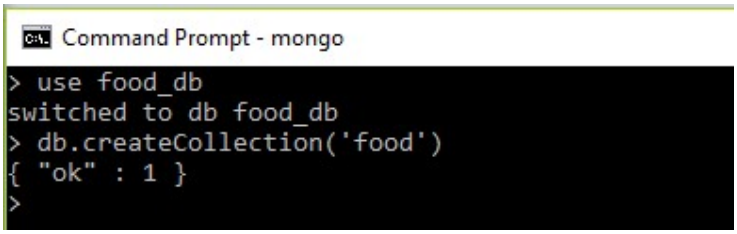
Name: Anushka Harshavadan Nevgi
Roll no: A31
Experiment: 10 Implement arrays and aggregate function in MongoDB
Problem Statement:
A. Implement arrays in MongoDB for a given set of questions.
B. Implement aggregate functions in MongoDB for a given set of questions.

Questions:

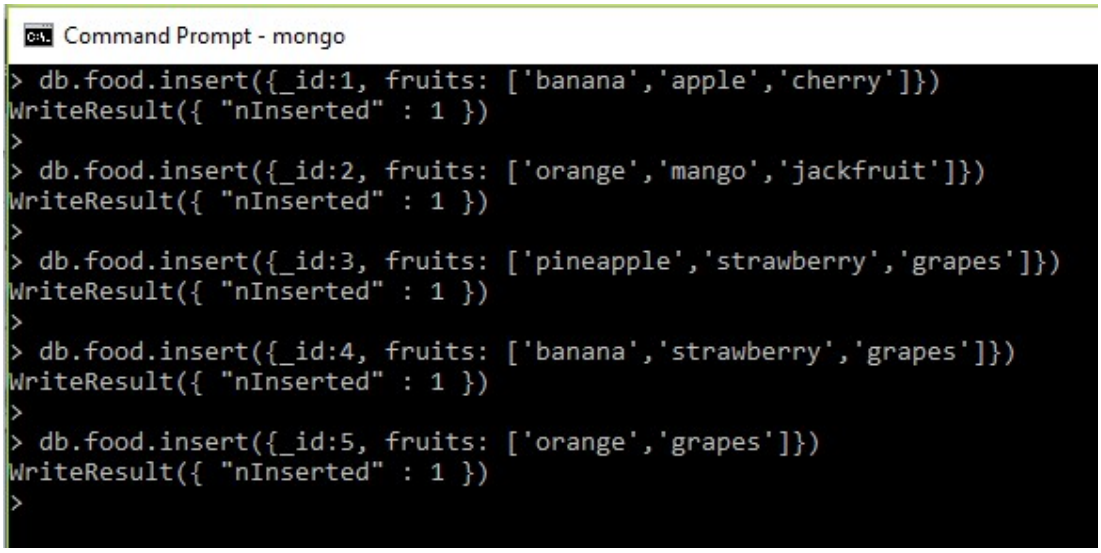
A. Implement arrays in MongoDB for a given set of questions.

1. Create a database 'food_db', Create collection 'food' in 'food_db' and printthe collection.



2. Insert following documents in collection 'food'.

_id	fruits		
1	banana	apple	Cherry
2	orange	mango	Jackfruit
3	pineapple	strawberry	Grapes
4	banana	strawberry	grapes
5	Orange	grapes	--



```
> db.food.find()
{ "_id" : 1, "fruits" : [ "banana", "apple", "cherry" ] }
{ "_id" : 2, "fruits" : [ "orange", "mango", "jackfruit" ] }
{ "_id" : 3, "fruits" : [ "pineapple", "strawberry", "grapes" ] }
{ "_id" : 4, "fruits" : [ "banana", "strawberry", "grapes" ] }
{ "_id" : 5, "fruits" : [ "orange", "grapes" ] }
>
```

3. Print all the documents from collection ‘food’ in a formatted manner.

Command Prompt - mongo

```
> db.food.find().pretty()
{ "_id" : 1, "fruits" : [ "banana", "apple", "cherry" ] }
{ "_id" : 2, "fruits" : [ "orange", "mango", "jackfruit" ] }
{ "_id" : 3, "fruits" : [ "pineapple", "strawberry", "grapes" ] }
{ "_id" : 4, "fruits" : [ "banana", "strawberry", "grapes" ] }
{ "_id" : 5, "fruits" : [ "orange", "grapes" ] }
>
```

