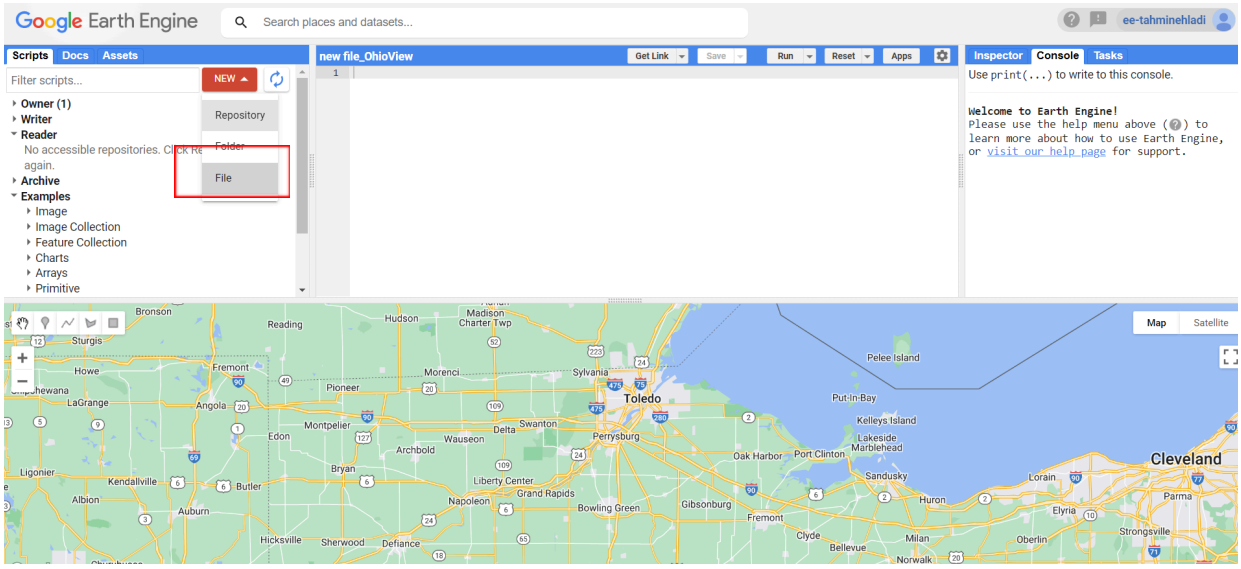
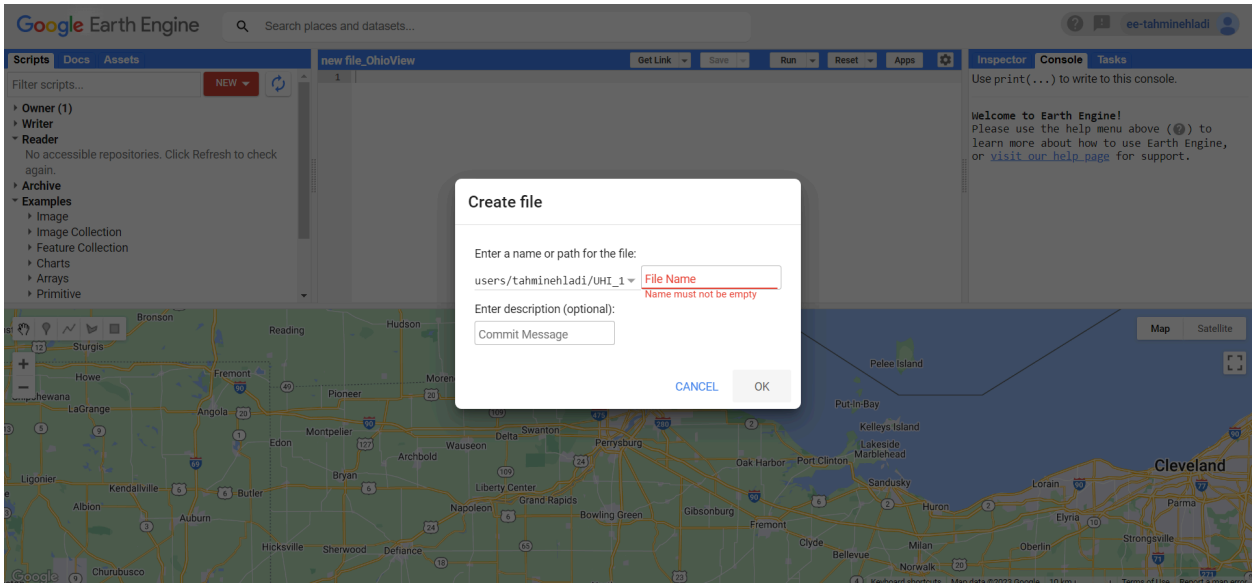
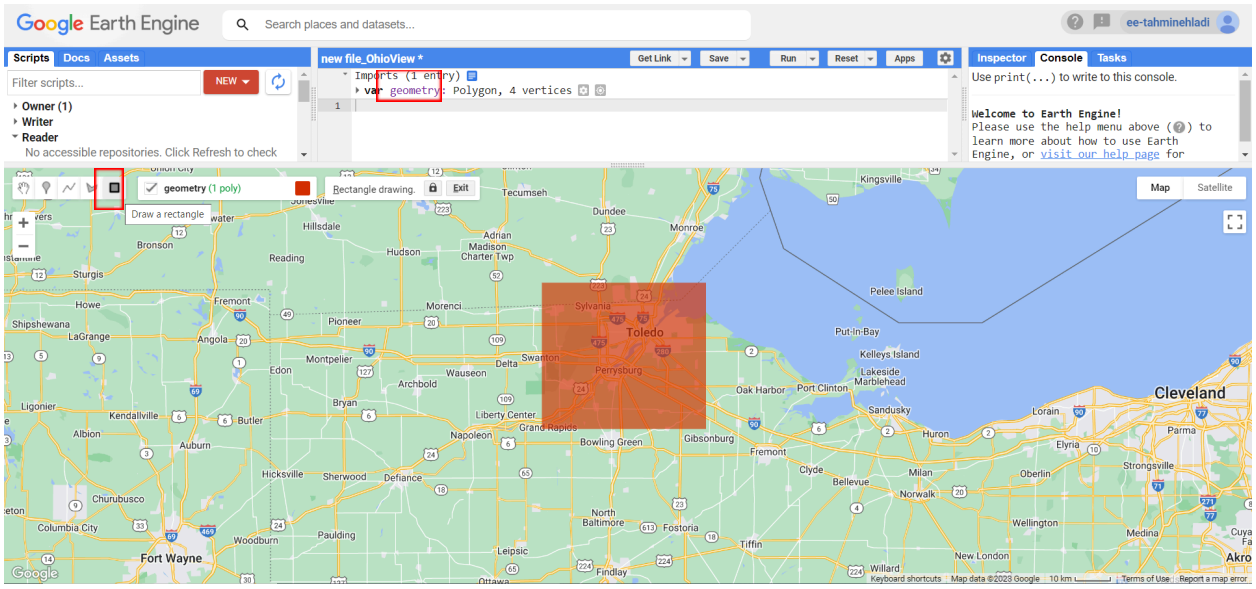
Coding in Google Earth Engine to measure NDVI and LST

