PUNE INSTITUTE OF COMPUTER TECHNOLOGY

LP LAB Practicals

Name: Aditya Kangune **Roll No. : 33323**

Batch: K11 **Academic Year: 2021-22**

Assignment Code 13
"Movie" database

Aim:

Create a NoSQL DB on "Movie"(Movie_id, name, type, budget,date_of_release) using MongoDB and implement the following operations on the document.

- Insert (batch insert, insert validation) the single and multiple documents.
- Remove the documents.
- Update the type of movie.
- Upserts.
- Use the aggregate function to retrieve the big-budget, small-budget movie name.
- Find the average budget for the year "2017"

Output Screenshots:

Note: Code attached in the later document

1) Creating collection “movie”

```
> show collections
Books
Books1
books
inventory
teachers
> db.createCollection("movie")
{ "ok" : 1 }
> use movie
```

2) Inserting a single record:

```
> db.movie.insert({Movie_id:123456789, name:"Harry Potter", type:"Thriller", budget:340000, date_of_release:"12-04-2001"});
WriteResult({ "nInserted" : 1 })
> db.movie.find().pretty()
{
        "_id" : ObjectId("61c96afcf8b3239875f8e034"),
        "Movie_id" : 123456789,
        "name" : "Harry Potter",
        "type" : "Thriller",
        "budget" : 340000,
        "date_of_release" : "12-04-2001"
}
```

3) Inserting multiple documents(Batch Insert):

```
> db.movie.insertMany([
... {Movie_id:7, name:"Interstellar", type:"Fiction", budget:980000, date_of_release:2015},
... {Movie_id:8, name:"Your Name", type:"Anime", budget:20000, date_of_release:2018},
... {Movie_id:2, name:"The Wolf of Wall Street", type:"Dark Comedy", budget:600000, date_of_release:2013}
... ]);
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("61ca186af8b3239875f8e03c"),
                ObjectId("61ca186af8b3239875f8e03d"),
                ObjectId("61ca186af8b3239875f8e03e")
        ]
}
>
```

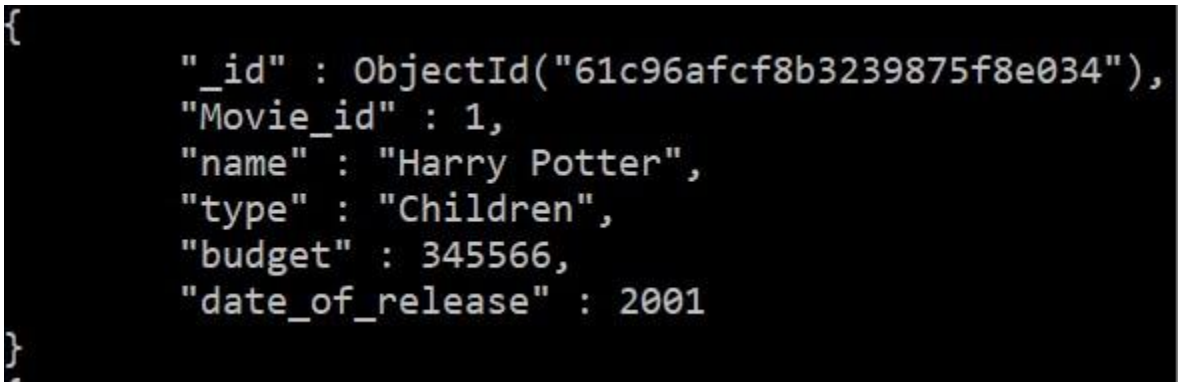
4) Validating the records:

```
> db.runCommand( {collMod: "movie",validator: { $jsonSchema: {bsonType: "object",required: [ "Movie_id", "name" ],properties: {name: {bsonType: "string",description: "Must be a string type"}}}},})
{ "ok" : 1 }
>
```

