SENTINEL & COPERNICUS DATA COLLECTION GUIDE

Sentinel Hub is a cloud-based platform and an Earth observation service developed by Sinergise, a company based in Slovenia. It provides access to a vast amount of satellite imagery and related data from various satellite missions, including the European Space Agency's (ESA) Copernicus program.

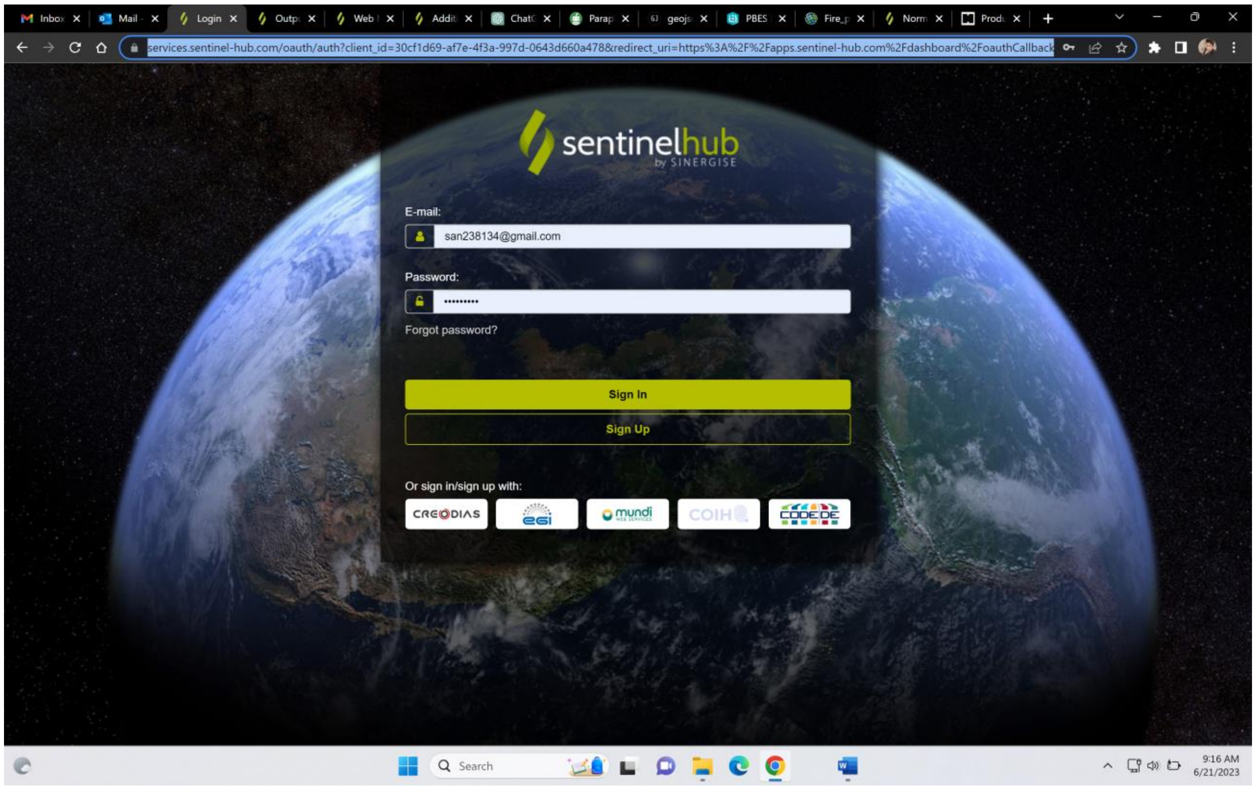
Sentinel Hub primarily focuses on data acquired by the Sentinel satellites, which are part of the Copernicus program. The Sentinel satellites are a constellation of Earth observation satellites designed to monitor the Earth's environment and gather data on various parameters such as land, oceans, atmosphere, and climate.

The Sentinel Hub platform enables users to access and analyze satellite imagery and data through a user-friendly interface or an Application Programming Interface (API). It offers a wide range of functionalities, including image processing, data visualization, and the integration of satellite data with other geospatial data sources.

By leveraging the capabilities of Sentinel Hub, users can monitor and analyze changes in land cover, track vegetation dynamics, assess environmental conditions, detect natural disasters, and conduct research in fields such as agriculture, forestry, urban planning, and environmental management.

Sentinel Hub has gained popularity among researchers, developers, and businesses due to its ease of use, scalability, and the rich collection of satellite data it offers. It has become an essential tool in Earth observation and remote sensing applications, empowering users to explore and understand our planet from space.

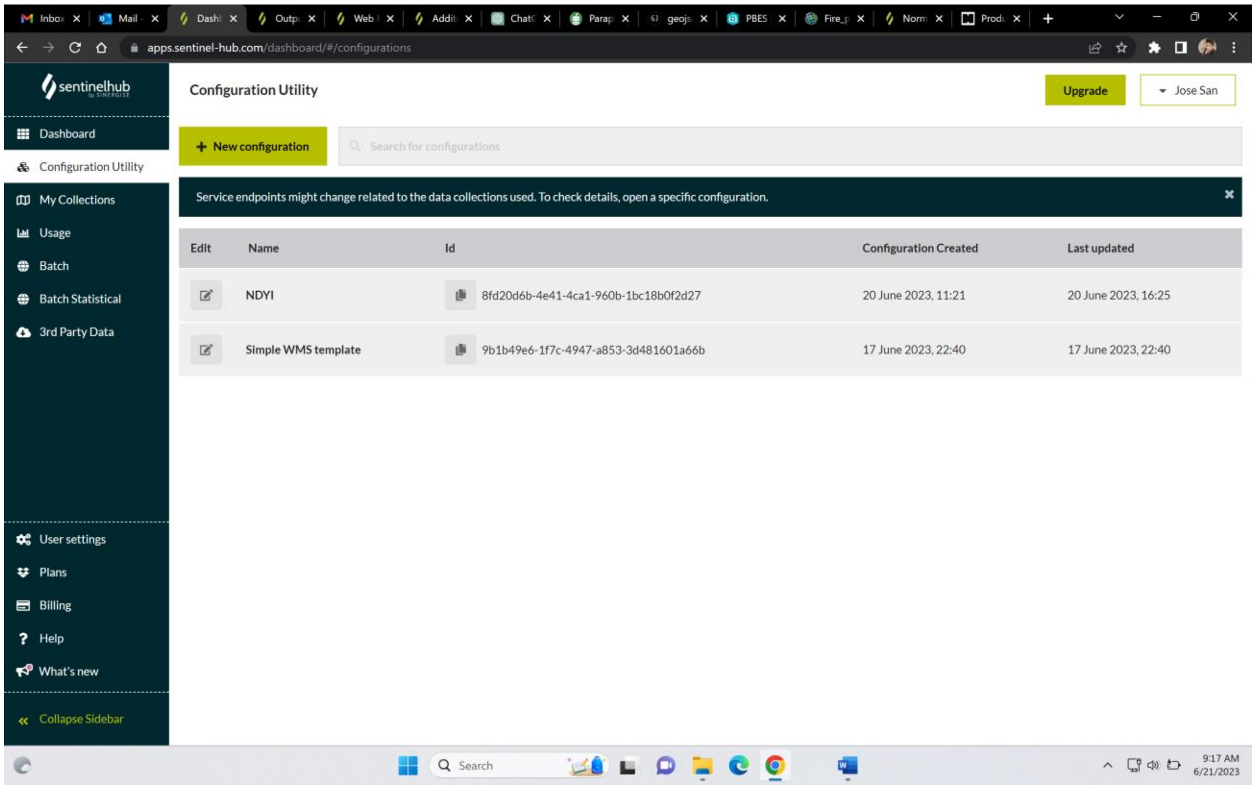
I. Configure Sentinel Hub:



https://services.sentinel-hub.com/oauth/auth?client_id=30cf1d69-af7e-4f3a-997d-0643d660a478&redirect_uri=https%3A%2F%2Fapps.sentinel-hub.com%2Fdashboard%2FoauthCallback.html&scope=&response_type=token&state=%252Fconfigurations%252F8fd20d6b-4e41-4ca1-960b-1bc18b0f2d27

Here is the website to signup for Sentinel Hub Account

After signing up:



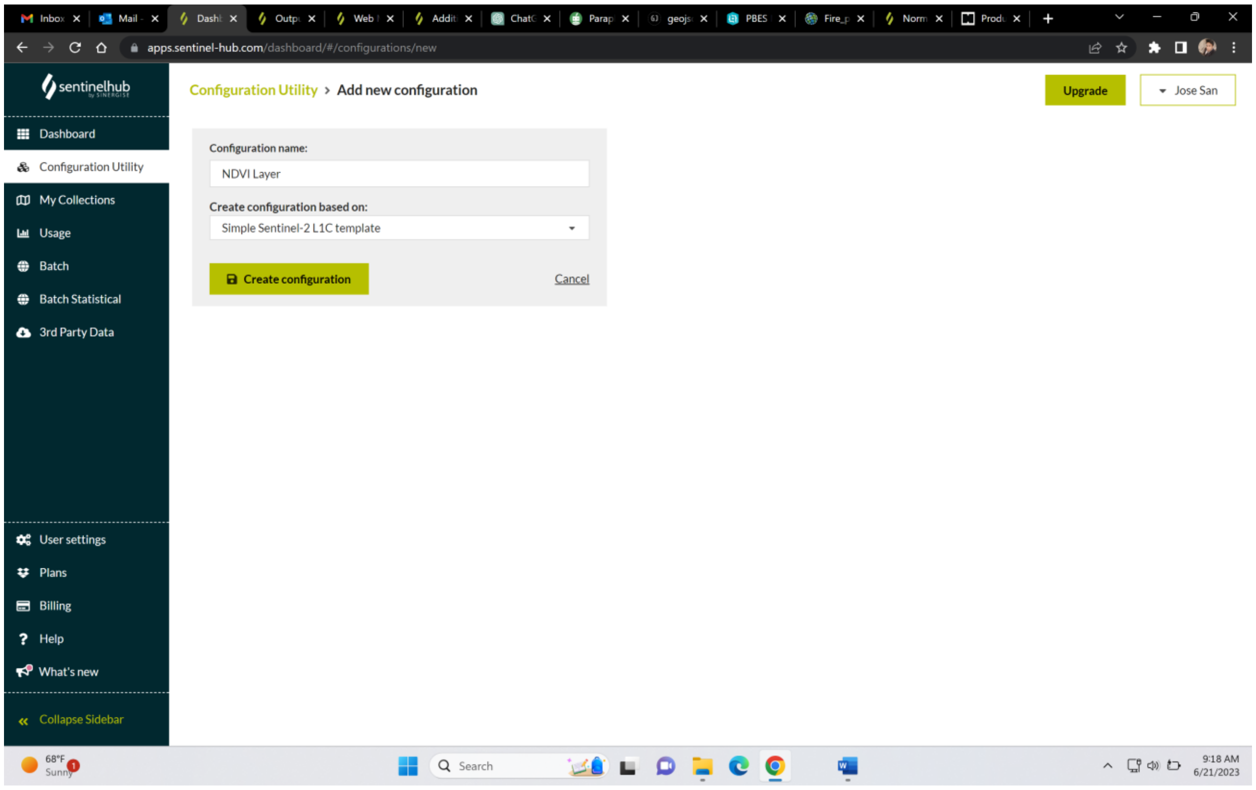
Configuration Utility

Service endpoints might change related to the data collections used. To check details, open a specific configuration.

