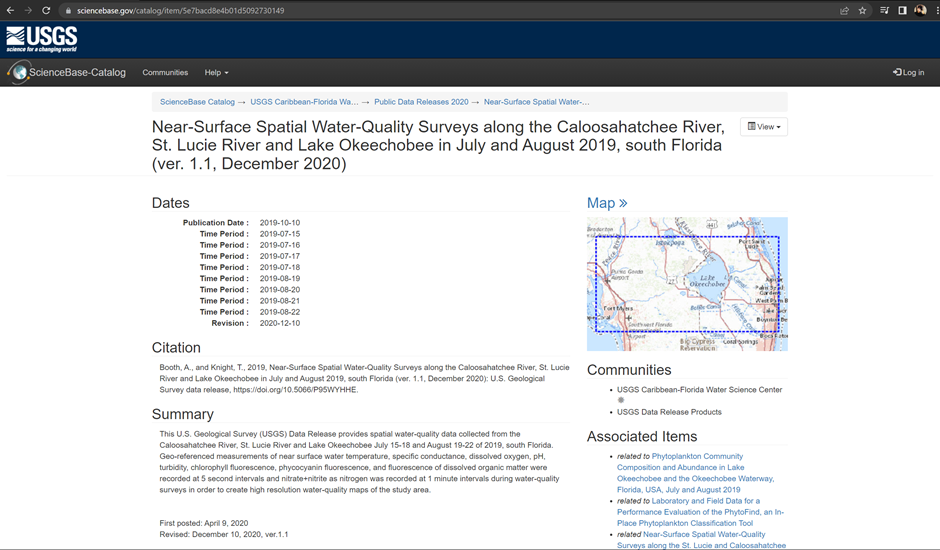
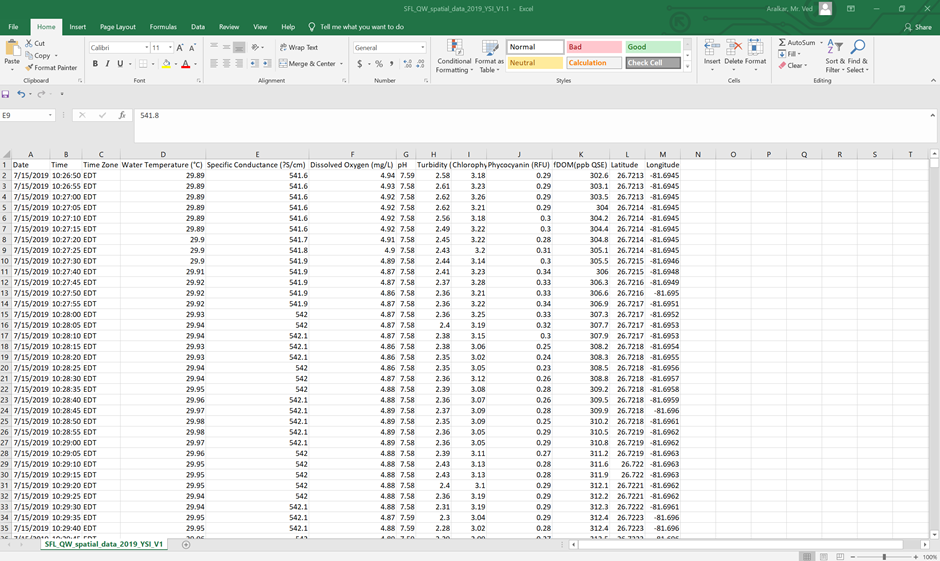
Group 22: Project 2

Group Members: Ved Aralkar, Abhrajit Das, Aditya Gadhave

**Project Description**: We have collected a water quality dataset from the following website:<https://www.sciencebase.gov/catalog/item/5e7bacd8e4b01d5092730149>



**Data:** The .csv file is a collection of water-quality survey data collected from the Southern part of Florida. The date range of the survey is July 15-18 and August 19-22, 2019.



