**Lab Report**

Title: APIs: Conceptual Models and Application

Notice: Dr. Bryan Runck

Author: Jake Ford

Date: 10-03-2022

**Project Repository:**https://github.com/ThisFord/GIS5571-arc1.git

**Google Drive Link:**

Time Spent: 30+hours

**Abstract**

Understanding APIs and implementing them in an ETL pipeline is an essential task in GIS. Automation and reproducibility are two features of ETLs that dramatically improve efficiency in GIS workflows. This lab compares and contrasts three APIs with different formats and interfaces by constructing informal conceptual models. The lab then demonstrates the difficulties and advantages presented in the process by constructing custom ETLs for each API, by downloading two data sets through each interface, performing cleaning and spatial transformations on the data, and loading it into a geodatabse in Esri’s ArcPro software using the Jupyter Notebooks interface in Arc Pro. The results are based on user experience, with varying levels of success using the APIs in a custom ETL, the formats of the data varied widely, from raw csv tables in NDAWN, to more readymade shapefiles with MN Geospatial Commons, to nicely packages GeoJSON objects through Google Places. Overall the complexity and difficulty working directly with an API and an ETL environment is outweighed by the advantages presented by programmatic automation and the reproducibility of established workflows.

**Problem Statement**

Application Programming Interfaces are useful tools for programmers, the interfaces make it possible to automate tasks and queries programmatically, they allow all the actions of a website interface and more. Understanding how APIs work and developing experience in their use is essential for many GIS tasks. (*The CKAN API — CKAN Documentation 2.1.5 Documentation*, n.d.; *What Is an Application Programming Interface (API)?*, 2022) This lab strengthens understanding by comparing and contrasting three APIs, using informal conceptual models and custom built Extract Transform and Load (ETL) routines derived from user interface; and deepens experience by building an ETL pipeline with Open Source Tools in Jupyter Notebooks in Arc Pro and Arc Online platforms, fetching data and and performing a spatial join.(*What Is ETL (Extract, Transform, Load)?*, 2022)

*Diagram

Description automatically generated.*

*Flowchart of an API in the context of GIS*

Diagram, schematic

Description automatically generated

An example ETL with spatial transformations

*Table 1. API comparison and use requirements*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **#** | **Requirement** | **Defined As** | **(Spatial) Data** | **Attribute Data** | **Dataset** | **Preparation** |
| 1 | Internet connection | Ability to access information on the web |  |  |  |  |
| 2 | Target databases | the information you’re trying to access programmatically |  |  |  |  |
| 3 | Jupyter Notebook in ArcPro | Python programming interface in esri’s arcpro software |  |  |  |  |
| 4 | Jupyter Notebook in Arc Online | Python programming interface in esri’s online version of ArcGIS |  |  |  |  |
| 5 | Visual model builder | A graphic representation of your informal models |  |  |  |  |

**Input Data**

The input data is largely user generated, as experience and usability is key to the compare and contrast method of analysis. The target data we’re trying to access is relatively arbitrary, as the experience is derived from mapping the APIs and using the ETL pipeline. In the case of this study three APIs are used, the Minnesota Geospatial Commons, Google Places, and NDAWN. The data gathered differs for each API, the data listed in table 2 are the datasets used in the fetch and join operations.

Table 2. data collected from the APIs

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Title** | **Purpose in Analysis** | **Link to Source** |
| 1 | MN Stream Gauges Location | Demonstrate ETL capability | [here](https://resources.gisdata.mn.gov/pub/gdrs/data/pub/us_mn_state_dnr/env_wiski_coop_stream_gaging/shp_env_wiski_coop_stream_gaging.zip) |
| 2 | MN Superfund Sites | Demonstrate ETL capability | [here](https://resources.gisdata.mn.gov/pub/gdrs/data/pub/us_mn_state_pca/env_remediation_plp/shp_env_remediation_plp.zip) |
| 3 | NDAWN station 77 data max temp | Demonstrate ETL capability | NDAWNMAX |
| 4 | NDAWN station 77 Min temp | Demonstrate ETL capability | NDAWNMIN |
| 5 | Google place search nearby location | Demonstrate ETL capability | Google nearby |
| 6 | Google place search from text | Demonstrate ETL capability | Google from text |

**Methods**

1. Build Informal Conceptual Models
2. Test model comprehension with custom ETL
   1. Build ETL pipelines with Jupyter Notebooks
      1. downloads two data sets
      2. transform both datasets to the same [coordinate reference system](https://pro.arcgis.com/en/pro-app/latest/help/mapping/properties/coordinate-systems-and-projections.htm) (geographic and projected)
      3. spatial join them
      4. print to screen the head of the table showing the merged attributes
      5. save the integrated dataset to a geodatabase.

