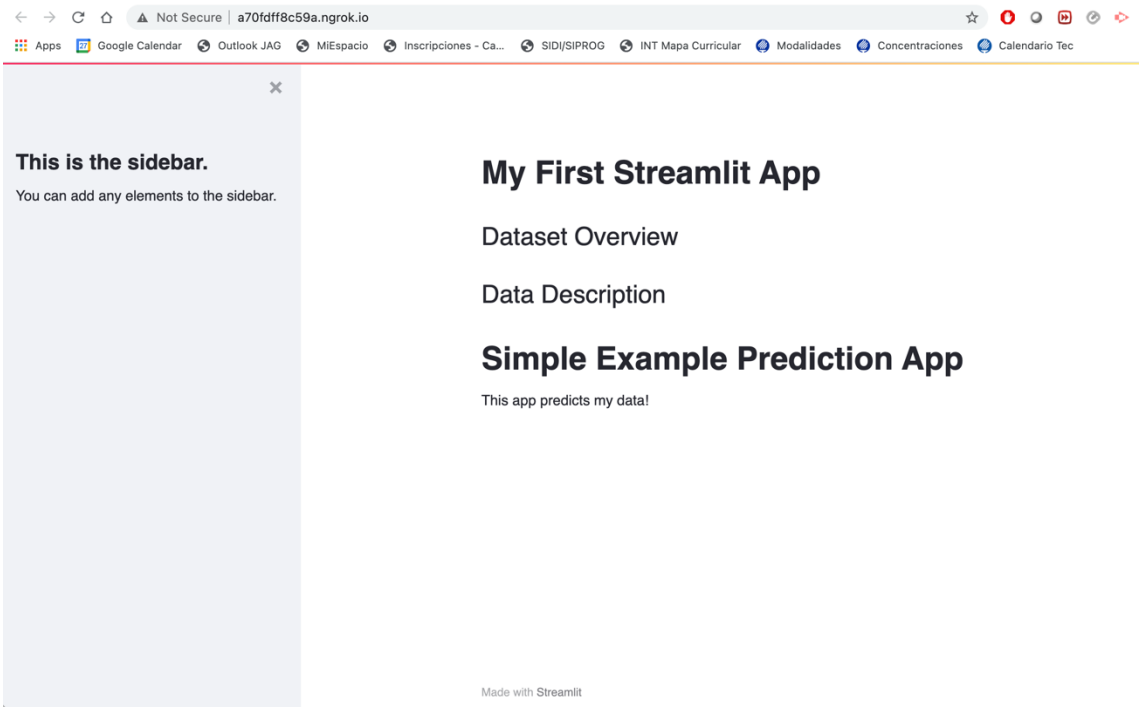


Figura 1. Vista de la aplicación web con Streamlit y Google Colab usando una barra lateral (sidebar).

- **SIGNIFICA**

Ahora que ya aprendiste cómo generar una barra horizontal para una aplicación web utilizando el marco de referencia de Streamlit, cómo podrías construir un sitio web que contenga lo siguiente:

- Una sección de bienvenida que contenga una barra lateral
- Una sección que contenga un apartado para tener checkboxes
- Una sección que contenga un apartado para tener select boxes

- **ACTIVATE**
  - Te invito a que revises el siguiente documento para fortalecer algunos conceptos relacionados a controles básicos de Streamlit (**streamlit_main_concepts.pdf**) lo puedes encontrar en materiales adicionales.

- **PRACTICA TEMA 1**
  - **Es momento de poner en práctica lo que aprendiste en este tema.**

Ejemplo 1 Actívate:

Descripción breve:

Deberás construir la estructura básica de una aplicación web que contenga una sección principal y además un apartado denominado barra lateral, el cual contendrá algunos controles para que se pueda manipular los datos sobre el proyecto de visualización de analítica de datos para WalMart USA.

En el siguiente enlace podrás encontrar el enlace al conjunto de datos que pertenecen a WaltMart USA.

https://raw.githubusercontent.com/jeaggo/tc3068/master/Superstore.csv

Análogamente el dataset **Superstore.csv** puede ser encontrado en materiales adicionales.

Instrucciones generales

Utilizando las instrucciones para desplegar una barra lateral en Streamlit se deberá construir la estructura básica general de una aplicación Web para analizar los datos de WalMart USA.

Ejercicio:

La estructura de la aplicación web deberá tener lo siguiente:

- **Un título que describa el proyecto de analítica de datos.**
- **Un subtítulo (este puede ser opcional) que de más contexto sobre el proyecto de analítica de datos.**
- **Un breve párrafo que describa el objetivo de este proyecto de analítica de datos; por ejemplo: predicción de ventas de productos de línea blanca en el noroeste de los Estados Unidos.**
- **Una sección del lado derecho a manera de una barra lateral que contenga dos secciones una para los controles de checkboxes y otra para los controles select boxes. Adicionalmente deberá tener una descripción explicando el porqué de la barra lateral. Nota: Los apartados para los controles solo se deberán indicar, no se espera que se incluyan los controles, solo indicar que ahí estarán.**

TEMA 2: Interacción con otros componentes (slider, radio, checkbox, date_input)

Con los widgets, Streamlit permite crear interactividad directamente en las aplicaciones con casillas de verificación, controles de fecha, botones, controles deslizantes y más. Esto es una de las cosas más significativas pues permite al científico de datos crear elementos que permitan que el análisis que está realizando sea más profundo.
Es recomendable indicar que existen muchos widgets pero nos vamos a concentrar en los 4 más importantes: controles deslizantes o sliders; botones de opciones ó radio buttons, controles checkbox y entradas de fecha.

NUTRE

Para recopilar las respuestas de los usuarios, en este ejemplo recopilaremos del usuario la fecha actual con el control **st.date_input()**. En este caso obtendremos del sistema la fecha actual con la función **datetime.data.today()** para sugerirla como fecha por default al control **st.date_input()**. Otra funcion importante es la de filtrar columnas tipo fecha de un Dataframe, este ejemplo se abordará más adelante.

En caso que la fecha haya sido seleccionada de forma adecuada usaremos st.sucess() para enviar un mensaje con la fecha seleccionada.

```
# Give user the current date
today = datetime.date.today()
today_date = st.date_input('Current date', today)
st.success('Current date: `%s`' % (today_date))
```

La figura 2 muestra el componente **st.date_input** generado con el código anterior.

