Spatial Database Design

Temple University

Dr. Lee Hachadoorian

Final Project Report

Fall 2024

Pennsylvania Fish and Boat Commission
Trout Streams Database Design

Christina Malyshko

Professional Science Masters in GIS

Table of Contents

I. DATA BACKGROUND

This project was designed to create a comprehensive database that stores data about both streams with naturally reproducing trout populations and streams stocked by the Pennsylvania Fish and Boat Commission (PAFBC) annually, which can help predict trout population health, abundance, and biomass throughout the waterways of Pennsylvania (PA). The initial data for this database was acquired from the Pennsylvania State University's PA Spatial Data Access Portal (PASDA) from the 2024 Class A Wild Trout Streams and the 2024 Trout Stocked Streams data tables, components of the PAFBC Fisheries Resource Database. Published in March 2024, this data is for the purpose of mapping and analysis of the approved trout fishing waters in PA, for decision making and planning procedures, and for promotion of PA trout fishing opportunities. The intended use of this data is for PAFBC employees and subsidiaries to monitor trout populations to determine if waterbodies are fit for public fishing and determine which locations are best for stocking. It may also be used by fisherman to determine where the best local waterbodies are for trout fishing.

In conjunction with two other data sources, this data will be used to analyze spatial questions related to fishing access, fishing conditions, and trout population source. The first of the secondary data sources is the PAFBC Best Fishing Waters dataset acquired from PASDA and published in November 2024. This data is based off a PAFBC program that identifies and records waterways that are known to have specific species of fish. The purpose of this data is to map and promote PA streams that are known to have good fishing opportunities for use in fishing trip planning and government projects. The second of the secondary data sources is the PA Department of Conservation and Natural Resources (DCNR) PA Trail Heads dataset acquired

from the Commonwealth of PA's Open Data Portal and last updated November 18, 2023. This dataset provides the point location of every DCNR maintained trail head across the state of PA.

Utilizing these datasets, three ecological based spatial queries were performed. The first being to determine the number of trail heads that are within one mile of the trout stream sections. This query will be useful to determine how accessible a stream section is and to see if other tools might be necessary to access the trail. It may help answer questions like if there is parking nearby or if a multiple trail hike is necessary to access the stream section. How secluded or how close to a city a stream section is to determine water quality and fish health. The second query will be if any of the trout stream sections coincide with streams from the best fishing waters data as well as if there are any inconsistencies in their naming conventions. This query will allow one to establish whether a stream section has a high likelihood of having significant fish abundance. This will also allow for errors in naming to be discovered and to be corrected in both datasets. The final query will be determining the three closest stocked streams to each naturally producing stream. This will call attention to any regions that are lacking in fish abundance so that the PAFBC can focus on those for future stocking efforts. This will also help determine if any of the natural fish populations are reproducing with the stocked population and adding genetic variability to the trout gene pool. An important factor for species rehabilitation and survival over time.

II. DATA STRUCTURE

These two datasets are similarly structured as they originate from the same government organization and database. The 2024 Class A Wild Trout Streams, a 3.7MB zipped shapefile, consists of 1,216 rows and 36 columns, where each row represents an individual stream section,

and each column represents a specific attribute describing that section. Of these attributes there is a combination of numeric and categorical variables that provide an in-depth view on the location, ecosystem, and fishing information of the specific stream section. Examining the data, there are numerous areas of the tables that could use improvement. One example being attributes where there are about 0.05% of the rows contain data, e.g., mentored_youth, fishing_hole, and wtr_programs, or attributes that contain the same data value for every row, e.g. stocking_year. Another example is attribute data that can be quickly calculated using a spatial PostgreSQL query or with a GIS analysis software, e.g., section_length_ft and lower_limit_longitude. These inconsistencies will become important later when the database becomes normalized.

The other dataset, 2024 Trout Stocked Streams, is a 3.2MB zipped shapefile that consists of 1,036 rows representing individual stream sections and 34 columns of attributes describing each section. Designed similarly to the Class A Wild Trout Streams data, there are a both categorical and numeric data variables that help to give the user a better understanding of the stream section. There are also multiple attributes containing limited, repeated, and easily calculatable values that will be handled in the normalization step.

Before normalization, the two datasets were loaded into DBeaver and the join query below was run to create an import table called combined_streams that will contain all the data being used to design the database. The natural full join function was implemented to ensure that if a column name was the same in both datasets, then it be coalesced into a single column in the combined_streams table.

```
DROP TABLE IF EXISTS combined_streams CASCADE;

CREATE TABLE combined_streams AS
SELECT *
FROM class_a_streams
```

```
NATURAL FULL JOIN stocked_streams;
```

III. DATA NORMALIZATION

After being downloaded and unzipped, the data was uploaded into PostgreSQL, using QGIS, where it can be further normalized and used for spatial queries in DBeaver. However, since the datasets are acquired from a larger government database and updated by many people, there are inconsistences in the database structure that needed to be normalized first. To execute the normalization process efficiently, an Entity Relationship Diagram (ERD) was utilized to dissect the original datasets, to identify key components used in the database, and to design the structure of the final database.

To start the normalization process, the metadata of each dataset was explored to define the variables and learn the origin of each attribute. By reading the PAFBC data dictionary, it became clear that the post-creation GIS software products are not necessary for the final dataset, since they can be easily acquired if needed. However, at this point, all original data was conserved to start. The first step in the normalization process was to create individual ERDs for each of the datasets and establish initial relationships in the data. All ERDs for this project converted the data into first normal form. This is the process of splitting the data into multiple tables so that every attribute is a single value. To get the initial data into first normal form, data that is repeated for multiple rows and data that is essential to describe a stream section had to be defined.

For both datasets, data that describes the entire stream, i.e., stream id, stream name, stream mouth latitude and longitude, and county, could be used to describe multiple stream sections. So, that data can be separated into its own table with stream_id being a unique identifier

for each stream, hence the primary key of the stream table. Moving down in stream hierarchy, isolating the individual stream sections data into its own table was the next step. Since each stream section had a unique identifier and multiple sections can be related back to one stream_id, there was a many relationship going from the stream to section table but a one relationship going from the section to stream table. The primary key of the stream table will become a foreign key in the section table and section_id of the section table will become its primary key. The section table contained the data section_id, stream_ id, section geometry, watershed basin, watershed sub-basin, and reference points to the upper and lower limits of the section. Finally, the descriptive data, i.e., fish biomass, trout species, and fishing regulations, that can be applied to multiple sections. These were each broken down into individual tables with a one relationship to each table from the section table and a many relationship from the descriptive tables to the section table.

Now that each table had an individual ERD, it was easier to visualize how the final database will be organized. Once the two datasets were joined, it needed to be further evaluated whether the combined_streams table was completely in first normal form. The final ERD can be reference in Figure 1 as the normalization process is explained. Moving in stream hierarchy, the watershed basin and watershed sub-basin data could be split into two separate tables. This is because many streams can populate a sub-basin and many sub-basins can populate a watershed basin. These two tables were created as lookup tables that will be referenced by the stream table. Each basin and sub-basin was assigned an arbitrary integer identifier in the id column and the description column will provide the name of the location in a variable-length character string format. The id columns are the primary keys for each table. The primary key of the watershed basin table is a foreign key in the sub-basin table. The sub-basin will be a foreign key in the

stream table, which will only provide the integer value for the sub-basin and the user will need to reference the sub-basin lookup table for the complete name data.

Moving down in stream hierarchy to the stream table, the stream table remains in a similar format to the individual dataset ERDs. It will contain an integer stream_id acting as the primary key of the table, a stream_name column in variable-length character string format, mouth_lat and mouth_lon integer columns to represent the stream mouth latitude and longitude in decimal degree, a county column in variable-length character string format, and the subbasin_id integer foreign key referencing the sub_basin table. This table will be related to the sub_basin table with a one relationship and the sub_basin table relates with a many relationship. The stream table will also relate to the section table with a many relationship and the section table will relate with a one relationship, since there are many stream sections that fall into one stream.