**Conceptual Models**

Conceptual models were difficult to frame. Using the opaque CKAN documentation the models below were constructed for the MN Geo API. Initially the flow and actual process behind an API and defining what an API is was difficult to conceptualize with minimal experience. The first iteration in figure 1 shows the initial understanding of the API based on some experimentation with the URL directly through the /api interface. (*API Developer Resources | Gisdata.Mn.Gov*, n.d.) This resulted in an interesting way of navigating the Geocommons with the direct CKAN commands, seen in figure 2 and 3. (*The CKAN API — CKAN Documentation 2.1.5 Documentation*, n.d.) Ultimately this was not a fruitful pursuit, as attempted get requests using the /api endpoint resulted in error codes pointing to ssh authentication problems. This might be the back end interface for the site’s programmers, not the direct user interface this lab is designed to have us explore. Further exploration using the inspector and direct GUI interface uncovered similar patters in the MN geo website as seen elsewhere across the web. Experimenting with NDAWN led to a more direct understanding of what the API can look like, with the url/function/parameters and queries pattern directly expressed, as in figure 4.

*Diagram

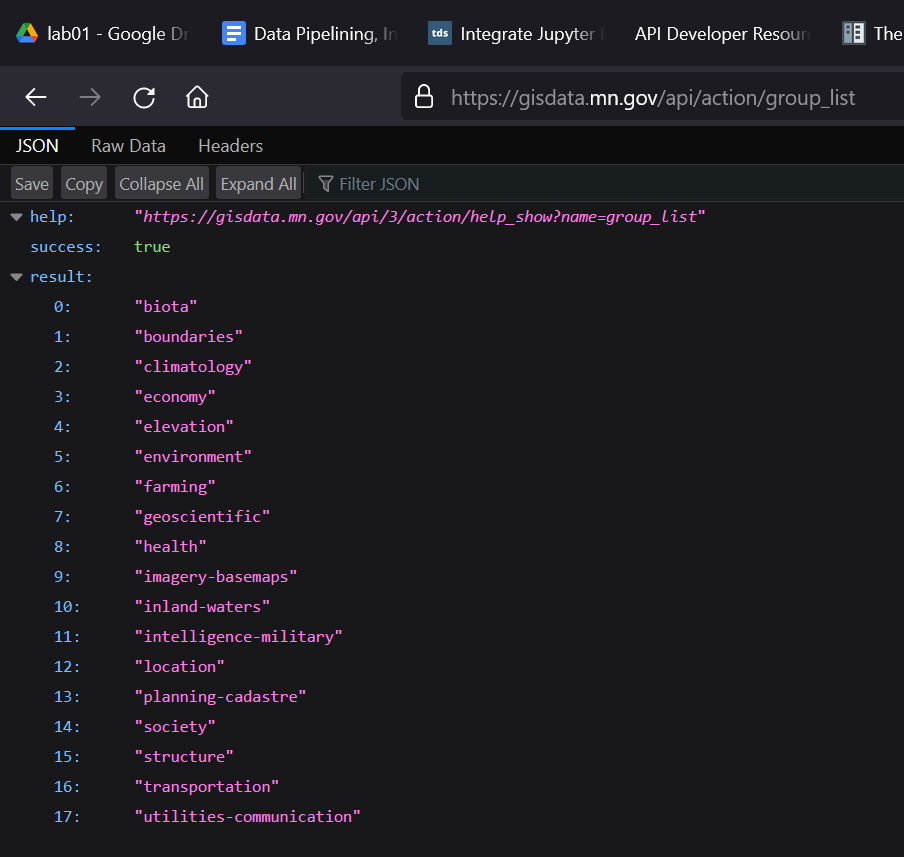
Description automatically generated*

*Figure 1, an early conceptual model of the MNGeo API*

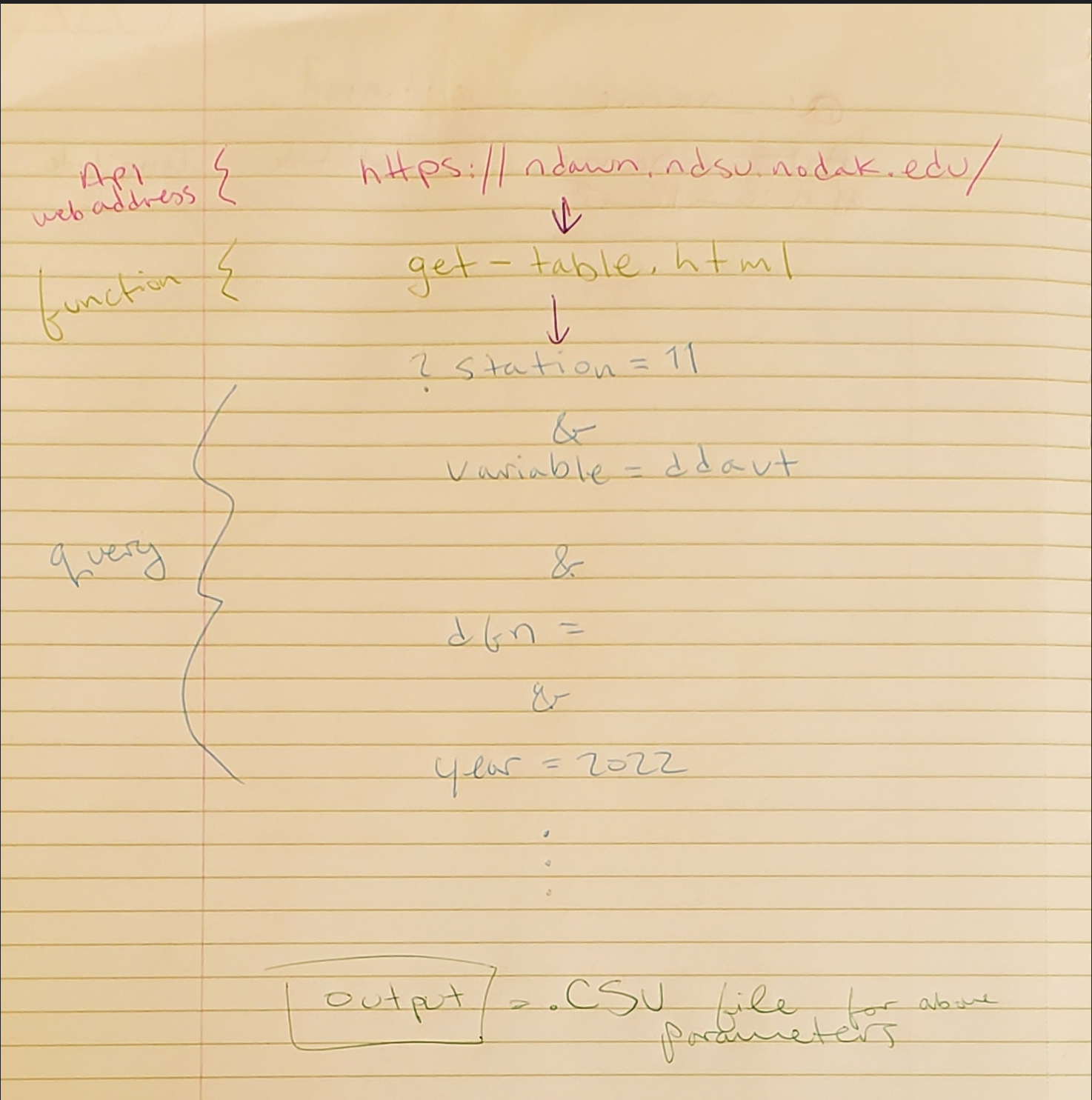
*Diagram

Description automatically generated*

*Figure 2, a revised abstracted conceptual model of the MNGeo API based on the CKAN Documentation*

