A close up of a logo

Description automatically generated

CSCU9YQ Assignment

– NoSQL Databases

Student IDs: 2634056 and 2617433

Table of Contents

[Question 1 3](#_Toc67210043)

[Query 1.1. 3](#_Toc67210044)

[Query 3](#_Toc67210045)

[Formatted 3](#_Toc67210046)

[Result: 3](#_Toc67210047)

[Query 1.2. 4](#_Toc67210048)

[Query 4](#_Toc67210049)

[Formatted 4](#_Toc67210050)

[Results: 4](#_Toc67210051)

[Question 2 6](#_Toc67210052)

[Query 2.1. 6](#_Toc67210053)

[Query 6](#_Toc67210054)

[Formatted 6](#_Toc67210055)

[Results: 6](#_Toc67210056)

[Query 2.2. 6](#_Toc67210057)

[Query 6](#_Toc67210058)

[Formatted 7](#_Toc67210059)

[Results: 7](#_Toc67210060)

[Question 3 8](#_Toc67210061)

[Query 3.1. 8](#_Toc67210062)

[Query 8](#_Toc67210063)

[Formatted 8](#_Toc67210064)

[Query 3.2. 9](#_Toc67210065)

[Query 9](#_Toc67210066)

[Formatted 9](#_Toc67210067)

[Result: 10](#_Toc67210068)

# Question 1

## Query 1.1.

For this query, we are firstly using the **$unwind** operation on all genres to deconstruct the array of genres into individual elements. From there, we use the **$group** operation to group movies using the genres as ID for the grouping. We calculate the average duration of each groups taking the **$avg** operation on the **$runtime** field.

Finally, we use the **$sort** operation to sort the results in decreasing order of **average runtime** field and limit the query results to only show five results using the **$limit** operation.

### Query

**db.movies.aggregate([{$unwind:"$genres"},{$group:{\_id:"$genres", "average runtime": {$avg:"$runtime"}}},{$sort:{"average runtime":-1}},{$limit: 5}])**

### Formatted

**db.movies.aggregate([**

**{**

**$unwind: "$genres"**

**},**

**{**

**$group: {**

**\_id: "$genres",**

**"average runtime": {**

**$avg: "$runtime"**

**}**

**}**

**},**

**{**

**$sort: {**

**"average runtime": -1**

**}**

**},**

**{**

**$limit: 5**

**}**

**])**

### Result:

**{ "\_id" : "Musical", "average runtime" : 113.375 }**

**{ "\_id" : "Romance", "average runtime" : 107.18571428571428 }**

**{ "\_id" : "Crime", "average runtime" : 106.56571428571428 }**

**{ "\_id" : "Action", "average runtime" : 105.208 }**

**{ "\_id" : "Western", "average runtime" : 105.12121212121212 }**

## Query 1.2.

For this query, we first match on all movies that has UK in their country field. We then unwind on the country field giving us each element(country) that appears alongside UK(due to the match). We do another match on countries not equal to the UK so what we are left with is all countries that have collaborated with the UK but not the UK itself. After this we use group to bring together all the same countries and count them, giving us the name of the country and number of times, it was found alongside the UK. Another match is used so we only get countries that collaborated with the UK 10 or more times.

### Query

**db.movies.aggregate([{$match: {countries: "UK"}},{$unwind: "$countries"},{$match: {countries: {$ne: "UK"}}},{$group: {\_id: "$countries","UK collaborations": {$sum: 1}}},{$match: {"UK collaborations": {$gte: 10}}}])**

### Formatted

**db.movies.aggregate([**

**{**

**$match: {**

**countries: "UK"**

**}**

**},**

**{**

**$unwind: "$countries"**

**},**

**{**

**$match: {**

**countries: {**

**$ne: "UK"**

**}**

**}**

**},**

**{**

**$group: {**

**\_id: "$countries",**

**"UK collaborations": {**

**$sum: 1**

**}**

**}**

**},**

**{**

**$match: {**

**"UK collaborations": {**

**$gte: 10**

**}**

**}**

**}**

**])**

### Results:

**{ "\_id" : "Canada", "UK collaborations" : 11 }**

**{ "\_id" : "USA", "UK collaborations" : 62 }**

**{ "\_id" : "France", "UK collaborations" : 16 }**

**{ "\_id" : "Germany", "UK collaborations" : 14 }**

# Question 2

## Query 2.1.

For the query, we unwind the genres field to separate each element. We then want to match the elements that match the “Sport” genre. Once we have all those records, we use the **$count** operation to count the number of elements returned by the **$match** pipeline.

### Query

**db.movies.aggregate([{$unwind:"$genres"},{$match: {"genres":"Sport"}},{$count:"sportMovies"}])**

### Formatted

**db.movies.aggregate([**

**{**

**$unwind: "$genres"**

**},**

**{**

**$match: {**

**"genres": "Sport"**

**}**

**},**

**{**

**$count: "sportMovies"**

**}**

**])**

### Results:

**{ "sportMovies" : 28 }**

## Query 2.2.

For this query, we start from the work of the query in 2.1, which has filtered all movies with genre matching “Sport”. From there, we sort the results in decreasing order of IMDB ratings, and only limit the results to 3.

