1. Creating database employee

- **Create collections emp_personal_details with fields:**
  - **emp_id, emp_name, emp_address, emp_DOB, emp_age, emp_mobilenumber**

```
test> use employee

switched to db employee

employee>db.emp_personal_details.insertMany([

  { emp_id: 1,

emp_name: "Amit Sharma",

emp_address: "Delhi",

emp_DOB: "1980-05-15",

emp_age: 44,

emp_mobilenumber: "9876543210"

  },

  {  emp_id: 2,

emp_name: "NehaVerma",

emp_address: "Mumbai",

emp_DOB: "1992-08-20",

emp_age: 32,

emp_mobilenumber: "9823456789"

  },

  {   emp_id: 3,

emp_name: "Raj Mehta",

emp_address: "Ahmedabad",

emp_DOB: "1985-12-10",

emp_age: 39,

emp_mobilenumber: "9911223344"

  },
```

```
  {  emp_id: 4,
emp_name: "Kiran Desai",
emp_address: "Pune",
emp_DOB: "1990-04-05",
emp_age: 34,
emp_mobilenumber: "9900887766"
  },
  {  emp_id: 5,
emp_name: "Sunil Joshi",
emp_address: "Chennai",
emp_DOB: "1988-11-25",
emp_age: 36,
emp_mobilenumber: "9845123456"
  }]);
db.emp_personal_details.find();
```

OUTPUT:

```
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('68dca748b1904447cfeec4a9'),
    '1': ObjectId('68dca748b1904447cfeec4aa'),
    '2': ObjectId('68dca748b1904447cfeec4ab'),
    '3': ObjectId('68dca748b1904447cfeec4ac'),
    '4': ObjectId('68dca748b1904447cfeec4ad')
  }
}
employee> db.emp_personal_details.find();
[
  {
    _id: ObjectId('68a93e6c9092ca61b6eec4a9'),
    emp_id: 1,
    emp_name: 'Amit sharma',
    emp_address: 'Delhi',
    emp_DOB: '1980-05-15',
    emp_age: 44,
    emp_mobilenumber: '9876543210'
  },
  {
    _id: ObjectId('68a93e6c9092ca61b6eec4aa'),
    emp_id: 2,
    emp_name: 'Neha Verma',
    emp_address: 'Mumbai',
    emp_DOB: '1992-08-20',
    emp_age: 32,
    emp_mobilenumber: '9823456789'
employee>
  {
    _id: ObjectId('68a93e6c9092ca61b6eec4ab'),
    emp_id: 3,
    emp_name: 'Raj Mehta',
    emp_address: 'Ahmedabad',
    emp_DOB: '1985-12-10',
    emp_age: 39,
    emp_mobilenumber: '1234567890'
  },
```

```
{
  _id: ObjectId('68a93ec29092ca61b6eec4b1'),
  emp_id: 4,
  emp_name: 'Kiran Desai',
  emp_address: 'Pune',
  emp_DOB: '1990-04-05',
  emp_age: 34,
  emp_mobilenumber: '9900887766'
},
{
  _id: ObjectId('68a93ec29092ca61b6eec4b2'),
  emp_id: 5,
  emp_name: 'Sunil Joshi',
  emp_address: 'Chennai',
  emp_DOB: '1988-11-25',
  emp_age: 36,
  emp_mobilenumber: '9845123456'
},
```

2. Create another collection emp_professional_details

- **Fields:**
  - **emp_id, emp_name, designation, salary, incentive, working_hours**

```
test> use company

switched to db company

company>db.emp_profesional_details.insertMany([{

... emp_id:1,

... emp_name:"Amit",

... designation:"Manager",

... salary:9000,

... incentive:1200,

... working_hours:8

... },

... {emp_id:2,

... emp_name:"Neha",

... designation:"Developer",

... salary:6000,

... incentive:800,

... working_hours:9

... },

... {emp_id:3,

... emp_name:"Raj",

... designation:"Tester",

... salary:5000,

... incentive:600,

... working_hours:8

... },
```

```
... {emp_id:4,
... emp_name:"Priya",
... designation:"Designer",
... salary:5500,
... incentive:700,
... working_hours:7
... },
... {emp_id:5,
... emp_name:"Sunil",
... designation:"Analyst",
... salary:5800,
... incentive:500,
... working_hours:8
... }]);
```

OUTPUT:

```
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('68dcaef98de13592bbeec4a9'),
    '1': ObjectId('68dcaef98de13592bbeec4aa'),
    '2': ObjectId('68dcaef98de13592bbeec4ab'),
    '3': ObjectId('68dcaef98de13592bbeec4ac'),
    '4': ObjectId('68dcaef98de13592bbeec4ad')
  }
}
company> db.emp_professional_details.find();
[
  {
    _id: ObjectId('68a942645547ffa22feec4a9'),
    emp_id: 1,
    emp_name: 'Amit',
    designation: 'Manager',
    salary: 9000,
    incentive: 1200,
    working_hours: 8
  },
  {
    _id: ObjectId('68a942645547ffa22feec4aa'),
    emp_id: 2,
    emp_name: 'Neha',
    designation: 'Developer',
    salary: 6000,
    incentive: 800,
    working_hours: 9
  },
  {
    _id: ObjectId('68a942645547ffa22feec4ab'),
    emp_id: 3,
    emp_name: 'Raj',
    designation: 'Tester',
    salary: 5000,
    incentive: 600,
    working_hours: 8
  },
```

```
  {
    _id: ObjectId('68a942645547ffa22feec4ac'),
    emp_id: 4,
    emp_name: 'Priya',
    designation: 'Designer',
    salary: 5500,
    incentive: 700,
    working_hours: 7
company>
  {
    _id: ObjectId('68a942645547ffa22feec4ad'),
    emp_id: 5,
    emp_name: 'Sunil',
    designation: 'Analyst',
    salary: 5800,
    incentive: 500,
    working_hours: 8
  },
```

3. Insert 10 records

- **Insert into both collections:**
  - **emp_personal_details**
  - **emp_professional_details**
- **Show:**

1. **All employees having designation manager**
2. **All employees having salary 6000**

```
test> use employee

switched to db employee

employee>db.emp_personal_details.insertMany([{

emp_id: 1, name: "Amit", age: 45 },

  { emp_id: 2, name: "Neha", age: 30 },

  { emp_id: 3, name: "Raj", age: 50 },

  { emp_id: 4, name: "Priya", age: 28 },

  { emp_id: 5, name: "Vikram", age: 38 },

  { emp_id: 6, name: "Anjali", age: 41 },

  { emp_id: 7, name: "Sunil", age: 33 },

  { emp_id: 8, name: "Kiran", age: 39 },

  { emp_id: 9, name: "Manoj", age: 42 },

  { emp_id: 10, name: "Divya", age: 35 }

]);

test> use employee

switched to db employee

employee>db.emp_professional_details.insertMany([

  { emp_id: 1, designation: "manager", salary: 9000 },

  { emp_id: 2, designation: "developer", salary: 6000 },

  { emp_id: 3, designation: "manager", salary: 6000 },

  { emp_id: 4, designation: "analyst", salary: 5000 },

  { emp_id: 5, designation: "tester", salary: 4000 },
```

{ emp_id: 6, designation: "manager", salary: 9500 },

{ emp_id: 7, designation: "developer", salary: 6000 },

{ emp_id: 8, designation: "designer", salary: 5500 },

{ emp_id: 9, designation: "tester", salary: 6000 },

{ emp_id: 10, designation: "developer", salary: 8000 }

]);

Query:db.emp_professional_details.find({ designation: "manager" });

OUTPUT:

```
[
  {
    _id: ObjectId('68a947c9b849514eabeec4b3'),
    emp_id: 1,
    designation: 'manager',
    salary: 9000
  },
  {
    _id: ObjectId('68a947c9b849514eabeec4b5'),
    emp_id: 3,
    designation: 'manager',
    salary: 6000
  },
  {
    _id: ObjectId('68a947c9b849514eabeec4b8'),
    emp_id: 6,
    designation: 'manager',
    salary: 9500
  }
]
```

Query:db.emp_professional_details.find({ salary: 6000 });

OUTPUT:

```
[
  {
    _id: ObjectId('68a947c9b849514eabeec4b4'),
    emp_id: 2,
    designation: 'developer',
    salary: 6000
  },
  {
    _id: ObjectId('68a947c9b849514eabeec4b5'),
    emp_id: 3,
    designation: 'manager',
    salary: 6000
  },
  {
    _id: ObjectId('68a947c9b849514eabeec4b9'),
    emp_id: 7,
    designation: 'developer',
    salary: 6000
  },
  {
    _id: ObjectId('68a947c9b849514eabeec4bb'),
    emp_id: 9,
    designation: 'tester',
    salary: 6000
  }
]
```

