Assignment 3:

CSE 511: Data Processing at Scale – Fall 2025

Available: 10/15/2025 **Due Date**: 10/22/2025 11:59 PM **Points: 100**

Introduction & Background

In Assignment 2, you learned to run multiple spatial queries on the large geo-database of the taxi firm that hired you. Now, the firm has contacted you again to extend the solution. You now need to perform further analysis for them by locating passenger hot spots.

Problem Description

You are required to do spatial **hot spot analysis**. In particular, you need to complete two different hot spot analysis tasks.

- **Hot zone analysis**: This task will need to perform a range join operation on a rectangle dataset and a point dataset. For each rectangle, the number of points located within the rectangle will be obtained. The hotter rectangle means that it includes more points. So this task is to calculate the hotness of all the rectangles.
- **Hot cell analysis**: This task will focus on applying spatial statistics to spatio-temporal big data in order to identify statistically significant spatial hot spots using Apache Spark.

The topic of this project phase is inspired by ACM SIGSPATIAL GISCUP 2016. However, we have special requirements in this assignment which are different from the GIS Cup.

Please find the following references in the problem description:

- A detailed problem definition is available here
- Evaluation Instructions are available here

Problem Statement

As stated in the Problem Definition page, in this task, you are asked to implement a Spark program to calculate the Getis-Ord G_i^* statistic of NYC Taxi Trip datasets. The G_i^* statistic is a z-score. We call it "Hot cell analysis". To reduce the computation power need, we made the following changes:

- The input will be a monthly taxi trip dataset from 2009 - 2012. For example, `yellow_tripdata_2009-01_point.csv`, `yellow_tripdata_2010-02_point.csv`
- Each cell unit size is 0.01 * 0.01 in terms of latitude and longitude degrees.

- You should use 1 day as the Time Step size. The first day of the month is step 1. Every month has 31 days.
- You only need to consider the Pick-up Location.
- We don't use Jaccard similarity to check your answer. However, you don't need to worry about how to decide the cell coordinates because the code template generates cell coordinates. You just need to write the rest of the task.

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. Otherwise, you can find all the files here.

- You will find the template code also in the docker. Path: `/root/cse511`
- The main function/entrance is `cse511/Main`.scala file.
- DO NOT DELETE any existing code in the coding template unless you see this "YOU NEED TO CHANGE THIS PART"
- In the code template, for **Hot zone analysis**
    - You need to change HotzoneAnalysis.scala and HotzoneUtils.scala.
    - The template code has loaded the data and wrote the first step, the range join query, for you. Please finish the rest of the task.
    - The output DataFrame should be sorted by you according to "rectangle" string.
    - **Input format:** The input point data can be any small subset of NYC taxi dataset.
    - **Output format:** All zones with their count, sorted by "rectangle" string in ascending order.
- In the code template, for **Hot cell analysis**,
    - You need to change HotcellAnalysis.scala and HotcellUtils.scala.
    - The template code has loaded the data and decided the cell coordinate, x, y, and z and their min and max. Please finish the rest of the task.
    - The output DataFrame should be sorted by you according to G-score. The coding template will take the first 50 to output. DO NOT OUTPUT G-score.
    - **Input format:** The input point data is a monthly NYC taxi trip dataset (2009-2012) like "`yellow_tripdata_2009-01_point.csv`"
    - **Output format**: The coordinates of the top 50 hottest cells are sorted by their G score in descending order. Note, DO NOT OUTPUT G score.
      -7399,4075,15
      -7399,4075,29
      -7399,4075,22
- Example input and answer are placed in "`testcase`" folder in the provided.

Note:

- **Point data**: the input point dataset is the pickup point of New York Taxi trip datasets. The data format of this assignment is the original format of NYC taxi trip which is different from Assignment 2. Find the data in the .csv file: Yellow_tripdata_2009-01_point.csv
- **Zone data** (only for hot zone analysis): at "`src/resources/zone-hotzone`" in the provided docker template.

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 and** the number/order of tasks does not matter.
3. Input parameters:
   - Output file path (**Mandatory**): `result/output`
   - Task name: `hotzoneanalysis` or `hotcellanalysis`
   - If the task is `hotzoneanalysis` it requires two parameters
     1. NYC taxi data path
     2. Zone path
   - If the task is hotcellanalysis it requires one parameter
     1. NYC taxi data path
4. 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!
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
hotzoneanalysis src/resources/point-hotzone.csv src/resources/zone-hotzone.csv
hotcellanalysis src/resources/yellow_tripdata_2009-01_point.csv
```

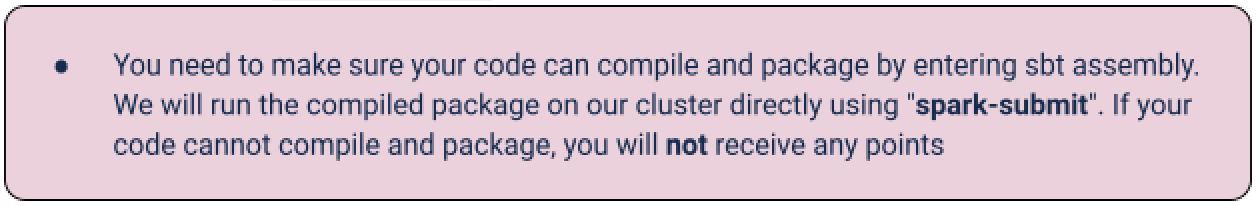
How to submit your code to Spark

If you are using the Scala template

1. Go to the project 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!!!

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 in different use cases.
- Each spatial hot spot analysis carries an equal weight of 50% of grade points.
- Two example datasets and a test case file and answer are also available in the docker for your reference. `/src/resources` and are also available here



- 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

1. This is an **individual** assignment.
2. There are 4 scala files you need to update
2. **What to submit on the canvas?**
    1. On the canvas, you need to upload a zip file containing the following:
        1. The **source code**. You don't need to submit the entire template folder. Ensure your project can compile by entering sbt assembly and verifying everything beforehand.
        2. One **jar** file was created using the command `sbt assembly`

3. **What is the Submission File nomenclature?**
    1. You **MUST** name your zip file as **Assignment3.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 *absolutely 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.