**

*Figure 3, results from the CKAN API model*

**

*Figure 4, NDAWN API structure with actual parameters*

**Diagram

Description automatically generated**

***Figure 5* https://www.geeksforgeeks.org/get-post-requests-using-python/**(“GET and POST Requests Using Python,” 2016)

**Custom ETL**

Three custom Extract, Transform, Load programs were scripted in the Arc Pro Jupyter Notebooks environment to demonstrate the capabilities of using an API for data gathering and processing. First, the MNGeo ETL pipeline was built. The URL endpoints were used in combination with the request library in python to grab shapefiles, the arcpro library added the transformations, and pandas was used to display the resulting joined tables.

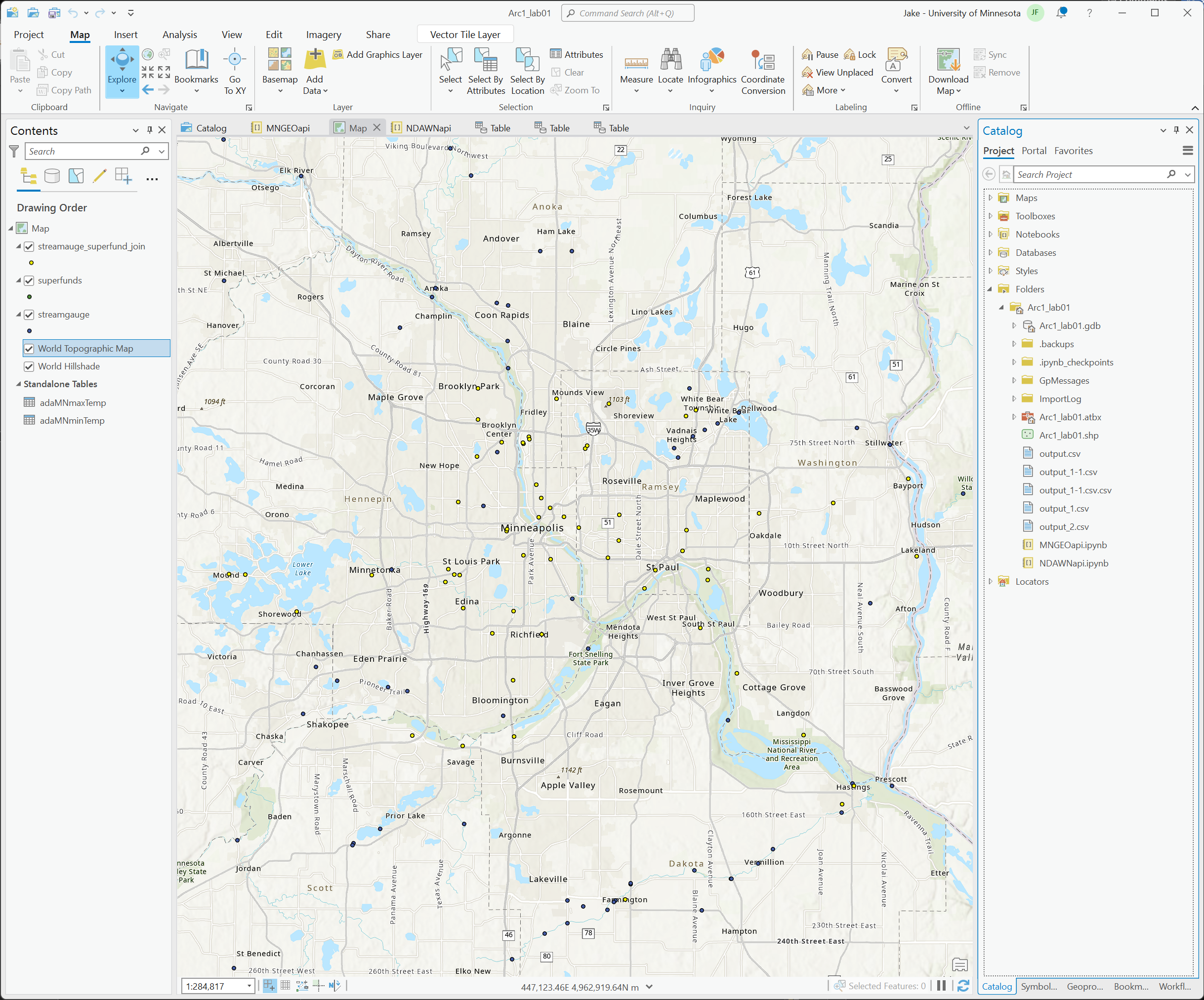
NDAWN presented unique challenges, with he requests and responses seeming to query a database wide dataset or csv as opposed to a nicely packaged shapefile. The NDAWN api required selecting parameters to pass into the request, to limit the responses to the desired ranges and values. This method conjures the image of a huge three dimensional database laid out with the z value as time steps. You select the coordinates of the data you want to retrieve by setting the parameters, almost like selecting a range in a numpy array. The remote database processes the request and returns a csv with the values of the ranges you selected for the input parameters, as seen in figure 4 above.