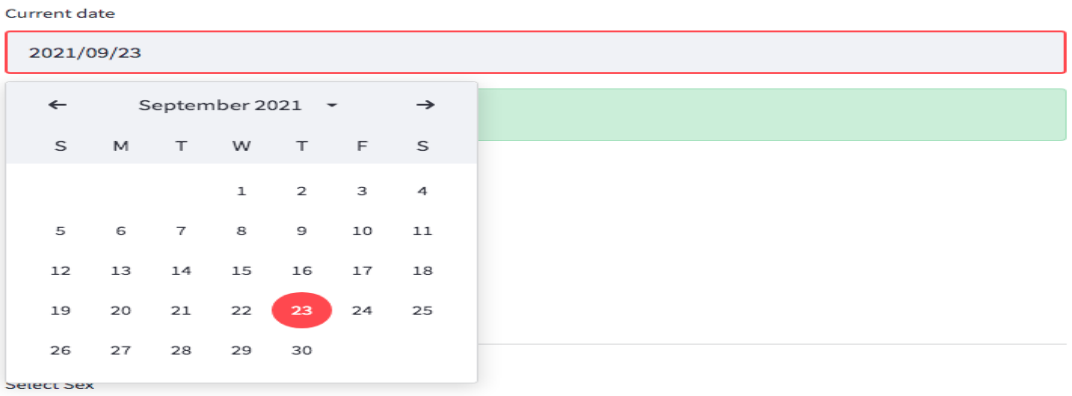


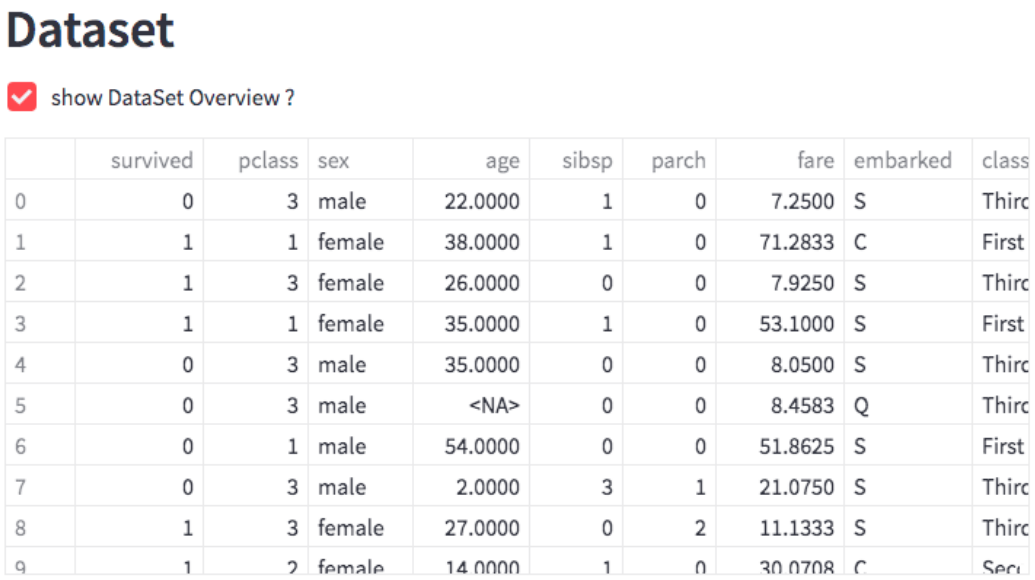
Figura 2. Selección de opciones con el botón de opción (radio button).

La siguiente sección usa el control checkbox, para preguntar al usuario si desea o no mostrar una tabla de datos. Una vez que se obtiene la respuesta se

se evalúa usando un if, si el resultado es true entonces se imprime el dataframe.
En las siguientes secciones se usará un dataset denominado **titanic.csv** que puedes encontrar en materiales adicionales.

```
# Display the content of the dataset if checkbox is true
st.header("Dataset")
agree = st.checkbox("show DataSet Overview ? ")
if agree:
 st.dataframe(titanic_data)
```

La figura 3 muestra la salida del componente **st.checkbox** generado con el código anterior.



	survived	pclass	sex	age	sibsp	parch	fare	embarked	class
0	0	3	male	22.0000	1	0	7.2500	S	Third
1	1	1	female	38.0000	1	0	71.2833	C	First
2	1	3	female	26.0000	0	0	7.9250	S	Third
3	1	1	female	35.0000	1	0	53.1000	S	First
4	0	3	male	35.0000	0	0	8.0500	S	Third
5	0	3	male	<NA>	0	0	8.4583	Q	Third
6	0	1	male	54.0000	0	0	51.8625	S	First
7	0	3	male	2.0000	3	1	21.0750	S	Third
8	1	3	female	27.0000	0	2	11.1333	S	Third
9	1	2	female	14.0000	1	0	30.0708	C	Seco

Figura 3. Control checkbox para mostrar/ocultar un dataframe

Para recopilar las respuestas de los usuarios, tenemos varias opciones para crear widgets de opción múltiple. El primer widget es el botón de opción (radio button). La instrucción **st.radio** se debe aplicar a una variable que para este ejemplo se denominará **selected_class**; la instrucción tiene dos secciones, una que indica el título del botón de opción y la otra que indica la información que alimentará dicho botón y para este ejemplo es la columna "***clase***" del conjunto de datos del Titanic.

Adicionalmente es necesario utilizar la instrucción **st.write** para plasmar el botón de opción en nuestra aplicación y además capturar la opción que el usuario haya seleccionado.

```
selected_town = st.radio("Select Embark Town",
titanic_data['embark_town'].unique())
st.write("Selected Embark Town:", selected_town)
```

Verá algo como se muestra a continuación (ver figura 4). Tenga en cuenta que la opción seleccionada será el valor real de las opciones mostradas en lugar del índice numérico seleccionado. En este caso, será el texto.

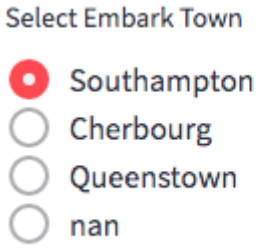


Figura 4. Selección de opciones con el botón de opción (radio button).