Edit	Name	Id	Configuration Created	Last updated
	NDYI	8fd20d6b-4e41-4ca1-960b-1bc18b0f2d27	20 June 2023, 11:21	20 June 2023, 16:25
	Simple WMS template	9b1b49e6-1f7c-4947-a853-3d481601a66b	17 June 2023, 22:40	17 June 2023, 22:40

Go to the Configuration Utility tab below the dashboard button

And create a new configuration file



Configuration Utility > Add new configuration

Configuration name: NDVI Layer

Create configuration based on: Simple Sentinel-2 L1C template

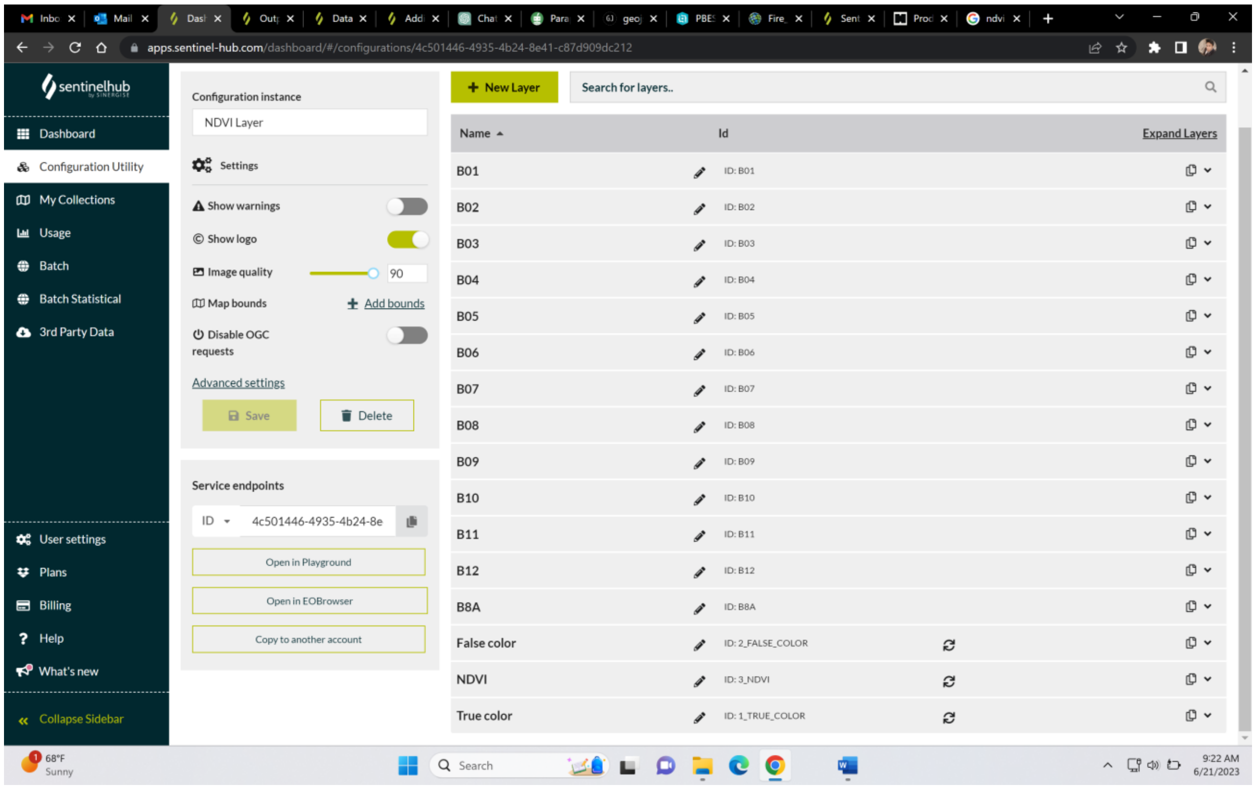
In my case I am selecting Sentinel-2 L1C template to access NDVI Layer which is Normalized Difference Vegetation Index for analyze soil patterns

There is a guide from which you can study what all components does each of these template have here is the link:

https://docs.sentinel-hub.com/api/latest/data/

This is a Sentinel Guide Book

Once you create your configuration file it should be like this :



apps.sentinel-hub.com/dashboard/#/configurations/4c501446-4935-4b24-8e41-c87d909dc212

Configuration instance: NDVI Layer

Settings
Show warnings: off
Show logo: on
Image quality: 90
Map bounds: Add bounds
Disable OGC requests: off

Service endpoints
ID 4c501446-4935-4b24-8e

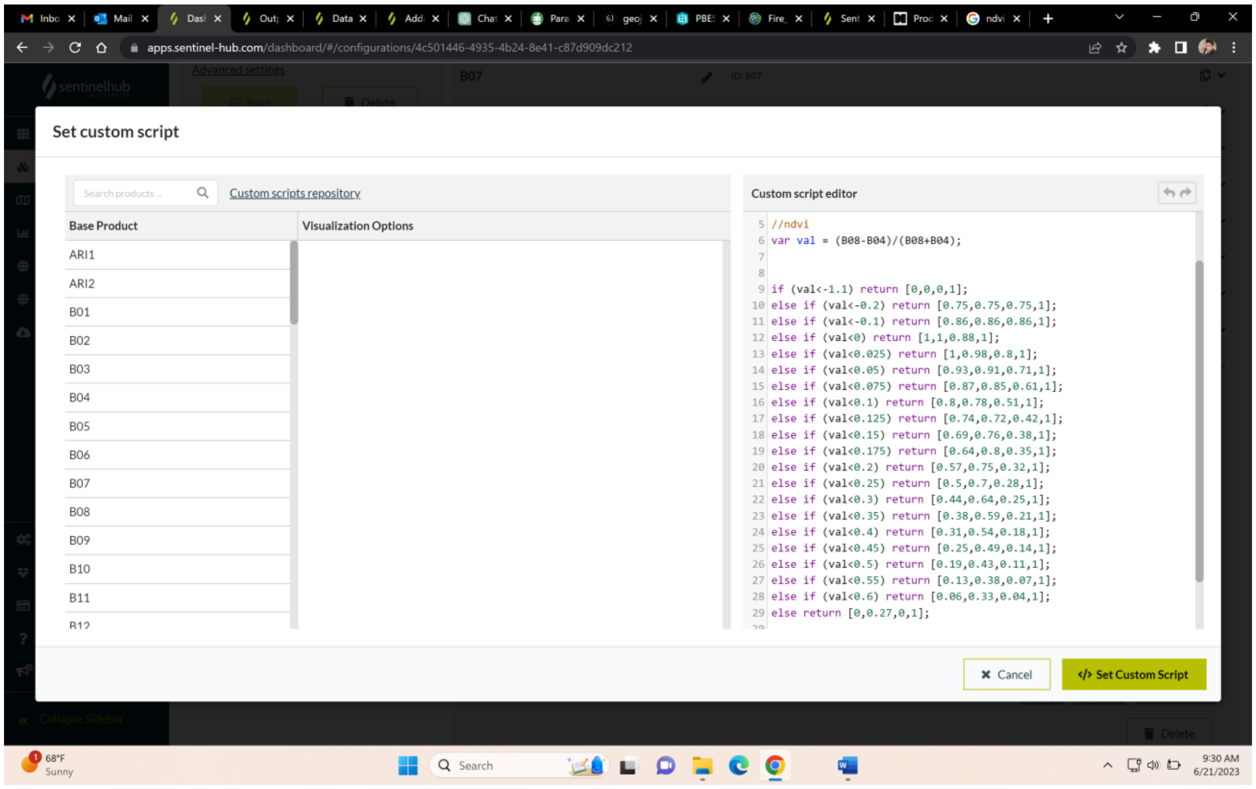
Name	Id
B01	ID: B01
B02	ID: B02
B03	ID: B03
B04	ID: B04
B05	ID: B05
B06	ID: B06
B07	ID: B07
B08	ID: B08
B09	ID: B09
B10	ID: B10
B11	ID: B11
B12	ID: B12
B8A	ID: B8A
False color	ID: 2_FALSE_COLOR
NDVI	ID: 3_NDVI
True color	ID: 1_TRUE_COLOR

You can remove the unnecessary layers and keep only NDVI on the right side it is simple to delete click on the icon on the right side which should open a panel for each of these layers and delete them individually

If you want to create your own layer with python script as NDVI layer is a combination of bands which are mentioned above sentinel hub uses formula like for instance NDVI = (NIR – RED) / (NIR + RED)

And the bands needed for the calculation of NDVI are as follows NDVI = (B08 – B04) / (B08 + B04) That's how sentinel hub made NDVI layer in their database

So we can do the same thing using python script here is an example script:



Set custom script

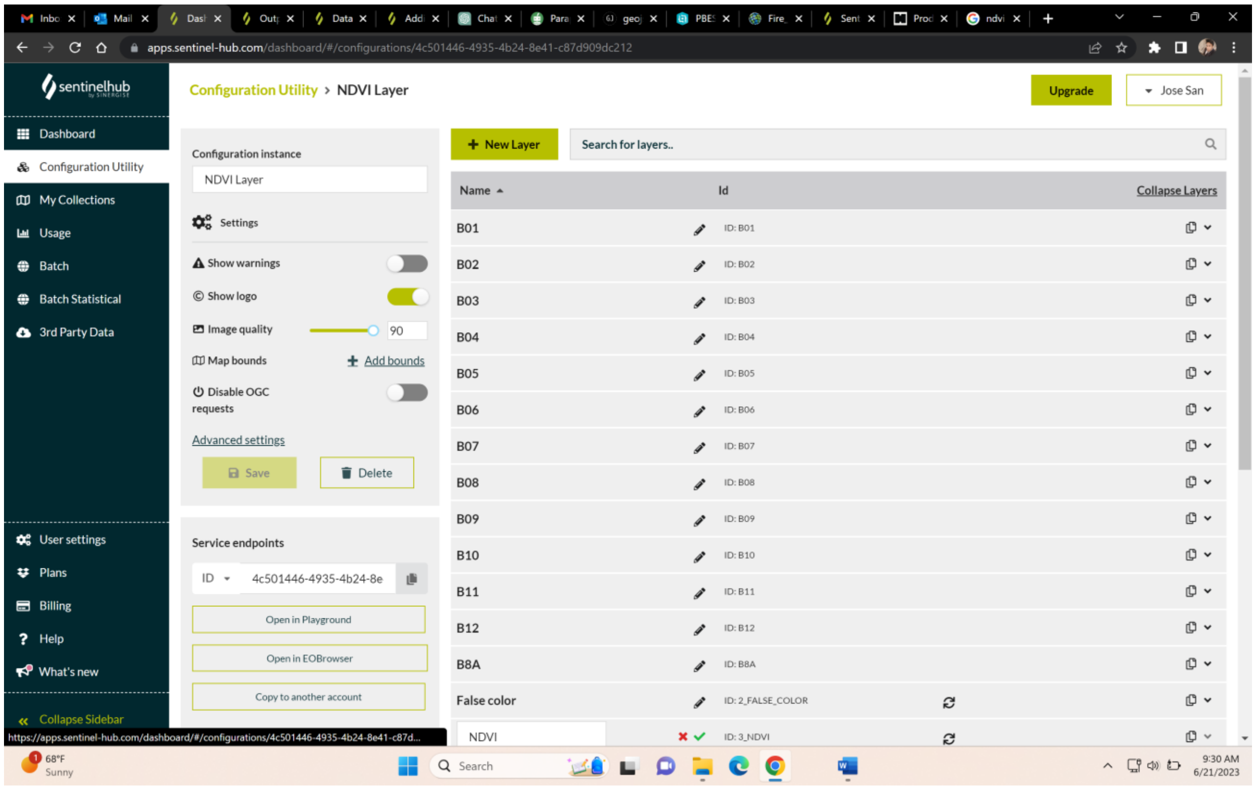
Custom scripts repository

Base Product	Visualization Options
ARI1	
ARI2	
B01	
B02	
B03	
B04	
B05	
B06	
B07	
B08	
B09	
B10	
B11	
B12	