- At first step, open Google Earth Engine, and create a new file through NEW, File, and define a name for the new file.

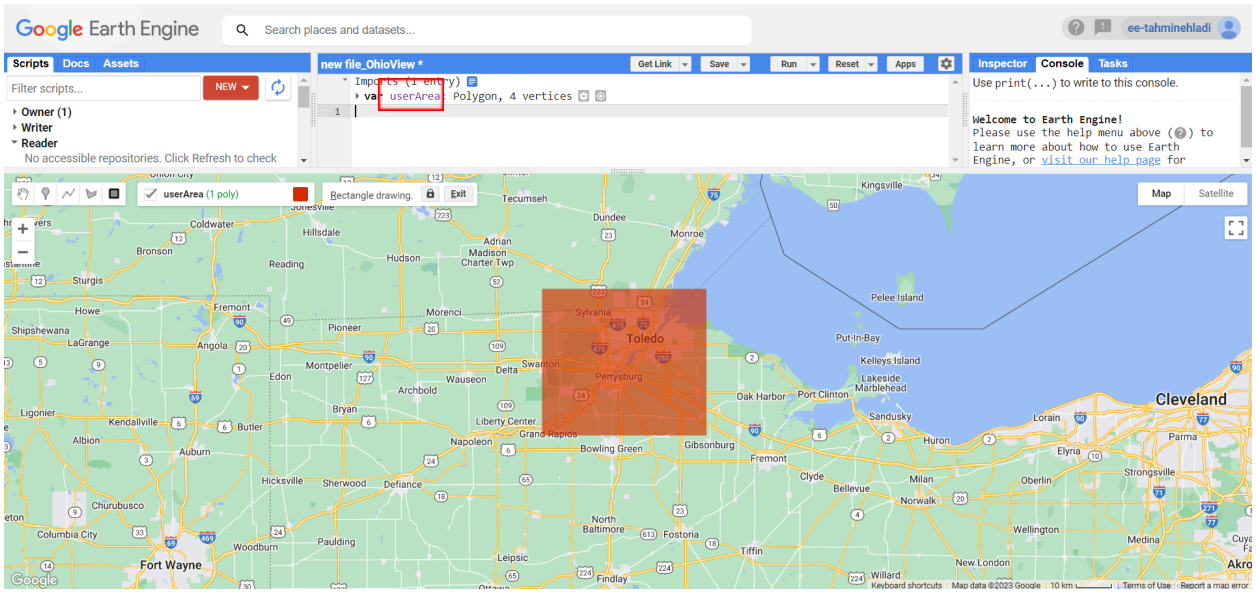




- Then, by using the drawing tools on the map, and selecting “Draw a rectangle”, draw an area of interest.



- Next, change the “geometry” into “userArea”.



