Mapping and Analyzing Waterbodies within Anumanadhi Sub-basin and Visualizing Water Status Patterns Over a Decade

Name: Alex Christopher M

Internship Duration: 1 Month

Organization Name: Uvakai Research Foundation

Start Date: 13 May 2024

End Date: 21 June 2024

Acknowledgements

I would like to express my deepest gratitude to Udhaya Rajan N, the Director of Uvakai Research Foundation, for providing me with this invaluable internship opportunity. His guidance and support have been instrumental in the successful completion of this project. I also extend my sincere thanks to the entire team at Uvakai Research Foundation for their cooperation and assistance. Special thanks to the research team for their continuous support and for providing the necessary resources.

Abstract

This report presents an analysis of the Radhapuram and Anumanadhi Canal systems, focusing on their contributions to the 194 interconnected lakes. Our primary objectives included analyzing water flow between lakes, assessing capacity during monsoon seasons, categorizing lakes into system and non-system tanks, and clustering lakes into groups.

Using data from a reliable web source, we evaluated lake capacities to identify blockages and insufficient capacities. By visualizing the interconnected lakes, we determined water flow patterns and identified anomalies requiring attention. The monsoon analysis evaluated lake storage capacities during the two annual monsoon seasons, offering insights into seasonal water level variations.

We categorized the lakes into system and non-system tanks to understand their management and usage better. Additionally, we clustered the lakes based on an external reference, which helped visualize the spatial organization and management of the lakes.

Table of Contents

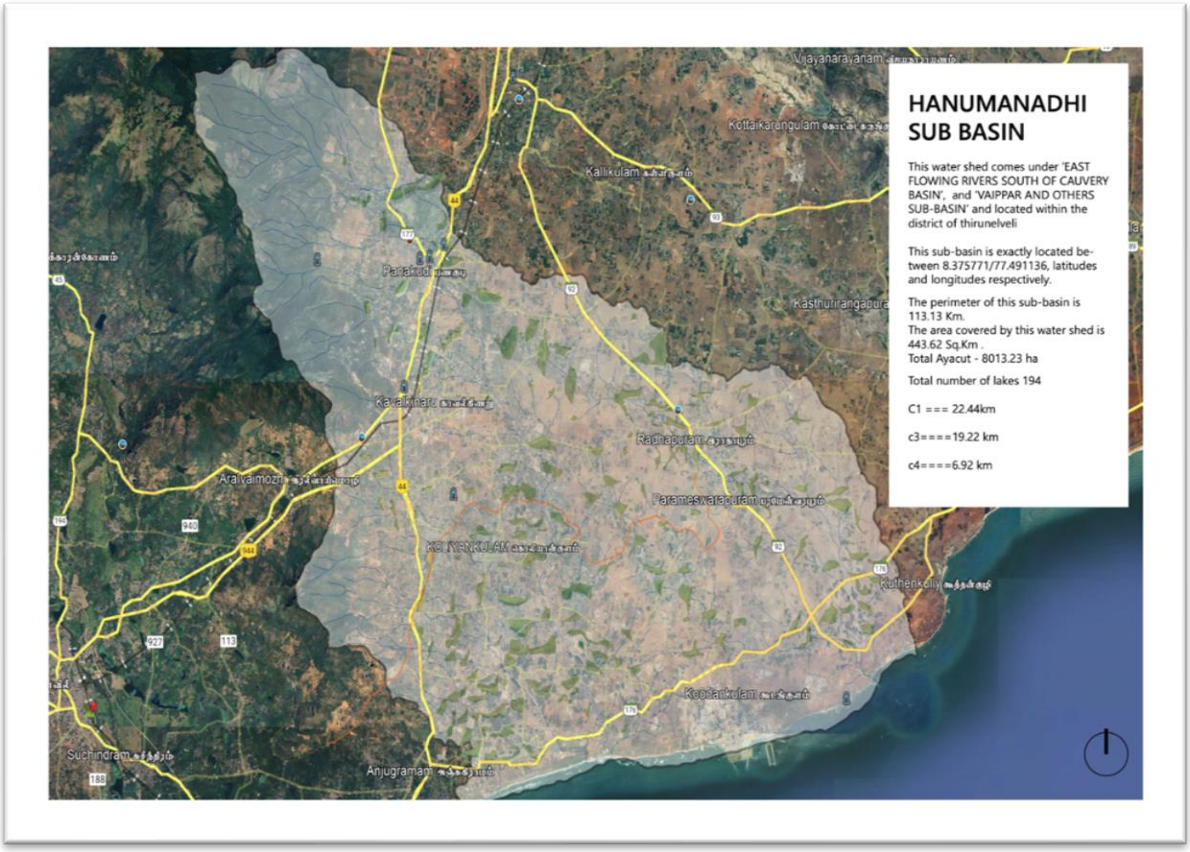
Introduction

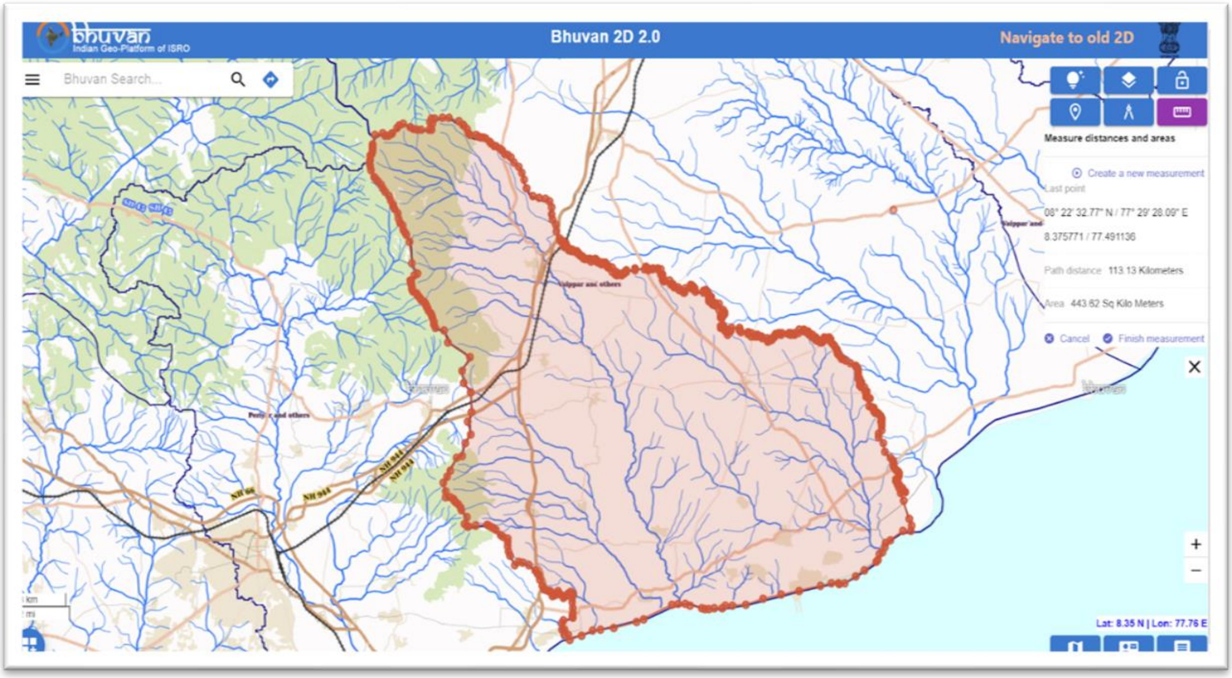
The Vaippar and Others Sub-Basin, located within the district of Tirunelveli in Tamil Nadu, India, forms a crucial part of the region's water management and agricultural infrastructure. Encompassing the southern region of Tirunelveli, this sub-basin plays an invaluable role in sustaining the local ecosystem and supporting socio-economic development. The Hanumanadhi Sub-Basin, a significant component of this larger sub-basin, is precisely situated between the latitudes of 8°37'35"N and 77°41'36"E, covering a substantial area of 443.62 square kilometers.

This watershed area is characterized by its network of 194 lakes, which are integral to local agricultural activities and water management practices. The lakes collectively hold an approximate ayacut of 30,812.23 hectares, contributing significantly to the irrigation needs of the region. The Hanumanadhi Sub-Basin, delineated in the satellite imagery, showcases the extensive network of water bodies and their interconnected nature.

The sub-basin's geography is vital not only for its direct water resources but also for its impact on the socio-economic development of the area. Despite the absence of major rivers, the watershed flows towards the Gulf of Mannar in southern Tamil Nadu, indicating a unique hydrological pattern that necessitates careful management and study.

The below pictures will provide a visual representation of the introduction, highlighting key features such as the location of the lakes, the flow patterns, and the overall topography of the region.


HANUMANADHI SUB BASIN
This water shed comes under 'EAST FLOWING RIVERS SOUTH OF CAUVERY BASIN', and 'VAIPPAR AND OTHERS SUB-BASIN' and located within the district of thirunelveli
This sub-basin is exactly located between 8.375771/77.491136, latitudes and longitudes respectively.
The perimeter of this sub-basin is 113.13 Km.
The area covered by this water shed is 443.62 Sq.Km .
Total Ayacut - 8013.23 ha
Total number of lakes 194
C1 === 22.44km
c3====19.22 km
c4====6.92 km
Vijayanarayanam
Kottaikarungulam
Kallikulam
Panakudi
Kasthurirangapuram
Kavalkinaru
Radhapuram
Aralvaimozhi
Parameswarapuram
KOLIYANKULAM
Kuthenkully
Kootankulam
Suchindram
Anjugramam


bhuvan
Indian Geo-Platform of ISRO
Bhuvan 2D 2.0
Navigate to old 2D
Bhuvan Search...
Measure distances and areas
Create a new measurement
Last point
08° 22' 32.77" N / 77° 29' 28.09" E
8.375771 / 77.491136
Path distance 113.13 Kilometers
Area 443.62 Sq Kilo Meters
Cancel
Finish measurement
Vaippar and others
NH 944
NH 66
Lat: 8.35 N | Lon: 77.76 E

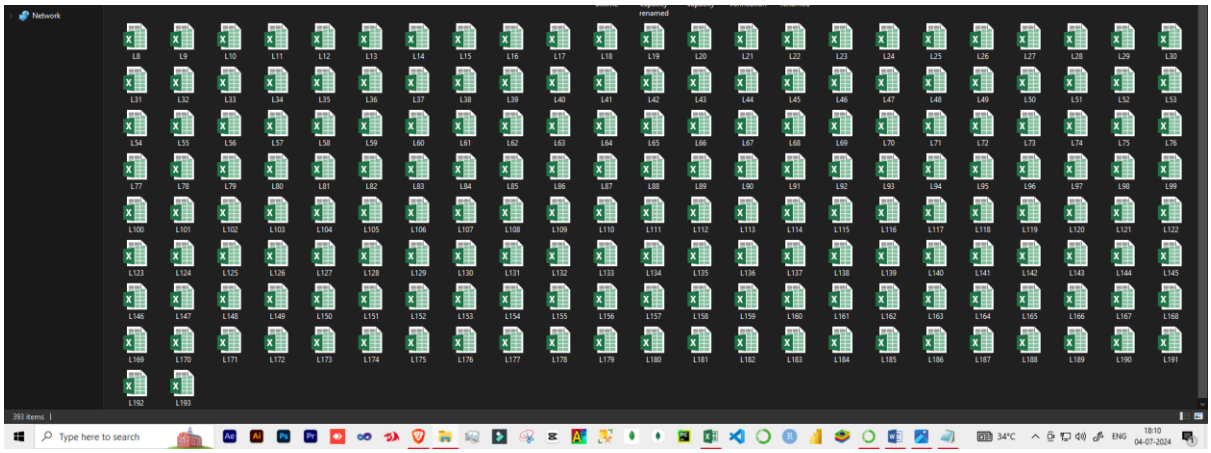
Tools and Technologies Used

- **Python programming language** played a crucial role in data pre-processing, including cleaning, integration of data from 194 lakes, and transformation into suitable formats for analysis.
- **Jupyter Notebook** emerged as a highly useful platform for conducting data analysis tasks efficiently and interactively, offering a conducive environment for coding in Python and documenting the analysis process.
- **Power BI** served as the primary tool for data visualization, enabling the creation of interactive charts and graphs that provided insights into lake capacities and water flow dynamics.
- **Synoptic Designer** was instrumental in creating custom SVG maps with unique identifiers for each lake, facilitating visual representation of lake data and aiding in the identification of key patterns and anomalies.
- **Microsoft Excel** was used for storing and accessing lake data, providing a structured platform for organizing and managing datasets.