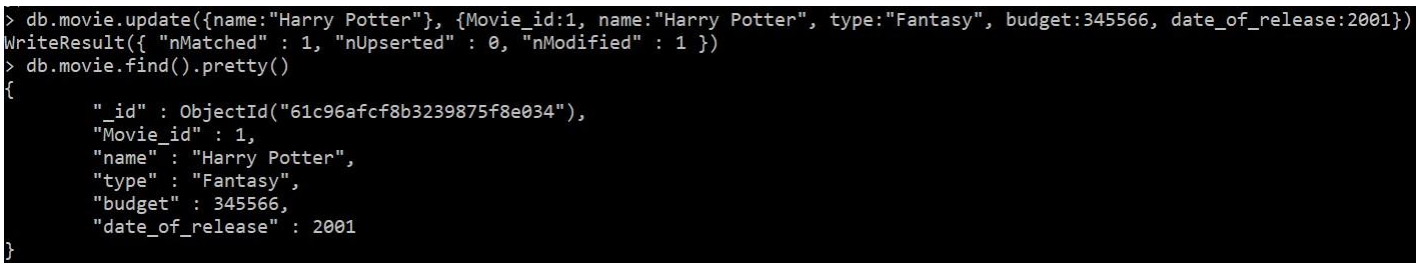
```
> db.movie.insert({name:9999})
WriteResult({
        "nInserted" : 0,
        "writeError" : {
                "code" : 121,
                "errmsg" : "Document failed validation",
                "errInfo" : {
                        "failingDocumentId" : ObjectId("61ca193ff8b3239875f8e03f"),
                        "details" : {
                                "operatorName" : "$jsonSchema",
                                "schemaRulesNotSatisfied" : [
                                        {
                                                "operatorName" : "properties",
                                                "propertiesNotSatisfied" : [
                                                        {
                                                                "propertyName" : "name",
                                                                "details" : [
                                                                        {
                                                                                "operatorName" : "bsonType",
                                                                                "specifiedAs" : {
                                                                                        "bsonType" : "string"
                                                                                },
                                                                                "reason" : "type did not match",
                                                                                "consideredValue" : 9999,
                                                                                "consideredType" : "double"
                                                                        }
                                                                ]
                                                        }
                                                ]
                                        },
                                        {
                                                "operatorName" : "required",
                                                "specifiedAs" : {
                                                        "required" : [
                                                                "Movie_id",
                                                                "name"
                                                        ]
                                                },
                                                "missingProperties" : [
                                                        "Movie_id"
                                                ]
                                        }
                                ]
                        }
                }
        }
})
>
```

5) Updating the type of movie:

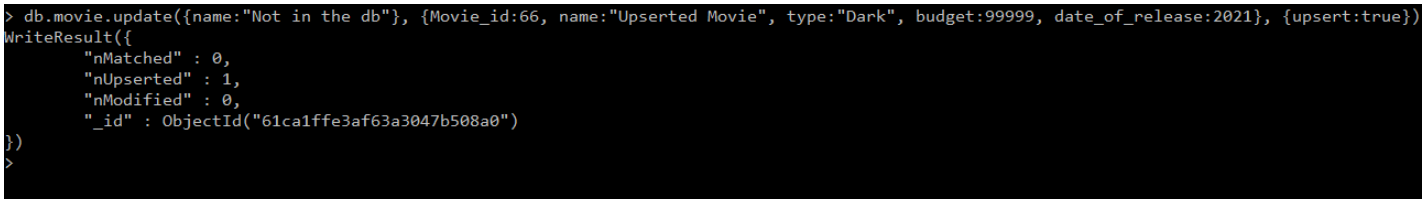
Before updating:



After updating:



6) Upserts:

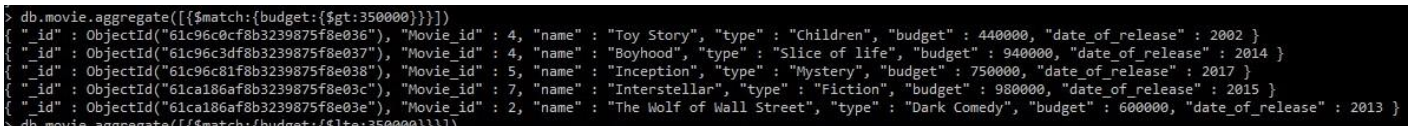


After upserting db becomes:

