IE6700 HW5 - Charles Schatmeyer

1. - 11.3E
a.
```
MATCH (b:Book)
WHERE NOT (b)<-[:LIKES]-()
RETURN b.title AS BookTitle
```

b.
```
MATCH (r1:Reader), (r2:Reader)
WHERE r1 <> r2 AND NOT ( (r1)-[:LIKES]->(:Book)<-[:LIKES]-(r2) )
RETURN r1.name AS Reader1, r2.name AS Reader2
ORDER BY Reader1, Reader2
```

c.
```
MATCH (:Reader)-[:LIKES]->(b:Book)-[:IS_GENRE]->(g:Genre)
RETURN g.name, COUNT(b) AS LikedBookCount
ORDER BY LikedBookCount DESC
LIMIT 1
```

d.
```
MATCH (r1:Reader)-[:LIKES]->(b:Book)<-[:LIKES]-(r2:Reader)
WHERE r1 <> r2
RETURN r1.name, COUNT(b) AS CommonLikes
ORDER BY CommonLikes DESC
LIMIT 1
```

2. - 11.5E
Aggregate Command:
```
db.books.aggregate([
        { $group: { _id: "$genre", totalPages: { $sum: "$nrPages" } }},
        { $sort: { _id: 1 }}
])
```
Aggregate becomes harder to use with more complex queries.

3.
a.

```
db.people.mapReduce(
   function() { emit(this.restaurant_id, this.rating); },
   function(key, values) { return Array.sum(values) / values.length; },
   { out: "average_rating_per_restaurant" }
);
db.average_rating_per_restaurant.find();
```

b.

```
db.people.aggregate([
   { $group: { _id: "$restaurant_id", averageRating: { $avg: "$rating" } } }
])
```

c.

```
db.people.mapReduce(
   function() { emit(this.restaurant_id, this.rating); },
   function(key, values) { return Math.max.apply(null, values); },
   { out: "max_rating_per_restaurant" }
);
db.max_rating_per_restaurant.find();
```

d.

```
db.people.aggregate([
   { $group: { _id: "$restaurant_id", maxRating: { $max: "$rating" } } }
]);
```

4.
a.
```
db.restaurants.mapReduce(
  function() { emit(this.location, 1); },
  function(key, values) { return Array.sum(values); },
  { out: "count_restaurants_per_location" }
);
db.count_restaurants_per_location.find();
```

b.
```
db.restaurants.aggregate([
  { $group: { _id: "$location", averageRating: { $avg: "$rating" } } }
]);
```

c.
```
db.restaurants.aggregate([
  { $group: { _id: "$type_of_food", averageRating: { $avg: "$rating" } } },
  { $sort: { averageRating: -1 } },
  { $limit: 1 }
]);
```

5.
a.
```
MATCH (seppe:Person {name: "Seppe"})-[:LIKES]->(beer:Beer)-[:ISTYPE]->(type:BeerType)
RETURN DISTINCT type.name AS BeerType
```

b.
```
MATCH (seppe:Person {name: "Seppe"})-[:LIKES]->(beer:Beer)<-[:LIKES]-(bart:Person {name:
"Bart"})
MATCH (beer)-[:ISTYPE]->(type:BeerType)
RETURN DISTINCT type.name AS BeerType
```

c.
```
MATCH (brewery:Brewery {name: "Brouwerij
Lupus"})-[:BREWS]->(beer:Beer)<-[:LIKES]-(seppe:Person {name: "Seppe"})
MATCH (beer)-[:ISTYPE]->(type:BeerType)
RETURN beer.name AS BeerName, beer.year AS Year, type.name AS BeerType
```

d.
```
MATCH (seppe:Person {name:
"Seppe"})-[:LIKES]->(beer:Beer)<-[:LIKES]-(otherPerson:Person)
RETURN DISTINCT otherPerson.name AS PersonName
```

6.
a.

```
MATCH (bart:Reader {name: "Bart Baesens"})-[:FRIEND_OF]->(friend:Reader)
RETURN bart.name AS Name, COUNT(friend) AS NumberOfFriends;
```

b.

```
MATCH (bart:Reader {name: "Bart Baesens"})-[:LIKES]->(book:Book)
OPTIONAL MATCH (seppe:Reader {name: "Seppe vanden Broucke"})-[:LIKES]->(book)
WHERE seppe IS NULL
RETURN DISTINCT book.title AS BookTitle
```

c.

```
MATCH (r1:Reader)-[:LIKES]->(book:Book)<-[:LIKES]-(r2:Reader)
WHERE r1 <> r2
WITH r1, r2, COUNT(book) AS CommonBooks
WHERE CommonBooks > 1
RETURN r1.name AS Reader1, r2.name AS Reader2, CommonBooks
```

d.

```
MATCH (seppe:Reader {name: "Seppe vanden
Broucke"})-[:FRIEND_OF]->(friend:Reader)-[:LIKES]->(book:Book)-[:IS_GENRE]->(genre:Genre
)
RETURN genre.name AS Genre, COUNT(book) AS LikeCount
ORDER BY LikeCount DESC
```

7.
a.
```
db.people.find(
  { name: "Seppe" },
  { rating: 1, _id: 0 }
)
```

b.
```
db.people.aggregate([
  { $match: { name: "Wilfried" } },
  { $group: { _id: "$name", averageRating: { $avg: "$rating" } } }
])
```

c.
```
db.restaurants.find(
  { rating: { $gte: 5 }, type_of_food: { $in: ["Pizza", "Curry"] } },
  { name: 1, location: 1, type_of_food: 1, _id: 0 }
)
```

d.
```
db.restaurants.aggregate([
  { $group: { _id: "$type_of_food", restaurantCount: { $sum: 1 } } },
  { $sort: { restaurantCount: -1 } }
])
```

e.
```
db.restaurants.aggregate([
  { $group: { _id: "$type_of_food", averageRating: { $avg: "$rating" } } },
  { $sort: { averageRating: -1 } }
])
```

8.
a.
```
db.products.find(
  { type: { $in: ["rose", "sparkling"] } }
)
```

b.
```
db.products.aggregate([
  { $match: { available_quantity: { $gte: 50, $lte: 90 } } },
  { $count: "productCount" }
])
```

c.
```
db.supplies.aggregate([
  { $match: { delivery_period: { $ne: null } } },
  { $group: { _id: "$supplier", totalSupplies: { $sum: 1 } } }
])
```

d.
```
db.products.aggregate([
  { $match: { name: { $not: /Chateau/ } } },
  { $group: { _id: "$type", totalQuantity: { $sum: { $ifNull: ["$available_quantity", 0] } } } }
])
```