Indiana University Bloomington

Management and Access of Big and Complex Data

Final Project

Project title: Time series analysis of UFO sightings

Note: web application in this project is deployed and live on GCP, to view click this link (http://34.125.118.242:8081/)

Project file repository: (https://github.com/ANASOMARY/Mgmt.-Access-of-Big-and-Complex-Data)

Introduction

Unidentified Flying Object (UFO) as a topic has gained traction in recent years, and has been an intriguing "spooky" topic for years that it became part of our culture with many Hollywood movies and series about UFOs, although it is part of our culture, the subject of UFOs has been stigmatized in science communities and scientists have approached this subject with caution in fear for their careers (Dirk Schulz-Makuch) and of being called pseudo scientists or being described as tin foil hat conspiracy theorists.

Background

with the recent pentagon release of classified videos showing unidentified objects (Denise Chow and Gadi Schwartz, 2021) doing aerial maneuvers that break our physical laws and challenge the realm of what we thought possible and after the US senate formed an official committee to investigate UFO phenomenon where scientists and analysts are involved, it also became easier to study and analyze UFOs phenomenon because the data needed to do so became available and it became acceptable for legitimate news outlets to report on incidents of UFO sightings. On the other hand, the subject of UFOs with all the mysteries surrounding it is very intriguing and exciting to study given that the study was purely scientific and based on data.

For this project I will follow a scientific approach in analyzing UFO sightings data to try to draw subjective conclusions on whether UFOs are real or not and where those sightings happen, for the pure fun of it.

Methodology

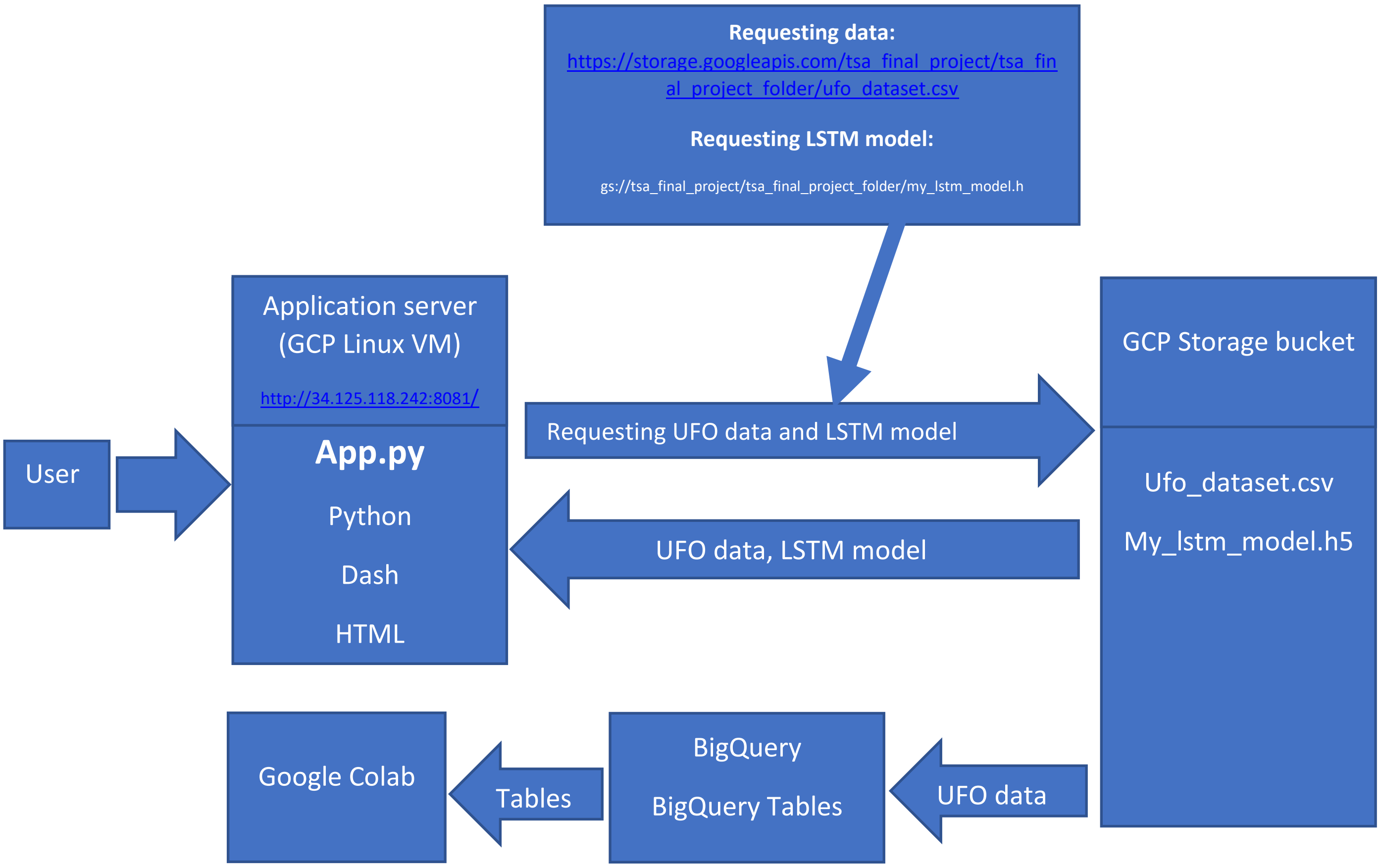
In this project I will apply some of the techniques that I have learned in this course to create and deploy a cloud-based web app that will plot basic distribution of UFO sightings and show a time series plot that reflects the count of sightings over time and predicts future number of UFO sightings using LSTM model, before building and deploying the web application, I will explore the dataset to get a feel of the data, I will create GCP big query tables and then will connect to those tables from google colab and explore data using python.

I will use UFO sightings dataset from data.world ((https://data.world/timothyrenner/ufo-sightings)), I will download the dataset CSV file from data.world and then will copy the file from my local machine into GCP storage bucket.

Data origin: The National UFO Research Center (NUFORC) This dataset contains the UFO sightings collected worldwide from 1969 till 2022 for the purpose of investigating and analyzing those sightings, as far as data reliability the dataset seems reliable and a result of actual sightings reported by actual people across the world, the dataset includes attributes like observation date and time, location, duration, UFO shape and other attributes in the raw form as it is recorded on the NUFORC site which makes this dataset a good candidate for time series analysis.

1. Architecture

- Dataset and other project files exist in a storage bucket
- Web application deployed on a VM instance accesses the storage bucket to get dataset and LSTM model
- Big Query accesses storage bucket to get dataset and populate Big Query tables
- I connect to Big Query tables through google Colab to explore data
- User accesses web application deployed on the VM instance and publicly available.


Requesting data:
https://storage.googleapis.com/tsa_final_project/tsa_final_project_folder/ufo_dataset.csv
Requesting LSTM model:
gs://tsa_final_project/tsa_final_project_folder/my_lstm_model.h
Application server
(GCP Linux VM)
http://34.125.118.242:8081/
App.py
Python
Dash
HTML
User
Requesting UFO data and LSTM model
UFO data, LSTM model
GCP Storage bucket
Ufo_dataset.csv
My_lstm_model.h5
Google Colab
Tables
BigQuery
BigQuery Tables
UFO data

In summary, these are the technologies that I used and how I used them:

1- Built a web application using python and Dash.
2- Created a GCP storage bucket to store web application file, dataset and LSTM.h5 model.
3- Created a VM instance, installed python and other libraries on it, configured the firewall, copied the application file from the storage bucket through ssh and finally, deployed the web application on it.

4- Created a Big Query dataset, and then connected it to the dataset in the storage bucket, then created 5 Big Query tables from that dataset.

5- Used google Colab to connect to the Big Query tables and explored dataset

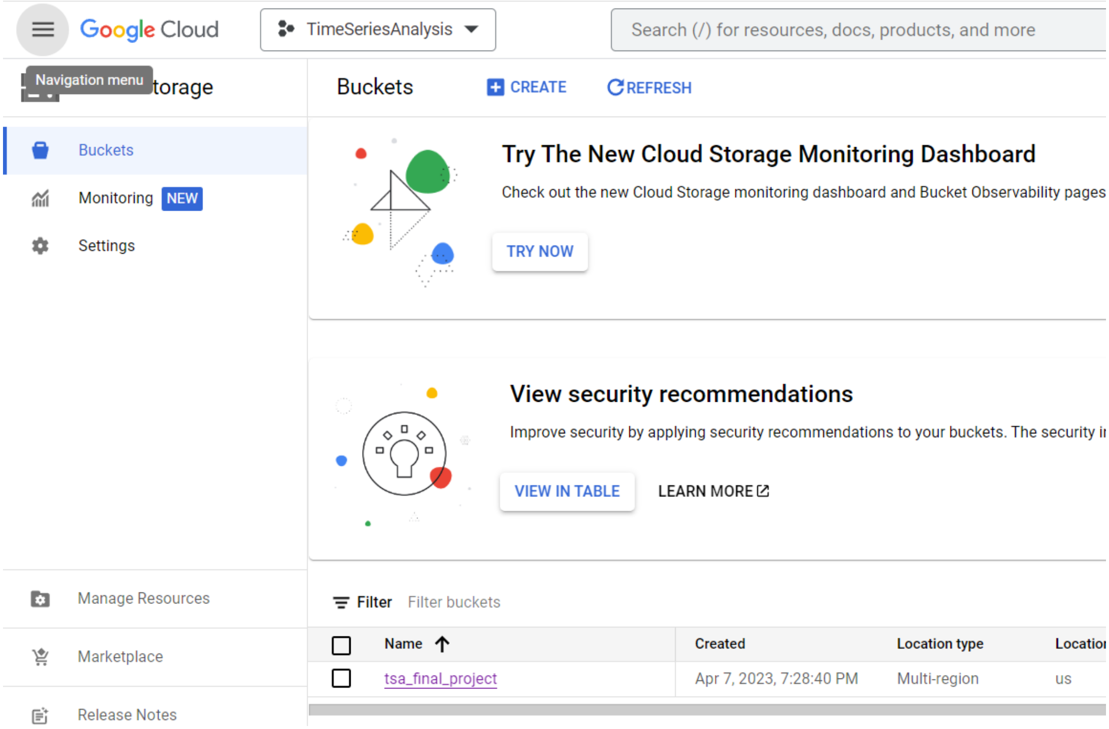
6- Within the web application I am connecting to the storage bucket and getting data from dataset as well as loading the LSTM model.