Custom script editor

```
//ndvi
var val = (B08-B04)/(B08+B04);


if (val<-1.1) return [0,0,0,1];
else if (val<-0.2) return [0.75,0.75,0.75,1];
else if (val<-0.1) return [0.86,0.86,0.86,1];
else if (val<0) return [1,1,0.88,1];
else if (val<0.025) return [1,0.98,0.8,1];
else if (val<0.05) return [0.93,0.91,0.71,1];
else if (val<0.075) return [0.87,0.85,0.61,1];
else if (val<0.1) return [0.8,0.78,0.51,1];
else if (val<0.125) return [0.74,0.72,0.42,1];
else if (val<0.15) return [0.69,0.76,0.38,1];
else if (val<0.175) return [0.64,0.8,0.35,1];
else if (val<0.2) return [0.57,0.75,0.32,1];
else if (val<0.25) return [0.5,0.7,0.28,1];
else if (val<0.3) return [0.44,0.64,0.25,1];
else if (val<0.35) return [0.38,0.59,0.21,1];
else if (val<0.4) return [0.31,0.54,0.18,1];
else if (val<0.45) return [0.25,0.49,0.14,1];
else if (val<0.5) return [0.19,0.43,0.11,1];
else if (val<0.55) return [0.13,0.38,0.07,1];
else if (val<0.6) return [0.06,0.33,0.04,1];
else return [0,0.27,0,1];
```

Similarly when you click on +New Layer button on the left top corner of the page



apps.sentinel-hub.com/dashboard/#/configurations/4c501446-4935-4b24-8e41-c87d909dc212

Configuration Utility > NDVI Layer

Configuration instance: NDVI Layer

Settings

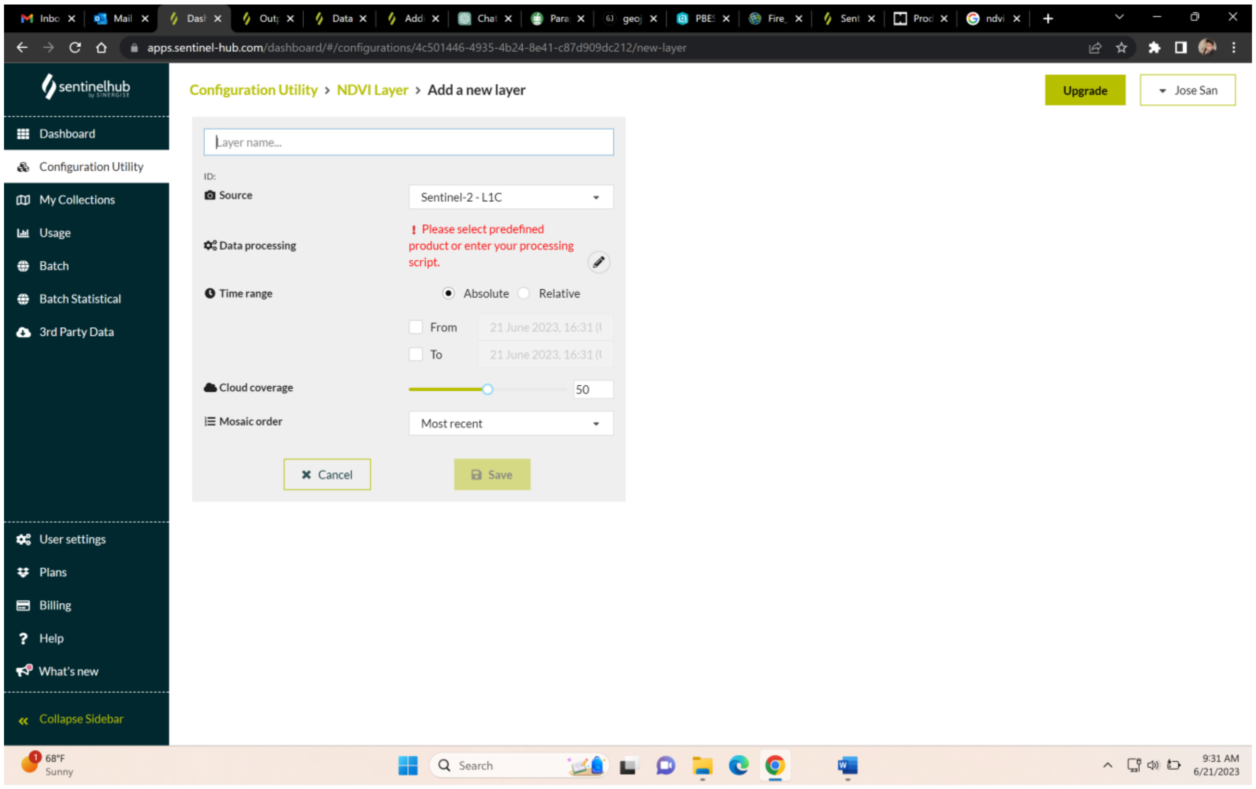
- Show warnings: off
- Show logo: on
- Image quality: 90
- Map bounds: Add bounds
- Disable OGC requests: off

Service endpoints

ID 4c501446-4935-4b24-8e

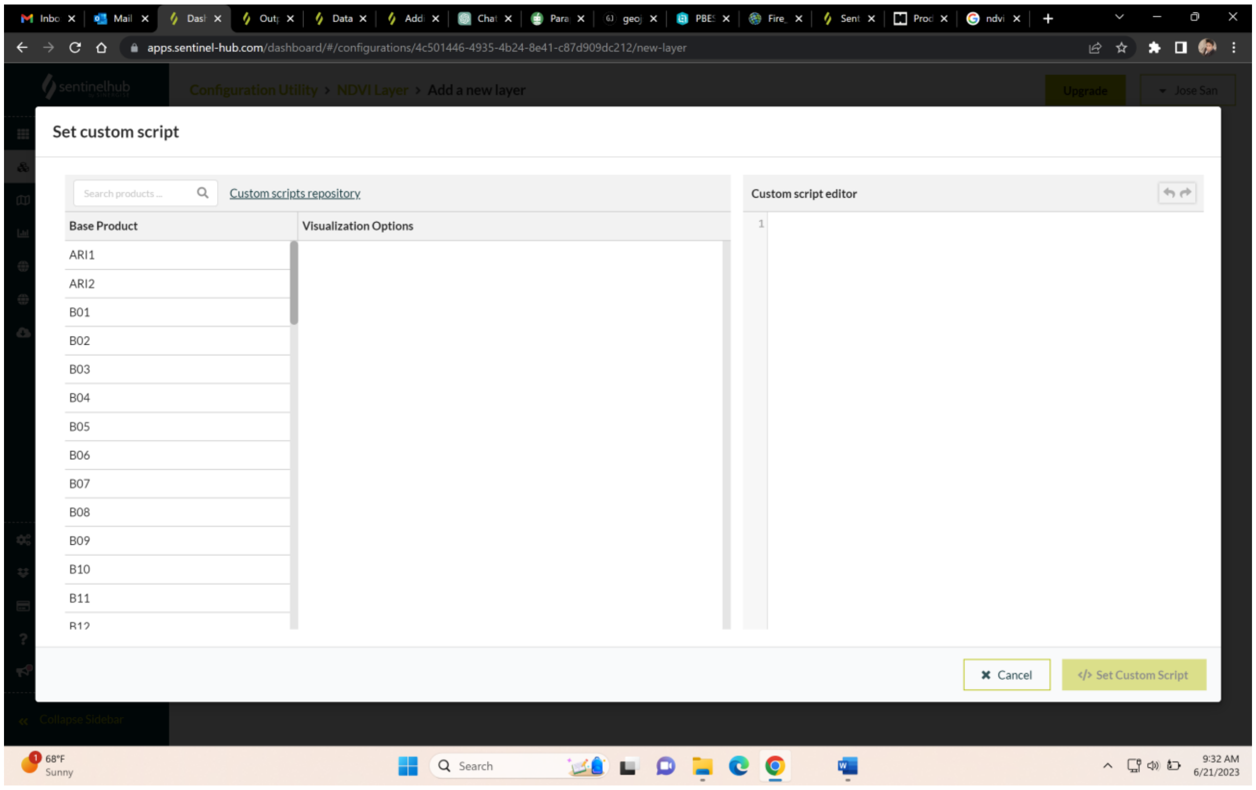
Name	Id
B01	ID: B01
B02	ID: B02
B03	ID: B03
B04	ID: B04
B05	ID: B05
B06	ID: B06
B07	ID: B07
B08	ID: B08
B09	ID: B09
B10	ID: B10
B11	ID: B11
B12	ID: B12
B8A	ID: B8A
False color	ID: 2_FALSE_COLOR
NDVI	ID: 3_NDVI

You get a configuration utility task bar



And then rename the layer you are configuring

Click on data processing pen icon



In the Custom script editor just copy and paste your code and set Custom script

Then next click on Save button

On the Configuring instance page save the layer

Here are some is a link to some custom scripts which you can try which is already written by Sentinel Hub coders:

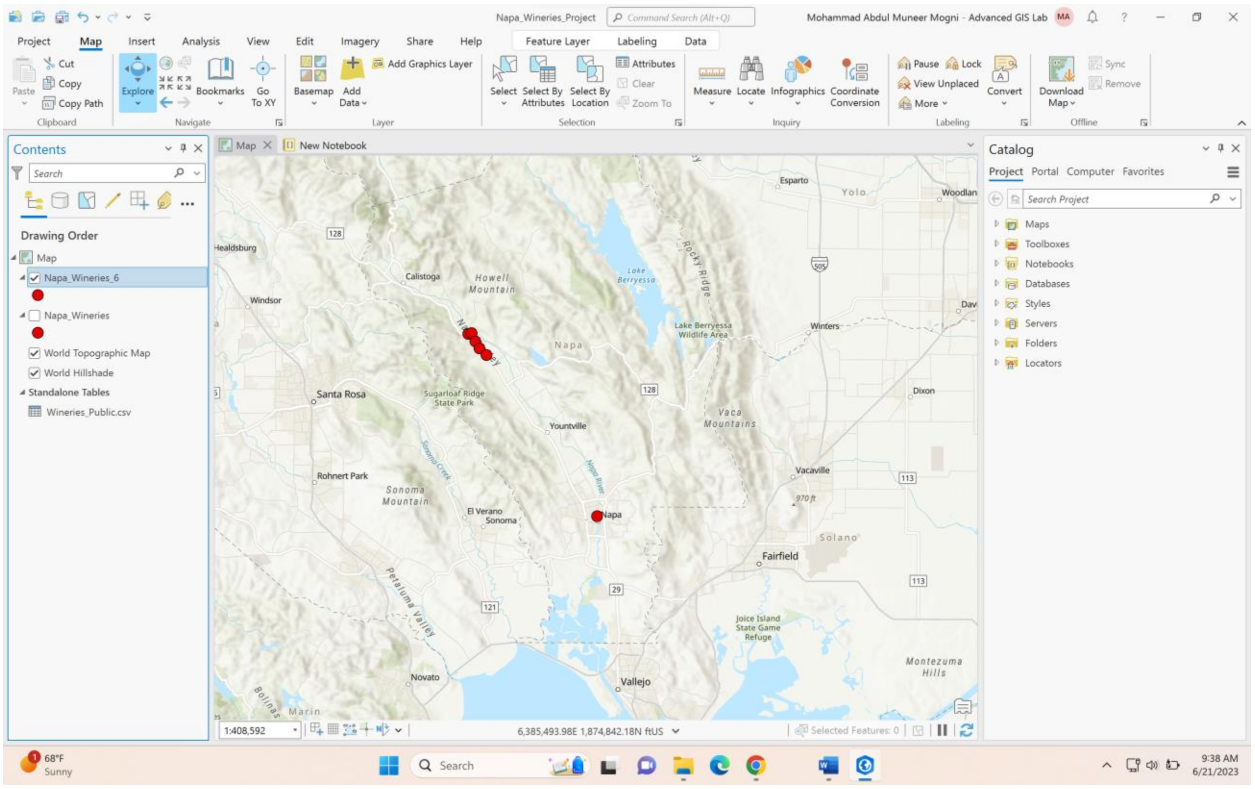
https://custom-scripts.sentinel-hub.com/custom-scripts/

