**1) Creating database employee**
**Create collections emp_personal_details with emp_id, emp_name, emp_address, emp_DOB, emp_age, emp_mobilenumber**

```
// Create Database and Collections
use employee

// Create Collection emp_personal_details
db.createCollection("emp_personal_details")


// Insert One Record
db.emp_personal_details.insertMany([
 {
  emp_id: 1,
  emp_name: "Amit Sharma",
  emp_address: "Delhi",
  emp_DOB: ISODate("1962-03-15T00:00:00Z"),
  emp_age: 63,
  emp_mobilenumber: ["9876543210", "9823456789"]
 }
])


// To show the data in collection
db.emp_personal_details.find()
```

///////////output//////////////

```
employee> db.emp_personal_details.find()
[
  {
    _id: ObjectId('68ef55fa5813c60aea748a5f'),
    emp_id: 1,
    emp_name: 'Amit Sharma',
    emp_address: 'Delhi',
    emp_DOB: ISODate('1962-03-15T00:00:00.000Z'),
    emp_age: 63,
    emp_mobilenumber: [ '9876543210', '9823456789' ]
  }
]
employee>
```

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

```
// Use the employee database
use employee
```

```
// Create Collection emp_professional_details
db.createCollection("emp_professional_details")

// Insert One Record
db.emp_professional_details.insertMany([
 {
  emp_id: 1,
  emp_name: "Amit Sharma",
  designation: "Manager",
  salary: 8000,
  incentive: 2000,
  working_hours: 48
 }
])

// To show the data in collection
db.emp_professional_details.find()
////////////////output/////////////////
```

```
[
  {
    _id: ObjectId('68ef507431b707a838748a5f'),
    emp_id: 1,
    emp_name: 'Amit Sharma',
    designation: 'Manager',
    salary: 8000,
    incentive: 2000,
    working_hours: 48
  },
  {
    _id: ObjectId('68ef569b5813c60aea748a60'),
    emp_id: 1,
    emp_name: 'Amit Sharma',
    designation: 'Manager',
    salary: 8000,
    incentive: 2000,
    working_hours: 48
  }
]
```

**3) Insert 10 records in collection emp_personal_details and emp_professional_details**

**2. Show all the employees having designation manager**

**3. Show all the employees having salary 6000**

```
// Use the employee database
use employee

// note not necessary to define collection it will automatically when inserting
```

```
// Insert 10 records into emp_personal_details (mobile number as string)
db.emp_personal_details.insertMany([
  { emp_id: 1, emp_name: "Amit Sharma", emp_address: "Delhi", emp_DOB: "1962-03-15", emp_age: 63,
emp_mobilenumber: "9876543210" },
  { emp_id: 2, emp_name: "Priya Singh", emp_address: "Mumbai", emp_DOB: "1985-07-20", emp_age: 38,
emp_mobilenumber: "9123456789" },
  { emp_id: 3, emp_name: "Rahul Verma", emp_address: "Chennai", emp_DOB: "1978-01-05", emp_age: 45,
emp_mobilenumber: "9988776655" },
  { emp_id: 4, emp_name: "Sunita Patel", emp_address: "Ahmedabad", emp_DOB: "1990-11-12", emp_age: 33,
emp_mobilenumber: "9876501234" },
  { emp_id: 5, emp_name: "Ramesh Kumar", emp_address: "Kolkata", emp_DOB: "1982-05-30", emp_age: 41,
emp_mobilenumber: "9654321876" },
  { emp_id: 6, emp_name: "Neha Joshi", emp_address: "Pune", emp_DOB: "1995-09-15", emp_age: 28,
emp_mobilenumber: "9876123456" },
  { emp_id: 7, emp_name: "Vikas Gupta", emp_address: "Bangalore", emp_DOB: "1975-12-22", emp_age: 47,
emp_mobilenumber: "9345678912" },
  { emp_id: 8, emp_name: "Anita Desai", emp_address: "Hyderabad", emp_DOB: "1988-03-10", emp_age: 35,
emp_mobilenumber: "9988775544" },
  { emp_id: 9, emp_name: "Suresh Reddy", emp_address: "Chennai", emp_DOB: "1960-06-18", emp_age: 63,
emp_mobilenumber: "9874512365" },
  { emp_id: 10, emp_name: "Kavita Singh", emp_address: "Lucknow", emp_DOB: "1992-10-02", emp_age: 31,
emp_mobilenumber: "9123450987" }
])

// Insert 10 records into emp_professional_details
db.emp_professional_details.insertMany([
  { emp_id: 1, emp_name: "Amit Sharma", designation: "Manager", salary: 8000, incentive: 2000, working_hours: 48 },
  { emp_id: 2, emp_name: "Priya Singh", designation: "Software Engineer", salary: 7000, incentive: 1500, working_hours:
42 },
  { emp_id: 3, emp_name: "Rahul Verma", designation: "Accountant", salary: 6000, incentive: 1000, working_hours: 40 },
  { emp_id: 4, emp_name: "Sunita Patel", designation: "Manager", salary: 8500, incentive: 2500, working_hours: 50 },
  { emp_id: 5, emp_name: "Ramesh Kumar", designation: "Clerk", salary: 5000, incentive: 500, working_hours: 38 },
  { emp_id: 6, emp_name: "Neha Joshi", designation: "Software Engineer", salary: 7200, incentive: 1800, working_hours:
44 },
  { emp_id: 7, emp_name: "Vikas Gupta", designation: "Accountant", salary: 6000, incentive: 1000, working_hours: 39 },
  { emp_id: 8, emp_name: "Anita Desai", designation: "Manager", salary: 8200, incentive: 2100, working_hours: 46 },
  { emp_id: 9, emp_name: "Suresh Reddy", designation: "Clerk", salary: 4800, incentive: 400, working_hours: 36 },
  { emp_id: 10, emp_name: "Kavita Singh", designation: "Software Engineer", salary: 7000, incentive: 1600,
working_hours: 43 }
])

// Show all employees having designation "Manager"
db.emp_professional_details.find({ designation: "Manager" })

// Show all employees having salary 6000
db.emp_professional_details.find({ salary: 6000 })
```

//////////////output/////////////

```
employee> db.emp_professional_details.find({ salary: 6000 })

  {
    _id: ObjectId('68ef57d75813c60aea748a6d'),
    emp_id: 3,
    emp_name: 'Rahul Verma',
    designation: 'Accountant',
    salary: 6000,
    incentive: 1000,
    working_hours: 40
  },
  {
    _id: ObjectId('68ef57d75813c60aea748a71'),
    emp_id: 7,
    emp_name: 'Vikas Gupta',
    designation: 'Accountant',
    salary: 6000,
    incentive: 1000,
    working_hours: 39
  }
```