Finally, the section table and its related link and lookup tables. The section table being the main table of the database holds the geometry of each stream section and has a relationship to every table in the database, whether direct or indirect. The section table will have a gid as a unique integer identifier and primary key, which in this case is being deemed a “geometric identifier”. It will also hold the stream_id foreign key from the stream table, multi-line string geometry of the section reprojected into Pennsylvania State Plane (SRID:2272) from latitude and longitude (SRID: 4326), and the upper and lower limits of the section represented as integer values of mile markers along the stream. The remaining attributes in the table will all be the foreign keys for the related lookup and link tables.

Each of the section related lookup tables were constructed with similar structure to the sub_basin and watershed_basin tables. The regulation (PAFBC fishing regulations), fish_source

(section is stocked or naturally producing), and trout_biomass (average number of trout counted in an area divided by average weight of trout) tables all consist of a primary key unique identifier. For fish_source and regulation tables this id is an integer and for trout_biomass it is a variable-length character string with a letter A to E. All the tables were also given a description column to define the attribute variables in detail in variable-length character string format. Each lookup table relates back to the section table with a many relationship, since each can be matched with multiple stream sections. The section table relates to the lookup table with a one relationship, since each stream will only have one value from each lookup table. All the lookup table identifiers act as foreign keys in the section table.

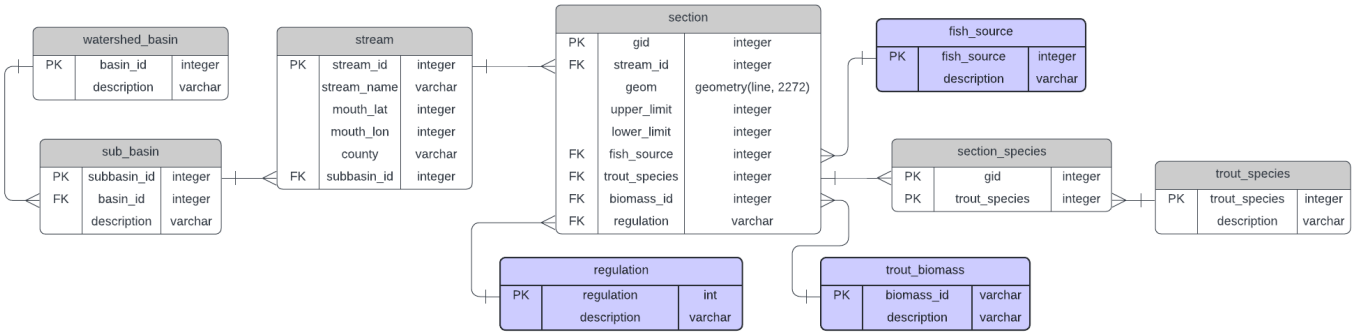
Transitioning to the link table, a section_species table was created to have multiple trout species present in a section if necessary. This structure will also become useful in the future if other fish species data is added to the database and needs to be assigned to a stream section. The link table creates a connection between the section table and a new trout_species lookup table and contains both primary keys to relate them to one another. The trout_species table is designed like the other lookup tables with a unique integer identifier primary key and a detailed description column. By going through the link table, this allows for one trout species to be present in the section table at a time without violating first normal form, if necessary.

After the design of the final ERD was completed, the previously joined data was accessed in DBeaver where a normalization script was optimized. In this process, Appendix II, a series of create table and insert statement queries were conducted to take the data from the import combined_streams table into their designated table from the ERD plan. A sample of the code can be seen below. Once all the new tables were created and connected, the data was ready to be used in spatial queries and spatial indexes can be produced to optimize the code.

```
DROP TABLE IF EXISTS fish_source CASCADE;

CREATE TABLE term_project.fish_source
    (fish_source int PRIMARY KEY,
    description varchar);

INSERT INTO fish_source
    VALUES
    (1, 'Natural Reproducing Stream'),
    (2, 'Government Stocked Stream');
```



<u>Figure 1</u>: This figure is the Entity Relationship Diagram (ERD) for the final database. The main structure of the database is outlined here as each table's, the rectangles, relationship is outlined using line connections from the primary key, i.e., the main unique identifier for the table, of one table to a foreign key in another table, i.e., an attribute referencing the primary key from the first table. The way two table are related is represented with a symbol. The primary symbols here are —|— for an "one" relationship, i.e., the data is related to only one set of attributes, and —<— for a "many" relationship, i.e., there are many attributes that can be combined in a relationship. For this ERD, the section table is the main table, and each section will only have one set of values from each of the other tables, but each table can supplement many sections.

IV. <u>SPATIAL ANALYTICS</u>

Now that the database is set up, the data can be used to asked spatial questions about trout streams in Pennsylvania. For the sake of space, three spatial questions were analyzed for this report.

The first question that was examined was how many trail heads that are within one-mile of each stream sections. This query, seen below, will be useful in determining how accessible a section of the stream is to a main road, park trail system, known point of interest, or parking lot. It will also allow the user of the stream section to know if they will need to bring trekking or forestry tools to access a part of the stream if it is not one of the queried results. Aside from accessibility, determining the proximity of a stream section to a trail head can also help you speculate water quality and fish health. The closer a stream is to a trail head, than the closer it is to a parking lot, roads, and general civilization. Being near civilization means a stream has a higher risk of being impacted by fertilizer and pollution runoff, which over time can negatively impact the health of the stream ecosystem and should be a determining factor for stocking choice and fishing location.

SPATIAL QUERY #1:

```
SELECT DISTINCT ON (t.gid, st.stream_name)
    t.name01 AS trail, t.gid AS trailhead_id, t.geom AS trail_geom,
    st.stream_name AS stream, s.gid AS stream_section_id,
    s.geom AS section_geom, (ST_Distance(ST_Transform(t.geom, 2272),
    s.geom)/1609.34) AS trail_stream_distance_mi
FROM trail_heads t
JOIN section s
ON ST_DWithin(ST_Transform(t.geom, 2272), s.geom, 1609.34)
JOIN stream st
ON (st.stream_id = s.stream_id);
```

For this query, shown above, a full spatial join is completed to join the trail_heads and the section table using the ST_DWithin function as the join on clause. This function is being used to extract all the rows in which the trail geometries are within 1 mile or 1609.34 meters of a stream geometry. 1609.34 needs to be used, since the PA State Plane projection utilizes meters, so a

distance conversion is needed is determine miles. The ST_DWithin function also needs to convert the trail geometry to PA State Plane from longitude and latitude using ST_Transform. The section table is also joined to the stream table on the condition that the stream_id is equal in both tables. Looking at the select statement, the distinct on function is used in to ensure that the trail heads are not selected twice for the same stream when they are in proximity to multiple sections of the same stream. The distinct on function allows unique rows to be extracted based on one or more columns in the table and discards duplicate row values. For example, if stream section A is close to trail head 1 and stream section B is also close to trail head 1, then the query will only include the trail head once for the whole stream extent. Distinct on here is being used to sort data based on the trail head gid and the stream name. The remainder of the select statement is looking to include trail name, trail head id, trail head geometry, stream name, stream section id, stream geometry, and distance from trail head to stream section in miles. Finally, once the query was run, the results were exported into a table and opened in QGIS. Here they were mapped over Pennsylvania state and county boundaries and the symbology was edited to make the relationship clearer. The QGIS depiction of the results can be seen below in Figure 2.

