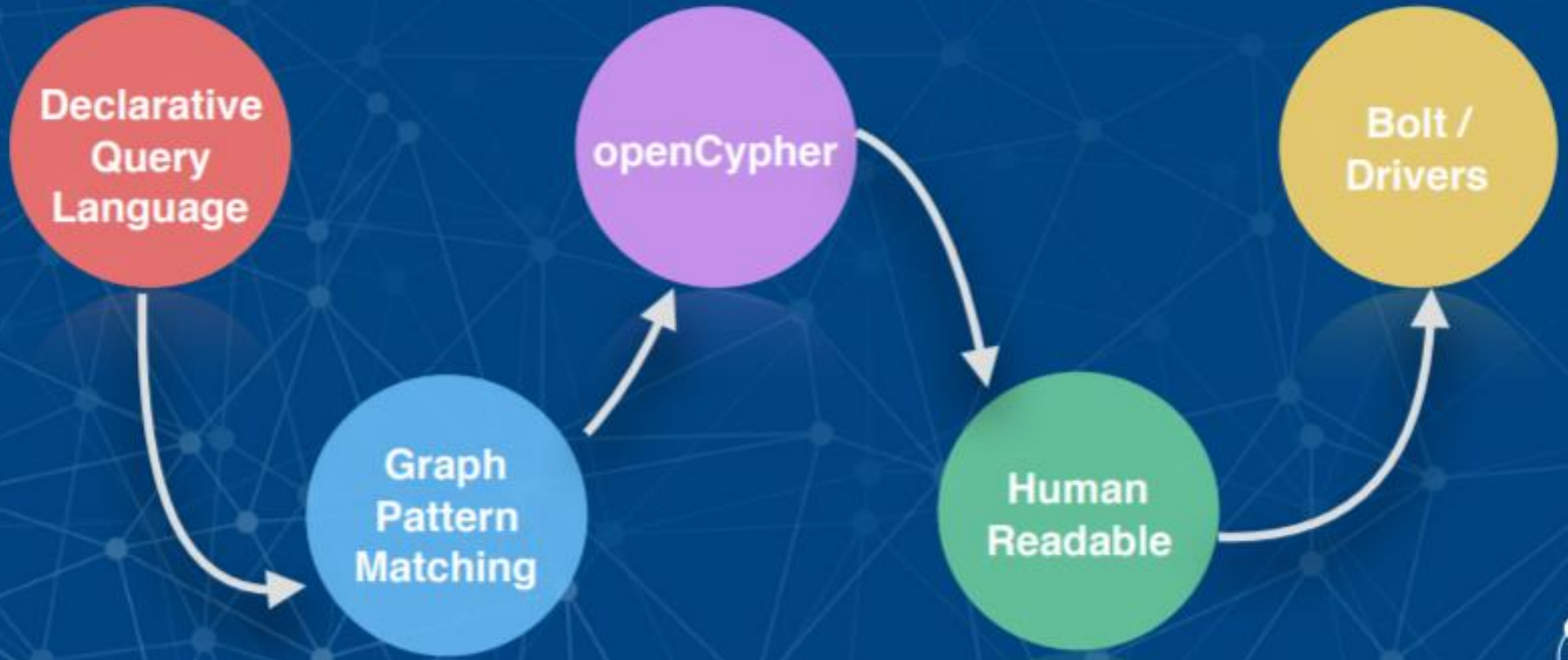
Neo4j Cypher & MovieLens

CYPHER QUERY

What is Cypher?



neo4j

What is Cypher?

- What data we want, not how

What is Cypher?

- What data we want, not how
- Expressiveness


```
(node)-[:RELATIONSHIP]->(node)
```

neo4j

What is Cypher?

