## No-SQL Scenario and Queries for Class Exercise:

Myfun Tourist Company is famous for its Solo member engaging different loners to enjoy holidays in group tours across India. Each Tour has the start Location, Destination, number of days, Tour_name, Tour_description, Tour_Minimal_cost, Tour_Zone

Each also has a Day-wise Itinerary in terms of Day_No, ActivityId, Activity_name, Activity_description, City, State, Cost_if_any

A Tourist books a tour and captures details like Tourist_first_name , Tourist_last_name, Tourist_Address, Tourist_State, Tourist_city, Tourist PinCode, Tourist_Phone, Tourist_email_id, Tourist_adhaar_no

Tourplan consists of Tour_name, Tour_start_date, Tour_end_date, Tour_cost, Tour_Tax, Tour_Manager_Name, Tour_status   and The Tourist details booked for that Tour_plan. Tour_status can be "Yet to Start", "Fun Began", "Experienced", "Cancelled"

A special gadget is designed by Myfun which is handed over to each Tourist on start of the Tour_plan and collected back before Tour_end. This gadget has a chat as well as myfun wall to capture tourists experiences, grapples, complaints as well as feedback.

| Question | Response |
|---|---|
| Design the database collections in NoSQL | `Database Name: myfunTour`<br>`Collection Name: Tour with`<br>`     DayWiseItinerary Embedded Document`<br>`Collection Name: TourPlan with`<br>`     TouristDetails Embedded Document`<br>`     TouristExperience Embedded Document` |
| Check on whether<br><br>1. Collections are created?<br><br>2. Give help on the database class<br><br>3. Give help on the collection class | `1. show collections OR`<br>`   db.getCollectionName();`<br><br>`2. db.help()`<br><br>`3. db,mycoll.help()` |
| Insert data into collection Tour, TourPlan | `Refer to Insert collection .text file`<br><br>`1. myfunTour_TourCollectionInsert.txt`<br><br>`2. myfunTour_TourPlanCollectionInsert.txt`<br><br>`3. myfunTour_TourExperienceCollectionInsert.txt` |
| It was observed that the Tour_manager_name is misspelt in many documents from "Mike" to "Mice". Help this to be | `db.TourPlan.updateOne(`<br>`{"TourManagerName" : "Mice"`<br>`},`<br>`{` |

| | |
|---|---|
| corrected | `$set: {"TourManagerName": "Mike"}`<br>`},true`<br>`)` |
| One of the Tour plan was created in month of 1st to 5 August 2023 but had lesser tourists booking. Mark the Tour_plan as cancelled. | `db.TourPlan.updateMany(`<br>`{"$and" : [{"TourStartDate" : ISODate("2023-08-01")},`<br>`{"TourEndDate" : ISODate("2023-08-05")}]`<br>`},`<br>`{`<br>`$set: {"TourStatus" : "Cancelled"}`<br>`}`<br>`);` |
| In Tour_plan "Greenery Munnar" , tourists with last_name "Dirv" opted out and are required to be removed from the Tour plan | // Note the delete query removes the entire plan rather than just one tourist. Also if u use deleteMany, it would remove all tourplans with this tourist<br>`db.TourPlan.deleteOne(`<br>`{"Tourist.TouristName.LastName" : "Dev"}`<br>`);`<br>// Modify the delete statement if only 1 plan is to be removed.<br>`db.TourPlan.deleteOne(`<br>`{"$and" :`<br>`[`<br>`{"Tourist.TouristName.LastName" : "Dirv"},`<br>`{"TourStartDate" : {"$gte" : ISODate("2023-09-01")}}`<br>`]`<br>`}`<br>`);`<br>// Modify the delete statement if only 1 document is to be removed. |
| | |

## Query Criteria, RegEx, Text Search, Projection, Sort, Limit

| SNo | Questions | Response |
|---|---|---|
| 1 | Find tour named "Blissful Nainital" | `db.Tour.find({"TourName":"Blissful Nainital"});` |
| 2 | Find tours whose tour minimal cost is less than | `db.Tour.find({"TourMinCost":{"$lt":5000}},{"TourName":1,_id:0});` |

