# HOMEWORK 4

Dipak Bange

Part 1:

CSV data: [Netflix Movies and Tv Shows](https://www.kaggle.com/datasets/shivamb/netflix-shows)

Description: This data contains the movies/TV shows details of Netflix. It has actors, directors, languages, countries etc data. This data also has descriptions and ratings of the Movies/TV shows. It also has the categories of the shows and movies.

Arrows Schema:

A picture containing diagram

Description automatically generated

Arrows Schema description:

In this schema, we have movies Node where all details of movies are stored. Then Created three new nodes Person, Category, Country. Added relationships like Directed, Acted, In\_category and Location.

Part 2:

LOAD CSV:

*LOAD CSV WITH HEADERS FROM "file:///D:/Dipak/ms\_docx\_aug\_22/spring\_23/ADT/adt-sping23/week\_12/HW/netflix\_titles.csv" AS row*

*CREATE (m:Movie {*

*id: row.show\_id,*

*title: row.title,*

*director : row.director,*

*country : row.country,*

*date\_str : row.date\_added,*

*release\_year : row.release\_year,*

*rating : row.rating,*

*duration : row.duration,*

*listed\_in : row.listed\_in,*

*description : row.description,*

*cast: row.cast,*

*year : row.year,*

*month : row.month,*

*day : row.day,*

*type : row.type\_movie*

*})*

Nodes & Relationships:

*- Create Nodes for Person/Actor, relationship with Movie(ACTED\_IN, DIRECTED)*

*MATCH (m:Movie)*

*WHERE m.cast IS NOT NULL*

*WITH m*

*UNWIND split(m.cast, ',') AS actor*

*MERGE (p:Person {name: trim(actor)})*

*MERGE (p)-[r:ACTED\_IN]->(m);*

*-- Create Nodes for Category, relationship with Movie(IN\_CATEGORY)*

*MATCH (m:Movie)*

*WHERE m.listed\_in IS NOT NULL*

*WITH m*

*UNWIND split(m.listed\_in, ',') AS category*

*MERGE (c:Category {name: trim(category)})*

*MERGE (m)-[r:IN\_CATEGORY]->(c);*

*-- Create Nodes for Type*

*MATCH (m:Movie)*

*WHERE m.type IS NOT NULL*

*WITH m*

*MERGE (t:Type {type: m.type})*

*MERGE (m)-[r:TYPED\_AS]->(t);*

*-- create countries nodes*

*MATCH (m:Movie)*

*WHERE m.country IS NOT NULL*

*MERGE (c:Country {name: trim(m.country)})*

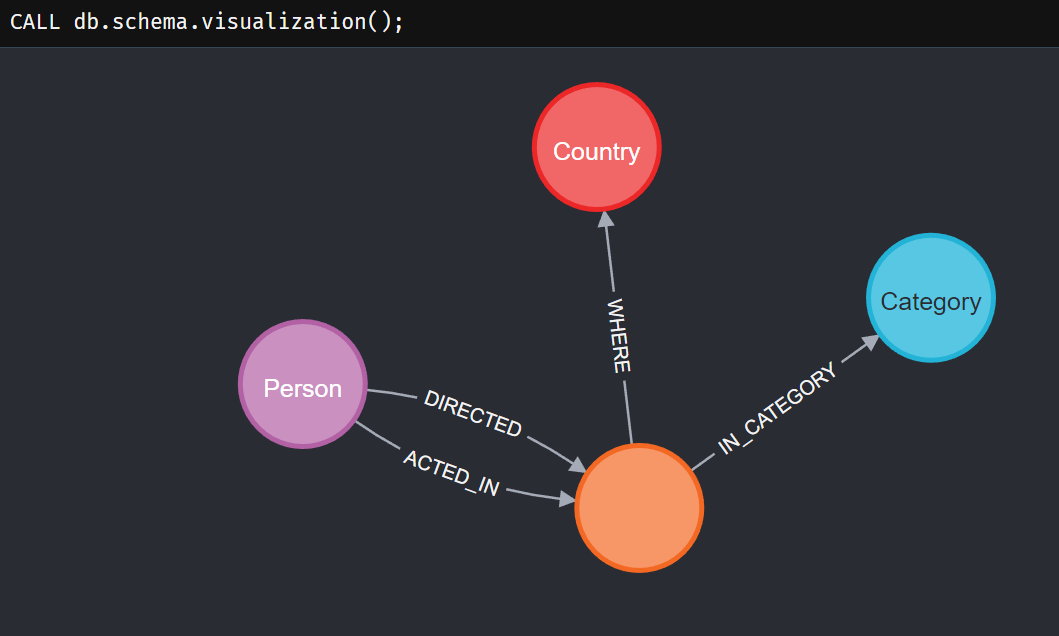
*MERGE (m)-[:WHERE]->(c);*

Description: In this cypher query, create 4 nodes as described in ARROW schema from CSV. Also created Relationship with Movies node with other three nodes. Also, added respective properties in Nodes.

Also 1st query loads data into Neo4j.

**Part 3:**

Schema:



Schema Description: Movies is the main node and has relations with Person(actor and director relationships), Country(where movie is produced), category( what category movie belongs to can be multiple).

Cypher queries:

-- Find movies with Brad Pitt and Brad Anderson

MATCH (cs:Person { name: 'Brad Anderson' }),(ms:Person { name: 'Brad Pitt' }), p = shortestPath((cs)-[:ACTED\_IN|DIRECTED\*]-(ms))

WHERE length(p)> 1

RETURN p;

-- Find movies with Adam Sandler and Jennifer Aniston

MATCH (cs:Person { name: 'Adam Sandler' }),(ms:Person { name: 'Jennifer Aniston' }), p = shortestPath((cs)-[:ACTED\_IN|DIRECTED\*]-(ms))

WHERE length(p)> 1

RETURN p;

-- top 5 actor with most movies

MATCH (p:Person)-[rel:ACTED\_IN]->(m:Movie)

WITH p,collect(m.title) as movies,count(\*) as total

RETURN p.name, movies,total

ORDER BY total DESC

LIMIT 5;

-- top 5 director with most movies

MATCH (p:Person)-[rel:DIRECTED\*]->(m:Movie)

WITH p,collect(m.title) as movies,count(\*) as total

RETURN p.name, movies,total

ORDER BY total DESC

LIMIT 5;

-- top 5 category with most movies

MATCH (c:Category)<-[rel:IN\_CATEGORY]-(m:Movie)

WITH c.name as category,count(\*) as total

RETURN category,total

ORDER BY total DESC;

-- top 5 type with most movies

MATCH (t:Type)<-[rel:TYPED\_AS]-(m:Movie)

WITH t.type as type,count(\*) as total

RETURN type,total

ORDER BY total DESC;

-- top 5 country with most movies

MATCH (c:Country)<-[rel:WHERE]-(m:Movie)

WITH c.name as country,count(\*) as total

RETURN country,total

ORDER BY total DESC;

Output Screenshots:

Chart

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

A picture containing chart

Description automatically generated

A picture containing text

Description automatically generated

A picture containing graphical user interface

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated