```
database: 5.3.0
GDS: 2.4.6

//delete
MATCH (n) DETACH DELETE n

// drop project
CALL gds.graph.drop('Transport')

//Step 1: Transport nodes
WITH "https://github.com/neo4j-graph-analytics/book/raw/master/data/" AS base
WITH base + "transport-nodes.csv" AS uri
LOAD CSV WITH HEADERS FROM uri AS row
MERGE (place:Place {name:row.id})
SET place.latitude = toFloat(row.latitude),
place.longitude = toFloat(row.latitude),
place.population = toInteger(row.population)

//Step 2: Transport rels
WITH "https://github.com/neo4j-graph-analytics/book/raw/master/data/" AS base
WITH base + "transport-relationships.csv" AS uri
LOAD CSV WITH HEADERS FROM uri AS row
MATCH (origin:Place {name: row.src})
MATCH (destination:Place {name: row.dst})
MERGE (origin)-[:EROAD {distance: toInteger(row.cost)}]->(destination)

//Step 3: show all
MATCH (n) RETURN n

//Step 4: Project
CALL gds.graph.project(
    'Transport',
    'Place',
    {EROAD: {orientation: 'UNDIRECTED'}}
)


//Step 5: BFS
MATCH (source:Place{name:"Ipswich"}), (target:Place{name:'Den Haag'})
CALL gds.bfs.stream('Transport', {sourceNode:source, targetNodes:target})
YIELD path
RETURN path

//Step 6: DFS
MATCH (source:Place{name:"Ipswich"}), (target:Place{name:'Den Haag'})
CALL gds.dfs.stream('Transport', {sourceNode:source, targetNodes:target})
YIELD path
RETURN path

//Step 7: project for A*
CALL gds.graph.project(
    'Transport',
    'Place',
    {EROAD: { orientation: 'UNDIRECTED' } },
    {relationshipProperties: 'distance',
    nodeProperties: ['latitude', 'longitude']}
)

//Step 8: A*
MATCH (source:Place {name: 'Ipswich'}), (target:Place {name: 'Utrecht'})
CALL gds.shortestPath.astar.stream('Transport', {
    sourceNode: source,
    targetNode: target,
    latitudeProperty: 'latitude',
    longitudeProperty: 'longitude',
    relationshipWeightProperty: 'distance'
})
YIELD index, sourceNode, targetNode, totalCost, nodeIds, costs, path
RETURN
```

```
    index,
    gds.util.asNode(sourceNode).name AS sourceNodeName,
    gds.util.asNode(targetNode).name AS targetNodeName,
    totalCost,
    [nodeId IN nodeIds | gds.util.asNode(nodeId).name] AS nodeNames,
    costs,
    nodes(path) as path
ORDER BY index

//Step 9: MST
MATCH (n:Place{name: 'Ipswich'})
CALL gds.beta.spanningTree.write(
    'Transport',
    {sourceNode: id(n),
    relationshipWeightProperty: 'distance',
    writeProperty: 'writeCost',
    writeRelationshipType: 'MINST'}
)
YIELD preProcessingMillis, computeMillis, writeMillis, effectiveNodeCount
RETURN preProcessingMillis, computeMillis, writeMillis, effectiveNodeCount;

//Step 10:display the MST
MATCH (p1:Place)-[m:MINST]->(p2:Place)
RETURN p1, m, p2

//Step 11: random walk
MATCH (source:Place {name:"London"})
CALL gds.randomWalk.stream(
    'Transport',
    {sourceNodes: [id(source)], walkLength: 5, walksPerNode: 1})
YIELD nodeIds, path
RETURN nodeIds, [nodeId IN nodeIds | gds.util.asNode(nodeId).name] AS nodeNames, path;
```