- Then, we create two variables as startDate and endDate. In this example, 2022-08-28 and 2022-11-16 are used as startDate and endDate respectively. These dates can be changed based on your research purpose.

```
var startDate= '2022-08-28';
var endDate= '2022-11-16';
```

- We import the Landsat 9 data, and filter it based on the defined area of interest filterBounds() and the defined date range filterDate().

```
var LC9 = ee.ImageCollection('LANDSAT/LC09/C02/T1_L2')
  .filterBounds(userArea)
  .filterDate(startDate, endDate);
```

- Before using the data, the scale factor must be applied to the Landsat surface reflectance and surface temperature bands.

Surface Reflectance has a scale factor of 0.0000275 and an additional offset of -0.2 per pixel.

Surface Temperature has a scale factor of 0.00341802 and an additional offset of 149.0 per pixel, running these calculations give surface temperature in Kelvin, to convert it into Celsius, it is also subtracted from 273.15.

```
function applyScaleFactors(image) {
                                    var opticalBands =
image.select(['SR_B4','SR_B3','SR_B2','SR_B5']).multiply(0.0000275).add(-0.2);
 var thermalBands = image.select('ST_B10').multiply(0.00341802).add(149.0).subtract(273.15);
 return image.addBands(opticalBands, null, true)
        .addBands(thermalBands, null, true);
}
var LC9True=LC9.map(applyScaleFactors);
```

- Then, we sort the data based on the cloud cover property. By running this code, the least cloudy image will be on top.

```
var LC9sort = LC9True.sort('CLOUD_COVER', false);
print (LC9sort);
```

- Next, we create a mosaic by using mosaic(). This method composites overlapping images according to their order in the collection (last on top).

```
var LC9mosaic = ee.Image(LC9sort.mosaic());
print (LC9mosaic);
```

- Then, use clip() to trim the data based on the defined area of interest.

```
var LC9mosaicClip = LC9mosaic.clip(userArea);
```

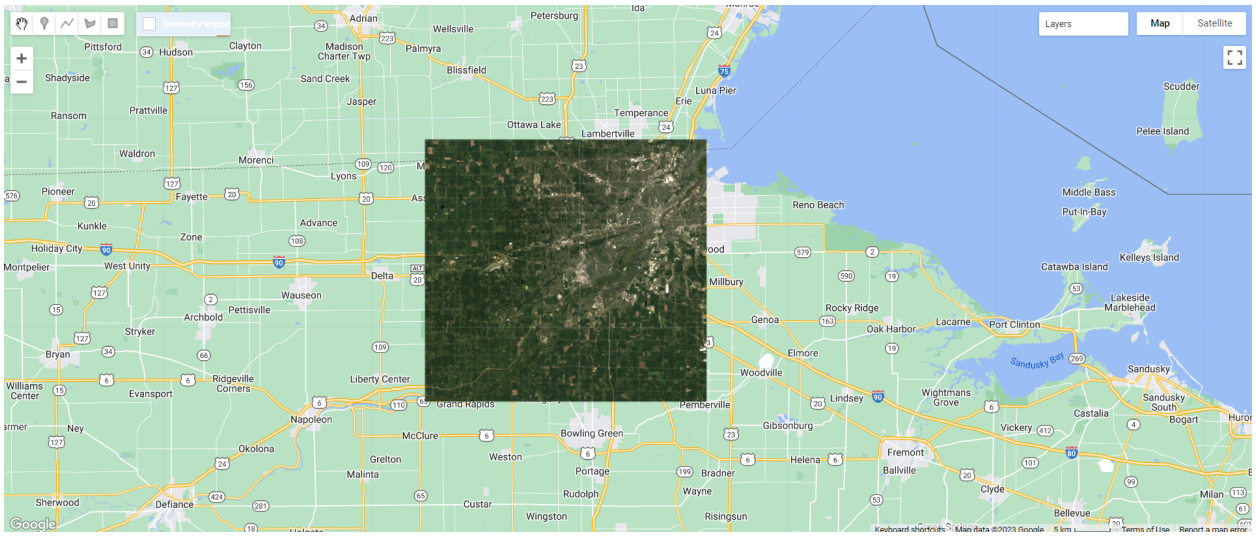
- **Mapping the Natural color of the image**

1) To visualize the natural color (true color) image, we can list the three bands, min, max and gamma.

```
var TCparams = {
  bands: ['SR_B4', 'SR_B3', 'SR_B2'],
  min: 0.0,
  max: 0.3,
  gamma: 1.4,
};
```

2) Then, to add the image into the map, we need to use Map.addLayer(), here LC9mosaicClip refers to the image, TCparams refers to image visualization parameter and 'Natural Color (RGB)' is defined as the layer name.

```
Map.addLayer(LC9mosaicClip, TCparams, 'Natural Color (RGB)');
```



- **Measuring and mapping NDVI**

1) To measure NDVI, we use normalizedDifference(bandNames). NDVI is calculated through (band 5-band 4) / (band 5+ band 4). Also, rename() is used to rename the new band to “NDVI”.