4)1.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.

```
test> use company

switched to db company

company>db.emp_personal_details.insertMany([{

... emp_id:1,

... name:"Aachal",

... age:58

... }

... ,

... {emp_id:2,

... name:"aparna",

... age:63

... },

... {emp_id:3,

... name:"shruti",

... age:64

... },

... {emp_id:4,

... name:"manisha",

... age:24

... }]);
```

Query1:db.emp_personal_details.updateMany(

```
    { age: { $gt: 60 } },

    { $set: { status: "retired" } }
```

);

db.emp_personal_details.find();

OUTPUT:

```
  {
    _id: ObjectId('68dcb59fcb1f31ea8ceec4a9'),
    emp_id: 1,
    name: 'Aachal',
    age: 58
  },
  {
    _id: ObjectId('68dcb59fcb1f31ea8ceec4aa'),
    emp_id: 2,
    name: 'aparna',
    age: 63,
    status: 'retired'
  },
  {
    _id: ObjectId('68dcb59fcb1f31ea8ceec4ab'),
    emp_id: 3,
    name: 'shruti',
    age: 64,
    status: 'retired'
  },
  {
    _id: ObjectId('68dcb59fcb1f31ea8ceec4ac'),
    emp_id: 4,
    name: 'manisha',
    age: 24
  }
]
```

company>db.emp_professional_details.insertMany([

... {emp_id:1,name:"ravi",designation:"tester",working_hours:50,incentive:0},

... {emp_id:2,name:"aachal",designation:"accountant",working_hours:42,incentive:0},

... {emp_id:3,name:"amit",designation:"developer",working_hours:48,incentive:2000}

...]

...);

Query2:db.emp_professional_details.updateMany({working_hours:{$gt:45}},{$inc:{incentive:5000}});

db.emp_professional_details.find();

OUTPUT:

```
{
  _id: ObjectId('68dcbe6de4ba3ccd31eec4a9'),
  emp_id: 1,
  name: 'ravi',
  designation: 'tester',
  working_hours: 50,
  incentive: 5000
},
{
  _id: ObjectId('68dcbe6de4ba3ccd31eec4aa'),
  emp_id: 2,
  name: 'aachal',
  designation: 'accountant',
  working_hours: 42,
  incentive: 0
},
{
  _id: ObjectId('68dcbe6de4ba3ccd31eec4ab'),
  emp_id: 3,
  name: 'amit',
  designation: 'developer',
  working_hours: 48,
  incentive: 7000
}
```

Query3:db.emp_professional_details.updateMany({designation:"accountant"},{$inc:{salary: 1000}});

db.emp_professional_details.find();

OUTPUT:

```
{
  _id: ObjectId('68dcbe6de4ba3ccd31eec4a9'),
  emp_id: 1,
  name: 'ravi',
  designation: 'tester',
  working_hours: 50,
  incentive: 5000
},
{
  _id: ObjectId('68dcbe6de4ba3ccd31eec4aa'),
  emp_id: 2,
  name: 'aachal',
  designation: 'accountant',
  working_hours: 42,
  incentive: 0,
  salary: 1000
},
{
  _id: ObjectId('68dcbe6de4ba3ccd31eec4ab'),
  emp_id: 3,
  name: 'amit',
  designation: 'developer',
  working_hours: 48,
  incentive: 7000
}
```

5)1.Create index on emp_id in collection emp_professional_details

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

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

emp_id_1

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

emp_id_1_emp_name_1

db.emp_professional_details.getIndexes();

OUTPUT:

```
company> db.emp_professional_details.createIndex({emp_id:1});
emp_id_1
company> db.emp_professional_details.createIndex({emp_id:1,emp_name:1});
emp_id_1_emp_name_1
company> db.emp_professional_details.getIndexes();
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { 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.

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

```
test> use company

switched to db company

company>db.employees.insertMany([{

... name:"Alice",

... designation:"clerk",

... salary:60000},

... {name:"aachal",

... designation:"clerk",

... salary:34000},

... {name:"kumkum",

... designation:"software engineer",

... salary:55000},

... {name:"shital",

... designation:"manager",

... salary:26000},

... {name:"pooja",

... designation:"developer",

... salary:45000

... }]);
```

Query1:company>db.employees.aggregate([

```
... {$match:{designation:"clerk"}},

... {

... $group:{

... _id:"clerk",
```

... totalsalary:{$sum:"$salary"}

... } }

]);

Output:

```
[ { _id: 'clerk', totalsalary: 94000 } ]
```

Query2:db.employees.aggregate([

{ $match: { designation: "software engineer" } },

{

$group: {

_id: "software engineer",

minSalary: { $min: "$salary" }

}

}

]);

Output:

```
[ { _id: 'software engineer', minsalary: 55000 } ]
```

7)1.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.

use employee

switched to db employee

employee>db.employees.insertMany([

... {name:"Alice",department:"HR",mobile:["9876543210","9123456789"]},

... {name:"Bob",department:"IT",mobile:["9988776655"]},

... {name:"Charlie",department:"Finanace",mobile:["9090909090","8888888888"]},

... {name:"David",department:"marketing",mobile:["7777777777"]},

... {name:"Eve",department:"IT",mobile:["9999999999","9111111111"]}]);

Query1:employee>db.employees.aggregate([

... {$unwind:"$mobile"}]);

```
employee> db.employees.aggregate([
... {$unwind:"$mobile"}
... ]);
[
  {
    _id: ObjectId('68a96870dfb0c34920eec4a9'),
    name: 'Alice',
    department: 'HR',
    mobile: '9876543210'
  },
  {
    _id: ObjectId('68a96870dfb0c34920eec4a9'),
    name: 'Alice',
    department: 'HR',
    mobile: '9123456789'
  },
  {
    _id: ObjectId('68a96870dfb0c34920eec4aa'),
    name: 'Bob',
    department: 'IT',
    mobile: '9988776655'
  },
  {
    _id: ObjectId('68a96870dfb0c34920eec4ab'),
    name: 'Charlie',
    department: 'Finanace',
    mobile: '9090909090'
  },
```

```
  {
    _id: ObjectId('68a96870dfb0c34920eec4ab'),
    name: 'Charlie',
    department: 'Finanace',
    mobile: '8888888888'
  },
  {
    _id: ObjectId('68a96870dfb0c34920eec4ac'),
    name: 'David',
    department: 'marketing',
    mobile: '7777777777'
  },
  {
    _id: ObjectId('68a96870dfb0c34920eec4ad'),
    name: 'Eve',
    department: 'IT',
    mobile: '9999999999'
  },
  {
    _id: ObjectId('68a96870dfb0c34920eec4ad'),
    name: 'Eve',
    department: 'IT',
    mobile: '9111111111'
  }
]
```

Query2:employee>db.employees.find().skip(3);

```
employee> db.employees.find().skip(3);
[
  {
    _id: ObjectId('68a96870dfb0c34920eec4ac'),
    name: 'David',
    department: 'marketing',
    mobile: [ '7777777777' ]
  },
  {
    _id: ObjectId('68a96870dfb0c34920eec4ad'),
    name: 'Eve',
    department: 'IT',
    mobile: [ '9999999999', '9111111111' ]
  }
]
```

Query3:employee>db.employees.find().limit(4);

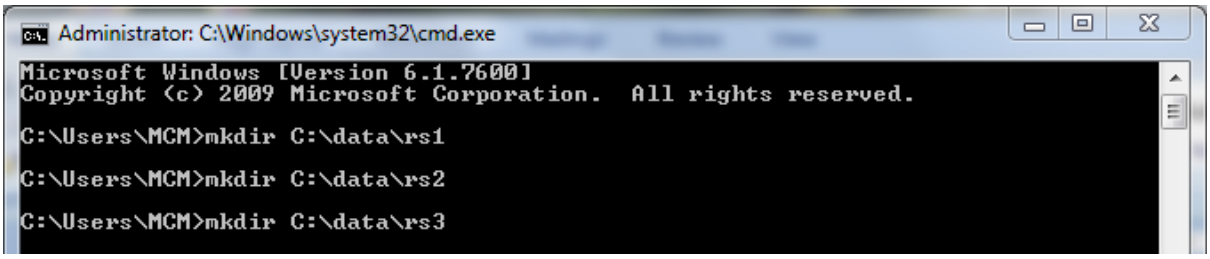
```
employee> db.employees.find().limit(4);
[
  {
    _id: ObjectId('68a96870dfb0c34920eec4a9'),
    name: 'Alice',
    department: 'HR',
    mobile: [ '9876543210', '9123456789' ]
  },
  {
    _id: ObjectId('68a96870dfb0c34920eec4aa'),
    name: 'Bob',
    department: 'IT',
    mobile: [ '9988776655' ]
  },
  {
    _id: ObjectId('68a96870dfb0c34920eec4ab'),
    name: 'Charlie',
    department: 'Finanace',
    mobile: [ '9090909090', '8888888888' ]
  },
  {
    _id: ObjectId('68a96870dfb0c34920eec4ac'),
    name: 'David',
    department: 'marketing',
    mobile: [ '7777777777' ]
  }
]
```

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

Step 1: Create directories

Command:

mkdir C:\data\rs1
mkdir C:\data\rs2
mkdir C:\data\rs3



Step 2: Start 3 MongoDB instances

Window 1 – Primary

mongod --replSet "rs0" --port 27017 --dbpath C:\data\rs1 --bind_iplocalhost

Window 2 – Secondary

mongod --replSet "rs0" --port 27018 --dbpath C:\data\rs2 --bind_iplocalhost

Window 3 – Secondary

mongod --replSet "rs0" --port 27019 --dbpath C:\data\rs3 --bind_iplocalhost

Step 3: Initialize Replica Set

Command:

mongo --port 27017

in mongodbshell