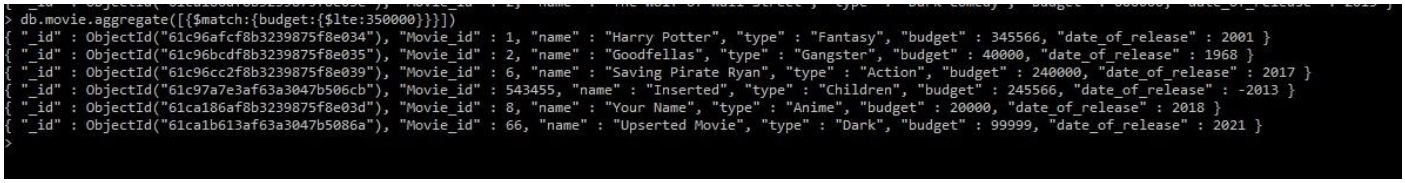
```
{
        "_id" : ObjectId("61ca1b613af63a3047b5086a"),
        "Movie_id" : 66,
        "name" : "Upserted Movie",
        "type" : "Dark",
        "budget" : 99999,
        "date_of_release" : 2021
}
```

7) **Use the aggregate function to retrieve the big-budget, small budget movie:**

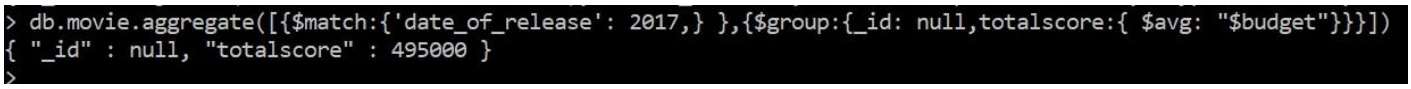
Considering budget > 350000 to be big budget:



Considering budget <= 350000 to be small budget:



8) **Find the average budget for year "2017":**



Final Database: (Before removal)

```
> db.movie.find().pretty()
{
        "_id" : ObjectId("61c96afcf8b3239875f8e034"),
        "Movie_id" : 1,
        "name" : "Harry Potter",
        "type" : "Fantasy",
        "budget" : 345566,
        "date_of_release" : 2001
}
{
        "_id" : ObjectId("61c96bcdf8b3239875f8e035"),
        "Movie_id" : 2,
        "name" : "Goodfellas",
        "type" : "Gangster",
        "budget" : 40000,
        "date_of_release" : 1968
}
{
        "_id" : ObjectId("61c96c0cf8b3239875f8e036"),
        "Movie_id" : 4,
        "name" : "Toy Story",
        "type" : "Children",
        "budget" : 440000,
        "date_of_release" : 2002
}
{
        "_id" : ObjectId("61c96c3df8b3239875f8e037"),
        "Movie_id" : 4,
        "name" : "Boyhood",
        "type" : "Slice of life",
        "budget" : 940000,
        "date_of_release" : 2014
}
{
        "_id" : ObjectId("61c96c81f8b3239875f8e038"),
        "Movie_id" : 5,
        "name" : "Inception",
        "type" : "Mystery",
        "budget" : 750000,
        "date_of_release" : 2017
}
{
        "_id" : ObjectId("61c96cc2f8b3239875f8e039"),
        "Movie_id" : 6,
        "name" : "Saving Pirate Ryan",
        "type" : "Action",
        "budget" : 240000,
        "date_of_release" : 2017
}
{
        "_id" : ObjectId("61c97a7e3af63a3047b506cb"),
        "Movie_id" : 543455,
        "name" : "Inserted",
        "type" : "Children",
        "budget" : 245566,
        "date_of_release" : -2013
}
{
        "_id" : ObjectId("61ca186af8b3239875f8e03c"),
        "Movie_id" : 7,
        "name" : "Interstellar",
        "type" : "Fiction",
```

