Here is the Python code that you will need to run in the Databricks notebook shown in Figure 21-9

stationData = spark.sql("select \* from station\_data")

tripData = spark.sql("select \* from trip\_data")

Here is the Python code that you will need to run in the Databricks notebook shown in Figure 21-10:

stationVertices = stationData.withColumnRenamed("station\_name", "station\_id").distinct()

tripEdges = tripData\

.withColumnRenamed("Start Location", "src")\

.withColumnRenamed("End Location", "dst")

Run the following code to build a GraphFrame object which will represent your graph using the edge and vertex dataframes that were defined earlier.

from graphframes import GraphFrame

stationGraph = GraphFrame(stationVertices, tripEdges)

stationGraph.cache()

Here is the Python code that you will need to run in the Databricks notebook shown in Figure 21-11:

from graphframes import GraphFrame

stationGraph = GraphFrame(stationVertices, tripEdges)

stationGraph.cache()

Here is the Python code that you will need to run in the Databricks notebook shown in Figure 21-12:

print ("Total Number of Stations:" + str(stationGraph.vertices.count()))

print ("Total Number of Trips in Graph:" + str(stationGraph.edges.count()))

print ("Total Number of Trips in Original Data:" + str(tripData.count()))

Here is the Python code that you will need to run in the Databricks notebook shown in Figure 21-13:

from pyspark.sql.functions import desc

stationGraph.edges.groupBy("src", "dst").count().orderBy(desc("count")).show(10)

Here is the Python code that you will need to run in the Databricks notebook shown in Figure 21-14:

stationGraph.edges\

.where("src = 'Cicero-Forest Park' OR dst = 'Cicero-Forest Park'")\

.groupBy("src", "dst").count()\

.orderBy(desc("count"))\

.show(10)

Here is the Python code that you will need to run in the Databricks notebook shown in Figure 21-16:

motifs = stationGraph.find("(a)-[ab]->(b); (b)-[bc]->(c); (c)-[ca]->(a)")

motifs.show()

Once the motif is added to a dataframe, it can be used in the following query, for example, to find the shortest time from station (a) to (b) to (c) back to (a) by leveraging the timestamps.

from pyspark.sql.functions import expr

motifs.selectExpr("\*",

"to\_timestamp(ab.`Start Date`, 'MM/dd/yyyy HH:mm') as abStart",

"to\_timestamp(bc.`Start Date`, 'MM/dd/yyyy HH:mm') as bcStart",

"to\_timestamp(ca.`Start Date`, 'MM/dd/yyyy HH:mm') as caStart")\

.where("ca.`Station\_Name` = bc.`Station\_Name`").where("ab.`Station\_Name` = bc.`Station\_Name`")\

.where("a.id != b.id").where("b.id != c.id")\

.where("abStart < bcStart").where("bcStart < caStart")\

.orderBy(expr("cast(caStart as long) - cast(abStart as long)"))\

.selectExpr("a.id", "b.id", "c.id", "ab.`Start Date`", "ca.`End Date`")

.limit(1).show(1, False)

Here is the Python code that you will need to run in the Databricks notebook shown in Figure 21-18:

from pyspark.sql.functions import desc

ranks = stationGraph.pageRank(resetProbability=0.15, maxIter=10)

ranks.vertices.orderBy(desc("pagerank")).select("station\_id", "pagerank").show(10)

Here is the Python code that you will need to run in the Databricks notebook shown in Figure 21-20:

inDeg = stationGraph.inDegrees

inDeg.orderBy(desc("inDegree")).show(5, False)

Here is the Python code that you will need to run in the Databricks notebook shown in Figure 21-21:

outDeg = stationGraph.outDegrees

outDeg.orderBy(desc("outDegree")).show(5, False)

Here is the Python code that you will need to run in the Databricks notebook shown in Figure 21-22:

degreeRatio = inDeg.join(outDeg, "id")\

.selectExpr("id", "double(inDegree)/double(outDegree) as degreeRatio")

degreeRatio.orderBy(desc("degreeRatio")).show(10, False)

degreeRatio.orderBy("degreeRatio").show(10, False)

Here is the Python code that you will need to run in the Databricks notebook shown in Figure 21-23:

stationGraph.bfs(fromExpr="station\_id = 'Belmont-North Main'",

toExpr="station\_id = 'Cicero-Forest Park'", maxPathLength=2).show(10)

Here is the Python code that you will need to run in the Databricks notebook shown in Figure 21-25:

spark.sparkContext.setCheckpointDir("/tmp/checkpoints")

minGraph = GraphFrame(stationVertices, tripEdges.sample(False, 0.1))

cc = minGraph.connectedComponents()

cc.where("component != 0").show()

Here is the Python code that you will need to run in the Databricks notebook shown in Figure 21-26:

scc = minGraph.stronglyConnectedComponents(maxIter=3)

scc.groupBy("component").count().show()