**Goal:**

1. Retrieve locations of specific features.

2. Calculate distance between points.

3. Calculate areas of interest (specific to each group)

4. Analyze the queries

5. Sorting and Limit executions.

6. Optimize the queries to speed up execution time.

7. N-Optimization of queries.

#1 Retrieve locations of specific features.

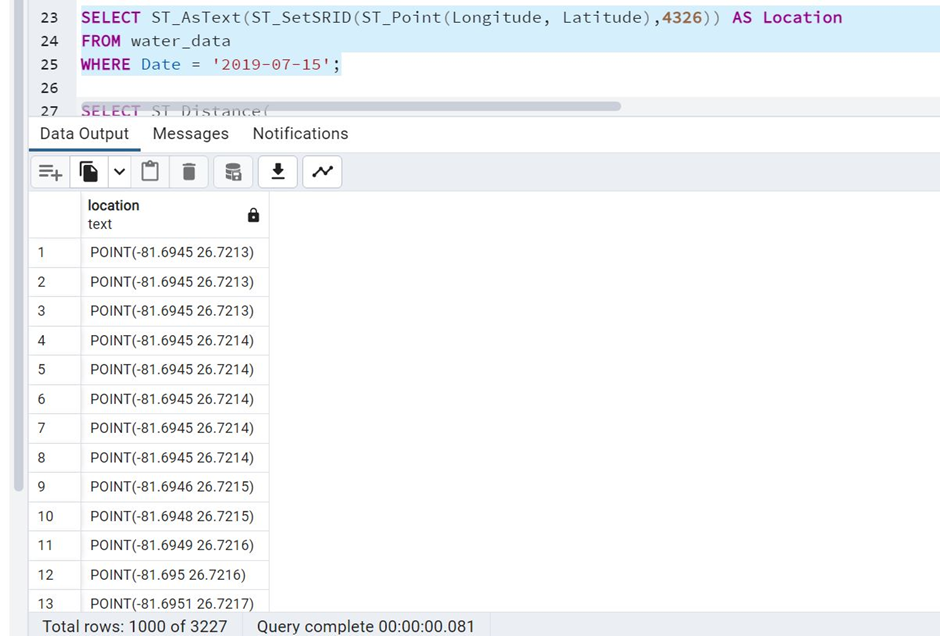
Code:

SELECT ST\_AsText(ST\_SetSRID(ST\_Point(Longitude, Latitude),4326)) AS Location

FROM water\_data

WHERE Date = '2019-07-15';

This code generates a list of the locations where water data was recorded on July 15, 2019.



1. Calculate distance between points.

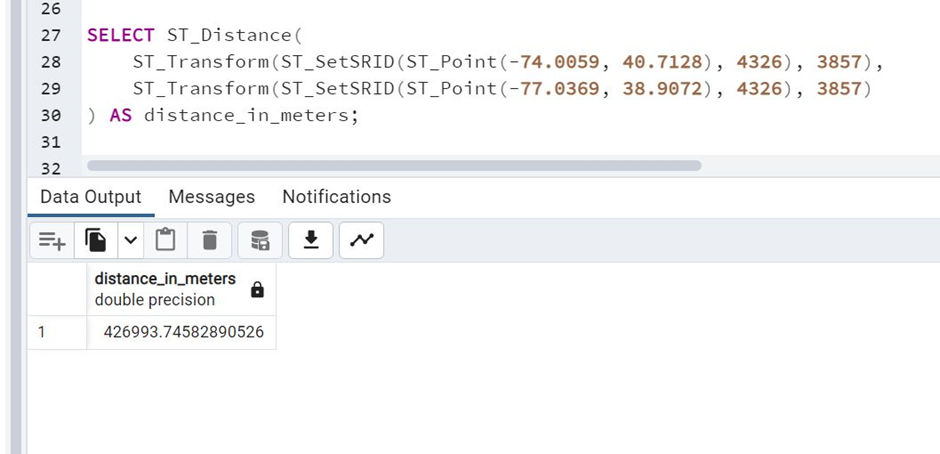
Code: SELECT ST\_Distance(

ST\_Transform(ST\_SetSRID(ST\_Point(-74.0059, 40.7128), 4326), 3857),

ST\_Transform(ST\_SetSRID(ST\_Point(-77.0369, 38.9072), 4326), 3857)