9) Remove the documents:

Removing one document:

```
> db.movie.remove({name:"Upserted Movie"})
WriteResult({ "nRemoved" : 1 })
>
```

Removing all documents:

```
> db.movie.remove({})
WriteResult({ "nRemoved" : 10 })
> db.movie.find().pretty()
>
>
```

Output Code:

1) Creating collection “movie”

> show collections

Book store

Books

> show dbs

Books 0.000GB

Books1 0.000GB

admin 0.000GB

config 0.000GB

local 0.000GB

mylib 0.000GB

```
teachers  0.000GB
test      0.000GB
> use Books
switched to db Books
> show collections
Books
Books1
books
inventory
teachers
> db.createCollection("movie")
{ "ok" : 1 }
> use movie
switched to db movie
```

2) Inserting a single record:

```
> db.movie.insert({Movie_id:123456789, name:"Harry Potter",
type:"Thriller", budget:340000, date_of_release:"12-04-2001"});
WriteResult({ "nInserted" : 1 })
> db.movie.find().pretty()
{
        "_id" : ObjectId("61c96afcf8b3239875f8e034"),
        "Movie_id" : 123456789,
        "name" : "Harry Potter",
```

"type" : "Thriller",

"budget" : 340000,

"date_of_release" : "12-04-2001"

}

3) Inserting multiple documents(Batch Insert):

> db.movie.insertMany([

... {Movie_id:7, name:"Interstellar", type:"Fiction", budget:980000, date_of_release:2015},

... {Movie_id:8, name:"Your Name", type:"Anime", budget:20000, date_of_release:2018},

... {Movie_id:2, name:"The Wolf of Wall Street", type:"Dark Comedy", budget:600000, date_of_release:2013}

...]);

{

"acknowledged" : true,

"insertedIds" : [

ObjectId("61ca186af8b3239875f8e03c"),

ObjectId("61ca186af8b3239875f8e03d"),

ObjectId("61ca186af8b3239875f8e03e")

]

}

>

4) Validating the records:

```
> db.runCommand( {collMod: "movie",validator: { $jsonSchema: {bsonType: "object",required: [ "Movie_id", "name" ],properties: {name: {bsonType: "string",description: "Must be a string type"}}}},})

{ "ok" : 1 }
```

Trying to insert against the validation:

```
> db.movie.insert({name:9999})

WriteResult({
        "nInserted" : 0,
        "writeError" : {
                "code" : 121,
                "errmsg" : "Document failed validation",
                "errInfo" : {
                        "failingDocumentId" : ObjectId("61ca193ff8b3239875f8e03f"),
                        "details" : {
                                "operatorName" : "$jsonSchema",
                                "schemaRulesNotSatisfied" : [
                                        {
                                                "operatorName" : "properties",
                                                "propertiesNotSatisfied" : [
                                                        {
                                                                "propertyName" : "name",
                                                                "details" : [
                                                                        {
```

```
                                        "operatorName" : "bsonType",
                                        "specifiedAs" : {
                                            "bsonType" : "string"
                                        },
                                        "reason" : "type did not
match",
                                        "consideredValue" : 9999,
                                        "consideredType" : "double"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "operatorName" : "required",
                        "specifiedAs" : {
                            "required" : [
                                "Movie_id",
                                "name"
                            ]
                        },
                        "missingProperties" : [
                            "Movie_id"
```

```
                                ]
                            }
                        ]
                    }
                }
            }
        })
        >
```

5) Updating the type of movie:

Before updating:

```
{
    "_id" : ObjectId("61c96afcf8b3239875f8e034"),
    "Movie_id" : 1,
    "name" : "Harry Potter",
    "type" : "Children",
    "budget" : 345566,
    "date_of_release" : 2001
}
```