2. Setting up cloud environment - Cloud computing

I created a cloud environment with the following components to deploy the web application:

i. **Storage bucket:** this will host all project files:

- App.py: this is the web application file, It is stored in the storage bucket in order to be re-deployed easily on the VM
- Dataset: the dataset CSV file is stored in the storage bucket and is accessed by GCP Big Query to read data and create Big Query tables, as well as by the web application deployed on the VM instance.
- LSTM model: I have created an LSTM model to forecast the number of future UFO sightings and then saved the h5 file in the storage bucket, this file is loaded within the web application then the resulted predictions are added to the time series plot.



Try The New Cloud Storage Monitoring Dashboard

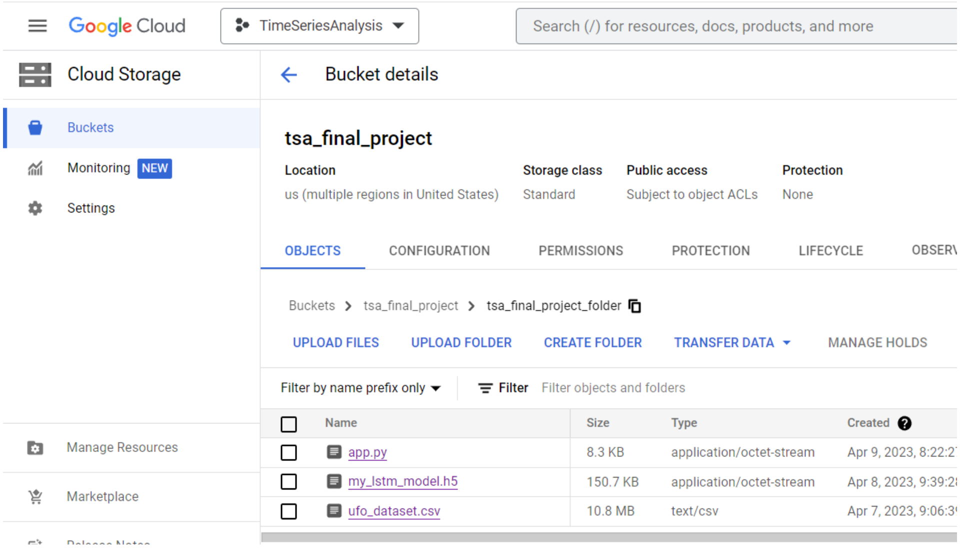
Check out the new Cloud Storage monitoring dashboard and Bucket Observability pages

View security recommendations

Improve security by applying security recommendations to your buckets. The security i

☐	Name ↑	Created	Location type	Locatio
☐	tsa_final_project	Apr 7, 2023, 7:28:40 PM	Multi-region	us

(screenshot: created a GCP storage bucket)



tsa_final_project

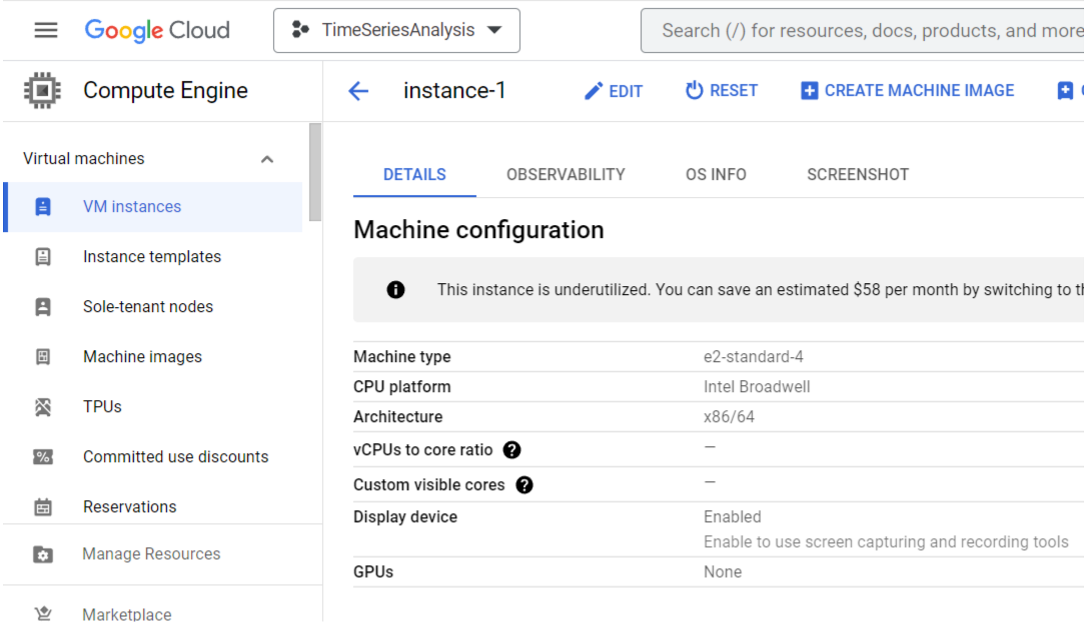
Location	Storage class	Public access	Protection
us (multiple regions in United States)	Standard	Subject to object ACLs	None

Buckets > tsa_final_project > tsa_final_project_folder

□	Name	Size	Type	Created
□	app.py	8.3 KB	application/octet-stream	Apr 9, 2023, 8:22:2
□	my_lstm_model.h5	150.7 KB	application/octet-stream	Apr 8, 2023, 9:39:2
□	ufo_dataset.csv	10.8 MB	text/csv	Apr 7, 2023, 9:06:3

(Screenshot: uploaded project files into storage bucket)

ii. **Virtual Machine:** this is where the web application file (app.py) is actually deployed, I have created a Linux ubuntu virtual machine and installed python on it in addition to all needed libraries like TensorFlow and Dash, then I did the needed firewall configurations to make the web application accessible and finally, I copied the application file (app.py) from the storage bucket into the virtual machine and deployed the web application file.

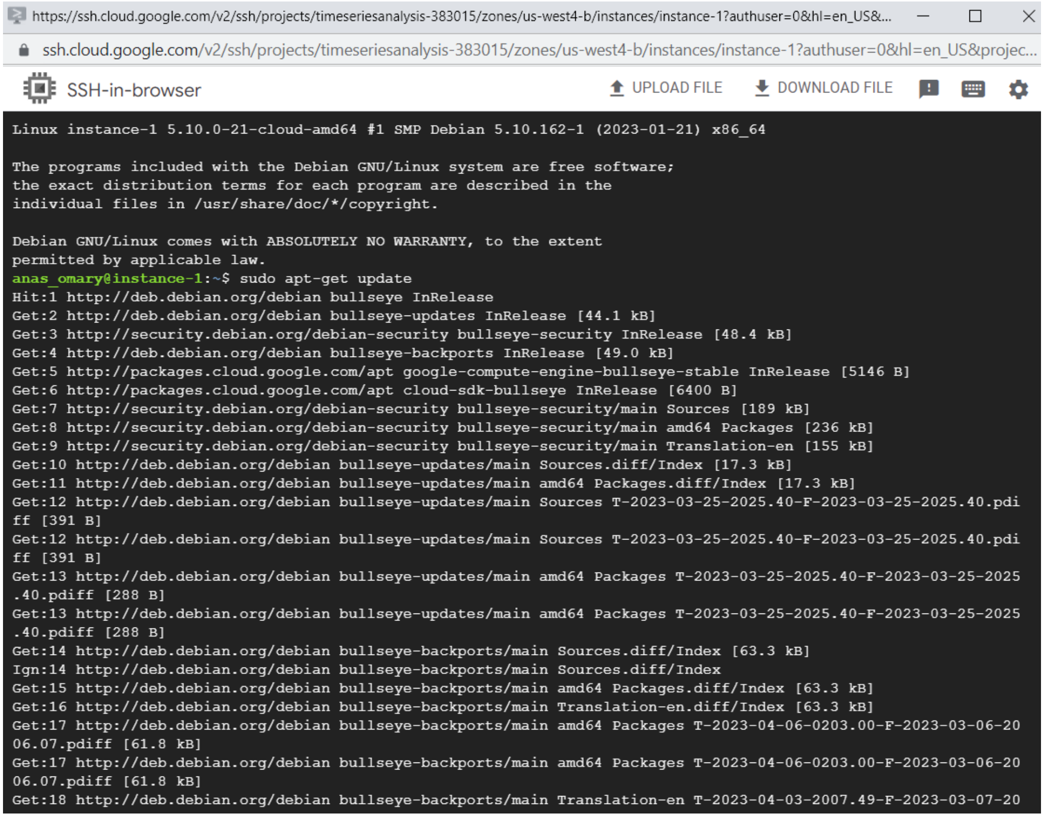


Machine configuration

This instance is underutilized. You can save an estimated $58 per month by switching to tl