```
rs.initiate({
  _id: "rs0",
  members: [
    { _id: 0, host: "localhost:27017" },
    { _id: 1, host: "localhost:27018" },
    { _id: 2, host: "localhost:27019" }
  ]
})
```

```
{
  "ok" : 1
}
rs0:PRIMARY>
```

Step 4: Insert Employee Records

Command in PRIMARY (port 27017):

use employee

db.employees.insertMany([
  { emp_id: 101, name: "John", department: "HR", salary: 50000 },
  { emp_id: 102, name: "Alice", department: "IT", salary: 60000 },
  { emp_id: 103, name: "Bob", department: "Finance", salary: 55000 }
])

db.employees.find();

```
{
  "_id": ObjectId("65123abcd1234"),
  "emp_id": 101,
  "name": "John",
  "department": "HR",
  "salary": 50000
}
{
  "_id": ObjectId("65123abcd1235"),
  "emp_id": 102,
  "name": "Alice",
  "department": "IT",
  "salary": 60000
}
{
  "_id": ObjectId("65123abcd1236"),
  "emp_id": 103,
  "name": "Bob",
  "department": "Finance",
  "salary": 55000
}
```

Step 5: Read Records from Secondary

Connect to Secondary (port 27018):

mongo --port 27018
rs.slaveOk()
use employee
db.employees.find().pretty()

```
{
  "_id": ObjectId("65123abcd1234"),
  "emp_id": 101,
  "name": "John",
  "department": "HR",
  "salary": 50000
}
{
  "_id": ObjectId("65123abcd1235"),
  "emp_id": 102,
  "name": "Alice",
  "department": "IT",
  "salary": 60000
}
{
  "_id": ObjectId("65123abcd1236"),
  "emp_id": 103,
  "name": "Bob",
  "department": "Finance",
  "salary": 55000
}
```

9. Create MongoDB collection restaurants (same as 5)

1) Fields:Building number, Street name, Zip code, Coordinates, Borough, Cuisine type 2)Grades (includes: date, grade, score)

```
test> use mydb

switched to dbmydb

mydb>db.restaurants.insertOne([{

... building:1007,

... street:"Morris Park Ave",

... zipcode:"10462",

... coordinates:[-73.56789,40.24567],

... borough:"Bronx",

... cuisine:"Italian",

... grades:[{

... date:new Date("2024-02-10"),

... grade:"A",

... score:11},

... {date:new Date("2023-09-15"),

... grade:"B",

... score:17}

... ] }]);

mydb>db.restaurants.find();
```

OUTPUT:

```
mydb> db.restaurants.find();
[
  {
    '0': {
      building: 1007,
      street: 'morris park ave',
      zipcode: 10462,
      coordinates: [ -73.56789, 40.24567 ],
      borough: 'bronx',
      cuisine: 'italian',
      grades: [
        {
          date: ISODate('2024-02-10T00:00:00.000Z'),
          grade: 'A',
          score: 11
        },
        {
          date: ISODate('2023-09-15T00:00:00.000Z'),
          grade: 'B',
          score: 17
        }
      ]
    },
    _id: ObjectId('68dccc25543c3b6e98eec4a9')
```

10)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)


```
[{
"building_number": "101",
"street_name": "MG Road",
"zip_code": "411001",
"coordinates": { "longitude": 73.8567, "latitude": 18.5204 },
"borough": "Pune",
"cuisine": "South Indian",
"grades": [
{ "date": "2025-01-15", "grade": "A", "score": 10 },
{ "date": "2025-05-20", "grade": "B", "score": 8 }
]
},
{
"building_number": "202",
"street_name": "FC Road",
"zip_code": "411004",
"coordinates": { "longitude": 73.8419, "latitude": 18.5286 },
"borough": "Pune",
"cuisine": "North Indian",
"grades": [
{ "date": "2025-02-10", "grade": "A", "score": 12 },
{ "date": "2025-06-01", "grade": "C", "score": 6 }
]
},
{
"building_number": "303",
"street_name": "Linking Road",
"zip_code": "400050",
"coordinates": { "longitude": 72.8347, "latitude": 19.0665 },
"borough": "Mumbai",
"cuisine": "Chinese",
"grades": [
{ "date": "2025-03-05", "grade": "B", "score": 9 },
{ "date": "2025-07-12", "grade": "A", "score": 11 }
]
},
```

{
"building_number": "404",
"street_name": "JM Road",
"zip_code": "411005",
"coordinates": { "longitude": 73.8499, "latitude": 18.5293 },
"borough": "Pune",
"cuisine": "Italian",
"grades": [
{ "date": "2025-04-08", "grade": "A", "score": 15 },
{ "date": "2025-08-20", "grade": "B", "score": 9 }
]
}
]

OUTPUT:

```
use rest
switched to db rest
db.restaurant.find();
{
  _id: ObjectId('68dcd160297bf72e06035651'),
  building_number: '101',
  street_name: 'MG Road',
  zip_code: '411001',
  coordinates: {
    longitude: 73.8567,
    latitude: 18.5204
  },
  borough: 'Pune',
  cuisine: 'South Indian',
  grades: [
    {
      date: '2025-01-15',
      grade: 'A',
      score: 10
    },
    {
      date: '2025-05-20',
      grade: 'B',
      score: 8
    }
  ]
}
```

```
{
  _id: ObjectId('68dcd160297bf72e06035652'),
  building_number: '202',
  street_name: 'FC Road',
  zip_code: '411004',
  coordinates: {
    longitude: 73.8419,
    latitude: 18.5286
  },
  borough: 'Pune',
  cuisine: 'North Indian',
  grades: [
    {
      date: '2025-02-10',
      grade: 'A',
      score: 12
    },
    {
      date: '2025-06-01',
      grade: 'C',
      score: 6
    }
  ]
}
{
  _id: ObjectId('68dcd160297bf72e06035653'),
  building_number: '303',
  street_name: 'Linking Road',
  zip_code: '400050',
  coordinates: {
    longitude: 72.8347,
    latitude: 19.0665
  },
  borough: 'Mumbai',
  cuisine: 'Chinese',
  grades: [
    {
      date: '2025-03-05',
      grade: 'B',
      score: 9
    },
```

```
      {
        date: '2025-07-12',
        grade: 'A',
        score: 11
      }
    ]
  }
  {
    _id: ObjectId('68dcd160297bf72e06035654'),
    building_number: '404',
    street_name: 'JM Road',
    zip_code: '411005',
    coordinates: {
      longitude: 73.8499,
      latitude: 18.5293
    },
    borough: 'Pune',
    cuisine: 'Italian',
    grades: [
      {
        date: '2025-04-08',
        grade: 'A',
        score: 15
      },
      {
        date: '2025-08-20',
        grade: 'B',
        score: 9
      }
    ]
  }
```

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


```
[{"restaurant_id": "1001",
   "name": "Spice Hub",
   "borough": "Brooklyn",
   "cuisine": "Indian",
   "address": {
     "building": "12",
     "street": "Main Street",
     "zipcode": "11201",
     "coord": [-73.856077, 40.848447]
   },
   "grades": [
     { "date": "2024-05-01", "grade": "A", "score": 10 },
     { "date": "2024-06-15", "grade": "B", "score": 15 }
   ]
  },
  {"restaurant_id": "1002",
   "name": "Pizza Town",
   "borough": "Queens",
   "cuisine": "Italian",
   "address": {
     "building": "45",
     "street": "Broadway",
     "zipcode": "11375",
     "coord": [-73.961704, 40.662942]
   },
   "grades": [
```

{ "date": "2024-05-10", "grade": "A", "score": 12 }

]},

{"restaurant_id": "1003",

"name": "Sushi Place",

"borough": "Manhattan",

"cuisine": "Japanese",

"address": {

"building": "89",

"street": "Lexington Ave",

"zipcode": "10016",

"coord": [-73.982419, 40.579505]

},

"grades": [

{ "date": "2024-07-01", "grade": "C", "score": 25 }

] }]

Query1:db.restaurant.find()