) AS distance\_in\_meters;

This code calculates the distance in meters between two points on the Earth's surface using their longitude and latitude coordinates. The first point is defined by -74.0059 longitude and 40.7128 latitude, and the second point is defined by -77.0369 longitude and 38.9072 latitude. The ST\_Distance function calculates the distance between the two points using the spherical mercator projection, which is why the coordinates are transformed using the ST\_Transform function. The result is returned as distance\_in\_meters.



3. Calculate areas of interest (specific to each group)

Code: WITH groups AS (

SELECT

CASE

WHEN water\_temperature >= 25 AND water\_temperature < 30 THEN 'Group A'

WHEN water\_temperature >= 20 AND water\_temperature < 25 THEN 'Group B'

ELSE 'Other'

END AS group\_name,

ST\_MakePoint(longitude, latitude) AS point\_geom

FROM water\_data

)

SELECT

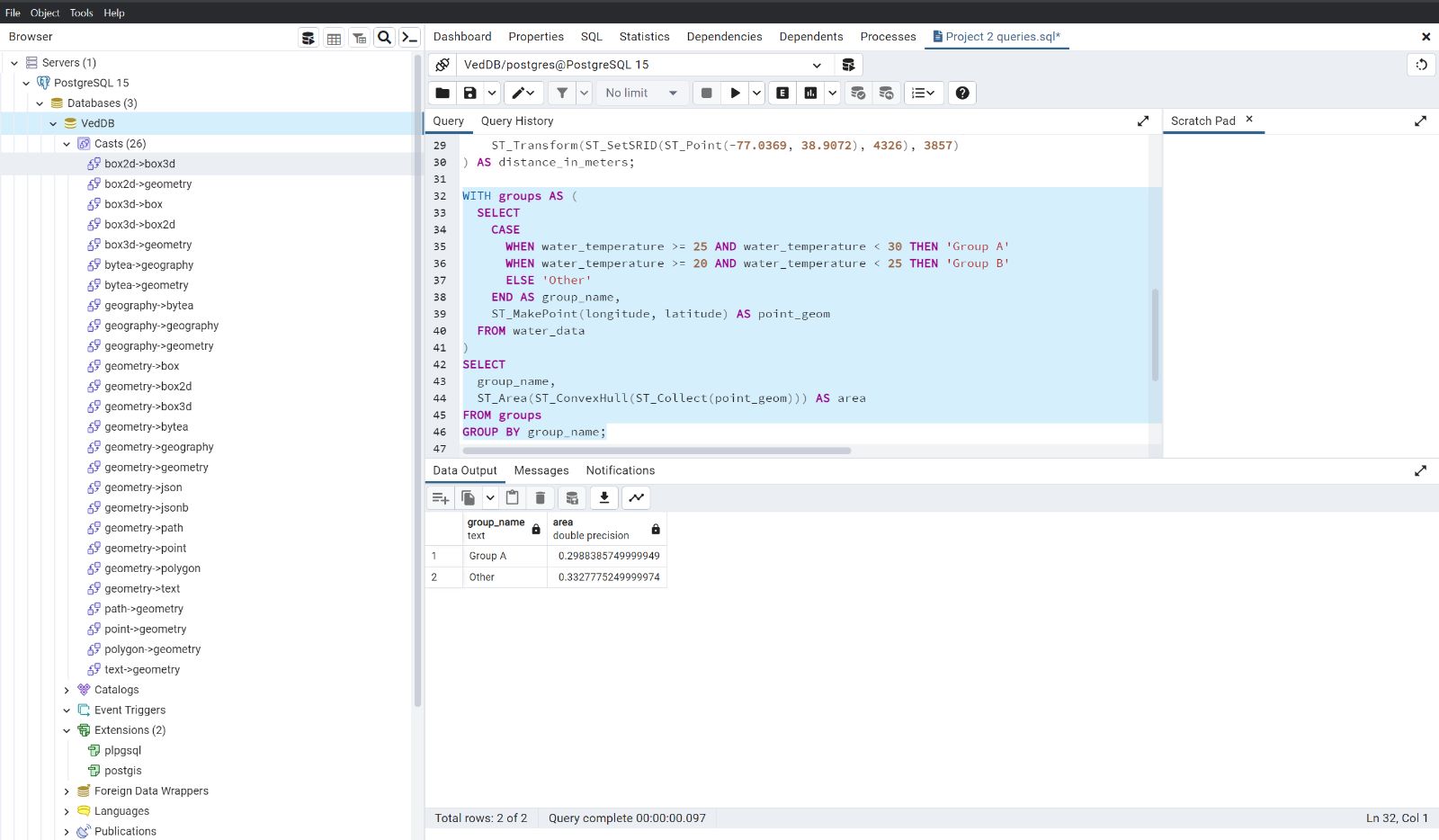
group\_name,

ST\_Area(ST\_ConvexHull(ST\_Collect(point\_geom))) AS area

FROM groups

GROUP BY group\_name;

This code creates groups of locations based on the water temperature of each location and calculates the area of interest for each group. First, it assigns each location to a group (Group A, Group B, or Other) based on the water temperature of that location. Then, it creates a point geometry for each location using the longitude and latitude values. Finally, it groups the locations by their assigned group and calculates the area of interest for each group using the convex hull of the point geometries of the locations in that group.



4. Analyze the queries.

We have analyzed the queries on Github.