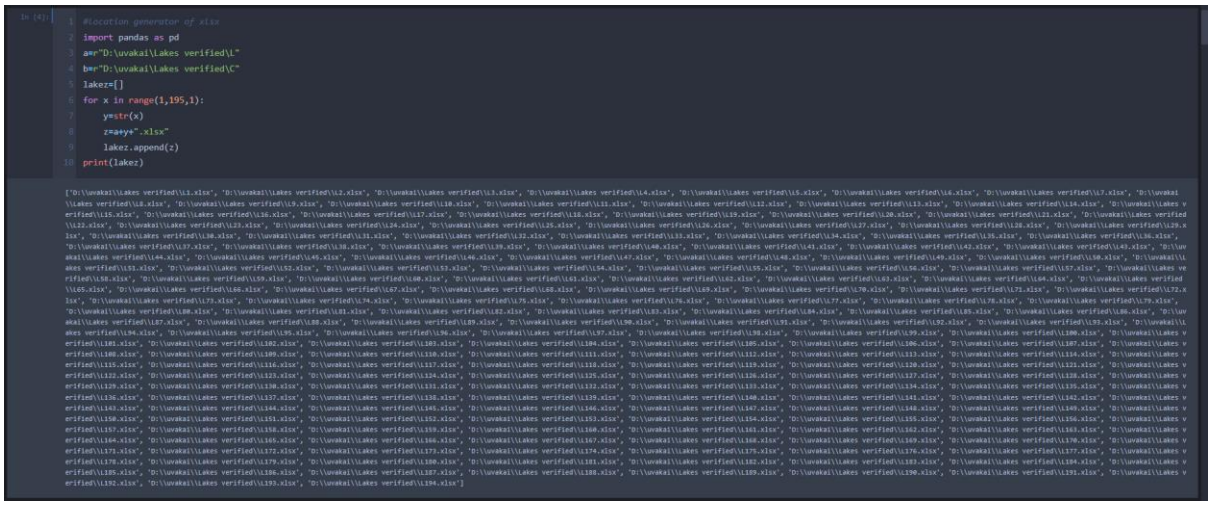
Data Collection

The data collection process involved accessing the Water Body Information System (WBIS) website, dedicated to information on 194 lakes associated with Radhapuram and Anumanadhi Canals. Data on lake capacity and water spread area was manually extracted from WBIS by copying details for each lake. Each lake's data was meticulously recorded in separate Excel files, ensuring accuracy and organization. This methodical approach facilitated comprehensive data analysis and visualization, supporting the assessment of water management and flow dynamics within the canal systems.

Converting XLSX Files to CSV Files for Python Import - To facilitate data import and processing in Python, the XLSX files containing data for all 194 lakes were converted to CSV format. This conversion streamlined the data handling process, making it compatible with Python's data analysis libraries such as pandas.

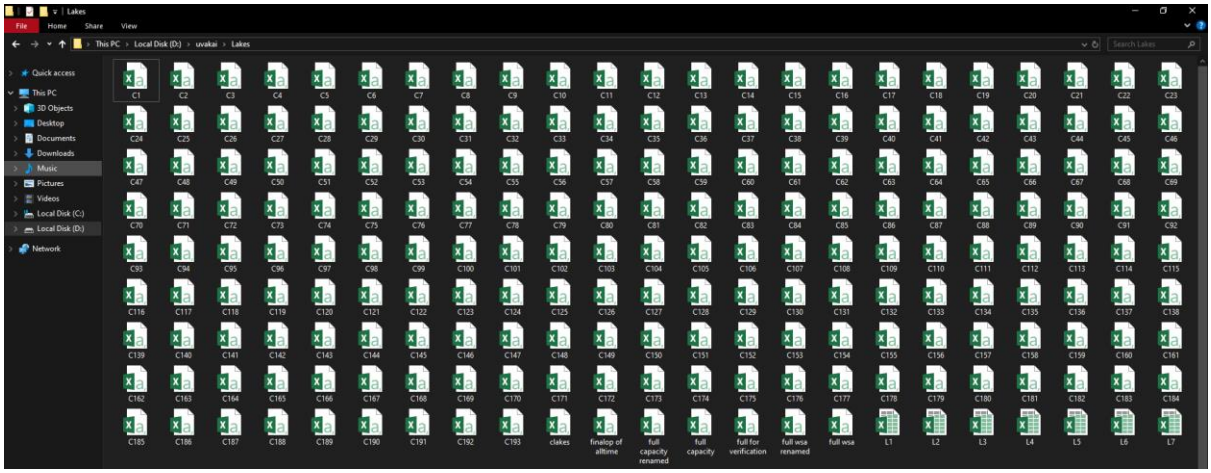


```
#location generator of xlsx
import pandas as pd
a="D:\uvakai\Lakes verified\L"
b="D:\uvakai\Lakes verified\C"
lakez=[]
for x in range(1,195,1):
    y=str(x)
    z=a+y+".xlsx"
    lakez.append(z)
print(lakez)
```



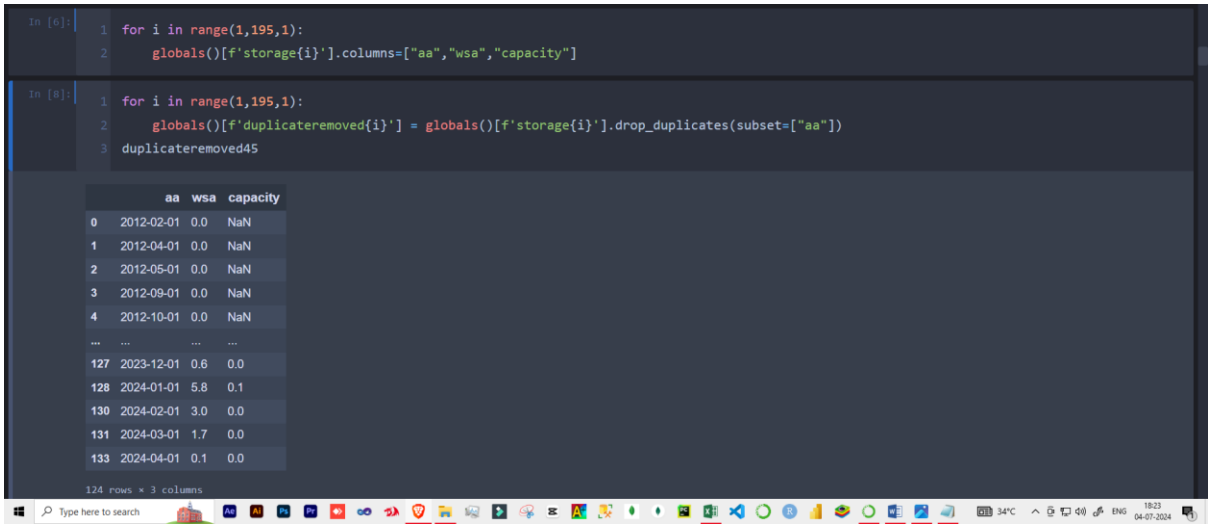
```
#location generator of csv
alexz=[]
for x in range(1,195,1):
    y=str(x)
    z=b+y+".csv"
    alexz.append(z)
print(alexz)
```

```
#converting xlsx to csv
for x in range(0,len(lakez),1):
    dummy1=pd.read_excel(lakez[x])
    dummy2=dummy1.to_csv(alexz[x],index=False)
```



Data Cleaning

During data cleaning, I focused on ensuring the accuracy and consistency of the dataset sourced from the Water Body Information System (WBIS). Using Python's Pandas library, I removed duplicate entries, particularly for lakes with capacity data spanning from 2012 to April 2024. This involved employing the drop_duplicates( ) function to eliminate repeated records based on date. Additionally, I standardized column names across the dataset to "capacity" and "water spread area" to unify the data format obtained from Bhuvan, which initially varied in naming conventions. These efforts were crucial in preparing a clean and reliable dataset for subsequent analysis, ensuring that insights derived from the data were based on accurate and consistent information.



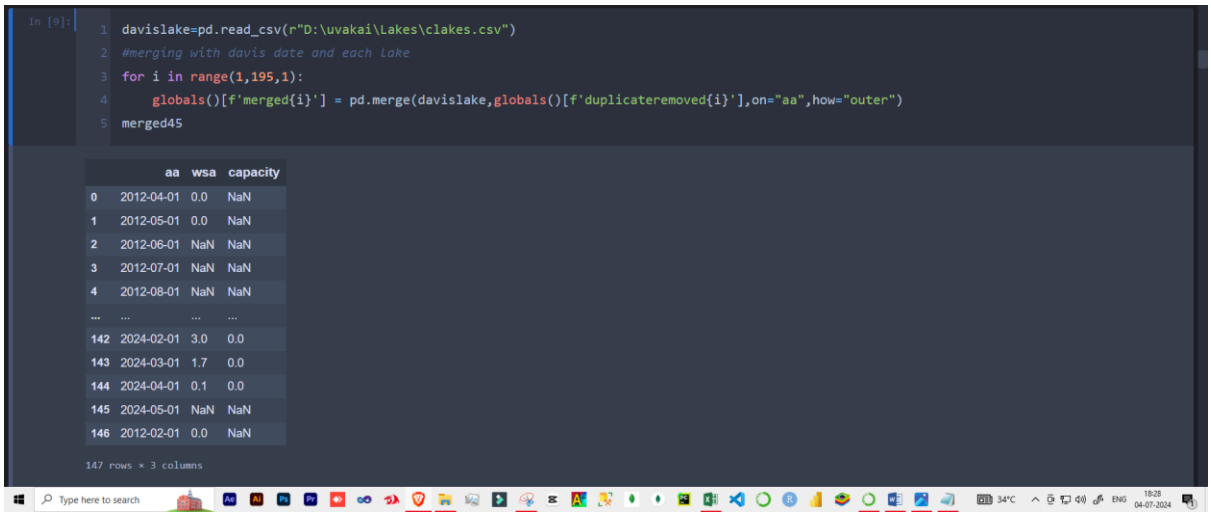
```
In [6]:
for i in range(1,195,1):
    globals()[f'storage{i}'].columns=["aa","wsa","capacity"]

In [8]:
for i in range(1,195,1):
    globals()[f'duplicateremoved{i}'] = globals()[f'storage{i}'].drop_duplicates(subset=["aa"])
duplicateremoved45
```

	aa	wsa	capacity
0	2012-02-01	0.0	NaN
1	2012-04-01	0.0	NaN
2	2012-05-01	0.0	NaN
3	2012-09-01	0.0	NaN
4	2012-10-01	0.0	NaN
...	...	...	...
127	2023-12-01	0.6	0.0
128	2024-01-01	5.8	0.1
130	2024-02-01	3.0	0.0
131	2024-03-01	1.7	0.0
133	2024-04-01	0.1	0.0