Machine type	e2-standard-4
CPU platform	Intel Broadwell
Architecture	x86/64
vCPUs to core ratio	—
Custom visible cores	—
Display device	Enabled Enable to use screen capturing and recording tools
GPUs	None

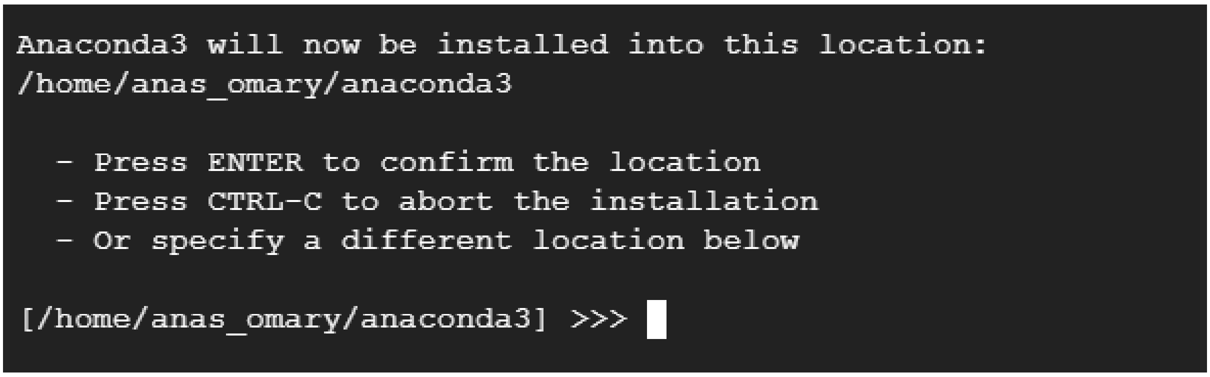
(Screenshot: created a virtual machine instance)



https://ssh.cloud.google.com/v2/ssh/projects/timeseriesanalysis-383015/zones/us-west4-b/instances/instance-1?authuser=0&hl=en_US&...
ssh.cloud.google.com/v2/ssh/projects/timeseriesanalysis-383015/zones/us-west4-b/instances/instance-1?authuser=0&hl=en_US&projec...
SSH-in-browser
UPLOAD FILE
DOWNLOAD FILE

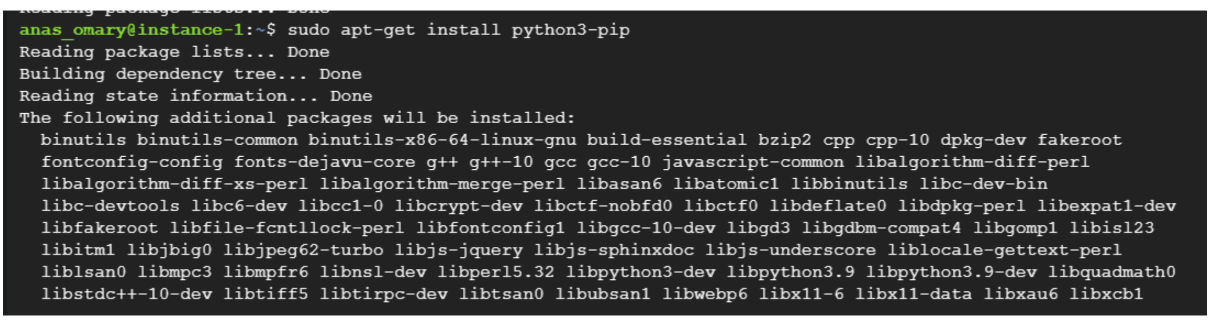
```
Linux instance-1 5.10.0-21-cloud-amd64 #1 SMP Debian 5.10.162-1 (2023-01-21) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
anas_omary@instance-1:~$ sudo apt-get update
Hit:1 http://deb.debian.org/debian bullseye InRelease
Get:2 http://deb.debian.org/debian bullseye-updates InRelease [44.1 kB]
Get:3 http://security.debian.org/debian-security bullseye-security InRelease [48.4 kB]
Get:4 http://deb.debian.org/debian bullseye-backports InRelease [49.0 kB]
Get:5 http://packages.cloud.google.com/apt google-compute-engine-bullseye-stable InRelease [5146 B]
Get:6 http://packages.cloud.google.com/apt cloud-sdk-bullseye InRelease [6400 B]
Get:7 http://security.debian.org/debian-security bullseye-security/main Sources [189 kB]
Get:8 http://security.debian.org/debian-security bullseye-security/main amd64 Packages [236 kB]
Get:9 http://security.debian.org/debian-security bullseye-security/main Translation-en [155 kB]
Get:10 http://deb.debian.org/debian bullseye-updates/main Sources.diff/Index [17.3 kB]
Get:11 http://deb.debian.org/debian bullseye-updates/main amd64 Packages.diff/Index [17.3 kB]
Get:12 http://deb.debian.org/debian bullseye-updates/main Sources T-2023-03-25-2025.40-F-2023-03-25-2025.40.pdi
ff [391 B]
Get:12 http://deb.debian.org/debian bullseye-updates/main Sources T-2023-03-25-2025.40-F-2023-03-25-2025.40.pdi
ff [391 B]
Get:13 http://deb.debian.org/debian bullseye-updates/main amd64 Packages T-2023-03-25-2025.40-F-2023-03-25-2025
.40.pdiff [288 B]
Get:13 http://deb.debian.org/debian bullseye-updates/main amd64 Packages T-2023-03-25-2025.40-F-2023-03-25-2025
.40.pdiff [288 B]
Get:14 http://deb.debian.org/debian bullseye-backports/main Sources.diff/Index [63.3 kB]
Ign:14 http://deb.debian.org/debian bullseye-backports/main Sources.diff/Index
Get:15 http://deb.debian.org/debian bullseye-backports/main amd64 Packages.diff/Index [63.3 kB]
Get:16 http://deb.debian.org/debian bullseye-backports/main Translation-en.diff/Index [63.3 kB]
Get:17 http://deb.debian.org/debian bullseye-backports/main amd64 Packages T-2023-04-06-0203.00-F-2023-03-06-20
06.07.pdiff [61.8 kB]
Get:17 http://deb.debian.org/debian bullseye-backports/main amd64 Packages T-2023-04-06-0203.00-F-2023-03-06-20
06.07.pdiff [61.8 kB]
Get:18 http://deb.debian.org/debian bullseye-backports/main Translation-en T-2023-04-03-2007.49-F-2023-03-07-20
```

(Screenshot: updated the virtual machine instance)



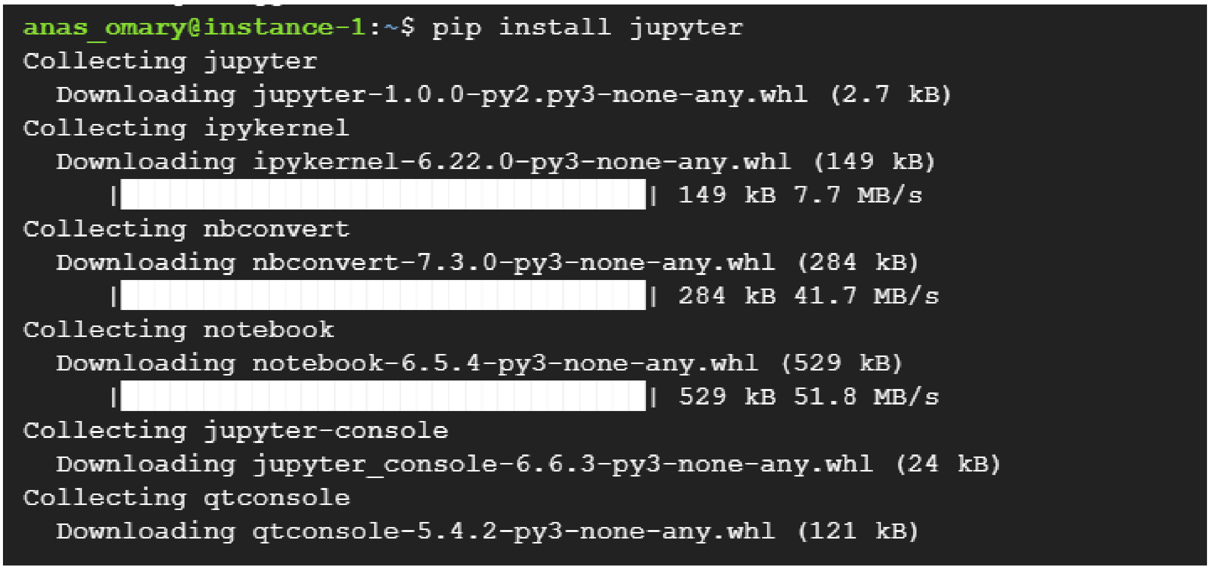
```
Anaconda3 will now be installed into this location:
/home/anas_omary/anaconda3

  - Press ENTER to confirm the location
  - Press CTRL-C to abort the installation
  - Or specify a different location below

[/home/anas_omary/anaconda3] >>>
```