| | | |
|---|---|---|
| | 5000 | |
| 3 | Find tours whose tour minimal cost is greater than 4000 | `db.Tour.find({"TourMinCost":{"$gt":4000}},{"Tou`<br>`rName":1,_id:0});` |
| 4 | Find tours whose tour tax is between 2000 and 4000 | `db.Tour.find({"TourMinCost":{"$gt":2000,"$lt":`<br>`4000}},{"TourName":1,_id:0});` |
| 5 | Find tours whose itinerary cost is greater than 200. Display tour name, the itinerary day number which matches the criteria. | `db.Tour.find({"Itinerary.ActivityCost": {$gte :`<br>`200}},{TourName:1,_id:0,"Itinerary.Day_No.$":1}`<br>`);`<br>`//Note .$ will display all those embedded`<br>`//documents that match the criteria. In`<br>`//display only on one attribute.` |
| 6 | Find tour whose activity name has “DelhiCityVisit” | `db.Tour.find({"Itinerary.ActivityName":"DelhiCi`<br>`tyVisit"},{_id:0,TourName:1,"Itinerary.Day_No":`<br>`1,"Itinerary.ActivityName":1});` |
| 7 | Find tourists who have taken "Blissful Nainital" | `// use the find operator`<br><br>`db.TourPlan.find({"TourName" : "Blissful`<br>`Nainital"}, {"_id": 0, "Tourist" : 1});` |
| 8 | Find tourists who are above 40 and taken " Blissful Nainital " | `// use of find with $and , $gte`<br>`db.TourPlan.find( {"$and":[ {"TourName" :`<br>`"Blissful Nainital"},`<br>`{"Tourist.TouristAge" :  {"$gte" : 40}} ] } );` |
| 9 | Find tourist who are between 20 and 30 and their tour plan details. | `// use of $gte and $lte`<br>`db.TourPlan.find(`<br>`{"Tourist.TouristAge":{"$gte" : 20, "$lte" : 40`<br>`} }`<br>`);` |
| 10 | Find all tours whose status is in cancelled, completed | `//use of $in`<br>`db.TourPlan.find({"TourStatus":`<br>`{"$in":["Cancelled","Completed"]}},{TourName:1,`<br>`_id:0,TourStatus:1,TourManagerName:1});` |
| 11 | Find all tours whose status is not in “Cancelled”, “Fun Began" | `// use of $nin`<br>`db.TourPlan.find(`<br>`{"TourStatus":{"$nin" : ["Cancelled", "Fun`<br>`Began" ] } } );` |
| 12 | Find those tours which are neither economical nor luxury.<br><br>Economy is below or | `db.Tour.find({$nor : [{TourMinCost : {$gte :`<br>`80000}}, {TourMinCost : {$lte:10000}}]},{_id:0,`<br>`TourName:1,TourMinCost:1});` |

| | | |
|---|---|---|
| | equal 10k and luxury is equal and above 80k | |
| 13 | Find those tours only whose cost are economical or ones which are luxury. Economy is where actual tour cost below 20K and luxury is above 50k | `db.Tour.find({$or : [{TourMinCost : {$gte : 50000}}, {TourMinCost : {$lte:20000}}]},{_id:0, TourName:1,TourMinCost:1});` |
| 14 | Find tours which has names containing blissful or BLISSFUL. | // use of RegEx search<br><br>`db.TourPlan.find({"TourName" : /blissful/i});` |
| 15 | Find tours which has names start with Blis or blis | db.Tour.find({"TourName":{$regex : /^Blis/i}}) |
| 16 | It is required to search all the tours which has been executed by tour_managers “Rajeev”,”Pritesh”, “Mice”. Let’s try the Text Search here. Restrict the columns to display in output to Tourname, manager name as well as tourstatus | `// Use of text Search`<br><br>`db.TourPlan.createIndex({ TourManagerName : "text"});`<br>`db.TourPlan.find({$text:{$search:"Pritesh Rajeev Mice"}},{TourName:1, _id:0,TourStatus:1,TourManagerName:1})` |
| 17 | Find the tour name, Date when experience was given, Experience comments who have commented “awesome experience ” , “are changing” in their experiences. | `db.TourExperience.createIndex({ "ExpComment.TouristReview" : "text"});`<br><br>`db.TourExperience.find({$text:{$search:"'\awesome experience\' '\are changing\'"}},{TourName:1, _id:0,TourStatus:1,ExpDate:1,"ExpComment" : 1});` |
| 18 | Find the Tours available. Get only the 3 out of them. | `//Use of limit`<br>`db.Tour.find({},{"Itinerary":0,_id:0,TourDesc:0 }).limit(3);` |

| | | |
|---|---|---|
| 19 | Find the TourName and Tourdesc for all the tours available. | `//Use of inclusion projection`<br><br>`db.Tour.find({},{TourName:1,TourDesc:1});`<br><br>`//In this case, we cannot give any other`<br>`//fields to 0 except _id.` |
| 20 | Find the Tour details for all the tours available without the itinerary details. | `// use of Exclusion projection`<br><br>`db.Tour.find({},{"Itinerary":0,_id:0});`<br><br>`// in this case, we cannot give any fields for display as :1, We have to identify fields to exclude.` |
| 21 | Find all the Tour sorted on the Tour Name alphabetically. | `// use of sort`<br><br>`db.Tour.find({},{"TourName":1,"TourDesc":1}).sort({"TourName": 1});` |
| 22 | Find all the Tour sorted on the Tour Name reverse | `// use of sort`<br><br>`db.Tour.find({},{"TourName":1,"TourDesc":1}).sort({"TourName": -1});` |
| 23 | Find the tours which has itinerary Activity cost greater or equal to 200 and also have ActivityCity as “Assam, Cochin” | `// here we need to use $elemMatch as we //need to get both embedded document value //to be matched with $and.`<br><br>`db.Tour.find({"Itinerary":`<br>`{"$elemMatch":{"ActivityCost" : {"$gte" :`<br>`200},"City" : {"$in" : ["Cochin","Assam"]} }}`<br>`});` |
| 24 | Find all the tour plans which started between 20 July 2022 to 1 Jan 2023. | |