Además de este widget, Streamlit también proporciona un control deslizante (**slider**) que permite al usuario especificar una entrada de número sin teclear nada. A continuación, se muestra un ejemplo de código.

```
# Select a range of the fare and then display the dataset
optionals = st.expander("Optional Configurations", True)
fare_min = optionals.slider(
   "Minimum Fare",
   min_value=float(titanic_data['fare'].min()),
   max_value=float(titanic_data['fare'].max())
)
fare_max = optionals.slider(
   "Maximum Fare",
   min_value=float(titanic_data['fare'].min()),
   max_value=float(titanic_data['fare'].max())
)
subset_fare = titanic_data[(titanic_data['fare'] <= fare_max) &
(fare_min <= titanic_data['fare'])]
```

La primera instrucción establece una variable **optionals** que contendrá un control **st.expander**. Esta función es útil cuando desea ocultar algunos widgets opcionales.

Posteriormente se establece una variable que contendrá el valor seleccionado en la barra deslizante, para este ejemplo se utilizará las tarifas del conjunto de datos del Titanic

(columna "**fare**").Esta variable utiliza la instrucción **optionals.slider** para generar los máximos y mínimos. Cabe mencionar que esta instrucción utiliza datos que son numéricos, como precios, costos, temperatura, etc.

Adicionalmente la variable para este ejemplo se denomina "**subset_fare**" contendrá el conjunto de datos máximos y mínimos de los datos que se utilizarán para generar las barras deslizantes.

A continuación, se presenta el código completo.

```
%%writefile tercero.py
import pandas as pd
import streamlit as st
import datetime

titanic_link = '/content/titanic.csv'

titanic_data = pd.read_csv(titanic_link)

# Create the title for the web app
st.title("My First Streamlit App")
sidebar = st.sidebar
sidebar.title("This is the sidebar.")
sidebar.write("You can add any elements to the sidebar.")

# Give user the current date

today = datetime.date.today()
today_date = st.date_input('Current date', today)

st.success('Current date: `%s`' % (today_date))

# Display the content of the dataset if checkbox is true

st.header("Dataset")
agree = st.checkbox("show DataSet Overview ? ")
if agree:
 st.dataframe(titanic_data)
```

```
# Select the embark town of the passanger and then display the
dataset with this selection
selected_town = st.radio("Select Embark Town",
titanic_data['embark_town'].unique())
st.write("Selected Embark Town:", selected_town)

st.write(titanic_data.query(f"""embark_town==@selected_town"""))

st.markdown("___")

# Select a range of the fare and then display the dataset with this
selection
optionals = st.expander("Optional Configurations", True)
fare_min = optionals.slider(
   "Minimum Fare",
   min_value=float(titanic_data['fare'].min()),
   max_value=float(titanic_data['fare'].max())
)
fare_max = optionals.slider(
   "Maximum Fare",
   min_value=float(titanic_data['fare'].min()),
   max_value=float(titanic_data['fare'].max())
)
subset_fare = titanic_data[(titanic_data['fare'] <= fare_max) &
(fare_min <= titanic_data['fare'])]
st.write(f"Number of Records With Fare Between {fare_min} and
{fare_max}: {subset_fare.shape[0]}")

# Display of the dataset
st.dataframe(subset_fare)
```

La figura 5 muestra un la visualización de este proyecto

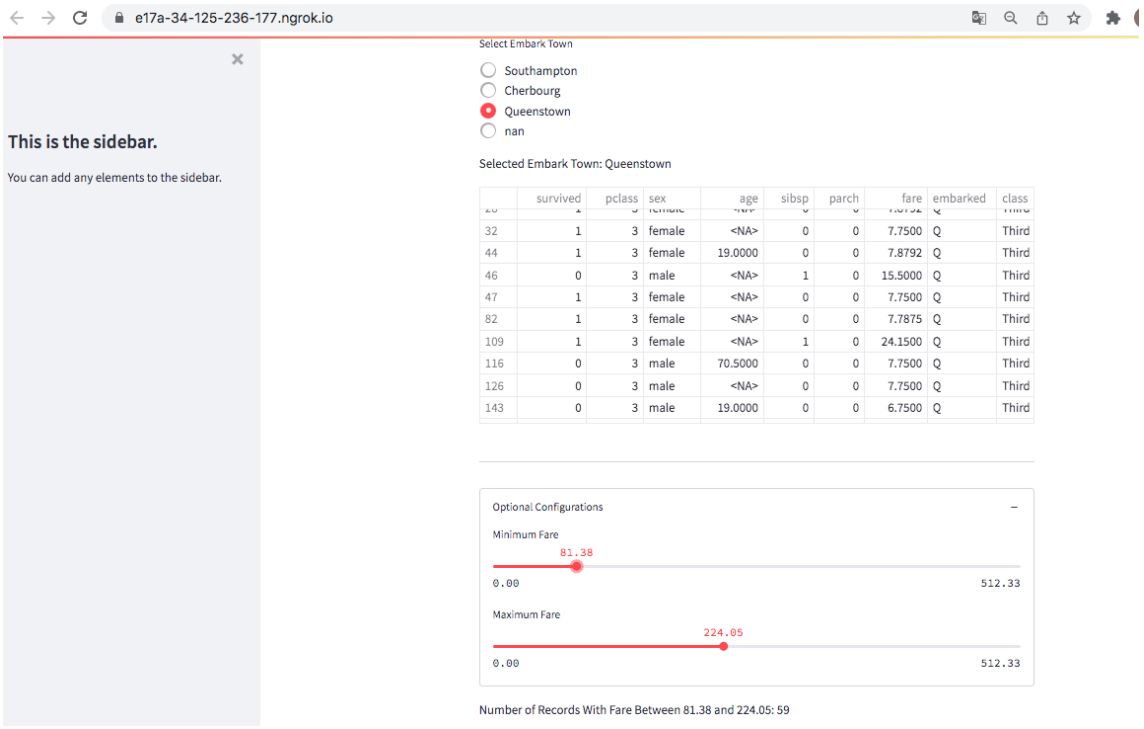


Figura 5. Vista de la aplicación web con Streamlit y Google Colab usando una barra lateral (sidebar) y los controles botón de opción; casilla de verificación y control deslizante.