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

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

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

```
// Use employee database
use employee

// 10 records into emp_personal_details
db.emp_personal_details.insertMany([
  { emp_id: 1, emp_name: "Amit Sharma", emp_address: "Delhi", emp_DOB: "1962-03-15", emp_age: 63, emp_mobilenumber: "9876543210" },
  { emp_id: 2, emp_name: "Priya Singh", emp_address: "Mumbai", emp_DOB: "1985-07-20", emp_age: 38, emp_mobilenumber: "9123456789" },
```

```
  { emp_id: 3, emp_name: "Rahul Verma", emp_address: "Chennai", emp_DOB: "1978-01-05", emp_age: 45,
emp_mobilenumber: "9988776655" },
  { emp_id: 4, emp_name: "Sunita Patel", emp_address: "Ahmedabad", emp_DOB: "1990-11-12", emp_age: 33,
emp_mobilenumber: "9876501234" },
  { emp_id: 5, emp_name: "Ramesh Kumar", emp_address: "Kolkata", emp_DOB: "1982-05-30", emp_age: 41,
emp_mobilenumber: "9654321876" },
  { emp_id: 6, emp_name: "Neha Joshi", emp_address: "Pune", emp_DOB: "1995-09-15", emp_age: 28,
emp_mobilenumber: "9876123456" },
  { emp_id: 7, emp_name: "Vikas Gupta", emp_address: "Bangalore", emp_DOB: "1975-12-22", emp_age: 47,
emp_mobilenumber: "9345678912" },
  { emp_id: 8, emp_name: "Anita Desai", emp_address: "Hyderabad", emp_DOB: "1988-03-10", emp_age: 35,
emp_mobilenumber: "9988775544" },
  { emp_id: 9, emp_name: "Suresh Reddy", emp_address: "Chennai", emp_DOB: "1960-06-18", emp_age: 63,
emp_mobilenumber: "9874512365" },
  { emp_id: 10, emp_name: "Kavita Singh", emp_address: "Lucknow", emp_DOB: "1992-10-02", emp_age: 31,
emp_mobilenumber: "9123450987" }
])

//  10 records into emp_professional_details
db.emp_professional_details.insertMany([
  { emp_id: 1, emp_name: "Amit Sharma", designation: "Manager", salary: 8000, incentive: 2000, working_hours: 48 },
  { emp_id: 2, emp_name: "Priya Singh", designation: "Software Engineer", salary: 7000, incentive: 1500, working_hours:
42 },
  { emp_id: 3, emp_name: "Rahul Verma", designation: "Accountant", salary: 6000, incentive: 1000, working_hours: 40 },
  { emp_id: 4, emp_name: "Sunita Patel", designation: "Manager", salary: 8500, incentive: 2500, working_hours: 50 },
  { emp_id: 5, emp_name: "Ramesh Kumar", designation: "Clerk", salary: 5000, incentive: 500, working_hours: 38 },
  { emp_id: 6, emp_name: "Neha Joshi", designation: "Software Engineer", salary: 7200, incentive: 1800, working_hours:
44 },
  { emp_id: 7, emp_name: "Vikas Gupta", designation: "Accountant", salary: 6000, incentive: 1000, working_hours: 39 },
  { emp_id: 8, emp_name: "Anita Desai", designation: "Manager", salary: 8200, incentive: 2100, working_hours: 46 },
  { emp_id: 9, emp_name: "Suresh Reddy", designation: "Clerk", salary: 4800, incentive: 400, working_hours: 36 },
  { emp_id: 10, emp_name: "Kavita Singh", designation: "Software Engineer", salary: 7000, incentive: 1600,
working_hours: 43 }
])

// 1. Update emp_personal_details: add field "status" = "retired" where emp_age > 60
db.emp_personal_details.updateMany(
  { emp_age: { $gt: 60 } },
  { $set: { status: "retired" } }
)

// Show updated emp_personal_details with status field
db.emp_personal_details.find({ emp_age: { $gt: 60 } })

// 2. Update emp_professional_details: set incentive to 5000 for employees working more than 45 hours/week
db.emp_professional_details.updateMany(
```

```
  { working_hours: { $gt: 45 } },
  { $set: { incentive: 5000 } }
)

// Show updated emp_professional_details with incentive 5000
db.emp_professional_details.find({ incentive: 5000 })

// 3. Add 1000 to salary for employees whose designation is "Accountant"
db.emp_professional_details.updateMany(
  { designation: "Accountant" },
  { $inc: { salary: 1000 } }
)

// Show updated emp_professional_details for Accountants
db.emp_professional_details.find({ designation: "Accountant" })
///////////////output///////////////
```

```
}
employee> db.emp_professional_details.find({ designation: "Accountant" })
[
  {
    _id: ObjectId('68ef57d75813c60aea748a6d'),
    emp_id: 3,
    emp_name: 'Rahul Verma',
    designation: 'Accountant',
    salary: 7000,
    incentive: 1000,
    working_hours: 40
  },
  {
    _id: ObjectId('68ef57d75813c60aea748a71'),
    emp_id: 7,
    emp_name: 'Vikas Gupta',
    designation: 'Accountant',
    salary: 7000,
    incentive: 1000,
    working_hours: 39
  },
  {
    _id: ObjectId('68ef58a65813c60aea748a81'),
    emp_id: 3,
    emp_name: 'Rahul Verma',
    designation: 'Accountant',
    salary: 7000,
    incentive: 1000,
    working_hours: 40
  },
  {
    _id: ObjectId('68ef58a65813c60aea748a85'),
    emp_id: 7,
    emp_name: 'Vikas Gupta',
    designation: 'Accountant',
    salary: 7000,
    incentive: 1000,
    working_hours: 39
  }
]
employee>
```

**5) Create index on emp_id in collection emp_professional_details**

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

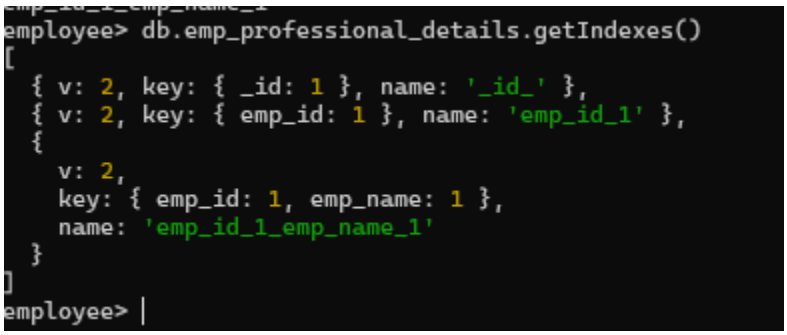
```
// Use employee database
use employee

// Insert sample data into emp_professional_details
db.emp_professional_details.insertMany([
  { emp_id: 1, emp_name: "Amit Sharma", designation: "Manager", salary: 8000, incentive: 2000, working_hours: 48 },
  { emp_id: 2, emp_name: "Priya Singh", designation: "Software Engineer", salary: 7000, incentive: 1500, working_hours: 42 },
  { emp_id: 3, emp_name: "Rahul Verma", designation: "Accountant", salary: 6000, incentive: 1000, working_hours: 40 },
  { emp_id: 4, emp_name: "Sunita Patel", designation: "Manager", salary: 8500, incentive: 2500, working_hours: 50 },
  { emp_id: 5, emp_name: "Ramesh Kumar", designation: "Clerk", salary: 5000, incentive: 500, working_hours: 38 }
])

// 1. Create single index on emp_id
db.emp_professional_details.createIndex({ emp_id: 1 })

// 2. Create compound index on emp_id and emp_name
db.emp_professional_details.createIndex({ emp_id: 1, emp_name: 1 })

// Show indexes
db.emp_professional_details.getIndexes()
```




**6) 1. Find sum of salaries of employees having designation clerk.**
**2. Filter the employees having the designation software engineer and find the minimum salary.**