```
>_MONGOSH
> use rest
< switched to db rest
> db.restaurant1.find()
< {
    _id: ObjectId('68dcde2b297bf72e06035669'),
    restaurant_id: '1001',
    name: 'Spice Hub',
    borough: 'Brooklyn',
    cuisine: 'Indian',
    address: {
      building: '12',
      street: 'Main Street',
      zipcode: '11201',
      coord: [
        -73.856077,
        40.848447
      ]
    },
    grades: [
      {
        date: '2024-05-01',
        grade: 'A',
        score: 10
      },
      {
        date: '2024-06-15',
        grade: 'B',
        score: 15
      }
```

```
{
  _id: ObjectId('68dcde2b297bf72e0603566a'),
  restaurant_id: '1002',
  name: 'Pizza Town',
  borough: 'Queens',
  cuisine: 'Italian',
  address: {
    building: '45',
    street: 'Broadway',
    zipcode: '11375',
    coord: [
      -73.961704,
      40.662942
    ]
  },
  grades: [
    {
      date: '2024-05-10',
      grade: 'A',
      score: 12
    }
  ]
}
```

Query2:db..restaurants.find({}, {

"restaurant_id": 1,

"name": 1,

"borough": 1,

"cuisine": 1,

"_id": 0})

```
> db.restaurant1.find({},{"restaurant_id":1,"name":1,"borough":1,"cuisine":1,"_id":0})
< {
    restaurant_id: '1001',
    name: 'Spice Hub',
    borough: 'Brooklyn',
    cuisine: 'Indian'
  }
  {
    restaurant_id: '1002',
    name: 'Pizza Town',
    borough: 'Queens',
    cuisine: 'Italian'
  }
  {
    restaurant_id: '1003',
    name: 'Sushi Place',
    borough: 'Manhattan',
    cuisine: 'Japanese'
  }
```

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.


```
[  {
    "restaurant_id": "1001",
    "name": "Spice Hub",
    "borough": "Brooklyn",
    "cuisine": "Indian"
  },
 {
    "restaurant_id": "1002",
    "name": "Pizza Town",
    "borough": "Bronx",
    "cuisine": "Italian"
  },
  {
    "restaurant_id": "1003",
    "name": "Sushi Place",
    "borough": "Manhattan",
    "cuisine": "Japanese" }
]
```


Query1:db.restaurants.find({}, {

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

```
< {
    restaurant_id: '1001',
    name: 'Spice Hub',
    borough: 'Brooklyn',
    cuisine: 'Indian'
  }
  {
    restaurant_id: '1002',
    name: 'Pizza Town',
    borough: 'Bronx',
    cuisine: 'Italian'
  }
  {
    restaurant_id: '1003',
    name: 'Sushi Place',
    borough: 'Manhattan',
    cuisine: 'Japanese'
  }
```

Query2:{ "borough": "Bronx" }

```
db.restaurant2.find({"borough":"Bronx"})
{
  _id: ObjectId('68dce7e9297bf72e0603567e'),
  restaurant_id: '1002',
  name: 'Pizza Town',
  borough: 'Bronx',
  cuisine: 'Italian'
}
```

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


```
[{   "restaurant_id": "1001",
   "name": "Bronx Diner",
   "borough": "Bronx",
   "cuisine": "American"
  },
  {"restaurant_id": "1002",
   "name": "Pizza Hub",
   "borough": "Bronx",
   "cuisine": "Italian"
  },
  {"restaurant_id": "1003",
   "name": "Curry House",
   "borough": "Bronx",
   "cuisine": "Indian"
  },
  {"restaurant_id": "1004",
   "name": "Dragon Express",
   "borough": "Bronx",
   "cuisine": "Chinese"
  },
  {"restaurant_id": "1005",
   "name": "Taco Villa",
   "borough": "Bronx",
   "cuisine": "Mexican"
  },
  {"restaurant_id": "1006",
   "name": "Bronx BBQ",
   "borough": "Bronx",
   "cuisine": "BBQ"
```

},
{"restaurant_id": "1007",
"name": "Bronx Sandwiches",
"borough": "Bronx",
"cuisine": "Fast Food"
},
{"restaurant_id": "1008",
"name": "Bronx Sweets",
"borough": "Bronx",
"cuisine": "Desserts"
},
{"restaurant_id": "1009",
"name": "Bronx Sushi",
"borough": "Bronx",
"cuisine": "Japanese"
},
{ "restaurant_id": "1010",
"name": "Bronx Grill",
"borough": "Bronx",
"cuisine": "Steakhouse"
}]

Query 1 – Display the first5restaurants in Bronx

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

```
> db.restaurant3.find({"borough":"Bronx"}).limit(5)
< {
    _id: ObjectId('68dce9cb297bf72e06035686'),
    restaurant_id: '1001',
    name: 'Bronx Diner',
    borough: 'Bronx',
    cuisine: 'American'
  }
  {
    _id: ObjectId('68dce9cb297bf72e06035687'),
    restaurant_id: '1002',
    name: 'Pizza Hub',
    borough: 'Bronx',
    cuisine: 'Italian'
  }
  {
    _id: ObjectId('68dce9cb297bf72e06035688'),
    restaurant_id: '1003',
    name: 'Curry House',
    borough: 'Bronx',
    cuisine: 'Indian'
  }
  {
    _id: ObjectId('68dce9cb297bf72e06035689'),
    restaurant_id: '1004',
    name: 'Dragon Express',
    borough: 'Bronx',
    cuisine: 'Chinese'
  }
  {
    _id: ObjectId('68dce9cb297bf72e0603568a'),
    restaurant_id: '1005',
    name: 'Taco Villa',
    borough: 'Bronx',
    cuisine: 'Mexican'
  }
```

Query2: Display the next 5 restaurants after skipping first 5

db.restaurant.find({ "borough": "Bronx" }).skip(5)

```
> db.restaurant3.find({"borough":"Bronx"}).skip(5)
< {
    _id: ObjectId('68dce9cb297bf72e0603568b'),
    restaurant_id: '1006',
    name: 'Bronx BBQ',
    borough: 'Bronx',
    cuisine: 'BBQ'
  }
  {
    _id: ObjectId('68dce9cb297bf72e0603568c'),
    restaurant_id: '1007',
    name: 'Bronx Sandwiches',
    borough: 'Bronx',
    cuisine: 'Fast Food'
  }
  {
    _id: ObjectId('68dce9cb297bf72e0603568d'),
    restaurant_id: '1008',
    name: 'Bronx Sweets',
    borough: 'Bronx',
    cuisine: 'Desserts'
  }
  {
    _id: ObjectId('68dce9cb297bf72e0603568e'),
    restaurant_id: '1009',
    name: 'Bronx Sushi',
    borough: 'Bronx',
    cuisine: 'Japanese'
  }
  {
    _id: ObjectId('68dce9cb297bf72e0603568f'),
    restaurant_id: '1010',
    name: 'Bronx Grill',
    borough: 'Bronx',
    cuisine: 'Steakhouse'
  }
```

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


```
[  {
    "restaurant_id": "2001",
    "name": "Bronx Diner",
    "borough": "Bronx",
    "cuisine": "American",
    "score": 95
  },
  {
    "restaurant_id": "2002",
    "name": "Pizza Hub",
    "borough": "Bronx",
    "cuisine": "Italian",
    "score": 88
  },
  {
    "restaurant_id": "2003",
    "name": "Curry House",
    "borough": "Brooklyn",
    "cuisine": "Indian",
    "score": 76
  },
  {
    "restaurant_id": "2004",
    "name": "Dragon Express",
    "borough": "Manhattan",
    "cuisine": "Chinese",
    "score": 92
  },
  {
```

"restaurant_id": "2005",
"name": "Taco Villa",
"borough": "Queens",
"cuisine": "Mexican",
"score": 82
}
]

Query1: Restaurants with score > 90

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

```
>_MONGOSH
> use rest
< switched to db rest
> db.restauramt4.find({"score":{$gt:90}})
< {
    _id: ObjectId('68dcef05297bf72e06035697'),
    restaurant_id: '2001',
    name: 'Bronx Diner',
    borough: 'Bronx',
    cuisine: 'American',
    score: 95
  }
  {
    _id: ObjectId('68dcef05297bf72e0603569a'),
    restaurant_id: '2004',
    name: 'Dragon Express',
    borough: 'Manhattan',
    cuisine: 'Chinese',
    score: 92
  }
```

Query2:db.restaurant.find({“score”:{$gt:80,$lt:100}})