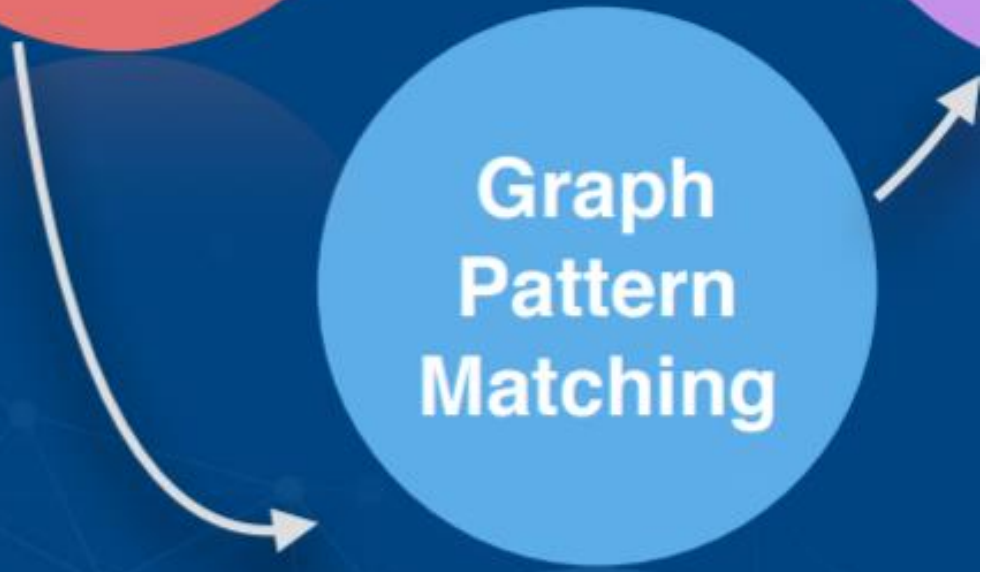
- What data we want, not how
- Expressiveness


```
(node {key: value})-[:RELATIONSHIP]->(node)
```

neo4j

What is Cypher?

- What data we want, not how
- Expressiveness




```
(node {key: value})-[:RELATIONSHIP*..2]->(node)
```

neo4j

Query Patterns: Create

- Create a node

```
CREATE (c:City:Country {name: "Busan", population_size: 1000000})
```

- Create nodes with relationships

```
CREATE (c1:City:Country {name: "Daegu"}),
       (c2:City {name: "Seoul", population_size: 9000000})
       (c1)<-[r:IN]-(c2)
RETURN c1, c2, r
```

- Create a relationship between existing nodes

```
MATCH (c1), (c2)
WHERE c1.name = "Seoul" AND c2.name = "Busan"
CREATE (c2)-[:IN]->(c1)
```

Query Patterns

Match labels

```
MATCH (c:City)
RETURN c
```

Match multiple labels

```
MATCH (c:City:Country)
RETURN c
```

Matching nodes with properties in a range

```
MATCH (c:City)
WHERE c.population_size >= 1000000 AND c.population_size <= 2000000
RETURN c
```

Query Patterns

Add or update nodes propeties

```
MATCH (c:City {name: “Seoul"})
SET c.name = “Seoul Special City"
```

Replace all node properties

```
MATCH (c:City)
WHERE c.name = “Seoul Special City"
SET c = {name: “Seoul", population_size: "66650000"}
```

Update multiple node properties

```
MATCH (c:City)
WHERE c.name = “Daegu"
SET c += {name: “Daegu", population_size: “5000000"}
```

Query Patterns

Delete a node

```
MATCH (c)-[r]-()
WHERE c.name = "Seoul"
DELETE r, c
```

Delete a property

```
MATCH (c:City)
WHERE c.name = "City" AND c.population_size IS NOT
null
DELETE c.population_size
```

