Lab 4

Task 1 :

Please connect to the class Atlas cluster through the mongo shell. The full command is:

```
mongo "mongodb://cluster0-shard-00-00-jxeqq.mongodb.net:27017,cluster0-shard-00-01-jxeqq.mongodb.net:27017,cluster0-shard-00-02-jxeqq.mongodb.net:27017/aggregations?replicaSet=Cluster0-shard-0" --authenticationDatabase admin --ssl -u m121 -p aggregations --norc
```

After connecting to the cluster, ensure you can see the movies collection by typing show collections and then run the command db.movies.findOne(). Take a moment to familiarize yourself with the schema.
Once you have familiarized yourself with the schema, continue to the next tab.

Task 2:

Help MongoDB pick a movie our next movie night! Based on employee polling, we've decided that potential movies must meet the following criteria.

- imdb.rating is at least 7
- genres does not contain "Crime" or "Horror"
- rated is either "PG" or "G"
- languages contains "English" and "Japanese"

Assign the aggregation to a variable named pipeline, like:

```
var pipeline = [ { $match: { ... } } ]
```

- As a hint, your aggregation should return 23 documents. You can verify this by typing db.movies.aggregate(pipeline).itcount()
- Load validateLab1.js into mongo shell

```
load('validateLab1.js')
```

- And run the validateLab1 validation method

```
validateLab1(pipeline)
```

What is the answer?

- 7
- 15
- 12
- 30

Task 3:

Our first movie night was a success. Unfortunately, our ISP called to let us know we're close to our bandwidth quota, but we need another movie recommendation! Using the same $match stage from the previous lab, add a $project stage to only display the the title and film rating (title and rated fields).

- Assign the results to a variable called pipeline.

```
var pipeline = [{ $match: {. . .} }, { $project: { . . . } }]
```

- Load validateLab2.js which was included in the same handout as validateLab1.js and execute validateLab2(pipeline)?

```
load('./validateLab2.js')
```

- And run the validateLab2 validation method

```
validateLab2(pipeline)
```

What is the answer?

- 30
- 15
- 7
- 4

Task 4:

Our movies dataset has a lot of different documents, some with more convoluted titles than others. If we'd like to analyze our collection to find movie titles that are composed of only one word, we could fetch all the movies in the dataset and do some processing in a client application, but the Aggregation Framework allows us to do this on the server!

Using the Aggregation Framework, find a count of the number of movies that have a title composed of one word. To clarify, "Cinderella" and "3-25" should count, where as "Cast Away" would not.

Make sure you look into the *$split* String expression and the *$size* Array expression

To get the count, you can append itcount() to the end of your pipeline

```
db.movies.aggregate([...]).itcount()
```

- 12373
- 9447
- 8068
- 144

Task 5:

MongoDB has another movie night scheduled. This time, we polled employees for their favorite actress or actor, and got these results

```
favorites = [
  "Sandra Bullock",
  "Tom Hanks",
  "Julia Roberts",
  "Kevin Spacey",
  "George Clooney"]
```

For movies released in the USA with a tomatoes.viewer.rating greater than or equal to 3, calculate a new field called num_favs that represets how many favorites appear in the cast field of the movie.
Sort your results by num_favs, tomatoes.viewer.rating, and title, all in descending order.
What is the title of the 25th film in the aggregation result?

- The Heat
- Wrestling Ernest Hemingway
- Erin Brockovich
- Recount

Task 6:

Calculate an average rating for each movie in our collection where English is an available language, the minimum imdb.rating is at least 1, the minimum imdb.votes is at least 1, and it was released in 1990 or after. You'll be required to rescale (or *normalize*)imdb.votes. The formula to rescale imdb.votes and calculate normalized_rating is included as a handout.
What film has the lowest normalized_rating?

- DMZ
- Avatar: The Last Airbender
- The Christmas Tree
- Twilight

Task 7:

In the last lab, we calculated a normalized rating that required us to know what the minimum and maximum values for imdb.voteswere. These values were found using the $group stage!
For all films that won at least 1 Oscar, calculate the standard deviation, highest, lowest, and average imdb.rating. Use the sample standard deviation expression.
HINT - All movies in the collection that won an Oscar begin with a string resembling one of the following in their awards field
`Won 13 Oscars`
`Won 1 Oscar`

Select the correct answer from the choices below. Numbers are truncated to 4 decimal places.

- `{ "highest_rating" : 9.2, "lowest_rating" : 4.5, "average_rating" : 7.5270, "deviation" : 0.5988 }`
- `{ "highest_rating" : 9.2, "lowest_rating" : 4.5, "average_rating" : 7.5270, "deviation" : 0.5984 }`
- `{ "highest_rating" : 9.8, "lowest_rating" : 6.5, "average_rating" : 7.5270, "deviation" : 0.5988 }`
- `{ "highest_rating" : 9.5, "lowest_rating" : 5.9, "average_rating" : 7.5290, "deviation" : 0.5988 }`

Task 8 :

Let's use our increasing knowledge of the Aggregation Framework to explore our movies collection in more detail. We'd like to calculate how many movies every cast member has been in and get an average imdb.rating for each cast member.
What is the name, number of movies, and average rating (truncated to one decimal) for the cast member that has been in the most number of movies with English as an available language?
Provide the input in the following order and format
`{ "_id": "First Last", "numFilms": 1, "average": 1.1 }`

Task 9:

Which alliance from air_alliances flies the most routes with either a Boeing 747 or an Airbus A380 (abbreviated 747 and 380 in air_routes)?