**Aggregation FW :**

| Sno | Queries | Response |
|---|---|---|
| 1 | For the tour plans with status completed and Tour start date is greater than 1 Oct 2022, get their tour details. Get the tour_name, tour_description, tourNoofdays, tour_start_date, tour_end_date, tour_plan_status, tour_managerName | `// Here we use two aggregation f/w stages`<br><br>`//$match – this is to extract data matching the criteria. In our case, its tour status as completed and Tour plan start date is greater than 1 oct 2022..`<br><br>`//$lookup – this is to get parent tour details using the tour_name which is common in tour_plan and tour.`<br><br>`//$set – to set the fields which are to be displayed from the referenced collection.`<br><br>`//$unset – to remove the other fields from the referenced collection.`<br><br>`Refer to AggregationFW_Response.txt, Ans-1` |
| 2 | Find all the tours available and get the relevant tour details like Tour Name, Tour Number of days, Day number, daywise total ActivityCost, total number of activities per day sorting them on the Tour Name and day number | `// Here we use two aggregation f/w stages`<br><br>`//$unwind : This makes the parent fields attributes to be repeated for each embedded document. Here for each day no, tour details like tour name, start location all those details will be repeated.`<br><br>`// $group : This does a grouping on the selected fields and helps in doing sum or average on numerical fields.`<br><br>`// $sort : This sorts the output in ascending or descending order`<br><br>`Refer to AggregationFW_Response.txt, Ans-2` |
| | | |
| 3 | Find Tours with number of activities per day. Provide details like Tour Name, Tour Start Location, Tour End location, Day_No, Count of | `// This is a case of unwinding the day`<br>`//itinerary and`<br>`// using the $group, $count.. Note to get`<br>`//the tour start locn, end location u`<br>`//might have to add them in the $group`<br><br>`var pipeline = [` |

| | | |
|---|---|---|
| | Activities per day | `// unwind Itinerary`<br>`{"$unwind": "$Itinerary"},`<br><br>`//Group by TourName`<br>`{"$group" : {"_id" : {"TourName" : "$TourName", "TourStartLocn" : "$TourStartLocn", "TourEndLocn" : "$TourEndLocn" , "TourNoOfDays" : "$TourNoOfDays", "Day_No" : "$Itinerary.Day_No"}, "TotalActivities" : { $sum : 1 } } },`<br><br>`// Omit unwanted fields`<br>` {"$unset": [`<br>`   "Itinerary",`<br>` ]},`<br><br>`// sort on Tourname, day_no`<br>`{"$sort": {`<br>`  "_id": 1,`<br>`}},`<br><br>`];`<br><br>`db.Tour.aggregate(pipeline);` |
| 4 | Find all the tour_plan for current financial year and get the relevant tour details like TourstartLocation, end location, number of days, count of tourists | // Use of $match to find the plans in the current year<br><br>// Use of $lookup for the tour related details from Tour.<br><br>`var pipeline = [`<br>`{"$match": {"TourStartDate": {"$gt" : ISODate("2023-04-01"), "$lt": ISODate("2024-03-31") } } },`<br><br>`// unwind Tourists`<br>`{"$unwind": "$Tourist"},`<br><br>`//Group by TourName`<br>` {"$group" : {"_id" : {"TourName" : "$TourName"}, "TotalTourists" : { $count : {} } } },`<br><br>`// lookup to the Tour for details of tour.`<br>`   {`<br>`      $lookup: {`<br>`         from: "Tour",`<br>`         localField: "_id.TourName",` |

```
// field in the TourPlan collection
         foreignField: "TourName",
// field in the Tour collection
         as: "fromTour",
      }},
// For this data model, will always be 1
record in right-side
  // of join, so take 1st joined array
element
  {"$set": {
    "fromTour": {"$first": "$fromTour"},
  }},
  // Extract the joined embeded fields
into top level fields
  {"$set": {
    "TourStartLocn":
"$fromTour.TourStartLocn",
    "TourEndLocn":
"$fromTour.TourEndLocn",
    "TourNoOfDays":
"$fromTour.TourNoOfDays",
   }},
   // Omit unwanted fields
  {"$unset": [
    "fromTour",
  ]},
  // sort on TourName
  {"$sort": {
    "_id": 1,
  }},

];

db.TourPlan.aggregate(pipeline);
```