Delete all nodes and relationships

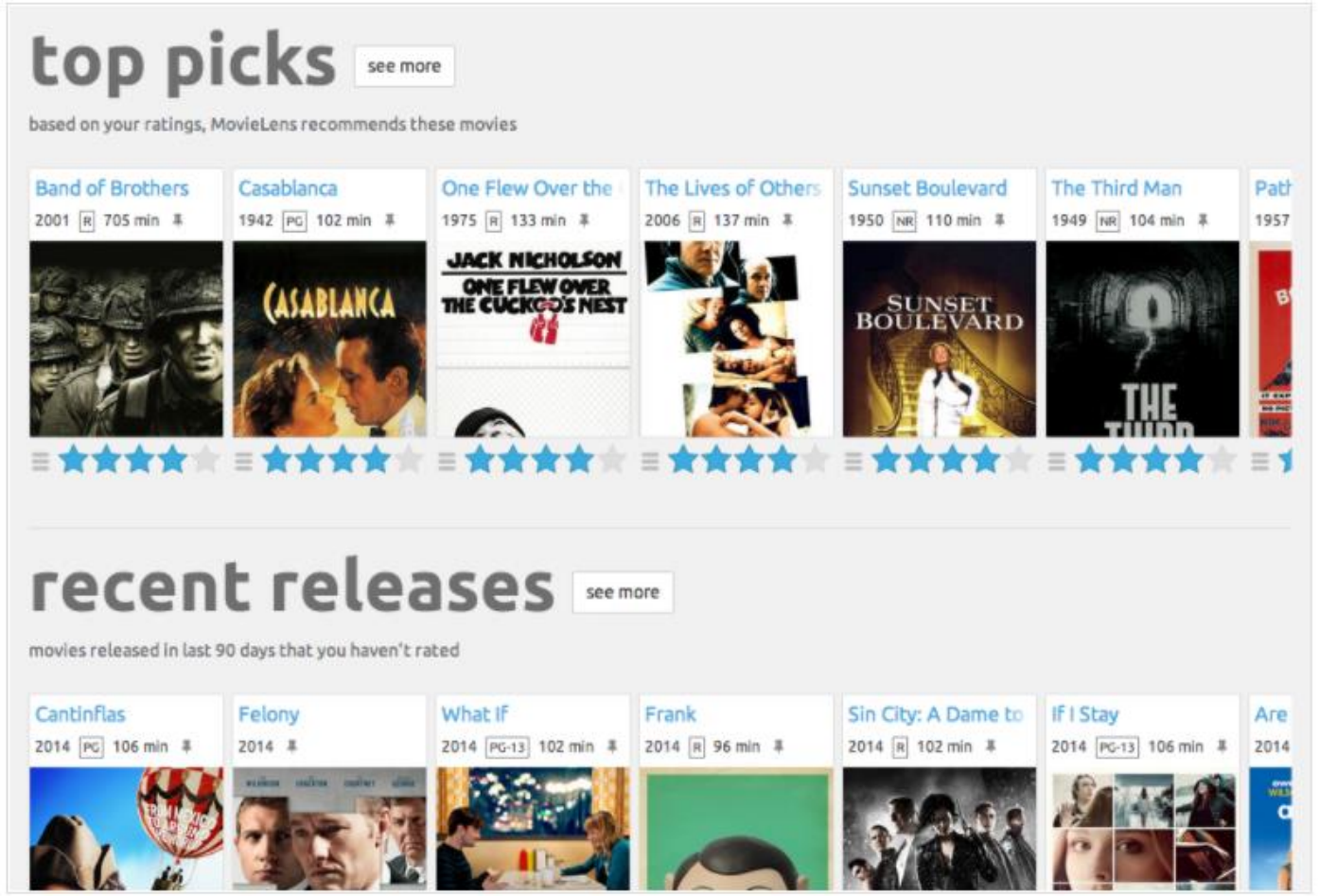
```
MATCH (n)
DETACH DELETE n
```

MOVIE LENS