From here the csv needed to be cleaned, as the first rows of the csv presented formatting errors in pandas and ArcPro. Using the skiprow parameter I the pandas read\_csv function worked for many people in the lab, but for this instance the parameter didn’t seem to perform the assigned task. I tried using the iloc[:3] method to skip the first three rows but ran out of time to work on the issue.

The Google Places api was nicely documented, with code examples, but the challenge of setting up an api key made it very complex. I was able to establish a request but ran put of time to experiment with the transformation and loading process. Overall, the places api seemed powerful and direct with good documentation. This would have been a good place to start as it would have established a clearer framework of what an api is early on. (*Find Place | Places API | Google Developers*, n.d.)

**Results**

The MnGeo site provided the most straight forward use case. After initial stumbles with the API, the concepts became clearer and the data was accessed according to custom parameters, transformed into useable data for the ArcPro mapping interface and loaded successfully after the join and buffer operations. The data was readily mappable as in Figure 6. NDAWN presented formatting difficulties that were difficult to overcome. The transform and load procedures went well, but the data format was unusable, as the CSV wasn’t separating the values by the comma position. With time and experimentation, a solution is plausible. The problem has been identified, the headers in the loaded csv are collapsing all the columns into one, due to how pandas interpret the input. (Figure 7) I need to remove those rows but skip rows in the read\_csv function doesn’t seem to work. Alternatively I tried to start the file from a defined index with iloc[:3] as the given parameter but didn’t have success there either. With that said, working out this issue is the next step, as the actual loading and processing worked fine mechanically. Unfortunately, this study encountered many, many time delay issues and as a result the Google API was left very unfinished. The requests were initiated, and data retrieved, but I couldn’t perform any transformations or loading operations due to prior difficulties. (Figure 8) However, the complexity and difficulty working directly with an API and an ETL environment is outweighed by the advantages presented by programmatic automation and the reproducibility of established workflows.



*Figure 6, ArcPro with data from MNGeo mapped. The catalogue shows other imported CSVs from NDAWN*

**

*Figure 7, NDAWN Difficulties*

Graphical user interface, text, application, email

Description automatically generated

*Figure 8, The beginnings of the google API process*

**Results Verification**

The results from the compare and contrast are qualitative and somewhat subjective, accuracy of the assessments is tested against a survey of the experiences of the class and is reviewed by expert instructors. Results for the ETL implementation are a clear succeed/fail outcome, as each ETL must perform each step properly to output an integrated dataset to the geodatabase. Successful implementation of the ETL results in a working output to the geodatabase.

**Discussion and Conclusion**

While APIs and ETLs appear to be essential to the collection and analysis of data from around the web, their documentation and accessibility are opaque and difficult to implement. It appears that the standards and guidelines established for designing an API allow for wide variances in usability and accessibility. The process for extracting, transforming and loading data is highly specialized depending on the data source and the intended user application. Automating this process with an ETL pipeline is a worthwhile, though incredibly finicky, time consuming, and difficult task to perform. I have used geopandas in the past to do a lot of this type of transformation and processing and am very confused as to why the module is not available in ArcPro Notebooks.