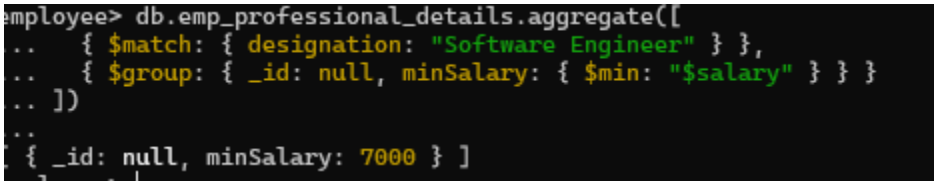
```
// Use employee database
use employee

// Insert sample data into emp_professional_details
db.emp_professional_details.insertMany([
  { emp_id: 1, emp_name: "Amit Sharma", designation: "Manager", salary: 8000, incentive: 2000, working_hours: 48 },
  { emp_id: 2, emp_name: "Priya Singh", designation: "Software Engineer", salary: 7000, incentive: 1500, working_hours: 42 },
  { emp_id: 3, emp_name: "Rahul Verma", designation: "Accountant", salary: 6000, incentive: 1000, working_hours: 40 },
  { emp_id: 4, emp_name: "Sunita Patel", designation: "Manager", salary: 8500, incentive: 2500, working_hours: 50 },
  { emp_id: 5, emp_name: "Ramesh Kumar", designation: "Clerk", salary: 5000, incentive: 500, working_hours: 38 },
```

```
  { emp_id: 6, emp_name: "Neha Joshi", designation: "Software Engineer", salary: 7200, incentive: 1800, working_hours: 44 },
  { emp_id: 7, emp_name: "Vikas Gupta", designation: "Accountant", salary: 6000, incentive: 1000, working_hours: 39 },
  { emp_id: 8, emp_name: "Anita Desai", designation: "Manager", salary: 8200, incentive: 2100, working_hours: 46 },
  { emp_id: 9, emp_name: "Suresh Reddy", designation: "Clerk", salary: 4800, incentive: 400, working_hours: 36 },
  { emp_id: 10, emp_name: "Kavita Singh", designation: "Software Engineer", salary: 7000, incentive: 1600, working_hours: 43 }
])


// 1. Find all employees with designation "Clerk"
db.emp_professional_details.find({ designation: "Clerk" })

// 2. Minimum salary of employees with designation "Software Engineer"
db.emp_professional_details.aggregate([
  { $match: { designation: "Software Engineer" } },
  { $group: { _id: null, minSalary: { $min: "$salary" } } }
])
//////////////output////////////
```




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

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

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

```
// To create database
use employee

// Insert sample data into emp_personal_details
db.emp_personal_details.insertMany([
  { emp_id: 1, emp_name: "Amit Sharma", emp_address: "Delhi", emp_DOB: ISODate("1962-03-15T00:00:00Z"), emp_age: 63, emp_mobilenumber: ["9876543210", "9823456789"] },
  { emp_id: 2, emp_name: "Priya Singh", emp_address: "Mumbai", emp_DOB: ISODate("1985-07-20T00:00:00Z"), emp_age: 38, emp_mobilenumber: ["9123456780"] },
  { emp_id: 3, emp_name: "Rahul Verma", emp_address: "Bangalore", emp_DOB: ISODate("1990-11-10T00:00:00Z"), emp_age: 33, emp_mobilenumber: ["9988776655", "9900112233"] },
  { emp_id: 4, emp_name: "Sunita Patel", emp_address: "Ahmedabad", emp_DOB: ISODate("1975-01-25T00:00:00Z"), emp_age: 48, emp_mobilenumber: ["9876001122"] },
```

{ emp_id: 5, emp_name: "Ramesh Kumar", emp_address: "Chennai", emp_DOB: ISODate("1982-09-05T00:00:00Z"), emp_age: 41, emp_mobilenumber: ["9123456799"] }
])

// 1. Use unwind command to show employees whose mobile number
db.emp_personal_details.aggregate([
{ $unwind: "$emp_mobilenumber" }
])

// 2. Use skip command to skip first 3 records and display the rest from emp_personal_details
db.emp_personal_details.find().skip(3)

// 3. Use limit command to show only first 4 records from emp_personal_details
db.emp_personal_details.find().limit(4)
//////////////////output//////////////

```
]
employee> db.emp_personal_details.find().limit(4)
[
  {
    _id: ObjectId('68ef55fa5813c60aea748a5f'),
    emp_id: 1,
    emp_name: 'Amit Sharma',
    emp_address: 'Delhi',
    emp_DOB: ISODate('1962-03-15T00:00:00.000Z'),
    emp_age: 63,
    emp_mobilenumber: [ '9876543210', '9823456789' ],
    status: 'retired'
  },
  {
    _id: ObjectId('68ef57c25813c60aea748a61'),
    emp_id: 1,
    emp_name: 'Amit Sharma',
    emp_address: 'Delhi',
    emp_DOB: '1962-03-15',
    emp_age: 63,
    emp_mobilenumber: '9876543210',
    status: 'retired'
  },
  {
    _id: ObjectId('68ef57c25813c60aea748a62'),
    emp_id: 2,
    emp_name: 'Priya Singh',
    emp_address: 'Mumbai',
    emp_DOB: '1985-07-20',
    emp_age: 38,
    emp_mobilenumber: '9123456789'
  },
  {
    _id: ObjectId('68ef57c25813c60aea748a63'),
    emp_id: 3,
    emp_name: 'Rahul Verma',
    emp_address: 'Chennai',
    emp_DOB: '1978-01-05',
    emp_age: 45,
    emp_mobilenumber: '9988776655'
  }
]
```

**9) Create a MongoDB collection named restaurants to store the following information about restaurants:**
**Building number**
**Street name**
**Zip code**
**Coordinates (longitude and latitude)**
**Borough**
**Cuisine type**
**Grades (each grade includes: date, grade (A/B/C), and score)**
**// Use your database (e.g., restaurantDB)**
**use restaurantDB**

// Find and dis

```
// Insert sample data into restaurants collection
db.restaurants.insertMany([
  {
    building_number: "123",
    street_name: "MG Road",
    zip_code: "400001",
    coordinates: { longitude: 73.8567, latitude: 18.5204 },
    borough: "Mumbai",
    cuisine_type: "Indian",
    grades: [
      { date: ISODate("2024-01-15"), grade: "A", score: 95 },
      { date: ISODate("2023-06-10"), grade: "B", score: 88 }
    ]
  },
  {
    building_number: "45",
    street_name: "Brigade Road",
    zip_code: "560001",
    coordinates: { longitude: 77.5946, latitude: 12.9716 },
    borough: "Bangalore",
    cuisine_type: "Continental",
    grades: [
      { date: ISODate("2024-02-20"), grade: "A", score: 92 },
      { date: ISODate("2023-08-18"), grade: "A", score: 90 }
    ]
  },
  {
    building_number: "67",
    street_name: "Connaught Place",
    zip_code: "110001",
    coordinates: { longitude: 77.2167, latitude: 28.6667 },
    borough: "Delhi",
    cuisine_type: "Chinese",
    grades: [
      { date: ISODate("2023-12-05"), grade: "B", score: 85 },
      { date: ISODate("2023-05-22"), grade: "C", score: 75 }
    ]
  }
])
play all restaurants
db.restaurants.find()
//////////////output///////////////
```

```
restaurantDB> db.restaurants.find()
[
  {
    _id: ObjectId('68ef4d8f406119d2c9748a5f'),
    restaurant_id: 1,
    name: 'Monalisa Cafe',
    borough: 'Manhattan',
    cuisine: 'Italian',
    address: { coord: [ -73.9557413, 40.7720266 ] }
  },
  {
    _id: ObjectId('68ef4d8f406119d2c9748a60'),
    restaurant_id: 2,
    name: 'Harmony Diner',
    borough: 'Queens',
    cuisine: 'American',
    address: { coord: [ -73.9874, 40.7359 ] }
  },
  {
    _id: ObjectId('68ef4d8f406119d2c9748a61'),
    restaurant_id: 3,
    name: 'Sunset Grill',
    borough: 'Brooklyn',
    cuisine: 'Mexican',
    address: { coord: [ -73.9934, 40.7218 ] }
  },
  {
    _id: ObjectId('68ef60025813c60aea748a8e'),
    building_number: '123',
    street_name: 'MG Road',
    zip_code: '400001',
    coordinates: { longitude: 73.8567, latitude: 18.5204 },
    borough: 'Mumbai',
    cuisine_type: 'Indian',
    grades: [
      {
        date: ISODate('2024-01-15T00:00:00.000Z'),
        grade: 'A',
        score: 95
      },
      {
        date: ISODate('2023-06-10T00:00:00.000Z'),
        grade: 'B',
        score: 88
      }
```