- **SIGNIFICA**

Ahora que ya aprendiste como generar los controles más comunes en Streamlit para una aplicación web, como podrías construir un sitio web que contenga lo siguiente:

  - ¿Cuál sería el código para generar un control de checkbox para las categorías: 1, 2 y 3?
  - ¿Cuál sería el código para generar un control de selectbox para las regiones: Norte y Sur?
  - ¿Cuál sería el código para generar un control de slider para máximo precio: 10.00 y mínimo precio: 5:00?

- **ACTIVATE**
  - Te invito a que revises el siguiente documento para fortalecer algunos conceptos relacionados a controles básicos de Streamlit (**streamlit_main_concepts.pdf**) lo puedes encontrar en materiales adicionales.

- **PRACTICA TEMA 3**
  - **Es momento de poner en práctica lo que aprendiste en este tema.**

Ejemplo 1 Actívate:

Descripción breve:

Desarrollar un código sobre la estructura de una aplicación web que contenga 3 controles (radio, selectbox y slider) sobre el proyecto de visualización de analítica de datos para WalMart USA.
En el siguiente enlace podrás encontrar el enlace al conjunto de datos que pertenecen a WaltMart USA.

https://raw.githubusercontent.com/jeaggo/tc3068/master/Superstore.csv
El dataset **Superstore.csv** puede ser encontrado también en materiales adicionales

Instrucciones generales

El conjunto de datos de WalMart USA contiene los siguientes campos:

Row ID - Indice del registro de la orden
Order ID - Indice de la orden
Order Date - Fecha de la orden
Ship Date - Fecha de embarque
Ship Mode - Modo de embarque
Customer ID - Clave del cliente
Customer Name - Nombre del cliente
Segment - Segmento
Country - Pais
City - Ciudad
State - Estado
Postal Code - Código postal
Region - Region
Product ID - Clave del producto
Category - Categoría
Sub-Category - Sub categoría
Product Name - Nombre del producto
Sales - Ventas
Quantity - Cantidad vendida
Discount - Descuento aplicado
Profit - Ganancia

Ejercicio:

La estructura de la aplicación web deberá tener lo siguiente:

- **Deberá incluir un titulo**
- **Deberá incluir un subtitulo (este puede ser opcional)**
- **Deberá incluir un breve párrafo que describa el objetivo de este proyecto de analítica de datos; por ejemplo: predicción de ventas de productos de línea blanca en el noroeste de los Estados Unidos.**
- **Una sección que contenga los controles antes mencionados. Se sugiere los siguientes campos para cada control: El campos Ship Mode para el control**

radio. El campo Category para el control selectbox. El campo Discount para el control slider.

TEMA 3: Gráficas en Streamlit

Streamlit admite varias bibliotecas de gráficos diferentes, como Altair, Bokeh, Matplotlib e incluso Plotly. También soporta bibliotecas de gráficos interactivos como Vega Lite (gráficos en 2D) y deck.gl (mapas y gráficos en 3D). Sin embargo, también proporciona algunos tipos de gráficos que son nativos de Streamlit como **st.line_chart** que te permite crear un gráfico de líneas o st.area_chart que te permite crear un gráfico de área.

Streamlit puede generar gráficas de la librería **Matplotib**. Esta librería es una de las más populares y la primera librería de Python para la graficación de datos. La estructura de las instrucciones para graficar con Matplotlib en Streamlit es la siguiente:

```
st.pyplot(fig)
```

A continuación, veremos como se puede graficar en Matplotlib en Streamlit.

NUTRE
En esencia, cada aplicación Web optimizada es una secuencia de comandos de Python. En la parte superior del archivo, importamos las dependencias necesarias. En este caso, utilizaremos las 3 principales, como se muestra a continuación.

```
import pandas as pd
import streamlit as st
import matplotlib.pyplot as ptl
```

Las 3 librerías que vamos a usar serán pandas para la manipulación de los datos; Streamlit para construir la aplicación Web y matplotlib.pyplot para visualizar las gráficas.

En cuanto al conjunto de datos, usaremos el conjunto de datos del **titanic.csv**, que puedes descargar en la sección de materiales adicionales.

```
titanic_link = '/content/titanic.csv'
titanic_data = load_dataset(titanic_link)
```

Posteriormente vamos a desplegar el contenido de nuestro conjunto de datos (dataframe) para que el usuario pueda visualizar la información que vamos a graficar.

```
st.dataframe(titanic_data)
st.header("Data Description")
```

Y se verá algo como a continuación (ver figura 8).

Dataset Overview

	survived	pclass	sex	age	sibsp	parch	fare	embarked	c
0	0	3	male	22	1	0	7.2500	S	
1	1	1	female	38	1	0	71.2833	C	
2	1	3	female	26	0	0	7.9250	S	
3	1	1	female	35	1	0	53.1000	S	
4	0	3	male	35	0	0	8.0500	S	
5	0	3	male	NaN	0	0	8.4583	Q	
6	0	1	male	54	0	0	51.8625	S	
7	0	3	male	2	3	1	21.0750	S	
8	1	3	female	27	0	2	11.1333	S	
9	1	2	female	14	1	0	30.0708	C	S
10	1	3	female	4	1	1	16.7000	S	

Figura 8. Resumen de los datos.

Ahora que ya tenemos la parte introductoria, vamos a construir nuestra primera gráfica. Vamos a graficar un histograma, para lo cual las instrucciones se muestran a continuación:

```
fig, ax = plt.subplots()
ax.hist(titanic_data.fare)
st.header("Histograma del Titanic")
st.pyplot(fig)
```

La primera instrucción inicializa la variable fig que contendrá el histograma y la variable ax que nos servirá para construirlo. La siguiente instrucción nos permite construir el histograma con el campo “**fare**” de la base de datos del Titanic. La instrucción st.header nos sirve para poner un encabezado para nuestra gráfica. La instrucción st.pyplot gráfica nuestro histograma.

