//Create Post Label

USING PERIODIC COMMIT

LOAD CSV WITH HEADERS

FROM "file:///Posts1.csv" AS line

CREATE(p:Post{Id:toInteger(line.Id),PostTypeId:toInteger(line.PostTypeId),

AcceptedAnswerId:toInteger(line.AcceptedAnswerId),CreationTime:toInteger(line.CreationTime),

OwnerUserId:toInteger(line.OwnerUserId),LastEditorUserId:toInteger(line.LastEditorUserId),

LastEditTime:toInteger(line.LastEditTime),Title:line.Title,Tags:line.Tags,

ParentId:toInteger(line.ParentId)})

//Set Post property

MATCH (p:Post)

SET p.Tags=split(p.Tags,',')

//Create User Label

USING PERIODIC COMMIT

LOAD CSV WITH HEADERS

FROM "file:///Users.csv" AS row

CREATE(u:User{id:toInteger(row.Id),creationdate:row.CreationDate,

displayname:row.DisplayName,upvotes:toInteger(row.UpVotes),

downvotes:toInteger(row.DownVotes)})

//Create Vote Label

USING PERIODIC COMMIT

LOAD CSV WITH HEADERS

FROM "file:///Votes1.csv" AS line

CREATE(v:Vote{Id:toInteger(line.Id),PostId:toInteger(line.PostId),

VoteTypeId:toInteger(line.VoteTypeId),CreationTime:toInteger(line.CreationTime),

CreationDate:line.CreationDate,UserId:toInteger(line.UserId),

BountyAmount:toInteger(line.BountyAmount)})

//Create OWNED relationship

MATCH (p:Post),(u:User)

WHERE p.OwnerUserId = u.id

CREATE (p)<-[:OWNED]-(u)

//Create EDITED relationship

MATCH (p:Post),(u:User)

WHERE p.LastEditorUserId = u.id

CREATE (p)<-[:EDITED]-(u)

//Create ANSWERED relationship

MATCH (p1:Post),(p2:Post)

WHERE p2.ParentId=p1.Id

CREATE (p2)-[a:ANSWERED]->(p1)

//Create ACCEPTED

MATCH (p1:Post),(p2:Post)

WHERE p1.AcceptedAnswerId=p2.Id

CREATE (p1)-[a:ACCEPTED{time:p2.CreationTime-p1.CreationTime}]->(p2)

//Create NOT\_ACCEPTED

MATCH (p1:Post)-[:ANSWERED]->(p2:Post)

WHERE NOT EXISTS(p2.AcceptedAnswerId) OR p1.Id<>p2.AcceptedAnswerId

CREATE (p1)<-[:NOT\_ACCEPTED]-(p2)

//UPVOTED

MATCH (p:Post),(v:Vote)

WHERE v.PostId=p.Id AND v.VoteTypeId=2

CREATE (v)-[:UPVOTED{CreationTime:v.CreationTime}]->(p)

//DECIDED

MATCH (p:Post),(v:Vote)

WHERE v.PostId=p.Id AND v.VoteTypeId=1

CREATE (v)<-[:DECIDED]-(p)

//Set Post property

MATCH (p:Post)-[d:DECIDED]-(v:Vote)

SET p.DecisionDay=v.CreationTime

//Create INVOLVED\_IN relationship

MATCH (p1:Post)<-[:ANSWERED]-(p2:Post)

OPTIONAL MATCH (u1:User)-[:EDITED]-(p1)

OPTIONAL MATCH (u2:User)-[:OWNED]-(p1)

OPTIONAL MATCH (u3:User)-[:EDITED]-(p2)

OPTIONAL MATCH (u4:User)-[:OWNED]-(p2)

WHERE EXISTS(u1.id) OR EXISTS(u2.id) OR EXISTS(u3.id) OR EXISTS(u4.id)

WITH p1,(collect(u1)+collect(u2)+collect(u3)+collect(u4)) as users

UNWIND users as user

WITH p1,user

MERGE (p1)<-[:INVOLVED\_IN]-(user)

//query 1

MATCH (p:Post)-[:INVOLVED\_IN]-(u:User)

WHERE p.Id=1

RETURN p,u

//query 2

MATCH (p:Post)

WHERE 'usa' in p.Tags

RETURN p.Title,p.ViewCount,p.Tags

ORDER BY p.ViewCount DESC

LIMIT 1

//query 3

MATCH (p1:Post)-[a:ACCEPTED]->(p2:Post)

UNWIND p1.Tags AS Tag

WITH Tag,p1,a

WHERE Tag='usa'

RETURN Tag,p1,a.time ORDER BY a.time LIMIT 1

//query 4

MATCH (p1:Post)-[a:ANSWERED]-(p2:Post)

MATCH (p1)-[:OWNED]-(o1)

MATCH (p2)-[:OWNED]-(o2)

WHERE 1368004750 < p1.CreationTime < 1368008000

OR 1368004750 < p2.CreationTime < 1368008000

UNWIND p1.Tags as Tag

WITH Tag,p1,o1,o2

RETURN DISTINCT Tag,(count(o1)+count(o2)) as num\_users

ORDER BY num\_users DESC LIMIT 5

//query 5

MATCH (p1:Post)-[a:ACCEPTED]->(p2:Post)-[:OWNED]-(u:User)

WHERE 'data-request' in p1.Tags

RETURN u,count(u)/2 as num\_accepted

ORDER BY num\_accepted DESC LIMIT 1

MATCH (p2:Post)-[:OWNED]-(u:User{id:33})

MATCH (p1:Post)-[a:ACCEPTED]->(p2:Post)

WHERE 'data-request' in p1.Tags

RETURN DISTINCT p1

//query 6

// find the potential answerer

MATCH (p1:Post)-[a:ACCEPTED]->(p2:Post)-[:OWNED]-(u:User)

UNWIND p1.Tags as Tag

WITH Tag,u,count(u) as num\_accepted

WHERE num\_accepted>10

RETURN DISTINCT Tag,u.id, num\_accepted ORDER BY num\_accepted DESC

//recommended 5 unanswered question to him

MATCH (p1:Post)

WHERE p1.AcceptedAnswerId=0 AND 'usa' in p1.Tags

RETURN p1 ORDER BY p1.CreationTime DESC LIMIT 5

//query 7

// question a

MATCH (p1:Post)-[:ACCEPTED]-(p2:Post)-[up:UPVOTED]-(v:Vote)

WHERE p1.Score >= 30 AND up.CreationTime > p2.DecisionDay

RETURN DISTINCT p1 as question,p2 as accepted\_answer, count(v)\*100/p2.Score as percentage ORDER BY percentage DESC LIMIT 1

//question b

MATCH (p3:Post)-[:ACCEPTED]-(p1:Post)-[:NOT\_ACCEPTED]-(p2:Post)-[up:UPVOTED]-(v:Vote)

WHERE p1.Score>30 AND up.CreationTime > p3.DecisionDay AND p2.Score>0

RETURN DISTINCT p1,count(v) as upvotes\_after,sum(p2.Score) as upvotes\_total,count(v)\*100/sum(p2.Score) as percentage ORDER BY percentage DESC LIMIT 1

//query 8

//solve the problem easily after build relationship

MATCH (u1:User{id:24})-[:INVOLVED\_IN]->(p1:Post)

MATCH (u2:User)-[:INVOLVED\_IN]->(p1)

RETURN u2,count(u2) as num\_cooperation ORDER BY num\_cooperation DESC LIMIT 5