NOSQL Databases

02-01-2026

Getting Started with MongoDB:

https://www.w3schools.com/mongodb/

https://www.mongodb.com/docs/manual/

MongoDB is a document database. It stores data in a type of JSON format called BSON.

Understanding JSON(**J**ava**S**cript **O**bject **N**otation):

JSON is a **plain text format** for storing and transporting data.

JSON is similar to the syntax for creating JavaScript objects.

JSON is used to **send**, **receive** and **store data**.

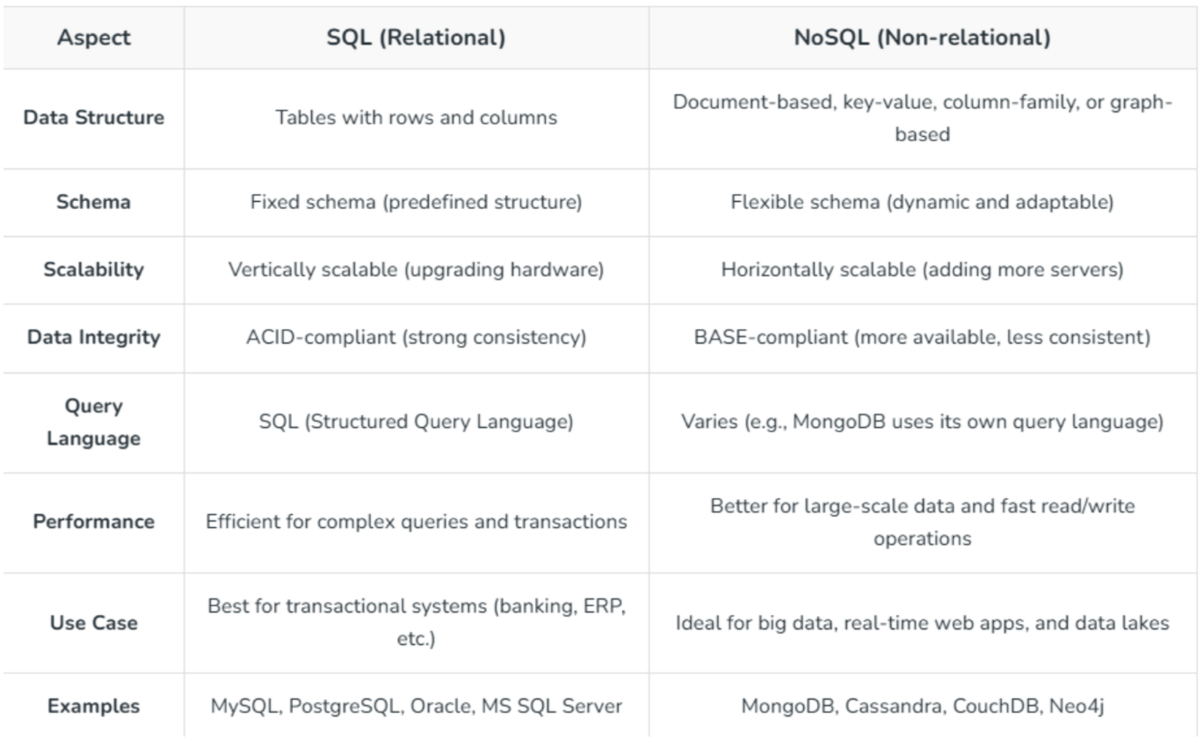
Object------JSON.stringify()----->String

String ------JSON.parse()----->Object

Data of objects accessed using dot(.) or []

In JSON, values must be one of the following data types:

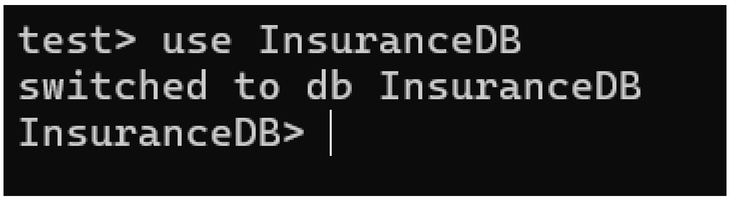
String ,number ,object (JSON object) ,array ,Boolean ,null



Aspect	SQL (Relational)	NoSQL (Non-relational)
Data Structure	Tables with rows and columns	Document-based, key-value, column-family, or graph-based
Schema	Fixed schema (predefined structure)	Flexible schema (dynamic and adaptable)
Scalability	Vertically scalable (upgrading hardware)	Horizontally scalable (adding more servers)
Data Integrity	ACID-compliant (strong consistency)	BASE-compliant (more available, less consistent)
Query Language	SQL (Structured Query Language)	Varies (e.g., MongoDB uses its own query language)
Performance	Efficient for complex queries and transactions	Better for large-scale data and fast read/write operations
Use Case	Best for transactional systems (banking, ERP, etc.)	Ideal for big data, real-time web apps, and data lakes
Examples	MySQL, PostgreSQL, Oracle, MS SQL Server	MongoDB, Cassandra, CouchDB, Neo4j

Let us consider of case of insurance database

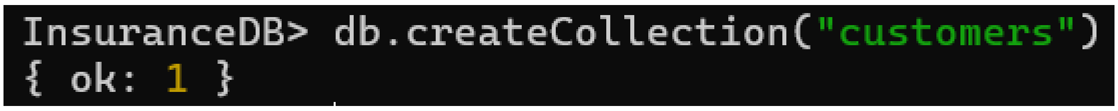
--Creating and using the document database:



```
test> use InsuranceDB
switched to db InsuranceDB
InsuranceDB>
```

--**Create Collection**

db.createCollection("customers")



```
InsuranceDB> db.createCollection("customers")
{ ok: 1 }
```

