MONGODB

Q1- Retrieve all employees

```
]
employees> //1. Retrieve all employee records.

employees> db.employee.find()
[
  {
    _id: 1,
    employee_id: 101,
    name: 'Alice',
    age: 30,
    department: 'HR',
    salary: 60000,
    joining_date: ISODate('2018-06-15T00:00:00.000Z')
  },
  {
    _id: 2,
    employee_id: 102,
    name: 'Bob',
    age: 25,
    department: 'IT',
    salary: 75000,
    joining_date: ISODate('2019-07-20T00:00:00.000Z')
  },
  {
    _id: 3,
    employee_id: 103,
    name: 'Charlie',
    age: 28,
    department: 'Finance',
    salary: 68000,
    joining_date: ISODate('2020-03-25T00:00:00.000Z')
  },
  {
    _id: 4,
```

2. Find employees in the IT department.

```
MongoInvalidArgumentError: Query filter must be a plain o
employees> db.employee.find({ department: "IT" })
[
  {
    _id: 2,
    employee_id: 102,
    name: 'Bob',
    age: 25,
    department: 'IT',
    salary: 75000,
    joining_date: ISODate('2019-07-20T00:00:00.000Z')
  },
  {
    _id: 4,
    employee_id: 104,
    name: 'David',
    age: 35,
    department: 'IT',
    salary: 90000,
    joining_date: ISODate('2017-05-30T00:00:00.000Z')
  }
]
```

3. Find employees aged between 25 and 30.

```
employees> db.employee.find({ age: { $gte: 25, $lte: 30 } })
[
  {
    _id: 1,
    employee_id: 101,
    name: 'Alice',
    age: 30,
    department: 'HR',
    salary: 60000,
    joining_date: ISODate('2018-06-15T00:00:00.000Z')
  },
  {
    _id: 2,
    employee_id: 102,
    name: 'Bob',
    age: 25,
    department: 'IT',
    salary: 75000,
    joining_date: ISODate('2019-07-20T00:00:00.000Z')
  },
  {
    _id: 3,
    employee_id: 103,
    name: 'Charlie',
    age: 28
```

4. Sort employees by salary in descending order.

```
employees> db.employee.find().sort({ salary: -1 })
[
  {
    _id: 4,
    employee_id: 104,
    name: 'David',
    age: 35,
    department: 'IT',
    salary: 90000,
    joining_date: ISODate('2017-05-30T00:00:00.000Z')
  },
  {
    _id: 2,
    employee_id: 102,
    name: 'Bob',
    age: 25,
    department: 'IT',
    salary: 75000,
    joining_date: ISODate('2019-07-20T00:00:00.000Z')
  },
  {
    _id: 3,
    employee_id: 103,
    name: 'Charlie',
    age: 28,
    department: 'Finance',
    salary: 68000,
    joining_date: ISODate('2020-03-25T00:00:00.000Z')
  },
  {
    _id: 5,
    employee_id: 105,
    name: 'Eve',
    age: 32,
    department: 'HR',
    salary: 65000,
```

5. Find the top 2 highest-paid employees.

```
]
employees> db.employee.find().sort({ salary: -1 }).limit(2)
[
  {
    _id: 4,
    employee_id: 104,
    name: 'David',
    age: 35,
    department: 'IT',
    salary: 90000,
    joining_date: ISODate('2017-05-30T00:00:00.000Z')
  },
  {
    _id: 2,
    employee_id: 102,
    name: 'Bob',
    age: 25,
    department: 'IT',
    salary: 75000,
    joining_date: ISODate('2019-07-20T00:00:00.000Z')
  }
]
```

6-Update salary of employee with employee_id 102 to 80,000.

```
employees> db.employee.updateOne({employee_id:102},{$set:{salary:80000}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 0,
  upsertedCount: 0
}
```

7. Increase salary by 10% for IT department employees.

```
employees> db.employee.updateMany({ department: "IT" },{ $mul: { salary: 1.10 } })
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 2,
  modifiedCount: 2,
  upsertedCount: 0
}
```

8. Remove an employee with employee_id 105.

```
employees> db.employee.deleteOne({ employee_id: 105 })
{ acknowledged: true, deletedCount: 1 }
employees>
```

9. Find distinct departments.

```
employees> db.employee.distinct("department")
[ 'Finance', 'HR', 'IT' ]
employees>
```

10. Find employees who are NOT in the HR department.

```
employees> db.employee.find({ department: { $ne: "HR" } })
[
  {
    _id: 2,
    employee_id: 102,
    name: 'Bob',
    age: 25,
    department: 'IT',
    salary: 88000,
    joining_date: ISODate('2019-07-20T00:00:00.000Z')
  },
  {
    _id: 3,
    employee_id: 103,
    name: 'Charlie',
    age: 28,
    department: 'Finance',
    salary: 68000,
    joining_date: ISODate('2020-03-25T00:00:00.000Z')
  },
  {
    _id: 4,
    employee_id: 104,
    name: 'David',
    age: 35,
    department: 'IT',
    salary: 99000.00000000001,
    joining_date: ISODate('2017-05-30T00:00:00.000Z')
  }
]
```

11. Find employees who do NOT work in IT or Finance.

```
employees> db.employee.find({ department: { $nin: ["IT", "Finance"] } })
[
  {
    _id: 1,
    employee_id: 101,
    name: 'Alice',
    age: 30,
    department: 'HR',
    salary: 60000,
    joining_date: ISODate('2018-06-15T00:00:00.000Z')
```

12. Find the average salary per department.

```
employees> db.employee.aggregate([{$group: {_id: "$department",avg_salary: { $avg: "$salary" }}}])
[
  { _id: 'HR', avg_salary: 60000 },
  { _id: 'IT', avg_salary: 93500 },
  { _id: 'Finance', avg_salary: 68000 }
]
```

13. Group employees by department and count them.

```
employees> db.employee.aggregate([{$group:{_id:"$department", totalemployees:{$sum:-1}}}])
[
  { _id: 'Finance', totalemployees: -1 },
  { _id: 'HR', totalemployees: -1 },
  { _id: 'IT', totalemployees: -2 }
]
```

15. Find employees who have an age field (existence check).

```
employees> db.employee.find({ age: { $exists: true } })
[
  {
    _id: 1,
    employee_id: 101,
    name: 'Alice',
    age: 30,
    department: 'HR',
    salary: 60000,
    joining_date: ISODate('2018-06-15T00:00:00.000Z')
  },
  {
    _id: 2,
    employee_id: 102,
    name: 'Bob',
    age: 25,
    department: 'IT',
    salary: 88000,
    joining_date: ISODate('2019-07-20T00:00:00.000Z')
  },
  {
    _id: 3,
    employee_id: 103,
    name: 'Charlie',
    age: 28,
    department: 'Finance',
    salary: 68000,
    joining_date: ISODate('2020-03-25T00:00:00.000Z')
  },
  {
```