Movie Lens

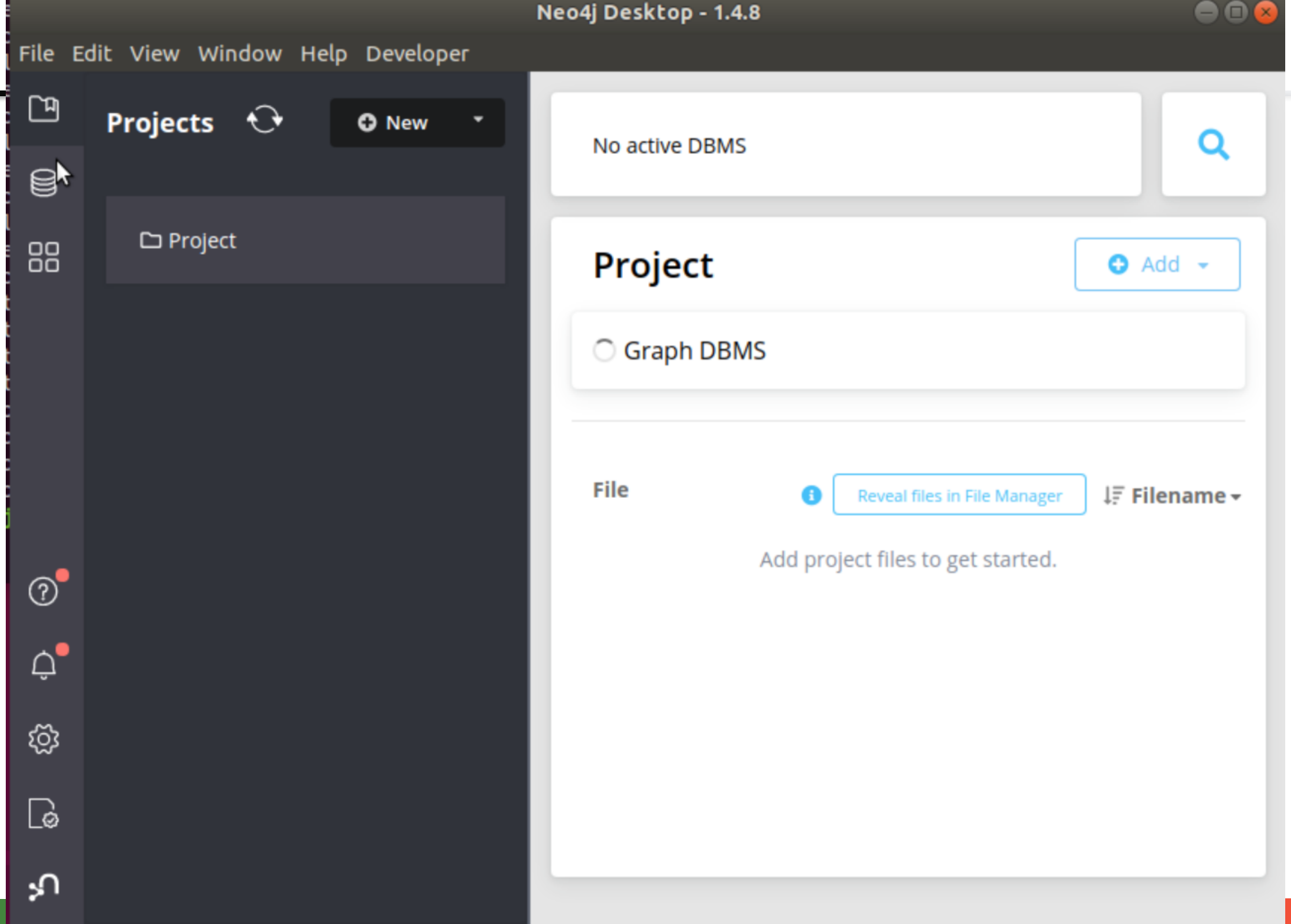
recommendations

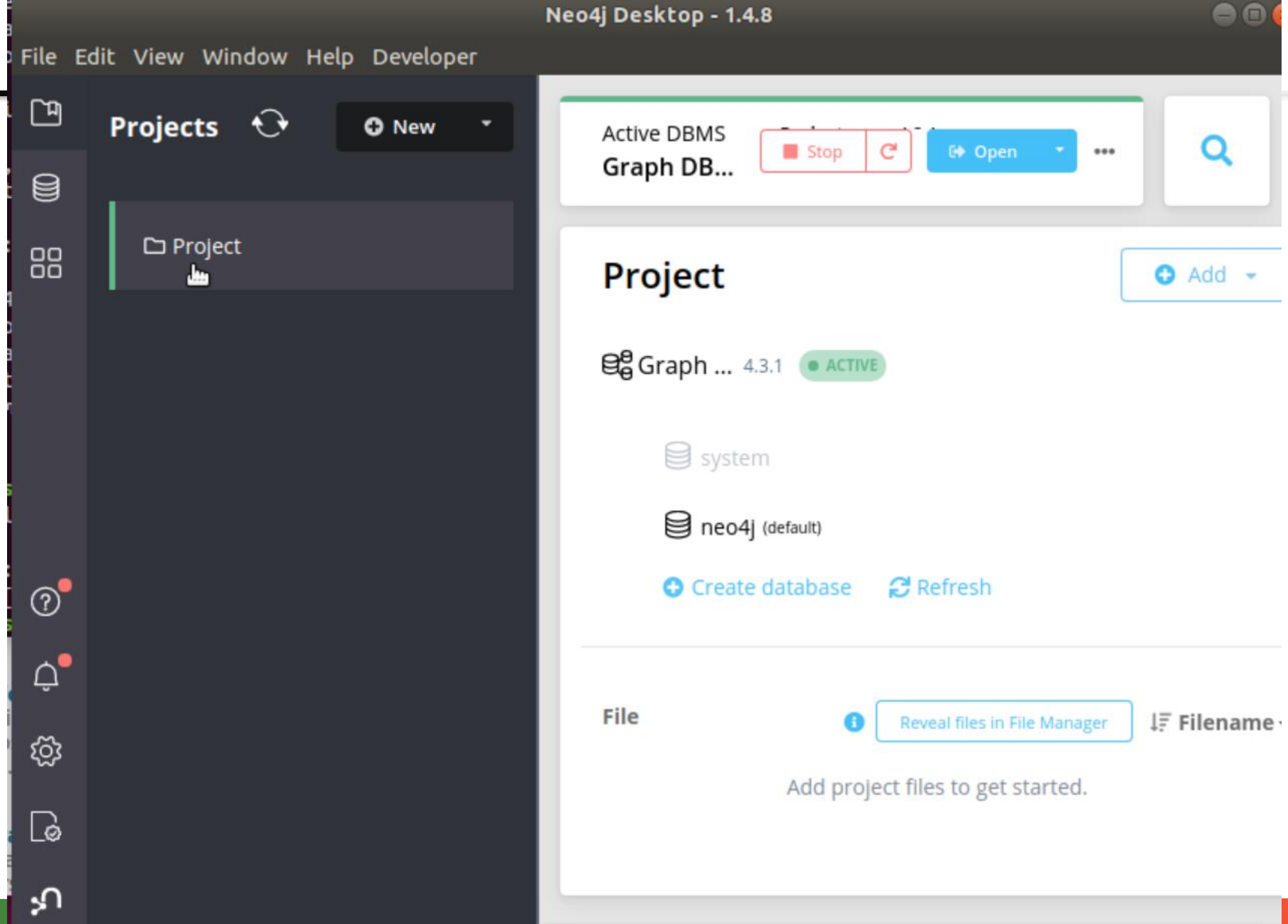
MovieLens helps you find movies you will like. Rate movies to build a custom taste profile, then MovieLens recommends other movies for you to watch.



