Assignment 2: SparkSQL for Real-Time Taxicab

CSE 511: Data Processing at Scale – Fall 2025

Available: 09/24/2025 **Due Date**: 10/08/2025 11:59 PM **Points: 100**

Introduction & Background

A major peer-to-peer taxicab firm has hired your team to develop and **run multiple spatial queries** on their large database that **contains geographic data and real-time location data** of their customers. A spatial query is a special query supported by geodatabases and spatial databases. The queries differ from traditional SQL queries because they allow for using points, lines, and polygons. The spatial queries also consider the relationship between these geometries. Since the database is large and mostly unstructured, your client wants you to use a popular Big Data software application, SparkSQL. The goal of the assignment is to **extract data from this database** that will be used by your client for operational (day-to-day) and strategic-level (long-term) decisions.

Getting Started

This is an **individual** assignment. The following section covers some helpful information to get you started! You can use the **docker image we provided (highly recommended, everything is ready)** or install the software yourself.

- Installing Apache Spark and SparkSQL on your computer. You will use Apache Spark and SparkSQL in this assignment, so having them is necessary. The following URL will help you get started with the installation as well. https://spark.apache.org/docs/latest/
- You will also need to research Apache SparkSQL and spatial queries. The following are some helpful resources:
    - https://www.ibm.com/docs/en/ias?topic=toolkit-executing-spatial-query-in-spark
    - https://www.tutorialspoint.com/spark_sql/spark_sql_quick_guide.htm
- **We have also provided a docker image** to streamline the installation process. You can find details about the image in a later section. To get started with Docker, here are some helpful links:
    - https://www.docker.com/get-started/
    - https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-u buntu-20-04

Problem Description

You will **write two user-defined functions**, `ST_Contains` and `ST_Within,` in SparkSQL, and use them to **run** the following **four** spatial queries.

- Find the details of `ST_Contains` below.
    - **Inputs**: pointString: String, queryRectangle: String
    - **Output**: Boolean (true or false)
    - **Definition**:
        - You first need to parse the pointString (e.g., "-88.331492,32.324142") and queryRectangle (e.g., "-155.940114,19.081331,-155.618917,19.5307") to a format that you are comfortable with.
        - Then check whether the queryRectangle fully contains the point. Also, consider on-boundary points.

- Find the details of `ST_Within` below:
    - **Input**:s pointString1:String, pointString2:String, distance: Double
    - **Output**: Boolean (true or false)
    - **Definition**:
        - You first need to parse the pointString1 (e.g., "-88.331492,32.324142") and pointString2 (e.g., "-88.331492,32.324142") to a format that you are comfortable with. Then check whether the two points are within the given distance. Also, consider on-boundary points.
        - To simplify the problem, please assume all coordinates are in planar space and calculate their Euclidean distance.

Problem Statement

In this assignment, a rectangle, **R**, represents a geographical boundary in a town or city, and a set of points, **P**, represents customers who request taxi cab service using the client firm's app.

- **Range query**:
    - Given a query rectangle **R** and a set of points **P**, find all the points within **R**. You need to use the `'ST_Contains'` function in this query.
- **Range join query**:
    - Given a set of rectangles **R** and a set of points **P**, find all (point, rectangle) pairs such that the point is within the rectangle.
- **Distance query**:
    - Given a fixed point location **P** and distance **D** (in kilometres), find all points within distance **D** from **P**. You need to use the `'ST_Within'` function in this query.
- **Distance join query:**

- Given two sets of points $\mathbf{P_1}$ and $\mathbf{P_2}$, and a distance **D** (in kilometres), find all ($\mathbf{p_1}$, $\mathbf{p_2}$) pairs such that $\mathbf{p_1}$ is within a distance **D** from $\mathbf{p_2}$ (i.e., $\mathbf{p_1}$ belongs to $\mathbf{P_1}$ and $\mathbf{p_2}$ belongs to $\mathbf{P_2}$). You need to use the `'ST_Within'` function in this query.