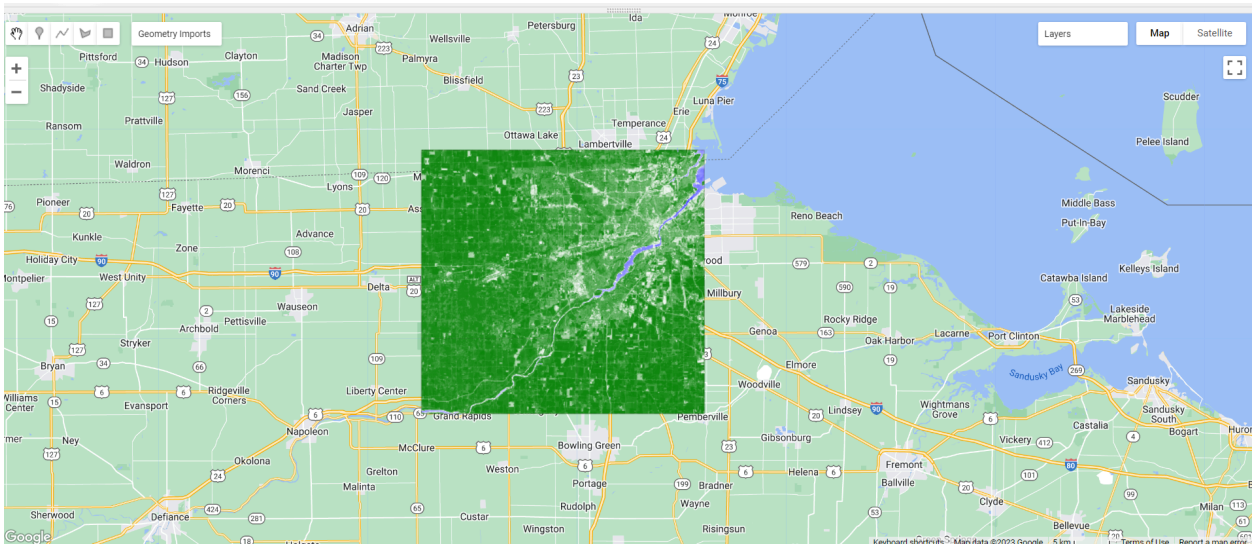
```
var ndvi = LC9mosaicClip.normalizedDifference(['SR_B5', 'SR_B4']).rename('NDVI')
```

2) To visualize the NDVI, we need to define the min and max and also a colors palette.

```
var ndviParams = {min: -1, max: 1, palette: ['blue', 'white', 'green']};
```

3) Then, we use Map.addLayer() to add the image into the map. Here ndvi refers to the image, ndviParams refers to image visualization parameter and 'Normalized Difference Vegetation Index (NDVI)' is defined as the layer name.

```
Map.addLayer(ndvi, ndviParams, 'Normalized Difference Vegetation Index (NDVI)');
```



- **Creating legend for NDVI**

1) By using ui.Thumbnail, we can create a color bar thumbnail image to be used in the legend from the given color palette.

```
var nSteps = 10;

var colorBar = ui.Thumbnail({
  image: ee.Image.pixelLonLat().select(0).int(),
  params: {
    bbox: [0, 0, nSteps, 0.1],
    dimensions: '100x10',
    format: 'png',
```

```
    min: 0,
    max: nSteps,
    palette: ['blue', 'white', 'green'],
  },
  style: {stretch: 'horizontal', margin: '0px 8px', maxHeight: '24px'},
});
```

2) Add labels to the color bar by using ui.Panel. In this case, we have defined three ui.Lable, one refers to the min label, one the center label and one for the max label.

```
var legendLabels = ui.Panel({
  widgets: [
    ui.Label(-1, {margin: '0px 0px'}),
    ui.Label(0,
       {margin: '0px 0px', textAlign: 'center', stretch: 'horizontal'}),
    ui.Label(1, {margin: '0px 0px'})
  ],
  layout: ui.Panel.Layout.flow('horizontal')
});
```

3) Create a legend title by using ui.Label(). Value refers to the title and style refers to the appearance characteristics of the title.

