**DB Model**

1. **Collection –**

Only 1 collection is used – **user Collection**

1. **Document Format (user collection) –**

{

firstName:"Nikhil",

lastName:"Choudhary",

password:"12345678",

calories\_per\_day:1800,

phone:"0123456789",

email:"nikhil@gmail.com",

username:"nikhil",

meals : [

{

"datetime": new Date(),

"name":"pasta",

"calorie": 300,

"description":"Red Sauce Pasta",

"id": <user\_name> + " " + new ObjectId(),

},

{

"datetime": new Date(),

"name":"maggie",

"calorie": 150,

"description":"Maggie",

"id": <user\_name>+ " " + new ObjectId(),

},

],

}

1. **Indexes –**

db.user.createIndex({"unsername":1}, {unique: 1})

1. **Pros (as compared with 2 collection approach) –** 
   1. **All the operations are 1 query operations –** so there is a saving of network calls on all the queries.
   2. **Reads for user specific data read (even all meals related to user) is superfast –** as we have used indexes on the user and from that we can just fetch details using the field names.
   3. **Updates and writes are also fast as we have just used 1 index instead of three**
   4. **No data repetition (in 2 collections). So saves a little space also.**
2. **Cons (As compared with 2 collection approach) –** 
   1. **Meals related Queries (5, 8, 9), for a user with a lot of meals, will be slower.**
3. **Queries –**
4. **Query description -** Write a query to add users to our database. Username while adding a new user should be unique.

**Query –**

db.user.insert({

firstName:"Nikhil",

lastName:"Choudhary",

password:"12345678",

calories\_per\_day:1800,

phone:"0123456789",

email:"nikhil@gmail.com",

username:"nikhil",

meals : [

{

"datetime": new Date(),

"name":"pasta",

"calorie": 300,

"description":"Red Sauce Pasta",

"id":"nikhil" + " " + new ObjectId(),

},

]

});

1. **Query description -** Write a query to delete user from our database based on the username (nikhil).

**Query –**

db.user.remove({“username”: nikhil});

1. **Query description -** Write a query to update user fields like password, caloriesper\_day, phone based on username(nikhil).

**Query –**

db.user.update({username: "nikhil"},{$set : {password: "011110", calories\_per\_day: 2100, phone: "1111111111"}});

1. **Query description -** Write a query to add a meal for a specific user based on the username (nikhil). Meal should have information as described in the meal object above.

**Query –**

db.user.update(

{"username" : "nikhil"},

{$push: {

"meals": {"dateTime": new Date(), "name":"dal", "calorie": 200, "description":"dal", "id":"nikhil" + " " + new ObjectId(),},

},

});

1. **Query description -** Write a query to delete meal based on \_id (123)

**Query –**

var id\_val = "nikhil 5fef95b233ad7dafefdae787";

var username\_val = id\_val.split(" ")[0];

db.user.update( {"username" : username\_val}, {$pull : { meals: { "id" : id\_val} } })

1. **Query description -** Write a query to delete all meals for a specific user as identified by the username (nikhil).

**Query –**

db.user.update( {"username" : "nikhil"}, {$unset : { "meals": 1}, });

1. **Query description -** Write a query to list all meals for a specific user (nikhil).

**Query –**

db.user.find({"username" : "nikhil"}, {"meals": true})

1. **Query description -** Write a query to list all meals for a specific user (nikhil), on a given date (2020-12-30) , ordered by time.

**Query –**

db.user.aggregate([

{$match: {"username" : "nikhil"} },

{

$project: {

filtered\_meals: {

$filter: {

input: "$meals",

as: "meals",

cond: { $and : [

{$gte: [ "$$meals.dateTime", ISODate("2020-12-30T00:00:00Z") ]},

{$lt: [ "$$meals.dateTime", ISODate("2020-12-31T00:00:00Z") ]},

]

}

},

},

},

},

{$unwind: '$filtered\_meals'},

{$sort: {"filtered\_meals.dateTime":1} },

])

1. **Query description -** Write a query which for a specific user (nikhil) for a specific date (2020-12-30), returns true if he has exceeded his daily limit and false otherwise (i.e true if sum of calories of all meals on that day > calories\_per\_day).

**Query –**

var cal = 0;

var cal\_per\_day = /\*get from Session\*/;

var res = false;

db.user.aggregate([

{$match: {"username" : "nikhil"} },

{

$project: {

filtered\_meals: {

$filter: {

input: "$meals",

as: "meals",

cond: { $and : [

{$gte: [ "$$meals.dateTime", ISODate("2020-12-30T00:00:00Z") ]},

{$lt: [ "$$meals.dateTime", ISODate("2020-12-31T00:00:00Z") ]},

]

}

},

},

},

},

{$unwind: '$filtered\_meals'},

]).forEach(function(record) {

cal+=record.filtered\_meals.calorie;

if(cal > cal\_per\_day)

return res = true ;

});

print(res);