Practical no:-1

Creating database employee

Create collections emp_personal_details with emp_id, emp_name, emp_address,

emp_DOB, emp_age, emp_mobilenumber

employee> db.createCollection("employee");

{ ok: 1 }

employee> db.employee.insertOne({

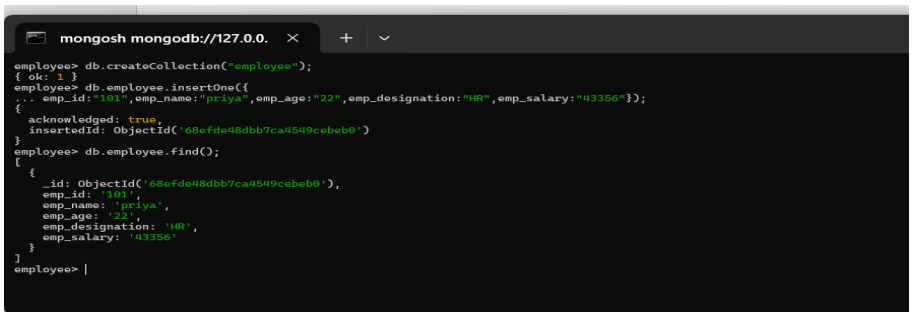
...
emp_id:"101",emp_name:"priya",emp_age:"22",emp_designation:"HR",emp_salary:"43356"});

employee> db.employee.find();

//To show collections

show collections

Output:-



Practical No.02

Create another collection emp_professional_details with emp_id, emp_name, designation, salary, incentive, working_hours

```
test> use employee

switched to db employee

employee> db.createCollection("emp_personal_details");

{ ok: 1 }

employee> db.emp_personal_details.insertMany([{

... emp_id:"101",

... emp_name:"priya",

... emp_designation:"software engg",

... salary:40000,

... incentive:5000,

... w_hours:8

... },

... {

... emp_id:102,

... emp_name:"sakshi",

... emp_designation:"HR",

... salary:30000,

... incentive:4000,

... w_hours:7

... },
```

```
... {
... emp_id:103,
... emp_name:"riya",
... emp_designation:"Manager",
... salary:60000,
... incentive:5500,
... w_hours:9
... },
... {
... emp_id:104,
... emp_name:"sayali",
... emp_designation:"Excutive",
... salary:30000,
... incentive:3000,
... w_hours:6
... },
... {
... emp_id:105,
... emp_name:"baghyashri",
... emp_designation:"web developer",
... salary:60000,
... incentive:6000,
... w_hours:9
... {
... emp_id:106,
... emp_name:"mohini",
... emp_designation:"manager",
... salary:40500,
```

... incentive:7500,

... w_hours:6

... },

... {

... emp_id:107,

... emp_name:"prachi",

... emp_designation:"HR",

... salary:20000,

... incentive:4000,

... w_houra:8

... }

...]

...);

Output:-

```
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('68ac4471904beb8f87eec4a9'),
    '1': ObjectId('68ac4471904beb8f87eec4aa'),
    '2': ObjectId('68ac4471904beb8f87eec4ab'),
    '3': ObjectId('68ac4471904beb8f87eec4ac'),
    '4': ObjectId('68ac4471904beb8f87eec4ad'),
    '5': ObjectId('68ac4471904beb8f87eec4ae'),
    '6': ObjectId('68ac4471904beb8f87eec4af')
  }
}
employee>
```

```
employee> db.emp_personal_details.find()
[
  {
    _id: ObjectId('68ac4471904beb8f87eec4a9'),
    emp_id: '101',
    emp_name: 'priya',
    emp_designation: 'software engg',
    salary: 40000,
    incentive: 5000,
    w_hours: 8
  },
  {
    _id: ObjectId('68ac4471904beb8f87eec4aa'),
    emp_id: 102,
    emp_name: 'sakshi',
    emp_designation: 'HR',
    salary: 30000,
    incentive: 4000,
    w_hours: 7
  },
  {
    _id: ObjectId('68ac4471904beb8f87eec4ab'),
    emp_id: 103,
    emp_name: 'riya',
    emp_designation: 'Manager',
    salary: 60000,
    incentive: 5500,
    w_hours: 9
  },
  {
```

Practical No.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

```
test> use employee
switched to db employee
employee> db.createCollection("emp_personal1");
{ ok: 1 }

employee> db.emp_personal1.insertMany([{
id:1,
name:"mayuri",
add: "pune" ,
dob:new Date("2000-05-20"),
age:66,
mob_no:2345232221},

{
id:2,
name:"Akanksha",
add: "Nashik" ,
dob:new Date("2010-05-10"),
age:33,
mob_no:9343232221},
{
```

id:3,

name:"Raj",

add: "patna" ,

dob:new Date(“2011-05-10”),

age:65,

mob_no:9343442221},

{

id:4,

name:"Akansh",

add: "thane" ,

dob:new Date(“2010-12-10”),

age:25,

mob_no:9343232222},

{

id:5,

name:"Yash",

add: "Nashik" ,

dob:new Date(“2012-05-10”),

age:12,

mob_no:9343232221},

{
id:6,
name:"Piyush",
add: "jalgoan" ,
dob:new Date(“2008-05-10”),
age:70,
mob_no:9343232221},

{
id:7,
name:"Chetan",
add: "chopda" ,
dob:new Date(“2012-05-10”),
age:30,
mob_no:9353232221},

{
id:8,
name:"laxmi",
add: "jalgoan" ,
dob:new Date(“2010-03-10”),
age:44,
mob_no:9343272221},
{
id:9,
name:"Ritik",
add: "thane" ,
dob:new Date(“2020-05-10”),
age:13,

```
mob_no:9643232221},

{
id:10,
name:"Minal",
add: "Nashik" ,
dob:new Date("2010-04-10"),
age:28,
mob_no:9443232221}])
employee>db.emp_personal1.find();
employee> db.createCollection("emp_professional");
{ ok: 1 }
employee> db.emp_ professional.insertMany([{
id:1,
name:"mayuri",
add: "pune" ,
desig: "manager",
salary:60000,
incentive:5000,
work:44 },
{
id:2,
name:"mayu",
add: "nashik" ,
desig: "accountant",
salary:30000,
incentive:3000,
work:60 },
{
```

id:3,
name:"rahul",
add: "nagpur" ,
desig: "HOD",
salary:90000,
incentive:3000,
work:80 },
{
id:4,
name:"Aachal",
add: "Mumbai" ,
desig: "accountant",
salary:80000,
incentive:7000,
work:30 },
id:3,
name:"mayuri",
add: "pune" ,
desig: "manager",
salary:60000,
incentive:5000,
work:44 },
{
id:4,
name:"mayu",
add: "nashik" ,
desig: "accountant",
salary:30000,
incentive:3000,