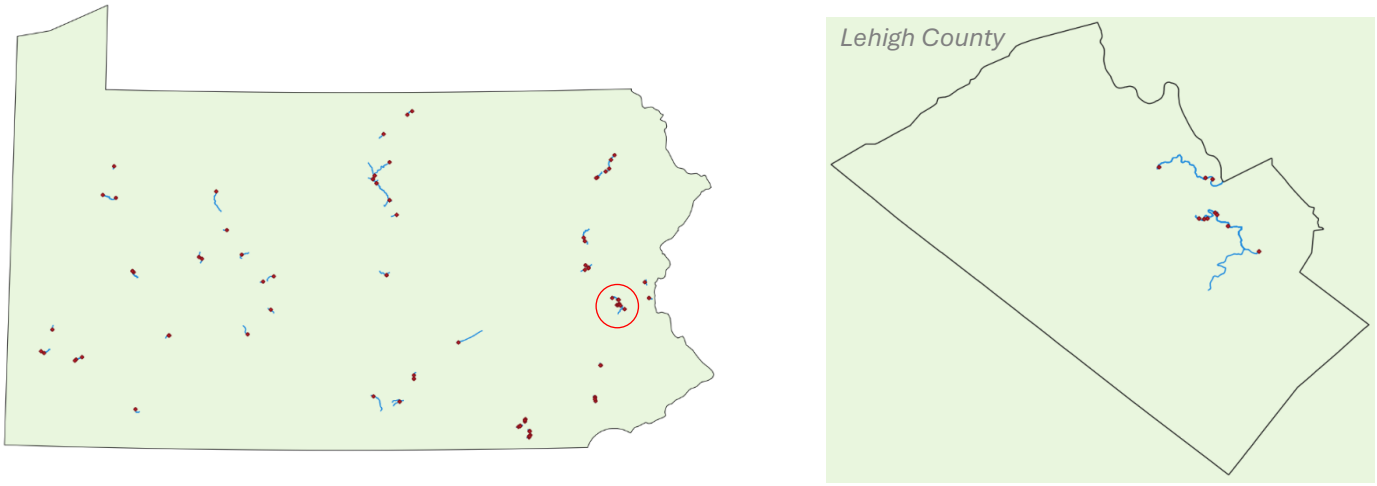


Figure 2: This figure represents the results of running a spatial query to return the stream sections in Pennsylvania that are within one mile of trail heads. The blue lines represent the selected stream sections that satisfy the query, the red diamonds are the trail heads within 1 mile, and the red circled region is the location of the right image. The image on the left shows the results across the entire state of Pennsylvania. The image on the right is a zoomed in view on the results of the query in Lehigh County, which was randomly chosen to depict the results.

The next spatial question that was determining whether any of the stream sections from this database are classified as best fishing waters based on the PAFBC best fishing data. Then, of these streams are there any inconsistencies in their naming conventions. Using this query will allow someone to determine if there is a high likelihood of them catching fishing in this stream. It will also allow stocking officials to interpret if this stream has a high fish abundance due to many people finding fish in it. So, they may not need to focus on planning large stocking efforts for this area. Aside from the physical benefits to this query, the added information of inconsistencies in naming conventions can also allow for the database managers to update the stream names if names have changed or have been recorded incorrectly with time.

SPATIAL QUERY #2:

```
SELECT b.wtrname AS best_fishing_stream, ST_Transform(b.geom, 2272)
       AS best_geom, st.stream_name AS stream, s.gid AS
       stream_section_id, s.geom AS section_geom
FROM best_fishing_waters b
JOIN section s
ON ST_Intersects(ST_Transform(b.geom, 2272), s.geom)
JOIN stream st
ON (st.stream_id = s.stream_id)
WHERE b.wtrname != st.stream_name;
```

For this query, shown above, a full spatial join between the best_fishing_waters table and the section table was completed on the condition that the best fishing line geometry intersects the section line geometry. In this step, the best fishing geometry needed to be reprojected from longitude/latitude to Pennsylvania State Plane, 2272 to make the intersection accurate. Next, the section table was joined to the stream table on the condition that the stream_id is the same in both tables, so the information about the entire stream can be accessed. For the select statement, the stream name and geometry in PA State Plane projection were selected from the best fishing table and the stream name, section gid, and section geometry were select from the stream and section tables respectively. Finally, the where condition was added to only show streams where the names were not the same giving us names with typos or naming inconsistencies. The results of this query were exported into a table to be mapped in QGIS. The data was plotted over the state and county boundaries of PA and the symbology was edited to clearly show the relationship, which can be seen below in Figure 3.

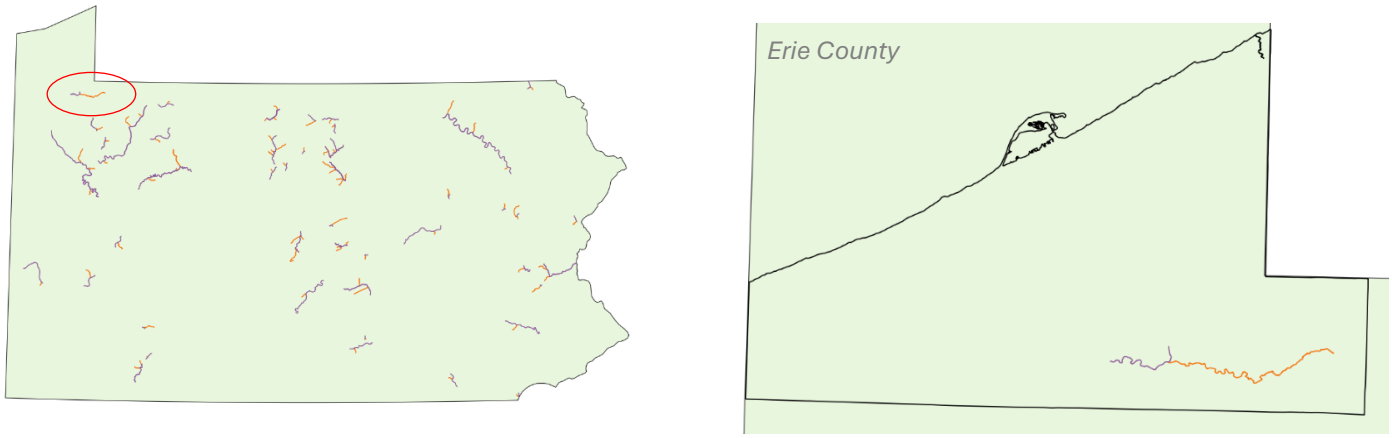


Figure 3: This figure represents the results of running a spatial query to return the stream sections within Pennsylvania that are also classified as best fishing waters. The orange lines represent the stream sections from the project database, the purple lines represent the streams from the best fishing dataset, and the red circled region is the location of the right image. The image on the left shows the results across the entire state of Pennsylvania. The image on the right is a zoomed in view on the results of the query in Erie County, which was randomly chosen to depict the results.

The final spatial question is what the three closest stocked stream section to are each naturally reproducing stream section. Obtaining this information will allow PAFBC officials to pinpoint areas of the stream that may have decreased fish abundance due to large gaps in the distance between both stocked sections and from section to section no matter the source. For example, if a naturally reproducing stream is 2 miles away from a stocked section and the next stocked section is 2 miles away from that, then there are four miles of stream that may be experiencing little to no fish activity. This may reveal issues in stream health due to external factors that many need state attention or reveal areas that may have been forgotten from previous stocking efforts. Once gaps in fish population abundance are addressed, this information can also

draw attention to areas in which naturally reproducing populations and stocking populations may become mixed and may introduce reproduction between the groups. The entire goal of stream stocking efforts is to revitalize the natural fish populations and increase genetic diversity within these species' throughput the state. Being able to have the data of where this is occurring, can prove to be a proxy for successful stocking efforts in a region and a return to a healthy natural trout population size.

SPATIAL QUERY #3:

```
SELECT a.gid AS natural_producing, a.geom AS natural_geom, r.gid AS
       three_closest_stocked, r.geom AS stocked_geom
FROM section AS a
    CROSS JOIN LATERAL
    (
        SELECT b.gid, b.geom
        FROM section AS b
        WHERE fish_source = 2
        ORDER BY b.geom <-> a.geom
        LIMIT 3
    ) AS r
WHERE fish_source = 1;
```

For this query, seen above, the section table is joined with itself. A cross join is used to join every record of the section table where the fish_source = 1 (a), i.e., where the stream is natural, with every record of the section table where the fish_source = 2 (r), i.e., where the stream is stocked. The lateral clause allows you to reference columns from prior items in the from statement. In this query, the lateral is helping to expand the geometries into component parts of the query and works with the nearest neighbor (`<->`) operator to overcome a constant geometry reference issue. In the subquery, the nearest neighbor operator is employed to determine what the closest geometries are from natural streams to the stocked stream and is limited to the three closest geometries. The rest of the subquery is formatted like a normal select

query where the stocked section gid and geometry are extracted from the section table. The main query is formatted similarly but the select statement references attributes from the main query section table and the sub-query section table. The natural section gid and geometry (a) and the stocked section gid and geometry (r) are obtained from their respective tables. This final dataset was added to a table and imported into QGIS where it was mapped on a PA state and county boundary map as seen below in Figure 4.