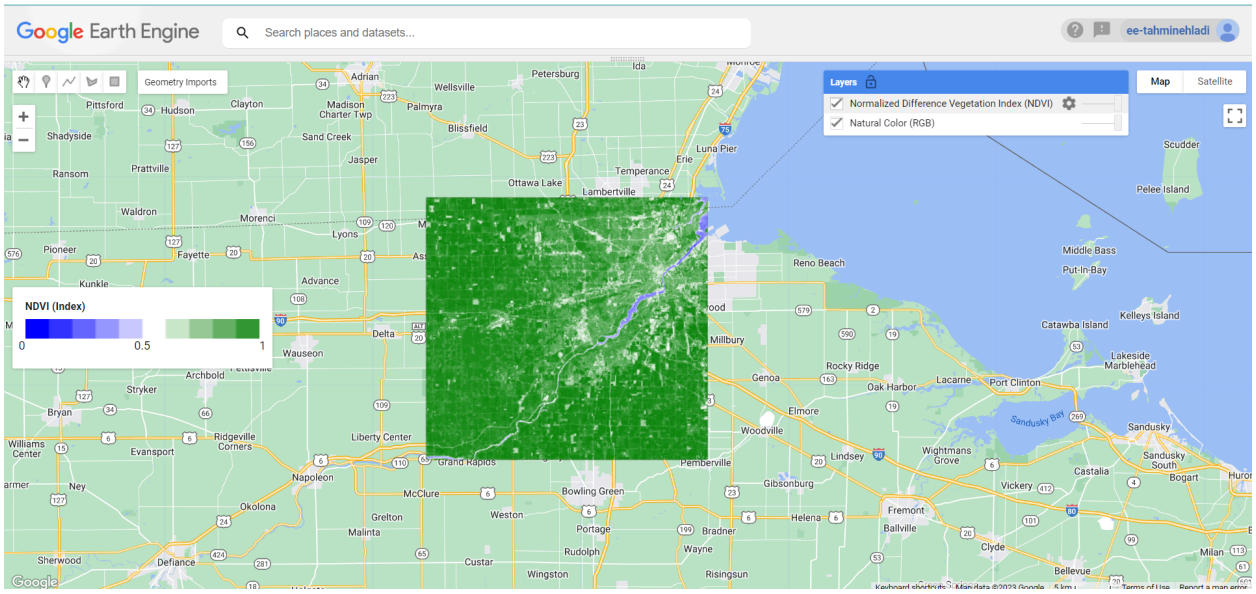
```
var legendTitle = ui.Label({
  value: 'NDVI (Index)',
  style: {fontSize: '13px', textAlign: 'left', fontWeight: 'bold'}
});
```

4) Then, create a panel which contains legendTitle, colorBar and legendLabels by using ui.Panel. Also, we can define the style and position of the panel.

```
var legendPanel = ui.Panel([legendTitle, colorBar, legendLabels]);
legendPanel.style().set({
  height: '100px',
  width: '320px',
  position: 'middle-left',
});
```

5) Finally, by using Map.add(), we can add the legend to the map.

```
Map.add(legendPanel);
```



- **Measuring and mapping LST**

1) To measure land surface temperature, we just need to select band 10 and rename it to LST.

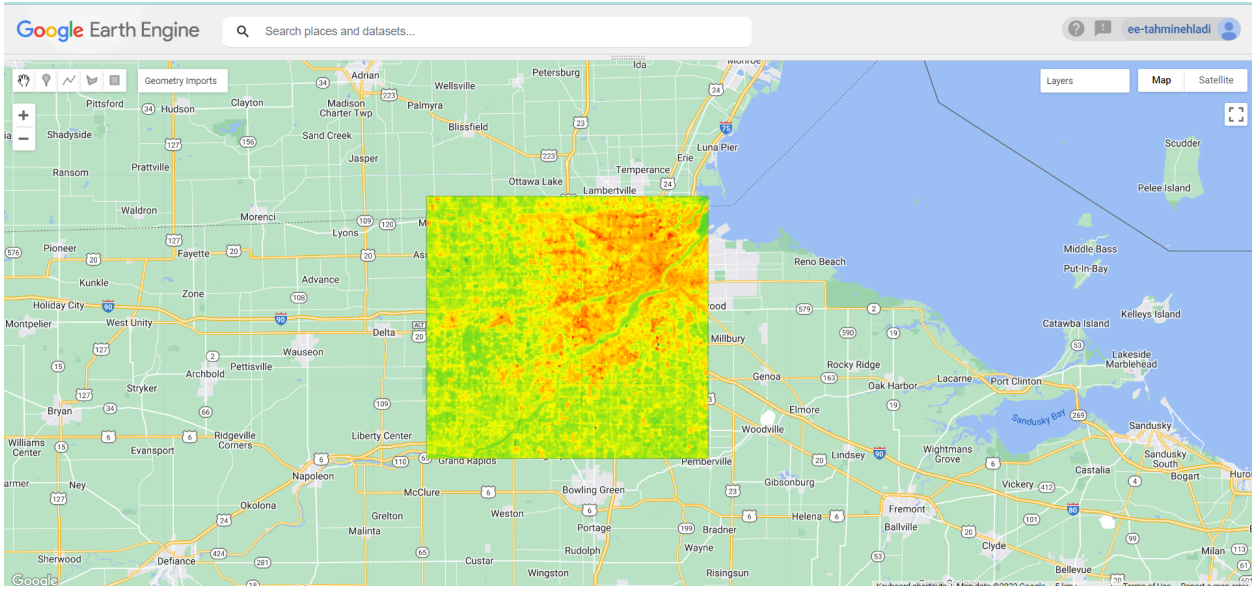
```
var LST = LC9mosaicClip.select("ST_B10").rename('LST');
```