work:60 },
{
id:5,
name:"Raj",
add: "Nashik" ,
desig: "manager",
salary:40000,
incentive:5000,
work:44 },
{
id:6,
name:"Prachi",
add: "Jalgoan" ,
desig: "Principle",
salary:70000,
incentive:3000,
work:60 },
{
id:7,
name:"Kajal",
add: "pune" ,
desig: "manager",
salary:80000,
incentive:5000,
work:44 },
{
id:8,
name:"Shruti",
add: "Thane" ,

```
desig: "Shweta",
salary:10000,
incentive:4000,
work:60 },

id:9,
name:"Piyush",
add: "Mumbai" ,
desig: "manager",
salary:80000,
incentive:5000,
work:20 },
{
id:10,
name:"Sapna",
add: "nashik" ,
desig: "accountant",
salary:50000,
incentive:3000,
work:60 },

employee>db.emp_professional.find();

employee>db.emp__professional.find({desig:”accountant”});
```

```
employee> db.emp_professional.find({desig:"doctor"});
[
  {
    _id: ObjectId('686f80ad344d559809748a60'),
    id: 102,
    name: 'Akshu',
    desig: 'doctor',
    salary: 40000,
    incentive: 5000,
    work: 6
  },
  {
    _id: ObjectId('686f82a5344d559809748a62'),
    id: 102,
    name: 'mayu',
    desig: 'doctor',
    salary: 70000,
    incentive: 5000,
    work: 8
  },
  {
    _id: ObjectId('686f82a5344d559809748a67'),
    id: 107,
    name: 'Rahul',
    desig: 'doctor',
    salary: 60000,
    incentive: 5000,
    work: 6
  }
]
```

employee>db.emp_professional.find({salary:60000});

```
employee> db.emp_professional.find({salary:60000});
[
  {
    _id: ObjectId('686f80ad344d559809748a5f'),
    id: 101,
    name: 'mayuri',
    desig: 'eng',
    salary: 60000,
    incentive: 5000,
    work: 5
  },
  {
    _id: ObjectId('686f82a5344d559809748a61'),
    id: 101,
    name: 'mayuri',
    desig: 'eng',
    salary: 60000,
    incentive: 5000,
    work: 5
  },
  {
    _id: ObjectId('686f82a5344d559809748a67'),
    id: 107,
    name: 'Rahul',
    desig: 'doctor',
    salary: 60000,
    incentive: 5000,
    work: 6
  }
]
```

Practical No.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

employee> db.Emp.insertMany([

... {emp_id:"1",emp_name:"priya",emp_add:"101 plot nehrunagar",emp_age:"22",emp_mobno:"5364575344"},

... {emp_id:"2",emp_name:"sakshi",emp_add:"102 plot shivajinagar",emp_age:"64",emp_mobno:"764346745"}
... {emp_id:"3",emp_name:"sayali",emp_add:"202 city apartment",emp_age:"45",amp_mobno:"568732866"},

... {emp_id:"4",emp_name:"riya",emp_add:"404 main street",emp_age:"68",amp_mobno:"335657246"}]);

```
employee> db.Emp.insertMany([
... {emp_id:"1",emp_name:"priya",emp_add:"101 plot nehrunagar",emp_age:"22",emp_mobno:"5364575344"},
... {emp_id:"2",emp_name:"sakshi",emp_add:"102 plot shivajinagar",emp_age:"64",emp_mobno:"764346745"},
... {emp_id:"3",emp_name:"sayali",emp_add:"202 city apartment",emp_age:"45",amp_mobno:"568732866"},
... {emp_id:"4",emp_name:"riya",emp_add:"404 main street",emp_age:"68",amp_mobno:"335657246"}]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('68efd943dbb7ca4549cebeac'),
    '1': ObjectId('68efd943dbb7ca4549cebead'),
    '2': ObjectId('68efd943dbb7ca4549cebeae'),
    '3': ObjectId('68efd943dbb7ca4549cebeaf')
  }
}
employee>
```

```
employee> db.Emp.find();
{
  _id: ObjectId('68efd943dbb7ca4549cebeac'),
  emp_id: '1',
  emp_name: 'priya',
  emp_add: '101 plot nehrunagar',
  emp_age: '22',
  emp_mobno: '5364575344'
},
{
  _id: ObjectId('68efd943dbb7ca4549cebead'),
  emp_id: '2',
  emp_name: 'sakshi',
  emp_add: '102 plot shivajinagar',
  emp_age: '64',
  emp_mobno: '764346745'
},
{
  _id: ObjectId('68efd943dbb7ca4549cebeae'),
  emp_id: '3',
  emp_name: 'sayali',
  emp_add: '202 city apartment',
  emp_age: '45',
  emp_mobno: '568732866'
},
{
  _id: ObjectId('68efd943dbb7ca4549cebeaf'),
  emp_id: '4',
  emp_name: 'riya',
  emp_add: '404 main street',
  emp_age: '68',
  emp_mobno: '335657246'
}
employee>
```

```
employee> db.Emp.updateMany({emp_age:{$gt:60}},{$set:{status:"retired"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 0
}
employee>
```

db.emp_personal_details.find()

Practical No:-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

employee> db.emp_professional.createIndex({id:1});

```
employee> db.emp_professional.createIndex({id:1});
id_1
employee> db.emp_professional.find();
[
  {
    _id: ObjectId('686f80ad344d559809748a5f'),
    id: 101,
    name: 'mayuri',
    desig: 'eng',
    salary: 60000,
    incentive: 5000,
    work: 5
  },
  {
    _id: ObjectId('686f80ad344d559809748a60'),
    id: 102,
    name: 'Akshu',
    desig: 'doctor',
    salary: 40000,
    incentive: 5000,
    work: 6
  },
  {
    _id: ObjectId('686f82a5344d559809748a61'),
    id: 101,
    name: 'mayuri',
    desig: 'eng',
    salary: 60000,
    incentive: 5000,
    work: 5
  },
  {
    _id: ObjectId('686f82a5344d559809748a62'),
    id: 102,
    name: 'mayu',
    desig: 'doctor',
    salary: 70000,
    incentive: 5000,
    work: 8
  },
  {
    _id: ObjectId('686f82a5344d559809748a63'),
    id: 103,
    name: 'tejas',
    desig: 'HOD',
    salary: 70000,
    incentive: 5000,
    work: 5
  },
```

employee> db.emp_professional.createIndex({id:1,name:1});

```
employee> db.emp_professional.createIndex({id:1,name:1});
id_1_name_1
employee> db.emp_professional.find();
[
  {
    _id: ObjectId('686f80ad344d559809748a5f'),
    id: 101,
    name: 'mayuri',
    desig: 'eng',
    salary: 60000,
    incentive: 5000,
    work: 5
  },
  {
    _id: ObjectId('686f80ad344d559809748a60'),
    id: 102,
    name: 'Akshu',
    desig: 'doctor',
    salary: 40000,
    incentive: 5000,
    work: 6
  },
  {
    _id: ObjectId('686f82a5344d559809748a61'),
    id: 101,
    name: 'mayuri',
    desig: 'eng',
    salary: 60000,
    incentive: 5000,
    work: 5
  },
  {
    _id: ObjectId('686f82a5344d559809748a62'),
    id: 102,
    name: 'mayu',
    desig: 'doctor',
    salary: 70000,
    incentive: 5000,
    work: 8
  },
  {
    _id: ObjectId('686f82a5344d559809748a63'),
    id: 103,
    name: 'tejas',
    desig: 'HOD',
    salary: 70000,
    incentive: 5000,
    work: 5
  },
```

Practical No:- 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

7 a

```
employee> db.emp_personal1.insertOne({
... id:1,
... name:"mayuri",
... add: "pune" ,
... dob:new Date("2000-05-20"),
... age:66,
... mob_no:2345232221,
... mob_no:5678654686 })
{
  acknowledged: true,
  insertedId: ObjectId('688b3600f063a561c6748a60')
}
employee> db.emp_personal1.aggregate([
... { $unwind:"$mob_no"}
... ])
```

Output:

```
[
  {
    _id: ObjectId('688b3527f063a561c6748a5f'),
    id: 1,
    name: 'mayuri',
    add: 'pune',
    dob: ISODate('2000-05-20T00:00:00.000Z'),
    age: 66,
    mob_no: 2345232221
  },
  {
    _id: ObjectId('688b3600f063a561c6748a60'),
    id: 1,
    name: 'mayuri',
    add: 'pune',
    dob: ISODate('2000-05-20T00:00:00.000Z'),
    age: 66,
    mob_no: 5678654686
  }
]
employee>
```