El siguiente paso será generar una gráfica de barras. Para lo cual vamos a utilizar dos campos de nuestro conjunto de datos, el campo “**class**” que tiene que ver con la clase de pasajeros del Titanic y el campo “**fare**” que indica la tarifa que pagó el pasajero. Para construir nuestra gráfica solo necesitamos las siguientes instrucciones:

```
fig2, ax2 = plt.subplots()

y_pos = titanic_data['class']
```

```
x_pos = titanic_data['fare']

ax2.barh(y_pos, x_pos)
ax2.set_ylabel("Class")
ax2.set_xlabel("Fare")
ax2.set_title('¿Cuanto pagaron las clases del Titanic')

st.header("Grafica de Barras del Titanic")
st.pyplot(fig2)
```

La primera instrucción inicializa la variable fig que contendrá la gráfica de barras y la variable ax que nos servirá para construirlo. Las variables **x_pos** y **y_pos** nos sirven para cargar los datos que corresponden a los campos "**class**" y "**fare**" respectivamente y que vamos a pasarle a la gráfica de barras.

Posteriormente la instrucción **ax2.barh** nos sirve para generar una gráfica de barras horizontal que contendrá los campos antes mencionados. Las instrucciones **ax2.set_xlabel** y **ax2.set_ylabel** nos sirven para poner etiquetas para el eje de las x y el de las y.

La instrucción **ax2.set_title** nos ayuda a poner un título para nuestra gráfica. La instrucción **st.header** como ya hemos visto, nos sirve para colocar un título para nuestra gráfica. La instrucción **st.pyplot** es la que le indica a Streamlit la gráfica que va a visualizar.

La última gráfica que vamos a realizar es una gráfica de dispersión, para lo cual utilizaremos los campos "**age**" que corresponde a la edad de los pasajeros y "**fare**" que corresponde a la tarifa que pagaron los pasajeros del Titanic. A continuación se muestra el código para hacer dicha gráfica:

```
fig3, ax3 = plt.subplots()

ax3.scatter(titanic_data.age, titanic_data.fare)
ax3.set_xlabel("Edad")
ax3.set_ylabel("Tarifa")

st.header("Grafica de Dispersión del Titanic")
st.pyplot(fig3)
```

La primera instrucción inicializa la variable fig que contendrá la gráfica de dispersión y la variable ax que nos servirá para construirlo. La instrucción **ax3.scatter** es la instrucción de Matplotlib que nos sirve para construir una gráfica de dispersión y esta recibe dos parámetros que en este caso serán los campos "**age**" y "**fare**".
Las instrucciones **ax3.set_xlabel** y **ax3.set_ylabel** nos permiten colocar etiquetas en el eje de las x y en el eje de las y. Vamos a utilizar la instrucción **st.header** para colocar un título para nuestra gráfica. Finalmente usamos la instrucción **st.pyplot** para indicarle a Streamlit que visualice la gráfica que deseamos.

Una vez que se ha entendido el funcionamiento de nuestra aplicación web en donde se pueden graficar diferentes tipos de gráficos, se muestra el código completo para su mejor comprensión.

```
%%writefile mi_app.py
import pandas as pd
import matplotlib.pyplot as plt
import streamlit as st

titanic_link = '/content/titanic.csv'
titanic_data = pd.read_csv(titanic_link)

fig, ax = plt.subplots()
ax.hist(titanic_data.fare)
st.header("Histograma del Titanic")
st.pyplot(fig)

st.markdown("___")

fig2, ax2 = plt.subplots()
y_pos = titanic_data['class']
x_pos = titanic_data['fare']
ax2.barh(y_pos, x_pos)
ax2.set_ylabel("Class")
ax2.set_xlabel("Fare")
ax2.set_title('¿Cuanto pagaron las clases del Titanic')

st.header("Grafica de Barras del Titanic")
st.pyplot(fig2)

st.markdown("___")

fig3, ax3 = plt.subplots()
ax3.scatter(titanic_data.age, titanic_data.fare)
ax3.set_xlabel("Edad")
ax3.set_ylabel("Tarifa")
st.header("Grafica de Dispersión del Titanic")
st.pyplot(fig3)
```

La figura 9, muestra la salida de la aplicación de gráficas.