(Screenshot: installed anaconda on VM instance)



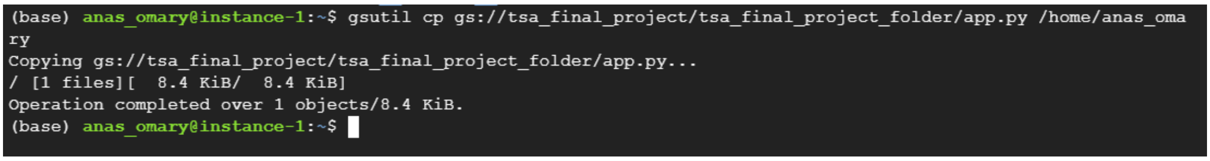
```
Reading package lists... Done
anas_omary@instance-1:~$ sudo apt-get install python3-pip
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu build-essential bzip2 cpp cpp-10 dpkg-dev fakeroot
  fontconfig-config fonts-dejavu-core g++ g++-10 gcc gcc-10 javascript-common libalgorithm-diff-perl
  libalgorithm-diff-xs-perl libalgorithm-merge-perl libasan6 libatomic1 libbinutils libc-dev-bin
  libc-devtools libc6-dev libcc1-0 libcrypt-dev libctf-nobfd0 libctf0 libdeflate0 libdpkg-perl libexpat1-dev
  libfakeroot libfile-fcntllock-perl libfontconfig1 libgcc-10-dev libgd3 libgdbm-compat4 libgomp1 libisl23
  libitm1 libjbig0 libjpeg62-turbo libjs-jquery libjs-sphinxdoc libjs-underscore liblocale-gettext-perl
  liblsan0 libmpc3 libmpfr6 libnsl-dev libperl5.32 libpython3-dev libpython3.9 libpython3.9-dev libquadmath0
  libstdc++-10-dev libtiff5 libtirpc-dev libtsan0 libubsan1 libwebp6 libx11-6 libx11-data libxau6 libxcb1
```

(Screenshot: installed python on VM)



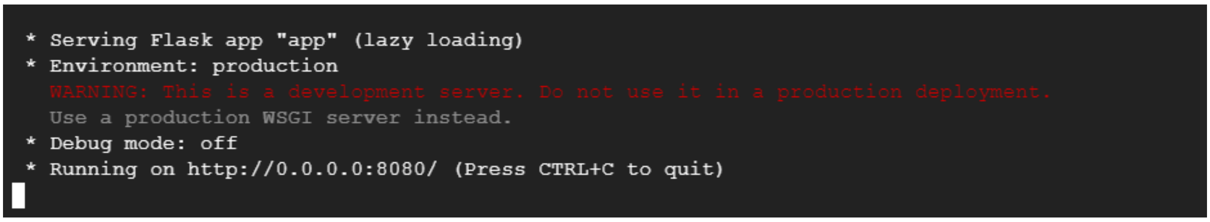
```
anas_omary@instance-1:~$ pip install jupyter
Collecting jupyter
  Downloading jupyter-1.0.0-py2.py3-none-any.whl (2.7 kB)
Collecting ipykernel
  Downloading ipykernel-6.22.0-py3-none-any.whl (149 kB)
     |████████████████████████████████| 149 kB 7.7 MB/s
Collecting nbconvert
  Downloading nbconvert-7.3.0-py3-none-any.whl (284 kB)
     |████████████████████████████████| 284 kB 41.7 MB/s
Collecting notebook
  Downloading notebook-6.5.4-py3-none-any.whl (529 kB)
     |████████████████████████████████| 529 kB 51.8 MB/s
Collecting jupyter-console
  Downloading jupyter_console-6.6.3-py3-none-any.whl (24 kB)
Collecting qtconsole
  Downloading qtconsole-5.4.2-py3-none-any.whl (121 kB)
```

(Screenshot: installed Jupyter on VM)



```
(base) anas_omary@instance-1:~$ gsutil cp gs://tsa_final_project/tsa_final_project_folder/app.py /home/anas_oma
ry
Copying gs://tsa_final_project/tsa_final_project_folder/app.py...
/ [1 files][  8.4 KiB/  8.4 KiB]
Operation completed over 1 objects/8.4 KiB.
(base) anas_omary@instance-1:~$ 
```

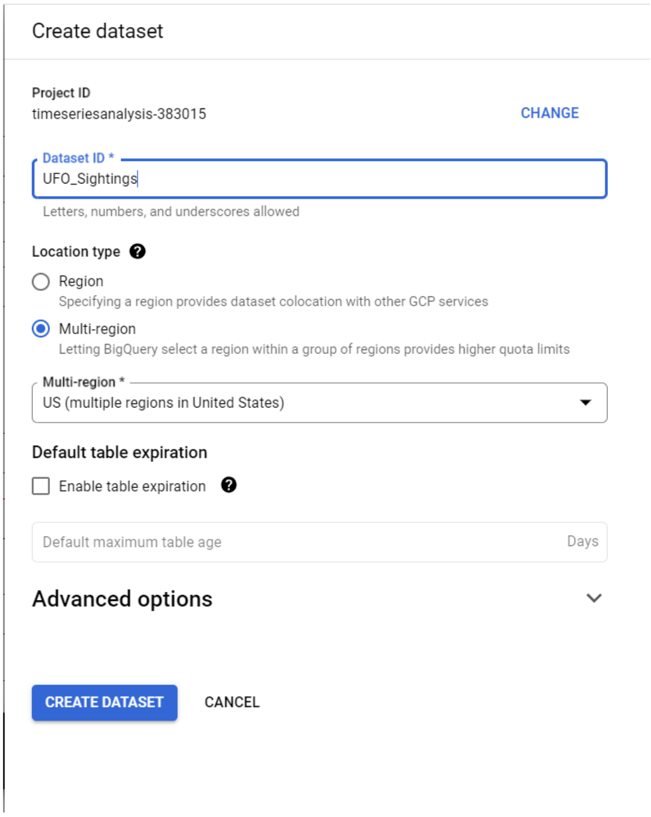
(Screenshot: copied the web application file from storage bucket into VM)



```
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://0.0.0.0:8080/ (Press CTRL+C to quit)
```

(Screenshot: deployed app.py on the VM)

iii. **Big Query:** I used big query to create 3 big query tables and used those tables to explore data in google colab, I imported the dataset from the storage bucket and used it in big query to create the mentioned tables.



Project ID
timeseriesanalysis-383015

Dataset ID *
UFO_Sightings
Letters, numbers, and underscores allowed

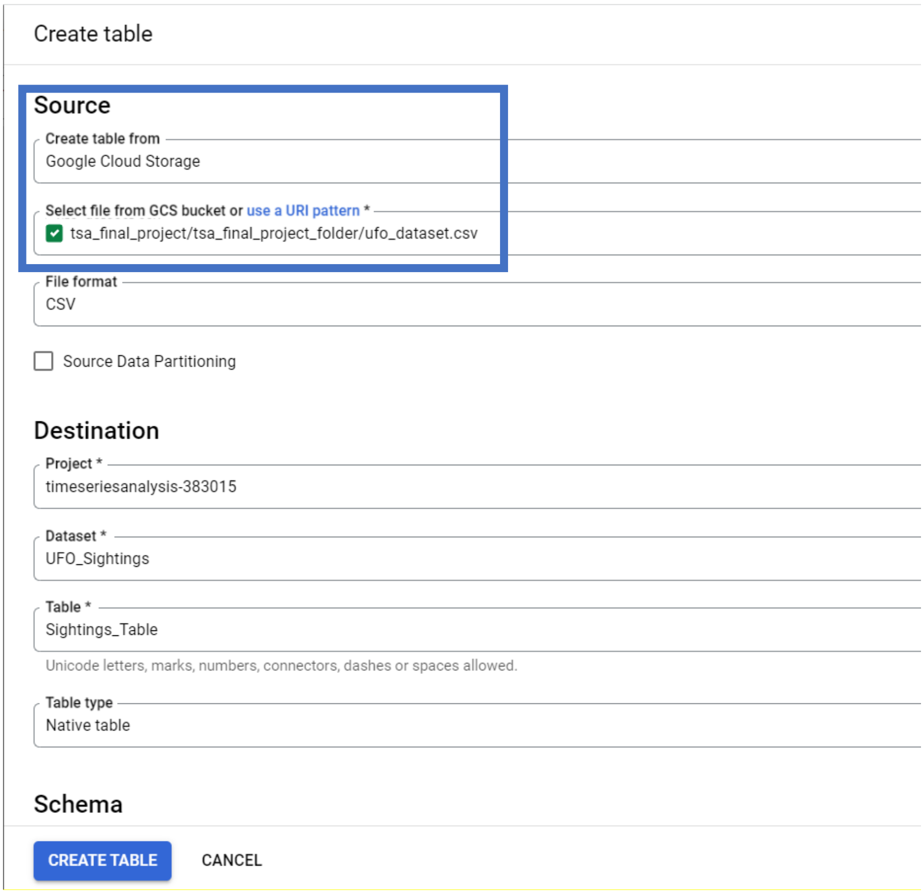
Location type
○ Region
Specifying a region provides dataset colocation with other GCP services
● Multi-region
Letting BigQuery select a region within a group of regions provides higher quota limits