7.b

```
employee> db.emp_personal1.aggregate([
... { $skip:3}
... ])
```

```
[
  {
    _id: ObjectId('6870cb30c6feebfd88748a62'),
    id: 4,
    name: 'minal ',
    add: 'patna ',
    DOB: ISODate('2009-05-21T18:30:00.000Z'),
    age: 25,
    mob: 2345678120
  },
  {
    _id: ObjectId('6870cb30c6feebfd88748a63'),
    id: 5,
    name: 'raj ',
    add: 'mumbai ',
    DOB: ISODate('2005-03-19T18:30:00.000Z'),
    age: 12,
    mob: 2345678121
  },
  {
    _id: ObjectId('6870cb30c6feebfd88748a64'),
    id: 6,
    name: 'aditya ',
    add: 'kolkata ',
    DOB: ISODate('2000-08-19T18:30:00.000Z'),
    age: 70,
    mob: 2345678122,
    status: 'retire'
```

7.c

```
employee> db.emp_personal1.aggregate([
... { $limit:4}
... ])
```

```
[
  {
    _id: ObjectId('6870cb30c6feebfd88748a5f'),
    id: 1,
    name: 'mayuri ',
    add: 'pune ',
    DOB: ISODate('2000-05-19T18:30:00.000Z'),
    age: 33,
    mob: 2345678122
  },
  {
    _id: ObjectId('6870cb30c6feebfd88748a60'),
    id: 2,
    name: 'mayu ',
    add: 'nashik ',
    DOB: ISODate('2010-04-19T18:30:00.000Z'),
    age: 60,
    mob: 2345678122
  },
  {
    _id: ObjectId('6870cb30c6feebfd88748a61'),
    id: 3,
    name: 'Akanksha ',
    add: 'thane ',
    DOB: ISODate('2022-04-30T18:30:00.000Z'),
    age: 65,
    mob: 2345678122,
    status: 'retire'
  },
  {
    _id: ObjectId('6870cb30c6feebfd88748a62'),
    id: 4,
    name: 'minal ',
    add: 'patna ',
    DOB: ISODate('2009-05-21T18:30:00.000Z'),
    age: 25,
    mob: 2345678120
  }
]
employee>
```

Practical no:-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 restaurant

switched to db restaurant

restaurant> db.createCollection("restaurant_details");

{ ok: 1 }

restaurant> db.restaurant_details.insertMany([

... {res_id:"1001",name:"Willaims
grill",borough:"Bronx",cuisine:"American",grades:[{grade:"A",score:92}]},

... {res_id:"1002",name:"Wilton
bbq",borough:"Queens",cuisine:"Italian",grades:[{grade:"B",score:88}]},

... {res_id:"1003",name:"Ces
delihts",borough:"Manhattan",cuisine:"Chinese",grades:[{grade:"C",score:78}]

... },

... {res_id:"1004",name:"Regal
taste",borough:"Brooklyn",cuisine:"Italian",grades:[{grade:"B",score:95}]},

... {res_id:"1005",name:"Tnadoori
flame",borough:"Bronx",cuisine:"Indian",grades:[{grade:"A",score:90}]},

... {res_id:"1006",name:"The Regal
diner",borough:"Queens",cuisine:"American",grades:[{grade:"A",score:85}]},

... {res_id:"1007",name:"Spicy hub",borough:"Staten
Island",cuisine:"Thai",grades:[{grade:"B",score:98}]},

... {res_id:"1008",name:"Desi
Express",borough:"Manhattan",cuisine:"Indian",grades:[{grade:"C",score:10}]}])
```

OutPut:-

```
mongosh mongodb://127.0.0.

   The server generated these startup warnings when booting
   2025-10-15T18:44:20.000+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
------

test> use restaurant
switched to db restaurant
restaurant> db.createCollection("restaurant_details");
{ ok: 1 }
restaurant> db.restaurant_details.insertMany([
... {res_id:"1001",name:"Willaims grill",borough:"Bronx",cuisine:"American",grades:[{grade:"A",score:92}]},
... {res_id:"1002",name:"Wilton bbq",borough:"Queens",cuisine:"Italian",grades:[{grade:"B",score:88}]},
... {res_id:"1003",name:"Ces delihts",borough:"Manhattan",cuisine:"Chinese",grades:[{grade:"C",score:78}]
... },
... {res_id:"1004",name:"Regal taste",borough:"Brooklyn",cuisine:"Italian",grades:[{grade:"B",score:95}]},
... {res_id:"1005",name:"Tnadoori flame",borough:"Bronx",cuisine:"Indian",grades:[{grade:"A",score:90}]},
... {res_id:"1006",name:"The Regal diner",borough:"Queens",cuisine:"American",grades:[{grade:"A",score:85}]},
... {res_id:"1007",name:"Spicy hub",borough:"Staten Island",cuisine:"Thai",grades:[{grade:"B",score:98}]},
... {res_id:"1008",name:"Desi Express",borough:"Manhattan",cuisine:"Indian",grades:[{grade:"C",score:10}]}])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('68efb7eadbb7ca4549cebea4'),
    '1': ObjectId('68efb7eadbb7ca4549cebea5'),
    '2': ObjectId('68efb7eadbb7ca4549cebea6'),
    '3': ObjectId('68efb7eadbb7ca4549cebea7'),
    '4': ObjectId('68efb7eadbb7ca4549cebea8'),
    '5': ObjectId('68efb7eadbb7ca4549cebea9'),
    '6': ObjectId('68efb7eadbb7ca4549cebeaa'),
    '7': ObjectId('68efb7eadbb7ca4549cebeab')
  }
}
restaurant>
```

```
mongosh mongodb://127.0.0.

taurant> db.restaurant_details.find();

  _id: ObjectId('68efb7eadbb7ca4549cebea4'),
  res_id: '1001',
  name: 'Willaims grill',
  borough: 'Bronx',
  cuisine: 'American',
  grades: [ { grade: 'A', score: 92 } ]
},
  _id: ObjectId('68efb7eadbb7ca4549cebea5'),
  res_id: '1002',
  name: 'Wilton bbq',
  borough: 'Queens',
  cuisine: 'Italian',
  grades: [ { grade: 'B', score: 88 } ]
},
  _id: ObjectId('68efb7eadbb7ca4549cebea6'),
  res_id: '1003',
  name: 'Ces delihts',
  borough: 'Manhattan',
  cuisine: 'Chinese',
  grades: [ { grade: 'C', score: 78 } ]
},
  _id: ObjectId('68efb7eadbb7ca4549cebea7'),
  res_id: '1004',
  name: 'Regal taste',
  borough: 'Brooklyn',
  cuisine: 'Italian',
  grades: [ { grade: 'B', score: 95 } ]
},
  _id: ObjectId('68efb7eadbb7ca4549cebea8'),
  res_id: '1005',
  name: 'Tnadoori flame',
  borough: 'Bronx',
  cuisine: 'Indian',
  grades: [ { grade: 'A', score: 90 } ]
},
  _id: ObjectId('68efb7eadbb7ca4549cebea9'),
  res_id: '1006',
  name: 'The Regal diner',
  borough: 'Queens',
  cuisine: 'American',
  grades: [ { grade: 'A', score: 85 } ]
},
  _id: ObjectId('68efb7eadbb7ca4549cebeaa'),
  res_id: '1007',
  name: 'Spicy hub',
  borough: 'Staten Island',
  cuisine: 'Thai',
  grades: [ { grade: 'B', score: 98 } ]
},
  _id: ObjectId('68efb7eadbb7ca4549cebeab'),
  res_id: '1008',
  name: 'Desi Express',
  borough: 'Manhattan',
  cuisine: 'Indian',
  grades: [ { grade: 'C', score: 10 } ]
]