--Insert documents

Insert one:

db.customers.insertOne({

firstName: "Abhi",

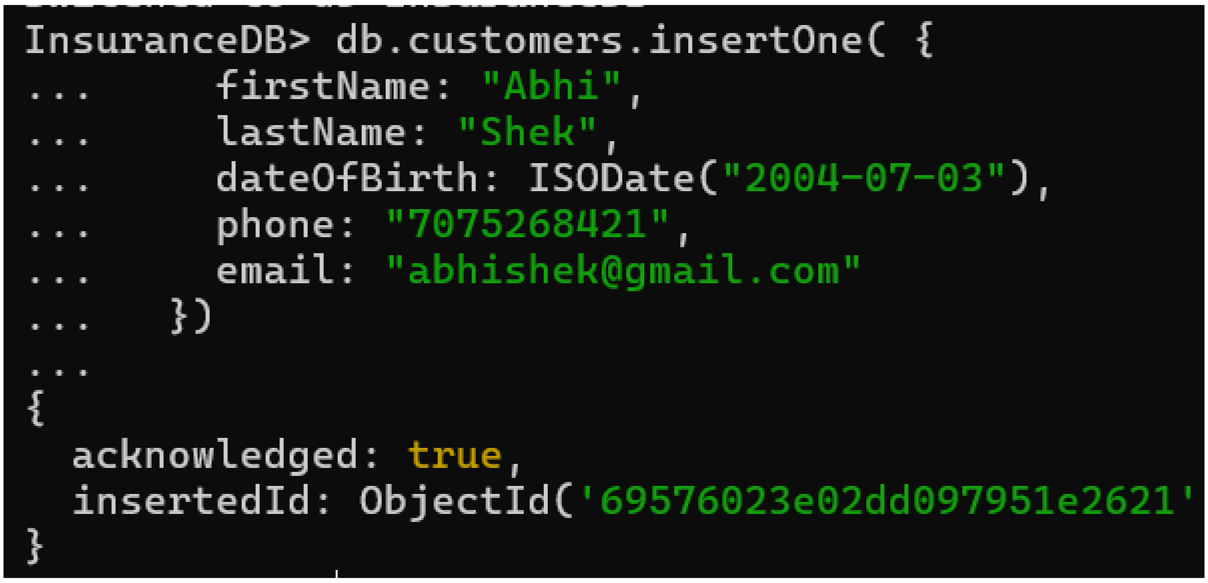
lastName: "Shek",

dateOfBirth: ISODate("2004-07-03"),

phone: "7075268421",

email: "abhishek@gmail.com"

})



```
InsuranceDB> db.customers.insertOne( {
...     firstName: "Abhi",
...     lastName: "Shek",
...     dateOfBirth: ISODate("2004-07-03"),
...     phone: "7075268421",
...     email: "abhishek@gmail.com"
...   })
...
{
  acknowledged: true,
  insertedId: ObjectId('69576023e02dd097951e2621'
}
```

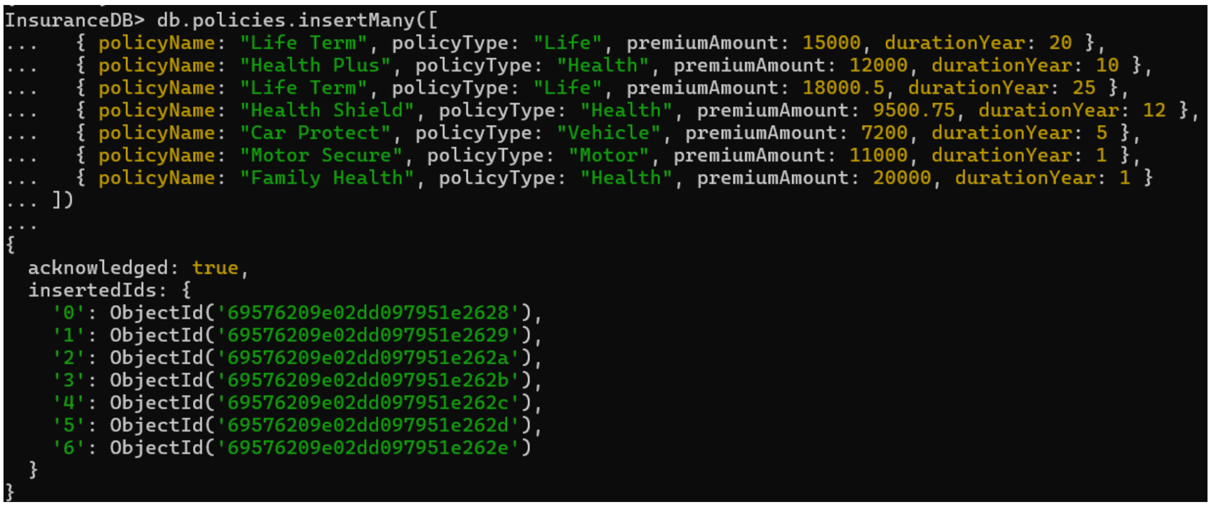
Insert Many:

```
db.customers.insertMany([
{
firstName: "Santhu",
lastName: "Chepuri",
dateOfBirth: ISODate("2004-05-24"),
phone: "9080706050",
email: "chepuri@gmail.com"
},
{
firstName: "Varun",
lastName: "Kumar",
dateOfBirth: ISODate("1999-08-12"),
phone: "9001112233",
email: "varun@gmail.com"
},
{
firstName: "Bhanu",
lastName: "Prakash",
dateOfBirth: ISODate("1998-03-25"),
phone: "9002223344",
email: "bhanu@gmail.com"
},
{
firstName: "Ramu",
lastName: "Reddy",
dateOfBirth: ISODate("2000-12-05"),
phone: "9003334455",
email: "ramu@gmail.com"
},
{
firstName: "Sangu",
lastName: "Rao",
dateOfBirth: ISODate("2002-02-10"),
phone: "9876543210",
email: "sangu@gmail.com"
},
{
firstName: "Manu",
lastName: "Pal",
dateOfBirth: ISODate("1998-11-15"),
phone: "9876543211",
email: "manu@gmail.com"
}
])
```