```
> db.restauramt4.find({"score":{$gt:80,$lt:100}})
< {
    _id: ObjectId('68dcef05297bf72e06035697'),
    restaurant_id: '2001',
    name: 'Bronx Diner',
    borough: 'Bronx',
    cuisine: 'American',
    score: 95
  }
  {
    _id: ObjectId('68dcef05297bf72e06035698'),
    restaurant_id: '2002',
    name: 'Pizza Hub',
    borough: 'Bronx',
    cuisine: 'Italian',
    score: 88
  }
  {
    _id: ObjectId('68dcef05297bf72e0603569a'),
    restaurant_id: '2004',
    name: 'Dragon Express',
    borough: 'Manhattan',
    cuisine: 'Chinese',
    score: 92
  }
  {
    _id: ObjectId('68dcef05297bf72e0603569b'),
    restaurant_id: '2005',
    name: 'Taco Villa',
    borough: 'Queens',
    cuisine: 'Mexican',
    score: 82
  }
```

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


```
[
  {
    "restaurant_id": "3001",
    "name": "Bronx Diner",
    "borough": "Bronx",
    "cuisine": "Mexican",
    "grades": [ { "grade": "A", "score": 90 } ]
  },
  {
    "restaurant_id": "3002",
    "name": "Pizza Hub",
    "borough": "Manhattan",
    "cuisine": "Italian",
    "grades": [ { "grade": "B", "score": 75 } ]
  },
  {
    "restaurant_id": "3003",
    "name": "Curry House",
    "borough": "Queens",
    "cuisine": "Indian",
    "grades": [ { "grade": "A", "score": 85 } ]
  },
  {
    "restaurant_id": "3004",
    "name": "Dragon Express",
    "borough": "Brooklyn",
    "cuisine": "Chinese",
    "grades": [ { "grade": "A", "score": 92 } ]
  },
```

```
  {
    "restaurant_id": "3005",
    "name": "Taco Villa",
    "borough": "Bronx",
    "cuisine": "Mexican",
    "grades": [ { "grade": "C", "score": 70 } ]
  }
]
```

Query1: db.restaurant.find({

```
  "cuisine": { "$ne": "American" },
  "grades.grade": "A",
  "borough": { "$ne": "Brooklyn" }
})
```

```
> use rest
< switched to db rest
> db.restaurant5.find({
    "cuisine": { "$ne": "American" },
    "grades.grade": "A",
    "borough": { "$ne": "Brooklyn" }
  })
< {
    _id: ObjectId('68dcf3c6297bf72e060356a2'),
    restaurant_id: '3001',
    name: 'Bronx Diner',
    borough: 'Bronx',
    cuisine: 'Mexican',
    grades: [
      {
        grade: 'A',
        score: 90
      }
    ]
  }
  {
    _id: ObjectId('68dcf3c6297bf72e060356a4'),
    restaurant_id: '3003',
    name: 'Curry House',
    borough: 'Queens',
    cuisine: 'Indian',
    grades: [
      {
        grade: 'A',
        score: 85
      }
    ]
  }
```

Query2: Sort by Cuisine (Descending) db.restaurant5.find().sort({ "cuisine": -1 })

```
> db.restaurant5.find().sort({ "cuisine": -1 })
< {
    _id: ObjectId('68dcf3c6297bf72e060356a2'),
    restaurant_id: '3001',
    name: 'Bronx Diner',
    borough: 'Bronx',
    cuisine: 'Mexican',
    grades: [
      {
        grade: 'A',
        score: 90
      }
    ]
  }
  {
    _id: ObjectId('68dcf3c6297bf72e060356a6'),
    restaurant_id: '3005',
    name: 'Taco Villa',
    borough: 'Bronx',
    cuisine: 'Mexican',
    grades: [
      {
        grade: 'C',
        score: 70
      }
    ]
  }
  {
    _id: ObjectId('68dcf3c6297bf72e060356a3'),
    restaurant_id: '3002',
    name: 'Pizza Hub',
    borough: 'Manhattan',
    cuisine: 'Italian',
    grades: [
      {
        grade: 'B',
        score: 75
```

```
      }
    ]
  }
  {
    _id: ObjectId('68dcf3c6297bf72e060356a4'),
    restaurant_id: '3003',
    name: 'Curry House',
    borough: 'Queens',
    cuisine: 'Indian',
    grades: [
      {
        grade: 'A',
        score: 85
      }
    ]
  }
  {
    _id: ObjectId('68dcf3c6297bf72e060356a5'),
    restaurant_id: '3004',
    name: 'Dragon Express',
    borough: 'Brooklyn',
    cuisine: 'Chinese',
    grades: [
      {
        grade: 'A',
        score: 92
      }
    ]
  }
```

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.


```
[  {   "restaurant_id": "4001",
    "name": "Wilson Diner",
    "borough": "Bronx",
    "cuisine": "American"
  },
  {"restaurant_id": "4002",
    "name": "Wild Pizza",
    "borough": "Manhattan",
    "cuisine": "Italian"
  },
  {  "restaurant_id": "4003",
    "name": "Curry House",
    "borough": "Queens",
    "cuisine": "Indian"
  },
{ "restaurant_id": "4004",
    "name": "Wilkins Sushi",
    "borough": "Brooklyn",
    "cuisine": "Japanese"
  }]
```


Query1: Names starting with 'Wil'

{ "name": { "$regex": "^Wil" } }

```
> db.restaurant6.find({ "name": { "$regex": "^Wil" } })
< {
    _id: ObjectId('68dcf7ad297bf72e060356aa'),
    restaurant_id: '4001',
    name: 'Wilson Diner',
    borough: 'Bronx',
    cuisine: 'American'
  }
  {
    _id: ObjectId('68dcf7ad297bf72e060356ab'),
    restaurant_id: '4002',
    name: 'Wild Pizza',
    borough: 'Manhattan',
    cuisine: 'Italian'
  }
  {
    _id: ObjectId('68dcf7ad297bf72e060356ad'),
    restaurant_id: '4004',
    name: 'Wilkins Sushi',
    borough: 'Brooklyn',
    cuisine: 'Japanese'
  }
```

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.

[{"restaurant_id": "R001",
"name": "Palaces",
"borough": "Manhattan",
"cuisine": "Italian"
},
{"restaurant_id": "R002",
"name": "Delices",
"borough": "Brooklyn",
"cuisine": "French"
},
{ "restaurant_id": "R003",
"name": "Taste Hub",
"borough": "Queens",
"cuisine": "American"
},
{"restaurant_id": "R004",
"name": "Spices",
"borough": "Bronx",
"cuisine": "Indian"
}]

Query1: db.restaurant.find({ "name": { "$regex": "ces$", "$options": "i" } })

```
> db.restaurant7.find({ "name": { "$regex": "ces$", "$options": "i" } })
< {
    _id: ObjectId('68dcf933297bf72e060356b1'),
    restaurant_id: 'R001',
    name: 'Palaces',
    borough: 'Manhattan',
    cuisine: 'Italian'
  }
  {
    _id: ObjectId('68dcf933297bf72e060356b2'),
    restaurant_id: 'R002',
    name: 'Delices',
    borough: 'Brooklyn',
    cuisine: 'French'
  }
  {
    _id: ObjectId('68dcf933297bf72e060356b4'),
    restaurant_id: 'R004',
    name: 'Spices',
    borough: 'Bronx',
    cuisine: 'Indian'
  }
```

{

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

[{ "restaurant_id": "R001",
  "name": "Regal Palace",
  "borough": "Manhattan",
  "cuisine": "Italian"
 },
 { "restaurant_id": "R002",
  "name": "Delicious Treats",
  "borough": "Brooklyn",
  "cuisine": "French"
 },
 { "restaurant_id": "R003",
  "name": "The Great Regale",
  "borough": "Queens",
  "cuisine": "American"
 },
 { "restaurant_id": "R004",
  "name": "Spices Hub",
  "borough": "Bronx",
  "cuisine": "Indian"
 },
 { "restaurant_id": "R005",
  "name": "Food Corner",
  "borough": "Staten Island",
"cuisine": "Chinese" }]

Query1: Query Restaurants Whose Names Contain “Reg”

db.restaurant.find({ "name": { "$regex": "reg, "$options": "i" } })