taurant>
```

db.restaurant_details.find({},{res_id:1,name:1,borough:1,cuisine:1})

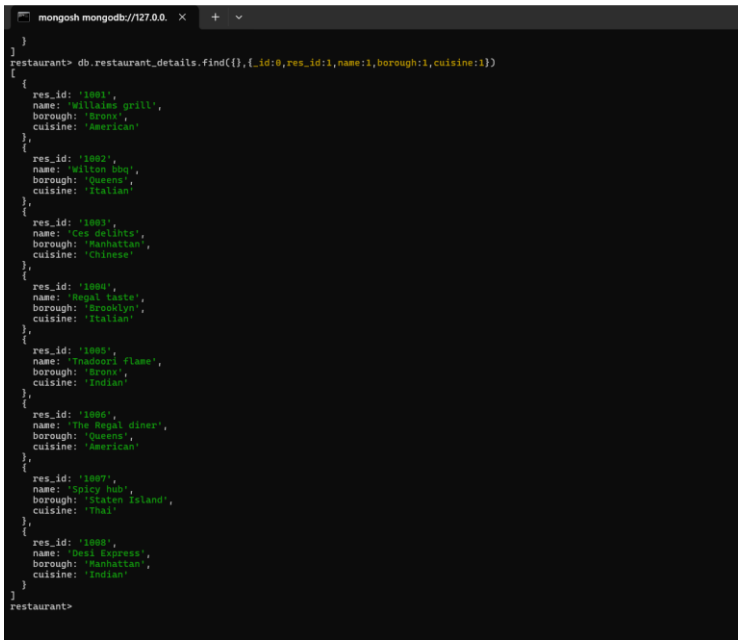
```
mongosh mongodb://127.0.0.

restaurant> db.restaurant_details.find({},{res_id:1,name:1,borough:1,cuisine:1})
[
  {
    _id: ObjectId('68efb7eadbb7ca4549cebea4'),
    res_id: '1001',
    name: 'Willaims grill',
    borough: 'Bronx',
    cuisine: 'American'
  },
  {
    _id: ObjectId('68efb7eadbb7ca4549cebea5'),
    res_id: '1002',
    name: 'Wilton bbq',
    borough: 'Queens',
    cuisine: 'Italian'
  },
  {
    _id: ObjectId('68efb7eadbb7ca4549cebea6'),
    res_id: '1003',
    name: 'Ces delihts',
    borough: 'Manhattan',
    cuisine: 'Chinese'
  },
  {
    _id: ObjectId('68efb7eadbb7ca4549cebea7'),
    res_id: '1004',
    name: 'Regal taste',
    borough: 'Brooklyn',
    cuisine: 'Italian'
  },
  {
    _id: ObjectId('68efb7eadbb7ca4549cebea8'),
    res_id: '1005',
    name: 'Tnadoori flame',
    borough: 'Bronx',
    cuisine: 'Indian'
  },
  {
    _id: ObjectId('68efb7eadbb7ca4549cebea9'),
    res_id: '1006',
    name: 'The Regal diner',
    borough: 'Queens',
    cuisine: 'American'
  },
  {
    _id: ObjectId('68efb7eadbb7ca4549cebeaa'),
    res_id: '1007',
    name: 'Spicy hub',
    borough: 'Staten Island',
    cuisine: 'Thai'
  },
  {
    _id: ObjectId('68efb7eadbb7ca4549cebeab'),
    res_id: '1008',
    name: 'Desi Express',
    borough: 'Manhattan',
    cuisine: 'Indian'
  }
]
restaurant>
```

Practical no:-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 restaurant>
db.restaurant_details.find({},{_id:0,res_id:1,name:1,borough:1,cuisine:1})



```
mongosh mongodb://127.0.0.
  }
]
restaurant> db.restaurant_details.find({},{_id:0,res_id:1,name:1,borough:1,cuisine:1})
[
  {
    res_id: '1001',
    name: 'Willaims grill',
    borough: 'Bronx',
    cuisine: 'American'
  },
  {
    res_id: '1002',
    name: 'Wilton bbq',
    borough: 'Queens',
    cuisine: 'Italian'
  },
  {
    res_id: '1003',
    name: 'Ces delihts',
    borough: 'Manhattan',
    cuisine: 'Chinese'
  },
  {
    res_id: '1004',
    name: 'Regal taste',
    borough: 'Brooklyn',
    cuisine: 'Italian'
  },
  {
    res_id: '1005',
    name: 'Tnadoori flame',
    borough: 'Bronx',
    cuisine: 'Indian'
  },
  {
    res_id: '1006',
    name: 'The Regal diner',
    borough: 'Queens',
    cuisine: 'American'
  },
  {
    res_id: '1007',
    name: 'Spicy hub',
    borough: 'Staten Island',
    cuisine: 'Thai'
  },
  {
    res_id: '1008',
    name: 'Desi Express',
    borough: 'Manhattan',
    cuisine: 'Indian'
  }
]
restaurant>
```

2 restaurant> db.restaurant_details.find({borough:"Bronx"})

```
}
restaurant> db.restaurant_details.find({borough:"Bronx"})
[
  {
    _id: ObjectId('68efb7eadbb7ca4549cebea4'),
    res_id: '1001',
    name: 'Willaims grill',
    borough: 'Bronx',
    cuisine: 'American',
    grades: [ { grade: 'A', score: 92 } ]
  },
  {
    _id: ObjectId('68efb7eadbb7ca4549cebea8'),
    res_id: '1005',
    name: 'Tnadoori flame',
    borough: 'Bronx',
    cuisine: 'Indian',
    grades: [ { grade: 'A', score: 90 } ]
  }
]
restaurant> |
```

Practical no:-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 test> use restaurant
switched to db restaurant
restaurant> db.restaurant_detail.insertMany([{
... building_number:"1007",
... street:"Morris Park Ave",
... zipcode:"10462",
... coordinates:[-73.856077, 40.848447],
... borough:"Bronx",
... cuisine:"Italian",
... grades:[
... {date:new Date("2023-01-20"),
... grade:"A",
... score:10
... },
... {date:new Date("2023-07-14"),
... grade:"B",
... score:15
```

```
... }
... ]
... },
... {
... building_number:"469",
... street:"Flatbush Ave",
... zipcode:"11225",
... coordinates:[-73.961704, 40.662942],
... borough:"Bronx",
... cuisine:"Chinese",
... grades:[
... {date:new Date("2022-12-05"),
... grade:"A",
... score:12
... },
... {
... date:new Date("2023-08-10"),
... grade:"C",
... score:22
... }
... ]
... },
... {
... building_number:"55",
... street:"Broadway",
... zipcode:"10006",
... coordinates:[-74.012083, 40.707496],
... borough:"Bronx",
... cuisine:"Mexican",
```

```
... grades:[
... {date:new Date("2023-03-18"),
... grade:"A",
... score:9
... },
... {date:new Date("2024-01-12"),
... grade:"A",
... score:11
... }
... ]
... },
... {
... building_number:"11",
... street:"Arthur Ave",
... zipcode:"10458",
... coordinates:[-73.880123, 40.855789],
... borough:"Bronx",
... cuisine:"Pizza",
... grades:[
... {date:new Date("2023-04-10"),
... grade:"B",
... score:18
... },
... {date:new Date("2023-09-12"),
... grade:"A",
... score:12
... }
... ]
... },
```

```
... {
... building_number:"250",
... street:"Grand Concourse",
... zipcode:"10451",
... coordinates:[-73.925312, 40.827632],
... borough:"Bronx",
... cuisine:"American",
... grades:[
... {date:new Date("2023-05-25"),
... grade:"C",
... score:25
... },
... {date:new Date("2023-11-19"),
... grade:"B",
... score:14
... }
... ]
... },
... {
... building_number:"98",
... street:"Third Ave",
... zipcode:"10455",
... coordinates:[-73.918332, 40.815623],
... borough:"Bronx",
... cuisine:"Indian",
... grades:[
... {date:new Date("2023-06-05"),
... grade:"A",
... score:9
```

```
... },
... {date:new Date("2024-01-18"),
... grade:"B",
... score:16
... }
... ]
... },
... {
... building_number:"210",
... street:"Fordham Rd",
... zipcode:"10468",
... coordinates:[-73.900562, 40.862633],
... borough:"Bronx",
... cuisine:"Thai",
... grades:[
... {date:new Date("2023-03-11"),
... grade:"B",
... score:17
... },
... {date:new Date("2023-10-22"),
... grade:"C",
... score:20
... }
... ]
... }
... ]);
```

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeou...
... grades:[
... {date:new Date("2023-03-11"),
... grade:"B",
... score:17
... },
... {date:new Date("2023-10-22"),
... grade:"C",
... score:20
... }
... ]
... }
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('68b70c3d0489648cc8eec4a9'),
    '1': ObjectId('68b70c3d0489648cc8eec4aa'),
    '2': ObjectId('68b70c3d0489648cc8eec4ab'),
    '3': ObjectId('68b70c3d0489648cc8eec4ac'),
    '4': ObjectId('68b70c3d0489648cc8eec4ad'),
    '5': ObjectId('68b70c3d0489648cc8eec4ae'),
    '6': ObjectId('68b70c3d0489648cc8eec4af')
  }
}
```

```
restaurant> db.restaurant_detail.find({borough:"Bronx"}).limit(5);
[
  {
    _id: ObjectId('68b70c3d0489648cc8eec4a9'),
    building_number: '1007',
    street: 'Morris Park Ave',
    zipcode: '10462',
    coordinates: [ -73.856077, 40.848447 ],
    borough: 'Bronx',
    cuisine: 'Italian',
    grades: [
      {
        date: ISODate('2023-01-20T00:00:00.000Z'),
        grade: 'A',
        score: 10
      },
      {
        date: ISODate('2023-07-14T00:00:00.000Z'),
        grade: 'B',
        score: 15
      }
    ]
  },
  {
    _id: ObjectId('68b70c3d0489648cc8eec4aa'),
    building_number: '469',
    street: 'Flatbush Ave',
restaurant>
    coordinates: [ -73.961704, 40.662942 ],
    borough: 'Bronx',
    cuisine: 'Chinese',
    grades: [
      {
        date: ISODate('2022-12-05T00:00:00.000Z'),
        grade: 'A',
        score: 12
      },
```

```
Select mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
restaurant> db.restaurant_detail.find({borough:"Bronx"}).skip(5).limit(5);
[
  {
    _id: ObjectId('68b70c3d0489648cc8eec4ae'),
    building_number: '98',
    street: 'Third Ave',
    zipcode: '10455',
    coordinates: [ -73.918332, 40.815623 ],
    borough: 'Bronx',
    cuisine: 'Indian',
    grades: [
      {
        date: ISODate('2023-06-05T00:00:00.000Z'),
        grade: 'A',
        score: 9
      },
      {
        date: ISODate('2024-01-18T00:00:00.000Z'),
        grade: 'B',
        score: 16
      }
    ]
  },
  {
    _id: ObjectId('68b70c3d0489648cc8eec4af'),
    building_number: '210',
    street: 'Fordham Rd',
    zipcode: '10468',
    coordinates: [ -73.900562, 40.862633 ],
    borough: 'Bronx',
    cuisine: 'Thai',
    grades: [
      {
        date: ISODate('2023-03-11T00:00:00.000Z'),
        grade: 'B',
        score: 17
      },
      {
        date: ISODate('2023-10-22T00:00:00.000Z'),
        grade: 'C',
        score: 20
      }
    ]
```

Practical no:-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 restaurant> db.restaurant_details.find({"grades.score":{$gt:90}})

```
restaurant> db.restaurant_details.find({"grades.score":{$gt:90}})
[
  {
    _id: ObjectId('68efb7eadbb7ca4549cebea4'),
    res_id: '1001',
    name: 'Willaims grill',
    borough: 'Bronx',
    cuisine: 'American',
    grades: [ { grade: 'A', score: 92 } ]
  },
  {
    _id: ObjectId('68efb7eadbb7ca4549cebea7'),
    res_id: '1004',
    name: 'Regal taste',
    borough: 'Brooklyn',
    cuisine: 'Italian',
    grades: [ { grade: 'B', score: 95 } ]
  },
  {
    _id: ObjectId('68efb7eadbb7ca4549cebeaa'),
    res_id: '1007',
    name: 'Spicy hub',
    borough: 'Staten Island',
    cuisine: 'Thai',
    grades: [ { grade: 'B', score: 98 } ]
  }
]
restaurant>
```

2 restaurant> db.restaurant_details.find({"grades.score":{$gt:80,$lt:100}})

```
mongosh mongodb://127.0.0.

restaurant> db.restaurant_details.find({"grades.score":{$gt:80,$lt:100}})
[
  {
    _id: ObjectId('68efb7eadbb7ca4549cebea4'),
    res_id: '1001',
    name: 'Willaims grill',
    borough: 'Bronx',
    cuisine: 'American',
    grades: [ { grade: 'A', score: 92 } ]
  },
  {
    _id: ObjectId('68efb7eadbb7ca4549cebea5'),
    res_id: '1002',
    name: 'Wilton bbq',
    borough: 'Queens',
    cuisine: 'Italian',
    grades: [ { grade: 'B', score: 88 } ]
  },
  {
    _id: ObjectId('68efb7eadbb7ca4549cebea7'),
    res_id: '1004',
    name: 'Regal taste',
    borough: 'Brooklyn',
    cuisine: 'Italian',
    grades: [ { grade: 'B', score: 95 } ]
  },
  {
    _id: ObjectId('68efb7eadbb7ca4549cebea8'),
    res_id: '1005',
    name: 'Tnadoori flame',
    borough: 'Bronx',
    cuisine: 'Indian',
    grades: [ { grade: 'A', score: 90 } ]
  },
  {
    _id: ObjectId('68efb7eadbb7ca4549cebea9'),
    res_id: '1006',
    name: 'The Regal diner',
    borough: 'Queens',
    cuisine: 'American',
    grades: [ { grade: 'A', score: 85 } ]
  },
  {
    _id: ObjectId('68efb7eadbb7ca4549cebeaa'),
    res_id: '1007',
    name: 'Spicy hub',
    borough: 'Staten Island',
    cuisine: 'Thai',
    grades: [ { grade: 'B', score: 98 } ]
  }
]
restaurant>
```

Practical no:-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

borough Brooklyn. The document must be displayed according to the cuisine in

descending order

restaurant>
db.restaurant_details.find({cuisine:{$ne:"American"},"grades.grade":"A",borouugh:{$ne:"Brooklyn"}}).sort({cuisine:-1})