4. Find the documents from ‘food collection which has the ‘fruits’ array having‘banana’ as an element.

Command Prompt - mongo

```
> db.food.find( { fruits: 'banana' })
{ "_id" : 1, "fruits" : [ "banana", "apple", "cherry" ] }
{ "_id" : 4, "fruits" : [ "banana", "strawberry", "grapes" ] }
>
```

5. Find the documents from ‘food collection which has the ‘fruits’ array having‘grapes’ in the first index position.

```
> db.food.find({ 'fruits.1':'grapes' })
{ "_id" : 5, "fruits" : [ "orange", "grapes" ] }
>
```

6. Find the documents from ‘food collection where the size of the array is 2.

```
> db.food.find( { 'fruits': {$size: 2} } )
{ "_id" : 5, "fruits" : [ "orange", "grapes" ] }
>
```

7. Find the documents from ‘food collection with ‘_id=1’ and display the firsttwo elements from the array ‘fruits’.

```
> db.food.find( { _id:1 }, { 'fruits': { $slice:2 } } )
{ "_id" : 1, "fruits" : [ "banana", "apple" ] }
>
```

8. Find the documents from ‘food collection which have elements ‘orange’ and‘grapes’ in the array ‘fruits’.

```
> db.food.find( { fruits: { $all: ['grapes','orange'] } } )
{ "_id" : 5, "fruits" : [ "orange", "grapes" ] }
>
```

9. Find the documents from ‘food collection with ‘_id=1’ and display twoelements from the array ‘fruits’ starting with the element at index 1.

```
> db.food.find( { _id:1 }, { fruits: {$slice: [1,3] } } )
{ "_id" : 1, "fruits" : [ "apple", "cherry" ] }
>
```

10. Update the document with ‘_id=4’ and replace ‘fruit’ array element atindex1 with ‘apple’.

```
> db.food.update( { _id:4 }, {$set: {'fruits.1': 'apple'} } )
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
```

Collection after update

```
> db.food.find()
{ "_id" : 1, "fruits" : [ "banana", "apple", "cherry" ] }
{ "_id" : 2, "fruits" : [ "orange", "mango", "jackfruit" ] }
{ "_id" : 3, "fruits" : [ "pineapple", "strawberry", "grapes" ] }
{ "_id" : 4, "fruits" : [ "banana", "apple", "grapes" ] }
{ "_id" : 5, "fruits" : [ "orange", "grapes" ] }
>
```

11. Update the document with ‘_id=4’ and replace ‘fruit’ array element ‘apple’with ‘orange’.

```
> db.food.update( {_id:4, 'fruits': 'apple'}, {$set: {'fruits.$': 'orange'} } )
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
```

Collection after update

```
> db.food.find()
{ "_id" : 1, "fruits" : [ "banana", "apple", "cherry" ] }
{ "_id" : 2, "fruits" : [ "orange", "mango", "jackfruit" ] }
{ "_id" : 3, "fruits" : [ "pineapple", "strawberry", "grapes" ] }
{ "_id" : 4, "fruits" : [ "banana", "orange", "grapes" ] }
{ "_id" : 5, "fruits" : [ "orange", "grapes" ] }
>
```

12. Update the document with ‘_id=2’ and add new array ‘price’.

```
> db.food.update( {_id:2}, {$push: {price: {orange:60, mango:200, jackfruit:150 } } } )
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
```

Collection after update

```
> db.food.find()
{ "_id" : 1, "fruits" : [ "banana", "apple", "cherry" ] }
{ "_id" : 2, "fruits" : [ "orange", "mango", "jackfruit" ], "price" : [ { "orange" : 60, "mango" : 200,
"jackfruit" : 150 } ] }
{ "_id" : 3, "fruits" : [ "pineapple", "strawberry", "grapes" ] }
{ "_id" : 4, "fruits" : [ "banana", "orange", "grapes" ] }
{ "_id" : 5, "fruits" : [ "orange", "grapes" ] }
>
```