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

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

```
// Use database
use restaurantDB

// Insert sample data into restaurants collection
db.restaurants.insertMany([
 {
  restaurant_id: 101,
  name: "Spice Villa",
  borough: "Mumbai",
  cuisine: "Indian",
  building_number: "123",
  street_name: "MG Road",
  zip_code: "400001",
  coordinates: { longitude: 73.8567, latitude: 18.5204 },
  grades: [
   { date: ISODate("2024-01-15"), grade: "A", score: 95 },
   { date: ISODate("2023-06-10"), grade: "B", score: 88 }
  ]
 },
 {
  restaurant_id: 102,
  name: "Continental Delight",
  borough: "Bangalore",
```

```
    cuisine: "Continental",
    building_number: "45",
    street_name: "Brigade Road",
    zip_code: "560001",
    coordinates: { longitude: 77.5946, latitude: 12.9716 },
    grades: [
      { date: ISODate("2024-02-20"), grade: "A", score: 92 },
      { date: ISODate("2023-08-18"), grade: "A", score: 90 }
    ]
  }
])

// 1. Display all documents in the restaurants collection
db.restaurants.find()

// 2. Display only restaurant_id, name, borough, and cuisine fields for all documents
db.restaurants.find(
  {},
  { restaurant_id: 1, name: 1, borough: 1, cuisine: 1, _id: 0 }
)
///////////////output/////////////////
```

```
..    {},
..    { restaurant_id: 1, name: 1, borough: 1, cuisine: 1, _id: 0 }
.. )
..

 {
   restaurant_id: 1,
   name: 'Monalisa Cafe',
   borough: 'Manhattan',
   cuisine: 'Italian'
 },
 {
   restaurant_id: 2,
   name: 'Harmony Diner',
   borough: 'Queens',
   cuisine: 'American'
 },
 {
   restaurant_id: 3,
   name: 'Sunset Grill',
   borough: 'Brooklyn',
   cuisine: 'Mexican'
 },
 { borough: 'Mumbai' },
 { borough: 'Bangalore' },
 { borough: 'Delhi' },
 {
   restaurant_id: 101,
   name: 'Spice Villa',
   borough: 'Mumbai',
   cuisine: 'Indian'
 },
 {
   restaurant_id: 102,
   name: 'Continental Delight',
   borough: 'Bangalore',
   cuisine: 'Continental'
 }
```

**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**

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

**Bronx**

```
// 1. Use the database
use restaurantDB

// 2. Insert sample data
db.restaurants.insertMany([
  {
    restaurant_id: 1,
    name: "Spice Villa",
    borough: "Mumbai",
    cuisine: "Indian",
    building_number: "123",
    street_name: "MG Road",
    zip_code: "400001",
    coordinates: { longitude: 73.8567, latitude: 18.5204 },
    grades: [
      { date: ISODate("2024-01-15"), grade: "A", score: 95 }
    ]
  },
  {
    restaurant_id: 2,
    name: "Bronx Diner",
    borough: "Bronx",
    cuisine: "American",
    building_number: "45",
    street_name: "Fordham Road",
    zip_code: "10458",
    coordinates: { longitude: -73.9030, latitude: 40.8620 },
    grades: [
      { date: ISODate("2023-11-12"), grade: "B", score: 85 }
    ]
  }
])

// 3. Display restaurant_id, name, borough, and cuisine (without _id)
db.restaurants.find(
  {},
  { restaurant_id: 1, name: 1, borough: 1, cuisine: 1, _id: 0 }
)

// 4. Display all restaurants in the borough "Bronx"
db.restaurants.find({ borough: "Bronx" })
/////////////////output/////////////////
```

```
restaurantDB> db.restaurants.find({ borough: "Bronx" })
[
  {
    _id: ObjectId('68ef64085813c60aea748a94'),
    restaurant_id: 2,
    name: 'Bronx Diner',
    borough: 'Bronx',
    cuisine: 'American',
    building_number: '45',
    street_name: 'Fordham Road',
    zip_code: '10458',
    coordinates: { longitude: -73.903, latitude: 40.862 },
    grades: [
      {
        date: ISODate('2023-11-12T00:00:00.000Z'),
        grade: 'B',
        score: 85
      }
    ]
  }
]
```

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

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

```
// 1. Use the database
use restaurantDB

// 2. Insert 10 sample restaurants in the borough "Bronx"
db.restaurants.insertMany([
  { restaurant_id: 1, name: "Bronx Eatery 1", borough: "Bronx", cuisine: "Indian" },
  { restaurant_id: 2, name: "Bronx Eatery 2", borough: "Bronx", cuisine: "Italian" },
  { restaurant_id: 3, name: "Bronx Eatery 3", borough: "Bronx", cuisine: "Mexican" },
  { restaurant_id: 4, name: "Bronx Eatery 4", borough: "Bronx", cuisine: "Chinese" },
  { restaurant_id: 5, name: "Bronx Eatery 5", borough: "Bronx", cuisine: "Thai" },
  { restaurant_id: 6, name: "Bronx Eatery 6", borough: "Bronx", cuisine: "Japanese" },
  { restaurant_id: 7, name: "Bronx Eatery 7", borough: "Bronx", cuisine: "Korean" },
  { restaurant_id: 8, name: "Bronx Eatery 8", borough: "Bronx", cuisine: "Greek" },
  { restaurant_id: 9, name: "Bronx Eatery 9", borough: "Bronx", cuisine: "French" },
  { restaurant_id: 10, name: "Bronx Eatery 10", borough: "Bronx", cuisine: "American" }
])

// 3. Display the first 5 restaurants in the borough "Bronx"
db.restaurants.find({ borough: "Bronx" }).limit(5)

// 4. Display the next 5 restaurants in the borough "Bronx" (skip first 5)
db.restaurants.find({ borough: "Bronx" }).skip(5).limit(5)
///////////////output//////////////
```

```
]
restaurantDB> db.restaurants.find({ borough: "Bronx" }).skip(5).limit(5)
[
  {
    _id: ObjectId('68ef655c5813c60aea748a99'),
    restaurant_id: 5,
    name: 'Bronx Eatery 5',
    borough: 'Bronx',
    cuisine: 'Thai'
  },
  {
    _id: ObjectId('68ef655c5813c60aea748a9a'),
    restaurant_id: 6,
    name: 'Bronx Eatery 6',
    borough: 'Bronx',
    cuisine: 'Japanese'
  },
  {
    _id: ObjectId('68ef655c5813c60aea748a9b'),
    restaurant_id: 7,
    name: 'Bronx Eatery 7',
    borough: 'Bronx',
    cuisine: 'Korean'
  },
  {
    _id: ObjectId('68ef655c5813c60aea748a9c'),
    restaurant_id: 8,
    name: 'Bronx Eatery 8',
    borough: 'Bronx',
    cuisine: 'Greek'
  },
  {
    _id: ObjectId('68ef655c5813c60aea748a9d'),
    restaurant_id: 9,
    name: 'Bronx Eatery 9',
    borough: 'Bronx',
    cuisine: 'French'
  }
]
```