**References**

*API Developer Resources | gisdata.mn.gov*. (n.d.). Retrieved October 3, 2022, from <https://gisdata.mn.gov/content/?q=help/api>

*DNR/MPCA Cooperative Stream Gaging Locations—Minnesota Geospatial Commons*. (n.d.). Retrieved October 4, 2022, from <https://gisdata.mn.gov/dataset/env-wiski-coop-stream-gaging>

*Essential notebook commands—ArcGIS Notebook Server | Documentation for ArcGIS Enterprise*. (n.d.). Retrieved October 4, 2022, from <https://enterprise.arcgis.com/en/notebook/latest/use/windows/essential-notebook-commands.htm>

*Find Place | Places API | Google Developers*. (n.d.). Retrieved October 9, 2022, from <https://developers.google.com/maps/documentation/places/web-service/search-find-place>

GET and POST requests using Python. (2016, December 7). *GeeksforGeeks*. <https://www.geeksforgeeks.org/get-post-requests-using-python/>

Gette, C. (2021). *Get Weather Data* [Python]. <https://github.com/gettecr/get_us_weather/blob/ebf1aeed77a4530b932909c3e844ef34b302860e/get_weather_ndawn.py> (Original work published 2018)

*How to add data to a feature class from pandas DataFrame using InsertCursor class?* (2019, November 1). Esri Community. <https://community.esri.com/t5/python-blog/how-to-add-data-to-a-feature-class-from-pandas/ba-p/883854>

Inthavong, A. (2020, December 21). Building a ETL pipeline. *Analytics Vidhya*. <https://medium.com/analytics-vidhya/building-a-etl-pipeline-226656a22f6d>

*MnGeo: Minnesota Geospatial Information Office*. (n.d.). MN IT Services. Retrieved October 3, 2022, from <https://www.mngeo.state.mn.us/>

Newbies, A. P. I. (2021, September 20). *API Requests*. API Newbies. <https://www.apinewbies.com/api-request/>

*Permanent List of Priorities (PLP)—Minnesota Geospatial Commons*. (n.d.). Retrieved October 4, 2022, from <https://gisdata.mn.gov/dataset/env-remediation-plp>

*The CKAN API — CKAN Documentation 2.1.5 documentation*. (n.d.-a). Retrieved October 2, 2022, from <https://docs.ckan.org/en/ckan-2.1.5/api.html>

Tobin, D. (n.d.). *The 6 Parts of ETL Data Pipeline Architecture*. Integrate.Io. Retrieved October 3, 2022, from <https://www.integrate.io/blog/etl-architecture-building-blocks/>

Waibel, X. (2020, March 6). *Integrate Jupyter Notebook into Data Pipelines*. Medium. <https://towardsdatascience.com/integrate-jupyter-into-your-data-pipeline-9a02fab3cee5>

*What is an Application Programming Interface (API)?* (2022, September 28). <https://www.ibm.com/cloud/learn/api>

*What is ETL (Extract, Transform, Load)?* (2022, July 5). <https://www.ibm.com/cloud/learn/etl>

**Self-score**

|  |  |  |  |
| --- | --- | --- | --- |
| **Category** | **Description** | **Points Possible** | **Score** |
| **Structural Elements** | All elements of a lab report are included **(2 points each)**:  Title, Notice: Dr. Bryan Runck, Author, Project Repository, Date, Abstract, Problem Statement, Input Data w/ tables, Methods w/ Data, Flow Diagrams, Results, Results Verification, Discussion and Conclusion, References in common format, Self-score | 28 | **28** |
| **Clarity of Content** | Each element above is executed at a professional level so that someone can understand the goal, data, methods, results, and their validity and implications in a 5-minute reading at a cursory-level, and in a 30 minute meeting at a deep level **(12 points)**. There is a clear connection from data to results to discussion and conclusion **(12 points)**. | 24 | **24** |
| **Reproducibility** | Results are completely reproducible by someone with basic GIS training. There is no ambiguity in data flow or rationale for data operations. Every step is documented and justified. | 28 | **28** |
| **Verification** | Results are correct in that they have been verified in comparison to some standard. The standard is clearly stated **(10 points)**, the method of comparison is clearly stated **(5 points)**, and the result of verification is clearly stated **(5 points)**. | 20 | **20** |
|  |  | 100 | **100** |