```
]
restaurant> db.restaurant_details.find({cuisine:{$ne:"American"},"grades.grade":"A",borouugh:{$ne:"Brooklyn"}}).sort({cuisine:-1})
[
  {
    _id: ObjectId('68efb7eadbb7ca4549cebea8'),
    res_id: '1005',
    name: 'Tnadoori flame',
    borough: 'Bronx',
    cuisine: 'Indian',
    grades: [ { grade: 'A', score: 90 } ]
  }
]
restaurant>
```

Practical no:-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>
db.restaurant_details.find({name:{$regex:/^Wil/}},{_id:0,res_id:1,name:1,borough:1,cuisine:1})

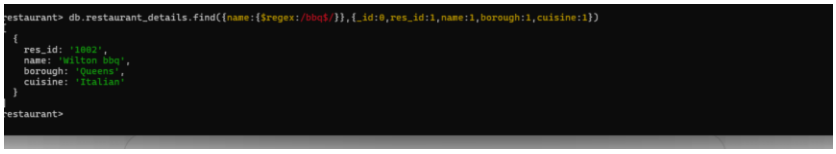
```
]
restaurant> db.restaurant_details.find({name:{$regex:/^Wil/}},{_id:0,res_id:1,name:1,borough:1,cuisine:1})
[
  {
    res_id: '1001',
    name: 'Willaims grill',
    borough: 'Bronx',
    cuisine: 'American'
  },
  {
    res_id: '1002',
    name: 'Wilton bbq',
    borough: 'Queens',
    cuisine: 'Italian'
  }
]
restaurant>
```

Practical no:-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>
db.restaurant_details.find({name:{$regex:/bbq$/}},{_id:0,res_id:1,name:1,borough:1,cuisine:1})



Practical no:-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>
db.restaurant_details.find({name:{$regex:/Reg/}},{_id:0,res_id:1,name:1,borough:1,cuisine:1})

Output:-

```
restaurant> db.restaurant_details.find({name:{$regex:/Reg/}},{_id:0,res_id:1,name:1,borough:1,cuisine:1})
[
  {
    res_id: '1004',
    name: 'Regal taste',
    borough: 'Brooklyn',
    cuisine: 'Italian'
  },
  {
    res_id: '1006',
    name: 'The Regal diner',
    borough: 'Queens',
    cuisine: 'American'
  }
]
restaurant>
```

Practical no:-18

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

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

Output:-

```
]
restaurant> db.restaurant_details.find({borough:"Bronx",cuisine:{$in:["American","Chinese"]}})
[
  {
    _id: ObjectId('68efb7eadbb7ca4549cebea4'),
    res_id: '1001',
    name: 'Willaims grill',
    borough: 'Bronx',
    cuisine: 'American',
    grades: [ { grade: 'A', score: 92 } ]
  }
]
restaurant>
```

Practical no:-19

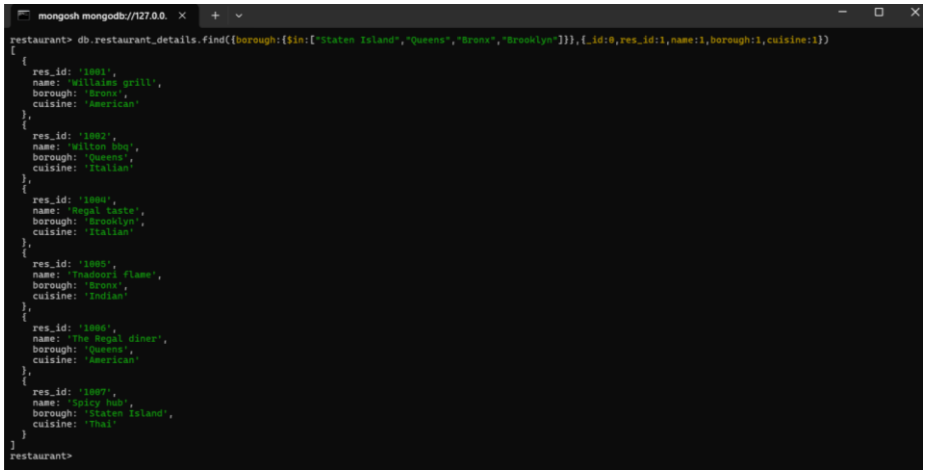
Write a MongoDB query to find the restaurant Id,name, borough and cuisine for

those restaurants which belong to the borough Staten Island or Queens or Bronx

or Brooklyn

restaurant> db.restaurant_details.find({borough:{$in:["Staten Island","Queens","Bronx","Brooklyn"]}},{_id:0,res_id:1,name:1,borough:1,cuisine:1})

Output:-



```
mongosh mongodb://127.0.0.
restaurant> db.restaurant_details.find({borough:{$in:["Staten Island","Queens","Bronx","Brooklyn"]}},{_id:0,res_id:1,name:1,borough:1,cuisine:1})
[
  {
    res_id: '1001',
    name: 'Willaims grill',
    borough: 'Bronx',
    cuisine: 'American'
  },
  {
    res_id: '1002',
    name: 'Wilton bbq',
    borough: 'Queens',
    cuisine: 'Italian'
  },
  {
    res_id: '1004',
    name: 'Regal taste',
    borough: 'Brooklyn',
    cuisine: 'Italian'
  },
  {
    res_id: '1005',
    name: 'Tandoori flame',
    borough: 'Bronx',
    cuisine: 'Indian'
  },
  {
    res_id: '1006',
    name: 'The Regal diner',
    borough: 'Queens',
    cuisine: 'American'
  },
  {
    res_id: '1007',
    name: 'Spicy hub',
    borough: 'Staten Island',
    cuisine: 'Thai'
  }
]
restaurant>
```

Practical no:-20

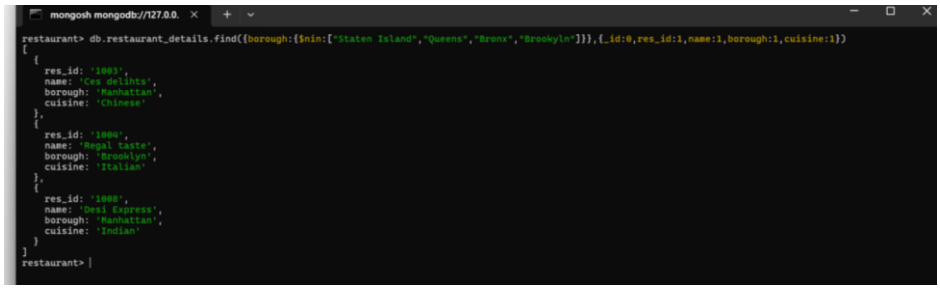
Write a MongoDB query to find the restaurant Id,name, borough and cuisine for

those restaurants which are not belonging to the borough Staten Island Or Queens

or Bronxor Brooklyn.