Multi-region *
US (multiple regions in United States)

Default table expiration
□ Enable table expiration

Default maximum table age Days

(Screenshot: defined a dataset)



Source

Create table from
Google Cloud Storage

Select file from GCS bucket or use a URI pattern *
✓ tsa_final_project/tsa_final_project_folder/ufo_dataset.csv

File format
CSV

□ Source Data Partitioning

Destination

Project *
timeseriesanalysis-383015

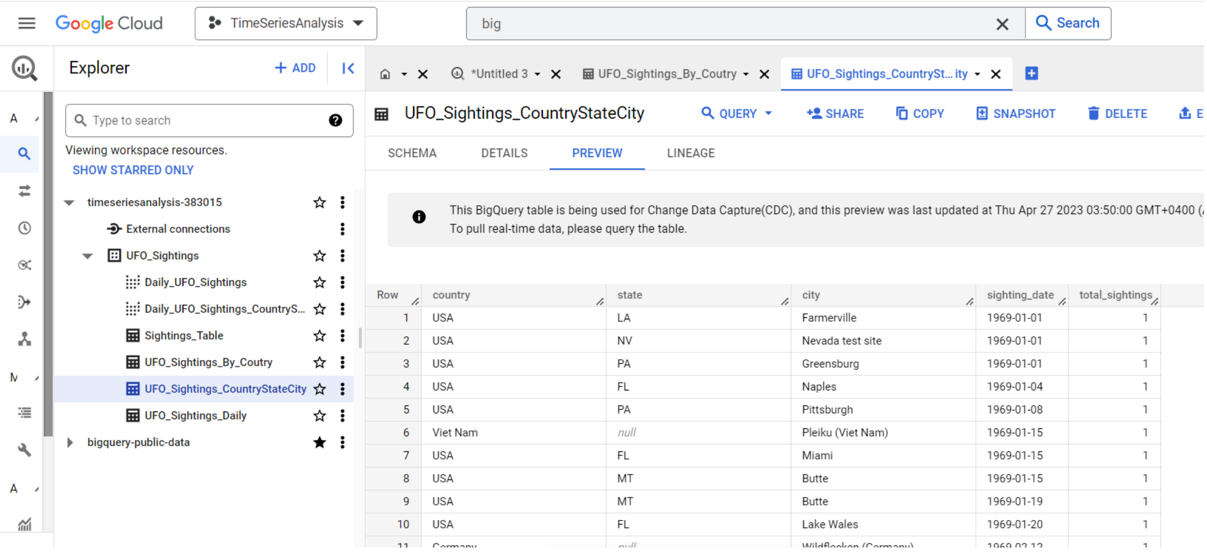
Dataset *
UFO_Sightings

Table *
Sightings_Table
Unicode letters, marks, numbers, connectors, dashes or spaces allowed.

Table type
Native table

Schema

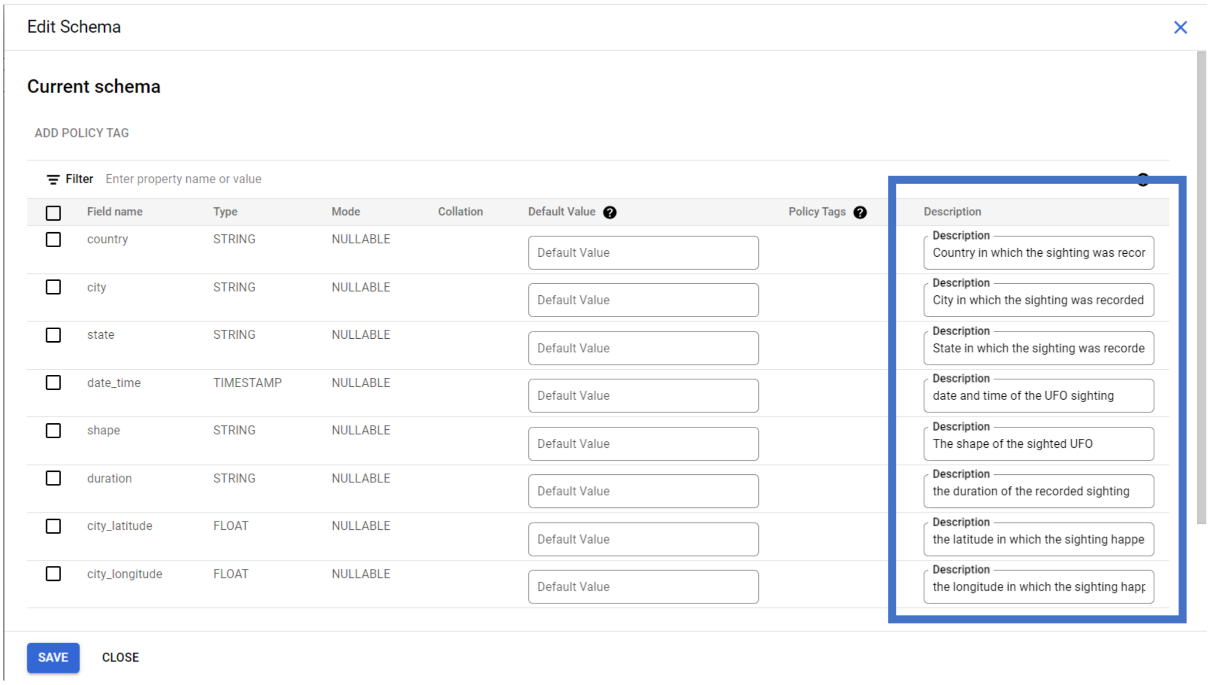
(Screenshot: imported data from GCP storage bucket)



This BigQuery table is being used for Change Data Capture(CDC), and this preview was last updated at Thu Apr 27 2023 03:50:00 GMT+0400 (
To pull real-time data, please query the table.

Row	country	state	city	sighting_date	total_sightings
1	USA	LA	Farmerville	1969-01-01	1
2	USA	NV	Nevada test site	1969-01-01	1
3	USA	PA	Greensburg	1969-01-01	1
4	USA	FL	Naples	1969-01-04	1
5	USA	PA	Pittsburgh	1969-01-08	1
6	Viet Nam	null	Pleiku (Viet Nam)	1969-01-15	1
7	USA	FL	Miami	1969-01-15	1
8	USA	MT	Butte	1969-01-15	1
9	USA	MT	Butte	1969-01-19	1
10	USA	FL	Lake Wales	1969-01-20	1
11	Germany	null	Wildflecken (Germany)	1969-02-12	1

(Screenshot: Created 3 big query tables)



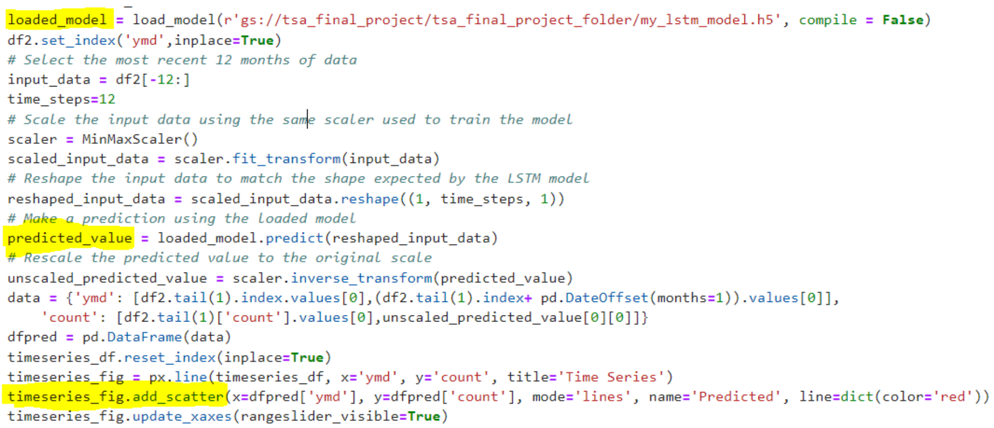
	Field name	Type	Mode	Collation	Default Value	Policy Tags	Description
□	country	STRING	NULLABLE		Default Value		Country in which the sighting was recor
□	city	STRING	NULLABLE		Default Value		City in which the sighting was recorded
□	state	STRING	NULLABLE		Default Value		State in which the sighting was recorde
□	date_time	TIMESTAMP	NULLABLE		Default Value		date and time of the UFO sighting
□	shape	STRING	NULLABLE		Default Value		The shape of the sighted UFO
□	duration	STRING	NULLABLE		Default Value		the duration of the recorded sighting
□	city_latitude	FLOAT	NULLABLE		Default Value		the latitude in which the sighting happe
□	city_longitude	FLOAT	NULLABLE		Default Value		the longitude in which the sighting happ