**13) 1.Write a MongoDB query to find the restaurants who achieved a score more than 90**
**2.Write a MongoDB query to find the restaurantsthat achieved a score, more than 80 but less than 100**

```
// 1. Use the database
use restaurantDB

// 2. Insert 3 sample records into restaurants collection
db.restaurants.insertMany([
  { restaurant_id: 1, name: "Spice Villa", borough: "Mumbai", cuisine: "Indian", grades: [ { date: "2024-01-15", grade: "A", score: 92 } ] },
  { restaurant_id: 2, name: "Bronx Diner", borough: "Bronx", cuisine: "American", grades: [ { date: "2024-03-12", grade: "B", score: 85 } ] },
  { restaurant_id: 3, name: "Green Garden", borough: "Delhi", cuisine: "Chinese", grades: [ { date: "2024-05-10", grade: "A", score: 78 } ] }
])

// 3. Find restaurants with score greater than 90
db.restaurants.find({ "grades.score": { $gt: 90 } })

// 4. Find restaurants with score more than 80 but less than 100
db.restaurants.find({ "grades.score": { $gt: 80, $lt: 100 } })
///////////////output/////////////
```

```
]
restaurantDB> db.restaurants.find({ "grades.score": { $gt: 80, $lt: 100 } })
[
  {
    _id: ObjectId('68ef60025813c60aea748a8e'),
    building_number: '123',
    street_name: 'MG Road',
    zip_code: '400001',
    coordinates: { longitude: 73.8567, latitude: 18.5204 },
    borough: 'Mumbai',
    cuisine_type: 'Indian',
    grades: [
      {
        date: ISODate('2024-01-15T00:00:00.000Z'),
        grade: 'A',
        score: 95
      },
      {
        date: ISODate('2023-06-10T00:00:00.000Z'),
        grade: 'B',
        score: 88
      }
    ]
  },
  {
    _id: ObjectId('68ef60025813c60aea748a8f'),
    building_number: '45',
    street_name: 'Brigade Road',
    zip_code: '560001',
    coordinates: { longitude: 77.5946, latitude: 12.9716 },
    borough: 'Bangalore',
    cuisine_type: 'Continental',
    grades: [
      {
        date: ISODate('2024-02-20T00:00:00.000Z'),
        grade: 'A',
        score: 92
      },
      {
        date: ISODate('2023-08-18T00:00:00.000Z'),
        grade: 'A',
        score: 90
      }
    ]
  },
```

**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**

```
// 1. Use the database
use restaurantDB

// 2. Insert 3 sample records into restaurants collection
db.restaurants.insertMany([
 { restaurant_id: 1, name: "Spice Villa", borough: "Mumbai", cuisine: "Indian", grades: [ { date: "2024-01-15", grade: "A", score: 92 } ] },
 { restaurant_id: 2, name: "Bronx Diner", borough: "Bronx", cuisine: "American", grades: [ { date: "2024-03-12", grade: "B", score: 85 } ] },
 { restaurant_id: 3, name: "Green Garden", borough: "Delhi", cuisine: "Chinese", grades: [ { date: "2024-05-10", grade: "A", score: 78 } ] }
])

// 3. Find restaurants with score greater than 90
```

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

// 4. Find restaurants with score more than 80 but less than 100
db.restaurants.find({ "grades.score": { $gt: 80, $lt: 100 } })
////////////////////output///////////////

```
restaurantDB> db.restaurants.find({ "grades.score": { $gt: 80, $lt: 100 } })
[
  {
    _id: ObjectId('68ef60025813c60aea748a8e'),
    building_number: '123',
    street_name: 'MG Road',
    zip_code: '400001',
    coordinates: { longitude: 73.8567, latitude: 18.5204 },
    borough: 'Mumbai',
    cuisine_type: 'Indian',
    grades: [
      {
        date: ISODate('2024-01-15T00:00:00.000Z'),
        grade: 'A',
        score: 95
      },
      {
        date: ISODate('2023-06-10T00:00:00.000Z'),
        grade: 'B',
        score: 88
      }
    ]
  },
  {
    _id: ObjectId('68ef60025813c60aea748a8f'),
    building_number: '45',
    street_name: 'Brigade Road',
    zip_code: '560001',
    coordinates: { longitude: 77.5946, latitude: 12.9716 },
    borough: 'Bangalore',
    cuisine_type: 'Continental',
    grades: [
      {
        date: ISODate('2024-02-20T00:00:00.000Z'),
        grade: 'A',
        score: 92
      },
      {
        date: ISODate('2023-08-18T00:00:00.000Z'),
        grade: 'A',
        score: 90
      }
    ]
  },
```

**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**

// Use database
use restaurantDB

// Insert sample records
db.restaurants.insertMany([
 { restaurant_id: 1, name: "Wild Bites", borough: "Mumbai", cuisine: "Indian", grades: [ { date: "2024-01-10", grade: "A", score: 90 } ] },
 { restaurant_id: 2, name: "Wilson Grill", borough: "Delhi", cuisine: "Continental", grades: [ { date: "2024-02-15", grade: "B", score: 85 } ] },
 { restaurant_id: 3, name: "Bella Casa", borough: "Pune", cuisine: "Italian", grades: [ { date: "2024-03-20", grade: "A", score: 88 } ] },
 { restaurant_id: 4, name: "Willow Tree", borough: "Bangalore", cuisine: "Chinese", grades: [ { date: "2024-04-25", grade: "A", score: 92 } ] }
])

// Query to find restaurants with names starting with "Wil"
db.restaurants.find(
 { name: { $regex: /^Wil/ } },
 { restaurant_id: 1, name: 1, borough: 1, cuisine: 1, _id: 0 }
)
//////////////////output//////////////////

```
restaurantDB> db.restaurants.find(
...   { name: { $regex: /^Wil/ } },
...   { restaurant_id: 1, name: 1, borough: 1, cuisine: 1, _id: 0 }
... )
...
[
  {
    restaurant_id: 1,
    name: 'Wild Bites',
    borough: 'Mumbai',
    cuisine: 'Indian'
  },
  {
    restaurant_id: 2,
    name: 'Wilson Grill',
    borough: 'Delhi',
    cuisine: 'Continental'
  },
  {
    restaurant_id: 4,
    name: 'Willow Tree',
    borough: 'Bangalore',
    cuisine: 'Chinese'
  }
]
```

**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.**

```
// Use database
use restaurantDB

// Insert sample records
db.restaurants.insertMany([
  { restaurant_id: 1, name: "Spices", borough: "Mumbai", cuisine: "Indian", grades: [ { date: "2024-01-10", grade: "A", score: 90 } ] },
  { restaurant_id: 2, name: "Delices", borough: "Delhi", cuisine: "French", grades: [ { date: "2024-02-15", grade: "B", score: 85 } ] },
  { restaurant_id: 3, name: "Flavors", borough: "Pune", cuisine: "Italian", grades: [ { date: "2024-03-20", grade: "A", score: 88 } ] },
  { restaurant_id: 4, name: "Choices", borough: "Bangalore", cuisine: "Chinese", grades: [ { date: "2024-04-25", grade: "A", score: 92 } ] }
])

// Query to find restaurants with names ending with "ces"
db.restaurants.find(
  { name: { $regex: /ces$/ } },
  { restaurant_id: 1, name: 1, borough: 1, cuisine: 1, _id: 0 }
)
//////////////////output//////////////////
```

```
restaurantDB> db.restaurants.find(
...   { name: { $regex: /ces$/ } },
...   { restaurant_id: 1, name: 1, borough: 1, cuisine: 1, _id: 0 }
... )
[
  {
    restaurant_id: 1,
    name: 'Spices',
    borough: 'Mumbai',
    cuisine: 'Indian'
  },
  {
    restaurant_id: 2,
    name: 'Delices',
    borough: 'Delhi',
    cuisine: 'French'
  },
  {
    restaurant_id: 4,
    name: 'Choices',
    borough: 'Bangalore',
    cuisine: 'Chinese'
  }
]
```