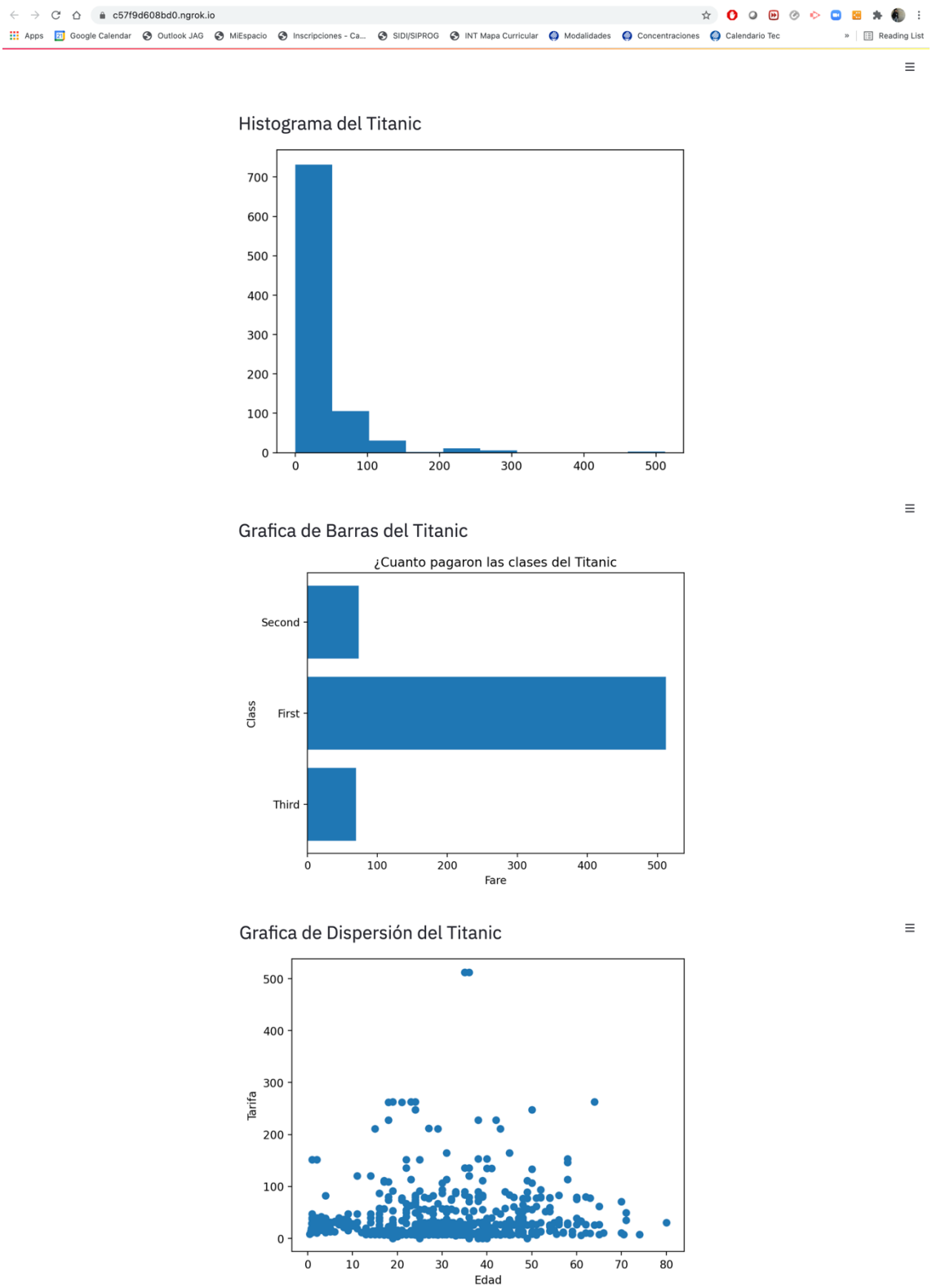


Figura 9. Vista de la aplicación web con Streamlit y Google Colab con la visualización de las gráficas.

- **SIGNIFICA**

Ahora que ya aprendiste cómo graficar en Streamlit para una aplicación Web, cómo podrías construir un sitio web que conteste a las siguientes interrogantes:

  - ¿Cuál sería el código para hacer una gráfica de barras que muestre las ventas de las categorías de WalMart USA?
  - ¿Cuál sería el código para hacer una gráfica de pastel que muestre las ventas por región de WaltMart USA?
  - ¿Cuál sería el código para hacer un histograma de los descuentos de los datos de WaltMart USA?

- **ACTIVATE**
  - Te invito a que revises el documento **streamlit_api_reference.pdf** para profundizar en conceptos relacionados con graficas en Streamlit. Este documento lo encontrarás en materiales adicionales.

- **PRACTICA TEMA 3**

  - **Es momento de poner en práctica lo que aprendiste en este tema.**

Ejemplo 1 Actívate:

Descripción breve:
Desarrollar un código sobre la estructura de una aplicación web que contenga 3 gráficas (barras, pastel y un histograma) sobre el proyecto de visualización de analítica de datos para WalMart USA.
En el siguiente enlace podrás encontrar el enlace al conjunto de datos que pertencen a WaltMart USA.
https://raw.githubusercontent.com/jeaggo/tc3068/master/Superstore.csv
El dataset **Superstore.csv** lo encontrarás en la sección de materiales adicionales.

Instrucciones generales
El conjunto de datos de WalMart USA contiene los siguientes campos:

Row ID - Indice del registro de la orden
Order ID - Indice de la orden
Order Date - Fecha de la orden
Ship Date - Fecha de embarque
Ship Mode - Modo de embarque
Customer ID - Clave del cliente

Customer Name - Nombre del cliente
Segment - Segmento
Country - Pais
City - Ciudad
State - Estado
Postal Code - Código postal
Region - Region
Product ID - Clave del producto
Category - Categoría
Sub-Category - Sub categoría
Product Name - Nombre del producto
Sales - Ventas
Quantity - Cantidad vendida
Discount - Descuento aplicado
Profit - Ganancia

Ejercicio:

La estructura de la aplicación web deberá tener lo siguiente:

- **Deberá incluir un titulo**
- **Deberá incluir un subtitulo (este puede ser opcional)**
- **Deberá incluir un breve párrafo que describa el objetivo de este proyecto de analítica de datos; por ejemplo: predicción de ventas de productos de línea blanca en el noroeste de los Estados Unidos.**
- **Una sección que contenga las gráficas antes mencionadas.**

TEMA 4: Visualización de Mapas

Streamlit admite bibliotecas para la visualización de mapas. Streamlit usa una motor de mapas llamado mapbox (https://www.mapbox.com/) Mapbox es una plataforma de código abierto para la creación de mapas enfocada principalmente a desarrolladores. Se basa en mapas vectoriales para diseñar y personalizar el estilo de los mapas según las necesidades de los usuarios.
Mapbox está basado en un proyecto de software libre openstreetmap https://www.openstreetmap.org/

Con la función st.map() podemos desplegar puntos GPS sobre un mapa. Para este primer ejemplo usaremos Numpy para generar algunos datos de ejemplo y los pintaremos sobre un mapa de la ciudad de San Francisco.

```
map_data = pd.DataFrame(
   np.random.randn(1000, 2) / [50, 50] + [37.76, -122.4],
   columns=['lat', 'lon'])

st.map(map_data)
```

Como se podrá observar el dataframe creado, tiene dos columnas llamadas **lat** y **lon**, las cuales son reconocidas por **st.map** como los puntos GPS que serán pintados en el mapa. Luego entonces cualquier fuente de datos de una API, o bien de excel que contenga esas 2 columnas **lat** y **lon** también serán reconocidas por **st.map()** como puntos GPS.