Install python

- Terminal
- python --version
- sudo apt update
- sudo apt install software-properties-common
- sudo add-apt-repository ppa:deadsnakes/ppa
- sudo apt update
- sudo apt install python3.8
- python --version
- sudo apt install python-pip3
- sudo apt install curl


Neo4j Desktop - 1.4.8
File Edit View Window Help Developer
Projects
New
Project
No active DBMS
Project
Add
Graph DBMS
File
Reveal files in File Manager
Filename
Add project files to get started.


Neo4j Desktop - 1.4.8
File Edit View Window Help Developer
Projects
New
Project
Active DBMS
Graph DB...
Stop
Open
Project
Add
Graph ... 4.3.1
ACTIVE
system
neo4j (default)
Create database
Refresh
File
Reveal files in File Manager
Filename
Add project files to get started.

Open Terminal

- wget -q https://github.com/chunsejin/KGProjects_DAU/raw/master/movielens/Makefile
- graphdb_base.py, config.ini, import_movielens.py, requirement.txt, string_util.py

Open Terminal

- sudo make download
- sudo python import_movielens.py –u neo4j –p <password> -b bolt://localhost:7687 –s ../../dataset/movielens/ml-latest-small/

Dataset Import

```
140 movie details imported
150 movie details imported
160 movie details imported
170 movie details imported
'writers' ['206', '0114805', '77350'] 176
180 movie details imported
190 movie details imported
200 movie details imported
210 movie details imported
220 movie details imported
230 movie details imported
240 movie details imported
250 movie details imported
260 movie details imported
270 movie details imported
280 movie details imported
290 movie details imported
300 movie details imported
310 movie details imported
```

Open Neo4j Browser

Basic

- `MATCH (m:Movie)<-[:RATED]-(u:User)`

Query #4_1

```
CREATE (p:Movie {
            title: 'Pulp Fiction',
            actors: ['John Travolta', 'Samuel L. Jackson', 'Bruce Willis', 'Uma Thurman'],
            director: 'Quentin Tarantino',
            genres: ['Action', 'Crime', 'Triller'],
            writers: ['Quentin Tarantino', 'Roger Avary'],
    year: 1994
})
```

Query #4_1

```
CREATE (t:Movie {
            title: 'The Punisher',
            actors: ['Thomas Jane', 'John Travolta', 'Samantha Mathis'],
            director: 'Jonathan Hensleigh',
            genres: ['Action', 'Adventure', 'Crime, Drama', 'Thriller'],
            writers: ['Jonathan Hensleigh', 'Michael France'],
    year: 2004
})
```

Query #4_1

```
CREATE (k:Movie {
            title: 'Kill Bill: Volume 1',
            actors: ['Uma Thurman', 'Lucy Liu', 'Vivica A. Fox'],
            director: 'Quentin Tarantino',
            genres: ['Action', 'Crime', 'Triller'],
            writers: ['Quentin Tarantino', 'Uma Thurman'],
    year: 2003
})
```

Create 오징어게임

- 위키피디아를 검색해서 넣어보자

Query #4_2

```
MATCH (m:Movie)
WHERE m.director = '####'
RETURN m
```

Query #4_3

```
MATCH (m:Movie)
WITH m.actors as actors
UNWIND actors as actor
MATCH (n:Movie)
WHERE actor IN n.actors
WITH actor, n.actors as otherActors, n.title as title
UNWIND otherActors as otherActor
WITH actor, otherActor, title
WHERE actor <> otherActor
RETURN actor, otherActor, title
ORDER BY actor
```