124 rows × 3 columns

Data Integration

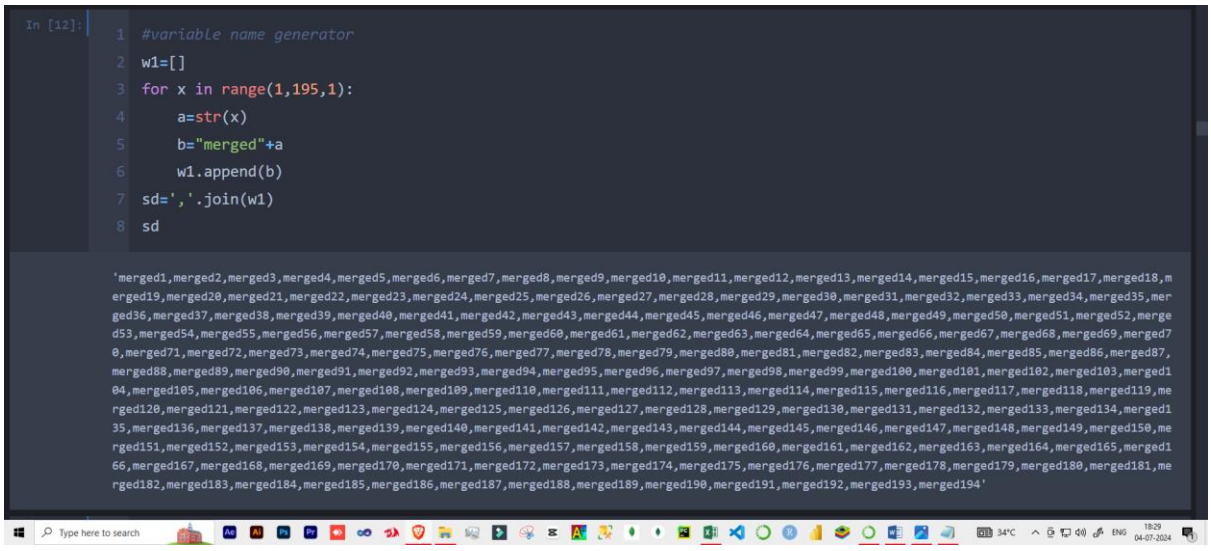
After cleaning 194 individual lake datasets by removing duplicates and standardizing column names, a common dataset of 147 dates was established. Using Python's Pandas library, these dates served as a reference for merging each lake's data into a unified dataset. The merge( ) function was preferred over concat( ) due to varying row counts across datasets, ensuring that all available capacity and water spread area data aligned correctly by date. Dates missing from any lake's dataset were handled as null, preserving data integrity. This approach facilitated the creation of a cohesive dataset ready for comprehensive analysis and visualization in subsequent stages of the project.



```
In [9]:
davislake=pd.read_csv(r"D:\uvakai\Lakes\clakes.csv")
#merging with davis date and each lake
for i in range(1,195,1):
    globals()[f'merged{i}'] = pd.merge(davislake,globals()[f'duplicateremoved{i}'],on="aa",how="outer")
merged45
```

	aa	wsa	capacity
0	2012-04-01	0.0	NaN
1	2012-05-01	0.0	NaN
2	2012-06-01	NaN	NaN
3	2012-07-01	NaN	NaN
4	2012-08-01	NaN	NaN
...	...	...	...
142	2024-02-01	3.0	0.0
143	2024-03-01	1.7	0.0
144	2024-04-01	0.1	0.0
145	2024-05-01	NaN	NaN
146	2012-02-01	0.0	NaN

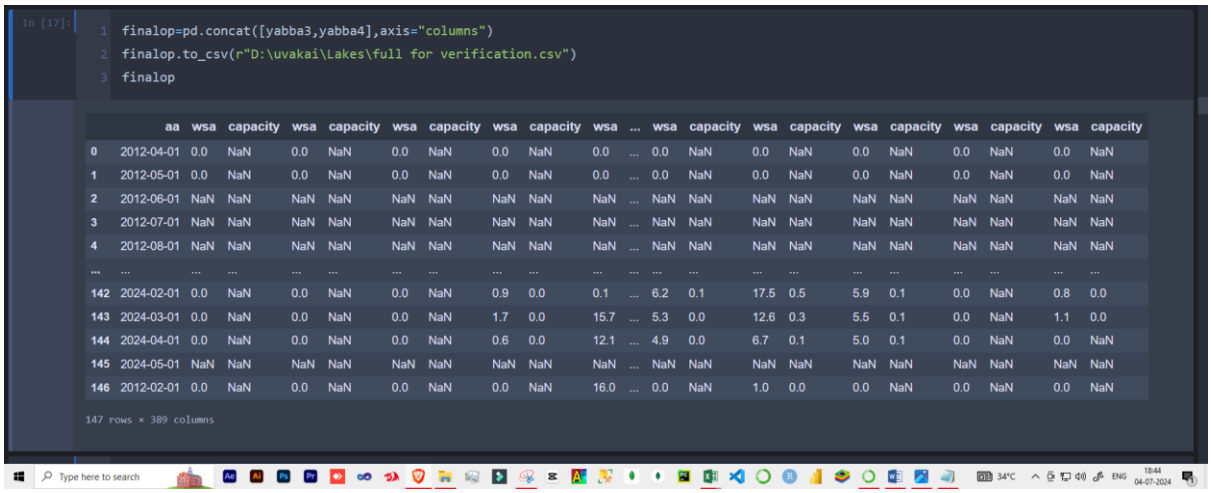
147 rows × 3 columns



```
In [12]:
#variable name generator
w1=[]
for x in range(1,195,1):
    a=str(x)
    b="merged"+a
    w1.append(b)
sd=','.join(w1)
sd
```

```
'merged1,merged2,merged3,merged4,merged5,merged6,merged7,merged8,merged9,merged10,merged11,merged12,merged13,merged14,merged15,merged16,merged17,merged18,merged19,merged20,merged21,merged22,merged23,merged24,merged25,merged26,merged27,merged28,merged29,merged30,merged31,merged32,merged33,merged34,merged35,merged36,merged37,merged38,merged39,merged40,merged41,merged42,merged43,merged44,merged45,merged46,merged47,merged48,merged49,merged50,merged51,merged52,merged53,merged54,merged55,merged56,merged57,merged58,merged59,merged60,merged61,merged62,merged63,merged64,merged65,merged66,merged67,merged68,merged69,merged70,merged71,merged72,merged73,merged74,merged75,merged76,merged77,merged78,merged79,merged80,merged81,merged82,merged83,merged84,merged85,merged86,merged87,merged88,merged89,merged90,merged91,merged92,merged93,merged94,merged95,merged96,merged97,merged98,merged99,merged100,merged101,merged102,merged103,merged104,merged105,merged106,merged107,merged108,merged109,merged110,merged111,merged112,merged113,merged114,merged115,merged116,merged117,merged118,merged119,merged120,merged121,merged122,merged123,merged124,merged125,merged126,merged127,merged128,merged129,merged130,merged131,merged132,merged133,merged134,merged135,merged136,merged137,merged138,merged139,merged140,merged141,merged142,merged143,merged144,merged145,merged146,merged147,merged148,merged149,merged150,merged151,merged152,merged153,merged154,merged155,merged156,merged157,merged158,merged159,merged160,merged161,merged162,merged163,merged164,merged165,merged166,merged167,merged168,merged169,merged170,merged171,merged172,merged173,merged174,merged175,merged176,merged177,merged178,merged179,merged180,merged181,merged182,merged183,merged184,merged185,merged186,merged187,merged188,merged189,merged190,merged191,merged192,merged193,merged194'
```

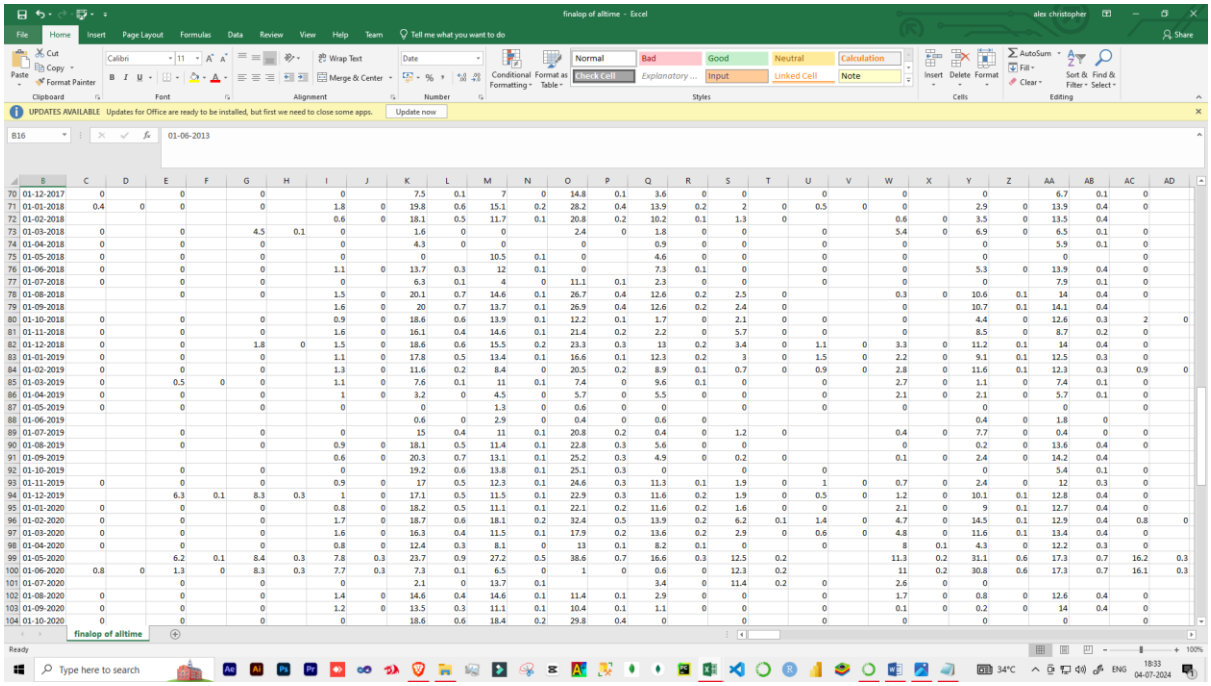
After merging data frames for 194 lakes using a common date reference, 147 merged data frames were concatenated column-wise to form a unified dataset. This consolidated dataset, containing capacity and water spread area data for each lake across 147 dates, was exported to a single Excel file. This file served as the input for Power BI, enabling effective visualization and analysis of the integrated lake data.



```
In [17]:
finalop=pd.concat([yabba3,yabba4],axis="columns")
finalop.to_csv(r"D:\uvakai\Lakes\full for verification.csv")
finalop
```

	aa	wsa	capacity	wsa	capacity	wsa	capacity	wsa	capacity	wsa	...	wsa	capacity	wsa	capacity	wsa	capacity	wsa	capacity	wsa	capacity
0	2012-04-01	0.0	NaN	0.0	NaN	0.0	NaN	0.0	NaN	0.0	...	0.0	NaN	0.0	NaN	0.0	NaN	0.0	NaN	0.0	NaN
1	2012-05-01	0.0	NaN	0.0	NaN	0.0	NaN	0.0	NaN	0.0	...	0.0	NaN	0.0	NaN	0.0	NaN	0.0	NaN	0.0	NaN
2	2012-06-01	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	2012-07-01	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	2012-08-01	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
142	2024-02-01	0.0	NaN	0.0	NaN	0.0	NaN	0.9	0.0	0.1	...	6.2	0.1	17.5	0.5	5.9	0.1	0.0	NaN	0.8	0.0
143	2024-03-01	0.0	NaN	0.0	NaN	0.0	NaN	1.7	0.0	15.7	...	5.3	0.0	12.6	0.3	5.5	0.1	0.0	NaN	1.1	0.0
144	2024-04-01	0.0	NaN	0.0	NaN	0.0	NaN	0.6	0.0	12.1	...	4.9	0.0	6.7	0.1	5.0	0.1	0.0	NaN	0.0	NaN
145	2024-05-01	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
146	2012-02-01	0.0	NaN	0.0	NaN	0.0	NaN	0.0	NaN	16.0	...	0.0	NaN	1.0	0.0	0.0	NaN	0.0	NaN	0.0	NaN

147 rows × 389 columns



Data Transformation

Following the integration of data from 194 lakes into a unified dataset, the next crucial step was data transformation to tailor the dataset for specific analyses. Initially containing columns for dates, capacity, and water spread area for each lake, the dataset underwent selective column filtering. Depending on the analysis requirements, either capacity or water spread area columns were retained while the other was removed, ensuring focus on essential variables. Capacity data was transformed into categorical types: 0-20%, 20-40%, 40-60%, 60-80%, and 80-100%, facilitating qualitative analysis of lake capacities. This categorization streamlined the dataset, making it easier to interpret and analyze lake characteristics. The transformation process upheld data integrity, maintaining consistency throughout. By aligning transformations with project objectives, the dataset was structured for efficient analysis and visualization in subsequent stages, enhancing decision-making support for lake management and water resource planning.

In [19]:
```
ok1=finalop.drop("capacity",axis="columns")
ok1
```

	aa	wsa	wsa	wsa	wsa	wsa	wsa	wsa	wsa	wsa	...	wsa	wsa	wsa	wsa	wsa	wsa	wsa	wsa	wsa	wsa
0	2012-04-01	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.2	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	2012-05-01	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	2012-06-01	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	2012-07-01	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	2012-08-01	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
142	2024-02-01	0.0	0.0	0.0	0.9	0.1	1.1	18.3	14.1	1.7	...	4.3	15.2	3.2	0.1	8.4	6.2	17.5	5.9	0.0	0.8
143	2024-03-01	0.0	0.0	0.0	1.7	15.7	10.6	21.4	13.5	1.0	...	7.0	6.4	0.7	0.0	7.4	5.3	12.6	5.5	0.0	1.1
144	2024-04-01	0.0	0.0	0.0	0.6	12.1	8.8	5.9	7.9	0.0	...	2.1	7.3	0.3	0.0	5.7	4.9	6.7	5.0	0.0	0.0
145	2024-05-01	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2.2	NaN	...	NaN	1.3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
146	2012-02-01	0.0	0.0	0.0	0.0	16.0	NaN	NaN	3.4	0.0	...	NaN	NaN	NaN	1.7	NaN	0.0	1.0	0.0	0.0	0.0

147 rows × 195 columns

In [20]:
```
h1=[]
for x in range(1,195,1):
    a="wsa"
    b=str(x)
    c=a+b
    h1.append(c)
print(h1)
```

```
['wsa1', 'wsa2', 'wsa3', 'wsa4', 'wsa5', 'wsa6', 'wsa7', 'wsa8', 'wsa9', 'wsa10', 'wsa11', 'wsa12', 'wsa13', 'wsa14', 'wsa15', 'wsa16', 'wsa17', 'wsa18', 'wsa19', 'wsa20', 'wsa21', 'wsa
22', 'wsa23', 'wsa24', 'wsa25', 'wsa26', 'wsa27', 'wsa28', 'wsa29', 'wsa30', 'wsa31', 'wsa32', 'wsa33', 'wsa34', 'wsa35', 'wsa36', 'wsa37', 'wsa38', 'wsa39', 'wsa40', 'wsa41', 'wsa42',
'wsa43', 'wsa44', 'wsa45', 'wsa46', 'wsa47', 'wsa48', 'wsa49', 'wsa50', 'wsa51', 'wsa52', 'wsa53', 'wsa54', 'wsa55', 'wsa56', 'wsa57', 'wsa58', 'wsa59', 'wsa60', 'wsa61', 'wsa62', 'wsa6
3', 'wsa64', 'wsa65', 'wsa66', 'wsa67', 'wsa68', 'wsa69', 'wsa70', 'wsa71', 'wsa72', 'wsa73', 'wsa74', 'wsa75', 'wsa76', 'wsa77', 'wsa78', 'wsa79', 'wsa80', 'wsa81', 'wsa82', 'wsa83',
'wsa84', 'wsa85', 'wsa86', 'wsa87', 'wsa88', 'wsa89', 'wsa90', 'wsa91', 'wsa92', 'wsa93', 'wsa94', 'wsa95', 'wsa96', 'wsa97', 'wsa98', 'wsa99', 'wsa100', 'wsa101', 'wsa102', 'wsa103',
'wsa104', 'wsa105', 'wsa106', 'wsa107', 'wsa108', 'wsa109', 'wsa110', 'wsa111', 'wsa112', 'wsa113', 'wsa114', 'wsa115', 'wsa116', 'wsa117', 'wsa118', 'wsa119', 'wsa120', 'wsa121', 'wsa1
22', 'wsa123', 'wsa124', 'wsa125', 'wsa126', 'wsa127', 'wsa128', 'wsa129', 'wsa130', 'wsa131', 'wsa132', 'wsa133', 'wsa134', 'wsa135', 'wsa136', 'wsa137', 'wsa138', 'wsa139', 'wsa140',
'wsa141', 'wsa142', 'wsa143', 'wsa144', 'wsa145', 'wsa146', 'wsa147', 'wsa148', 'wsa149', 'wsa150', 'wsa151', 'wsa152', 'wsa153', 'wsa154', 'wsa155', 'wsa156', 'wsa157', 'wsa158', 'wsa1
59', 'wsa160', 'wsa161', 'wsa162', 'wsa163', 'wsa164', 'wsa165', 'wsa166', 'wsa167', 'wsa168', 'wsa169', 'wsa170', 'wsa171', 'wsa172', 'wsa173', 'wsa174', 'wsa175', 'wsa176', 'wsa177',
'wsa178', 'wsa179', 'wsa180', 'wsa181', 'wsa182', 'wsa183', 'wsa184', 'wsa185', 'wsa186', 'wsa187', 'wsa188', 'wsa189', 'wsa190', 'wsa191', 'wsa192', 'wsa193', 'wsa194']
```

In [21]:
```
ok2=ok1.drop("aa",axis="columns")
ok2.columns=h1
gg1=pd.concat([yabba3,ok2],axis="columns")
gg1
```

	aa	wsa1	wsa2	wsa3	wsa4	wsa5	wsa6	wsa7	wsa8	wsa9	...	wsa185	wsa186	wsa187	wsa188	wsa189	wsa190	wsa191	wsa192	wsa193	wsa194
0	2012-04-01	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.2	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	2012-05-01	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	2012-06-01	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	2012-07-01	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	2012-08-01	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
142	2024-02-01	0.0	0.0	0.0	0.9	0.1	1.1	18.3	14.1	1.7	...	4.3	15.2	3.2	0.1	8.4	6.2	17.5	5.9	0.0	0.8
143	2024-03-01	0.0	0.0	0.0	1.7	15.7	10.6	21.4	13.5	1.0	...	7.0	6.4	0.7	0.0	7.4	5.3	12.6	5.5	0.0	1.1
144	2024-04-01	0.0	0.0	0.0	0.6	12.1	8.8	5.9	7.9	0.0	...	2.1	7.3	0.3	0.0	5.7	4.9	6.7	5.0	0.0	0.0
145	2024-05-01	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2.2	NaN	...	NaN	1.3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
146	2012-02-01	0.0	0.0	0.0	0.0	16.0	NaN	NaN	3.4	0.0	...	NaN	NaN	NaN	1.7	NaN	0.0	1.0	0.0	0.0	0.0

147 rows × 195 columns

In [22]:
```
ok3=finalop.drop("wsa",axis="columns")
ok3
```

	aa	capacity	capacity	capacity	capacity	capacity	capacity	capacity	capacity	capacity	...	capacity	capacity	capacity	capacity	capacity	capacity	capacity	capacity
0	2012-04-01	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.0	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	2012-05-01	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	2012-06-01	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	2012-07-01	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	2012-08-01	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
142	2024-02-01	NaN	NaN	NaN	0.0	0.0	0.0	0.2	0.2	0.0	...	0.0	0.2	0.0	0.0	0.1	0.1	0.5	0.1
143	2024-03-01	NaN	NaN	NaN	0.0	0.4	0.1	0.2	0.2	0.0	...	0.0	0.0	0.0	NaN	0.0	0.0	0.3	0.1
144	2024-04-01	NaN	NaN	NaN	0.0	0.2	0.1	0.0	0.1	NaN	...	0.0	0.0	0.0	NaN	0.0	0.0	0.1	0.1
145	2024-05-01	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.0	NaN	...	NaN	0.0	NaN	NaN	NaN	NaN	NaN	NaN
146	2012-02-01	NaN	NaN	NaN	NaN	0.4	NaN	NaN	0.0	NaN	...	NaN	NaN	NaN	0.0	NaN	NaN	0.0	NaN