Creation of Policies Collection:

```
db.createCollection("policies")
```

Insertion of data into collection

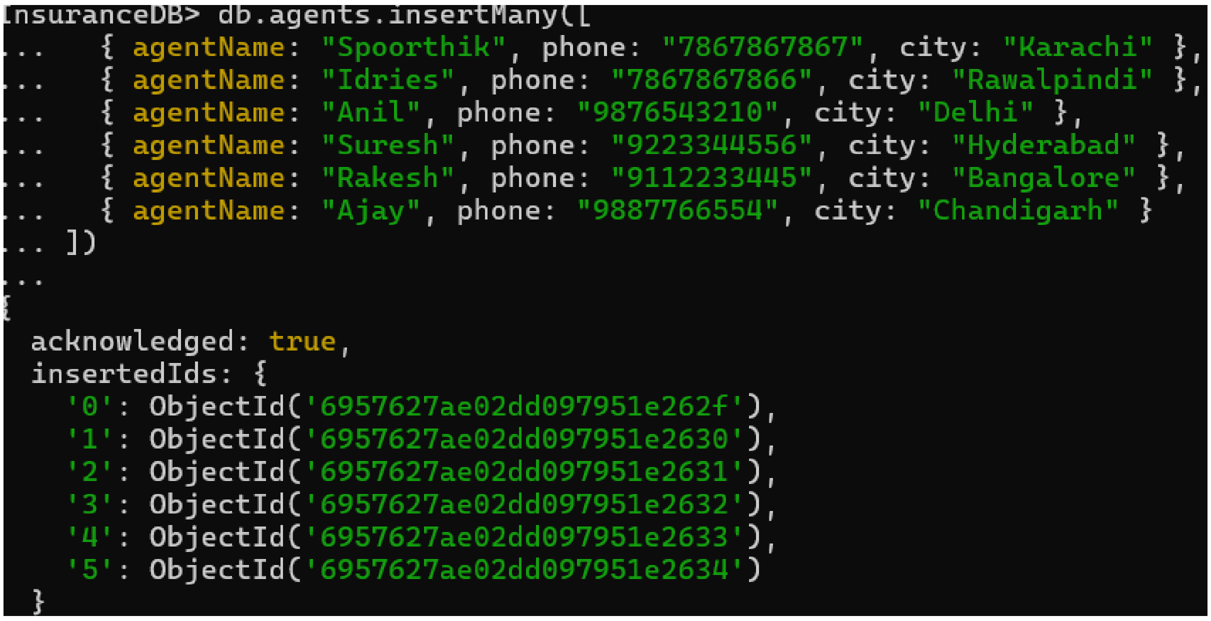


```
InsuranceDB> db.policies.insertMany([
...   { policyName: "Life Term", policyType: "Life", premiumAmount: 15000, durationYear: 20 },
...   { policyName: "Health Plus", policyType: "Health", premiumAmount: 12000, durationYear: 10 },
...   { policyName: "Life Term", policyType: "Life", premiumAmount: 18000.5, durationYear: 25 },
...   { policyName: "Health Shield", policyType: "Health", premiumAmount: 9500.75, durationYear: 12 },
...   { policyName: "Car Protect", policyType: "Vehicle", premiumAmount: 7200, durationYear: 5 },
...   { policyName: "Motor Secure", policyType: "Motor", premiumAmount: 11000, durationYear: 1 },
...   { policyName: "Family Health", policyType: "Health", premiumAmount: 20000, durationYear: 1 }
... ])
...
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('69576209e02dd097951e2628'),
    '1': ObjectId('69576209e02dd097951e2629'),
    '2': ObjectId('69576209e02dd097951e262a'),
    '3': ObjectId('69576209e02dd097951e262b'),
    '4': ObjectId('69576209e02dd097951e262c'),
    '5': ObjectId('69576209e02dd097951e262d'),
    '6': ObjectId('69576209e02dd097951e262e')
  }
}
```

Creation of agents collection:

db.createCollection("agents")

Insertion:

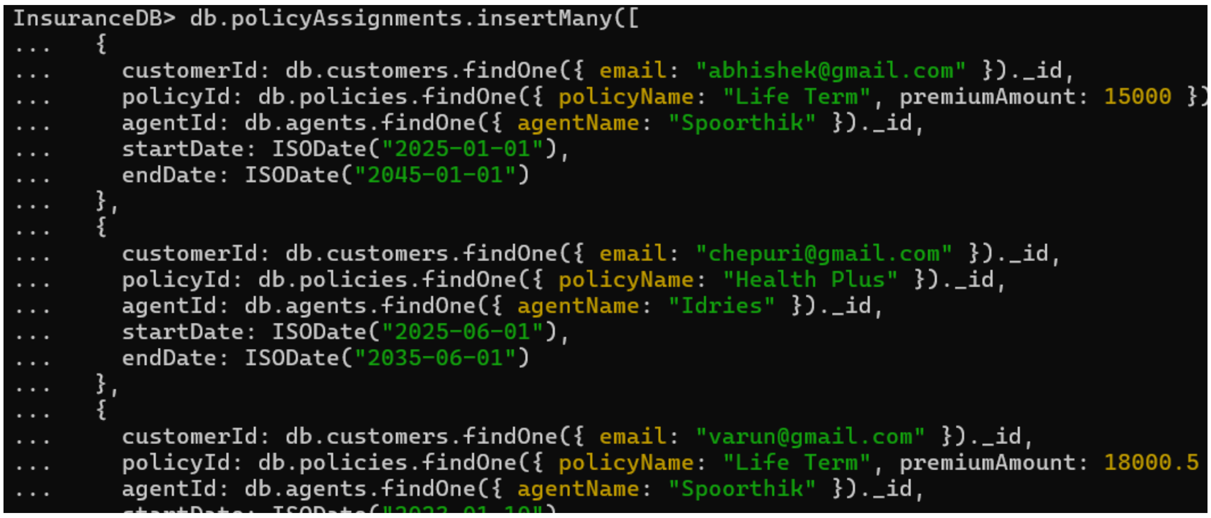


```
InsuranceDB> db.agents.insertMany([
...   { agentName: "Spoorthik", phone: "7867867867", city: "Karachi" },
...   { agentName: "Idries", phone: "7867867866", city: "Rawalpindi" },
...   { agentName: "Anil", phone: "9876543210", city: "Delhi" },
...   { agentName: "Suresh", phone: "9223344556", city: "Hyderabad" },
...   { agentName: "Rakesh", phone: "9112233445", city: "Bangalore" },
...   { agentName: "Ajay", phone: "9887766554", city: "Chandigarh" }
... ])
...
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('6957627ae02dd097951e262f'),
    '1': ObjectId('6957627ae02dd097951e2630'),
    '2': ObjectId('6957627ae02dd097951e2631'),
    '3': ObjectId('6957627ae02dd097951e2632'),
    '4': ObjectId('6957627ae02dd097951e2633'),
    '5': ObjectId('6957627ae02dd097951e2634')
  }
```

Creation of policyAssignments collection:

db.createCollection(“policyAssignments”)

Insertion:



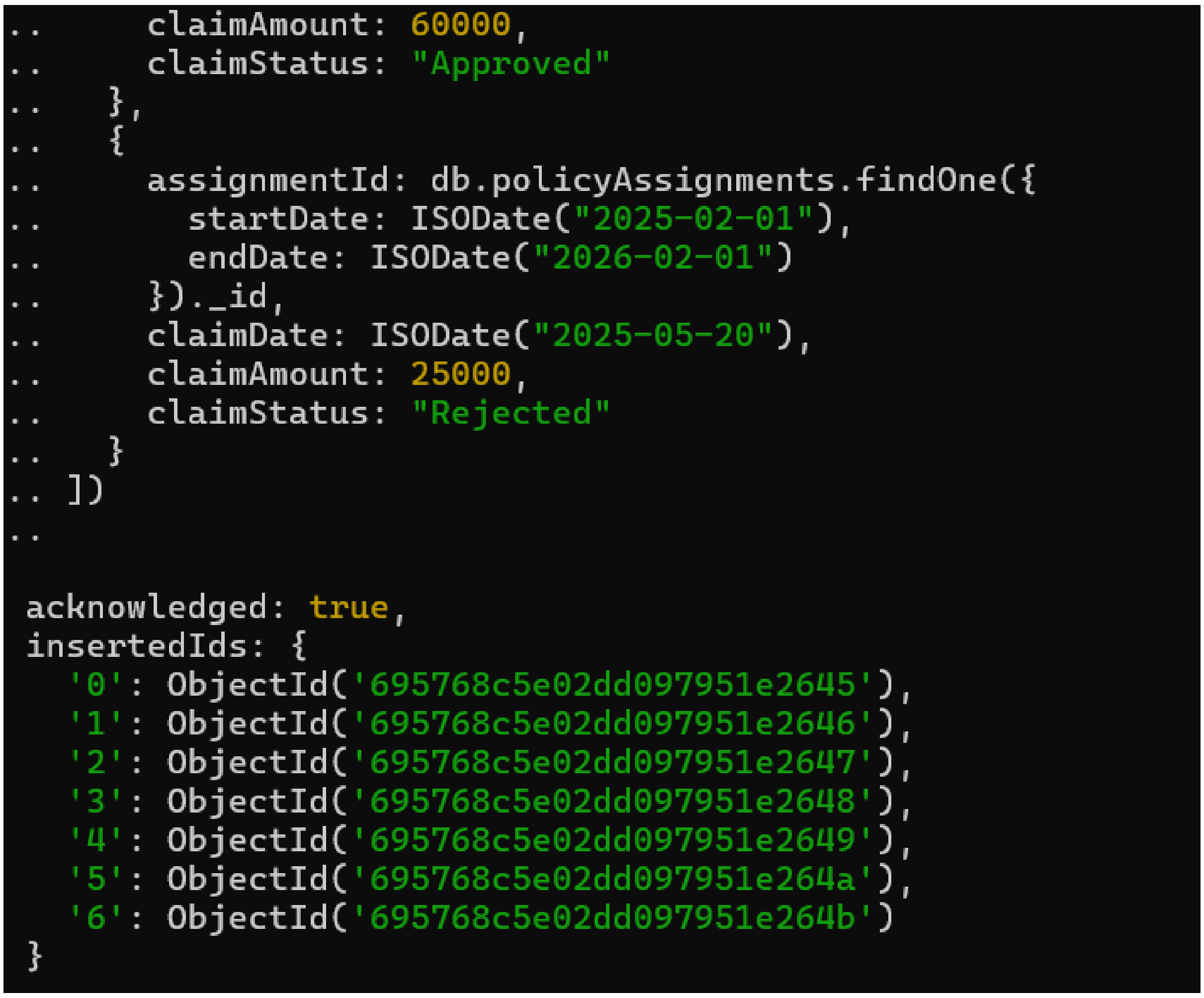
```
InsuranceDB> db.policyAssignments.insertMany([
...   {
...     customerId: db.customers.findOne({ email: "abhishek@gmail.com" })._id,
...     policyId: db.policies.findOne({ policyName: "Life Term", premiumAmount: 15000 })
...     agentId: db.agents.findOne({ agentName: "Spoorthik" })._id,
...     startDate: ISODate("2025-01-01"),
...     endDate: ISODate("2045-01-01")
...   },
...   {
...     customerId: db.customers.findOne({ email: "chepuri@gmail.com" })._id,
...     policyId: db.policies.findOne({ policyName: "Health Plus" })._id,
...     agentId: db.agents.findOne({ agentName: "Idries" })._id,
...     startDate: ISODate("2025-06-01"),
...     endDate: ISODate("2035-06-01")
...   },
...   {
...     customerId: db.customers.findOne({ email: "varun@gmail.com" })._id,
...     policyId: db.policies.findOne({ policyName: "Life Term", premiumAmount: 18000.5
...     agentId: db.agents.findOne({ agentName: "Spoorthik" })._id,
```

Creation of claims collection:

db.createCollection("claims")

Insertion:

```
db.claims.insertMany([
 {
   assignmentId: db.policyAssignments.findOne({
     startDate: ISODate("2025-01-01"),
     endDate: ISODate("2045-01-01")
   })._id,
   claimDate: ISODate("2026-02-15"),
   claimAmount: 50000,
   claimStatus: "Approved"
 },
 {
   assignmentId: db.policyAssignments.findOne({
     startDate: ISODate("2025-06-01"),
     endDate: ISODate("2035-06-01")
   })._id,
   claimDate: ISODate("2026-03-10"),
   claimAmount: 30000,
   claimStatus: "Pending"
 },
 {
   assignmentId: db.policyAssignments.findOne({
     startDate: ISODate("2023-01-10"),
     endDate: ISODate("2048-01-10")
   })._id,
   claimDate: ISODate("2024-11-20"),
   claimAmount: 45000,
   claimStatus: "Approved"
 },
 {
   assignmentId: db.policyAssignments.findOne({
     startDate: ISODate("2024-06-15"),
     endDate: ISODate("2036-06-15")
   })._id,
   claimDate: ISODate("2025-07-05"),
   claimAmount: 15000,
   claimStatus: "Rejected"
 },
 {
   assignmentId: db.policyAssignments.findOne({
     startDate: ISODate("2025-03-01"),
     endDate: ISODate("2030-03-01")
   })._id,
   claimDate: ISODate("2025-09-18"),
   claimAmount: 8000,
   claimStatus: "Pending"
  },
  {
    assignmentId: db.policyAssignments.findOne({
      startDate: ISODate("2024-01-01"),
      endDate: ISODate("2025-01-01")
    })._id,
    claimDate: ISODate("2025-03-15"),
    claimAmount: 60000,
    claimStatus: "Approved"
  },
  {
    assignmentId: db.policyAssignments.findOne({
      startDate: ISODate("2025-02-01"),
      endDate: ISODate("2026-02-01")
    })._id,
    claimDate: ISODate("2025-05-20"),
    claimAmount: 25000,
    claimStatus: "Rejected"
  }
])
```



```
..      claimAmount: 60000,
..      claimStatus: "Approved"
..    },
..    {
..      assignmentId: db.policyAssignments.findOne({
..        startDate: ISODate("2025-02-01"),
..        endDate: ISODate("2026-02-01")
..      })._id,
..      claimDate: ISODate("2025-05-20"),
..      claimAmount: 25000,
..      claimStatus: "Rejected"
..    }
.. ])
..

 acknowledged: true,
 insertedIds: {
   '0': ObjectId('695768c5e02dd097951e2645'),
   '1': ObjectId('695768c5e02dd097951e2646'),
   '2': ObjectId('695768c5e02dd097951e2647'),
   '3': ObjectId('695768c5e02dd097951e2648'),
   '4': ObjectId('695768c5e02dd097951e2649'),
   '5': ObjectId('695768c5e02dd097951e264a'),
   '6': ObjectId('695768c5e02dd097951e264b')
 }
```