After Finishing Sentinel Hub Configuration

Lets Jump on to ArcGIS

II. Importing GeoJSON or JSON file in ArcGIS from Sentinel Hub to create a shape file:

Here is a map of Napa Valley where first six wineries are marked off:



So we will try to import NDVI layer from sentinel hub to observe vegetation index on Napa Valley and analyze or observe some of the areas near these 6 wineries

So in order to import JSON file here is the link or instructions page to use Interface functions to download the file from Sentinel Hub Cloud:

Some of the Interface functions are available in Sentinel Hub Cloud are in the form of:

Web Mapping Service (WMS)

Web Coverage Service (WCS)

Web Feature Service (WFS)

Web Mapping Tile Service (WMTS)

Regarding these Interfaces there are few parameters which are mandatory to use with these Interfaces: Here is a list of parameters in this link or guide:

https://docs.sentinel-hub.com/api/latest/api/ogc/

And API Overview: (Syntax)

https://docs.sentinel-hub.com/api/latest/api/overview/

In order to request for a JSON file for our project here is a web service link used to create or import a JSON file:

```
https://services.sentinel-
hub.com/ogc/wms/<INSTANCE_ID>?SERVICE=WMS&REQUEST=GetMap&S
HOWLOGO=false&VERSION=1.3.0&LAYERS=NDVI&MAXCC=20&WIDTH
=640&HEIGHT=640&CRS=EPSG:4326&BBOX=46.697956,16.223885,46.6998
40,16.2276628&FORMAT=application/json
```

and the parameters used:

```json
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "properties": {
        "COLOR_HEX": "FFFFFF",
        "ID": 0
      },
      "geometry": {
        "type": "MultiPolygon",
        "crs": {
           "type": "name",
           "properties": {
              "name": "urn:ogc:def:crs:OGC::CRS84"
           }
        },
        "coordinates": [[[
           [16.225567302, 46.698948044],
           [16.225567302, 46.6989451],
           [16.225561399, 46.6989451],
           [16.225561399, 46.698942156],
           ...
        ]]]
      }
    },
    ...
  ]
}
```

Here is a python code which we created to import the JSON file:

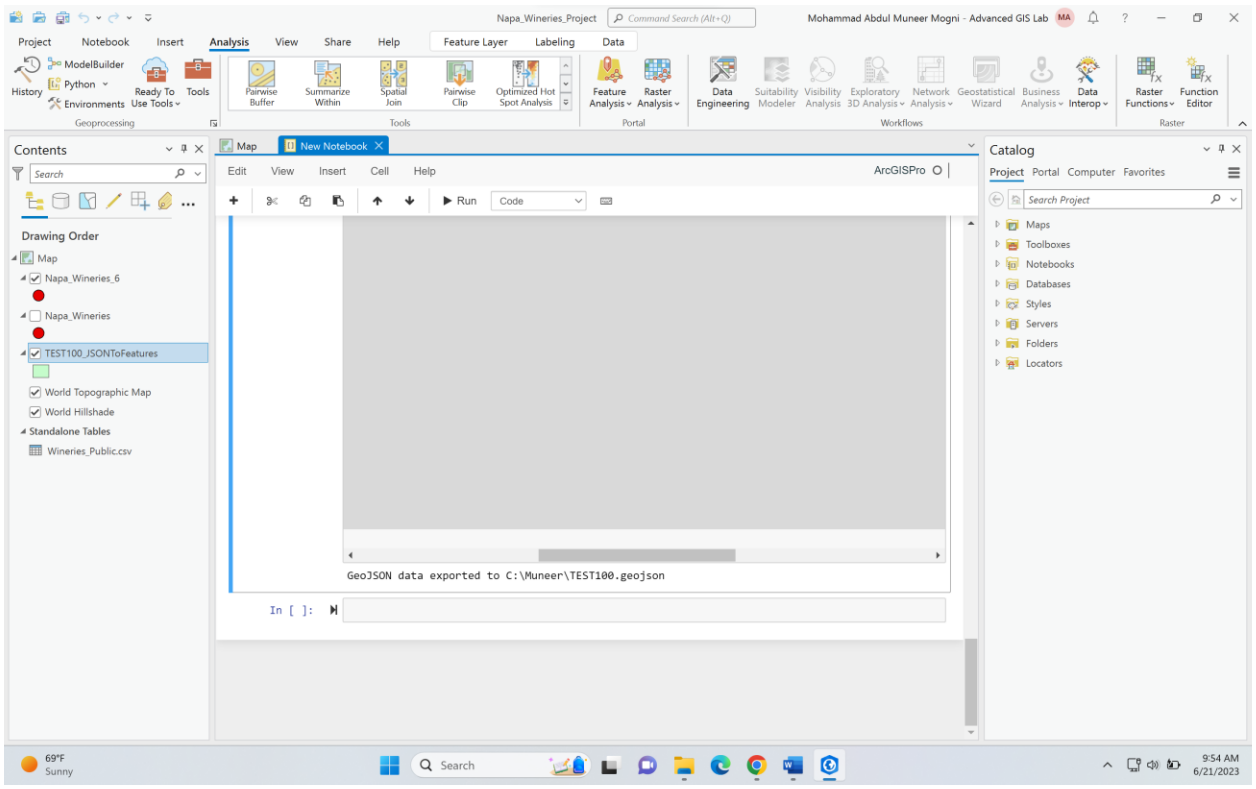
```
import requests

import json


# WMS URL

wms_url = "https://services.sentinel-hub.com/ogc/wms/9b1b49e6-1f7c-4947-a853-3d481601a66b?SERVICE=WMS&REQUEST=GetMap&SHOWLOGO=false&VERSION=1.3.0&LAYERS=NDVI&MAXCC=20&WIDTH=640&HEIGHT=640&CRS=EPSG:4326&BBOX=38.509621,-122.470664,38.524721,-122.488175&FORMAT=application/json"


# WMS parameters

params = {

  "type": "FeatureCollection",

  "features": [
  {
   "type": "Feature",
   "properties": {
     "COLOR_HEX": "FFFFFF",
     "ID": 0
   },
   "geometry": {
     "type": "MultiPolygon",
     "crs": {
        "type": "name",
        "properties": {
           "name": "urn:ogc:def:crs:OGC::CRS84"
        }
     },
     "coordinates": [[[
        [-122.488175, 38.517837],
        [-122.481356, 38.524721],
        [-122.477786, 38.509621],
        [-122.470664, 38.519511],
        ...
     ]]]
   }
  },
  ...
]
```

```
}

# Send WMS request
response = requests.get(wms_url, params=params)

# Check if the request was successful
if response.status_code == 200:
    # Load the GeoJSON data
    geojson_data = response.json()

    # Save GeoJSON data to a file
    output_file = r"C:\Muneer\TEST100.geojson"
    with open(output_file, 'w') as file:
        json.dump(geojson_data, file)

    print(f"GeoJSON data exported to {output_file}")

else:
    print("Error: Failed to retrieve data from the WMS server.")
```

SO GO TO ARCGIS PRO > ANALYSIS > OPEN A NEW NOTEBOOK COPY THE ABOVE CODE AND RUN IT

Note: Read the parameter requirements to request for **different dataset and at different locations**

Here is a link: https://docs.sentinel-hub.com/api/latest/api/ogc/output-formats/



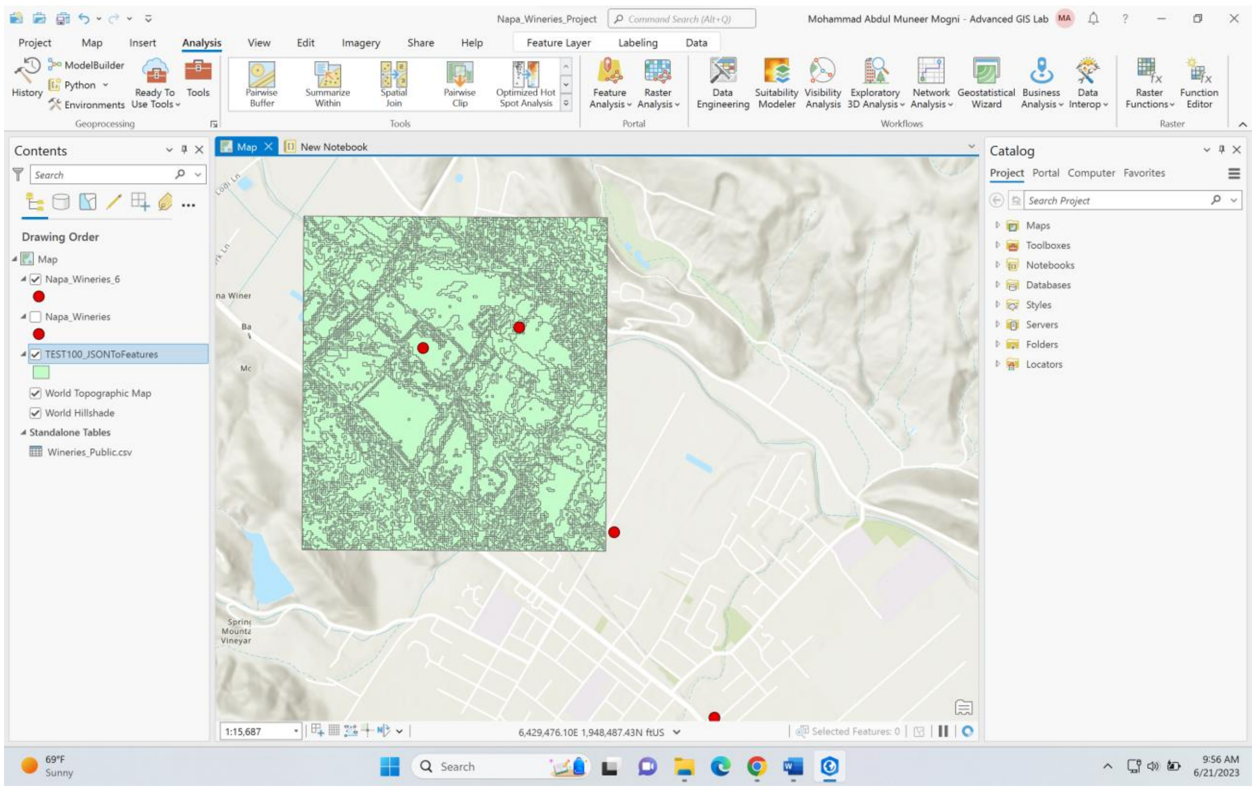
```
GeoJSON data exported to C:\Muneer\TEST100.geojson
```

