Name: Troy Zhongyi Zhang Netid: zhongyiz@uchicago.edu

DATA ENGINEERING PLATFORMS (MSCA 31012)

sbharadwaj@uchicago.edu | jchan530@uchicago.edu

Submissions

• Single file (txt/word/Pdf) that contains the queries for both MongoDB and Neo4j along with any assumptions made

Part A (MongoDB): Manipulating, Sorting and Grouping & Summarizing data

Data (Sakila dataset):

- customers.json
- films.json
- stores.json

Note: Import dataset (sample scripts to load data) mongoimport --db sakila --collection customers --drop --file "C:\Users\SBharadwaj\Desktop\Shree\DEPA\03-Assignments\4\solution\customers.json" (imports 599 documents)

mongoimport --db sakila --collection films --drop --file "C:\Users\SBharadwaj\Desktop\Shree\DEPA\03-Assignments\4\solution\films.json" (imports 1000 documents)

mongoimport --db sakila --collection stores --drop --file "C:\Users\SBharadwaj\Desktop\Shree\DEPA\03-Assignments\4\solution\stores.json" (imports 2 documents)

Screenshot for successful importing files:

```
[(base) Zhongyis-MBP:Assignment4 zhongyizhang$ pwd
/Users/zhongyizhang/Desktop/Assignment4
[(base) Zhongyis-MBP:Assignment4 zhongyizhang$ mongoimport --db sakila --collecti]
on customers --drop --file "/Users/zhongyizhang/Desktop/Assignment4/dataset/cust
                                 connected to: localhost
2019-08-17T10:12:38.078-0500
2019-08-17T10:12:38.079-0500
                                 dropping: sakila.customers
2019-08-17T10:12:38.194-0500
                                 imported 599 documents
(base) Zhongyis-MBP:Assignment4 zhongyizhang$ mongoimport --db sakila --collecti
on films --drop --file "/Users/zhongyizhang/Desktop/Assignment4/dataset/films.js
2019-08-17T10:13:43.237-0500
                                 connected to: localhost
2019-08-17T10:13:43.237-0500
                                 dropping: sakila.films
2019-08-17T10:13:43.295-0500
                                 imported 1000 documents
[(base) Zhongyis-MBP:Assignment4 zhongyizhang$ mongoimport --db sakila --collecti]
on stores --drop --file "/Users/zhongyizhang/Desktop/Assignment4/dataset/stores.
ison"
2019-08-17T10:14:20.549-0500
                                 connected to: localhost
2019-08-17T10:14:20.549-0500
                                 dropping: sakila.stores
2019-08-17T10:14:20.594-0500
                                 imported 2 documents
(base) Zhongyis-MBP:Assignment4 zhongyizhang$
```

Questions: – {50 Points}

1.List total number of customers living in California?

db.customers.find({ District: "California" }).count()



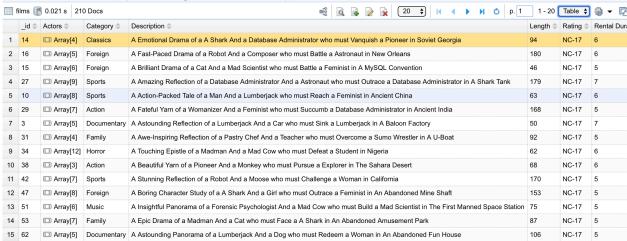
Or

db.customers.count({District:"California"})



2. List all movies that are rated NC-17

db.films.find({ Rating: "NC-17" })



3. List the count of movies by category



4. Find the top 2 movies with movie length greater than 25mins and which has commentaries as a special feature

```
// List ordered by _id, first appeared first results come (or ordered by _id): db.films.find(
```

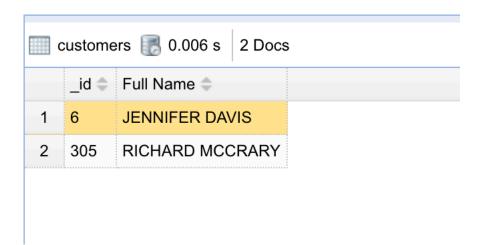
{\$and: [

```
{"Length": {$gt: "25"}}, {"Special Features": {$regex:
"Commentaries"}}
   ]}).limit(2)
                                                        0€ Q P. 1 1-2 (Table $) ∰ ▼ [7]
   _id \( \) Actors \( \) Category \( \) Description \( \)
                                                                      Length Rating Rental Duration Replacement Cost Special
 1 15 In Array[6] Foreign A Brilliant Drama of a Cat And a Mad Scientist who must Battle a Feminist in A MySQL Convention 46
                                                                              NC-17 5
 2 20 🖾 Array[6] Music A Boring Drama of a Woman And a Squirrel who must Conquer a Student in A Baloon
// Ordered by length from the longest length to the shorter length (99
mins come first):
db.films.find(
    {"Length": {$gt: "25"}, "Special Features": {$regex:
"Commentaries"}}
   ).sort({Length: -1}).limit(2)
iii films 8 0.011 s 2 Docs
                                                        Length Rating Rental Duration Replacement Cost Special
                                                                              NC-17 5
 15 🗓 Array[6] Foreign A Brilliant Drama of a Cat And a Mad Scientist who must Battle a Feminist in A MySQL Convention 46
 2 469 🖸 Array[8] Classics A Fast-Paced Documentary of a Mad Cow And a Boy who must Pursue a Dentist in A Baloon 46
```

