1 MongoDB aggregation

In this section we shall discuss on the different aggregation that the application will implement, the values between «» represent a values passed by an above level.

1 .1 Return the best years by top critic and user ratings

```
db.movie.aggregate([
    {$group:
        {
        _id: "$year",
        topCritic:{$avg:"$top_critic_rating"},
        rate:{$avg:"$user_rating"},
        count:{$sum:1}
        }
    },
    {$match:{count:{$gte:<<i>>}}},
    {$sort:{topCritic:-1, rate:-1}},
    {$limit: <<j>>}
])
```

1 .2 Return the best genres by top critic and user ratings

```
db.movie.aggregate([
    {
        $unwind:"$genres"
    },
    {$group:
        {
            _id: "$genres",
            topCritic:{$avg:"$top_critic_rating"},
            rate:{$avg:"$user_rating"},
            count:{$sum:1}
        }
    },
    {$match:{count:{$gte:<<i>>}}},
    {$sort:{topCritic:-1, rate:-1}},
    {$limit:<<j>>}
])
```

1 .3 Return the best production houses by top critic and user ratings

```
db.movie.aggregate([
    {$group:
        {
            _id: "$production_company",
            topCritic:{$avg:"$top_critic_rating"},
            rate:{$avg:"$user_rating"},
            count:{$sum:1}
        }
    },
    {$match:{count:{$gte:<<i>>}}},
    {$sort:{topCritic:-1, rate:-1}},
    {$limit:<<j>>}
])
```

1 .4 Given a movie count the review it has received by each month

```
db.movie.aggregate([
    {
        $match:{id:<<"id">>}
    },
    {$unwind:"$review"},
    {
        $group:
        {
            _id:{year:{$year:"$review.review_date"}, month:{$month:"$review.review_date"}},
            count:{$sum:1}
        }
    },
    {$sort:{_id:1}}

])
```

1 .5 Given a user count the review he/she has made divided by genres

```
db.movie.aggregate([
    {$match:
        {"review.critic_name":{$eq:<<"name">>}}
    },
    {$unwind:"$genres"},
    {$group:{_id:"$genres", count:{$sum:1}}},
    {$sort:{count:-1}},
    {$limit: <<n>>}
])
```

1 .6 Return the number of user divided by an age bucket

```
db.user.aggregate([
    {
        $match:{"date_of_birth":{$exists:true}}
    },

    {
        $bucket:
        {
            groupBy: {$year:"$date_of_birth"},
            boundaries: [<<values>>],
            output:
            {
                "population": {$sum:1}
            }
        }
    }
])
```