2) Then, to visualize the LST image, we should define the min and max temperature and the appropriate palette to present the LST. The min and max can be changed according to the defined area of interest and date range.

```
var LSTvis = {min: 15, max: 45, palette: ['blue', 'limegreen', 'yellow', 'darkorange', 'red']};
```

3) To add the LST image into the map, again, we need to use Map.addLayer(), here LST is the image, LSTvis is the image visualization parameter and 'Land Surface Temperature (LST)' is the layer name.

```
Map.addLayer(LST, LSTvis, 'Land Surface Temperature (LST)');
```



- **Creating legend for LST**

1) Again, by using ui.Thumbnail, we create a color bar thumbnail image to be used in the legend from the given color palette.

```
var nSteps = 20;

var colorBar2 = ui.Thumbnail({
  image: ee.Image.pixelLonLat().select(0).int(),
  params: {
    bbox: [0, 0, nSteps, 0.1],
    dimensions: '100x10',
    format: 'png',
    min: 0,
    max: nSteps,
    palette: ['blue', 'limegreen', 'yellow', 'darkorange', 'red'],
  },
  style: {stretch: 'horizontal', margin: '0px 8px', maxHeight: '24px'},
});
```

2) Then, we add labels to the color bar by using ui.Panel. In this case, we have defined three ui.Lable, one refers to the min label, one the center label and one for the max label.

```
var legendLabels2 = ui.Panel({
  widgets: [
    ui.Label(15, {margin: '0px 0px'}),
    ui.Label(30,
        {margin: '0px 0px', textAlign: 'center', stretch: 'horizontal'}),
    ui.Label(45, {margin: '0px 0px'})
  ],
  layout: ui.Panel.Layout.flow('horizontal')
});
```

3) We can create a legend title by using ui.Label(). Value shows the title and style refers to the appearance characteristics of the title.

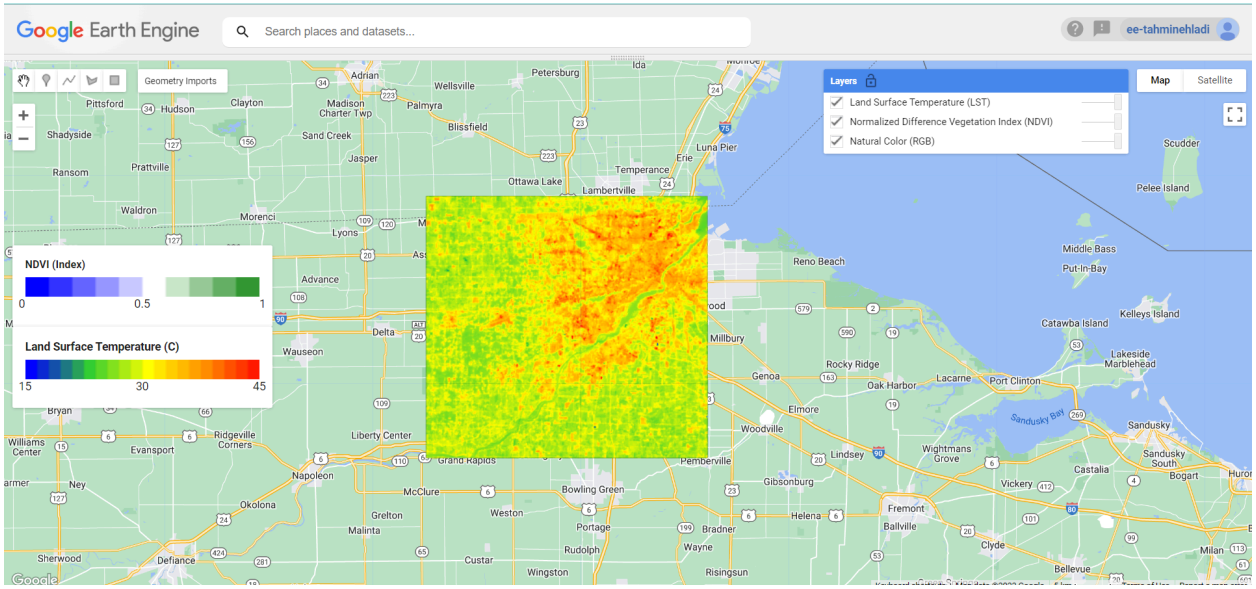
```
var legendTitle2 = ui.Label({
  value: 'Land Surface Temperature (C)',
  style: {fontWeight: 'bold'}
});
```

4) Then, create a panel which contains legendTitle2, colorBar2 and legendLabels2 by using ui.Panel. We can define the style and position of the panel.

```
var legendPanel2 = ui.Panel([legendTitle2, colorBar2, legendLabels2]);
legendPanel2.style().set({
  height: '100px',
  width: '320px',
  position: 'middle-left',
});
```

5) Finally, by using Map.add(), we can add the legend to the map.

```
Map.add(legendPanel2);
```



- **Displaying Date range on the map**

1) To display the date range on the map, we can create two labels using ui.Lable, start date should be defined as the value of one of them, and end date should be defined for the other one.

```
var DateLabels1= ui.Label({
  value: startDate,
  style: {fontWeight: 'normal'}
});

var DateLabels2= ui.Label({
  value: endDate,
  style: {fontWeight: 'normal'}
});
```

2) Again, we can create another variable as DateTitle using ui.Label and define “Date Range” as the value of the variable.