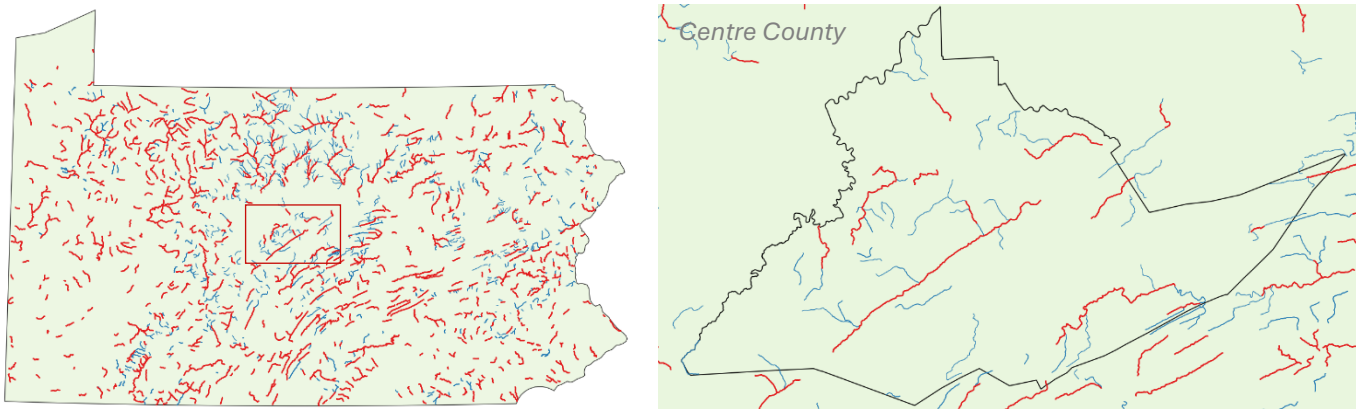


Figure 4: This figure represents the results of running a spatial query to return the three closest stocked stream sections to each naturally reproducing stream section within Pennsylvania. The red lines represent the natural stream sections, the blue lines represent stocked stream sections, and the red rectangle region is the location of the right image. The image on the left shows the results across the entire state of Pennsylvania. The image on the right is a zoomed in view on the results of the query in Centre County, which was randomly chosen to depict the results.

V. DATABASE OPTIMIZATION

In the intention of improving timing and efficiency for running the above spatial queries, each query was analyzed using the EXPLAIN ANALYZE function and a query plan was developed by DBeaver. Each of these query plans was exported and brough into a graphical

explain visualizer at https://explain.dalibo.com/ where the software analyzed the query plan, produced a report on speed and cost of the query, and produces a flow chart style diagram of the query. Using this tool allows one to identify areas of the query that are slowing the processing speed down and can use additional editing to streamline the querying process. The diagrams for each spatial question can be seen below along with their optimized versions.

For the first spatial query, trail heads near stream sections, the initial query run took 3.914 seconds to complete. Looking at the query plan diagram, figure 5, the second nested loop containing the section and trail head data, and the ST_Transform function being run for each row was causing the query to slow down. To fix this problem, a new column was added to the trail_heads table to include the reprojected geometries into 2272, so the query did not have to spend space and time running this function each time. Next step was adding indexes. A GIST index was created for the section and reprojected trail_heads geometries. A normal index was created on the section table to store the gid and trail_heads table to store the trail name and trail head gid. Finally, a HASH index was created on the stream_id column in the stream table to alleviate a slowdown found in an intermediate optimization step. The finalized query with the indexes in place, seen in figure 6, took 7.71 milliseconds to run, about a 99.8% decrease.

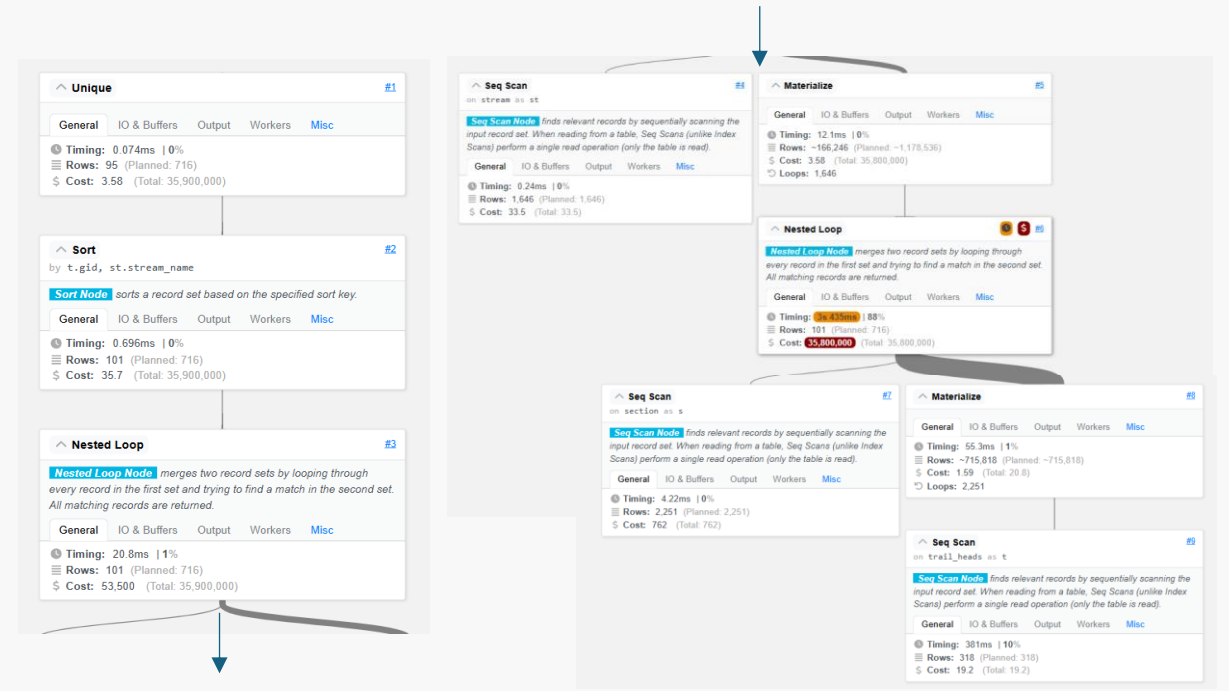


Figure 5: This figure is the graphical depiction of the optimization query plan created from the first spatial query of this project, determining the trail heads within one mile of the stream sections, before optimization. This is a tool to aid in improving performance of the database's queries. The chart starts on the left and continues to the right. The lower half of the right image shows red circles indicating this step is causing a lag in run speed.

