MongoDB

/* 1. List total number of customers living in california ? */

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

/* 2. List all movies that are rated NC-17 */

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

/* 3. List the count of movies by category */

```
db.films.aggregate([ {"$group": {_id:"$Category", count:{$sum:1}}} ])
```

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

```
db.films.find(
   {$or: [{Length:{$gt: '25'}}, {'Special Features':{'$regex':"Commentaries"}} ]},
   {Title: 1, Length:1, 'Special Features':1}).limit(2)
```

/* 5. Find the top 10 customers based on number of rentals */

```
db.customers.aggregate(
   {$project:{item:1,
   "First Name":1,"Last Name":1,
   numberRental: {$size: "$Rentals"}}}).sort({numberRental: -1}).limit(10)
```

/* 6. Provide 5 additional queries and indicate the specific business use cases they address

Note: Insights should not be a flavor of the previously addressed queries within Assignment 4. */

/* 1. List the count of movies by special features */

```
db.films.aggregate([ {"$group": {_id:"$special features", count:{$sum:1}}} ])
```

```
/* 2. Find the top 10 films based on number of actors */

db.films.aggregate(

   {$project:{item:1,

   "Title":1,

   numberActor: {$size: "$Actors"}}}).sort({numberActor: -1}).limit(10)

/* 3. List all customers that are from China */

db.customers.find({Country:"China"})

/* 4. Count the number of customers with last name Grey */

db.customers.find({"Last Name": "GREY"}).count()

/* 5. Sort the films by rental duration */

db.films.find().sort({"Rental Duration": 1})
```

Neo4j

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

```
MATCH (a:Person)-[:PRODUCED]->(m:Movie)
WHERE m.title = 'When Harry Met Sally'
RETURN a.name as producer
```

2. Find directors who have directed more than 2 movies

```
MATCH (a:Person)-[:DIRECTED]->(m:Movie)
WITH a, count(m) AS numMovies
WHERE numMovies > 2
RETURN a.name
```

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

```
MATCH (a:Person)-[:ACTED_IN]->(m:Movie)
WITH a, count(m) AS numMovies, collect(m.title) AS movies
WHERE numMovies > 5
RETURN a.name, movies
```

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

   MATCH (moviehoffa:Movie {title:"Hoffa"})-[*3] -(movies_actors) RETURN DISTINCT movies_actors

5. Find all actors who have also directed movies and the movies that they directed

   MATCH (actors:Person)-[:ACTED_IN]->(m:Movie)WHERE exists((actors)-[:DIRECTED]->(m))RETURN actors.name as `Actor/Director`, m.title as Movie

6. Provide 5 additional queries and indicate the specific business use cases they address Note: Insights should not be a flavor of the previously addressed queries within Assignment 4.

1. Retrieve the movies that have more than2 directors

```
MATCH (m:Movie)
WITH m, size((:Person)-[:DIRECTED]->(m)) AS directors
WHERE directors > 2
RETURN m.title
```

2. Retrieve the top 5 ratings and their associated movies, returning the movie title and the rating.

```
MATCH (:Person)-[r:REVIEWED]->(m:Movie)
RETURN m.title AS movie, r.rating AS rating
ORDER BY r.rating DESC LIMIT 10
```

3. what actors acted in movies that was released between 2000 to 2005

```
MATCH (a:Person)-[:ACTED_IN]->(m:Movie)
WHERE m.released >= 2000 AND m.released < 2005
RETURN m.released, collect(m.title), collect(a.name)
```

4. Retrieve all actors that have not appeared in more than 4 movies

```
MATCH (a:Person)-[:ACTED_IN]->(m:Movie)
```

```
WITH a, count(a) AS numMovies, collect(m.title) AS movies
WHERE numMovies <= 4
RETURN a.name, movies
```

5. retrieve nodes that are one and two hops away and has the *FOLLOWS* relationship with Paul Blythe in either direction

```
MATCH (p1:Person)-[:FOLLOWS*1..2]-(p2:Person)
WHERE p1.name = 'Paul Blythe'
RETURN p1, p2
```