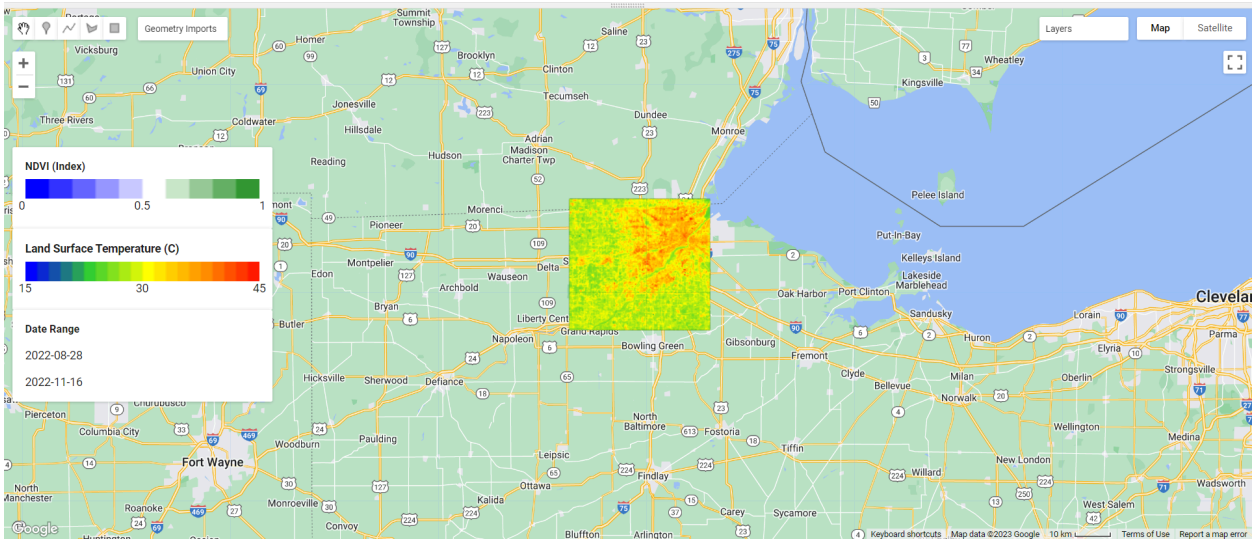
```
var DateTitle = ui.Label({
  value: 'Date Range',
   style: {fontSize: '13px', textAlign: 'left', fontWeight: 'bold'}
});
```

3) Then, we need to create a panel using ui.Panel which contains the three labels that we already created in the previous steps.

```
var DatePanel= ui.Panel([DateTitle, DateLabels1, DateLabels2]);
DatePanel.style().set({
  width: '320px',
  position: 'middle-left'
   });
```

4) Finally, by using Map.add(), we can add the date range into the map.

```
  Map.add(DatePanel)
```



- **Exporting NDVI data through Export to Drive**

1) To export the NDVI image into SHP or CSV, we need to convert the image to featureCollection. And to do so, we need to convert the NDVI band values from decimal numbers to integer. The point about NDVI is that its values are between -1 to +1, therefore, if we convert it into integer all will become to be Zero. Therefore, here, we measure NDVI again and this time multiply it with 100.

```
var ndvi2= LC9mosaicClip.normalizedDifference(['SR_B5',
'SR_B4']).multiply(100).rename('NDVI');
```

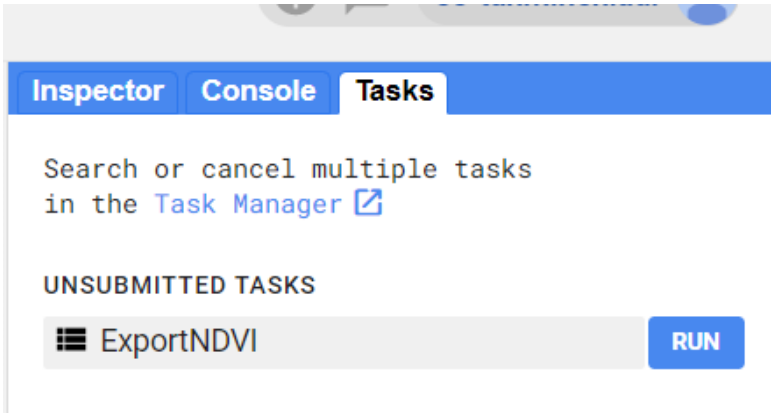
2) Then by using toInt(), convert the band values into integer and by using .reduceToVectors(), convert the NDVI image to a feature collection.

```
var ndviExport=ndvi2.toInt();
    var vectorsNDVI =ndviExport.reduceToVectors({
      geometry: userArea,
       scale: 500,
      geometryType: 'centroid',
      labelProperty: 'NDVI'
      });
```