```
db.restaurant8.find({ "name": { "$regex": "reg", "$options": "i" } })
{
  _id: ObjectId('68dcfb44297bf72e060356b8'),
  restaurant_id: 'R001',
  name: 'Regal Palace',
  borough: 'Manhattan',
  cuisine: 'Italian'
}
{
  _id: ObjectId('68dcfb44297bf72e060356ba'),
  restaurant_id: 'R003',
  name: 'The Great Regale',
  borough: 'Queens',
  cuisine: 'American'
}
```

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

[{ "restaurant_id": "R001",
"name": "Bronx Diner",
"borough": "Bronx",
"cuisine": "American"
},
{ "restaurant_id": "R002",
"name": "Golden Wok",
"borough": "Bronx",
"cuisine": "Chinese"
},
{ "restaurant_id": "R003",
"name": "Taste of Italy",
"borough": "Manhattan",
"cuisine": "Italian"
},
{ "restaurant_id": "R004",
"name": "Spicy Hub",
"borough": "Bronx",
"cuisine": "Indian"
},
{ "restaurant_id": "R005",
"name": "Burger Place",
"borough": "Queens",
"cuisine": "American"}]

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

```
> db.restaurant9.find( { "borough": "Bronx", "cuisine": { "$in": ["American", "Chinese"] } })
< {
    _id: ObjectId('68dcfcef297bf72e060356c0'),
    restaurant_id: 'R001',
    name: 'Bronx Diner',
    borough: 'Bronx',
    cuisine: 'American'
  }
  {
    _id: ObjectId('68dcfcef297bf72e060356c1'),
    restaurant_id: 'R002',
    name: 'Golden Wok',
    borough: 'Bronx',
    cuisine: '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.


```
[{  "restaurant_id": "R001",
  "name": "Bronx Diner",
  "borough": "Bronx",
  "cuisine": "American"
},
{"restaurant_id": "R002",
  "name": "Golden Wok",
  "borough": "Bronx",
  "cuisine": "Chinese"
},
{  "restaurant_id": "R003",
  "name": "Taste of Italy",
  "borough": "Manhattan",
  "cuisine": "Italian"
},
{"restaurant_id": "R004",
  "name": "Burger Place",
  "borough": "Queens",
  "cuisine": "American"
},
{  "restaurant_id": "R005",
  "name": "Food Corner",
  "borough": "Staten Island",
  "cuisine": "Chinese"
}]
```

Query: db.restaurant.find({ "borough": { "$in": ["Staten Island", "Queens", "Bronx", "Brooklyn"] } })

```
> db.restaurant10.find({ "borough": { "$in": ["Staten Island", "Queens", "Bronx", "Brooklyn"] } })
< {
    _id: ObjectId('68dcfdf75ea7cecbeadea5d0'),
    restaurant_id: 'R001',
    name: 'Bronx Diner',
    borough: 'Bronx',
    cuisine: 'American'
  }
  {
    _id: ObjectId('68dcfdf75ea7cecbeadea5d1'),
    restaurant_id: 'R002',
    name: 'Golden Wok',
    borough: 'Bronx',
    cuisine: 'Chinese'
  }
  {
    _id: ObjectId('68dcfdf75ea7cecbeadea5d3'),
    restaurant_id: 'R004',
    name: 'Burger Place',
    borough: 'Queens',
    cuisine: 'American'
  }
  {
    _id: ObjectId('68dcfdf75ea7cecbeadea5d4'),
    restaurant_id: 'R005',
    name: 'Food Corner',
    borough: 'Staten Island',
    cuisine: 'Chinese'
  }
```

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.

[{ "restaurant_id": "R001",

"name": "Bronx Diner",

"borough": "Bronx",

"cuisine": "American"

},

{ "restaurant_id": "R002",

"name": "Golden Wok",

"borough": "Bronx",

"cuisine": "Chinese"

},

{"restaurant_id": "R003",

"name": "Taste of Italy",

"borough": "Manhattan",

"cuisine": "Italian"

},

{ "restaurant_id": "R004",

"name": "Burger Place",

"borough": "Queens",

"cuisine": "American"

},

{ "restaurant_id": "R005",

"name": "Food Corner",

"borough": "Staten Island",

"cuisine": "Chinese"}]

Query:db.restaurant.find({ "borough": { $nin: ["Staten Island", "Queens", "Bronx", "Brooklyn"] } })

```
> db.restaurant11.find( { "borough": { $nin: ["Staten Island", "Queens", "Bronx", "Brooklyn"] } })
< {
    _id: ObjectId('68dcff505ea7cecbeadea5db'),
    restaurant_id: 'R003',
    name: 'Taste of Italy',
    borough: 'Manhattan',
    cuisine: 'Italian'
  }
```

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.


```
[{
"restaurant_id": "R001",
  "name": "Bronx Diner",
  "borough": "Bronx",
  "cuisine": "American",
  "grades": [{ "score": 8 }, { "score": 12 }]
},
  {
"restaurant_id": "R002",
  "name": "Golden Wok",
  "borough": "Brooklyn",
  "cuisine": "Chinese",
  "grades": [{ "score": 10 }, { "score": 15 }]
},
  {
  "restaurant_id": "R003",
  "name": "Taste of Italy",
  "borough": "Manhattan",
  "cuisine": "Italian",
  "grades": [{ "score": 12 }, { "score": 14 }]
},
  {
 "restaurant_id": "R004",
  "name": "Burger Place",
  "borough": "Queens",
  "cuisine": "American",
  "grades": [{ "score": 9 }, { "score": 7 }]
},
  {
 "restaurant_id": "R005",
```

"name": "Spicy Hub",

"borough": "Bronx",

"cuisine": "Indian",

"grades": [{ "score": 11 }, { "score": 13 }]}]

Query:db.restaurant.find({ "grades.score": { "$lte": 10 } })

```
> use rest
< switched to db rest
> db.restaurant.find( { "grades.score": { "$lte": 10 } })
< {
    _id: ObjectId('68dcd160297bf72e06035651'),
    building_number: '101',
    street_name: 'MG Road',
    zip_code: '411001',
    coordinates: {
      longitude: 73.8567,
      latitude: 18.5204
    },
    borough: 'Pune',
    cuisine: 'South Indian',
    grades: [
      {
        date: '2025-01-15',
        grade: 'A',
        score: 10
      },
      {
        date: '2025-05-20',
        grade: 'B',
        score: 8
      }
    ]
  }
  {
    _id: ObjectId('68dcd160297bf72e06035652'),
    building_number: '202',
    street_name: 'FC Road',
    zip_code: '411004',
    coordinates: {
      longitude: 73.8419,
      latitude: 18.5286
    },
    borough: 'Pune',
    cuisine: 'North Indian',
    grades: [
      {
        date: '2025-02-10',
```

```
      grade: 'A',
      score: 12
    },
    {
      date: '2025-06-01',
      grade: 'C',
      score: 6
    }
  ]
}
{
  _id: ObjectId('68dcd160297bf72e06035653'),
  building_number: '303',
  street_name: 'Linking Road',
  zip_code: '400050',
  coordinates: {
    longitude: 72.8347,
    latitude: 19.0665
  },
  borough: 'Mumbai',
  cuisine: 'Chinese',
  grades: [
    {
      date: '2025-03-05',
      grade: 'B',
      score: 9
    },
    {
      date: '2025-07-12',
      grade: 'A',
      score: 11
    }
  ]
}
{
  _id: ObjectId('68dcd160297bf72e06035654'),
```

```
  {
    _id: ObjectId('68dcd160297bf72e06035654'),
    building_number: '404',
    street_name: 'JM Road',
    zip_code: '411005',
    coordinates: {
      longitude: 73.8499,
      latitude: 18.5293
    },
    borough: 'Pune',
    cuisine: 'Italian',
    grades: [
      {
        date: '2025-04-08',
        grade: 'A',
        score: 15
      },
      {
        date: '2025-08-20',
        grade: 'B',
        score: 9
      }
    ]
  }
  {
    _id: ObjectId('68dcd560297bf72e06035657'),
    restaurant_id: '1001',
    name: 'Spice Hub',
    borough: 'Brooklyn',
    cuisine: 'Indian',
    address: {
      building: '12',
      street: 'Main Street',
      zipcode: '11201',
      coord: [
        -73.856077,
        40.848447
      ]
    },
    grades: [
      {
        date: '2024-05-01',
```

```
      date: '2024-05-01',
      grade: 'A',
      score: 10
    },
    {
      date: '2024-06-15',
      grade: 'B',
      score: 15
    }
  ]
}
{
  _id: ObjectId('68dcde03297bf72e06035664'),
  restaurant_id: '1001',
  name: 'Spice Hub',
  borough: 'Brooklyn',
  cuisine: 'Indian',
  address: {
    building: '12',
    street: 'Main Street',
    zipcode: '11201',
    coord: [
      -73.856077,
      40.848447
    ]
  },
  grades: [
    {
      date: '2024-05-01',
      grade: 'A',
      score: 10
    },
    {
      date: '2024-06-15',
      grade: 'B',
      score: 15
    }
  ]
}
```