JSON file exports to the location you have specified to create an output

Then go to VIEW > GEOPROCESSING TOOLS > TYPE GEOJSON to Feature tool

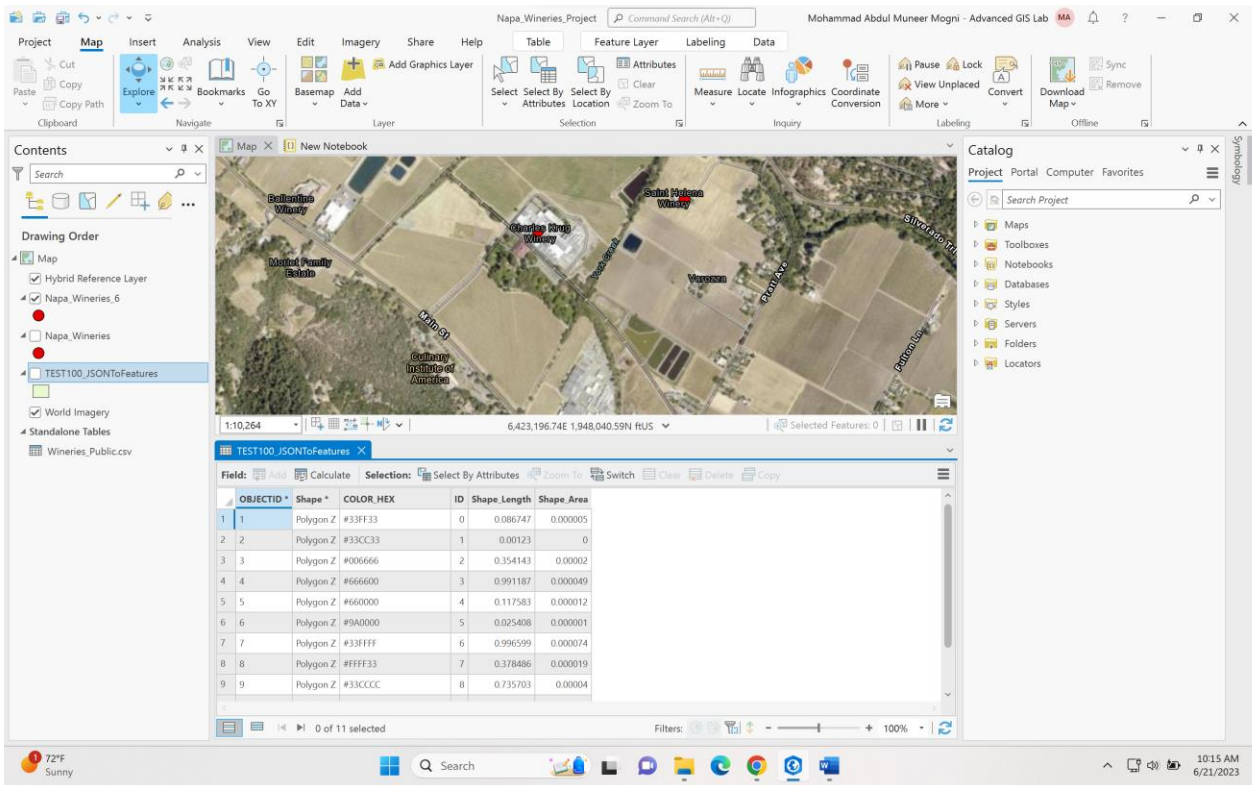
Select the exported JSON file, rename the output file and run it

Here is the Output layer:



It is cropped based on the parameters I have used in the code

Now to Analyze the NDVI Layer open the attribute table:



TEST100_JSONToFeatures

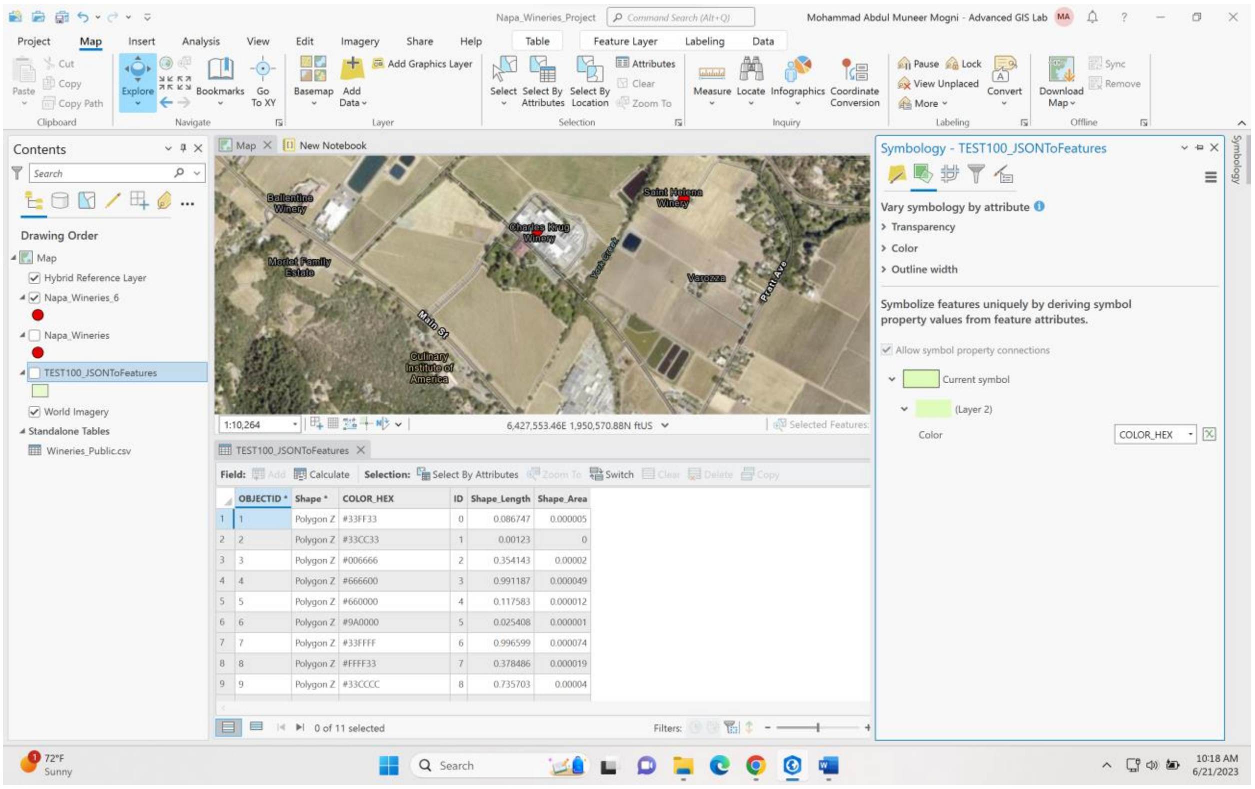
	OBJECTID *	Shape *	COLOR_HEX	ID	Shape_Length	Shape_Area
1	1	Polygon Z	#33FF33	0	0.086747	0.000005
2	2	Polygon Z	#33CC33	1	0.00123	0
3	3	Polygon Z	#006666	2	0.354143	0.00002
4	4	Polygon Z	#666600	3	0.991187	0.000049
5	5	Polygon Z	#660000	4	0.117583	0.000012
6	6	Polygon Z	#9A0000	5	0.025408	0.000001
7	7	Polygon Z	#33FFFF	6	0.996599	0.000074
8	8	Polygon Z	#FFFF33	7	0.378486	0.000019
9	9	Polygon Z	#33CCCC	8	0.735703	0.00004

0 of 11 selected

It gives us all the hexa decimal codes

So right click on the feature file and go to symbology

Select properties



TEST100_JSONToFeatures

Field: Add · Calculate · Selection: Select By Attributes · Zoom To · Switch · Clear · Delete · Copy

	OBJECTID *	Shape *	COLOR_HEX	ID	Shape_Length	Shape_Area
1	1	Polygon Z	#33FF33	0	0.086747	0.000005
2	2	Polygon Z	#33CC33	1	0.00123	0
3	3	Polygon Z	#006666	2	0.354143	0.00002
4	4	Polygon Z	#666600	3	0.991187	0.000049
5	5	Polygon Z	#660000	4	0.117583	0.000012
6	6	Polygon Z	#9A0000	5	0.025408	0.000001
7	7	Polygon Z	#33FFFF	6	0.996599	0.000074
8	8	Polygon Z	#FFFF33	7	0.378486	0.000019
9	9	Polygon Z	#33CCCC	8	0.735703	0.00004

0 of 11 selected

Symbology - TEST100_JSONToFeatures

Vary symbology by attribute

- Transparency
- Color
- Outline width

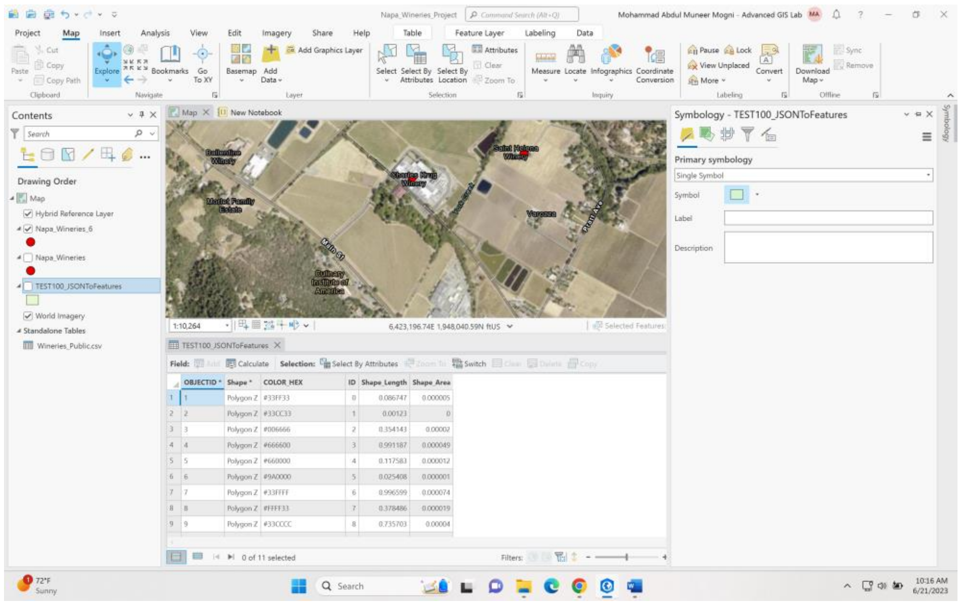
Symbolize features uniquely by deriving symbol property values from feature attributes.

