1]Creating database employee

****

Create collections emp_personal_details with emp_id, emp_name, emp_address, emp_DOB, emp_age, emp_mobilenumber

```
test> use employee

switched to db employee

employee> db.createCollection("emp_personal_details");


employee> db.emp_personal_details.insertOne({

emp_id:1,

emp_name:"pooja",

 emp_address:"jalgaon",

 emp_DOB:new Date("2003-01-19"),

 emp_age:22,

 emp_mobilenumber:9087384328})


employee> db.emp_personal_details.find();

output:-

[

  {

    _id: ObjectId('68cbee53acf3de0d43718dc4'),

    emp_id: 1,

    emp_name: 'pooja',

    emp_address: 'jalgaon',

    emp_DOB: ISODate('2003-01-19T00:00:00.000Z'),

    emp_age: 22,

    emp_mobilenumber: 9087384328

  }

]
```

2] Create another collection emp_professional_details with emp_id, emp_name, designation , salary , incentive, working hours

****

```
db.createCollection("emp_professional_details");
employee> db.emp_professional_details.insertOne({
 emp_id : 1,
 emp_name:"siya",
 designation:"manager",
 salary:90000,
 incentive:5000,
 working_hour:30})

employee> db.emp_professional_details.find();

output:-
[
  {
    _id: ObjectId('68cbef80acf3de0d43718dc5'),
    emp_id: 1,
    emp_name: 'siya',
    designation: 'manager',
    salary: 90000,
    incentive: 5000,
    working_hour: 30
  }
]
```

3] 1)Insert 10 records in collection emp_personal_details and emp professional details

****

```
   db.emp_personal_details.insertMany([
  {
   emp_id: 1,
   emp_name: "Pooja",
   emp_address: "Jalgaon",
   emp_DOB: new Date("2003-01-19"),
   emp_age: 22,
   emp_mobilenumber: 9087384328
  },
  {
   emp_id: 2,
   emp_name: "Rahul",
   emp_address: "Mumbai",
   emp_DOB: new Date("1995-05-15"),
   emp_age: 28,
   emp_mobilenumber: 9876543210
  },
  {
   emp_id: 3,
   emp_name: "Anita",
   emp_address: "Pune",
   emp_DOB: new Date("1992-11-12"),
   emp_age: 31,
   emp_mobilenumber: 9988776655
  },
  {
```

```
  emp_id: 4,
  emp_name: "Sanjay",
  emp_address: "Delhi",
  emp_DOB: new Date("1990-08-20"),
  emp_age: 33,
  emp_mobilenumber: 9123456789
},
{
  emp_id: 5,
  emp_name: "Neha",
  emp_address: "Bangalore",
  emp_DOB: new Date("1998-03-05"),
  emp_age: 25,
  emp_mobilenumber: 9871234567
},
{
  emp_id: 6,
  emp_name: "Vikram",
  emp_address: "Chennai",
  emp_DOB: new Date("1993-07-22"),
  emp_age: 30,
  emp_mobilenumber: 9012345678
},
{
  emp_id: 7,
  emp_name: "Priya",
  emp_address: "Hyderabad",
  emp_DOB: new Date("2000-12-11"),
  emp_age: 23,
  emp_mobilenumber: 9870987654
  },
  {
    emp_id: 8,
    emp_name: "Amit",
    emp_address: "Kolkata",
    emp_DOB: new Date("1996-06-30"),
    emp_age: 27,
    emp_mobilenumber: 9765432109
  },
  {
    emp_id: 9,
    emp_name: "Sonal",
    emp_address: "Nagpur",
    emp_DOB: new Date("1994-09-18"),
    emp_age: 29,
    emp_mobilenumber: 9123459876
  },
  {
    emp_id: 10,
    emp_name: "Rohan",
    emp_address: "Ahmedabad",
    emp_DOB: new Date("1997-02-25"),
    emp_age: 26,
    emp_mobilenumber: 9988123456
  }
 ])

employee> db.emp_professional_details.insertMany([
  db.emp_professional_details.insertMany([
  {
```

```
  emp_id: 1,
  emp_name: "Siya",
  designation: "Manager",
  salary: 90000,
  incentive: 5000,
  working_hour: 30
},
{
  emp_id: 2,
  emp_name: "Rahul",
  designation: "Developer",
  salary: 75000,
  incentive: 3000,
  working_hour: 40
},
{
  emp_id: 3,
  emp_name: "Anita",
  designation: "Designer",
  salary: 70000,
  incentive: 2500,
  working_hour: 35
},
{
  emp_id: 4,
  emp_name: "Sanjay",
  designation: "Team Lead",
  salary: 85000,
  incentive: 4000,
  working_hour: 38
```

```
},
{
  emp_id: 5,
  emp_name: "Neha",
  designation: "Developer",
  salary: 72000,
  incentive: 2800,
  working_hour: 40
},
{
  emp_id: 6,
  emp_name: "Vikram",
  designation: "Tester",
  salary: 65000,
  incentive: 2000,
  working_hour: 36
},
{
  emp_id: 7,
  emp_name: "Priya",
  designation: "HR",
  salary: 60000,
  incentive: 1500,
  working_hour: 35
},
{
  emp_id: 8,
  emp_name: "Amit",
  designation: "Developer",
  salary: 73000,
    incentive: 2700,
    working_hour: 40
  },
  {
    emp_id: 9,
    emp_name: "Sonal",
    designation: "Designer",
    salary: 71000,
    incentive: 2600,
    working_hour: 37
  },
  {
    emp_id: 10,
    emp_name: "Rohan",
    designation: "Manager",
    salary: 88000,
    incentive: 4500,
    working_hour: 39
  }
 ])
```

