CALL gds.graph.create(

'Graph\_student',

['Student','ReadingGroup','WritingGroup','MathGroup','Race','PrentalLevelOfEducation','Race'],

{HAS\_MATH\_GRADE: {

type: 'HAS\_MATH\_GRADE'},

HAS\_READING\_GRADE: {

type: 'HAS\_MATH\_GRADE'},

HAS\_WRIING\_GRADE:{

type:'HAS\_WRITING\_GRADE'},

HAS:{type:'HAS'},

BELONG:{

type:'BELONG'} });

CALL gds.wcc.stream('Graph\_student')

YIELD nodeId, componentId

RETURN gds.util.asNode(nodeId).student\_id AS name, componentId

ORDER BY componentId, name

CALL gds.pageRank.stream('Graph\_student')

YIELD nodeId, score

RETURN gds.util.asNode(nodeId).group AS name, score

ORDER BY score DESC, name

LIMIT 4

CALL gds.nodeSimilarity.stream('Graph\_student')

YIELD node1,node2, similarity

RETURN gds.util.asNode(node1).student\_id AS Student1, gds.util.asNode(node2).student\_id AS Student2,similarity

ORDER BY similarity DESCENDING, Student1,Student2;