(Screenshot: defined metadata for each table)

3. Building the web app

Note: the app is deployed and live on (http://34.125.118.242:8081/)

The app.py application running on the virtual machine is a web application developed using python and dash, it connects directly to the CSV dataset that is located on a GCP storage bucket and is configured to be publicly available, the app.py calls a saved LSTM model which is the outcome of testing 3 different models (ARIMA, SARIMA and LSTM) I decided to use the LSTM model because it scored the highest, the LSTM model is saved on the same storage bucket (my_lstm_model.h5), Following screenshot shows how the model is loaded, how predictions are made, and how the resulted predictions are included in the time series plot.



```
loaded_model = load_model(r'gs://tsa_final_project/tsa_final_project_folder/my_lstm_model.h5', compile = False)
df2.set_index('ymd',inplace=True)
# Select the most recent 12 months of data
input_data = df2[-12:]
time_steps=12
# Scale the input data using the same scaler used to train the model
scaler = MinMaxScaler()
scaled_input_data = scaler.fit_transform(input_data)
# Reshape the input data to match the shape expected by the LSTM model
reshaped_input_data = scaled_input_data.reshape((1, time_steps, 1))
# Make a prediction using the loaded model
predicted_value = loaded_model.predict(reshaped_input_data)
# Rescale the predicted value to the original scale
unscaled_predicted_value = scaler.inverse_transform(predicted_value)
data = {'ymd': [df2.tail(1).index.values[0],(df2.tail(1).index+ pd.DateOffset(months=1)).values[0]],
    'count': [df2.tail(1)['count'].values[0],unscaled_predicted_value[0][0]]}
dfpred = pd.DataFrame(data)
timeseries_df.reset_index(inplace=True)
timeseries_fig = px.line(timeseries_df, x='ymd', y='count', title='Time Series')
timeseries_fig.add_scatter(x=dfpred['ymd'], y=dfpred['count'], mode='lines', name='Predicted', line=dict(color='red'))
timeseries_fig.update_xaxes(rangeslider_visible=True)
```

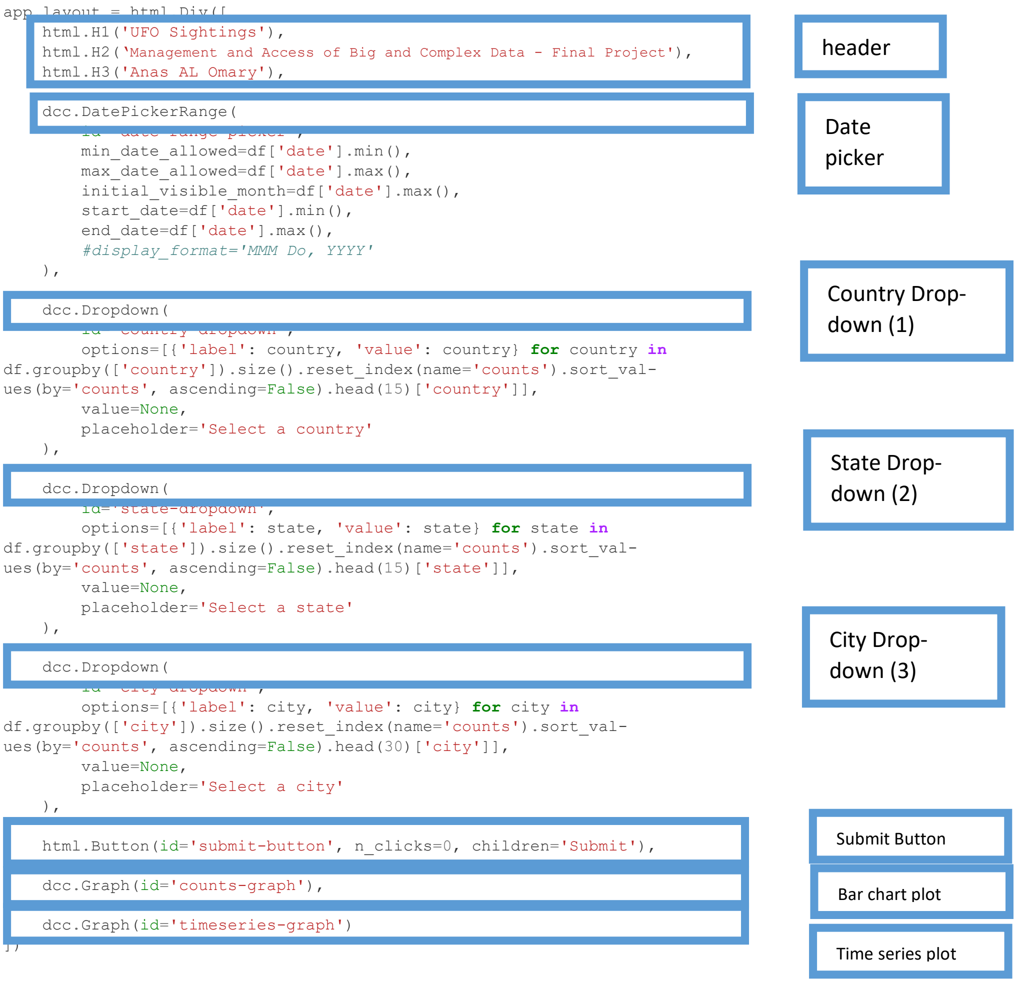
The connection security was not a big concern since the data is already publicly available and does not contain any sensitive information.

The front-end layout is created using dash and HTML, I used “dash_html_components” library to create HTML elements, and used dash_core_components to create the UI controls, the UI consists of one page which has a header, date range picker, three drop-down lists and two graph elements:

When the app first loads, it shows the date range picker with date period between January first 1969 and December 22 2022, then it loads data into the three drop down lists.

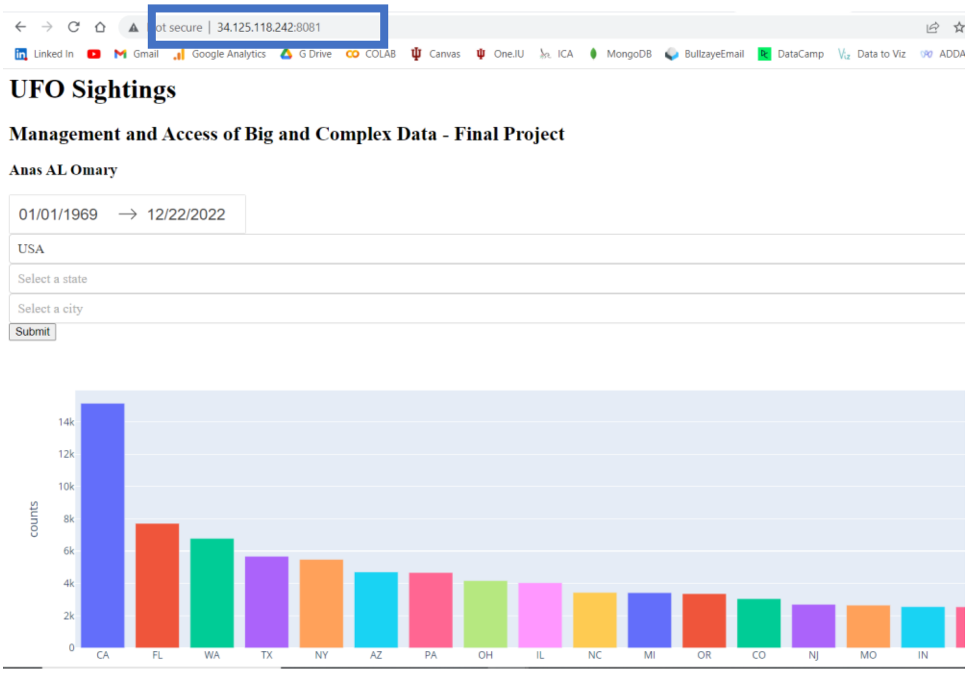
By default, the app loads the entire time series data from January first 1969 till December 22 2022 showing observations worldwide.