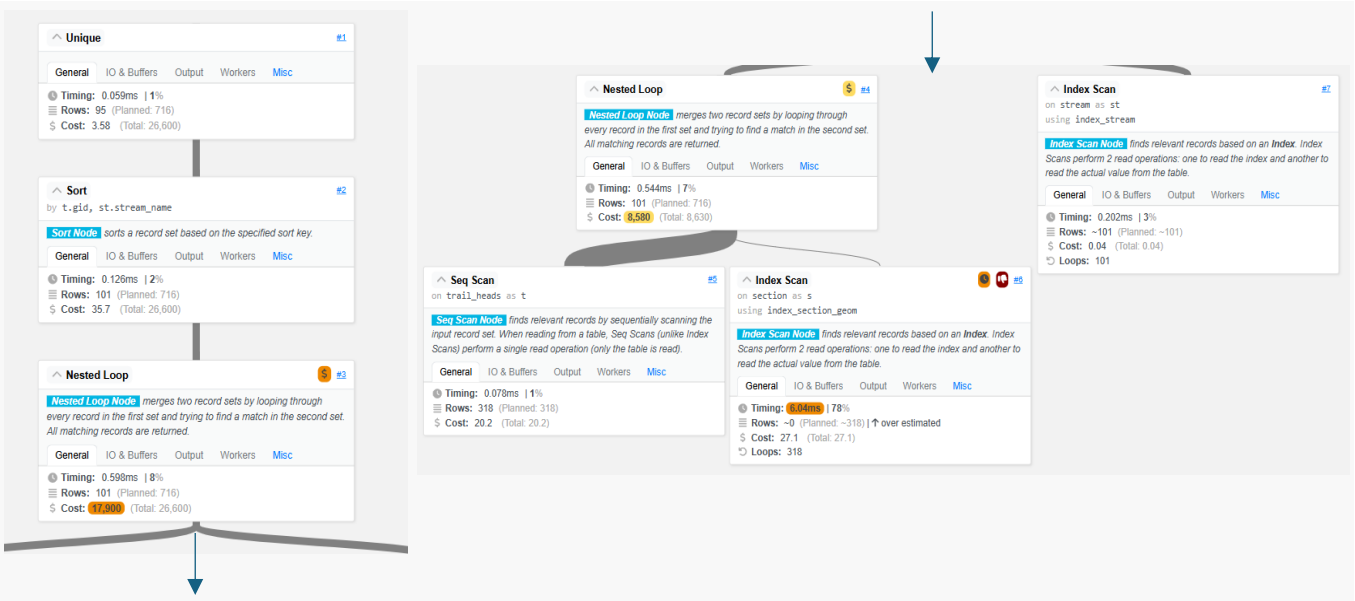


Figure 6: This figure is the graphical depiction of the optimization query plan created from the first spatial query of this project, determining the trail heads within one mile of the stream sections, after optimization. This is a tool to help aid in improving the performance of the database's queries. The chart starts on the left and continues to the right. After optimization, much of the red is nearly gone and was replaced with orange and yellow warnings.

Advancing to the next spatial query, best fishing waters coinciding with stream sections, the initial run took 1 minute and 37 seconds to complete. Examining the query plan, figure 7, the second nested loop containing the section and best fishing data and the ST_Transform function

being run for each row was causing the query to slow down. Like the previous query, a new column was added to the best_fishing_waters table to include the reprojected geometries into 2272. Next indexes were created. A GIST index was created for the reprojected best_fishing_waters geometries and a normal index was created on the best_fishing_waters table to store the stream name. The section table indexes, and HASH stream_id index created previously was reused here. The finalized query with the indexes in place, seen in figure 8, took 81.4 milliseconds to run, about a 99.92% decrease in time.

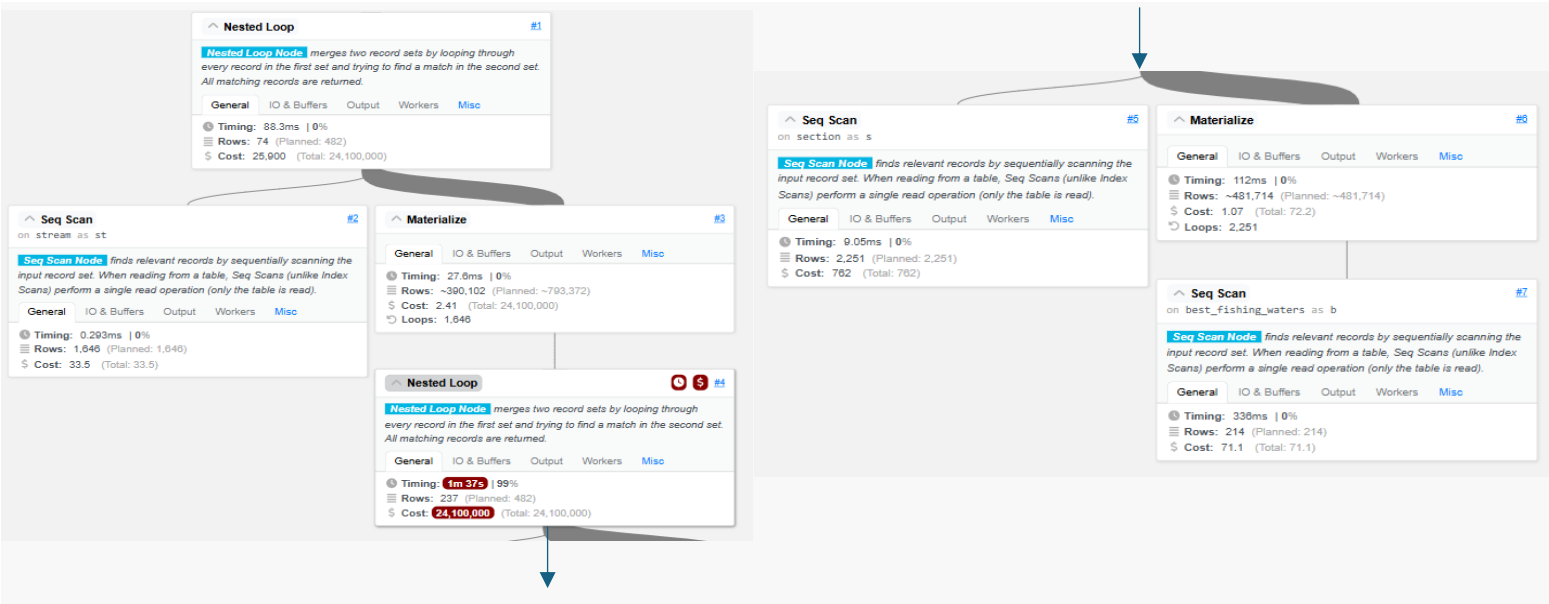


Figure 7: This figure is the graphical depiction of the optimization query plan created from the second spatial query of this project, are the stream sections best fishing classified streams, before optimization. This is a tool to help aid in improving the performance of the database's queries. The chart starts on the left and continues to the right. The lower half of the left image shows red circles indicating this step is causing a lag in run speed.

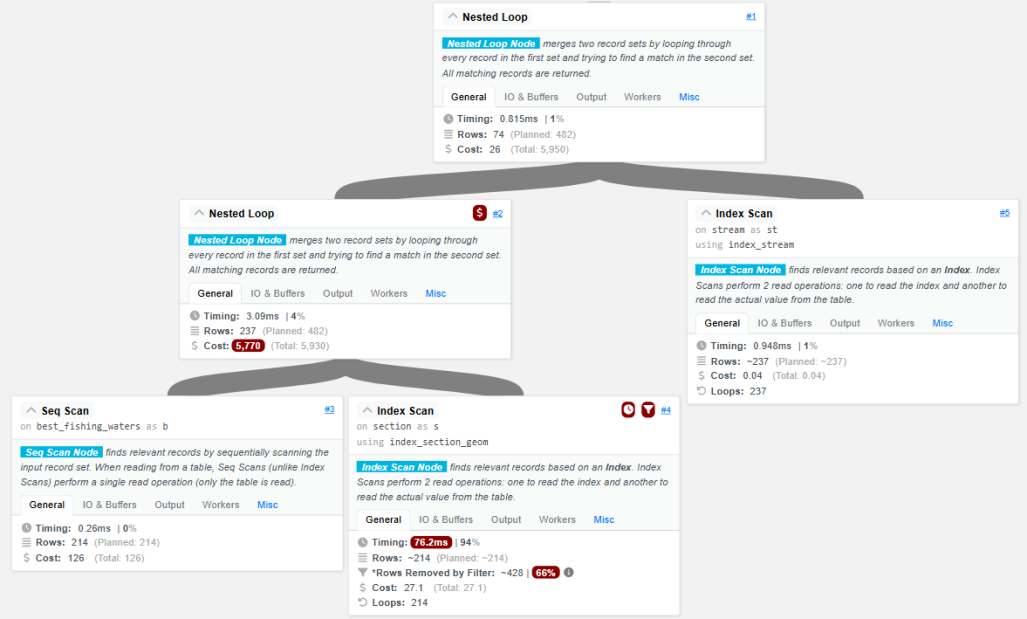


Figure 8: This figure is the graphical depiction of the optimization query plan created from the second spatial query of this project, are the stream sections best fishing classified streams, after optimization. This is a tool to help aid in improving the performance of the database's queries. The chart had a reorganization after optimization and with a large decrease in run speed.