After updating:

```
> db.movie.update({name:"Harry Potter"}, {Movie_id:1, name:"Harry Potter", type:"Fantasy", budget:345566, date_of_release:2001})
```

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

> db.movie.find().pretty()

{

"_id" : ObjectId("61c96afcf8b3239875f8e034"),

"Movie_id" : 1,

"name" : "Harry Potter",

"type" : "Fantasy",

"budget" : 345566,

"date_of_release" : 2001

}

6) Upserts:

> db.movie.update({name:"Not in the DB"}, {Movie_id:66, name:"Upserted Movie", type:"Dark", budget:99999, date_of_release:2021}, {**upsert:true**})

WriteResult({

"nMatched" : 0,

"nUpserted" : 1,

"nModified" : 0,

"_id" : ObjectId("61ca1b613af63a3047b5086a")

})

>

After upserting db becomes:

{

"_id" : ObjectId("61ca1b613af63a3047b5086a"),

"Movie_id" : 66,

"name" : "Upserted Movie",

"type" : "Dark",

"budget" : 99999,

"date_of_release" : 2021

}

>

7) **Use the aggregate function to retrieve the big-budget, small budget movie:**

Considering budget > 350000 to be big budget:

> db.movie.aggregate([{$match:{budget:{$gt:350000}}}])

{ "_id" : ObjectId("61c96c0cf8b3239875f8e036"), "Movie_id" : 4, "name" : "Toy Story", "type" : "Children", "budget" : 440000, "date_of_release" : 2002 }

{ "_id" : ObjectId("61c96c3df8b3239875f8e037"), "Movie_id" : 4, "name" : "Boyhood", "type" : "Slice of life", "budget" : 940000, "date_of_release" : 2014 }

{ "_id" : ObjectId("61c96c81f8b3239875f8e038"), "Movie_id" : 5, "name" : "Inception", "type" : "Mystery", "budget" : 750000, "date_of_release" : 2017 }

{ "_id" : ObjectId("61ca186af8b3239875f8e03c"), "Movie_id" : 7, "name" : "Interstellar", "type" : "Fiction", "budget" : 980000, "date_of_release" : 2015 }

{ "_id" : ObjectId("61ca186af8b3239875f8e03e"), "Movie_id" : 2, "name" : "The Wolf of Wall Street", "type" : "Dark Comedy", "budget" : 600000, "date_of_release" : 2013 }

>

Considering budget <= 350000 to be small budget:

> db.movie.aggregate([{$match:{budget:{$lte:350000}}}])

{ "_id" : ObjectId("61c96afcf8b3239875f8e034"), "Movie_id" : 1, "name" : "Harry Potter", "type" : "Fantasy", "budget" : 345566, "date_of_release" : 2001 }

{ "_id" : ObjectId("61c96bcdf8b3239875f8e035"), "Movie_id" : 2, "name" : "Goodfellas", "type" : "Gangster", "budget" : 40000, "date_of_release" : 1968 }

{ "_id" : ObjectId("61c96cc2f8b3239875f8e039"), "Movie_id" : 6, "name" : "Saving Pirate Ryan", "type" : "Action", "budget" : 240000, "date_of_release" : 2017 }

{ "_id" : ObjectId("61c97a7e3af63a3047b506cb"), "Movie_id" : 543455, "name" : "Inserted", "type" : "Children", "budget" : 245566, "date_of_release" : -2013 }

{ "_id" : ObjectId("61ca186af8b3239875f8e03d"), "Movie_id" : 8, "name" : "Your Name", "type" : "Anime", "budget" : 20000, "date_of_release" : 2018 }

{ "_id" : ObjectId("61ca1b613af63a3047b5086a"), "Movie_id" : 66, "name" : "Upserted Movie", "type" : "Dark", "budget" : 99999, "date_of_release" : 2021 }

>

8) Find the average budget for year “2017”:

> db.movie.aggregate([{$match:{'date_of_release': 2017,} },{$group:{_id: null,AverageBudgetIn2017:{ $avg: "$budget"}}}])

{ "_id" : null, "AverageBudgetIn2017" : 495000 }

>

Final Database: (Before removal)

```
> db.movie.find().pretty()
{
        "_id" : ObjectId("61c96afcf8b3239875f8e034"),
        "Movie_id" : 1,
        "name" : "Harry Potter",
        "type" : "Fantasy",
        "budget" : 345566,
        "date_of_release" : 2001
}
{
        "_id" : ObjectId("61c96bcdf8b3239875f8e035"),
        "Movie_id" : 2,
        "name" : "Goodfellas",
        "type" : "Gangster",
        "budget" : 40000,
        "date_of_release" : 1968
}
{
        "_id" : ObjectId("61c96c0cf8b3239875f8e036"),
        "Movie_id" : 4,
```

```
        "name" : "Toy Story",
        "type" : "Children",
        "budget" : 440000,
        "date_of_release" : 2002
}
{
        "_id" : ObjectId("61c96c3df8b3239875f8e037"),
        "Movie_id" : 4,
        "name" : "Boyhood",
        "type" : "Slice of life",
        "budget" : 940000,
        "date_of_release" : 2014
}
{
        "_id" : ObjectId("61c96c81f8b3239875f8e038"),
        "Movie_id" : 5,
        "name" : "Inception",
        "type" : "Mystery",
        "budget" : 750000,
        "date_of_release" : 2017
}
{
```

```
        "_id" : ObjectId("61c96cc2f8b3239875f8e039"),
        "Movie_id" : 6,
        "name" : "Saving Pirate Ryan",
        "type" : "Action",
        "budget" : 240000,
        "date_of_release" : 2017
}
{
        "_id" : ObjectId("61c97a7e3af63a3047b506cb"),
        "Movie_id" : 543455,
        "name" : "Inserted",
        "type" : "Children",
        "budget" : 245566,
        "date_of_release" : -2013
}
{
        "_id" : ObjectId("61ca186af8b3239875f8e03c"),
        "Movie_id" : 7,
        "name" : "Interstellar",
        "type" : "Fiction",
        "budget" : 980000,
        "date_of_release" : 2015
```

```
}
{
    "_id" : ObjectId("61ca186af8b3239875f8e03d"),
    "Movie_id" : 8,
    "name" : "Your Name",
    "type" : "Anime",
    "budget" : 20000,
    "date_of_release" : 2018
}
{
    "_id" : ObjectId("61ca186af8b3239875f8e03e"),
    "Movie_id" : 2,
    "name" : "The Wolf of Wall Street",
    "type" : "Dark Comedy",
    "budget" : 600000,
    "date_of_release" : 2013
}
{
    "_id" : ObjectId("61ca1b613af63a3047b5086a"),
    "Movie_id" : 66,
    "name" : "Upserted Movie",
    "type" : "Dark",
```

```
        "budget" : 99999,
        "date_of_release" : 2021
}
{
        "_id" : ObjectId("61ca1ffe3af63a3047b508a0"),
        "Movie_id" : 66,
        "name" : "Upserted Movie",
        "type" : "Dark",
        "budget" : 99999,
        "date_of_release" : 2021
}
>
```

9) Remove the documents:

Removing one document:

```
> db.movie.remove({name:"Upserted Movie"})
WriteResult({ "nRemoved" : 1 })
>
```

Removing all documents:

```
> db.movie.remove({})
WriteResult({ "nRemoved" : 10 })
> db.movie.find().pretty()
```

>

>

Conclusion:

After the end of the practical,

NoSQL DB on “Movie”(Movie_id, name, type, budget,date_of_release) using MongoDB was implemented along with the following operations:

- Inserting one and many records at once.
- Updating the type of a particular movie.
- Upserts.
- Using the aggregate function to retrieve the big-budget, small-budget movies.
- The average budget for the year “2017”.
- Remove the documents(One and all).