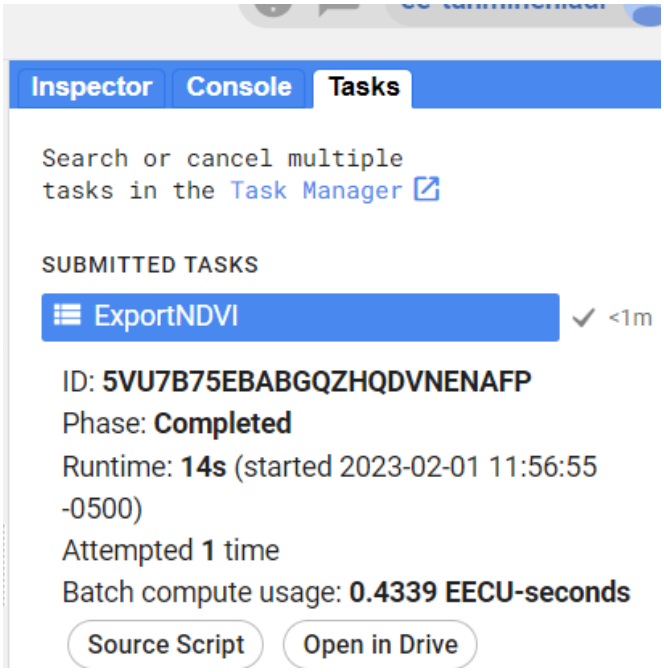
3) Then, by using Export.table.toDrive(), we can export the data. In fileFormat section we can either use SHP or CSV.

```
Export.table.toDrive({
 collection: vectorsNDVI,
 description: 'ExportNDVI',
 fileFormat: 'SHP'

 });
```



4) Press Run. It may take a few minutes.



5) Then, press Open in Drive.

- **Exporting NDVI data through getDownloadURL**

1) To get the download link to export the data, we can use getDownloadURL(), and define the format, selectors, and filename.

```
var downloadUrlNDVI = vectorsNDVI.getDownloadURL({
 format: 'CSV',
 selectors: ['NDVI', '.geo'],
 filename: 'NDVI layer'
 });
```

2) Then, we can define the label and set the Url of the label, and finally add it to the map.

```
 var urlLabelNDVI = ui.Label('Download NDVI as a csv file', {shown: false, position: ('bottom-right')});
 urlLabelNDVI.setUrl(downloadUrlNDVI);
 urlLabelNDVI.style().set({shown: true});
 Map.add(urlLabelNDVI);
```

- **Exporting LST data through Export to Drive**

1) Again to export the LST data into SHP and CSV, at first, we need to convert LST image to featureCollection, to do so we use toInt(), and reduceToVectors().

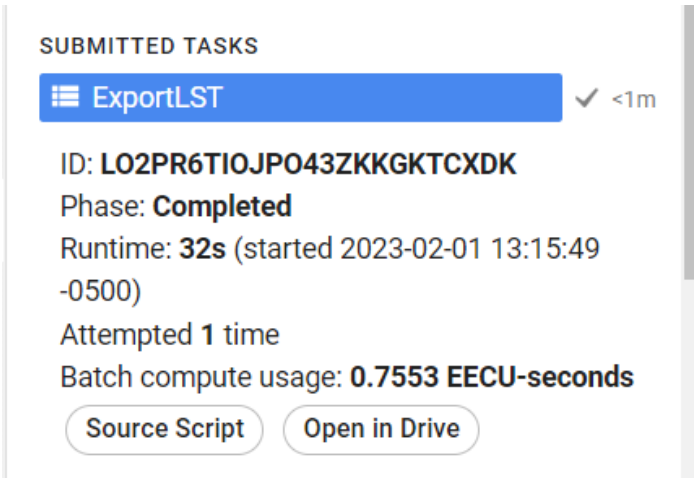
```
var LSTExport=LST.toInt();
    var vectorsLST =LSTExport.reduceToVectors({
      geometry: userArea,
      scale: 500,
      geometryType: 'centroid',
      labelProperty: 'LST'
    });
```

2) Then, we can use Export.table.toDrive() to export it into SHP or CSV.

```
Export.table.toDrive({
collection: vectorsLST,
description: 'ExportLST',
fileFormat: 'SHP'

});
```

3) Finally, press run, wait few minutes and then press open in Drive to download the data.



- **Exporting LST data through getDownloadURL**

1) To get the download link to export the LST data, again, we can use getDownloadURL(), and define the format, selectors, and filename.

```
var downloadUrlLST = vectorsLST.getDownloadURL({
format: 'CSV',
selectors: ['LST', '.geo'],
filename: 'LST layer'
});
```

2) Then, we can define the label and set the Url of the label, and finally add it to the map.

```
var urlLabelLST = ui.Label('Download LST as a csv file', {shown: false, position: ('bottom-right')});
 urlLabelLST.setUrl(downloadUrlLST);
 urlLabelLST.style().set({shown: true});
 Map.add(urlLabelLST);
```