147 rows × 195 columns

In [23]:
```
h2=[]
for x in range(1,195,1):
    a="capacity"
    b=str(x)
    c=a+b
    h2.append(c)
print(h2)
```

```
['capacity1', 'capacity2', 'capacity3', 'capacity4', 'capacity5', 'capacity6', 'capacity7', 'capacity8', 'capacity9', 'capacity10', 'capacity11', 'capacity12', 'capacity13', 'capacity14', 'capacity1
5', 'capacity16', 'capacity17', 'capacity18', 'capacity19', 'capacity20', 'capacity21', 'capacity22', 'capacity23', 'capacity24', 'capacity25', 'capacity26', 'capacity27', 'capacity28', 'capacity29',
'capacity30', 'capacity31', 'capacity32', 'capacity33', 'capacity34', 'capacity35', 'capacity36', 'capacity37', 'capacity38', 'capacity39', 'capacity40', 'capacity41', 'capacity42', 'capacity43', 'ca
pacity44', 'capacity45', 'capacity46', 'capacity47', 'capacity48', 'capacity49', 'capacity50', 'capacity51', 'capacity52', 'capacity53', 'capacity54', 'capacity55', 'capacity56', 'capacity57', 'capac
ity58', 'capacity59', 'capacity60', 'capacity61', 'capacity62', 'capacity63', 'capacity64', 'capacity65', 'capacity66', 'capacity67', 'capacity68', 'capacity69', 'capacity70', 'capacity71', 'capacity
72', 'capacity73', 'capacity74', 'capacity75', 'capacity76', 'capacity77', 'capacity78', 'capacity79', 'capacity80', 'capacity81', 'capacity82', 'capacity83', 'capacity84', 'capacity85', 'capacity8
6', 'capacity87', 'capacity88', 'capacity89', 'capacity90', 'capacity91', 'capacity92', 'capacity93', 'capacity94', 'capacity95', 'capacity96', 'capacity97', 'capacity98', 'capacity99', 'capacity10
0', 'capacity101', 'capacity102', 'capacity103', 'capacity104', 'capacity105', 'capacity106', 'capacity107', 'capacity108', 'capacity109', 'capacity110', 'capacity111', 'capacity112', 'capacity113',
'capacity114', 'capacity115', 'capacity116', 'capacity117', 'capacity118', 'capacity119', 'capacity120', 'capacity121', 'capacity122', 'capacity123', 'capacity124', 'capacity125', 'capacity126', 'cap
acity127', 'capacity128', 'capacity129', 'capacity130', 'capacity131', 'capacity132', 'capacity133', 'capacity134', 'capacity135', 'capacity136', 'capacity137', 'capacity138', 'capacity139', 'capacit
y140', 'capacity141', 'capacity142', 'capacity143', 'capacity144', 'capacity145', 'capacity146', 'capacity147', 'capacity148', 'capacity149', 'capacity150', 'capacity151', 'capacity152', 'capacity15
3', 'capacity154', 'capacity155', 'capacity156', 'capacity157', 'capacity158', 'capacity159', 'capacity160', 'capacity161', 'capacity162', 'capacity163', 'capacity164', 'capacity165', 'capacity166',
'capacity167', 'capacity168', 'capacity169', 'capacity170', 'capacity171', 'capacity172', 'capacity173', 'capacity174', 'capacity175', 'capacity176', 'capacity177', 'capacity178', 'capacity179', 'cap
acity180', 'capacity181', 'capacity182', 'capacity183', 'capacity184', 'capacity185', 'capacity186', 'capacity187', 'capacity188', 'capacity189', 'capacity190', 'capacity191', 'capacity192', 'capacit
y193', 'capacity194']
```

In [24]:
```
ok4=ok3.drop("aa",axis="columns")
ok4
```

	capacity	capacity	capacity	capacity	capacity	capacity	capacity	capacity	capacity	capacity	...	capacity	capacity	capacity	capacity	capacity	capacity	capacity	capaci
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.0	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
142	NaN	NaN	NaN	0.0	0.0	0.0	0.2	0.2	0.0	0.0	...	0.0	0.2	0.0	0.0	0.1	0.1	0.5	0.1
143	NaN	NaN	NaN	0.0	0.4	0.1	0.2	0.2	0.0	NaN	...	0.0	0.0	0.0	NaN	0.0	0.0	0.3	0.1
144	NaN	NaN	NaN	0.0	0.2	0.1	0.0	0.1	NaN	NaN	...	0.0	0.0	0.0	NaN	0.0	0.0	0.1	0.1
145	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.0	NaN	NaN	...	NaN	0.0	NaN	NaN	NaN	NaN	NaN	NaN
146	NaN	NaN	NaN	NaN	0.4	NaN	NaN	0.0	NaN	NaN	...	NaN	NaN	NaN	0.0	NaN	NaN	0.0	NaN

147 rows × 194 columns

In [25]:

```python
ok4.columns=h2
ok4
```

	capacity1	capacity2	capacity3	capacity4	capacity5	capacity6	capacity7	capacity8	capacity9	capacity10	...	capacity185	capacity186	capacity187	capacity188	capacity18
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.0	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
142	NaN	NaN	NaN	0.0	0.0	0.0	0.2	0.2	0.0	0.0	...	0.0	0.2	0.0	0.0	0.1
143	NaN	NaN	NaN	0.0	0.4	0.1	0.2	0.2	0.0	NaN	...	0.0	0.0	0.0	NaN	0.0
144	NaN	NaN	NaN	0.0	0.2	0.1	0.0	0.1	NaN	NaN	...	0.0	0.0	0.0	NaN	0.0
145	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.0	NaN	NaN	...	NaN	0.0	NaN	NaN	NaN
146	NaN	NaN	NaN	NaN	0.4	NaN	NaN	0.0	NaN	NaN	...	NaN	NaN	NaN	0.0	NaN

147 rows × 194 columns

In [26]:

```python
gg2=pd.concat([yabba3,ok4],axis="columns")
gg2.to_csv(r"D:\uvakai\Lakes\full capacity renamed.csv")
```

In [27]:

```python
gg2
```

	aa	capacity1	capacity2	capacity3	capacity4	capacity5	capacity6	capacity7	capacity8	capacity9	...	capacity185	capacity186	capacity187	capacity188	capacity189	capacity190
0	2012-04-01	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.0	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
1	2012-05-01	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
2	2012-06-01	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
3	2012-07-01	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
4	2012-08-01	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
142	2024-02-01	NaN	NaN	NaN	0.0	0.0	0.0	0.2	0.2	0.0	...	0.0	0.2	0.0	0.0	0.1	0.1
143	2024-03-01	NaN	NaN	NaN	0.0	0.4	0.1	0.2	0.2	0.0	...	0.0	0.0	0.0	NaN	0.0	0.0
144	2024-04-01	NaN	NaN	NaN	0.0	0.2	0.1	0.0	0.1	NaN	...	0.0	0.0	0.0	NaN	0.0	0.0
145	2024-05-01	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.0	NaN	...	NaN	0.0	NaN	NaN	NaN	NaN
146	2012-02-01	NaN	NaN	NaN	NaN	0.4	NaN	NaN	0.0	NaN	...	NaN	NaN	NaN	0.0	NaN	NaN

147 rows × 195 columns

```python
love1=[]
for x in gg2["capacity1"]:
    love1.append(x)
print(love1)
print()
print(max(love1))
print(type(love1[2]))
```

```
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan,
nan, nan, nan, nan, nan, nan, nan, nan, nan, 0.0, nan, nan, nan, nan, 0.1, nan, nan, nan, nan, nan, nan, nan, nan, nan, 0.0, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, n
an, nan, nan, nan, nan, nan, nan, nan, 0.0, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, 0.0, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, 0.
0, nan, nan, nan, nan, nan, 0.0, nan, nan, nan, nan, nan]

nan
<class 'float'>
```

In [29]:

```python
love2=[]
for x in love1:
    aa=x/max(love1)
    bb=aa*100
    love2.append(bb)
print(love2)
max(love2)
```

```
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan,
nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, n
an, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, na
n, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan]

nan
```

In [31]:

```python
for x in range(1,195,1):
    globals()[f'oolove{x}']=[]
for x in range(1,195,1):
    globals()[f'pplove{x}']=[]
for x in range(1,195,1):
    globals()[f'tiger{x}']=[]
```

In [32]:

```python
for x in range(1,195,1):
    for y in gg2[h2[x-1]]:
        globals()[f'oolove{x}'].append(y)
```

In [33]:

```python
for x in range(1,195,1):
    for y in globals()[f'oolove{x}']:
        if y==y:
            globals()[f'tiger{x}'].append(y)
```

In [34]:
```
for x in range(1,195,1):
    for y in globals()[f'oolove{x}']:
        qq=sum(globals()[f'tiger{x}'])
        if qq==0.0:
            globals()[f'pplove{x}'].append(y)
        else:
            aa=y/max(globals()[f'tiger{x}'])
            bb=aa*100
            if bb>=0 and bb<=20:
                globals()[f'pplove{x}'].append("0-20%")
            elif bb>20 and bb<=40:
                globals()[f'pplove{x}'].append("20-40%")
            elif bb>40 and bb<=60:
                globals()[f'pplove{x}'].append("40-60%")
            elif bb>60 and bb<=80:
                globals()[f'pplove{x}'].append("60-80%")
            elif bb>80 and bb<=100:
                globals()[f'pplove{x}'].append("80-100%")
            else:
                globals()[f'pplove{x}'].append(float('nan'))
```

In [35]:
```
joshwin=[]
for x in range(1,195,1):
    a="lake"
    b=str(x)
    c=a+b
    joshwin.append(c)
print(joshwin)
```

```
['lake1', 'lake2', 'lake3', 'lake4', 'lake5', 'lake6', 'lake7', 'lake8', 'lake9', 'lake10', 'lake11', 'lake12', 'lake13', 'lake14', 'lake15', 'lake16', 'lake17', 'lake18', 'lake19', 'lake20', 'lake21', 'lake22', 'lake23', 'lake24', 'lake25', 'lake26', 'lake27', 'lake28', 'lake29', 'lake30', 'lake31', 'lake32', 'lake33', 'lake34', 'lake35', 'lake36', 'lake37', 'lake38', 'lake39', 'lake40', 'lake41', 'lake42', 'lake43', 'lake44', 'lake45', 'lake46', 'lake47', 'lake48', 'lake49', 'lake50', 'lake51', 'lake52', 'lake53', 'lake54', 'lake55', 'lake56', 'lake57', 'lake58', 'lake59', 'lake60', 'lake61', 'lake62', 'lake63', 'lake64', 'lake65', 'lake66', 'lake67', 'lake68', 'lake69', 'lake70', 'lake71', 'lake72', 'lake73', 'lake74', 'lake75', 'lake76', 'lake77', 'lake78', 'lake79', 'lake80', 'lake81', 'lake82', 'lake83', 'lake84', 'lake85', 'lake86', 'lake87', 'lake88', 'lake89', 'lake90', 'lake91', 'lake92', 'lake93', 'lake94', 'lake95', 'lake96', 'lake97', 'lake98', 'lake99', 'lake100', 'lake101', 'lake102', 'lake103', 'lake104', 'lake105', 'lake106', 'lake107', 'lake108', 'lake109', 'lake110', 'lake111', 'lake112', 'lake113', 'lake114', 'lake115', 'lake116', 'lake117', 'lake118', 'lake119', 'lake120', 'lake121', 'lake122', 'lake123', 'lake124', 'lake125', 'lake126', 'lake127', 'lake128', 'lake129', 'lake130', 'lake131', 'lake132', 'lake133', 'lake134', 'lake135', 'lake136', 'lake137', 'lake138', 'lake139', 'lake140', 'lake141', 'lake142', 'lake143', 'lake144', 'lake145', 'lake146', 'lake147', 'lake148', 'lake149', 'lake150', 'lake151', 'lake152', 'lake153', 'lake154', 'lake155', 'lake156', 'lake157', 'lake158', 'lake159', 'lake160', 'lake161', 'lake162', 'lake163', 'lake164', 'lake165', 'lake166', 'lake167', 'lake168', 'lake169', 'lake170', 'lake171', 'lake172', 'lake173', 'lake174', 'lake175', 'lake176', 'lake177', 'lake178', 'lake179', 'lake180', 'lake181', 'lake182', 'lake183', 'lake184', 'lake185', 'lake186', 'lake187', 'lake188', 'lake189', 'lake190', 'lake191', 'lake192', 'lake193', 'lake194']
```