```
restaurant> db.restaurant_details.find({borough:{$nin:["Staten Island","Queens","Bronx","Brookyln"]}},{_id:0,res_id:1,name:1,borough:1,cuisine:1})
```

Output:-



Practical no:-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>
db.restaurant_details.find({"grades.score":{$lte:10}},{_id:0,res_id:1,name:1,borough:1,cuisine:1})

Output:-

```
restaurant> db.restaurant_details.find({"grades.score":{$lte:10}},{_id:0,res_id:1,name:1,borough:1,cuisine:1})
[
  {
    res_id: '1008',
    name: 'Desi Express',
    borough: 'Manhattan',
    cuisine: 'Indian'
  }
]
restaurant>
```

Practical no:-22

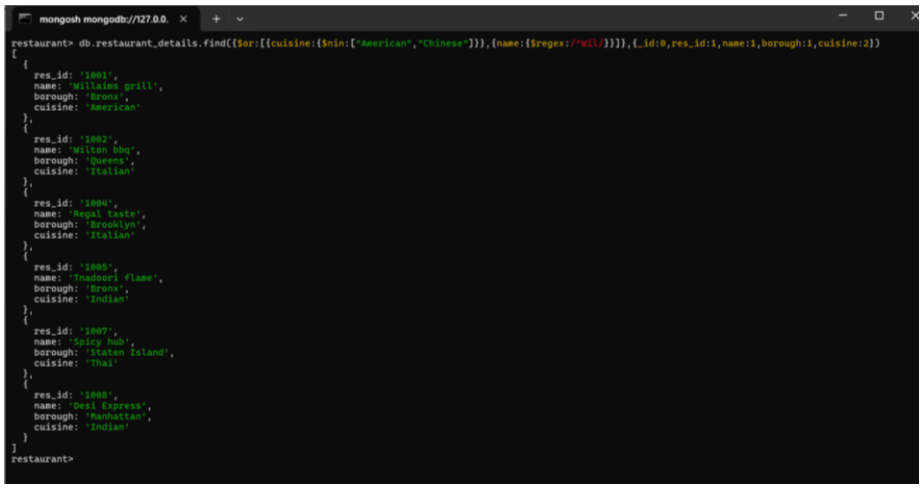
Write a MongoDB query to find the restaurant Id,name, borough and cuisine for

those restaurants which prepared dish except 'American' and 'Chinese' or

restaurant's name begins with letter 'Wil'

restaurant>
db.restaurant_details.find({$or:[{cuisine:{$nin:["American","Chinese"]}},{name:{$regex:/^Wil/}}]},{_id:0,res_id:1,name:1,borough:1,cuisine:2})

Output:-



Practical no:-23

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

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

Output:-

```
mongosh mongodb://127.0.0.
}
restaurant> db.restaurant_details.find().sort({name:-1})
[
  {
    _id: ObjectId('68efb7eadbb7ca4549cebea5'),
    res_id: '1002',
    name: 'Wilton bbq',
    borough: 'Queens',
    cuisine: 'Italian',
    grades: [ { grade: 'B', score: 88 } ]
  },
  {
    _id: ObjectId('68efb7eadbb7ca4549cebea4'),
    res_id: '1001',
    name: 'Williams grill',
    borough: 'Bronx',
    cuisine: 'American',
    grades: [ { grade: 'A', score: 92 } ]
  },
  {
    _id: ObjectId('68efb7eadbb7ca4549cebea8'),
    res_id: '1005',
    name: 'Tandoori flame',
    borough: 'Bronx',
    cuisine: 'Indian',
    grades: [ { grade: 'A', score: 90 } ]
  },
  {
    _id: ObjectId('68efb7eadbb7ca4549cebea9'),
    res_id: '1006',
    name: 'The Bagel diner',
    borough: 'Queens',
    cuisine: 'American',
    grades: [ { grade: 'A', score: 85 } ]
  },
  {
    _id: ObjectId('68efb7eadbb7ca4549cebeaa'),
    res_id: '1007',
    name: 'Spicy Hub',
    borough: 'Staten Island',
    cuisine: 'Thai',
    grades: [ { grade: 'B', score: 98 } ]
  },
  {
    _id: ObjectId('68efb7eadbb7ca4549cebea7'),
    res_id: '1004',
    name: 'Regal taste',
    borough: 'Brooklyn',
    cuisine: 'Italian',
    grades: [ { grade: 'B', score: 95 } ]
  },
  {
    _id: ObjectId('68efb7eadbb7ca4549cebeab'),
    res_id: '1008',
    name: 'Desi express',
    borough: 'Manhattan',
    cuisine: 'Indian',
    grades: [ { grade: 'C', score: 10 } ]
  },
  {
    _id: ObjectId('68efb7eadbb7ca4549cebea6'),
    res_id: '1003',
    name: 'Cee dulhte',
    borough: 'Manhattan',
    cuisine: 'Chinese',
    grades: [ { grade: 'C', score: 78 } ]
  }
]
```

Practical no:-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> db.restaurant_details.find().sort({cuisine:1,borough:-1})
```

Output:-

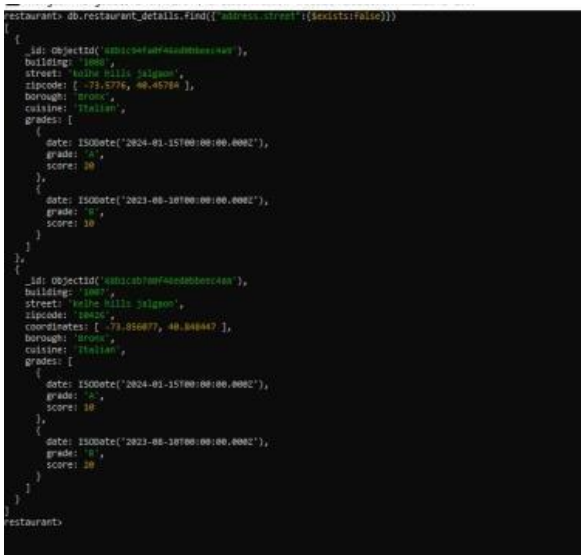
Practical no:-25

Write a MongoDB query to know whether all the addresses contains the street or

No

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

Output:-

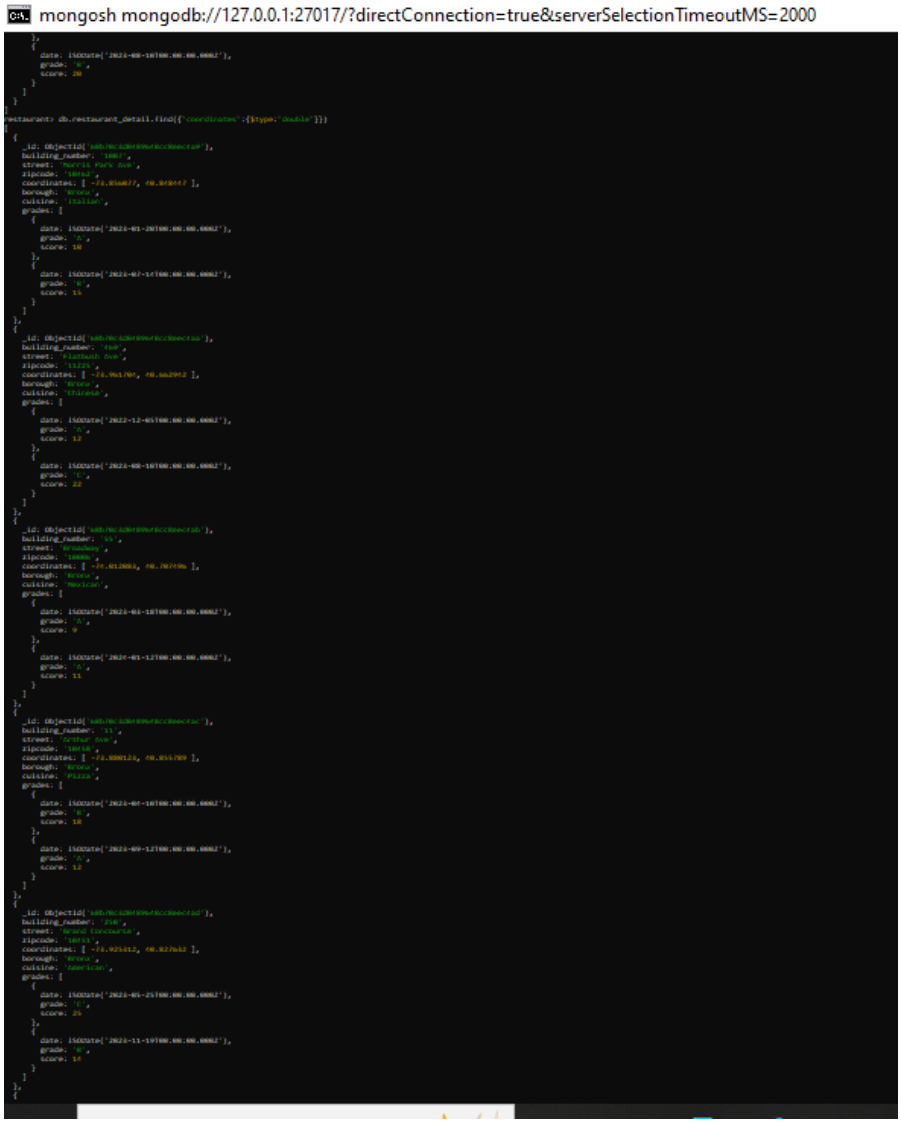


Practical no:-26

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

db.restaurant_detail.find({"coordinates":{$type:"double"}})