For the last spatial query, determining the three closest stocked streams to the natural reproducing streams, the initial query run took 46.127 seconds. Looking at the query plan, figure 9, it was clear the last sequential scan containing the section geometry was causing the lag in querying speed. To fix this, the previously created section GIST index for section geometry was used along with a normal index containing the section gid and fish source data. The HASH stream id index was also included. These indexes resulted in a finalized query, figure 10, that took 3.404 seconds to run, which is a 92.6% decrease in time. Compared to the other two queries, this query was not as well optimized, so in the future some more advanced optimization steps will be needed to take to make this query match the speed of the other two.

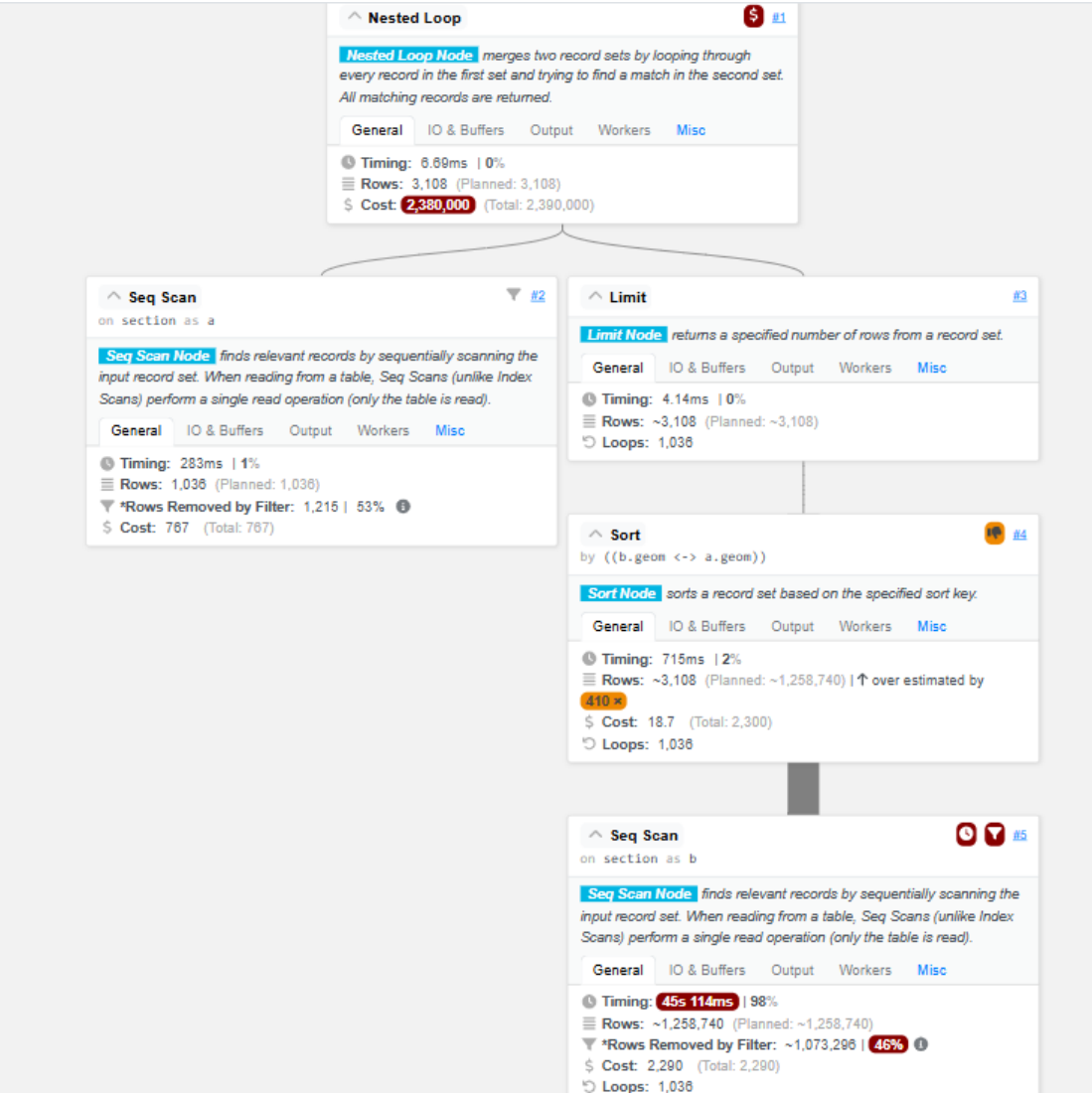


Figure 9: This figure is the graphical depiction of the optimization query plan created from the third spatial query of this project, which are the three closest stream sections to each natural reproducing stream section, before optimization. This is a tool to help aid in improving the performance of the database's queries. This chart reveals many areas of improvement, mainly the lower half.

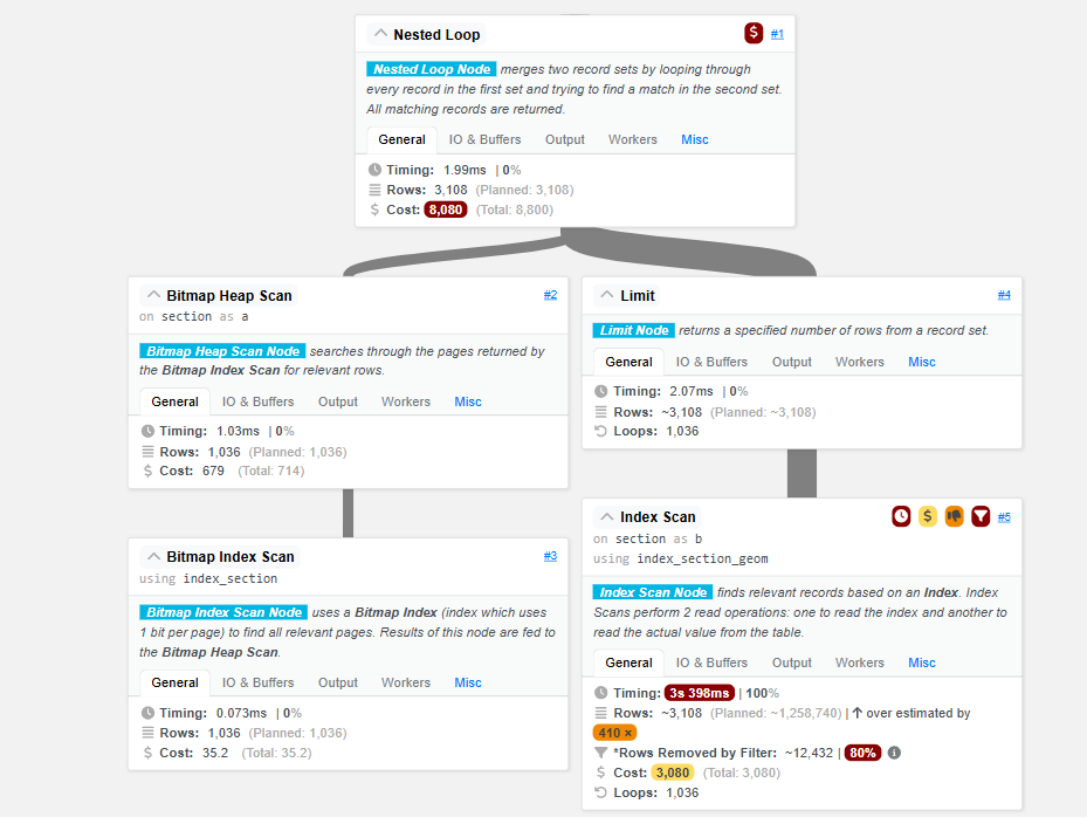


Figure 10: This figure is the graphical depiction of the optimization query plan created from the third spatial query of this project, which are the three closest stream sections to each natural reproducing stream section, after optimization. This is a tool to help aid in improving the performance of the database's queries. This chart reveals a large decrease in run speed after optimization, but still have areas left for improvement.