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'.


```
[{"restaurant_id": "R001",
  "name": "Wilton Diner",
  "borough": "Bronx",
  "cuisine": "American"
},
  { "restaurant_id": "R002",
  "name": "Golden Wok",
  "borough": "Brooklyn",
  "cuisine": "Chinese"
},
{  "restaurant_id": "R003",
  "name": "Taste of Italy",
  "borough": "Manhattan",
  "cuisine": "Italian"
},
  {  "restaurant_id": "R004",
  "name": "Burger Place",
  "borough": "Queens",
  "cuisine": "American"
},
  { "restaurant_id": "R007",
  "name": "Food Corner",
  "borough": "Staten Island",
  "cuisine": "Chinese"
}]
```


Query:db.restaurant.find(

```
  {
 "$or": [
     { "cuisine": { "$nin": ["American", "Chinese"] } },
     { "name": { "$regex": "^Wil", "$options": "i" } }
```

```
  ]
})
```

```
< {
    _id: ObjectId('68dd03005ea7cecbeadea5e9'),
    restaurant_id: 'R001',
    name: 'Wilton Diner',
    borough: 'Bronx',
    cuisine: 'American'
  }
  {
    _id: ObjectId('68dd03005ea7cecbeadea5eb'),
    restaurant_id: 'R003',
    name: 'Taste of Italy',
    borough: 'Manhattan',
    cuisine: 'Italian'
  }
```

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

[{ "restaurant_id": "R001",
"name": "Wilton Diner",
"borough": "Bronx",
"cuisine": "American"
},
{ "restaurant_id": "R002",
"name": "Golden Wok",
"borough": "Brooklyn",
"cuisine": "Chinese"
},
{ "restaurant_id": "R003",
"name": "Taste of Italy",
"borough": "Manhattan",
"cuisine": "Italian"
},
{ "restaurant_id": "R004",
"name": "Burger Place",
"borough": "Queens",
"cuisine": "American"
},
{ "restaurant_id": "R007",
"name": "Food Corner",
"borough": "Staten Island",
"cuisine": "Chinese"
}]

Query: Tosort restaurants by name in descending order

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

```
db.re.find().sort({ name: -1 })
{
  _id: ObjectId('68dd04a15ea7cecbeadea5f2'),
  restaurant_id: 'R001',
  name: 'Wilton Diner',
  borough: 'Bronx',
  cuisine: 'American'
}
{
  _id: ObjectId('68dd04a15ea7cecbeadea5f4'),
  restaurant_id: 'R003',
  name: 'Taste of Italy',
  borough: 'Manhattan',
  cuisine: 'Italian'
}
{
  _id: ObjectId('68dd04a15ea7cecbeadea5f3'),
  restaurant_id: 'R002',
  name: 'Golden Wok',
  borough: 'Brooklyn',
  cuisine: 'Chinese'
}
{
  _id: ObjectId('68dd04a15ea7cecbeadea5f6'),
  restaurant_id: 'R007',
  name: 'Food Corner',
  borough: 'Staten Island',
  cuisine: 'Chinese'
}
{
  _id: ObjectId('68dd04a15ea7cecbeadea5f5'),
  restaurant_id: 'R004',
  name: 'Burger Place',
  borough: 'Queens',
  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.

[{ "restaurant_id": "R001",
  "name": "Wilton Diner",
  "borough": "Bronx",
  "cuisine": "American"
},
  { "restaurant_id": "R002",
  "name": "Golden Wok",
  "borough": "Brooklyn",
  "cuisine": "Chinese"
},
  { "restaurant_id": "R003",
  "name": "Taste of Italy",
  "borough": "Manhattan",
  "cuisine": "Italian"
},
  {"restaurant_id": "R004",
  "name": "Burger Place",
  "borough": "Queens",
  "cuisine": "American"
},
  { "restaurant_id": "R007",
  "name": "Food Corner",
  "borough": "Staten Island",
  "cuisine": "Chinese"
}]

Query:db.restaurant.find().sort({ "cuisine": 1, "borough": -1 })

```
db.r.find().sort({ "cuisine": 1, "borough": -1 })
{
  _id: ObjectId('68dd06285ea7cecbeadea5fd'),
  restaurant_id: 'R004',
  name: 'Burger Place',
  borough: 'Queens',
  cuisine: 'American'
}
{
  _id: ObjectId('68dd06285ea7cecbeadea5fa'),
  restaurant_id: 'R001',
  name: 'Wilton Diner',
  borough: 'Bronx',
  cuisine: 'American'
}
{
  _id: ObjectId('68dd06285ea7cecbeadea5fe'),
  restaurant_id: 'R007',
  name: 'Food Corner',
  borough: 'Staten Island',
  cuisine: 'Chinese'
}
{
  _id: ObjectId('68dd06285ea7cecbeadea5fb'),
  restaurant_id: 'R002',
  name: 'Golden Wok',
  borough: 'Brooklyn',
  cuisine: 'Chinese'
}
{
  _id: ObjectId('68dd06285ea7cecbeadea5fc'),
  restaurant_id: 'R003',
  name: 'Taste of Italy',
  borough: 'Manhattan',
  cuisine: 'Italian'
}
```

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

[{ "restaurant_id": "R001",
"name": "Bronx Diner",
"borough": "Bronx",
"cuisine": "American",
"address": { "street": "Main Street", "zipcode": "10453", "building": "101" }
},
{ "restaurant_id": "R002",
"name": "Golden Wok",
"borough": "Brooklyn",
"cuisine": "Chinese",
"address": { "zipcode": "11201", "building": "12" }
},
{"restaurant_id": "R003",
"name": "Taste of Italy",
"borough": "Manhattan",
"cuisine": "Italian",
"address": { "street": "Broadway", "zipcode": "10001", "building": "45" }
}]

Query:db.restaurant.find({ "address.street": { "$exists": false } })

```
> db.rj.find({ "address.street": { "$exists": false } })
< {
    _id: ObjectId('68dd07195ea7cecbeadea603'),
    restaurant_id: 'R002',
    name: 'Golden Wok',
    borough: 'Brooklyn',
    cuisine: 'Chinese',
    address: {
      zipcode: '11201',
      building: '12'
    }
  }
```

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

[{"restaurant_id": "R001",

"name": "Bronx Diner",

"borough": "Bronx",

"cuisine": "American",

"address": { "coord": [40.1234, -73.5678] }

},

{ "restaurant_id": "R002",

"name": "Golden Wok",

"borough": "Brooklyn",

"cuisine": "Chinese",

"address": { "coord": [40.5678, -73.1234] }

},

{"restaurant_id": "R003",

"name": "Taste of Italy",

"borough": "Manhattan",

"cuisine": "Italian",

"address": { "coord": ["40.0000", "-73.0000"] }

}]

Query:db.restaurant.find({ "address.coord": { "$type": "double" } })

```
> db.restaurant.find(
    { "address.coord": { "$type": "double" } },
    { restaurant_id: 1, name: 1, "address.coord": 1, _id: 0 }
  );
< {
    restaurant_id: '1001',
    name: 'Spice Hub',
    address: {
      coord: [
        -73.856077,
        40.848447
      ]
    }
  }
  {
    restaurant_id: '1002',
    name: 'Pizza Town',
    address: {
      coord: [
        -73.961704,
        40.662942
      ]
    }
  }
  {
    restaurant_id: '1003',
    name: 'Sushi Place',
    address: {
      coord: [
        -73.982419,
        40.579505
      ]
    }
  }
  {
    restaurant_id: '1001',
    name: 'Spice Hub',
    address: {
      coord: [
        -73.856077,
        40.848447
```

```
        40.848447
      ]
    }
  }
  {
    restaurant_id: '1002',
    name: 'Pizza Town',
    address: {
      coord: [
        -73.961704,
        40.662942
      ]
    }
  }
  {
    restaurant_id: '1003',
    name: 'Sushi Place',
    address: {
      coord: [
        -73.982419,
        40.579505
      ]
    }
  }
```

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.

[{

"restaurant_id": "R001",

"name": "Bronx Diner",

"grades": [{ "score": 14 }, { "score": 8 }]

},

{

"restaurant_id": "R002",

"name": "Golden Wok",

"grades": [{ "score": 10 }, { "score": 21 }]

},

{

"restaurant_id": "R003",

"name": "Taste of Italy",

"grades": [{ "score": 9 }, { "score": 15 }]

}]

Query:db.restaurant.find({ "grades.score": { "$mod": [7, 0] } })