Query #4_4

```
CREATE CONSTRAINT ON (a:Movie) ASSERT a.title IS UNIQUE;
CREATE CONSTRAINT ON (a:Genre) ASSERT a.genre IS UNIQUE;
CREATE CONSTRAINT ON (a:Person) ASSERT a.name IS UNIQUE;

CREATE (pulp:Movie {title: 'Pulp Fiction'})
FOREACH (director IN ['Quentin Tarantino']
| MERGE (p:Person {name: director}) SET p:Director MERGE (p)-[:DIRECTED]->(pulp))
FOREACH (actor IN ['John Travolta', 'Samuel L. Jackson', 'Bruce Willis', 'Uma Thurman']
| MERGE (p:Person {name: actor}) SET p:Actor MERGE (p)-[:ACTS_IN]->(pulp))
FOREACH (writer IN ['Quentin Tarantino', 'Roger Avary']
| MERGE (p:Person {name: writer}) SET p:Writer MERGE (p)-[:WRITES]->(pulp))
FOREACH (genre IN ['Action', 'Crime', 'Triller']
| MERGE (g:Genre {genre: genre}) MERGE (pulp)-[:HAS_GENRE]->(g))

CREATE (punisher:Movie {title: 'The Punisher'})
FOREACH (director IN ['Jonathan Hensleigh']
| MERGE (p:Person {name: director}) SET p:Director MERGE (p)-[:DIRECTED]->(punisher))
FOREACH (actor IN ['Thomas Jane', 'John Travolta', 'Samantha Mathis']
| MERGE (p:Person {name: actor}) SET p:Actor MERGE (p)-[:ACTS_IN]->(punisher))
FOREACH (writer IN ['Jonathan Hensleigh', 'Michael France']
| MERGE (p:Person {name: writer}) SET p:Writer MERGE (p)-[:WRITES]->(punisher))
FOREACH (genre IN ['Action', 'Adventure', 'Crime', 'Drama', 'Thriller']
| MERGE (g:Genre {genre: genre}) MERGE (punisher)-[:HAS_GENRE]->(g))
```

Query #4_5

```
MATCH (actor:Actor)-[:ACTS_IN]->(movie:Movie)<-[:ACTS_IN]-(otherActor:Actor)
WHERE actor <> otherActor
RETURN actor.name as actor, otherActor.name as otherActor, movie.title as title
ORDER BY actor
```

Query #4_6

```
//If you need to run over a bigger dataset, better use apoc
CALL apoc.periodic.iterate("MATCH (user:User)
where not user:Processed
return user",
"SET user:Processed WITH user MATCH (user)-[:RATES]->(movie:Movie)-
[:ACTS_IN|WRITES|DIRECTED|PRODUCES|HAS_GENRE]-(feature)
WITH user, feature, count(feature) as occurrences
WHERE occurrences > 2
MERGE (user)-[:INTERESTED_IN]->(feature)",
{batchSize:10, parallel:false})
```

Query #4_7

```
MATCH (user:User)-[i:INTERESTED_IN]->(feature)-[]-(movie:Movie)
WHERE user.userId = <userId> AND NOT exists((user)-[]->(movie))
RETURN movie.title, count(i) as occurrences
ORDER BY occurrences desc
```

Query #4_8

```
MATCH (feature)
WHERE "Genre" in labels(feature) OR "Director" in labels(feature)
WITH feature
ORDER BY id(feature)
MATCH (movie:Movie)
WHERE movie.title STARTS WITH “Four Rooms"
OPTIONAL MATCH (movie)-[r:DIRECTED|HAS_GENRE]-(feature)
RETURN CASE WHEN r IS null THEN 0 ELSE 1 END as Value,
CASE WHEN feature.genre IS null THEN feature.name ELSE feature.genre END as Feature
```

Query #4_9

```
CALL apoc.periodic.iterate("MATCH (user:User)
where not user:Processed
return user",
"SET user:Processed WITH user MATCH (user)-[:RATES]->(movie:Movie)-
[:ACTS_IN|WRITES|DIRECTED|PRODUCES|HAS_GENRE]-(feature)
WITH user, feature, count(feature) as occurrences
WHERE occurrences > 2
MERGE (user)-[r:INTERESTED_IN]->(feature)
SET r.weight = occurrences",
{batchSize:10, parallel:false})
```