How to run your code on Apache Spark using "spark-submit"

If you are using the Scala template, note that:

1. You **only have to replace the logic** (currently is "true") in all User-Defined Functions.
2. The main function in this template takes the **dynamic length of parameters** as follows:
   - Output file path (**Mandatory**): `result/output`
   - Range query data file path, query window:
     `rangequery src/resources/arealm.csv -155.940114,19.081331,-155.618917,19.5307`
   - Range join query data file path, range join query window data file path:
     `rangejoinquery src/resources/arealm.csv src/resources/zcta510.csv`
   - Distance query data file path, query point, distance:
     `distancequery src/resources/arealm.csv -88.331492,32.324142 10`
   - Distance join query data A file path, distance join query data B file path, distance:
     `distancejoinquery src/resources/arealm.csv src/resources/arealm.csv 10`
3. The number of queries and the order of queries in the input does not matter. The code template will detect the corresponding query and call it!
4. Two example datasets and a test case file and answer are available here.
   - `arealm10000` is a point dataset, and `zcta10000` is a rectangle dataset.
   - The test case file is `exampleinput`
   - The correct answer is `exampleanswer`
5. Here is an example that tells you how to submit your jar using "`spark-submit`"

```
spark-submit CSE511-assembly-0.1.0.jar result/output
rangequery src/resources/arealm10000.csv -93.63173,33.0183,-93.359203,33.219456
rangejoinquery src/resources/arealm10000.csv src/resources/zcta10000.csv
distancequery src/resources/arealm10000.csv -88.331492,32.324142 1
distancejoinquery src/resources/arealm10000.csv src/resources/arealm10000.csv 0.1
```

You can use provided example dataset to test your code. You can also use **NYC taxi trip dataset** to test your code. The auto-grading system will also run your code on different datasets.

How to debug your code in IDE

Using the Scala template

1. Use IntelliJ IDE or any other Scala IDE with a **Scala plug-in**
2. Define and Implement Functions `ST_Contains` and `ST_Within` in `SpatialQuery.scala`
3. append .master("local[*]") after .config("spark.some.config.option", "some-value") to tell IDE the master IP is localhost.
4. Sometimes, you may need to go to the `build.sbt` file and change % "provided" to % "compile" to debug your code in IDE.
5. Run your code in IDE

How to submit your code to Spark

If you are using the Scala template

1. Go to the assignment folder in the provided docker image
2. Run `sbt assembly`
3. Find the packaged jar in "`./target/scala-2.11/CSE511-assembly-0.1.0.jar`"
4. Submit the jar to Spark using the Spark command "spark-submit"
5. You must revert Steps 3 and 4 in "How to debug your code in IDE" and recompile your code before using spark-submit!!!

How to work on the assignment and test your code

You are provided with a **docker image** with all the packages (java, scala, spark) installed and ready to go.

- You will find the template code also in the docker. Path: `/root/cse511`
- Make the changes in the docker template. The file you need to update is `SpatialQuery.scala`
- Make sure your code can compile and package by using `sbt assembly`

Grading:

- The example input has already been provided to you in the docker image as well. The format for our input will be similar; however, test cases might be different. Make sure your code works with different use cases.
- Each query (Range, Rangejoin, Distance, Distancejoin) carries **equal weight of 25%** of grade points.
- Two example datasets and a test case file and answer are also available here for your reference.
    - `arealm10000` is a point dataset, and `zcta10000` is a rectangle dataset.

- The test case file is `exampleinput`
- The correct answer is `exampleanswer`

- You need to make sure your code can compile and package by entering sbt assembly. We will run the compiled package on our cluster directly using "**spark-submit**". If your code cannot compile and package, you will **not** receive any points

Submission Requirements & Guidelines

Assignment 2 is due on **10/08/2025 at 11:59 PM.** Submit the assignment following the below guidelines:

1. This is an **individual** assignment.
2. There is only 1 file (SpatialQuery.scala) you need to update
3. **What to submit on the canvas ?**
   On the canvas you need to upload a zip file containing the following:
   1. The **source code**. (`SpatialQuery.scala` you don't need to submit the entire template folder. Ensure your assignment can be compiled by entering **sbt assembly** and verifying everything beforehand.
   2. One **jar** file created using the command `sbt assembly`
4. **What is the Submission File nomenclature?**
   1. You **MUST** name your zip file as **Assignment2.zip**
   2. Failure to follow the naming nomenclature will fail the automated grading scripts, and you will be awarded **0** grade points.

Submission Policies

1. Late submissions will *not* be graded (unless you have verifiable proof of emergency). It is much better to submit partial work on time and get partial credit for your work than to submit late for no credit.
2. Every student needs to *work independently* on this exercise. We encourage high-level discussions among students to help each other understand the concepts and principles. However, a code-level discussion is prohibited, and plagiarism will directly lead to failure of this course. We will use anti-plagiarism tools to detect violations of this policy.