13. Update the document with ‘_id=4’ and add element ‘apple’ to the array‘fruits’.

```
> db.food.update( {_id:4}, {$addToSet: {fruits: 'apple'} } )
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
```

Collection after update

```
> db.food.find()
{ "_id" : 1, "fruits" : [ "banana", "apple", "cherry" ] }
{ "_id" : 2, "fruits" : [ "orange", "mango", "jackfruit" ], "price" : [ { "orange" : 60,
 "mango" : 200, "jackfruit" : 150 } ] }
{ "_id" : 3, "fruits" : [ "pineapple", "strawberry", "grapes" ] }
{ "_id" : 4, "fruits" : [ "banana", "orange", "grapes", "apple" ] }
{ "_id" : 5, "fruits" : [ "orange", "grapes" ] }
>
```

14. Update the document with '_id=4' and remove an element from array 'fruits'.
('1' removes the element in an array which is present at the end)

```
> db.food.update( {_id:4}, {$pop: {fruits: 1} } )
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
```

```
> db.food.find()
{ "_id" : 1, "fruits" : [ "banana", "apple", "cherry" ] }
{ "_id" : 2, "fruits" : [ "orange", "mango", "jackfruit" ], "price" : [ { "orange" : 60,
 "mango" : 200, "jackfruit" : 150 } ] }
{ "_id" : 3, "fruits" : [ "pineapple", "strawberry", "grapes" ] }
{ "_id" : 4, "fruits" : [ "banana", "orange", "grapes" ] }
{ "_id" : 5, "fruits" : [ "orange", "grapes" ] }
>
```

('-1' removes the element in an array which is present at the beginning)

```
> db.food.update( {_id:4}, {$pop: {fruits: -1} } )
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
```

```
> db.food.find()
{ "_id" : 1, "fruits" : [ "banana", "apple", "cherry" ] }
{ "_id" : 2, "fruits" : [ "orange", "mango", "jackfruit" ], "price" : [ { "orange" : 60,
 "mango" : 200, "jackfruit" : 150 } ] }
{ "_id" : 3, "fruits" : [ "pineapple", "strawberry", "grapes" ] }
{ "_id" : 4, "fruits" : [ "orange", "grapes" ] }
{ "_id" : 5, "fruits" : [ "orange", "grapes" ] }
>
```

15. Update the document with '_id=3' and remove 'pineapple' and 'grapes' from array 'fruits'.

```
> db.food.update( {_id:3}, {$pullAll: {fruits: ['pineapple','grapes'] } } )
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
```

Collection after update

```
> db.food.find()
{ "_id" : 1, "fruits" : [ "banana", "apple", "cherry" ] }
{ "_id" : 2, "fruits" : [ "orange", "mango", "jackfruit" ], "price" : [ { "orange" : 60,
 "mango" : 200, "jackfruit" : 150 } ] }
{ "_id" : 3, "fruits" : [ "strawberry" ] }
{ "_id" : 4, "fruits" : [ "orange", "grapes" ] }
{ "_id" : 5, "fruits" : [ "orange", "grapes" ] }
>
```

16. Update the document by removing element 'banana' from array 'fruits'.

```
> db.food.update( {fruits: 'banana'}, {$pull: {fruits: 'banana'} } )
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
```

Collection after update

```
> db.food.find()
{ "_id" : 1, "fruits" : [ "apple", "cherry" ] }
{ "_id" : 2, "fruits" : [ "orange", "mango", "jackfruit" ], "price" : [ { "orange" : 60,
 "mango" : 200, "jackfruit" : 150 } ] }
{ "_id" : 3, "fruits" : [ "strawberry" ] }
{ "_id" : 4, "fruits" : [ "orange", "grapes" ] }
{ "_id" : 5, "fruits" : [ "orange", "grapes" ] }
>
```

B.Implement aggregate functions in MongoDB for a given set of questions.