**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**

```
// Use database
use restaurantDB

// Insert sample records
db.restaurants.insertMany([
  { restaurant_id: 1, name: "Regal Dine", borough: "Mumbai", cuisine: "Indian", grades: [ { date: "2024-01-10", grade: "A", score: 90 } ] },
  { restaurant_id: 2, name: "The Great Region", borough: "Delhi", cuisine: "Continental", grades: [ { date: "2024-02-15", grade: "B", score: 85 } ] },
  { restaurant_id: 3, name: "Delight", borough: "Pune", cuisine: "Italian", grades: [ { date: "2024-03-20", grade: "A", score: 88 } ] },
  { restaurant_id: 4, name: "Oregano", borough: "Bangalore", cuisine: "Mexican", grades: [ { date: "2024-04-25", grade: "A", score: 92 } ] }
])

// Query to find restaurants with 'Reg' in the name
db.restaurants.find(
  { name: { $regex: /Reg/ } },
  { restaurant_id: 1, name: 1, borough: 1, cuisine: 1, _id: 0 }
)
/////////////////output//////////////
```

```
restaurantDB> db.restaurants.find(
...   { name: { $regex: /Reg/ } },
...   { restaurant_id: 1, name: 1, borough: 1, cuisine: 1, _id: 0 }
... )
...
[
  {
    restaurant_id: 1,
    name: 'Regal Dine',
    borough: 'Mumbai',
    cuisine: 'Indian'
  },
  {
    restaurant_id: 2,
    name: 'The Great Region',
    borough: 'Delhi',
    cuisine: 'Continental'
  }
]
```

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

```
// Use database
use restaurantDB

// Insert sample records
db.restaurants.insertMany([
  { restaurant_id: 1, name: "Bronx Diner", borough: "Bronx", cuisine: "American", grades: [{ date: "2024-01-10", grade: "A", score: 90 }] },
  { restaurant_id: 2, name: "Chinese Delight", borough: "Bronx", cuisine: "Chinese", grades: [{ date: "2024-02-15", grade: "B", score: 85 }] },
  { restaurant_id: 3, name: "Italian Bistro", borough: "Bronx", cuisine: "Italian", grades: [{ date: "2024-03-20", grade: "A", score: 88 }] },
  { restaurant_id: 4, name: "Queens Grill", borough: "Queens", cuisine: "American", grades: [{ date: "2024-04-25", grade: "A", score: 92 }] }
])

// Find restaurants in Bronx serving American or Chinese cuisine (all fields shown)
db.restaurants.find(
  {
    borough: "Bronx",
    cuisine: { $in: ["American", "Chinese"] }
  }
)
//////////////////output///////////////
```

```
[
  {
    _id: ObjectId('68ef64085813c60aea748a94'),
    restaurant_id: 2,
    name: 'Bronx Diner',
    borough: 'Bronx',
    cuisine: 'American',
    building_number: '45',
    street_name: 'Fordham Road',
    zip_code: '10458',
    coordinates: { longitude: -73.903, latitude: 40.862 },
    grades: [
      {
        date: ISODate('2023-11-12T00:00:00.000Z'),
        grade: 'B',
        score: 85
      }
    ]
  },
  {
    _id: ObjectId('68ef655c5813c60aea748a98'),
    restaurant_id: 4,
    name: 'Bronx Eatery 4',
    borough: 'Bronx',
    cuisine: 'Chinese'
  },
  {
    _id: ObjectId('68ef655c5813c60aea748a9e'),
    restaurant_id: 10,
    name: 'Bronx Eatery 10',
    borough: 'Bronx',
    cuisine: 'American'
  },
  {
    _id: ObjectId('68ef67615813c60aea748aa0'),
    restaurant_id: 2,
    name: 'Bronx Diner',
```

**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.**

```
// Use database
use restaurantDB

// Insert sample records
db.restaurants.insertMany([
  { restaurant_id: 1, name: "Bronx Diner", borough: "Bronx", cuisine: "American" },
  { restaurant_id: 2, name: "Queens Grill", borough: "Queens", cuisine: "Chinese" },
  { restaurant_id: 3, name: "Brooklyn Pizza", borough: "Brooklyn", cuisine: "Italian" },
  { restaurant_id: 4, name: "Staten Island Eatery", borough: "Staten Island", cuisine: "Mexican" },
  { restaurant_id: 5, name: "Manhattan Deli", borough: "Manhattan", cuisine: "American" }
])

// Query: Find restaurants in Staten Island, Queens, Bronx or Brooklyn with specific fields
db.restaurants.find(
  { borough: { $in: ["Staten Island", "Queens", "Bronx", "Brooklyn"] } },
  { restaurant_id: 1, name: 1, borough: 1, cuisine: 1, _id: 0 }
)
//////////////////output//////////////////
```

```
restaurantDB> db.restaurants.find(
...   { borough: { $in: ["Staten Island", "Queens", "Bronx", "Brooklyn"] } },
...   { restaurant_id: 1, name: 1, borough: 1, cuisine: 1, _id: 0 }
... )
...
[
  {
    restaurant_id: 2,
    name: 'Harmony Diner',
    borough: 'Queens',
    cuisine: 'American'
  },
  {
    restaurant_id: 3,
    name: 'Sunset Grill',
    borough: 'Brooklyn',
    cuisine: 'Mexican'
  },
  {
    restaurant_id: 2,
    name: 'Bronx Diner',
    borough: 'Bronx',
    cuisine: 'American'
  },
  {
    restaurant_id: 1,
    name: 'Bronx Eatery 1',
    borough: 'Bronx',
    cuisine: 'Indian'
  },
```

**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.**

```
// Use database
use restaurantDB

// Insert sample records
db.restaurants.insertMany([
  { restaurant_id: 1, name: "Bronx Diner", borough: "Bronx", cuisine: "American" },
  { restaurant_id: 2, name: "Queens Grill", borough: "Queens", cuisine: "Chinese" },
  { restaurant_id: 3, name: "Brooklyn Pizza", borough: "Brooklyn", cuisine: "Italian" },
  { restaurant_id: 4, name: "Staten Island Eatery", borough: "Staten Island", cuisine: "Mexican" },
  { restaurant_id: 5, name: "Manhattan Deli", borough: "Manhattan", cuisine: "American" },
  { restaurant_id: 6, name: "Harlem Cafe", borough: "Harlem", cuisine: "Caribbean" }
])

// Query: Find restaurants NOT in Staten Island, Queens, Bronx, or Brooklyn
db.restaurants.find(
  { borough: { $nin: ["Staten Island", "Queens", "Bronx", "Brooklyn"] } },
  { restaurant_id: 1, name: 1, borough: 1, cuisine: 1, _id: 0 }
)
//////////////////output//////////////////
```

```
restaurantDB> db.restaurants.find(
...   { borough: { $nin: ["Staten Island", "Queens", "Bronx", "Brooklyn"] } },
...   { restaurant_id: 1, name: 1, borough: 1, cuisine: 1, _id: 0 }
... )
...
[
  {
    restaurant_id: 1,
    name: 'Monalisa Cafe',
    borough: 'Manhattan',
    cuisine: 'Italian'
  },
  { borough: 'Mumbai' },
  { borough: 'Bangalore' },
  { borough: 'Delhi' },
  {
    restaurant_id: 101,
    name: 'Spice Villa',
    borough: 'Mumbai',
    cuisine: 'Indian'
  },
  {
    restaurant_id: 102,
    name: 'Continental Delight',
    borough: 'Bangalore',
    cuisine: 'Continental'
  },
  {
    restaurant_id: 1,
    name: 'Spice Villa',
    borough: 'Mumbai',
    cuisine: 'Indian'
  },
  {
    restaurant_id: 1,
    name: 'Spice Villa',
    borough: 'Mumbai',
    cuisine: 'Indian'
  },
```

**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**

```
// Use database
use restaurantDB

// Insert sample data
db.restaurants.insertMany([
 {
  restaurant_id: 1,
  name: "Sunny Side Cafe",
  borough: "Manhattan",
  cuisine: "American",
  grades: [
   { date: new Date("2023-01-10"), grade: "A", score: 8 },
   { date: new Date("2022-11-15"), grade: "B", score: 12 }
  ]
```

```
  },
  {
    restaurant_id: 2,
    name: "Ocean Breeze",
    borough: "Brooklyn",
    cuisine: "Seafood",
    grades: [
      { date: new Date("2023-03-05"), grade: "A", score: 15 },
      { date: new Date("2022-12-22"), grade: "A", score: 9 }
    ]
  },
  {
    restaurant_id: 3,
    name: "Green Garden",
    borough: "Queens",
    cuisine: "Vegetarian",
    grades: [
      { date: new Date("2023-02-18"), grade: "C", score: 11 },
      { date: new Date("2022-10-10"), grade: "B", score: 7 }
    ]
  }
])

// Query: Find restaurants with any score not more than 10
db.restaurants.find(
  { "grades.score": { $lte: 10 } },
  { restaurant_id: 1, name: 1, borough: 1, cuisine: 1, _id: 0 }
)
///////////////output/////////////
```