The user can filter by a specific time period, country, state and city, then click on submit, this will apply the filter on the bar-chart accordingly, then it will apply the filter on the time series visualization and recalculate predictions.



```
app.layout = html.Div([
    html.H1('UFO Sightings'),
    html.H2('Management and Access of Big and Complex Data - Final Project'),
    html.H3('Anas AL Omary'),
    dcc.DatePickerRange(
        id='date-range-picker',
        min_date_allowed=df['date'].min(),
        max_date_allowed=df['date'].max(),
        initial_visible_month=df['date'].max(),
        start_date=df['date'].min(),
        end_date=df['date'].max(),
        #display_format='MMM Do, YYYY'
    ),
    dcc.Dropdown(
        id='country-dropdown',
        options=[{'label': country, 'value': country} for country in
df.groupby(['country']).size().reset_index(name='counts').sort_val-
ues(by='counts', ascending=False).head(15)['country']],
        value=None,
        placeholder='Select a country'
    ),
    dcc.Dropdown(
        id='state-dropdown',
        options=[{'label': state, 'value': state} for state in
df.groupby(['state']).size().reset_index(name='counts').sort_val-
ues(by='counts', ascending=False).head(15)['state']],
        value=None,
        placeholder='Select a state'
    ),
    dcc.Dropdown(
        id='city-dropdown',
        options=[{'label': city, 'value': city} for city in
df.groupby(['city']).size().reset_index(name='counts').sort_val-
ues(by='counts', ascending=False).head(30)['city']],
        value=None,
        placeholder='Select a city'
    ),
    html.Button(id='submit-button', n_clicks=0, children='Submit'),
    dcc.Graph(id='counts-graph'),
    dcc.Graph(id='timeseries-graph')
])
```

(Screenshot: UI in python and dash code (zoom-in to view code))



UFO Sightings

Management and Access of Big and Complex Data - Final Project

Anas AL Omary

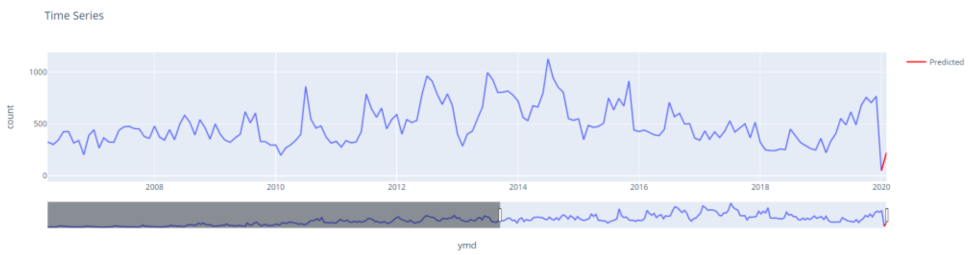
01/01/1969 → 12/22/2022

USA

Select a state

Select a city

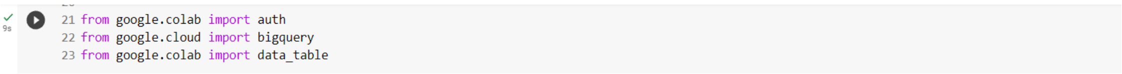
Submit



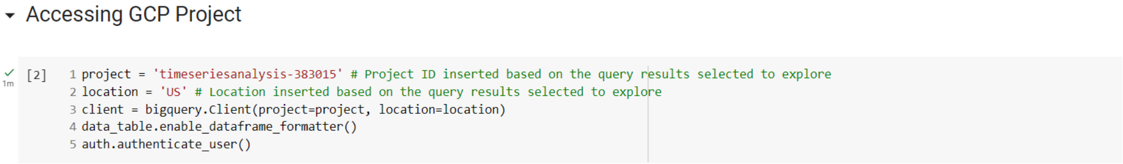
(Screenshot: Web application deployed on VM instance (http://34.125.118.242:8081/))

Results

To explore data, I connected to the three Big Query tables through google Colab, then saved the rows from each table into a pandas data frame.

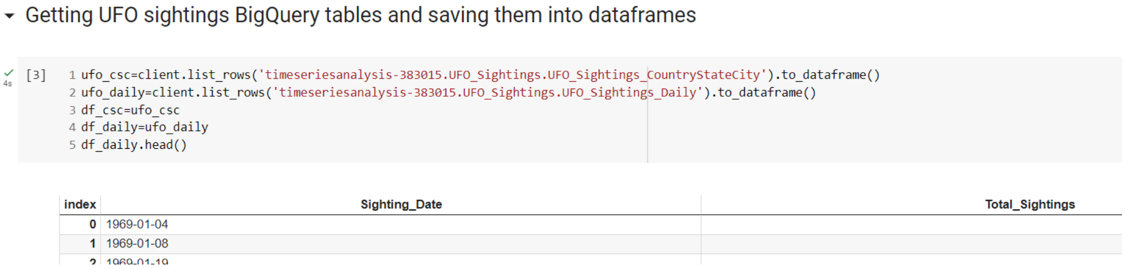


```
from google.colab import auth
from google.cloud import bigquery
from google.colab import data_table
```



▾ Accessing GCP Project

```
[2]
project = 'timeseriesanalysis-383015' # Project ID inserted based on the query results selected to explore
location = 'US' # Location inserted based on the query results selected to explore
client = bigquery.Client(project=project, location=location)
data_table.enable_dataframe_formatter()
auth.authenticate_user()
```

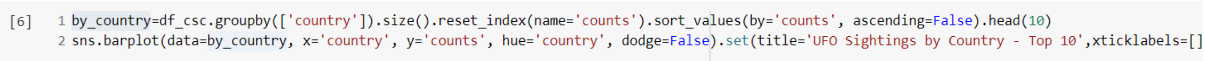


▾ Getting UFO sightings BigQuery tables and saving them into dataframes

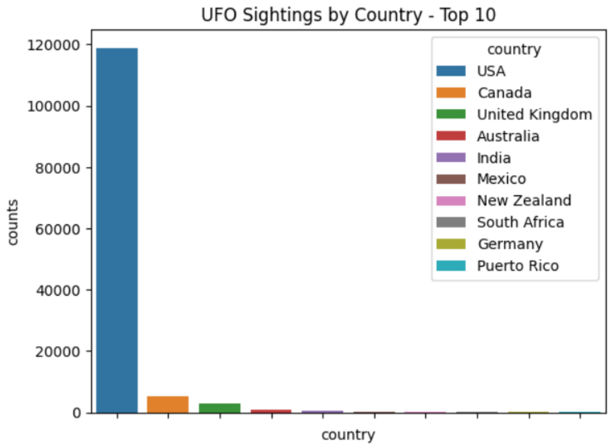
```
[3]
ufo_csc=client.list_rows('timeseriesanalysis-383015.UFO_Sightings.UFO_Sightings_CountryStateCity').to_dataframe()
ufo_daily=client.list_rows('timeseriesanalysis-383015.UFO_Sightings.UFO_Sightings_Daily').to_dataframe()
df_csc=ufo_csc
df_daily=ufo_daily
df_daily.head()
```

index	Sighting_Date	Total_Sightings
0	1969-01-04	
1	1969-01-08	
2	1969-01-10	

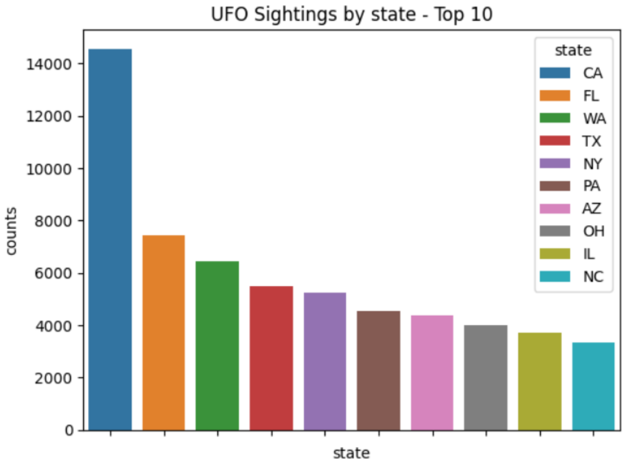
I started by examining the distribution of the total number of daily UFO sightings world-wide, I found that USA has significantly more total UFO sightings compared to other countries, USA had (118,865) sightings from 1969-2020 compared to the next country which is Canada with (5,466) sightings so more than 20 times the number of sightings.



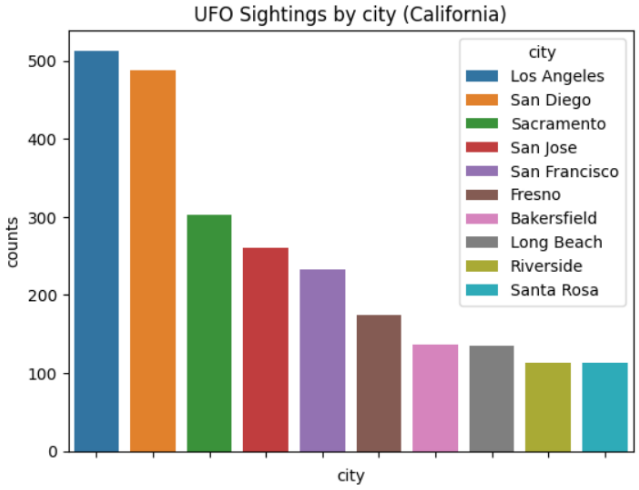
```
[6]
by_country=df_csc.groupby(['country']).size().reset_index(name='counts').sort_values(by='counts', ascending=False).head(10)
sns.barplot(data=by_country, x='country', y='counts', hue='country', dodge=False).set(title='UFO Sightings by Country - Top 10',xticklabels=[]
```



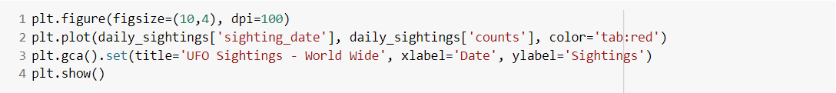
I then examined USA states to see the distribution of the number of daily UFO sightings and plotted the top 10 states with the highest number of UFO sightings, California came first with (14,559) sightings from 1969-2020 which is significantly higher compared to the second highest number of UFO sightings which was recorded in Florida with (7,413) sightings