[✓] Allow symbol property connections

Current symbol
(Layer 2)
Color: COLOR_HEX

Go to vary symbology by attribute pane and check the box allow symbol property connection

Then:



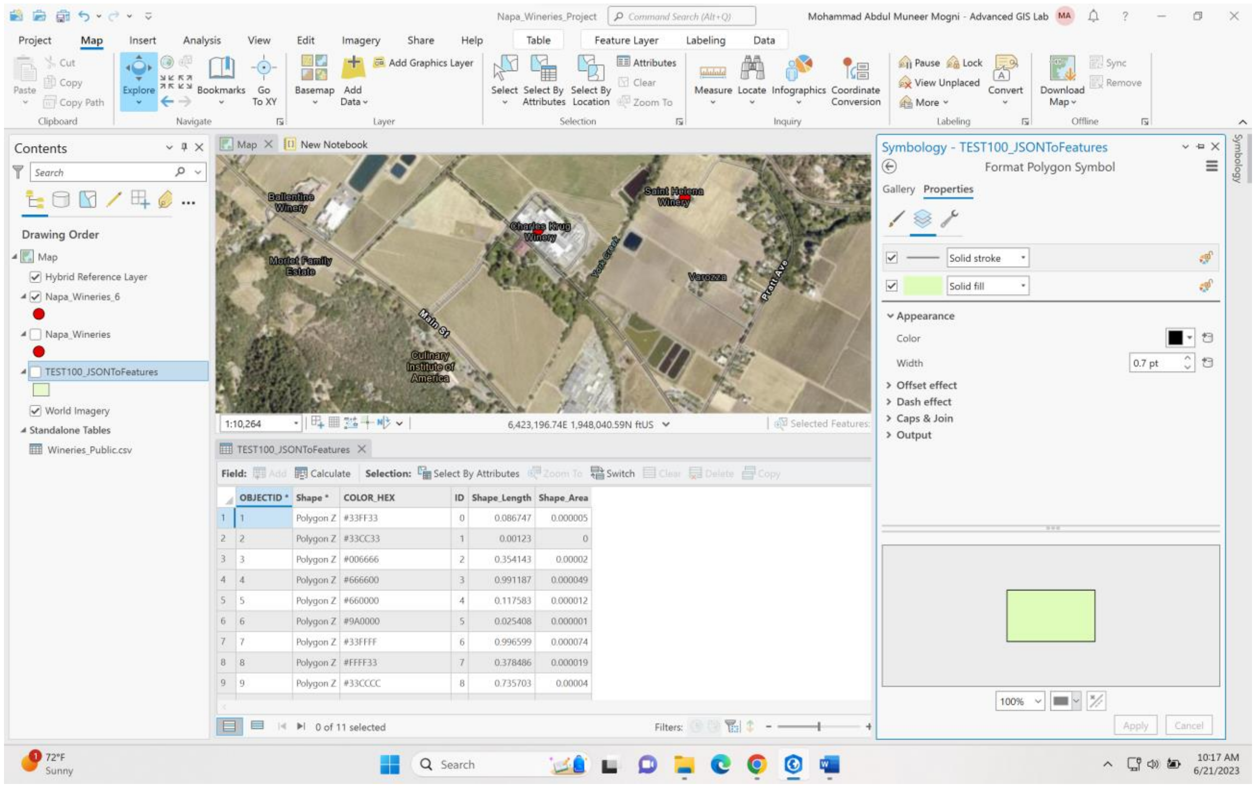
Symbology - TEST100_JSONToFeatures

Primary symbology: Single Symbol

OBJECTID *	Shape *	COLOR_HEX	ID	Shape_Length	Shape_Area
1	Polygon Z	#33FF33	0	0.086747	0.000005
2	Polygon Z	#33CC33	1	0.00123	0
3	Polygon Z	#006666	2	0.354143	0.00002
4	Polygon Z	#666600	3	0.991187	0.000049
5	Polygon Z	#660000	4	0.117583	0.000012
6	Polygon Z	#9A0000	5	0.025408	0.000001
7	Polygon Z	#33FFFF	6	0.996599	0.000074
8	Polygon Z	#FFFF33	7	0.378486	0.000019
9	Polygon Z	#33CCCC	8	0.735703	0.00004

0 of 11 selected

Click on the symbol color button which is highlighted in the image above



Symbology - TEST100_JSONToFeatures

Format Polygon Symbol

Solid stroke

Solid fill

Appearance

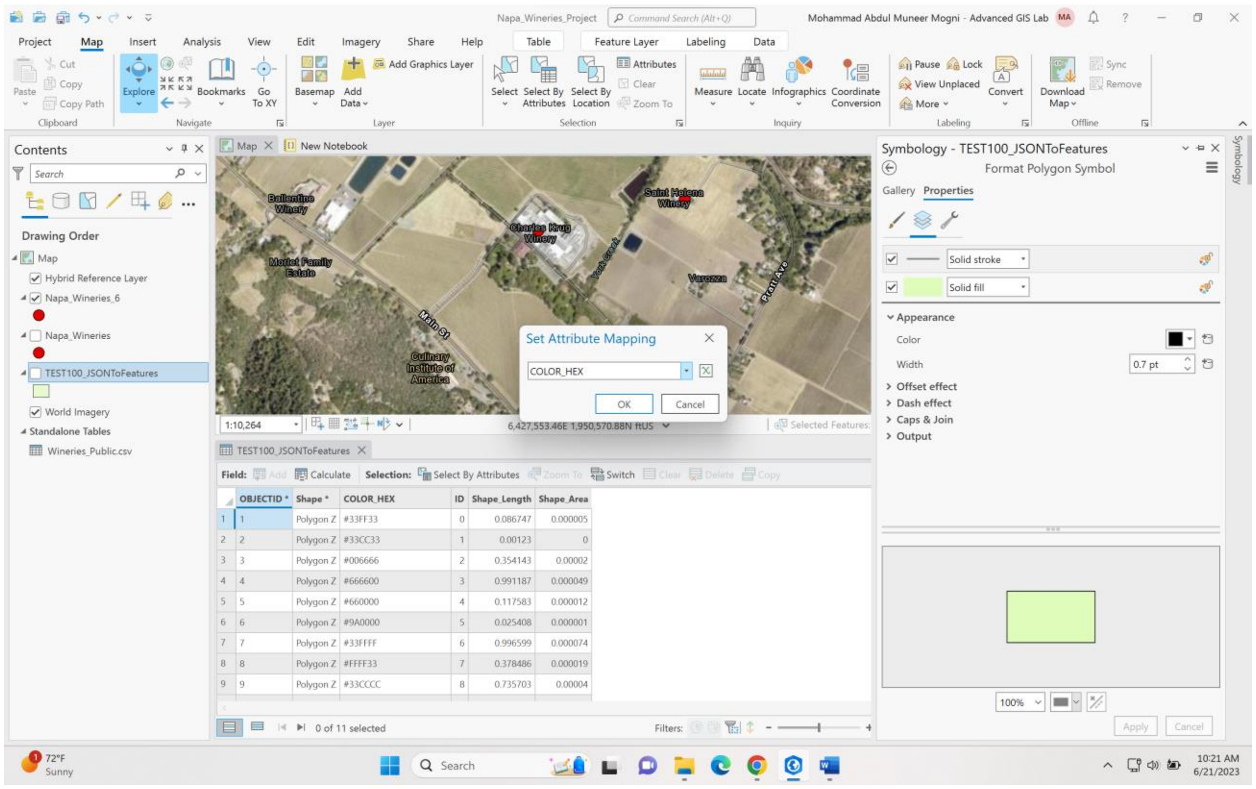
Color

Width 0.7 pt

OBJECTID *	Shape *	COLOR_HEX	ID	Shape_Length	Shape_Area
1	Polygon Z	#33FF33	0	0.086747	0.000005
2	Polygon Z	#33CC33	1	0.00123	0
3	Polygon Z	#006666	2	0.354143	0.00002
4	Polygon Z	#666600	3	0.991187	0.000049
5	Polygon Z	#660000	4	0.117583	0.000012
6	Polygon Z	#9A0000	5	0.025408	0.000001
7	Polygon Z	#33FFFF	6	0.996599	0.000074
8	Polygon Z	#FFFF33	7	0.378486	0.000019
9	Polygon Z	#33CCCC	8	0.735703	0.00004

0 of 11 selected

In the appearance section click on the database symbol for color



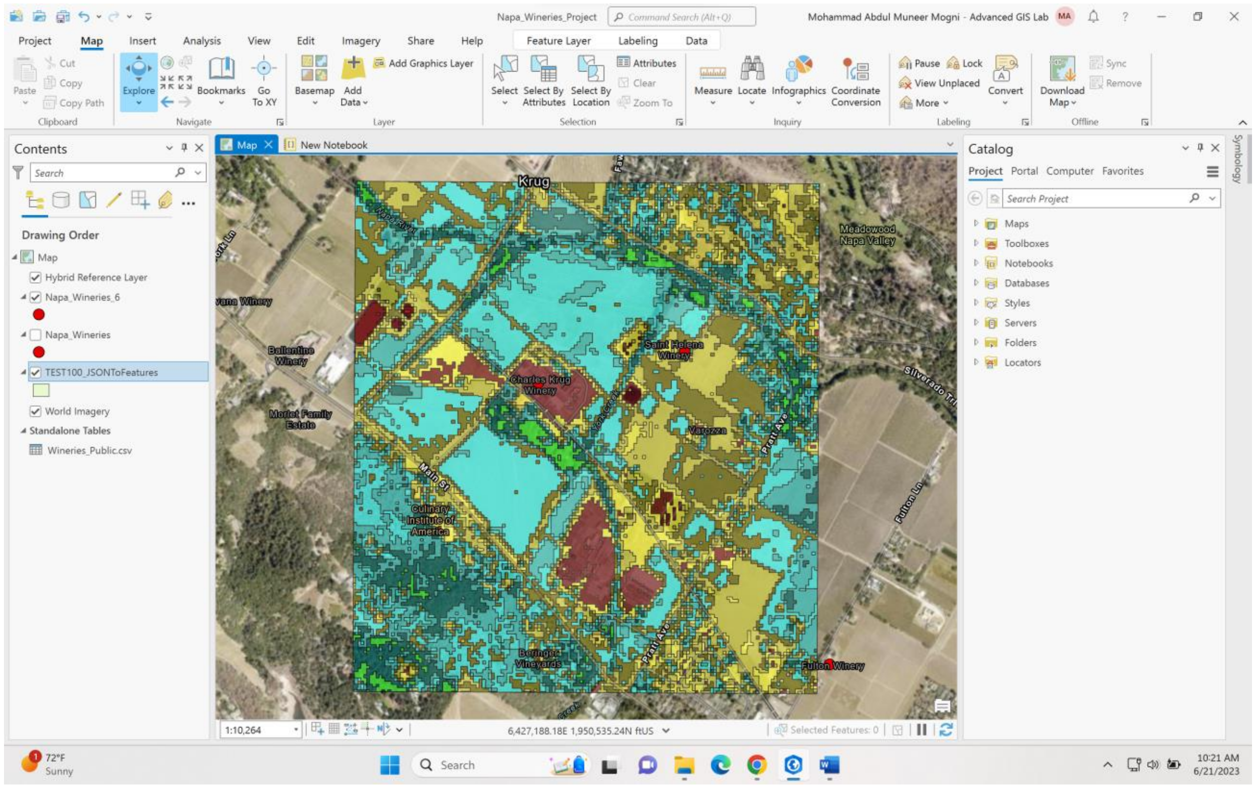
TEST100_JSONToFeatures

	OBJECTID *	Shape *	COLOR_HEX	ID	Shape_Length	Shape_Area
1	1	Polygon Z	#33FF33	0	0.086747	0.000005
2	2	Polygon Z	#33CC33	1	0.00123	0
3	3	Polygon Z	#006666	2	0.354143	0.00002
4	4	Polygon Z	#666600	3	0.991187	0.000049
5	5	Polygon Z	#660000	4	0.117583	0.000012
6	6	Polygon Z	#9A0000	5	0.025408	0.000001
7	7	Polygon Z	#33FFFF	6	0.996599	0.000074
8	8	Polygon Z	#FFFF33	7	0.378486	0.000019
9	9	Polygon Z	#33CCCC	8	0.735703	0.00004