In [36]:
```
for x in range(1,195,1):
    globals()[f'dictt{x}']={joshwin[x-1]:globals()[f'pplove{x}']}
```

In [37]:
```
for x in range(1,195,1):
    globals()[f'dataframebro{x}']=pd.DataFrame(globals()[f'dictt{x}'])
```

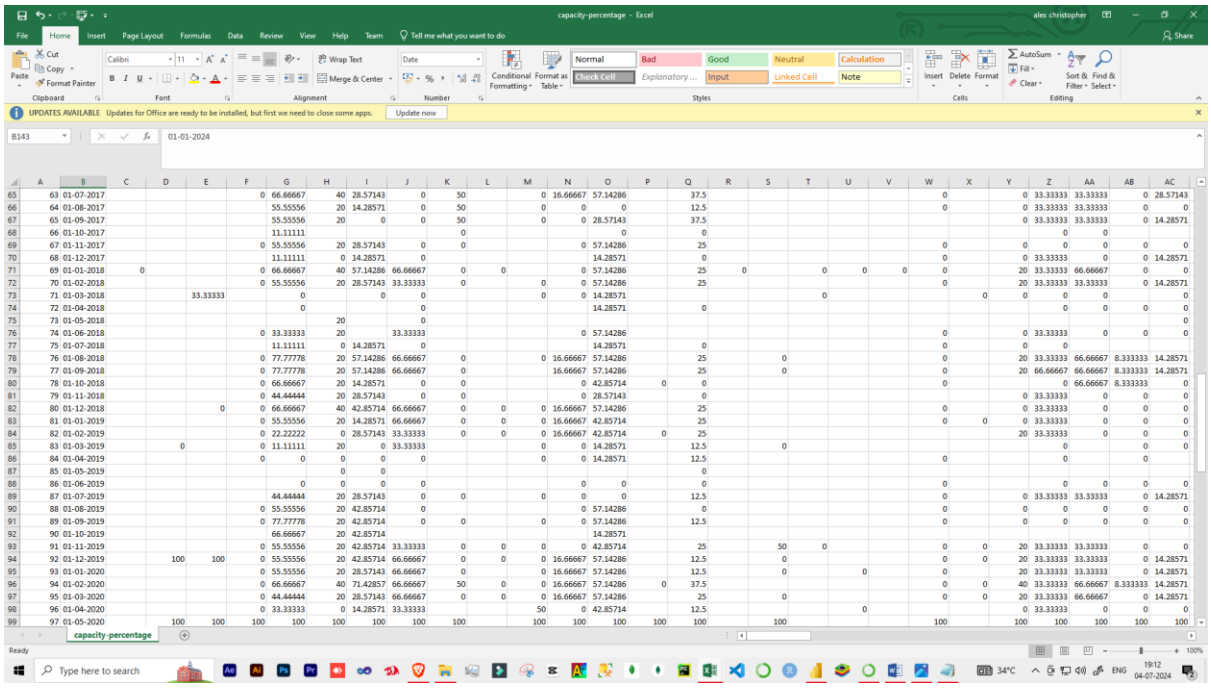
In [38]:
```
siu=[]
for x in range(1,195,1):
    a="dataframebro"
    b=str(x)
    c=a+b
    siu.append(c)
tty=','.join(siu)
print(tty)
```

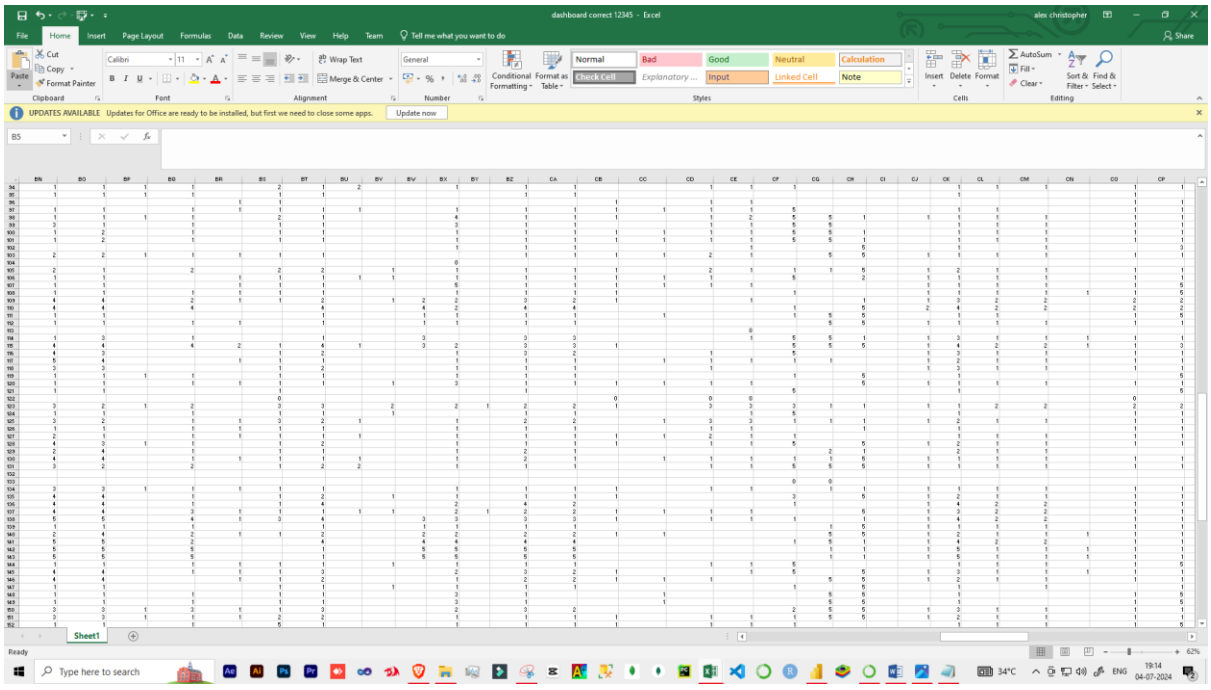
```
dataframebro1,dataframebro2,dataframebro3,dataframebro4,dataframebro5,dataframebro6,dataframebro7,dataframebro8,dataframebro9,dataframebro10,dataframebro11,dataframebro12,dataframebro13,dataframebro14,dataframebro15,dataframebro16,dataframebro17,dataframebro18,dataframebro19,dataframebro20,dataframebro21,dataframebro22,dataframebro23,dataframebro24,dataframebro25,dataframebro26,dataframebro27,dataframebro28,dataframebro29,dataframebro30,dataframebro31,dataframebro32,dataframebro33,dataframebro34,dataframebro35,dataframebro36,dataframebro37,dataframebro38,dataframebro39,dataframebro40,dataframebro41,dataframebro42,dataframebro43,dataframebro44,dataframebro45,dataframebro46,dataframebro47,dataframebro48,dataframebro49,dataframebro50,dataframebro51,dataframebro52,dataframebro53,dataframebro54,dataframebro55,dataframebro56,dataframebro57,dataframebro58,dataframebro59,dataframebro60,dataframebro61,dataframebro62,dataframebro63,dataframebro64,dataframebro65,dataframebro66,dataframebro67,dataframebro68,dataframebro69,dataframebro70,dataframebro71,dataframebro72,dataframebro73,dataframebro74,dataframebro75,dataframebro76,dataframebro77,dataframebro78,dataframebro79,dataframebro80,dataframebro81,dataframebro82,dataframebro83,dataframebro84,dataframebro85,dataframebro86,dataframebro87,dataframebro88,dataframebro89,dataframebro90,dataframebro91,dataframebro92,dataframebro93,dataframebro94,dataframebro95,dataframebro96,dataframebro97,dataframebro98,dataframebro99,dataframebro100,dataframebro101,dataframebro102,dataframebro103,dataframebro104,dataframebro105,dataframebro106,dataframebro107,dataframebro108,dataframebro109,dataframebro110,dataframebro111,dataframebro112,dataframebro113,dataframebro114,dataframebro115,dataframebro116,dataframebro117,dataframebro118,dataframebro119,dataframebro120,dataframebro121,dataframebro122,dataframebro123,dataframebro124,dataframebro125,dataframebro126,dataframebro127,dataframebro128,dataframebro129,dataframebro130,dataframebro131,dataframebro132,dataframebro133,dataframebro13
```

In [39]:
```
vij=pd.concat([dataframebro1,dataframebro2,dataframebro3,dataframebro4,dataframebro5,dataframebro6,dataframebro7,dataframebro8,dataframebro9,dataframe
vij
```

	lake1	lake2	lake3	lake4	lake5	lake6	lake7	lake8	lake9	lake10	...	lake185	lake186	lake187	lake188	lake189	lake190	lake191	lake192	lake193	lake194
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0-20%	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
142	NaN	NaN	NaN	0-20%	0-20%	0-20%	20-40%	60-80%	0-20%	0.0	...	0-20%	20-40%	0-20%	0-20%	20-40%	0-20%	0-20%	40-60%	NaN	0-20%
143	NaN	NaN	NaN	0-20%	40-60%	0-20%	20-40%	60-80%	0-20%	NaN	...	0-20%	0-20%	0-20%	NaN	0-20%	0-20%	0-20%	40-60%	NaN	0-20%
144	NaN	NaN	NaN	0-20%	20-40%	0-20%	0-20%	20-40%	NaN	NaN	...	0-20%	0-20%	0-20%	NaN	0-20%	0-20%	0-20%	40-60%	NaN	NaN
145	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0-20%	NaN	NaN	...	NaN	0-20%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
146	NaN	NaN	NaN	NaN	40-60%	NaN	NaN	0-20%	NaN	NaN	...	NaN	NaN	NaN	0-20%	NaN	NaN	0-20%	NaN	NaN	NaN

```
In [40]:
wsafinal=pd.concat([yabba3,vij],axis="columns")
wsafinal.to_csv(r"D:\uvakai\Lakes verified\dg.csv")
```