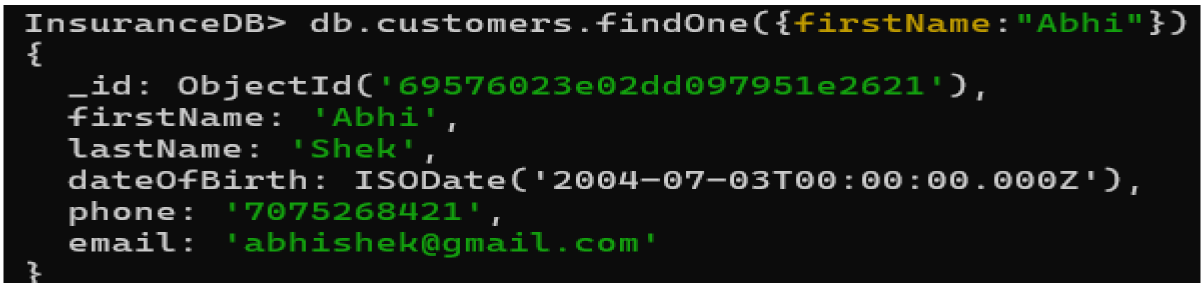
Creation of claims collection:

db.createCollection("claims")

FIND OPERATIONS:

Find all documents: db.customers.find()

Find one document: db.customers.findOne({ firstName: "Abhi" })

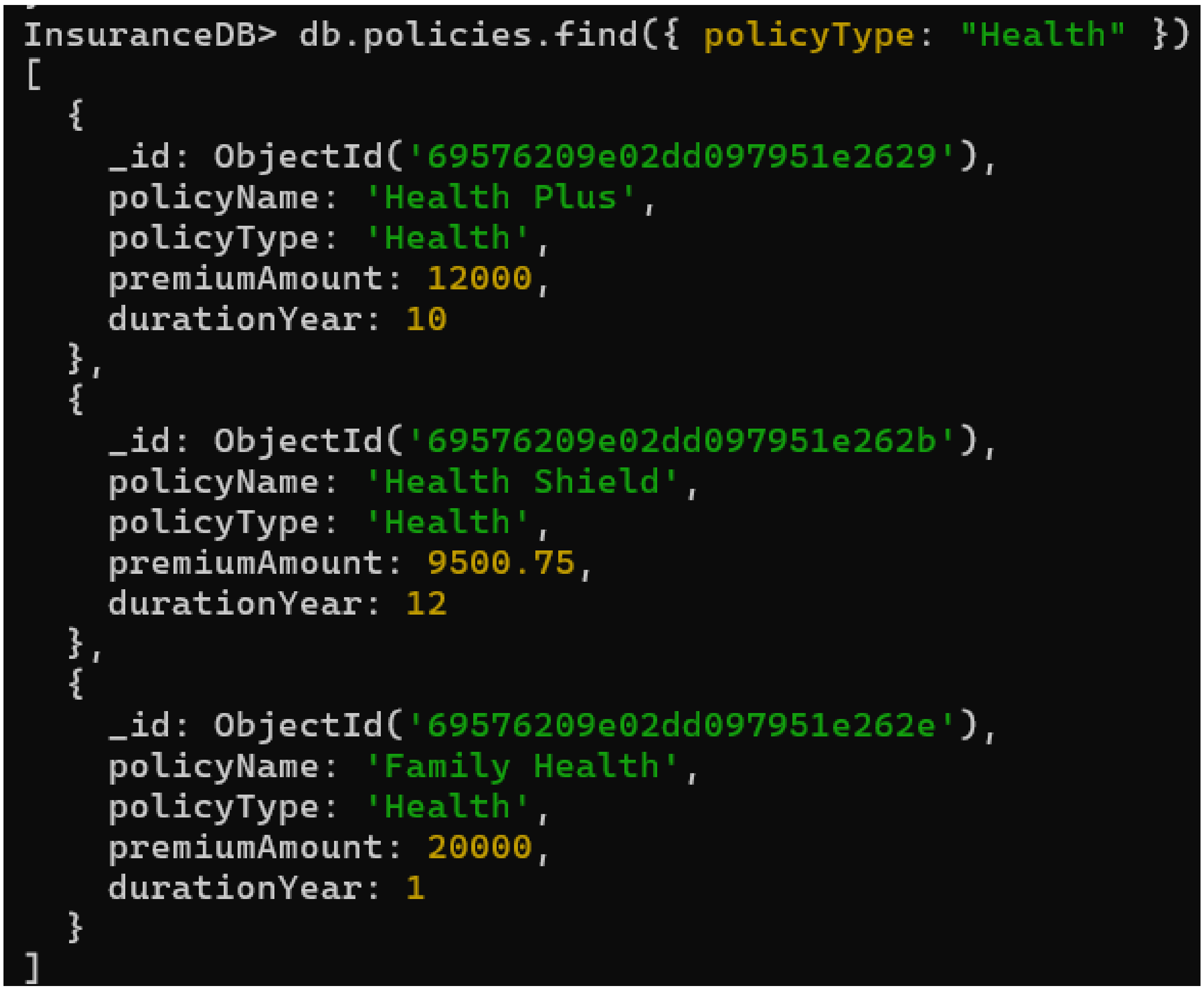


```
InsuranceDB> db.customers.findOne({firstName:"Abhi"})
{
  _id: ObjectId('69576023e02dd097951e2621'),
  firstName: 'Abhi',
  lastName: 'Shek',
  dateOfBirth: ISODate('2004-07-03T00:00:00.000Z'),
  phone: '7075268421',
  email: 'abhishek@gmail.com'
}
```

Find with condition:

Similar to where in sql:

db.policies.find({ policyType: "Health" })

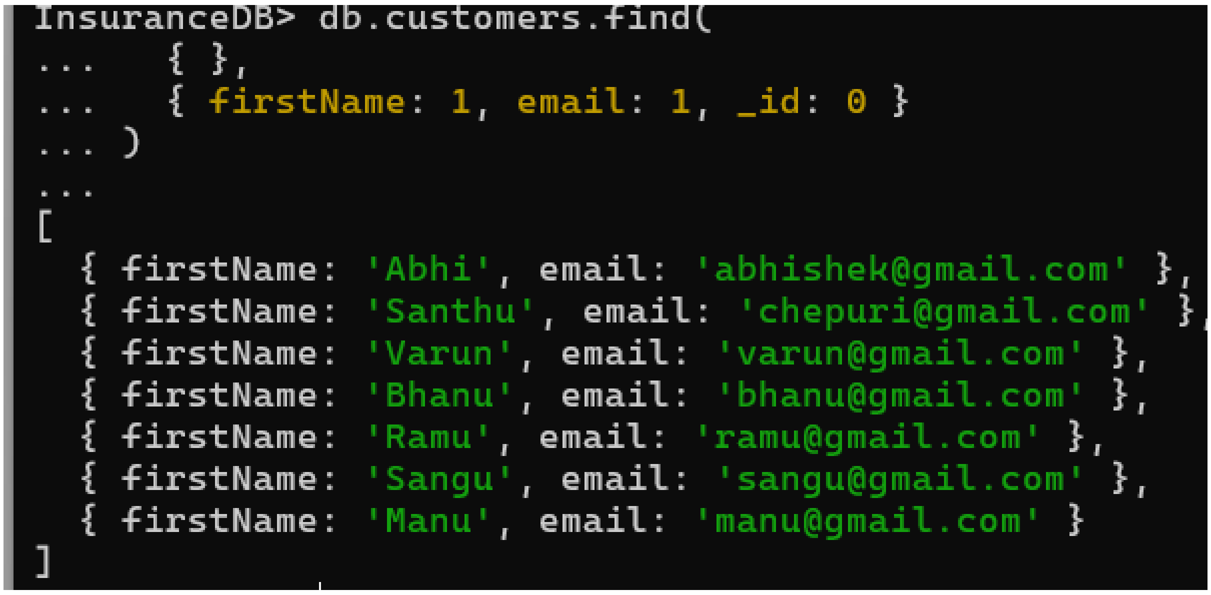


```
InsuranceDB> db.policies.find({ policyType: "Health" })
[
  {
    _id: ObjectId('69576209e02dd097951e2629'),
    policyName: 'Health Plus',
    policyType: 'Health',
    premiumAmount: 12000,
    durationYear: 10
  },
  {
    _id: ObjectId('69576209e02dd097951e262b'),
    policyName: 'Health Shield',
    policyType: 'Health',
    premiumAmount: 9500.75,
    durationYear: 12
  },
  {
    _id: ObjectId('69576209e02dd097951e262e'),
    policyName: 'Family Health',
    policyType: 'Health',
    premiumAmount: 20000,
    durationYear: 1
  }
]
```

Projection: MongoDB projection is the process of selecting only the specific fields we want to retrieve from a document rather than fetching the entire document.

- **Filters Documents:** The first parameter of find() selects which documents to retrieve.
- **Specifies Fields:** The second parameter (projection object) tells MongoDB which fields to include (1/true) or exclude (0/false).
- **Controls _id:** By default, _id is always returned unless explicitly excluded (_id: 0).

```
db.customers.find(
...   { },
...   { firstName: 1, email: 1, _id: 0 }
... )
```



```
InsuranceDB> db.customers.find(
...    { },
...    { firstName: 1, email: 1, _id: 0 }
... )
...
[
  { firstName: 'Abhi', email: 'abhishek@gmail.com' },
  { firstName: 'Santhu', email: 'chepuri@gmail.com' },
  { firstName: 'Varun', email: 'varun@gmail.com' },
  { firstName: 'Bhanu', email: 'bhanu@gmail.com' },
  { firstName: 'Ramu', email: 'ramu@gmail.com' },
  { firstName: 'Sangu', email: 'sangu@gmail.com' },
  { firstName: 'Manu', email: 'manu@gmail.com' }
]
```

Comparison operators: <,>,=

db.policies.find({ premiumAmount: { $gt: 10000 } })

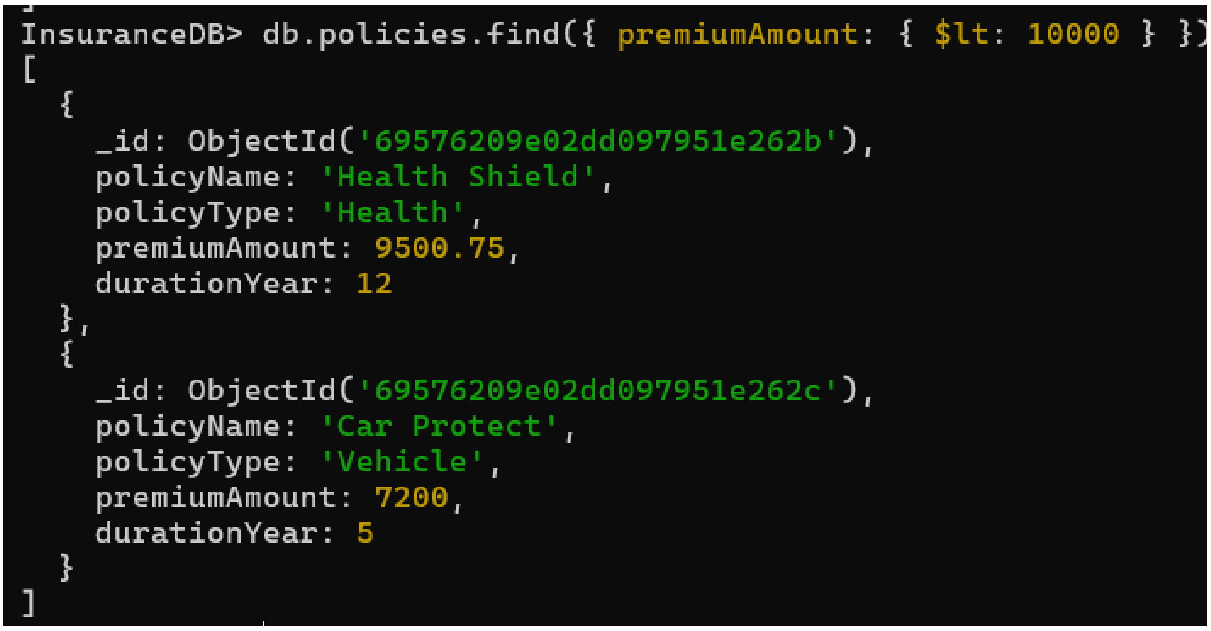
db.policies.find({ premiumAmount: { $lt: 10000 } })

