Neo4j Assignment

Task 1:

Code: -

```
CALL gds.graph.create(
  'neo4j-communities_7892',
  'zone',
  {
    CONNECTS: {
      type: 'CONNECTS',
      orientation: 'UNDIRECTED',
      properties: 'trips'
    }
  }
)
```

This will create the Graph projection of type UNDIRECTED. Where the name the graph projection as c2059666-communities, weighted by the trips property in :CONNECTS type of relationships.

Code: -

```
CALL gds.louvain.stats(‘c2059666-communities’)
YIELD communityCount
```

This code will Report the number of communities using the stats mode,

Code: -

```
CALL gds.louvain.stream(‘c2059666-communities’, { relationshipWeightProperty: 'trips' })

YIELD nodeId, communityId

RETURN gds.util.asNode(nodeId).id AS zone_id,communityId AS community_id
```

Now in stream mode, return the id and community properties of each zone. he results of running the algorithm in stream as a CSV file with two columns named zone_id, community_id.

Task 2:

Code: -

```
CALL gds.graph.create(

  'neo4j-centrality_789',

  'zone',

  'CONNECTS',

  {

    relationshipProperties: 'trips'

  }

)
```

Directed graph projection with name the graph projection as c2059666-centrality with

Damping factor: 0.75 and Weighted by the trips property in :CONNECTS type of relationships.

Code: -

```
CALL gds.pageRank.stats(‘c2059666-centrality’, {

  dampingFactor: 0.75,

  relationshipWeightProperty: 'trips'
})
YIELD centralityDistribution
RETURN centralityDistribution.max AS MAX , centralityDistribution.min AS MIN
-----------------------------------------------------------------------------------------------------------------
```

This will report the maximum and minimum centrality score using the stats mode.

Code: -

```
-----------------------------------------------------------------------------------------------------------------
CALL gds.pageRank.stream(‘c2059666-centrality’, {
  dampingFactor: 0.75,
  relationshipWeightProperty: 'trips'
})
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).id AS zone_id, score AS centrality_score
ORDER BY score DESC
-----------------------------------------------------------------------------------------------------------------
```

In stream mode, return the id, centrality properties of each zone (in this order),

Export the results of running the algorithm in stream as a CSV file with two columns named zone_id and centrality_score

Task 3:

Code: -

1) Including zones in the 'Manhattan' borough.

```
-----------------------------------------------------------------------------------------------------
MATCH (n:zone)
with n order by n.centrality desc
with n.community as class,collect({id:n.id, score:n.centrality}) as listt
UNWIND listt[0..3] AS l
```

```
return l.id as zone_id ,class as community_id

order by class
```

Using the available zone properties community and centrality we will return two columns: zone_id and community_id.

Code: -

2) Excluding zones in the 'Manhattan' borough.

```
MATCH (n:zone)-[r:IN]->(b:borough) WHERE b.name <> 'Manhattan' with n order by n.centrality desc

with n.community as class,collect({id:n.id, score:n.centrality}) as listt

UNWIND listt[0..3] AS l

return l.id as zone_id ,class as community_id

order by class.
```

Using the available zone properties community and centrality we will return two columns: zone_id and community_id.