Finally, with the **$project** pipeline, we selected which values we wanted to get displayed and in which order. We display the title, year, IMDB rating and votes, and we omit the ID of the element.

### Query

**db.movies.aggregate([{$unwind:"$genres"},{$match: {"genres":"Sport"}},{$sort:{"imdb.rating":-1}}, {$limit:3}, {$project:{\_id:0,title:1, year:1,"imdb.rating":1, "imdb.votes":1}}])**

### Formatted

**db.movies.aggregate([**

**{**

**$unwind: "$genres"**

**},**

**{**

**$match: {**

**"genres": "Sport"**

**}**

**},**

**{**

**$sort: {**

**"imdb.rating": -1**

**}**

**},**

**{**

**$limit: 3**

**},**

**{**

**$project: {**

**\_id: 0,**

**title: 1,**

**year: 1,**

**"imdb.rating": 1,**

**"imdb.votes": 1**

**}**

**}**

**])**

### Results:

**{ "title" : "Liverpool FC: Champions of Europe 2005", "year" : 2005, "imdb" : { "rating" : 9.5, "votes" : 410 } }**

**{ "title" : "WWE: DX: One Last Stand", "year" : 2011, "imdb" : { "rating" : 7.7, "votes" : 51 } }**

**{ "title" : "O sport, ty - mir!", "year" : 1981, "imdb" : { "rating" : 7.5, "votes" : 23 } }**

# Question 3

## Query 3.1.

For this query, we decided to create a field that would take into account the four different ratings of a movies (IMDB, Rotten Tomatoes rating and user rating and Metacritic), but would also take into account any awards wins and nominations.

We decided we would make an average of the ratings, who all need to be set out of 10 first so as to not have disproportionate values. The average of those ratings is then multiplied by 7.5 to represent 75% of our created ratings.

The 25% left are reserved for the awards wins and nominations. If a movie has won any award, we multiply the number of awards won by 0.1875, so as to represent 18.75% of the final result. Finally, for the 6.25% left, we decided to have the number of award nominations multiplied by 0.0625.

This calculation is wrapped in **$addFields** and an **$out** which takes the final myRating calculated for each field and adds this as a new field. The out makes this change permanent and specified to the “movies” collection.

### Query

**db.movies.aggregate([{ "$addFields" : { "myRating" : { "$add" : [{ "$multiply" : [{ "$avg" : ["$imdb.rating", "$tomato.rating", { "$multiply" : ["$tomato.userRating", 2.0]}, { "$multiply" : ["$metacritic", 0.1]}]}, 7.5]}, { "$multiply" : ["$awards.wins", 0.1875]}, { "$multiply" : ["$awards.nominations", 0.0625]}]}}}, { "$out" : "movies"}])**

### Formatted

**db.movies.aggregate([**

**{**

**"$addFields": {**

**"myRating": {**

**"$add": [**

**{**

**"$multiply": [**

**{**

**"$avg": [**

**"$imdb.rating",**

**"$tomato.rating",**

**{**

**"$multiply": [**

**"$tomato.userRating",**

**2.0**

**]**

**},**

**{**

**"$multiply": [**

**"$metacritic",**

**0.1**

**]**

**}**

**]**

**},**

**7.5**

**]**

**},**

**{**

**"$multiply": [**

**"$awards.wins",**

**0.1875**

**]**

**},**

**{**

**"$multiply": [**

**"$awards.nominations",**

**0.0625**

**]**

**}**

**]**

**}**

**}**

**},**

**{**

**"$out": "movies"**

**}**

**])**

## Query 3.2.

This query demonstrates the new field was added properly. We match for movies with the director George Lucas and sort by the new “myRating” field highest to lowest to give and set a limit of 3. This gives us the top 3 movies directed by George Lucas

### Query

**db.movies.aggregate([{"$match" : {"director" : "George Lucas"}}, {"$sort" : { "myRating" : -1.0}}, {"$limit" : 3.0},{"$project" : { "\_id" : 0.0, "title" : 1.0, "director" : 1.0, "myRating" : 1.0}}])**

### Formatted

**db.movies.aggregate([**

**{**

**"$match": {**

**"director": "George Lucas"**

**}**

**},**

**{**

**"$sort": {**

**"myRating": -1.0**

**}**

**},**

**{**

**"$limit": 3.0**

**},**

**{**

**"$project": {**

**"\_id": 0.0,**

**"title": 1.0,**

**"director": 1.0,**

**"myRating": 1.0**

**}**

**}**

**])**

### Result:

**{ "title" : "Star Wars: Episode IV - A New Hope", "director" : "George Lucas", "myRating" : 73.6875 }**

**{ "title" : "Star Wars: Episode III - Revenge of the Sith", "director" : "George Lucas", "myRating" : 59.3125 }**

**{ "title" : "Star Wars: Episode II - Attack of the Clones", "director" : "George Lucas", "myRating" : 53 }**