---

2. Show all the employees having designation manager

```
     db.emp_professional_details.find({designation: "Manager"});
```

output:-

```
[
  {
    _id: ObjectId('68cbf10facf3de0d43718dd0'),
    emp_id: 1,
    emp_name: 'Siya',
    designation: 'Manager',
    salary: 90000,
    incentive: 5000,
    working_hour: 30
  },
  {
    _id: ObjectId('68cbf10facf3de0d43718dd9'),
    emp_id: 10,
    emp_name: 'Rohan',
    designation: 'Manager',
    salary: 88000,
    incentive: 4500,
    working_hour: 39
  }
]
```

---

3. Show all the employees having salary 6000

```
     db.emp_professional_details.find({salary:60000});
```

output:-

```
[
  {
    _id: ObjectId('68cbf10facf3de0d43718dd6'),
    emp_id: 7,
    emp_name: 'Priya',
    designation: 'HR',
    salary: 60000,
    incentive: 1500,
    working_hour: 35
  }
]
```

4]1) Update the collection emp_personal_details , add field status and set it to retired where age is greater than 60.

****

```
db.emp_personal_details.updateMany(
{ emp_age: { $gt: 60 } },
 { $set: { status: "retired" } }
)
```

Output :-

```
{
   _id: ObjectId('68cbf476acf3de0d43718dda'),
   emp_id: 11,
   emp_name: 'Ramesh',
   emp_address: 'Nagpur',
   emp_DOB: ISODate('1955-03-10T00:00:00.000Z'),
   emp_age: 68,
   emp_mobilenumber: 9123454321,
   status: 'retired'
  },
  {
   _id: ObjectId('68cbf476acf3de0d43718ddb'),
   emp_id: 12,
   emp_name: 'Sushma',
   emp_address: 'Pune',
   emp_DOB: ISODate('1958-07-22T00:00:00.000Z'),
   emp_age: 65,
   emp_mobilenumber: 9876541230,
   status: 'retired'
  },
  {
   _id: ObjectId('68cbf476acf3de0d43718ddc'),
   emp_id: 13,

   emp_name: 'Vikram Singh',

   emp_address: 'Delhi',

   emp_DOB: ISODate('1950-11-05T00:00:00.000Z'),

   emp_age: 73,

   emp_mobilenumber: 9012345678,

   status: 'retired'

  }
```

---

2. Update collection emp_professional_details, give incentive 5000 to employees whose working hours is greater than 45 per week

```
     db.emp_professional_details.updateMany(

   { working_hour: { $gt: 45 } },

   { $set: { incentive: 5000 } }

 )
```

Output:-

```
{

    _id: ObjectId('68cbf5bcacf3de0d43718ddd'),

    emp_id: 11,

    emp_name: 'Ajay',

    designation: 'Developer',

    salary: 80000,

    incentive: 5000,

    working_hour: 46

  },

  {

    _id: ObjectId('68cbf5bcacf3de0d43718dde'),

    emp_id: 12,

    emp_name: 'Seema',
    designation: 'Tester',
    salary: 65000,
    incentive: 5000,
    working_hour: 48
  },
  {
    _id: ObjectId('68cbf5bcacf3de0d43718ddf'),
    emp_id: 13,
    emp_name: 'Rohit',
    designation: 'Accountant',
    salary: 70000,
    incentive: 5000,
    working_hour: 50
  }
```

---

3. Add 1000 to the salary employee whose designation is accountant

```
db.emp_professional_details.updateMany(
   { designation: "Accountant" },
   { $inc: { salary: 1000 } }
 )
```