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

 {
   restaurant_id: 1,
   name: 'Sunny Side Cafe',
   borough: 'Manhattan',
   cuisine: 'American'
 },
 {
   restaurant_id: 2,
   name: 'Ocean Breeze',
   borough: 'Brooklyn',
   cuisine: 'Seafood'
 },
 {
   restaurant_id: 3,
   name: 'Green Garden',
   borough: 'Queens',
   cuisine: 'Vegetarian'
 },
 {
   restaurant_id: 1,
   name: 'Sunny Side Cafe',
   borough: 'Manhattan',
   cuisine: 'American'
 },
```

**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'**

```
// Use database
use restaurantDB

// Insert sample data
db.restaurants.insertMany([
  { restaurant_id: 1, name: "Wilson's Diner", borough: "Manhattan", cuisine: "American" },
  { restaurant_id: 2, name: "Green Garden", borough: "Queens", cuisine: "Vegetarian" },
  { restaurant_id: 3, name: "Spice Hub", borough: "Brooklyn", cuisine: "Indian" },
  { restaurant_id: 4, name: "China Palace", borough: "Bronx", cuisine: "Chinese" },
  { restaurant_id: 5, name: "Wild Orchid", borough: "Manhattan", cuisine: "Thai" }
])

// 1. Find restaurants which do NOT prepare 'American' or 'Chinese' cuisine:

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

// 2. Find restaurants whose name starts with 'Wil':

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

////////////////////output/////////////////

```
Type "it" for more
restaurantDB> db.restaurants.find(
...   { name: /^Wil/ },
...   { restaurant_id: 1, name: 1, borough: 1, cuisine: 1, _id: 0 }
... )
...
[
  {
    restaurant_id: 1,
    name: 'Wild Bites',
    borough: 'Mumbai',
    cuisine: 'Indian'
  },
  {
    restaurant_id: 2,
    name: 'Wilson Grill',
    borough: 'Delhi',
    cuisine: 'Continental'
  },
  {
    restaurant_id: 4,
    name: 'Willow Tree',
    borough: 'Bangalore',
    cuisine: 'Chinese'
  },
  {
    restaurant_id: 1,
    name: "Wilson's Diner",
    borough: 'Manhattan',
    cuisine: 'American'
  },
```

**23) Write a MongoDB query to arrange the name of the restaurants in descending**

**along with all the columns**

```
// Use database
use restaurantDB

// Insert sample data
db.restaurants.insertMany([
  { restaurant_id: 1, name: "Wilson's Diner", borough: "Manhattan", cuisine: "American" },
  { restaurant_id: 2, name: "Green Garden", borough: "Queens", cuisine: "Vegetarian" },
  { restaurant_id: 3, name: "Spice Hub", borough: "Brooklyn", cuisine: "Indian" },
  { restaurant_id: 4, name: "China Palace", borough: "Bronx", cuisine: "Chinese" },
  { restaurant_id: 5, name: "Wild Orchid", borough: "Manhattan", cuisine: "Thai" }
])

// Query to sort restaurants by name in descending order with all columns
db.restaurants.find().sort({ name: -1 })
```

///////////////output///////////////

```
}
restaurantDB> db.restaurants.find().sort({ name: -1 })
[
  {
    _id: ObjectId('68ef759e5813c60aea748ac8'),
    restaurant_id: 1,
    name: "Wilson's Diner",
    borough: 'Manhattan',
    cuisine: 'American'
  },
  {
    _id: ObjectId('68ef721e5813c60aea748ac3'),
    restaurant_id: 1,
    name: "Wilson's Diner",
    borough: 'Manhattan',
    cuisine: 'American'
  },
```

**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**

```
// Use database
use restaurantDB

// Insert sample data
db.restaurants.insertMany([
  { restaurant_id: 1, name: "Wilson's Diner", borough: "Manhattan", cuisine: "American" },
  { restaurant_id: 2, name: "Green Garden", borough: "Queens", cuisine: "Vegetarian" },
  { restaurant_id: 3, name: "Spice Hub", borough: "Brooklyn", cuisine: "Indian" },
  { restaurant_id: 4, name: "China Palace", borough: "Bronx", cuisine: "Chinese" },
  { restaurant_id: 5, name: "Wild Orchid", borough: "Manhattan", cuisine: "Thai" }
])

// Query to sort by cuisine ascending and borough descending
db.restaurants.find().sort({ cuisine: 1, borough: -1 })
```

//////////////output//////////////

```
}
restaurantDB> db.restaurants.find().sort({ cuisine: 1, borough: -1 })
[
  {
    _id: ObjectId('68ef60025813c60aea748a8e'),
    building_number: '123',
    street_name: 'MG Road',
    zip_code: '400001',
    coordinates: { longitude: 73.8567, latitude: 18.5204 },
    borough: 'Mumbai',
    cuisine_type: 'Indian',
    grades: [
      {
        date: ISODate('2024-01-15T00:00:00.000Z'),
        grade: 'A',
        score: 95
      },
      {
        date: ISODate('2023-06-10T00:00:00.000Z'),
        grade: 'B',
        score: 88
      }
    ]
  },
```

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

// Use database
use restaurantDB

// Insert sample data
db.restaurants.insertMany([
 { restaurant_id: 1, name: "Wilson's Diner", borough: "Manhattan", cuisine: "American" },
 { restaurant_id: 2, name: "Green Garden", borough: "Queens", cuisine: "Vegetarian" },
 { restaurant_id: 3, name: "Spice Hub", borough: "Brooklyn", cuisine: "Indian" },
 { restaurant_id: 4, name: "China Palace", borough: "Bronx", cuisine: "Chinese" },
 { restaurant_id: 5, name: "Wild Orchid", borough: "Manhattan", cuisine: "Thai" }
])

// Query to sort by cuisine ascending and borough descending
db.restaurants.find().sort({ cuisine: 1, borough: -1 })
////////////////////output/////////////////

```
restaurantDB> db.restaurants.find().sort({ cuisine: 1, borough: -1 })
[
  {
    _id: ObjectId('68ef60025813c60aea748a8e'),
    building_number: '123',
    street_name: 'MG Road',
    zip_code: '400001',
    coordinates: { longitude: 73.8567, latitude: 18.5204 },
    borough: 'Mumbai',
    cuisine_type: 'Indian',
    grades: [
      {
        date: ISODate('2024-01-15T00:00:00.000Z'),
        grade: 'A',
        score: 95
      },
      {
        date: ISODate('2023-06-10T00:00:00.000Z'),
        grade: 'B',
        score: 88
      }
    ]
  },
  {
    _id: ObjectId('68ef60025813c60aea748a90'),
    building_number: '67',
    street_name: 'Connaught Place',
    zip_code: '110001',
    coordinates: { longitude: 77.2167, latitude: 28.6667 },
    borough: 'Delhi',
    cuisine_type: 'Chinese',
    grades: [
      {
        date: ISODate('2023-12-05T00:00:00.000Z'),
        grade: 'B',
        score: 85
      },
```