5. Provide 2 additional queries and indicate the specific business use cases they address

```
// Use case 1:
// Show the cutomer's full name (concatenated the first name and last name into one field) if this customer is in the US and phone number includes 26

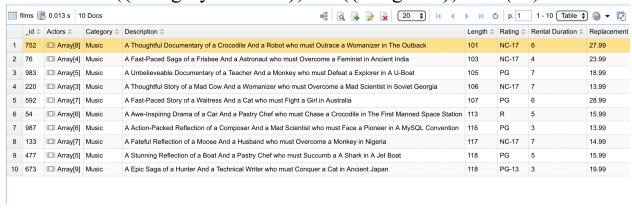
db.customers.aggregate([{$match:{$and:[{Country:"United States"}, {Phone:{$regex: "26"}}]}}}, {$project:
{ "Full Name": { $concat: [ "$First Name", " ", "$Last Name" ]}}}])
```



// Use case 2:

// List 10 movies in the music category ordered by length from the shortest to longer length movies

db.films.find({Category:"Music"}).sort({Length: 1}).limit(10)



Part B (Neo4J): Linking, Manipulating & viewing relationships within data

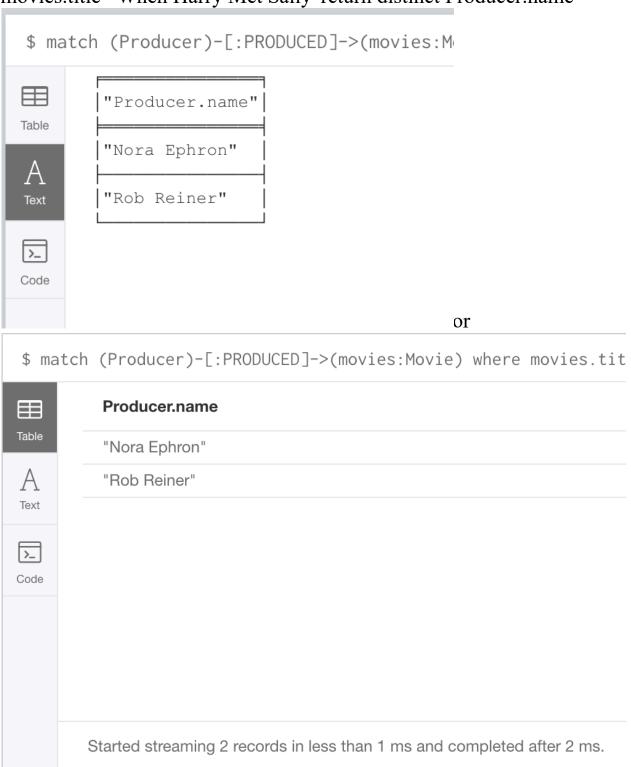
Data (Movies dataset):

• Moviesdb.txt

Questions: – {50 Points}

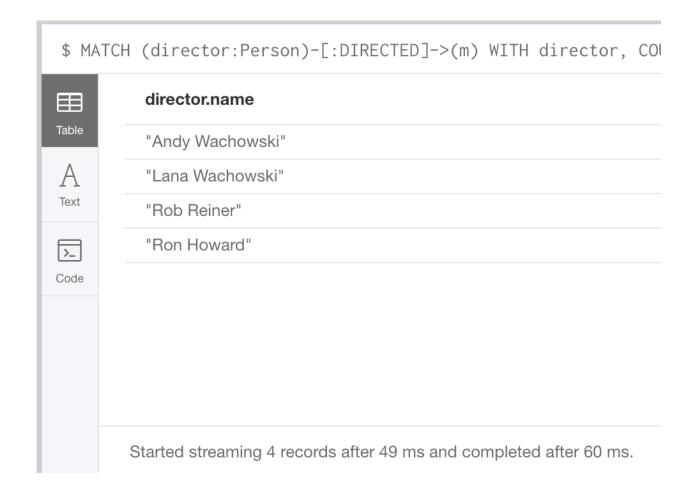
1.Find all Producers that produced the movie When Harry Met Sally

match (Producer)-[:PRODUCED]->(movies:Movie) where movies.title='When Harry Met Sally' return distinct Producer.name



2. Find directors who have directed more than 2 movies

MATCH (director:Person)-[:DIRECTED]->(m) WITH director, COUNT(m) AS CountMovie WHERE CountMovie>2 RETURN DISTINCT director.name



3. Find the actors with 5+ movies, and the movies in which they acted

match(actor:Person)-[:ACTED_IN]->(movie:Movie) with actor, collect(movie.title) as movies, count(movie) as mc where mc >= 5 return actor.name, movies



4. Movies and actors exactly 3 "hops" away from the movie Hoffa

MATCH (Hoffa:Movie {title:'Hoffa'})-[*3]-(hollywood) RETURN DISTINCT hollywood



5. Provide 2 additional queries and indicate the specific business use cases they address

1.Find all the title of movies that an actor in "The Matrix" acted in. For each movie title, also return the name of these actors

MATCH (:Movie {title:"The Matrix"})<-[:ACTED_IN]-(person)-[:ACTED_IN]->(film)

RETURN film.title as title, collect(person.name) as actors



2. Find the actors acting as more than one role in a movie.

MATCH (pers:Person)-[act:ACTED_IN]-(film:Movie) WITH pers, act.roles as roles, film WHERE size(roles) > 1 RETURN pers, roles, film