****

5]1) Create index on emp_id in collection emp_professional_details

```
      db.emp_professional_details.createIndex({ emp_id: 1 })
```

---

2. Create multiple index on emp_id,emp_name in collection emp_professonal details

```
      db.emp_professional_details.createIndex({ emp_id: 1, emp_name: 1 })
      db.emp_professional_details.getIndexes()
```

output:-

```
 [
 { v: 2, key: { emp_id: 1 }, name: 'emp_id_1' },
 {
   v: 2,
   key: { emp_id: 1, emp_name: 1 },
   name: 'emp_id_1_emp_name_1'
 }
]
```

**;

6]1) Find sum of salaries of employees having designation clerk.

```
  db.emp_professional_details.aggregate([
    { $match: { designation: "Clerk" } },       // filter by designation
    { $group: { _id: null, totalSalary: { $sum: "$salary" } } }  // sum salaries
])
```

---

2) Filter the employees having the designation software engineer and find the minimum salary.

```
db.emp_professional_details.aggregate([
  { $match: { designation: "Software Engineer" } },  // filter by designation
  { $group: { _id: null, minSalary: { $min: "$salary" } } } // find min salary
])
```

****

7]1) Use unwind command and show the employees whose mobile number is stored in array

```
  db.emp_personal_details.aggregate([
  { $unwind: "$emp_mobilenumber" },  // unwind array field
  { $project: { emp_id: 1, emp_name: 1, emp_mobilenumber: 1 } }
])
```

---

2. Use skip command to skip first 3 records and display rest of records

```
db.emp_personal_details.find().skip(3)
```

---

3. Use limit command to show only first four records of collection

```
db.emp_personal_details.find().limit(4)
```

****

8] Create replica set of employee database and insert records in primary node and display the same records in secondary nodes

****

****

9] Create a MongoDB collection named restaurants to store the following information about restaurants:

Building number

Street nameZip code

Coordinates (longitude and latitude)

Borough

Cuisine type

Grades (each grade includes: date, grade (A/B/C), and score)

```
db.restaurants.insertOne({
  restaurant_id: "12345",
  name: "Good Eats",
  building_number: "100",
  street_name: "Main St",
  zip_code: "10001",
  coordinates: { longitude: -73.856077, latitude: 40.848447 },
  borough: "Bronx",
  cuisine: "American",
  grades: [
    { date: ISODate("2025-09-01"), grade: "A", score: 95 },
    { date: ISODate("2025-06-01"), grade: "B", score: 88 }
  ]
 });
```

****

10] 1).Write a MongoDB query to display all the documents in the collection restaurants

```
db.restaurants.find()
```

---

2). Write a MongoDB query to display the fields,restaurant_id, name, borough and cuisine for all the documents in the collection restaurant

```
db.restaurants.find({}, { restaurant_id: 1, name: 1, borough: 1, cuisine: 1 })
```

****

11] 1)Write a MongoDB query to display the fields restaurant_id, name, borough and cuisine, but exclude the field _id for all the documents in the collection restaurant

```
db.restaurants.find({}, { _id: 0, restaurant_id: 1, name: 1, borough: 1, cuisine: 1 })
```

---

2). Write a MongoDB query to display all the restaurant which is in the borough Bronx

```
db.restaurants.find({ borough: "Bronx" })
```

****

12] 1). Write a MongoDB query to display the first 5 restaurants which are in the borough Bronx.

```
db.restaurants.find({ borough: "Bronx" }).limit(5)
```

---

2) Write a MongoDB query to display the next 5 restaurants after skipping first 5 which are in the borough Bronx

```
db.restaurants.find({ borough: "Bronx" }).skip(5).limit(5)
```

****

13] 1).Write a MongoDB query to find the restaurants who achieved a score more than 90

```
 db.restaurants.find({ "grades.score": { $gt: 90 } })
```

---

2).Write a MongoDB query to find the restaurantsthat achieved a score, more than 80 but less than 100

```
 db.restaurants.find({ "grades.score": { $gt: 80, $lt: 100 } })
```

****

14] Write a MongoDB query to find the restaurants which do not prepare any cuisine of 'American ' and achieved a grade point 'A' not belonging to the boroughBrooklyn. The document must be displayed according to the cuisine in descending order

```
db.restaurants.find({
  cuisine: { $ne: "American" },
  "grades.grade": "A",
  borough: { $ne: "Brooklyn" }
}).sort({ cuisine: -1 })
```