APPENDIX I - OBTAINING AND LOADING DATA

The stream data for this project can be obtained from The Pennsylvania Spatial Data Access Portal at https://www.pasda.psu.edu/. To get the datasets used in this project, each can be individually searched by a keyword in the left search bar or by searching Pennsylvania Fish and Boat Commission on the home page. Click on the dataset name hyperlink and once open, click download in the options bar under the title. The file should download to the browser's downloads tab.

The trail head data can be accessed from the Pennsylvania Open Data Portal at https://data.pa.gov/browse?sortBy=relevance&page=1&pageSize=20. To obtain the dataset, enter trail heads into the search catalog search bar and click on the Filtered View – Trail Heads Conservation & Natural Resources hyperlink. Once open, hit export in the top right corner and download file as a shapefile. The file should download to the browser's downloads tab.

The links to the data are below:

- 2024 Class A Trout Streams: https://www.pasda.psu.edu/uci/DataSummary.aspx?dataset=986
- 2024 Trout Stocked Streams: https://www.pasda.psu.edu/uci/DataSummary.aspx?dataset=1157
- Best Fishing Waters: https://www.pasda.psu.edu/uci/DataSummary.aspx?dataset=1107
- Pennsylvania Trail Heads: https://data.pa.gov/Services-Near-You/Filtered-View-Trail-Heads-Conservation-Natural-Res/e7zb-72e6/about_data

After downloaded, the data files will appear in a zip file format and will need to be extracted before continuing. Once unzipped, open a new project in QGIS and add each downloaded shapefile as a new layer. Next, navigate to the database tab and open DB Manager. Click on the PostGIS provider and choose the gis container. Navigate to the schema tab, click create new schema, and choose a name relevant to this project. Hit the refresh button, click on the new schema, and select import new layer/file from the task bar. In the input vector layer box, choose a layer file from the drop-down bar, rename the output table “table” box to be an all lowercase and meaningful name, and check the primary key and geom box. Click ok and repeat for all four shapefile layers. Once the data is added into the PostGIS schema, it can be accessed in DBeaver and utilized for the rest of the project.

APPENDIX II – NORMALIZATION SCRIPT

```
-- Christina Malyshko
-- PAFBC Trout Database Normalization Script

SET search_path TO term_project, public;

-- IMPORT TABLE WHAT ALL OTHER TABLES WILL REFERENCE FOR THEIR DATA INSERTS
-- Join class_a_streams and stocked_streams and make new table
-- Join all attributes from both tables (if same column name does not create new
columns, combines columns)

DROP TABLE IF EXISTS combined_streams CASCADE;

CREATE TABLE combined_streams AS
SELECT *
FROM class_a_streams
NATURAL FULL JOIN stocked_streams;


-- Create watershed_basin table

DROP TABLE IF EXISTS watershed_basin CASCADE;

CREATE TABLE term_project.watershed_basin
    (basin_id int PRIMARY KEY,
    description varchar);

INSERT INTO watershed_basin (basin_id, description)
    VALUES
    (1, 'Delaware'),
    (2, 'Upper/Middle Susquehanna'),
    (3, 'Lower Susquehanna'),
    (4, 'Potomac'),
    (5, 'Great Lakes'),


-- Create sub_basin table

DROP TABLE IF EXISTS sub_basin CASCADE;

CREATE TABLE term_project.sub_basin
    (subbasin_id int PRIMARY KEY,
    basin_id int REFERENCES watershed_basin(basin_id),
    description varchar);
```

```
INSERT INTO sub_basin (subbasin_id, basin_id, description)
    VALUES
    (1, 1, 'Upper Delaware River'),
    (2, 1, 'Central Delaware River'),
    (3, 1, 'Lower Delaware River'),
    (4, 2, 'Upper Susquehanna River'),
    (5, 2, 'Upper Central Susquehanna River'),
    (6, 3, 'Lower Central Susquehanna River'),
    (7, 3, 'Lower Susquehanna River'),
    (8, 2, 'Upper West Branch Susquehanna River'),
    (9, 2, 'Central West Branch Susquehanna River'),
    (10, 2, 'Lower West Branch Susquehanna River'),
    (11, 3, 'Upper Juniata River'),
    (12, 3, 'Lower Juniata River'),
    (13, 4, 'Potomac River'),
    (14, 5, 'Genesee River'),
    (15, 5, 'Lake Erie'),
    (16, 6, 'Upper Allegheny River'),
    (17, 6, 'Central Allegheny River'),
    (18, 6, 'Lower Allegheny River'),
    (19, 6, 'Monongahela River'),
    (20, 6, 'Ohio River');


-- Create Stream Table

DROP TABLE IF EXISTS stream CASCADE;

CREATE TABLE term_project.stream
    (stream_id int PRIMARY KEY,
    stream_name varchar,
    mouth_lat float8,
    mouth_lon float8,
    county varchar,
    subbasin_id int REFERENCES sub_basin(subbasin_id)
    );


INSERT INTO stream (stream_id, stream_name, mouth_lat, mouth_lon, county,
subbasin_id)
    SELECT DISTINCT wrds, wtrname, wtrlatdd, wtrlondd, county_nam, secsubbasi
    FROM combined_streams
    WHERE wrds IS NOT NULL
    ON CONFLICT (stream_id) DO NOTHING
```

```
    ;


-- Create Fish_Source Lookup Table
-- update in combined streamsto add column for which table the data originally
came from

ALTER TABLE combined_streams
ADD fish_source int;

UPDATE combined_streams
SET fish_source = CASE
WHEN stockingye IS NULL THEN 2
ELSE 1
END;

DROP TABLE IF EXISTS fish_source CASCADE;

CREATE TABLE term_project.fish_source
    (fish_source int PRIMARY KEY,
    description varchar);

INSERT INTO fish_source
    VALUES
    (1, 'Natural Reproducing Stream'),
    (2, 'Government Stocked Stream');



-- Create Regulation Lookup Table


DROP TABLE IF EXISTS regulation CASCADE;

CREATE TABLE term_project.regulation
    (regulation int PRIMARY KEY,
    description varchar);

INSERT INTO regulation (description, regulation)
VALUES
    ('Catch and Release, All Tackle', 1),
    ('Catch and Release, Artificial Lures Only', 2),
    ('Catch and Release, Fly Fishing Only', 3),
    ('Commonwealth Inland Waters', 4),
```

```
    ('Delayed Harvest, Artificial Lures Only', 5),
    ('Lake Erie, Presque Isle Bay, and Tribs', 6),
    ('Miscellaneous Special Regulations', 7),
    ('Stocked Trout Water', 8),
    ('Stocked Trout Waters Open Year-Round', 9);



-- Create Trout-Biomass-Classifier Lookup Table

DROP TABLE IF EXISTS trout_biomass CASCADE;

CREATE TABLE term_project.trout_biomass
    (biomass_id varchar PRIMARY KEY,
    description varchar);\

INSERT INTO trout_biomass
    VALUES
('A', 'a. Total trout biomass of at least 30 kg/ha (26.7 lbs/acre)
b. Total biomass of trout less than 15 cm (5.9 in.) total length of at least 0.1
kg/ha
c. Brook trout biomass must comprise at least 75% of total trout biomass'),
('B', 'a. Total trout biomass of at least 20 kg/ha (17.8 lbs/acre) and less than
30 kg/ha (26.7 lbs/ acre)
b. Total trout biomass of at least 20 kg/ha (17.8 lbs/ acre) and less than 40
kg/ha (35.6 lbs/acre)'),
('C', 'Total Trout biomass of at least 10 kg/ha (8.9 lbs/ acre) and less than 20
kg/ha (17.8 lbs/acre)'),
('D', 'Total trout biomass less than 10 kg/ha (8.9 lbs/ acre)'),
('E', 'Total trout biomass of 0 kg/ha');



-- Create Trout_Species Look up Table

DROP TABLE IF EXISTS trout_species CASCADE;

CREATE TABLE term_project.trout_species
    (trout_species int PRIMARY KEY,
    description varchar);

INSERT INTO trout_species(trout_species, description)
VALUES
    (1, 'Brook Trout'),
    (2, 'Brown Trout'),
```

```
    (3, 'Rainbow Trout'),
    (4, 'Mixed Brook/Brown Trout'),
    (5, 'Mixed Brook/Brown/Rainbow');



-- Create Section Table

DROP TABLE IF EXISTS "section" CASCADE;

CREATE TABLE term_project."section"
    (gid int PRIMARY KEY GENERATED ALWAYS AS IDENTITY,
    stream_id int REFERENCES stream(stream_id),
    geom geometry(MULTILINESTRING, 2272),
    upper_limit int,
    lower_limit int,
    fish_source int REFERENCES fish_source(fish_source),
    trout_species int REFERENCES trout_species (trout_species),
    biomass_id varchar REFERENCES trout_biomass(biomass_id),
    regulation int REFERENCES regulation(regulation)
);

INSERT INTO "section" (stream_id, geom, upper_limit, lower_limit, fish_source,
trout_species, biomass_id, regulation)
SELECT wrds,
    (SELECT ST_Force2D(ST_Transform(geom, 2272))),
secupperli, seclowerli, fish_source,
    CASE fishery
        WHEN 'Brook' THEN 1
        WHEN 'Brook Trout Fishery' THEN 1
        WHEN 'Wild Brook Trout Fisharies'THEN 1
        WHEN 'Wild Brook Trout Fishery' THEN 1
        WHEN 'Brown' THEN 2
        WHEN 'Brown Trout Fishery' THEN 2
        WHEN 'Wild Brown Trout Fishery' THEN 2
        WHEN 'Wild Rainbow Trout Fishery' THEN 3
        WHEN 'Mixed Brook/Brown' THEN 4
        WHEN 'Mixed Brook/Brown Trout Fishery' THEN 4
        WHEN 'Mixed Brook/Brown/Rainbow' THEN 5
        WHEN 'Mixed Wild Brook/Rainbow Trout Fishery' THEN 5
        ELSE NULL
    END,
troutbioma,
    CASE reg_name
        WHEN 'Catch and Release, All Tackle' THEN 1
```

```
        WHEN 'Catch and Release, Artificial Lures Only' THEN 2
        WHEN 'Catch and Release, Fly Fishing Only' THEN 3
        WHEN 'Commonwealth Inland Waters' THEN 4
        WHEN 'Delayed Harvest, Artificial Lures Only' THEN 5
        WHEN 'Lake Erie, Presque Isle Bay, and Tribs' THEN 6
        WHEN 'Miscellaneous Special Regulations' THEN 7
        WHEN 'Stocked Trout Water' THEN 8
        WHEN 'Stocked Trout Waters Open Year-Round' THEN 9
        ELSE NULL
END
FROM combined_streams
WHERE geom IS NOT NULL;



-- Create Section_Species Link Table

DROP TABLE IF EXISTS section_species CASCADE;

CREATE TABLE term_project.section_species
    (gid int,
    trout_species int,
    PRIMARY KEY (gid, trout_species),
    FOREIGN KEY (gid) REFERENCES "section"(gid),
    FOREIGN KEY (trout_species) REFERENCES trout_species(trout_species));

INSERT INTO section_species
SELECT gid, trout_species
FROM "section"
WHERE trout_species IS NOT NULL;
```