Output:-



Practical no:-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>
db.restaurant_details.find({"grades.score":{$mod:[7,0]}},{_id:0,res_id:1,name:1,grades:1})

Output:-

```
restaurant> db.restaurant_details.find({"grades.score":{$mod:[7,0]}},{_id:0,res_id:1,name:1,grades:1})
[
  {
    res_id: '1007',
    name: 'Spicy hub',
    grades: [ { grade: 'B', score: 98 } ]
  }
]
restaurant>
```

Practical no:-28

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

db.restaurant_detail.find({street:{$regex:"Mor",$options:"i"}},{street:1,borough:1,"coordinates":1,cuisine:1,_id:0})

Output:-

```
restaurant> db.restaurant_detail.find({street:{$regex:"Mor",$options:"i"}},{street:1,borough:1,"coordinates":1,cuisine:1,_id:0})
[
  {
    street: 'Morris Park Ave',
    coordinates: [ -73.856077, 40.848447 ],
    borough: 'Bronx',
    cuisine: 'Italian'
  }
]
restaurant>
```

Practical no:-29

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

```
db.createCollection("emp1");
db.emp1.insertMany([{
name: "Akanksha",
department: "Sceince",
salary: 24000},
{
name: "Aparna",
department: "Commerce",
salary: 54000},
{
name: "Puja",
department: "Sceince",
salary: 24000},
{
name: "Akanksha",
department: "Arts",
salary: 24000}
])
{
name: "Akanks
 db.emp1.find();
db.emp1.aggregate([
{
$group:{
 _id:null,
 totalsalary:{$sum:"$salary"}
}
}
]
```

Output:

```
emp> db.emp1.aggregate([
... {
... $group:{
... _id:null,
... totalsalary:{$sum:"$salary"}
... }
... }
... ])
[ { _id: null, totalsalary: 126000 } ]
emp>
```

db.emp1.find();
db.emp1.aggregate([

{

$group:{

 _id:null,

 totalsalary:{$min:"$salary"}

}

}

])

Practical No. 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"
```

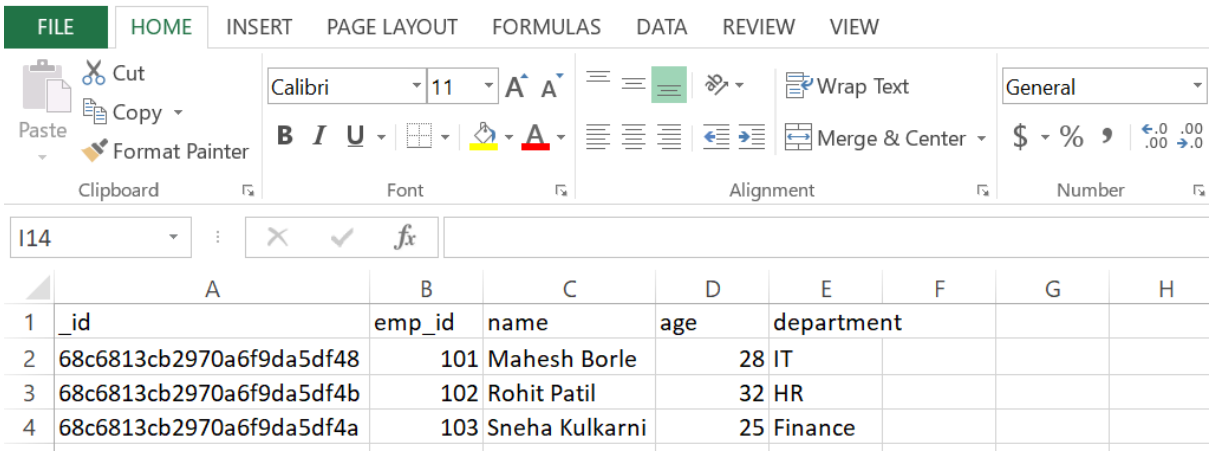
}
{
  "_id": {
    "$oid": "68c6813cb2970a6f9da5df4a"
  },
  "emp_id": 103,
  "name": "Sneha Kulkarni",
  "age": 25,
  "department": "Finance"
}
Emp_professional_details:
{
  "_id": {
    "$oid": "68c6814fb2970a6f9da5df4d"
  },
  "emp_id": 101,
  "designation": "Software Engineer",
  "salary": 50000,
  "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



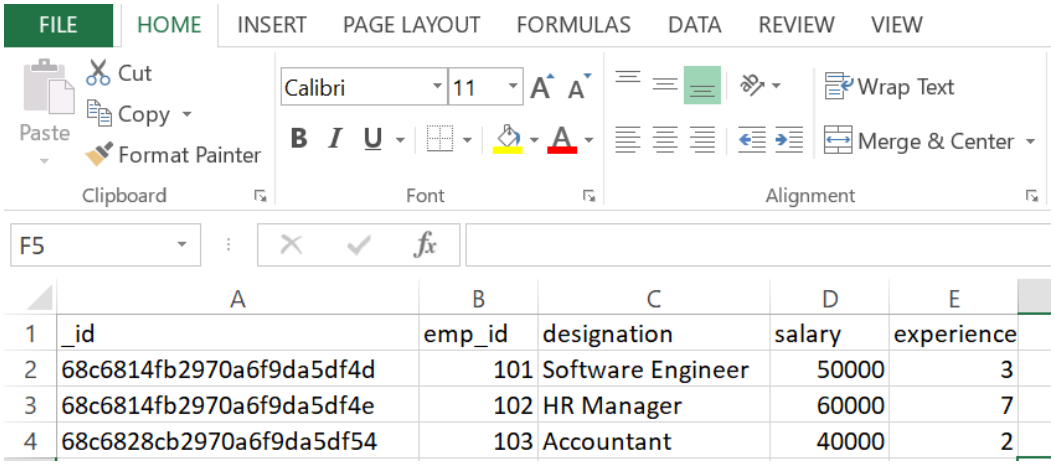
_id	emp_id	name	age	department
68c6813cb2970a6f9da5df48	101	Mahesh Borle	28	IT
68c6813cb2970a6f9da5df4b	102	Rohit Patil	32	HR
68c6813cb2970a6f9da5df4a	103	Sneha Kulkarni	25	Finance

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



	A	B	C	D	E
1	_id	emp_id	designation	salary	experience
2	68c6814fb2970a6f9da5df4d	101	Software Engineer	50000	3
3	68c6814fb2970a6f9da5df4e	102	HR Manager	60000	7
4	68c6828cb2970a6f9da5df54	103	Accountant	40000	2