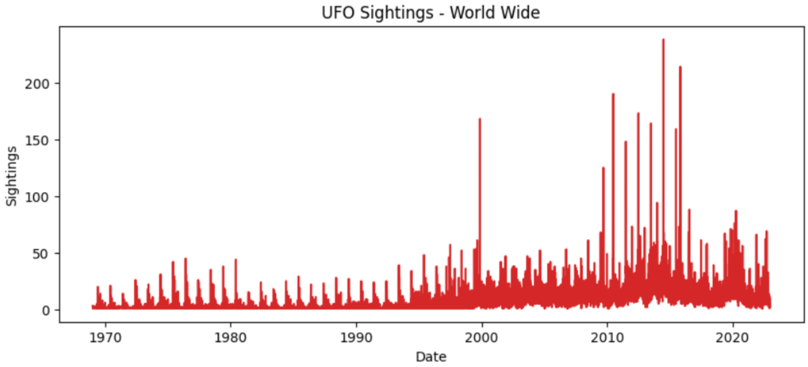
I then examined the distribution of the total number of daily UFO sightings in California cities and found that Los Angeles had the highest number with (513) total sightings followed by San Diego with (488) sightings.



Finally, I examined the daily total UFO sightings as a time series to see how UFO sightings are distributed over time and I found that till 1994 the total number of daily UFO sightings was approximately steady, but on 1995 onwards I noticed a significant increase in total sightings, after 1995 the total number of sightings continued increasing till 2020, there are spikes between 1995 and 2020 but the trend in general is moving upwards.



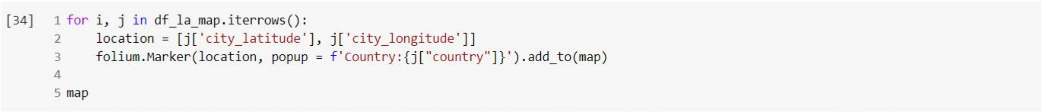
```
plt.figure(figsize=(10,4), dpi=100)
plt.plot(daily_sightings['sighting_date'], daily_sightings['counts'], color='tab:red')
plt.gca().set(title='UFO Sightings - World Wide', xlabel='Date', ylabel='Sightings')
plt.show()
```



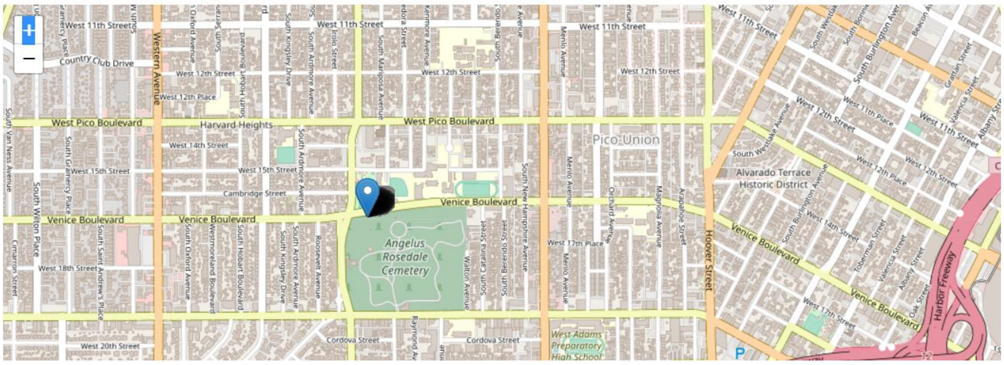
Discussion

1. Interpretation of the results

The results of examining UFO sightings dataset looked very interesting, the first interesting point is that – according to the dataset in hand – USA had significantly more recorded UFO sightings than the rest of the world combined, one theory might be that UFOs choose to always appear in USA, another theory might be that the NUFORC organization is more active in recording UFO sighting incidents in USA than other countries in the world, in other words NUFORC might be popular in USA and not very well known in other countries. Another interesting point is that among USA states California had the most UFO sightings almost twice the number than the second state Florida, zooming into California we notice that the Los Angeles and San Diego had the most UFO sighting incidents among other California cities, I couldn’t help thinking in a biased way because I know that Hollywood (where all the UFO movies are produced) is a district within the city of Los Angeles, but unfortunately the coordinates recorded with each sighting case does not correspond to the actual location of the sighting but rather the city coordinates, in other words all records of sightings that happened in Los Angeles have the same coordinates, so it was not possible to verify the district in which the sighting case happened, more data is required to verify the theory that most cases in Los Angeles happens in Hollywood.



```
[34]  for i, j in df_la_map.iterrows():
          location = [j['city_latitude'], j['city_longitude']]
          folium.Marker(location, popup = f'Country:{j["country"]}').add_to(map)

      map
```



Perhaps the most stunning discovery that I came across while examining the dataset is when I examined the time series plot and noticed a significant surge in the number of UFO sightings across the world starting the year 1995, I don’t know what happened or (what started to happen) from 1995 onwards, but the number of UFO sightings continued to trend upwards after that year till December

2020, one theory is that UFOs increased the number of earth visits after 1995, another theory is Alien and UFO movies and documentaries that might have an effect on people's judgment, according to Wikipedia there were 182 movies in the 80s and 90s in Hollywood alone (1.Wikipedia). To try to uncover the reason behind this sudden increase in UFO sightings, data from multiple sources should be collected, combined and analysed, sources like IMDB movies, Reuters news agency, NUFORC.

2. How I employed technologies / skills from this course

In this project I have employed technologies some of which I have learned throughout this course modules, starting with "**Cloud Computing**" module where I was introduced to google cloud platform and went through 3 qwiklabs to get familiar with it, this obviously influenced my decision to use GCP to deploy the web application and save project files. Then the "**Lifecycles and Pipelines**" module where we learned how to store data, transform it, and show it's output, I implemented a similar approach in this project where I saved my data in a storage bucket, then deployed my python application on a virtual machine where within the application code I was calling the dataset from the bucket and transform the data and show the output in the web application interface including transforming the data before loading it into the LSTM model. Another example of applying what I learned in this course into the project is applying what I learned from "**Ingest and Storage**" module where I ingested data from the CSV dataset located on the GCP storage bucket and used Big Query to create multiple Big Query tables, then I used google Colab to call those tables in order to explore the dataset.

In summary, these are the technologies that I used and how I used them:

7- Built a web application using python and Dash.
8- Created a GCP storage bucket to store web application file, dataset and LSTM.h5 model.
9- Created a VM instance, installed python and other libraries on it, configured the firewall, copied the application file from the storage bucket through ssh and finally, deployed the web application on it.
10- Created a Big Query dataset, and then connected it to the dataset in the storage bucket, then created 5 Big Query tables from that dataset.
11- Used google Colab to connect to the Big Query tables and explored dataset
12- Within the web application I am connecting to the storage bucket and getting data from dataset as well as loading the LSTM model.

Barriers or failures I encountered

Throughout this project I had no failures, but the only barrier I encountered was the fact that I haven't worked with Dash library before, I had to learn how to work with Dash to create a web application with Python, thankfully though the internet has many courses and tutorials on Dash and I learned the basics and fundamentals which was enough to get me through in this project.

Conclusion

In this project I explored UFO sightings data, then I created a web application to allow users to view UFO sightings dataset from different angles like the distribution of UFO sighting cases over countries, states and cities, as well as the distribution of UFO sightings over time in the form of time series, it also forecasts the future number of UFO sightings, I concluded that more data is needed in order to discover the reason behind USA having more UFO sightings than the rest of the world combined, and to explain the surge of UFO sightings world-wide from 1995 onwards, combining data from different sources like IMDB movies, Reuters news agency and NUFORC might bring us closer to the answers.

References:

1. **Wikipedia:** List of films featuring extraterrestrials (https://en.wikipedia.org/wiki/List_of_films_featuring_extraterrestrials)
2. **Denise Chow and Gadi Schwartz:** UFOs are about to make their way to the U.S. Senate. Here's what to know (https://www.nbcnews.com/science/science-news/ufos-are-make-way-us-senate-know-rcna973#:~:text=Last%20year%2C%20the%20Pentagon%20declassified%20three%20such%20videos%20captured%20by%20Navy%20pilots%2C%20intensifying%20speculation%20over%20the%20incidents%2C%20which%20have%20been%20confirmed%20by%20pilots%20who%20have%20observed%20them%20and%20even%20presidents%20who%20have%20been%20briefed%20on%20them.%C2%A0)
3. **Dirk Schulz-Makuch:** Why science suddenly has a lot to say about UFOs and UAP (https://bigthink.com/hard-science/ufo-uap-science/#:~:text=Add%20to%20this%20the%20giggle%20factor%20surrounding%20this%20topic.%20For%20scientists%2C%20the%20stigma%20connected%20with%20such%20research%20can%20have%20serious%20consequences%20for%20your%20career.%20The%20unfortunate%20result%20is%20that%20observations%20without%20easy%20explanations%20typically%20remain%20unexplained.)