APPENDIX III – SPATIAL QUERIES AND INDEXING SCRIPT

```
-- Christina Malyshko
-- PAFBC Trout Stream Database
-- Spatial Queries and Optimization


SET search_path TO term_project, public;


--  1. Count the Number of trails heads that are in within 1 miles of a stream
section

-- EXPLAIN ANALYZE
SELECT DISTINCT ON (t.gid, st.stream_name)
    t.name01 AS trail, t.gid AS trailhead_id, t.geom_2 AS trail_geom,
st.stream_name AS stream, s.gid AS stream_section_id,
    s.geom AS section_geom, (st_distance(t.geom_2, s.geom)/1609.34) AS
trail_stream_distance_mi
FROM trail_heads t
JOIN section s
ON ST_DWithin(t.geom_2, s.geom, 1609.34)
JOIN stream st
ON (st.stream_id = s.stream_id);



ALTER TABLE trail_heads
ADD COLUMN
    geom_2 geometry(POINT, 2272)
    GENERATED ALWAYS AS (
        ST_Transform(geom, 2272)
    )STORED
;


-- 2. Do any of the stream sections overlap with best fishing stream data? Of
those streams is there any stream naming inconsistencies?


-- EXPLAIN ANALYZE
SELECT b.wtrname AS best_fishing_stream, b.geom_2 AS best_geom, st.stream_name AS
      stream, s.gid as stream_section_id, s.geom as section_geom
FROM best_fishing_waters b
JOIN section s
ON st_intersects (b.geom_2, s.geom)
```

```
JOIN stream st
ON (st.stream_id = s.stream_id)
WHERE b.wtrname != st.stream_name;

ALTER TABLE best_fishing_waters
ADD COLUMN
    geom_2 geometry(multilinestringm, 2272)
    GENERATED ALWAYS AS (
        ST_Transform(geom, 2272)
    )STORED
;

-- 3. What are the three closest stocked sections to each natural producing
sections

--EXPLAIN ANALYZE
SELECT a.gid AS natural_producing, a.geom AS natural_geom, r.gid AS
three_closest_stocked, r.geom AS stocked_geom
FROM section AS a
    CROSS JOIN LATERAL
    (
        SELECT b.gid, b.geom
        FROM section AS b
        WHERE fish_source = 2
        ORDER BY b.geom <-> a.geom
        LIMIT 3
    ) AS r
WHERE fish_source = 1;



-- Create Indexes

CREATE INDEX index_section
ON "section" (gid, fish_source);

CREATE INDEX index_section_geom
ON "section"
USING GIST (geom);

CREATE INDEX index_stream
ON stream
USING HASH (stream_id);

CREATE INDEX index_trail
```

```
ON trail_heads (gid, name01);

CREATE INDEX index_trail_geom
ON trail_heads
USING GIST (geom_2);

CREATE INDEX index_best_fish
ON best_fishing_waters (wtrname);

CREATE INDEX index_best_fish_geom
ON best_fishing_waters
USING GIST (geom_2);
```

APPENDIX IV – DATA DICTIONARY

A. watershed_basin

basin_id (integer): identifier for one of six watershed basin regions in Pennsylvania

description (varchar): name of the watershed basin region

B. sub_basin

subbasin_id (integer): identifier for one of twenty watershed sub-basin regions in Pennsylvania

basin_id (integer): references watershed_basin table, which watershed basin each sub-basin resides in

description (varchar): name of the watershed sub-basin region

C. stream

stream_id (integer): identifier for entire stream

stream_name (varchar): name of stream

mouth_lat (float8): stream mouth (source) latitude in decimal degrees

mouth_lon (float8): stream mouth (source) longitude in decimal degrees

county (varchar): Pennsylvania county where stream is located

subbasin-id (integer): references sub_basin table, which watershed sub-basin stream is in

D. section

gid (integer): identifier for stream section

stream_id (integer): reference to stream table, which stream the section is a segment of

geom (multiline string geometry, SRID 2272): spatial geometry of the stream section in meters

upper_limit (integer): stream mile of upper limit of section

lower_limit (integer): stream mile of lower limit of section

fish_source (integer): references fish_source table, defines whether section is PAFBC stocked or naturally reproducing

trout_species (integer): references trout_species table, defines species of trout that have been recorded to inhabit stream section

biomass_id (varchar): references trout_biomass table, which PAFBC defined trout biomass quantity each stream section has been assigned

regulation (integer): references regulation table, which PAFBC fishing regulation (if any) stream section has assigned to it

E. fish_source

fish_source (integer): identifier for where the trout population came originated

description (varchar): defines whether section is PAFBC stocked or naturally reproducing

F. trout_biomass

biomass_id (varchar): identifier for trout biomass

description (varchar): PAFBC defined trout biomass classifications

G. regulation

regulation (integer): identifier for PAFBC fishing regulation

description (varchar): PAFBC fishing regulation name

H. trout_species

trout_species (integer): identifier for present trout species

description (varchar): species of trout that have been recorded to inhabit stream section

I. section_species

gid (integer): references section table, stream section identifier

trout_species (integer): references trout_species table, present trout species identifier