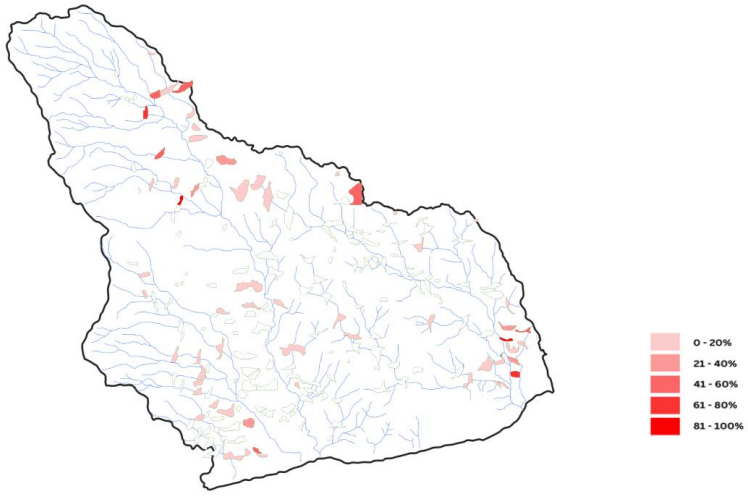




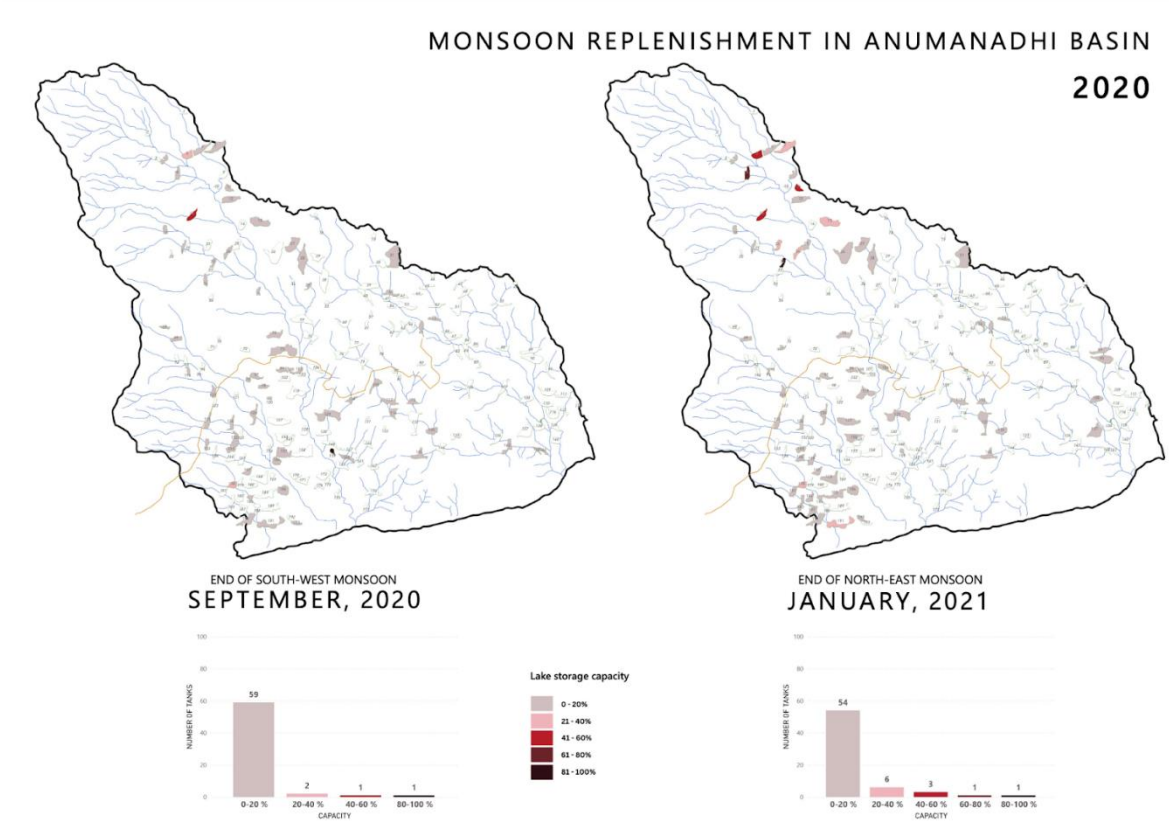
Data Visualization

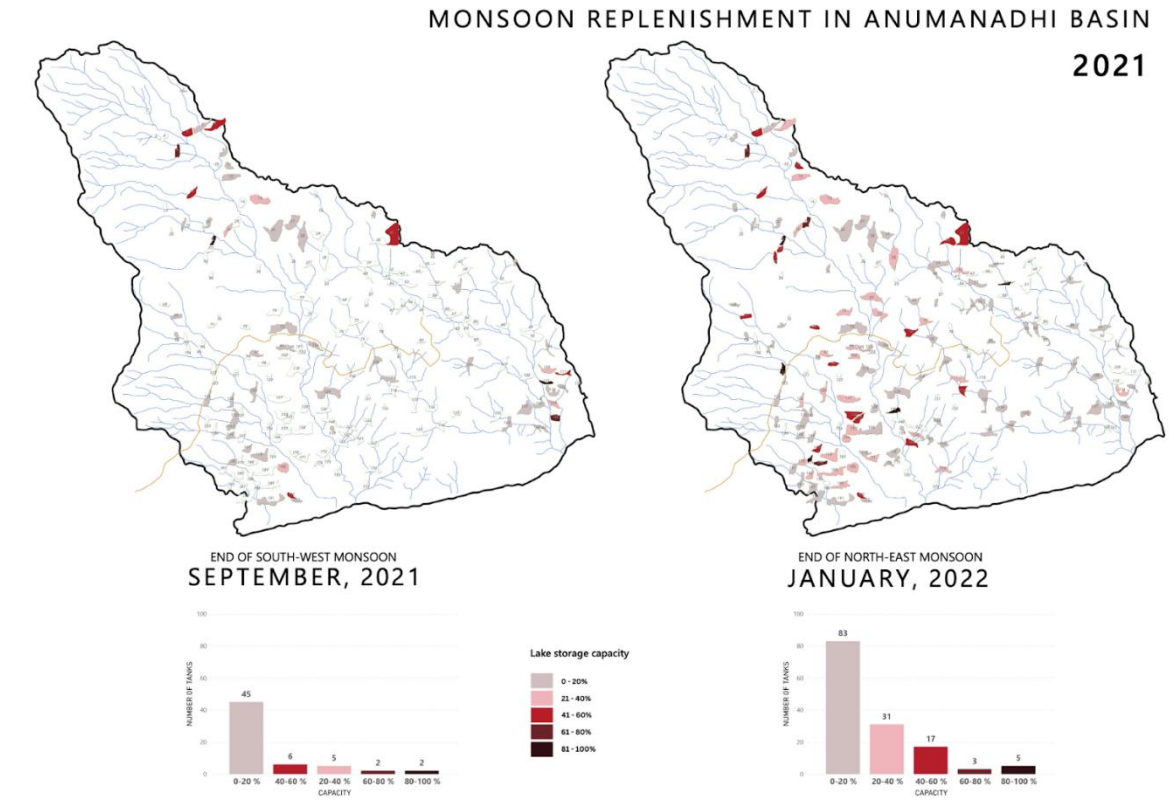
Data visualization played a crucial role in interpreting and presenting the findings from our analysis of the Radhapuram and Anumanadhi Canal systems. We created three primary visualizations to effectively communicate the data:

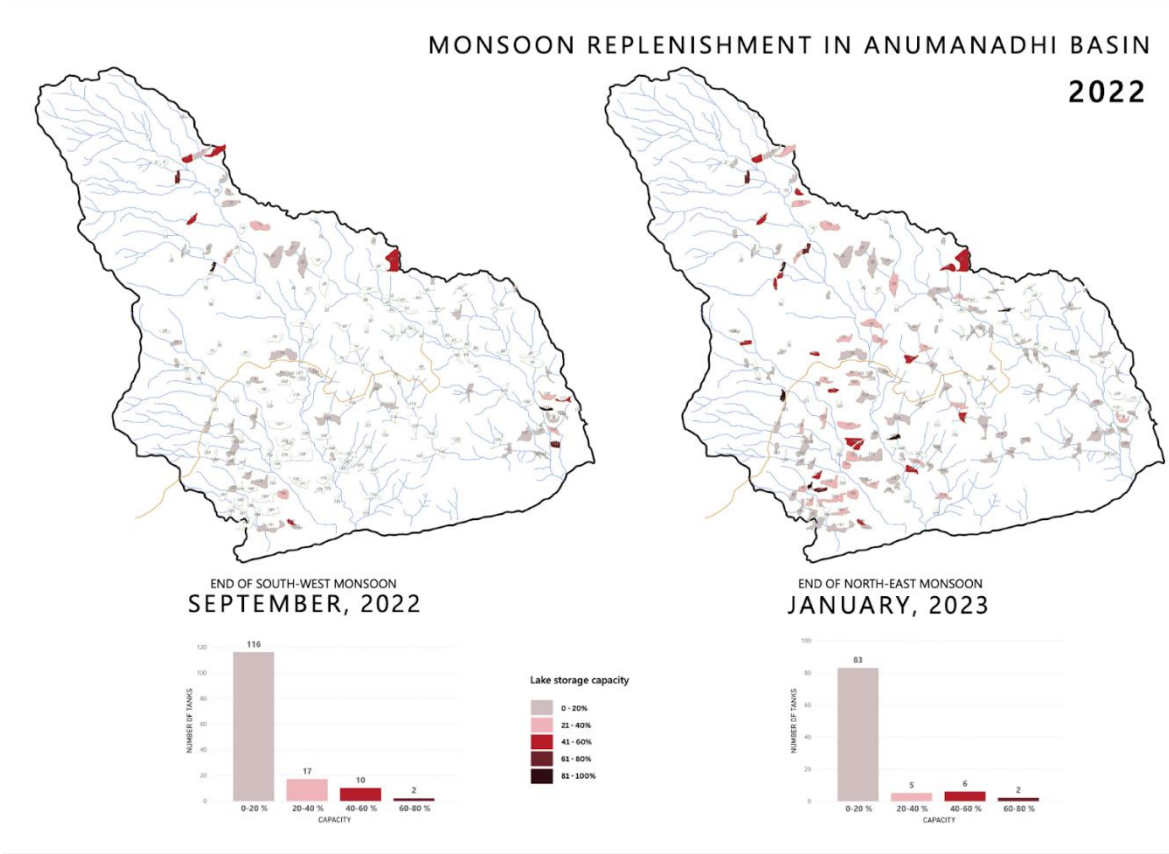
Capacity Flow Visualization: This visualization represents the capacity flow of each lake using categorical data with five different saturations of red colour. High red saturation (80-100%) indicates lakes with high capacity, while very pale red saturation (0-20%) indicates lakes with low capacity. Intermediate colours represent the capacity levels between these extremes. This visual representation allows users to easily identify the flow and capacity between different lakes from 2012 to 2024, helping to spot any potential issues or blocks in the system.