NUTRE

En esencia, cada aplicación Web con Streamlit puede contener mapas sin costos de licenciamiento o pago por uso. En este ejemplo completo colocaremos en la parte superior del archivo las dependencias necesarias. En este caso, utilizaremos las 3 principales, como se muestra a continuación.

```
import pandas as pd
import numpy as np
import streamlit as st
```

Las 3 librerías que vamos a usar serán pandas para la manipulación de los datos; Streamlit para construir la aplicación Web y numpy para la generación de los puntos GPS aleatorios.

```
map_data = pd.DataFrame(
   np.random.randn(1000, 2) / [50, 50] + [37.76, -122.4],
   columns=['lat', 'lon'])
```

La primera sección que va a tener nuestra aplicación Web será la correspondiente al título y a un encabezado de nuestro sitio web.

```
# Create the title for the web app
st.title("San francisco Map")
st.header("Using Streamlit and Mapbox")
```

Posteriormente vamos a desplegar el conjunto de datos GPS para que el usuario pueda visualizar la información que vamos a pintar en el mapa.

```
st.map(map_data)
```

Este archivo lo puede editar en Google Colab y deberá guardarlo con el nombre de **mapa_sanfrancisco.py**. La figura 10 muestra un ejemplo de la salida de este programa.

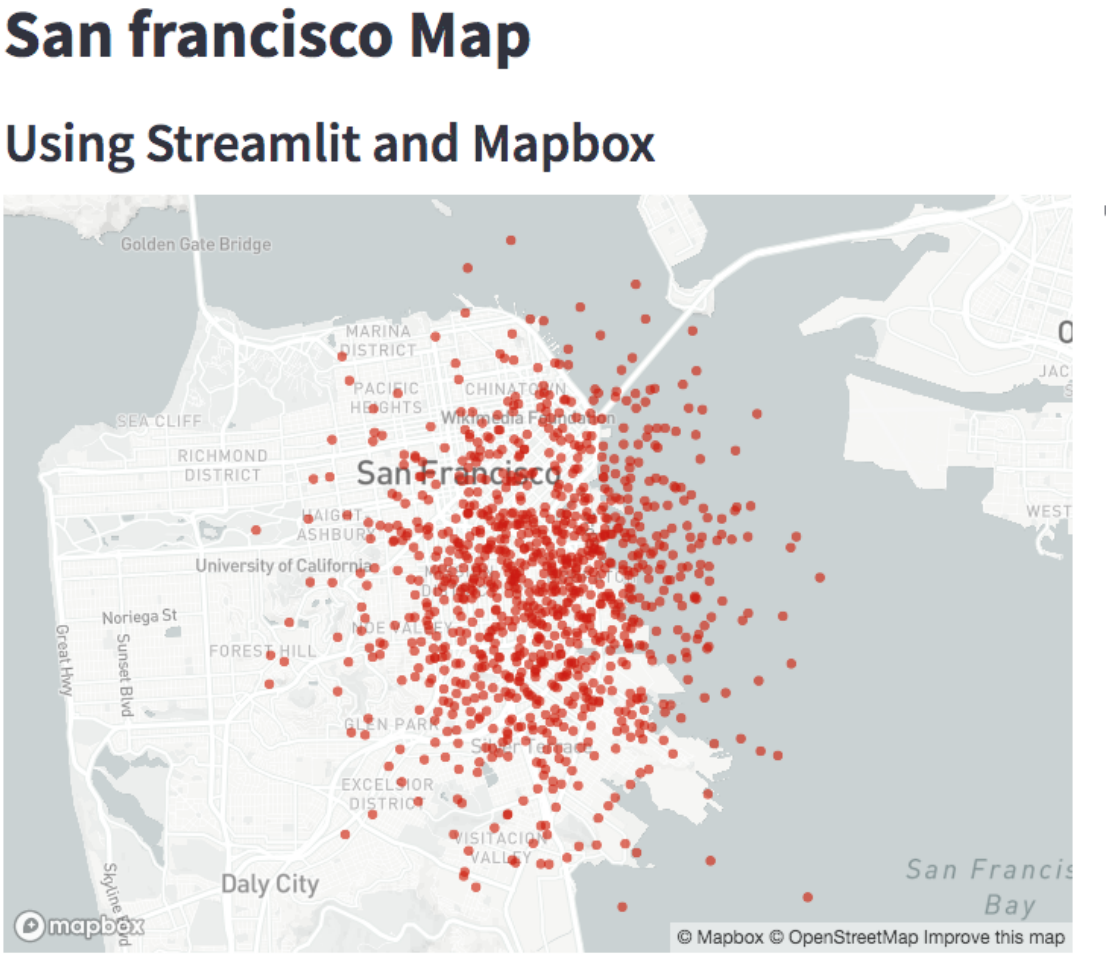


Figura 10. Vista de la aplicación Web con Streamlit y visualización de puntos gps generados por numpy

- **SIGNIFICA**

Ahora que ya aprendiste cómo visualizar mapas en Streamlit para una aplicación web, cómo podrías construir un sitio web que conteste a las siguientes interrogantes:

  - ¿Cuál sería el cambio en el código para poner la latitud y longitud de tu ubicación actual?
  - ¿Cuál sería el cambio en el código para generar solo 100 puntos gps?
  - ¿Cuál sería el código para generar esos puntos gps en un radio más amplio?

- **ACTIVATE**

- **PRACTICA TEMA 4**

- **Es momento de poner en práctica lo que aprendiste en este tema.**

Ejemplo 1 Actívate:

Descripción breve:

Desarrollar un código para una aplicación web que permita visualizar en un mapa de la ciudad de Nueva York los viajes realizados por la empresa Uber.

En el siguiente enlace podrás encontrar el enlace al conjunto de datos que pertenecen a Uber.

https://s3-us-west-2.amazonaws.com/streamlit-demo-data/uber-raw-data-sep14.csv.gz

La base de datos **uber_dataset.csv** la puedes encontrar en la sección de materiales adicionales.