```
> db.ang.find({ "grades.score": { "$mod": [7, 0] } })
< {
    _id: ObjectId('68dd09685ea7cecbeadea60e'),
    restaurant_id: 'R001',
    name: 'Bronx Diner',
    grades: [
      {
        score: 14
      },
      {
        score: 8
      }
    ]
  }
  {
    _id: ObjectId('68dd09685ea7cecbeadea60f'),
    restaurant_id: 'R002',
    name: 'Golden Wok',
    grades: [
      {
        score: 10
      },
      {
        score: 21
      }
    ]
  }
```

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

[{ "restaurant_id": "R001",

"name": "Bronx Diner",

"borough": "Bronx",

"cuisine": "American",

"address": { "coord": [-73.5678, 40.1234] }

},

{"restaurant_id": "R002",

"name": "Monaco Wok",

"borough": "Brooklyn",

"cuisine": "Chinese",

"address": { "coord": [-73.1234, 40.5678] }

},

{"restaurant_id": "R003",

"name": "Taste of Italy",

"borough": "Manhattan",

"cuisine": "Italian",

"address": { "coord": [-73.0000, 40.0000] }

}]

Query:db.restaurant.find({ "name": { "$regex": "mon", "$options": "i" } })

```
> db.program.find( { "name": { "$regex": "mon", "$options": "i" } })
< {
    _id: ObjectId('68dd0a995ea7cecbeadea615'),
    restaurant_id: 'R002',
    name: 'Monaco Wok',
    borough: 'Brooklyn',
    cuisine: 'Chinese',
    address: {
      coord: [
        -73.1234,
        40.5678
      ]
    }
  }
rest>
```

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

```
[ {
  "restaurant_id": "R001",
  "name": "Bronx Diner",
  "grades": [{ "score": 14 }, { "score": 7 }]
},
{  "restaurant_id": "R002",
  "name": "Golden Wok",
  "grades": [{ "score": 21 }, { "score": 10 }]
},
{"restaurant_id": "R003",
  "name": "Taste of Italy",
  "grades": [{ "score": 9 }, { "score": 15 }]
}]
```

Query:db.restaurants.aggregate([

```
  { $unwind: "$grades" },
  {
    $group: {
      _id: "$name",
totalScore: { $sum: "$grades.score" },
avgScore: { $avg: "$grades.score" },
minScore: { $min: "$grades.score" },
maxScore: { $max: "$grades.score" }
    }
  }
])
```

```
> db.silver.aggregate([
    { $unwind: "$grades" },
    {
      $group: {
        _id: "$name",
        totalScore: { $sum: "$grades.score" },
        avgScore: { $avg: "$grades.score" },
        minScore: { $min: "$grades.score" },
        maxScore: { $max: "$grades.score" }
      }
    }
  ])
< {
    _id: 'Bronx Diner',
    totalScore: 21,
    avgScore: 10.5,
    minScore: 7,
    maxScore: 14
  }
  {
    _id: 'Golden Wok',
    totalScore: 31,
    avgScore: 15.5,
    minScore: 10,
    maxScore: 21
  }
  {
    _id: 'Taste of Italy',
    totalScore: 24,
    avgScore: 12,
    minScore: 9,
    maxScore: 15
  }
test>
```

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.Emp_personal_details


```
[{
  "emp_id": 101,
  "name": "Mahesh Borle",
  "age": 28,
  "department": "IT"
},
  {
  "emp_id": 102,
  "name": "RohitPatil",
  "age": 32,
  "department": "HR"
},
  {
  "emp_id": 103,
  "name": "Sneha Kulkarni",
  "age": 25,
  "department": "Finance"
}]
```


2)Emp_professional_details


```
[{
  "emp_id": 101,
  "designation": "Software Engineer",
  "salary": 50000,
  "experience": 3
},
  {
  "emp_id": 102,
  "designation": "HR Manager",
```

"salary": 60000,
"experience": 7
},
{
"emp_id": 103,
"designation": "Accountant",
"salary": 40000,
"experience": 2
}]

Query:After deleting some records

Emp_personal_details:

{
"_id": {
"$oid": "68c6813cb2970a6f9da5df48"
},
"emp_id": 101,
"name": "Mahesh Borle",
"age": 28,
"department": "IT"
}
{
"_id": {
"$oid": "68c6813cb2970a6f9da5df4a"
},
"emp_id": 103,
"name": "Sneha Kulkarni",
"age": 25,
"department": "Finance"
}

Emp_professional_details:

{
"_id": {

"$oid": "68c6814fb2970a6f9da5df4d"
},
"emp_id": 101,
"designation": "Software Engineer",
"salary": 50000,
"experience": 3
}
{
"_id": {
"$oid": "68c6814fb2970a6f9da5df4e"
},
"emp_id": 102,
"designation": "HR Manager",
"salary": 60000,
"experience": 7
}

After Restoring from Backup

emp_personal_details :

{ emp_id: 101,
name: "Mahesh Borle",
age: 28,
department: "IT"
}
{ emp_id: 102,
name: "RohitPatil",
age: 32,
department: "HR"
}
{ emp_id: 103,
name: "Sneha Kulkarni",

age: 25,

department: "Finance" }

emp_professional_details:

{ emp_id: 101,

designation: "Software Engineer",

salary: 50000,

experience: 3

}

{ emp_id: 102,

designation: "HR Manager",

salary: 60000,

experience: 7 }

{ emp_id: 103,

designation: "Accountant",

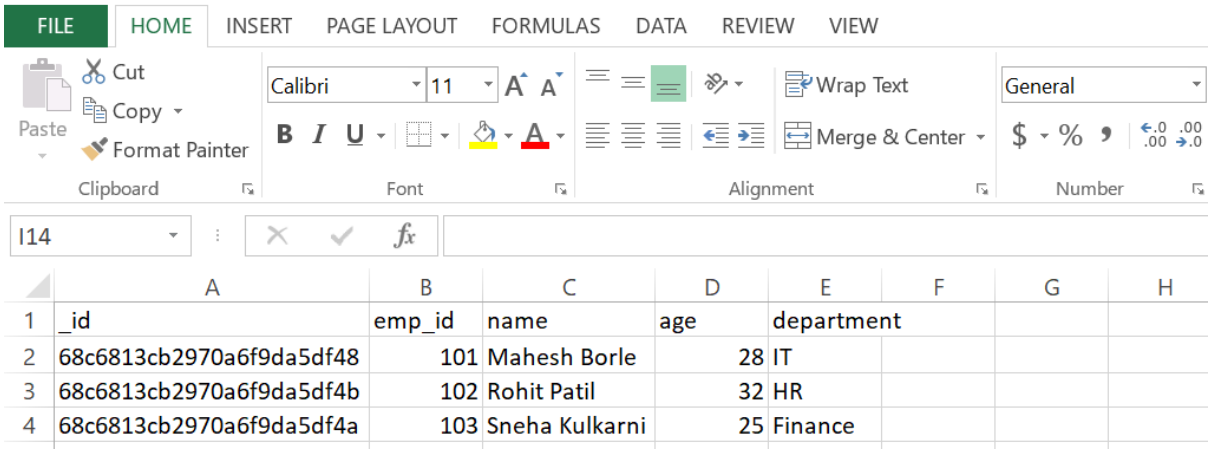
salary: 40000,

experience: 2 }

Emp_personal_details:Json Format


```
{
emp_id: 101,
  name: "Mahesh Borle",
  age: 28,
  department: "IT"
}
{
emp_id: 102,
  name: "RohitPatil",
  age: 32,
  department: "HR"
}
{
emp_id: 103,
  name: "Sneha Kulkarni",
  age: 25,
  department: "Finance"
}
```

Emp_personal_details:CSV format



	A	B	C	D	E	F	G	H
1	_id	emp_id	name	age	department			
2	68c6813cb2970a6f9da5df48	101	Mahesh Borle	28	IT			
3	68c6813cb2970a6f9da5df4b	102	Rohit Patil	32	HR			
4	68c6813cb2970a6f9da5df4a	103	Sneha Kulkarni	25	Finance			

Emp_professional_details:Json Format

{

emp_id: 101,

  designation: "Software Engineer",

  salary: 50000,

  experience: 3

}

{

emp_id: 102,

  designation: "HR Manager",

  salary: 60000,

  experience: 7

}

{

emp_id: 103,

  designation: "Accountant",

  salary: 40000,

  experience: 2

}

Emp_professional_details:CSV format

_id	emp_id	designation	salary	experience
68c6814fb2970a6f9da5df4d	101	Software Engineer	50000	3
68c6814fb2970a6f9da5df4e	102	HR Manager	60000	7
68c6828cb2970a6f9da5df54	103	Accountant	40000	2