1. Create a database ‘customer_db’, Create collection ‘customer’ in ‘customer_db’ and print the collection.

Command Prompt - mongo

```
> use customer_db
switched to db customer_db
>
> db.createCollection('customer')
{ "ok" : 1 }
>
```

2. Insert following documents in collection ‘customer’.

custID	bal	type
1	500	S
1	900	S
2	1200	S
1	1500	c

```
> db.customer.insert( {custID:1, bal:500, type:'s'} )
WriteResult({ "nInserted" : 1 })
>
> db.customer.insert( {custID:1, bal:900, type:'s'} )
WriteResult({ "nInserted" : 1 })
>
> db.customer.insert( {custID:2, bal:1200, type:'s'} )
WriteResult({ "nInserted" : 1 })
>
> db.customer.insert( {custID:1, bal:1500, type:'c'} )
WriteResult({ "nInserted" : 1 })
>
```

```
> db.customer.find()
{ "_id" : ObjectId("609eabf6fad0e8d6b5f9349a"), "custID" : 1, "bal" : 500, "type" : "s" }
{ "_id" : ObjectId("609eac6bfad0e8d6b5f9349b"), "custID" : 1, "bal" : 900, "type" : "s" }
{ "_id" : ObjectId("609eac83fad0e8d6b5f9349c"), "custID" : 2, "bal" : 1200, "type" : "s" }
{ "_id" : ObjectId("609eac9bfad0e8d6b5f9349d"), "custID" : 1, "bal" : 1500, "type" : "c" }
>
```

3. Print all the documents from collection ‘customer’ in a formatted manner.

```
> db.customer.find().pretty()
{
        "_id" : ObjectId("609eabf6fad0e8d6b5f9349a"),
        "custID" : 1,
        "bal" : 500,
        "type" : "s"
}
{
        "_id" : ObjectId("609eac6bfad0e8d6b5f9349b"),
        "custID" : 1,
        "bal" : 900,
        "type" : "s"
}
{
        "_id" : ObjectId("609eac83fad0e8d6b5f9349c"),
        "custID" : 2,
        "bal" : 1200,
        "type" : "s"
}
{
        "_id" : ObjectId("609eac9bfad0e8d6b5f9349d"),
        "custID" : 1,
        "bal" : 1500,
        "type" : "c"
}
>
```

4. Compute the sum of account balance by grouping on ‘custID’.

```
> db.customer.aggregate( {$group: {_id: '$custID', total: {$sum: '$bal'} } } )
{ "_id" : 1, "total" : 2900 }
{ "_id" : 2, "total" : 1200 }
>
```

5. Compute the sum of account balance by grouping on ‘custID’ whose account type is savings account ‘s’.

```
> db.customer.aggregate( {$match: {type:'s'} }, {$group: {_id: '$custID', total: {$sum: '$bal'} } } )
{ "_id" : 1, "total" : 1400 }
{ "_id" : 2, "total" : 1200 }
>
```

6. Compute the sum of account balance by grouping on ‘custID’ whose account type is savings account ‘s’. Display only those records whose sum isgreater than 1200.

```
> db.customer.aggregate( {$match: {type:'s'} }, {$group: {_id: '$custID', total: {$sum: '$bal'} } },
 {$match: {total: {$gt: 1200} } } )
{ "_id" : 1, "total" : 1400 }
>
```

7. Compute the average of account balance by grouping on ‘custID’ for each group.

```
> db.customer.aggregate( {$group: {_id: '$custID', total: {$avg: '$bal'} } } )
{ "_id" : 1, "total" : 966.6666666666666 }
{ "_id" : 2, "total" : 1200 }
>
```

8. Compute the maximum of account balance by grouping on ‘custID’ for each group.

```
> db.customer.aggregate( {$group: {_id: '$custID', total: {$max: '$bal'} } } )
{ "_id" : 2, "total" : 1200 }
{ "_id" : 1, "total" : 1500 }
>
```

9. Compute the minimum of account balance by grouping on ‘custID’ for each group.

```
> db.customer.aggregate( {$group: {_id: '$custID', total: {$min: '$bal'} } } )
{ "_id" : 1, "total" : 500 }
{ "_id" : 2, "total" : 1200 }
>
```