Instrucciones generales

El conjunto de datos de Uber contiene los siguientes campos:

Date/Time - Fecha y hora de inicio de viaje
Lat - Latitud
Lon - Longitud
Base - Clave de base de operación

Ejercicio:

La estructura de la aplicación web deberá tener lo siguiente:

- **Deberá incluir un título**
- **Deberá incluir un subtítulo (este puede ser opcional)**
- **Deberá incluir un breve párrafo que describa el objetivo de este proyecto de analítica de datos; por ejemplo: Viajes de Uber en la ciudad de Nueva York con filtros por hora.**
- **Un control slider que permita filtrar los viajes por hora del día, usar la columna Date/Time.**
- **Una sección que contenga el mapa.**

NOTA adicionales:

- **Considere el uso de crear una función para la obtención de datos, ejemplo de función:**

```
@st.cache
def load_data(nrows):
    data = pd.read_csv(DATA_URL, nrows=nrows)
    lowercase = lambda x: str(x).lower()
    data.rename(lowercase, axis='columns', inplace=True)
    data[DATE_COLUMN] = pd.to_datetime(data[DATE_COLUMN])
    return data
```

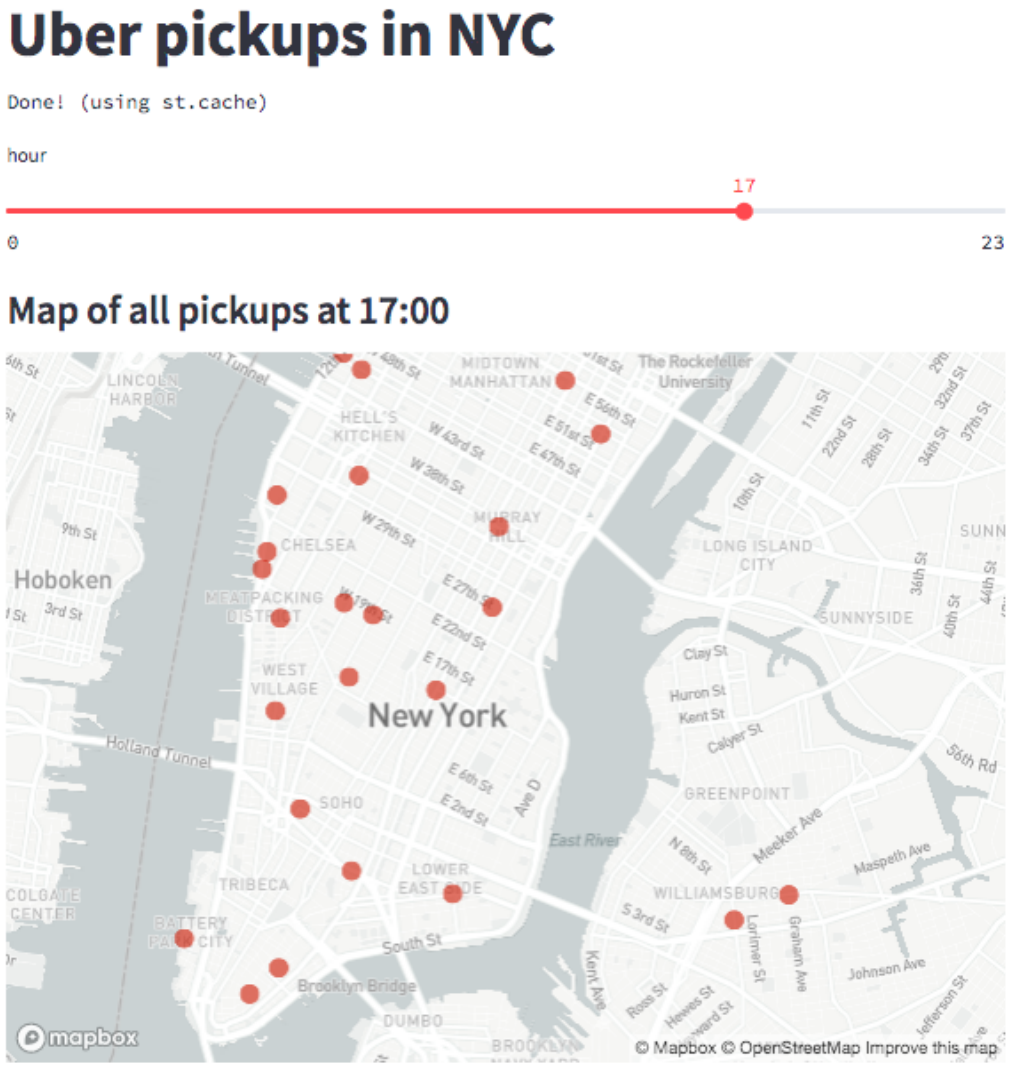
- **Debido a que la base de datos en muy grande, se sugiere recuperar solo una muestra:**

```
data = load_data(10000) # probar con 100, 1000, etc
```

- Como la columna Date/Time es de tipo Fecha y hora, se sugiere aplicar un filtro por hora:

```
filtered_data = data[data[DATE_COLUMN].dt.hour == hour_to_filter]
```

- Como ejemplo de salida se sugiere un template como el siguiente:



Para efectos de validación de resultados, se lista el código completo de esta proyecto:

```
%%writefile uber_ny.py
import pandas as pd
import numpy as np
import streamlit as st
```

```
st.title('Uber pickups in NYC')

DATE_COLUMN = 'date/time'
DATA_URL = ('/content/uber_dataset.csv')

@st.cache
def load_data(nrows):
    data = pd.read_csv(DATA_URL, nrows=nrows)
    lowercase = lambda x: str(x).lower()
    data.rename(lowercase, axis='columns', inplace=True)
    data[DATE_COLUMN] = pd.to_datetime(data[DATE_COLUMN])
    return data

data_load_state = st.text('Loading data...')
data = load_data(1000)
data_load_state.text("Done! (using st.cache)")

# Some number in the range 0-23
hour_to_filter = st.slider('hour', 0, 23, 17)
filtered_data = data[data[DATE_COLUMN].dt.hour == hour_to_filter]

st.subheader('Map of all pickups at %s:00' % hour_to_filter)
st.map(filtered_data)
```

3. Cierre (Ideas para llevar)

En este profundiza se aprendió acerca de los elementos principales con los que se puede construir una aplicación web sencilla que permita visualizar el análisis de datos de una organización o empresa. Hay que recordar que en una

aplicación web usando el marco de referencia Streamlit se debe de tener los siguientes elementos:

A) Titulo; Subtitulo y una descripción del proyecto
B) Una barra lateral (sidebar)
C) Controles o widgets
D) Y el uso de gráficas como barras, gráficas de pastel o histogramas.

4. Material adicional (se incluye por tema o al final del contenido)

Se recomienda revisar el siguiente enlace en donde vienen una serie de tutoriales que ayudaran a profundizar sobre este marco de referencia: https://docs.streamlit.io/en/stable/

- **Aquí se debe de incluir el COLAB GUIA NUTRE**