0 of 11 selected

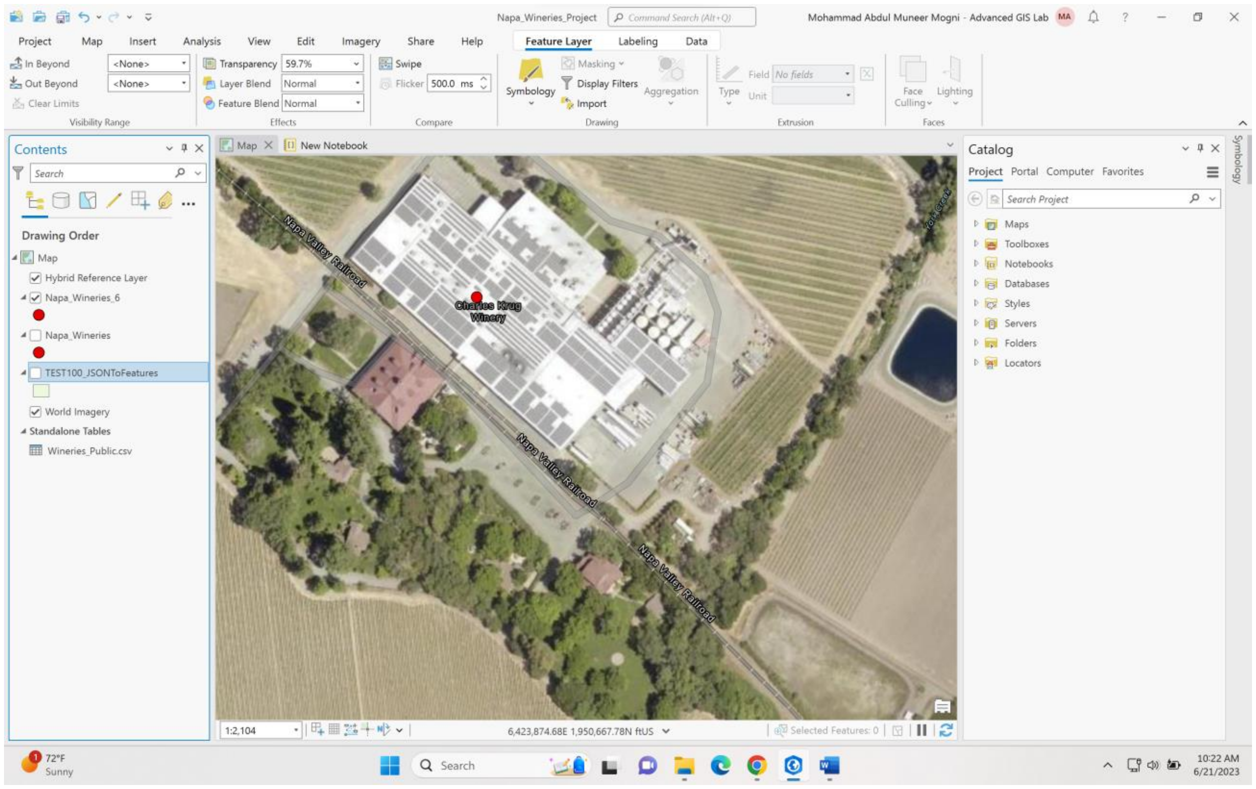
Select the attribute and click OK and the click on the apply button

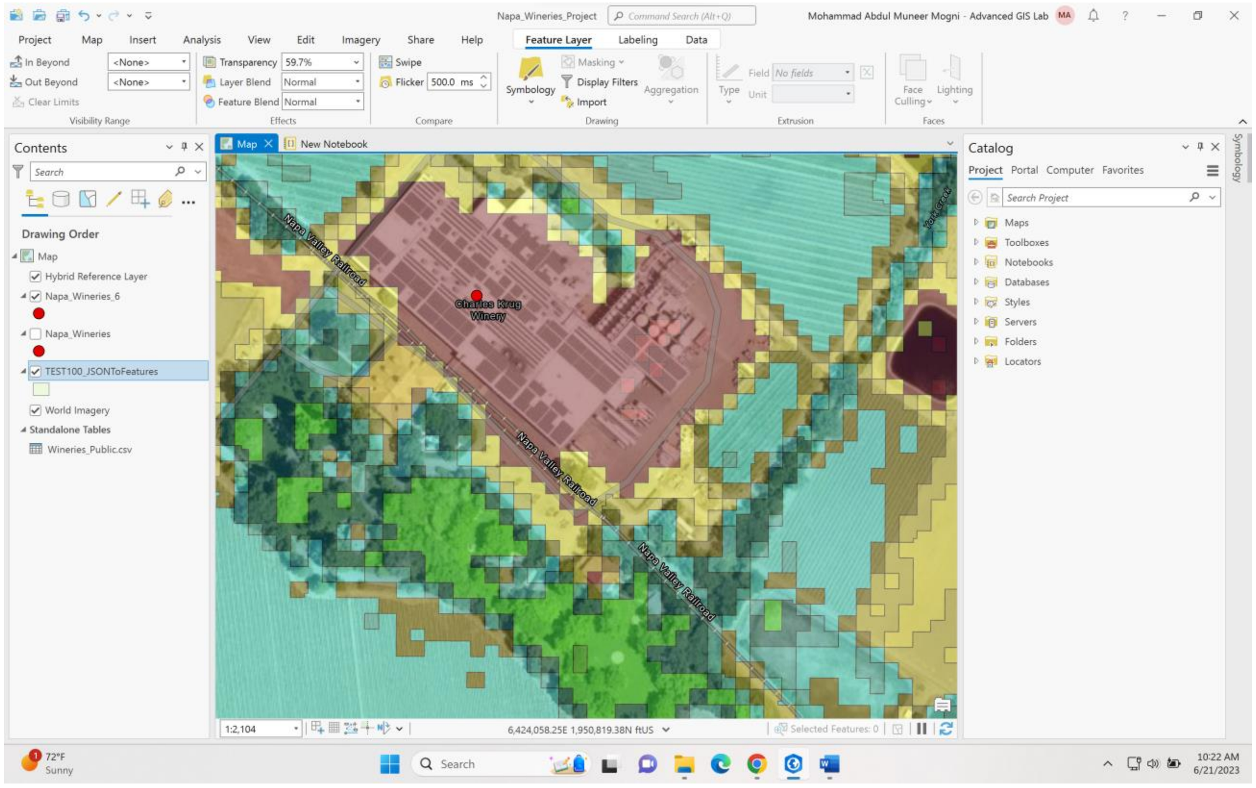
Here is the desired output:



For Better results change the base map and transparency

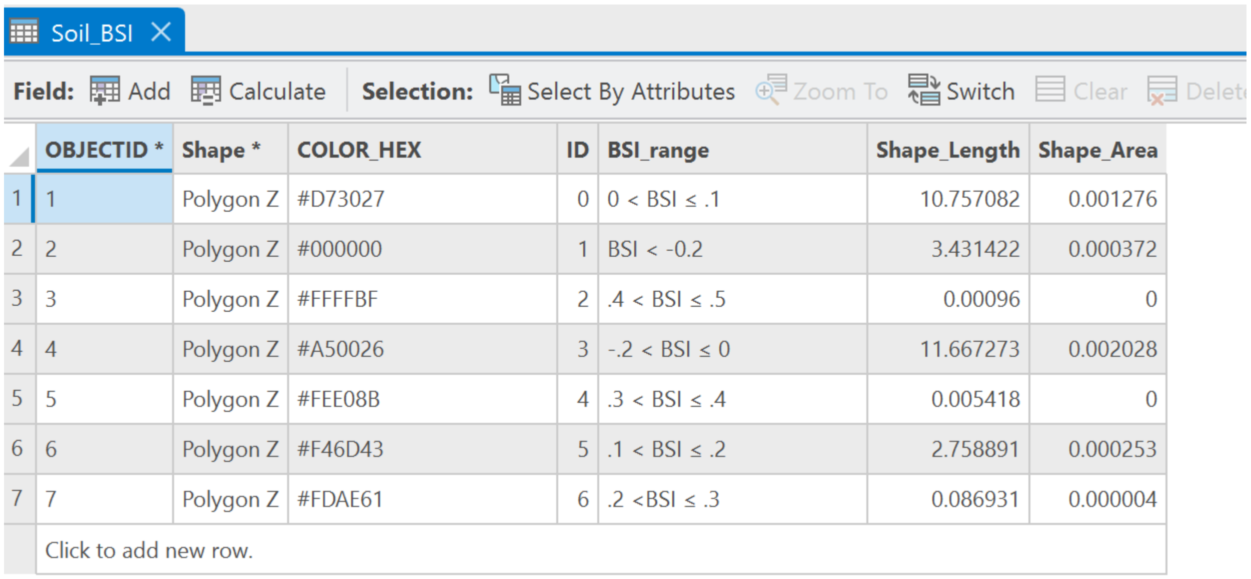
OUTPUT:





Code Snippets for SOIL BSI:

```
//VERSION=3

function evaluatePixel(sample) {
  let B11 = sample.B11;
  let B04 = sample.B04;
  let B08 = sample.B08;
  let B02 = sample.B02;

  let BSI = (B11 + B04 - B08 - B02) / (B11 + B04 + B08 + B02);

  if (BSI < -0.2)
    return [0, 0, 0]; // Black
  else if (BSI < 0)
    return [165 / 255, 0, 38 / 255]; // #a50026
  else if (BSI < 0.1)
    return [215 / 255, 48 / 255, 39 / 255]; // #d73027
  else if (BSI < 0.2)
    return [244 / 255, 109 / 255, 67 / 255]; // #f46d43
  else if (BSI < 0.3)
    return [253 / 255, 174 / 255, 97 / 255]; // #fdae61
  else if (BSI < 0.4)
    return [254 / 255, 224 / 255, 139 / 255]; // #fee08b
  else if (BSI < 0.5)
    return [255 / 255, 255 / 255, 191 / 255]; // #ffffbf
    else if (BSI < 0.6)
        return [217 / 255, 239 / 255, 139 / 255]; // #d9ef8b
    else if (BSI < 0.7)
        return [166 / 255, 217 / 255, 106 / 255]; // #a6d96a
    else if (BSI < 0.8)
        return [102 / 255, 189 / 255, 99 / 255]; // #66bd63
    else if (BSI < 0.9)
        return [26 / 255, 152 / 255, 80 / 255]; // #1a9850
    else
        return [0, 104 / 255, 55 / 255]; // #006837 (1.0)
}

function setup() {
    return {
        input: ["B02", "B04", "B08", "B11"],
        output: { bands: 3 }
    };
}
```
Output:



Soil_BSI

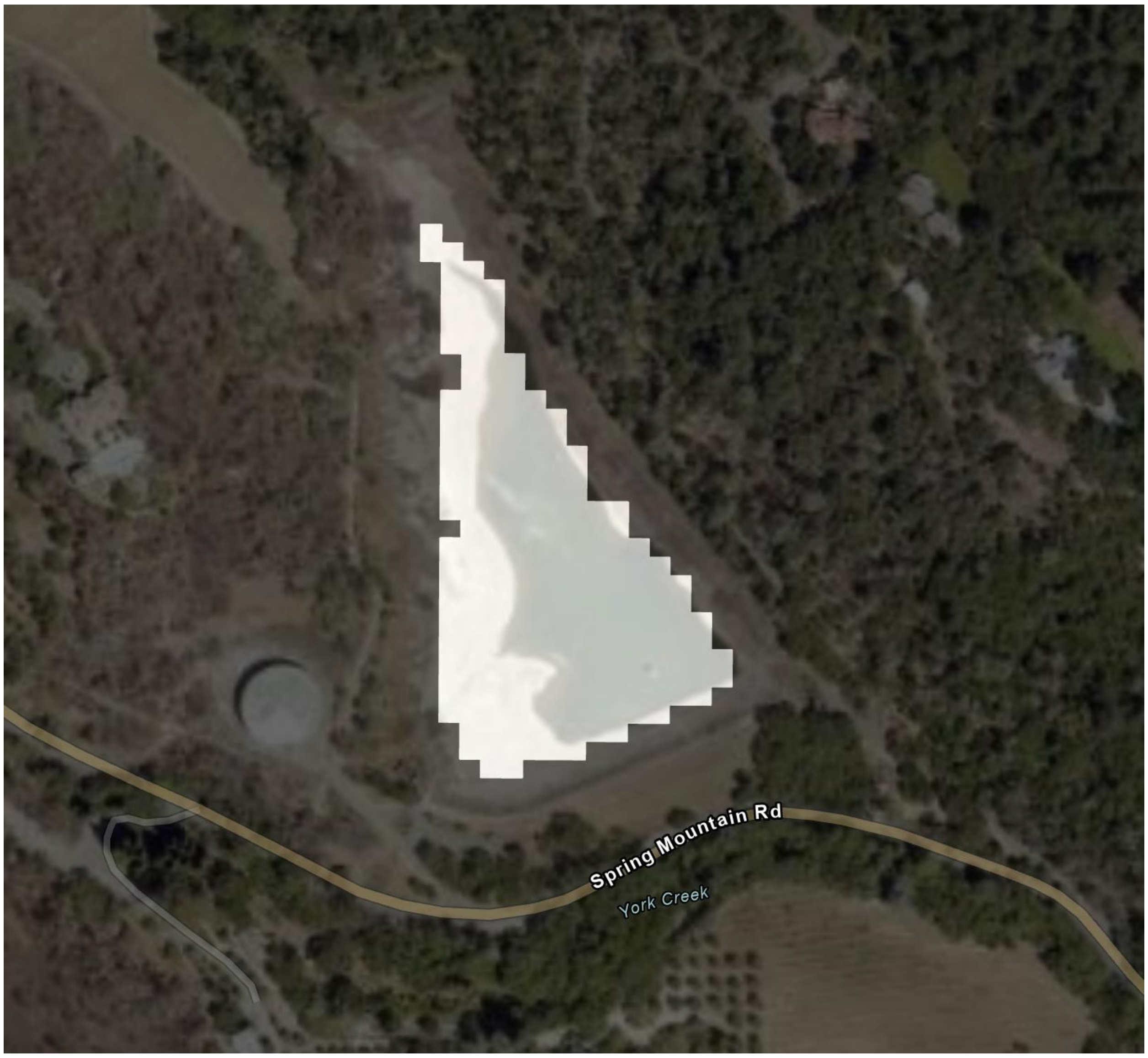
Field: Add Calculate Selection: Select By Attributes Zoom To Switch Clear Delet

	OBJECTID *	Shape *	COLOR_HEX	ID	BSI_range	Shape_Length	Shape_Area
1	1	Polygon Z	#D73027	0	0 < BSI ≤ .1	10.757082	0.001276
2	2	Polygon Z	#000000	1	BSI < -0.2	3.431422	0.000372
3	3	Polygon Z	#FFFFBF	2	.4 < BSI ≤ .5	0.00096	0
4	4	Polygon Z	#A50026	3	-.2 < BSI ≤ 0	11.667273	0.002028
5	5	Polygon Z	#FEE08B	4	.3 < BSI ≤ .4	0.005418	0
6	6	Polygon Z	#F46D43	5	.1 < BSI ≤ .2	2.758891	0.000253
7	7	Polygon Z	#FDAE61	6	.2 <BSI ≤ .3	0.086931	0.000004
	Click to add new row.						

Code Snippet for Water Bodies:

```
//VERSION=3
var source = "S2L2A";
var threshold = 0.1;

function setup() {
  return {
    input: ["B02", "B03", "B04", "B05", "B08", "B11"],
    output: { bands: 1 }
  };
}

function evaluatePixel(sample) {
  var b = sample.B02;
  var g = sample.B03;
  var r = sample.B04;
  var nr = sample.B08;
  var s1 = sample.B11;

  // Calculate the MNDWI index
  var mndwi = (g - nr) / (g + nr);

  // Determine if it's a water body based on the MNDWI threshold
  var water = mndwi > threshold ? 1 : 0;

  // Return the water body classification
  return [water];
}
```

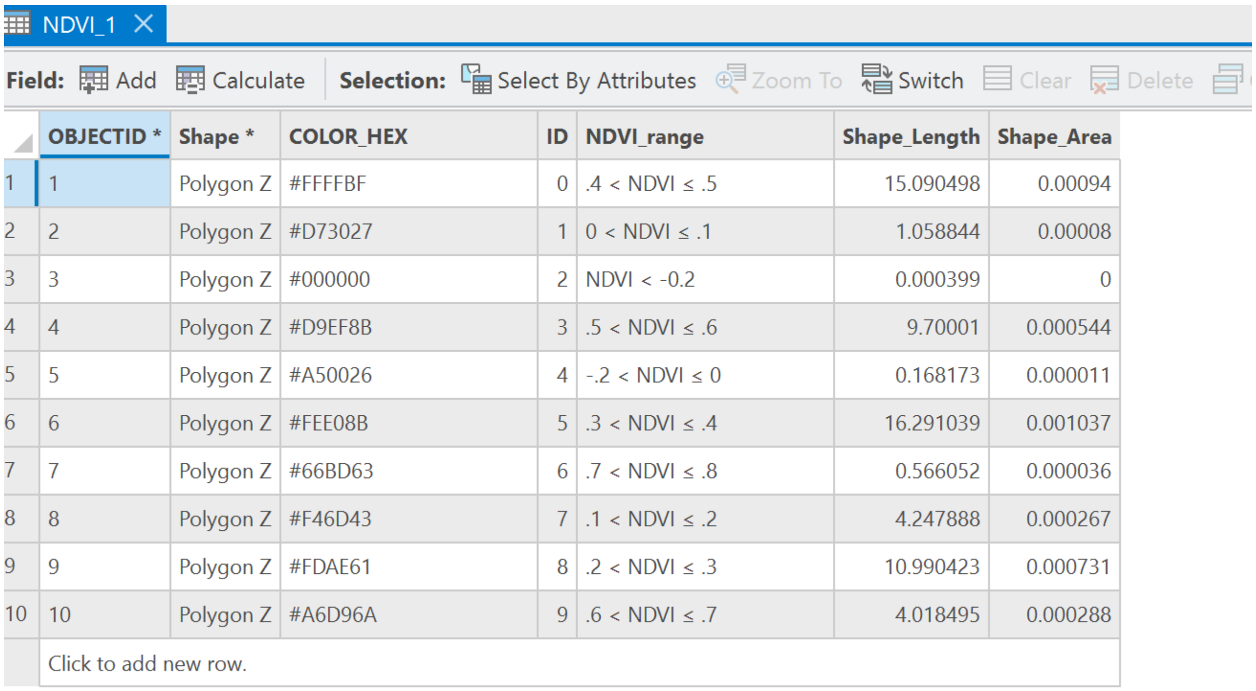
Output:



Code Snippet for NDVI:

```
//VERSION=3


let ndvi = (B08 - B04) / (B08 + B04);


  if (ndvi < -0.2)
    return [0, 0, 0]; // Black
  else if (ndvi < 0)
```

```
    return [165/255,0,38/255]; // #a50026
  else if (ndvi < 0.1)
    return [215/255,48/255,39/255]; // #d73027
  else if (ndvi < 0.2)
    return [244/255,109/255,67/255]; // #f46d43
  else if (ndvi < 0.3)
    return [253/255,174/255,97/255]; // #fdae61
  else if (ndvi < 0.4)
    return [254/255,224/255,139/255]; // #fee08b
  else if (ndvi < 0.5)
    return [255/255,255/255,191/255]; // #ffffbf
  else if (ndvi < 0.6)
    return [217/255,239/255,139/255]; // #d9ef8b
  else if (ndvi < 0.7)
    return [166/255,217/255,106/255]; // #a6d96a
  else if (ndvi < 0.8)
    return [102/255,189/255,99/255]; // #66bd63
  else if (ndvi < 0.9)
    return [26/255,152/255,80/255]; // #1a9850
  else return [0,104/255,55/255]; // #006837 (1.0)
```
Output:



	OBJECTID *	Shape *	COLOR_HEX	ID	NDVI_range	Shape_Length	Shape_Area
1	1	Polygon Z	#FFFFBF	0	.4 < NDVI ≤ .5	15.090498	0.00094
2	2	Polygon Z	#D73027	1	0 < NDVI ≤ .1	1.058844	0.00008
3	3	Polygon Z	#000000	2	NDVI < -0.2	0.000399	0
4	4	Polygon Z	#D9EF8B	3	.5 < NDVI ≤ .6	9.70001	0.000544
5	5	Polygon Z	#A50026	4	-.2 < NDVI ≤ 0	0.168173	0.000011
6	6	Polygon Z	#FEE08B	5	.3 < NDVI ≤ .4	16.291039	0.001037
7	7	Polygon Z	#66BD63	6	.7 < NDVI ≤ .8	0.566052	0.000036
8	8	Polygon Z	#F46D43	7	.1 < NDVI ≤ .2	4.247888	0.000267
9	9	Polygon Z	#FDAE61	8	.2 < NDVI ≤ .3	10.990423	0.000731
10	10	Polygon Z	#A6D96A	9	.6 < NDVI ≤ .7	4.018495	0.000288
	Click to add new row.						

Code Snippet for Temperature: From Copernicus

```python
import cdsapi

# Create a CDS API client
c = cdsapi.Client(url="https://cds.climate.copernicus.eu/api/v2",
key="210166:3c88417b-3015-4b7c-b2af-b68ed065f332")

# Define the request parameters
request = {
    'variable': [
        '2m_dewpoint_temperature', '2m_temperature', 'skin_temperature',
        'soil_temperature_level_1', 'soil_temperature_level_2',
'soil_temperature_level_3',
        'soil_temperature_level_4',
    ],
    'year': '2023',
    'month': '07',
    'day': '01',
    'format': 'netcdf',
    'time': [
        '00:00', '01:00', '02:00',
        '03:00', '04:00', '05:00',
        '06:00', '07:00', '08:00',
        '09:00', '10:00', '11:00',
        '12:00', '13:00', '14:00',
        '15:00', '16:00', '17:00',
        '18:00', '19:00', '20:00',
        '21:00', '22:00', '23:00',
    ],
}

# Send the request and download the data
c.retrieve('reanalysis-era5-land', request, 'C:/Face_Recognition/Temperature.nc')
```

pip install cdsapi

Then log in to Copernicus official website and copy both url and api_key

Use this snippet to download any data