- "OneWorld"
- "Star Alliance"
- "SkyTeam"

Task 10:

Now that you have been introduced to $graphLookup, let's use it to solve an interesting need. You are working for a travel agency and would like to find routes for a client! For this exercise, we'll be using the air_airlines, air_alliances, and air_routes collections in the aggregations database.

- The air_airlines collection will use the following schema:
- ```
  {
      "_id" : ObjectId("56e9b497732b6122f8790280"),
      "airline" : 4,
      "name" : "2 Sqn No 1 Elementary Flying Training School",
      "alias" : "",
      "iata" : "WYT",
      "icao" : "",
      "active" : "N",
      "country" : "United Kingdom",
      "base" : "HGH"
  ```
  ```

```
}
```

- The air_routes collection will use this schema:
- ```
  {
     "_id" : ObjectId("56e9b39b732b6122f877fa31"),
     "airline" : {
           "id" : 410,
           "name" : "Aerocondor",
           "alias" : "2B",
           "iata" : "ARD"
     },
     "src_airport" : "CEK",
     "dst_airport" : "KZN",
     "codeshare" : "",
     "stops" : 0,
     "airplane" : "CR2"
  }
  ```

- Finally, the air_alliances collection will show the airlines that are in each alliance, with this schema:
- ```
  {
     "_id" : ObjectId("581288b9f374076da2e36fe5"),
     "name" : "Star Alliance",
     "airlines" : [
           "Air Canada",
           "Adria Airways",
           "Avianca",
           "Scandinavian Airlines",
           "All Nippon Airways",
           "Brussels Airlines",
           "Shenzhen Airlines",
           "Air China",
           "Air New Zealand",
           "Asiana Airlines",
           "Brussels Airlines",
           "Copa Airlines",
           "Croatia Airlines",
           "EgyptAir",
           "TAP Portugal",
           "United Airlines",
           "Turkish Airlines",
           "Swiss International Air Lines",
  ```

```
        "Lufthansa",
        "EVA Air",
        "South African Airways",
        "Singapore Airlines"
    ]
}
```

Determine the approach that satisfies the following question in the most efficient manner:

Find the list of all possible distinct destinations, with at most one layover, departing from the base airports of airlines that make part of the "OneWorld" alliance. The airlines should be national carriers from Germany, Spain or Canada only. Include both the destination and which airline services that location. As a small hint, you should find 158 destinations.

Select the correct pipeline from the following set of options:

1.
```
db.air_airlines.aggregate(
  [
    {"$match": {"country": {"$in": ["Spain", "Germany",
"Canada"]}}},
    {"$lookup": {
      "from": "air_alliances",
      "foreignField": "airlines",
      "localField": "name",
      "as": "alliance"
    }},
    {"$match": {"alliance.name": "OneWorld"}},
    {"$graphLookup": {
      "startWith": "$base",
      "from": "air_routes",
      "connectFromField": "dst_airport",
      "connectToField": "src_airport",
      "as": "connections",
      "maxDepth": 1
    }},
    {"$project":{ "connections.dst_airport": 1 }},
```

```
    {"$unwind": "$connections"},
    {"$group": { "_id": "$connections.dst_airport" }}
  ]
)
```

2.
```
db.air_routes.aggregate(
  [
    {"$lookup": {
      "from": "air_alliances",
      "foreignField": "airlines",
      "localField": "airline.name",
      "as": "alliance"
    }},
    {"$match": {"alliance.name": "OneWorld"}},
    {"$lookup": {
      "from": "air_airlines",
      "foreignField": "name",
      "localField": "airline.name",
      "as": "airline"
    }},
    {"$graphLookup": {
      "startWith": "$airline.base",
      "from": "air_routes",
      "connectFromField": "dst_airport",
      "connectToField": "src_airport",
      "as": "connections",
      "maxDepth": 1
    }},
    {"$project":{ "connections.dst_airport": 1 }},
    {"$unwind": "$connections"},
    {"$group": { "_id": "$connections.dst_airport" }}
  ]
)
```

3.
```
db.air_alliances.aggregate([{
  $match: { name: "OneWorld" }
}, {
  $graphLookup: {
    startWith: "$airlines",
    from: "air_airlines",
    connectFromField: "name",
    connectToField: "name",
```

```
    as: "airlines",
    maxDepth: 0,
    restrictSearchWithMatch: {
      country: { $in: ["Germany", "Spain", "Canada"] }
    }
  }
}, {
  $graphLookup: {
    startWith: "$airlines.base",
    from: "air_routes",
    connectFromField: "dst_airport",
    connectToField: "src_airport",
    as: "connections",
    maxDepth: 1
  }
}, {
  $project: {
    validAirlines: "$airlines.name",
    "connections.dst_airport": 1,
    "connections.airline.name": 1
  }
},
{ $unwind: "$connections" },
{
  $project: {
    isValid: { $in: ["$connections.airline.name", "$validAirlines"] },
    "connections.dst_airport": 1
  }
},
{ $match: { isValid: true } },
{ $group: { _id: "$connections.dst_airport" } }
])
```

4. ```
   var airlines = [];
   db.air_alliances.find({"name": "OneWorld"}).forEach(function(doc){
     airlines = doc.airlines
   })
   var oneWorldAirlines = db.air_airlines.find({"name": {"$in":
   airlines}})

   oneWorldAirlines.forEach(function(airline){
     db.air_alliances.aggregate([
   ```
   ```

```
  {"$graphLookup": {
    "startWith": airline.base,
    "from": "air_routes",
    "connectFromField": "dst_airport",
    "connectToField": "src_airport",
    "as": "connections",
    "maxDepth": 1
  }}])
})
```