5. Sorting and Limiting Executions.

Code: SELECT

Date,

Time,

Water\_Temperature

FROM

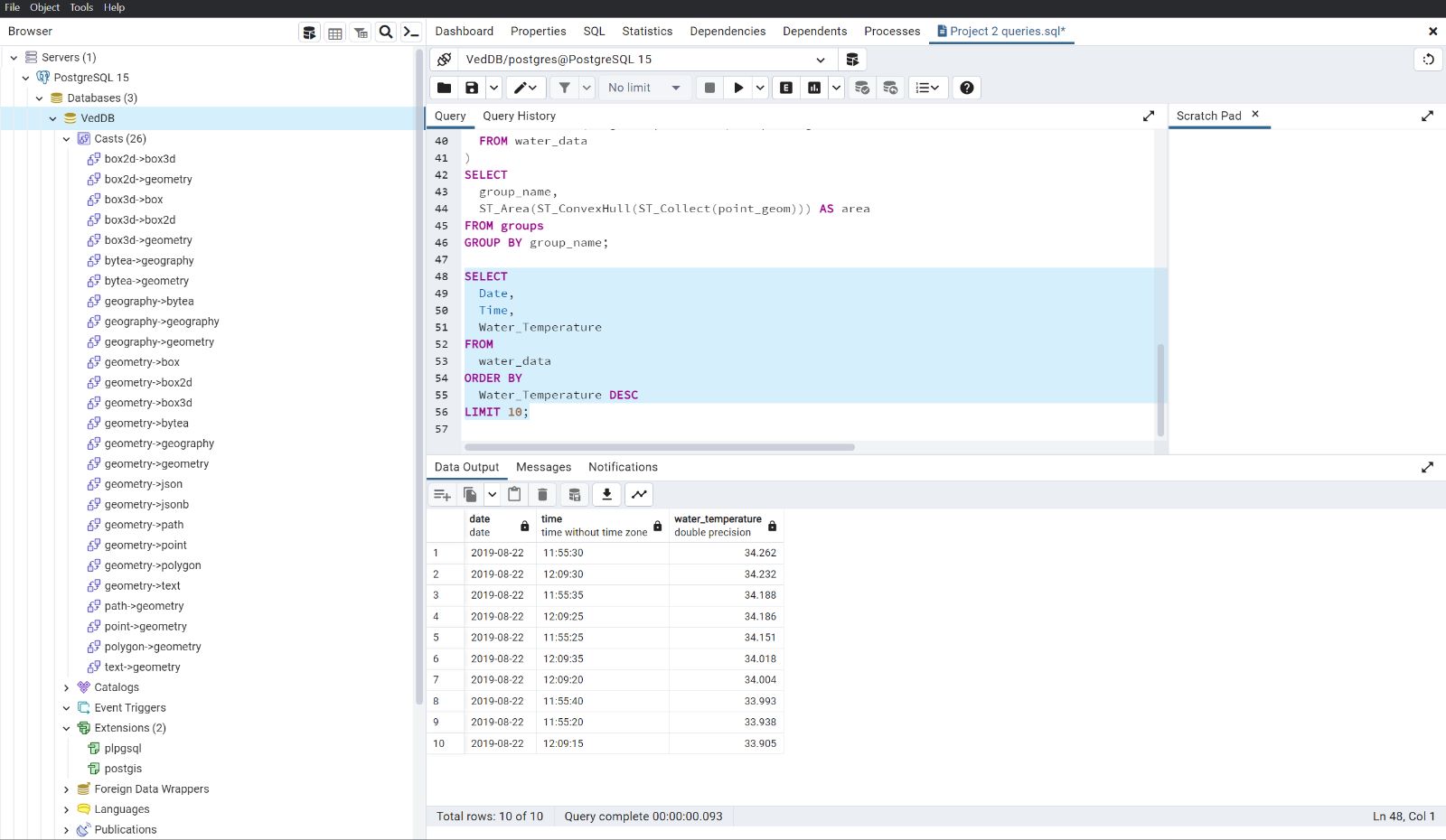
water\_data

ORDER BY

Water\_Temperature DESC

LIMIT 10;

This query selects the columns Date, Time, and Water\_Temperature from the table water\_data, sorts the results in descending order by Water\_Temperature, and returns only the top 10 rows using the LIMIT clause. Essentially, it returns the 10 rows with the highest water temperature values along with their corresponding dates and times. As you can see we have sorted the query by selecting Date and Time and then limiting it by 10 rows.



6. Optimize the queries to speed up execution time.

Code:

CREATE INDEX water\_temperature\_idx ON water\_data (Water\_Temperature);

SELECT

Date,

Time,

Water\_Temperature

FROM

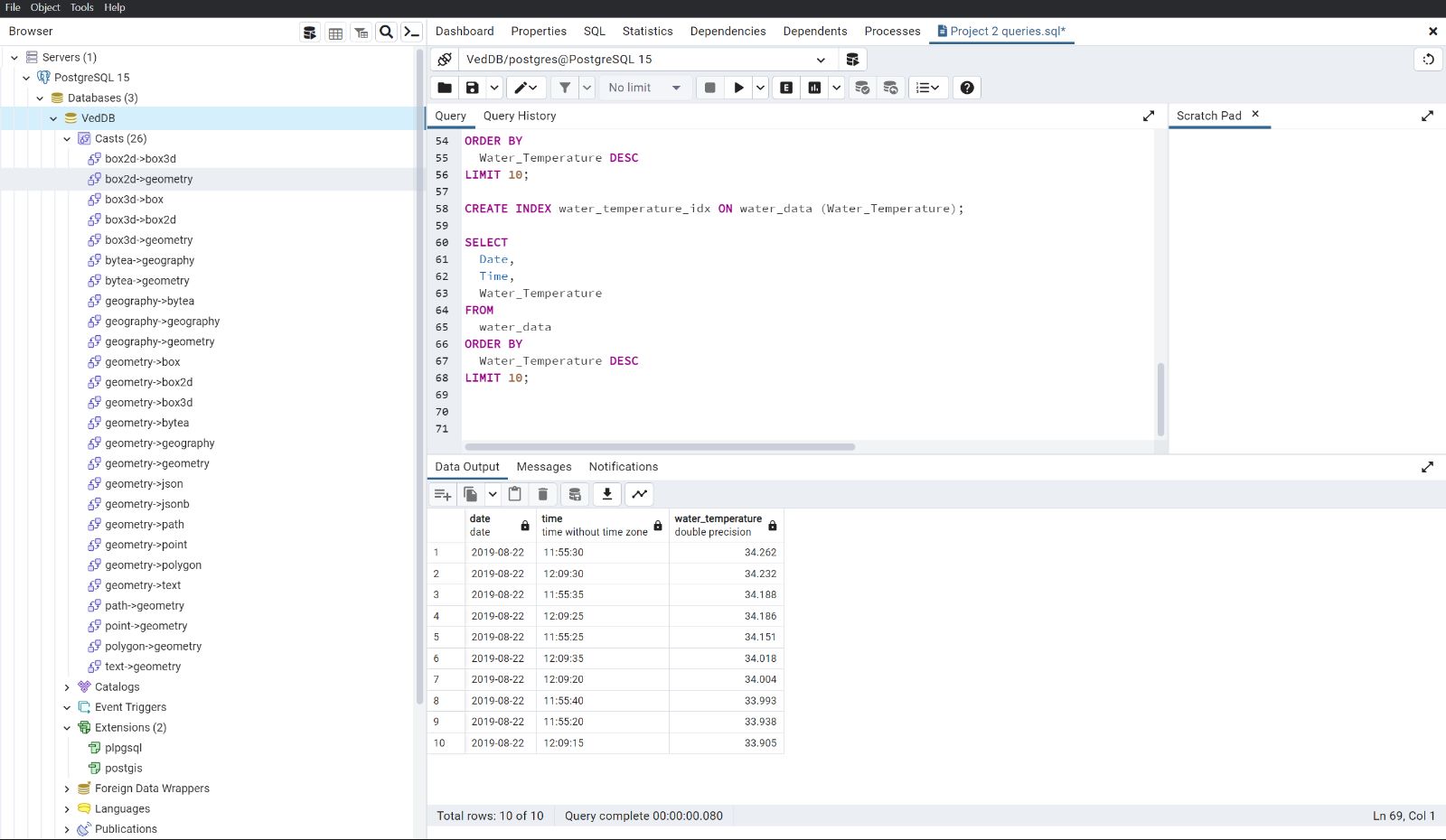
water\_data

ORDER BY

Water\_Temperature DESC

LIMIT 10;

To optimize this query, we have created an index on the Water\_Temperature column. This will allow the database to more efficiently sort the data based on the Water\_Temperature values, resulting in faster execution time.After creating the index, the query can be run again and should execute more quickly:



7. N-Optimization of queries.

Code: SELECT

water\_data.Date,

water\_data.Time,

water\_data.Water\_Temperature

FROM

water\_data

JOIN (

SELECT

Water\_Temperature

FROM

water\_data

ORDER BY

Water\_Temperature DESC

LIMIT 10

) top\_temps ON water\_data.Water\_Temperature = top\_temps.Water\_Temperature

ORDER BY

water\_data.Water\_Temperature DESC;

In this query, the subquery SELECT Water\_Temperature FROM water\_data ORDER BY Water\_Temperature DESC LIMIT 10 is used to identify the top 10 values of Water\_Temperature. We then join this subquery with the original water\_data table using the condition water\_data.Water\_Temperature = top\_temps.Water\_Temperature to retrieve the corresponding rows.

By using this approach, we can avoid sorting the entire water\_data table and instead only sort the 10 rows returned by the subquery. This can lead to significant performance improvements for large tables.