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

// Use database

```
use restaurantDB

// Insert sample data
db.restaurants.insertMany([
  {
    restaurant_id: 1,
    name: "Wilson's Diner",
    borough: "Manhattan",
    cuisine: "American",
    address: {
      building: "123",
      street: "5th Avenue",
      zipcode: "10001",
      coord: [ -73.9557413, 40.7720266 ]  // Double values
    }
  },
  {
    restaurant_id: 2,
    name: "Green Garden",
    borough: "Queens",
    cuisine: "Vegetarian",
    address: {
      building: "456",
      street: "Main Street",
      zipcode: "11354",
      coord: [ "-73.9874", "40.7359" ]  // Strings, not Doubles
    }
  },
  {
    restaurant_id: 3,
    name: "Spice Hub",
    borough: "Brooklyn",
    cuisine: "Indian",
    address: {
      building: "789",
      // street missing intentionally here
      zipcode: "11211"
    }
  }
])

// Query 1: Find restaurants missing 'address.street'
db.restaurants.find({ "address.street": { $exists: false } })

// Query 2: Find restaurants where 'coord' array contains Double type values
db.restaurants.find({ "address.coord": { $type: "double" } })
```

//////////////////////output/////////////////////

```
ype "it" for more
estaurantDB> db.restaurants.find({ "address.coord": { $type: "double" } })

  {
    _id: ObjectId('68ef4d8f406119d2c9748a5f'),
    restaurant_id: 1,
    name: 'Monalisa Cafe',
    borough: 'Manhattan',
    cuisine: 'Italian',
    address: { coord: [ -73.9557413, 40.7720266 ] }
  },
  {
    _id: ObjectId('68ef4d8f406119d2c9748a60'),
    restaurant_id: 2,
    name: 'Harmony Diner',
    borough: 'Queens',
    cuisine: 'American',
    address: { coord: [ -73.9874, 40.7359 ] }
  },
```

**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 7.**

```
// Use database
use restaurantDB

// Insert sample data
db.restaurants.insertMany([
 {
  restaurant_id: 1,
  name: "Wilson's Diner",
  borough: "Manhattan",
  cuisine: "American",
  grades: [
   { date: new Date("2023-01-01"), grade: "A", score: 14 },
   { date: new Date("2023-02-01"), grade: "B", score: 21 }
  ]
 },
 {
  restaurant_id: 2,
  name: "Green Garden",
  borough: "Queens",
  cuisine: "Vegetarian",
  grades: [
   { date: new Date("2023-01-05"), grade: "A", score: 15 },
   { date: new Date("2023-02-10"), grade: "A", score: 28 }
  ]
 },
 {
  restaurant_id: 3,
  name: "Spice Hub",
```

```
    borough: "Brooklyn",
    cuisine: "Indian",
    grades: [
      { date: new Date("2023-01-12"), grade: "C", score: 13 },
      { date: new Date("2023-02-15"), grade: "B", score: 19 }
    ]
  }
])

// Query: Find restaurant_id, name, and grades for restaurants where any grade's score % 7 == 0
db.restaurants.find(
  { "grades.score": { $mod: [7, 0] } },
  { restaurant_id: 1, name: 1, grades: 1, _id: 0 }
)
```

////////////////////output/////////////////

```
restaurantDB> db.restaurants.find(
...   { "grades.score": { $mod: [7, 0] } },
...   { restaurant_id: 1, name: 1, grades: 1, _id: 0 }
... )
...
[
  {
    restaurant_id: 3,
    name: 'Green Garden',
    grades: [
      {
        date: ISODate('2023-02-18T00:00:00.000Z'),
        grade: 'C',
        score: 11
      },
      {
        date: ISODate('2022-10-10T00:00:00.000Z'),
        grade: 'B',
        score: 7
      }
    ]
  },
```

**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**

```
// Use database
use restaurantDB

// Insert sample data
db.restaurants.insertMany([
  {
    restaurant_id: 1,
    name: "Monalisa Cafe",
    borough: "Manhattan",
    cuisine: "Italian",
    address: {
```

```
    coord: [-73.9557413, 40.7720266]
   }
  },
  {
   restaurant_id: 2,
   name: "Harmony Diner",
   borough: "Queens",
   cuisine: "American",
   address: {
    coord: [-73.9874, 40.7359]
   }
  },
  {
   restaurant_id: 3,
   name: "Sunset Grill",
   borough: "Brooklyn",
   cuisine: "Mexican",
   address: {
    coord: [-73.9934, 40.7218]
   }
  }
])

// Query: Find name, borough, longitude, latitude, and cuisine for restaurants with 'mon' in the name
db.restaurants.find(
 { name: { $regex: "mon", $options: "i" } },
 { name: 1, borough: 1, "address.coord": 1, cuisine: 1, _id: 0 }
)
////////////////////////output//////////////////////////
```

```
restaurantDB> db.restaurants.find(
...   { name: { $regex: "mon", $options: "i" } },
...   { name: 1, borough: 1, "address.coord": 1, cuisine: 1, _id: 0 }
... )
...
[
  {
    name: 'Monalisa Cafe',
    borough: 'Manhattan',
    cuisine: 'Italian',
    address: { coord: [ -73.9557413, 40.7720266 ] }
  },
  {
    name: 'Harmony Diner',
    borough: 'Queens',
    cuisine: 'American',
    address: { coord: [ -73.9874, 40.7359 ] }
  },
  {
    name: 'Monalisa Cafe',
    borough: 'Manhattan',
    cuisine: 'Italian',
    address: { coord: [ -73.9557413, 40.7720266 ] }
  },
```

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

```
// Use database
use employeeDB

// Insert sample data into emp_professional_details collection
db.emp_professional_details.insertMany([
  { emp_id: 1, emp_name: "Amit Sharma", designation: "Manager", salary: 8000, incentive: 2000, working_hours: 48 },
  { emp_id: 2, emp_name: "Priya Singh", designation: "Software Engineer", salary: 7000, incentive: 1500, working_hours: 42 },
  { emp_id: 3, emp_name: "Rahul Verma", designation: "Accountant", salary: 6000, incentive: 1000, working_hours: 40 },
  { emp_id: 4, emp_name: "Sunita Patel", designation: "Manager", salary: 8500, incentive: 2500, working_hours: 50 },
  { emp_id: 5, emp_name: "Ramesh Kumar", designation: "Clerk", salary: 5000, incentive: 500, working_hours: 38 }
])

// Aggregation query to calculate sum, avg, min, and max salary
db.emp_professional_details.aggregate([
  {
    $group: {
      _id: null,
      totalSalary: { $sum: "$salary" },
      avgSalary: { $avg: "$salary" },
      minSalary: { $min: "$salary" },
      maxSalary: { $max: "$salary" }
    }
  }
])
```

////////////////////output////////////////////

```
...   {
...     $group: {
...       _id: null,
...       totalSalary: { $sum: "$salary" },
...       avgSalary: { $avg: "$salary" },
...       minSalary: { $min: "$salary" },
...       maxSalary: { $max: "$salary" }
...     }
...   }
... ])
...
[
  {
    _id: null,
    totalSalary: 34500,
    avgSalary: 6900,
    minSalary: 5000,
    maxSalary: 8500
  }
]
```

**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**
**# 1. Backup collections**

```
mongodump --db employee --collection emp_personal_details --out backup/
mongodump --db employee --collection emp_professional_details --out backup/
```

# 2. Delete some records (run in mongo shell)

```
use employee

db.emp_personal_details.deleteMany({ emp_id: { $in: [1, 2] } })
db.emp_professional_details.deleteMany({ emp_id: { $in: [1, 2] } })
```

# 3. Restore collections from backup (run in terminal)

```
mongorestore --db employee --collection emp_personal_details backup/employee/emp_personal_details.bson
mongorestore --db employee --collection emp_professional_details backup/employee/emp_professional_details.bson
```

# 4. Export collections to CSV and JSON (run in terminal)

```
mongoexport --db employee --collection emp_personal_details --type=csv --fields
emp_id,emp_name,emp_address,emp_DOB,emp_age,emp_mobilenumber --out emp_personal_details.csv

mongoexport --db employee --collection emp_personal_details --out emp_personal_details.json
```