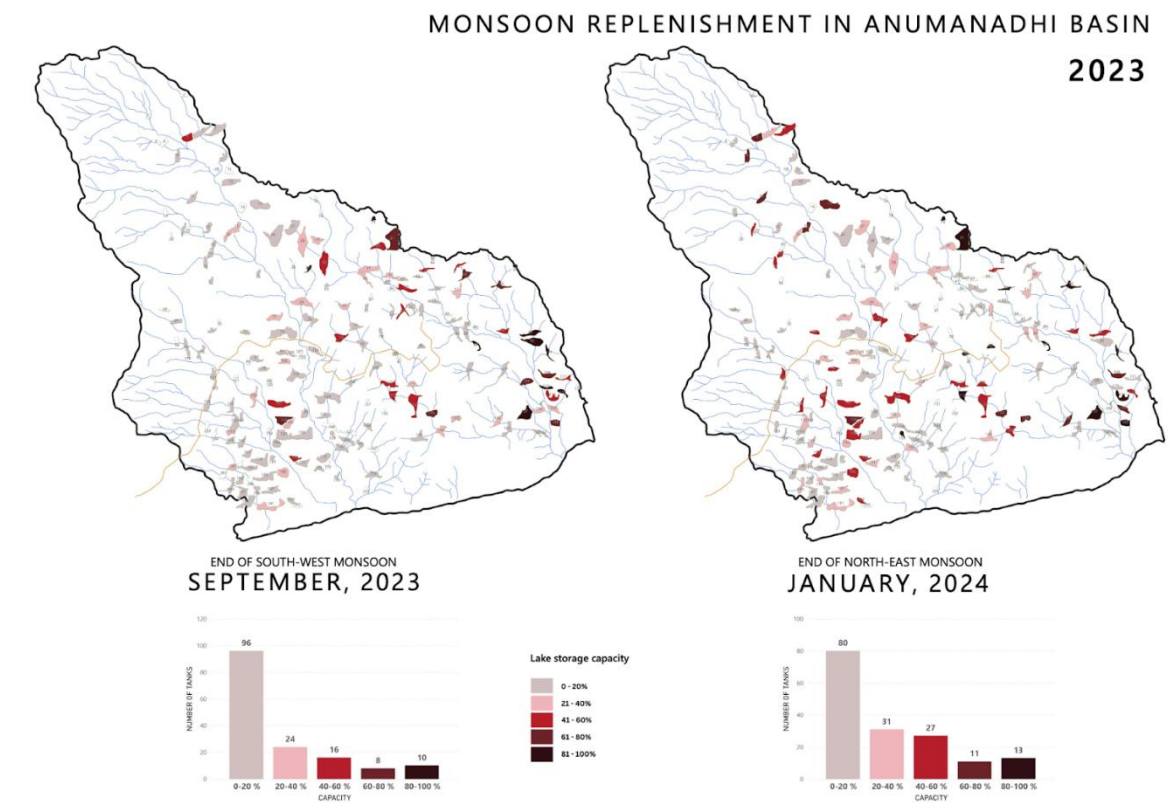


Monsoon Capacity Visualization: This visualization focuses on the capacity data of lakes during two monsoon seasons: southwest and northeast. We assumed September for the southwest monsoon and January for the northeast monsoon. This approach helps in understanding the impact of each monsoon season on the water levels of the lakes. The capacities are visualized using colours in a categorical manner, facilitating easy interpretation of the data across different monsoon periods.

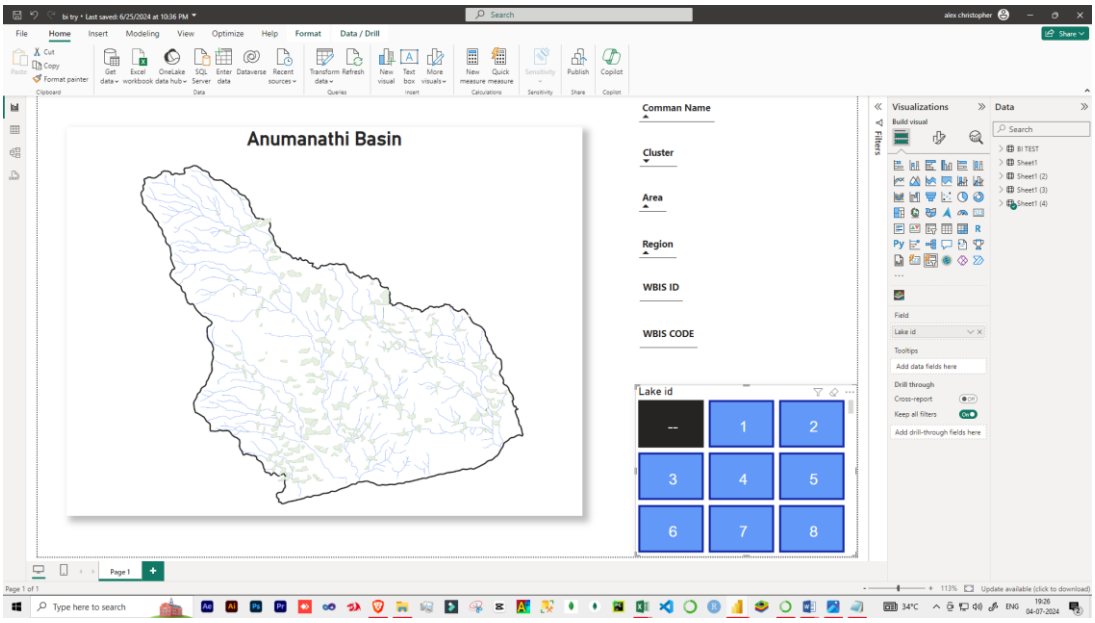

MONSOON REPLENISHMENT IN ANUMANADHI BASIN
2020
END OF SOUTH-WEST MONSOON
SEPTEMBER, 2020
END OF NORTH-EAST MONSOON
JANUARY, 2021
Lake storage capacity
0 - 20%
21 - 40%
41 - 60%
61 - 80%
81 - 100%
NUMBER OF TANKS
CAPACITY
0-20 %
20-40 %
40-60 %
60-80 %
80-100 %
59
2
1
1
54
6
3
1
1

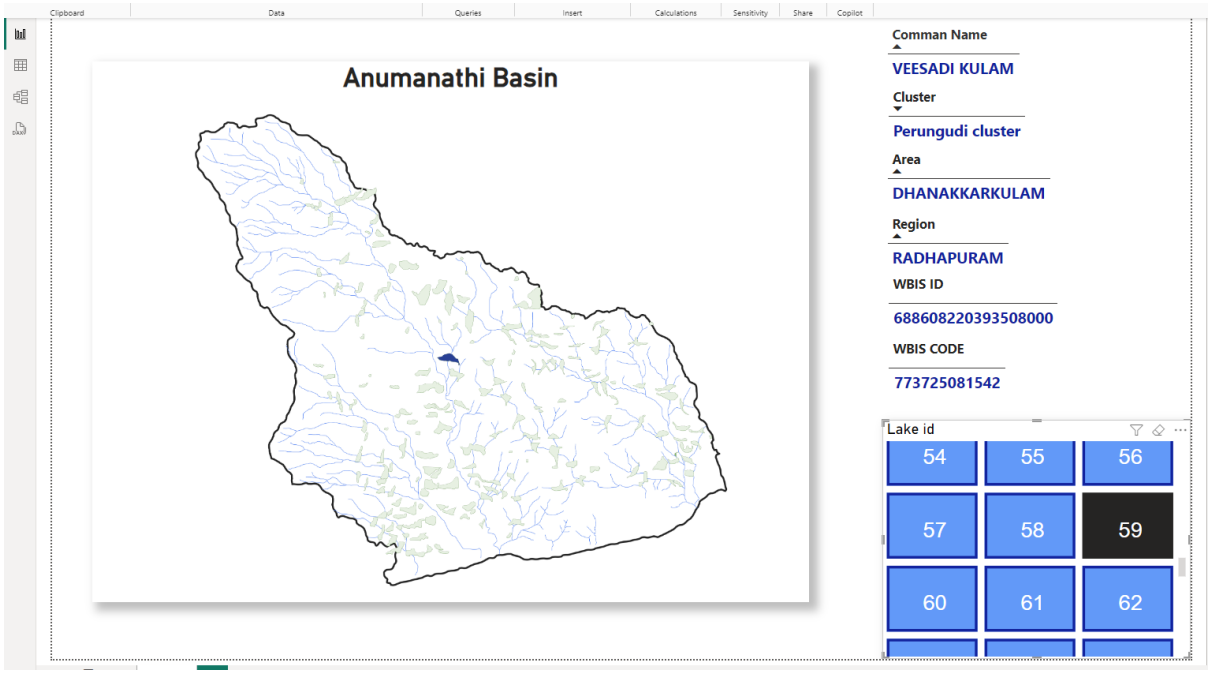

MONSOON REPLENISHMENT IN ANUMANADHI BASIN
2021
END OF SOUTH-WEST MONSOON
SEPTEMBER, 2021
END OF NORTH-EAST MONSOON
JANUARY, 2022
Lake storage capacity
0 - 20%
21 - 40%
41 - 60%
61 - 80%
81 - 100%
NUMBER OF TANKS
CAPACITY
0-20 %
40-60 %
20-40 %
60-80 %
80-100 %
45
6
5
2
2
83
31
17
3
5


MONSOON REPLENISHMENT IN ANUMANADHI BASIN
2022
END OF SOUTH-WEST MONSOON
SEPTEMBER, 2022
END OF NORTH-EAST MONSOON
JANUARY, 2023
NUMBER OF TANKS
116
17
10
2
0-20 %
20-40 %
40-60 %
60-80 %
CAPACITY
Lake storage capacity
0 - 20%
21 - 40%
41 - 60%
61 - 80%
81 - 100%
83
5
6
2


MONSOON REPLENISHMENT IN ANUMANADHI BASIN
2023
END OF SOUTH-WEST MONSOON
SEPTEMBER, 2023
END OF NORTH-EAST MONSOON
JANUARY, 2024
NUMBER OF TANKS
96
24
16
8
10
0-20 %
20-40 %
40-60 %
60-80 %
80-100 %
CAPACITY
Lake storage capacity
0 - 20%
21 - 40%
41 - 60%
61 - 80%
81 - 100%
80
31
27
11
13

Power BI Dashboard: The Power BI dashboard for the Radhapuram and Anumanadhi canal project displays an interactive map of 194 lakes, allowing users to select and highlight lakes for detailed examination. Alongside, five tables provide crucial lake details: name, WBIS code, WBIS ID, area, and cluster name derived from TWAD analysis. This setup facilitates efficient data exploration and informed decision-making on water resource management. The dashboard's user-friendly interface and comprehensive data visualization enhance accessibility, supporting stakeholders in monitoring and optimizing the canal system's water flow and capacity dynamics effectively.





Reference

Water Body Information System (Bhuvan)

https://bhuvan-wbis.nrsc.gov.in/

Nellai Neer Valam

https://nellaineervalam.in/

Tamilnadu Water Supply and Drainage Board

https://www.twadboard.tn.gov.in/

Certificate

CERTIFICATE

OF INTERNSHIP COMPLETION

This is to certify that

Alex Christopher M

a student of M.Sc Data Science (Batch 2023-2025) at Madras Christian College, has successfully completed the internship as a Data Analyst, at Uvakai Research Foundation from 13 May 2024 to 21 June 2024.

During this period, he participated in a group project.titled

'Mapping and Analyzing Waterbodies within Anumanadhi Sub-basin and Visualizing Water Status Patterns Over a Decade'.

Ref. No.: UI _01_2024_08

UVAKAI RESEARCH FOUNDATION

Udhaya Rajan N

Director

EMPOWERING HAPPINESS, INSPIRING CHANGE.

Conclusion

This internship's analysis of the Radhapuram and Anumanadhi canal system used Python for data cleaning and integration, Power BI for visualization, and Synoptic Designer for custom map creation. Key findings included insights into capacity flow, monsoon impacts, and lake clustering, crucial for water management decisions. The Power BI dashboard provided interactive insights with detailed lake information, supporting effective resource allocation and infrastructure planning. Overall, the project contributes to improving understanding and management of the canal system's water dynamics and capacity utilization.