****

15] Write a MongoDB query to find the restaurant Id,name, borough and cuisine for those restaurants which contain 'Wil' as first three letters for its name

```
db.restaurants.find(
  { name: /^Wil/ },
  { _id: 0, restaurant_id: 1, name: 1, borough: 1, cuisine: 1 }
)
```

****

16] Write a MongoDB query to find the restaurant Id,name, borough and cuisine for those restaurants which contain 'ces' as the last three letters for its name.

```
db.restaurants.find(
  { name: /ces$/ },
  { _id: 0, restaurant_id: 1, name: 1, borough: 1, cuisine: 1 }
)
```

****

17] Write a MongoDB query to find the restaurant Id,name, borough and cuisine for those restaurants which contain 'Reg' as three letters somewhere in its name

```
   db.restaurants.find(
  { name: /Reg/ },
  { _id: 0, restaurant_id: 1, name: 1, borough: 1, cuisine: 1 }
)
```

****

18] Write a MongoDB query to find the restaurants which belong to the borough Bronx and prepared either American or Chinese dish

```
   db.restaurants.find({
  borough: "Bronx",
  cuisine: { $in: ["American", "Chinese"] }
})
```

****

19] Write a MongoDB query to find the restaurant Id,name, borough and cuisine for those restaurants which belong to the borough Staten Island or Queens or Bronx or Brooklyn.

```
     db.restaurants.find(
  { borough: { $in: ["Staten Island", "Queens", "Bronx", "Brooklyn"] } },
  { _id: 0, restaurant_id: 1, name: 1, borough: 1, cuisine: 1 }
)
```

****

20] Write a MongoDB query to find the restaurant Id,name, borough and cuisine for those restaurants which are not belonging to the borough Staten Island Or Queens or Bronxor Brooklyn.

```
   db.restaurants.find(
 { borough: { $nin: ["Staten Island", "Queens", "Bronx", "Brooklyn"] } },
 { _id: 0, restaurant_id: 1, name: 1, borough: 1, cuisine: 1 }
)
```

****

21] Write a MongoDB query to find the restaurant Id,name, borough and cuisine for those restaurants which achieved a score which is not more than 10

```
db.restaurants.find(
 { "grades.score": { $lte: 10 } },
 { _id: 0, restaurant_id: 1, name: 1, borough: 1, cuisine: 1 }
)
```

****

22] Write a MongoDB query to find the restaurant Id,name, borough and cuisine for those restaurants which prepared dish except 'American' and 'Chinese' or restaurant's name begins with letter 'Wil'

```
 db.restaurants.find({
 $or: [
   { cuisine: { $nin: ["American", "Chinese"] } },
   { name: /^Wil/ }
 ]
}, { _id: 0, restaurant_id: 1, name: 1, borough: 1, cuisine: 1 })
```

****

23] Write a MongoDB query to arrange the name of the restaurants in descending along with all the columns

```
   db.restaurants.find().sort({ name: -1 })
```

****

24] Write a MongoDB query to arranged the name of the cuisine in ascending order and for that same cuisine borough should be in descending order.

```
   db.restaurants.find().sort({ cuisine: 1, borough: -1 })
```

****

25] Write a MongoDB query to know whether all the addresses contains the street or no

```
   db.restaurants.find({ street_name: { $exists: false } })
```

****

26] Write a MongoDBquery which will select all documents in the restaurants collection where the coord field value is Double

```
db.restaurants.find({
   "coordinates.longitude": { $type: "double" },
   "coordinates.latitude": { $type: "double" }
 })
```

****

27] Write a MongoDBquery which will select the restaurant Id, name and grades for those restaurants which returns 0 as a remainder after dividing thescore by

```
   db.restaurants.find(
  { "grades.score": { $mod: [5, 0] } },
  { _id: 0, restaurant_id: 1, name: 1, grades: 1 }
)
```

****

28] Write a MongoDB query to find the restaurant name, borough, longitude and attitude and cuisine for those restaurants which contains 'mon' as three letters somewhere in its name

```
  db.restaurants.find(
  { name: /mon/i },
  { _id: 0, name: 1, borough: 1, cuisine: 1, "coordinates.longitude": 1, "coordinates.latitude":
1 }
)
```

****

29] Write a MongoDB query to use sum, avg,min max expression

```
   db.restaurants.aggregate([
  { $unwind: "$grades" },
  {
    $group: {
      _id: null,
      totalScore: { $sum: "$grades.score" },
      avgScore: { $avg: "$grades.score" },
      minScore: { $min: "$grades.score" },
      maxScore: { $max: "$grades.score" }
    }
  }
])
```

****

30] 1).create backup of collections emp_personal_details and emp professional Details

2.Delete some record and then restore it from backup

3.Export the collection in csv and json format