

CSE512-Project-Phase2-Requirement

Requirement

In Project Phase 2, you need to write two User Defined Functions `ST_Contains` and `ST_Within` in SparkSQL and use them to do four spatial queries:

- Range query: Use `ST_Contains`. Given a query rectangle `R` and a set of points `P`, find all the points within `R`.
- Range join query: Use `ST_Contains`. Given a set of Rectangles `R` and a set of Points `S`, find all (Point, Rectangle) pairs such that the point is within the rectangle.
- Distance query: Use `ST_Within`. Given a point location `P` and distance `D` in km, find all points that lie within a distance `D` from `P`
- Distance join query: Use `ST_Within`. Given a set of Points `S1` and a set of Points `S2` and a distance `D` in km, find all (`s1`, `s2`) pairs such that `s1` is within a distance `D` from `s2` (i.e., `s1` belongs to `S1` and `s2` belongs to `S2`).

A Scala SparkSQL code template is given. You must start from the template. The main code is in "SparkSQLExample.scala"

The detailed requirements are as follows:

1. `ST_Contains`

Input: `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. Consider on-boundary point.

2. `ST_Within`

Input: `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. Consider on-boundary point. To simplify the problem, please assume all coordinates are on a planar space and calculate their Euclidean distance.

3. Use Your UDF in SparkSQL

The code template has loaded the original data (point data, arealm.csv, and rectangle data, zcta510.csv) into DataFrame using tsv format. You don't need to worry about the loading phase.

Range query:

```
select *
from point
where ST_Contains(point._c0, '-155.940114,19.081331,-155.618917,19.5307')
```

Range join query:

```
select *
from rectangle,point
where ST_Contains(rectangle._c0,point._c0)
```

Distance query:

```
select *
from point
where ST_Within(point._c0, '-88.331492,32.324142',10)
```

Distance join query:

```
select *
from point p1, point p2
where ST_Within(p1._c0, p2._c0, 10)
```

4. 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 Function.
2. The main function in this template takes 21 parameters as follows:
 - Output file path: /Users/ubuntu/Downloads/output.csv
 - Range query data file path, query window: /Users/ubuntu/Downloads/arealm.csv
-155.940114,19.081331,-155.618917,19.5307
 - Range join query data file path, range join query window data file path:
/Users/ubuntu/Downloads/arealm.csv /Users/ubuntu/Downloads/zcta510.csv
 - Distance query data file path, query point, distance: /Users/ubuntu/Downloads/arealm.csv
-88.331492,32.324142 10
 - Distance join query data A file path, distance join query data B file path, distance:
/Users/ubuntu/Downloads/arealm.csv /Users/ubuntu/Downloads/arealm.csv 10
 - Range query data file path, query window: /Users/ubuntu/Downloads/arealm.csv
-155.940114,19.081331,-155.618917,19.5307

- Range join query data file path, range join query window data file path:
`/Users/ubuntu/Downloads/arealm.csv /Users/ubuntu/Downloads/zcta510.csv`
 - Distance query data file path, query point, distance: `/Users/ubuntu/Downloads/arealm.csv`
`-88.331492,32.324142 10`
 - Distance join query data A file path, distance join query data B file path, distance:
`/Users/ubuntu/Downloads/arealm.csv /Users/ubuntu/Downloads/arealm.csv 10`
3. Two example datasets are put in "src/resources" folder. arealm is a point dataset and zcta510 is a rectangle dataset. You can use them to test your code but eventually you must run your code on NYC taxi trip dataset. Our auto-grading system will also run your code on NYC taxi trip data.
4. Here is an example that tells you how to submit your jar using "spark-submit"

```
./bin/spark-submit
~/GitHub/CSE512-Project-Phase3-Template/target/scala-2.11/CSE512-Project-Phase2-Template-assembly-0.1.0.jar ~/Downloads/output.csv ~/Downloads/arealm_small.csv
-155.940114,19.081331,-155.618917,19.5307 ~/Downloads/arealm_small.csv
~/Downloads/zcta510_small.csv ~/Downloads/arealm_small.csv -88.331492,32.324142 10
~/Downloads/arealm_small.csv ~/Downloads/arealm_small.csv 10
~/Downloads/arealm_small.csv -155.940114,19.081331,-155.618917,19.5307
~/Downloads/arealm_small.csv ~/Downloads/zcta510_small.csv
~/Downloads/arealm_small.csv -88.331492,32.324142 10 ~/Downloads/arealm_small.csv
~/Downloads/arealm_small.csv 10
```

5. Vocareum (submission site)

In Vocareum, all groups share a single but powerful cluster.

Vocareum is for submission only. After uploading your jar, directly click "submit".

Do not test your code here. You have limited submission times. Every submission impacts your grade!

Website

<https://www.vocareum.com/>

Submission

1. Submit your code on Vocareum website in order to get your grade.
2. Submit your project source code onto Blackboard in a compress zip file of "cse512-phase2-GROUPNAME" for plagiarism detection. Note that: 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".
3. If your code cannot compile and package, you will not receive any points.

How to debug your code in IDE

If you are using the Scala template

1. Use IntelliJ Idea with Scala plug-in or any other Scala IDE.
2. Replace the logic of User Defined Functions `ST_Contains` and `ST_Within` in `SparkSQLExample.scala`.
3. Append `.master("local[*]")` after `.config("spark.some.config.option", "some-value")` to tell IDE the master IP is localhost.
4. In some cases, you may need to go to "build.sbt" file and change `% "provided"` to `% "compile"` in order 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 project root folder
2. Run `sbt assembly`. You may need to install sbt in order to run this command.
3. Find the packaged jar in `./target/scala-2.11/CSE512-Project-Phase2-Template-assembly-0.1.0.jar`
4. Submit the jar to Spark using Spark command `./bin/spark-submit`