db.policies.find({ premiumAmount: { $gte: 12000 } })

gt:greater than

lt:less than

gte:greater than or equal to

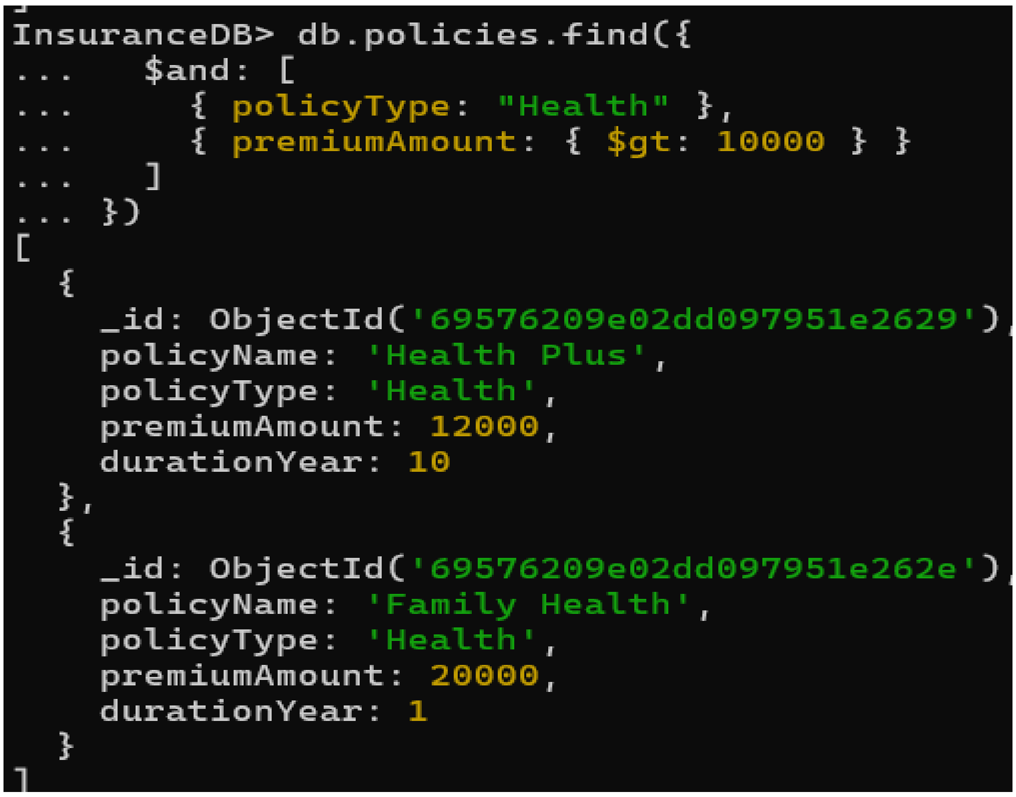


```
InsuranceDB> db.policies.find({ premiumAmount: { $lt: 10000 } })
[
  {
    _id: ObjectId('69576209e02dd097951e262b'),
    policyName: 'Health Shield',
    policyType: 'Health',
    premiumAmount: 9500.75,
    durationYear: 12
  },
  {
    _id: ObjectId('69576209e02dd097951e262c'),
    policyName: 'Car Protect',
    policyType: 'Vehicle',
    premiumAmount: 7200,
    durationYear: 5
  }
]
```

Logical operators: AND,OR,IN

AND operator:

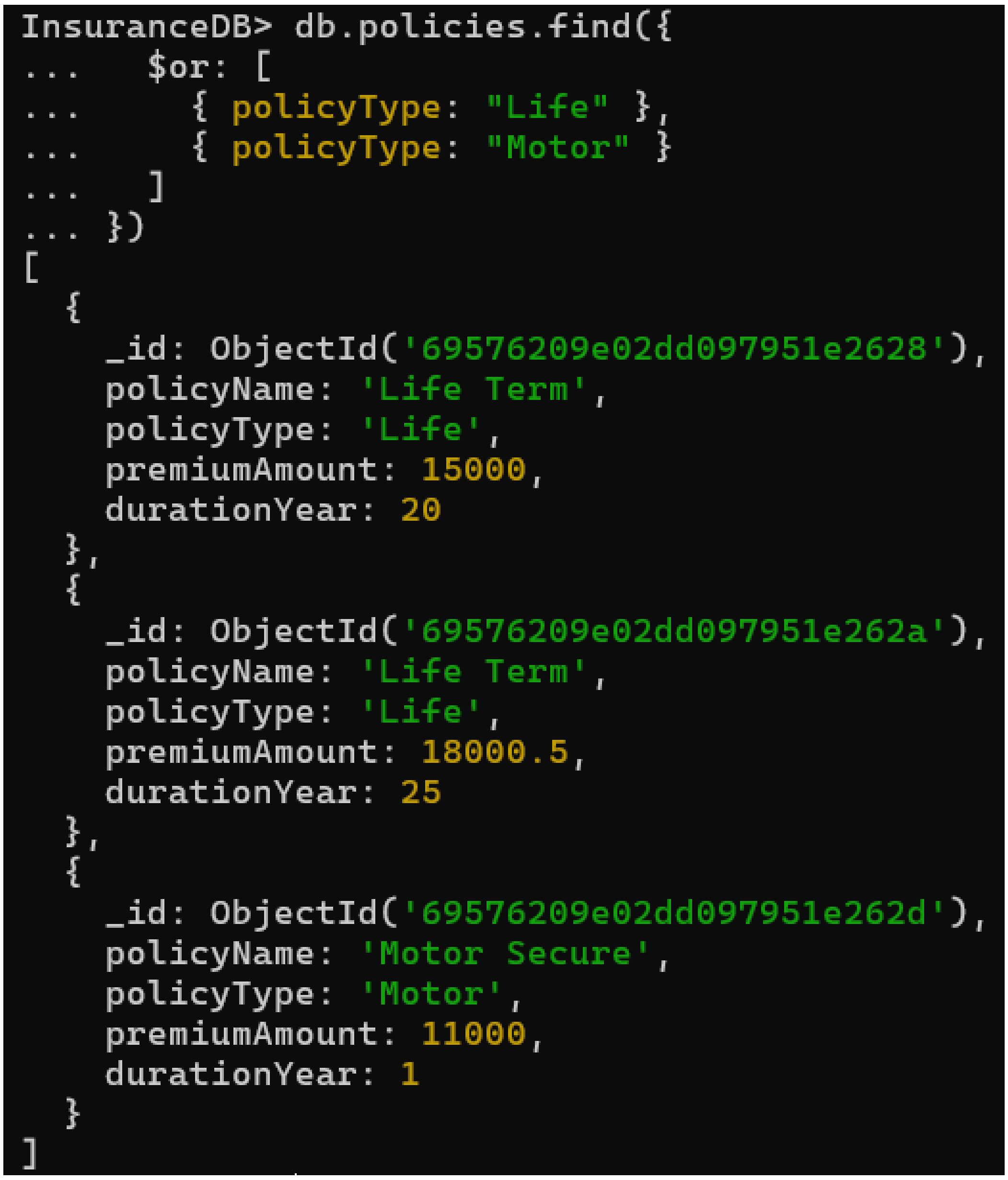
```
db.policies.find({
  $and: [
    { policyType: "Health" },
    { premiumAmount: { $gt: 10000 } }]})
```



```
InsuranceDB> db.policies.find({
...     $and: [
...       { policyType: "Health" },
...       { premiumAmount: { $gt: 10000 } }
...     ]
... })
[
  {
    _id: ObjectId('69576209e02dd097951e2629'),
    policyName: 'Health Plus',
    policyType: 'Health',
    premiumAmount: 12000,
    durationYear: 10
  },
  {
    _id: ObjectId('69576209e02dd097951e262e'),
    policyName: 'Family Health',
    policyType: 'Health',
    premiumAmount: 20000,
    durationYear: 1
  }
]
```

OR operator:

```
db.policies.find({
  $or: [
    { policyType: "Life" },
    { policyType: "Motor" }
  ]
})
```

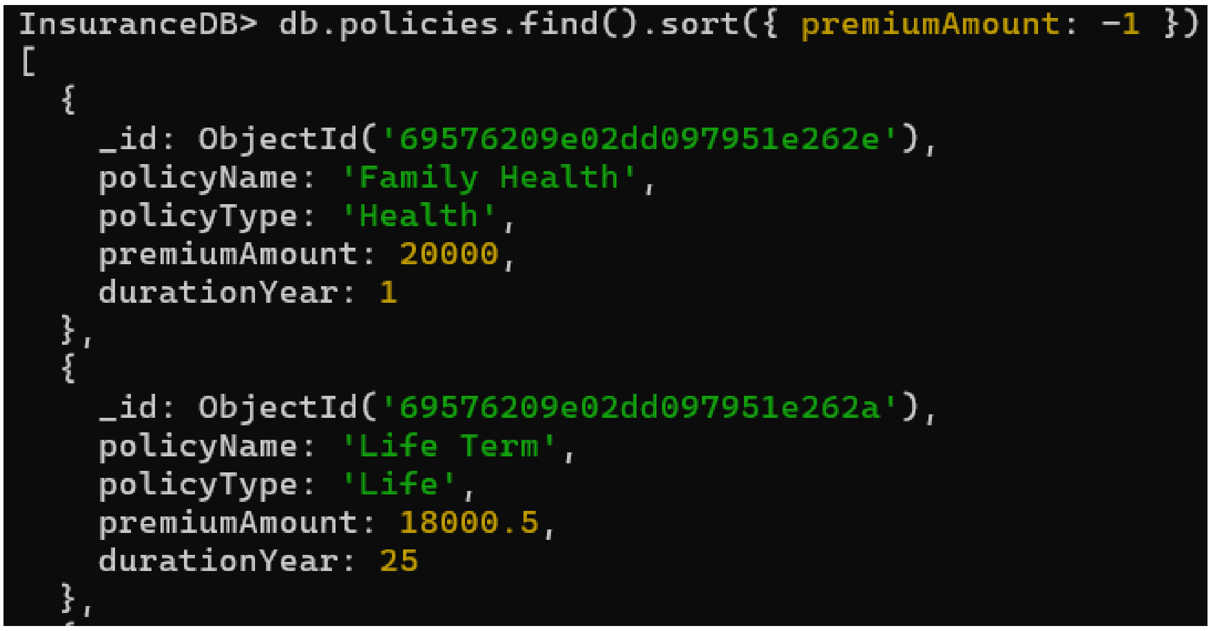


```
InsuranceDB> db.policies.find({
...   $or: [
...     { policyType: "Life" },
...     { policyType: "Motor" }
...   ]
... })
[
  {
    _id: ObjectId('69576209e02dd097951e2628'),
    policyName: 'Life Term',
    policyType: 'Life',
    premiumAmount: 15000,
    durationYear: 20
  },
  {
    _id: ObjectId('69576209e02dd097951e262a'),
    policyName: 'Life Term',
    policyType: 'Life',
    premiumAmount: 18000.5,
    durationYear: 25
  },
  {
    _id: ObjectId('69576209e02dd097951e262d'),
    policyName: 'Motor Secure',
    policyType: 'Motor',
    premiumAmount: 11000,
    durationYear: 1
  }
]
```

Basic pagination kind of things: SORT, LIMIT, SKIP

Displaying policies in descending order of premium amount:

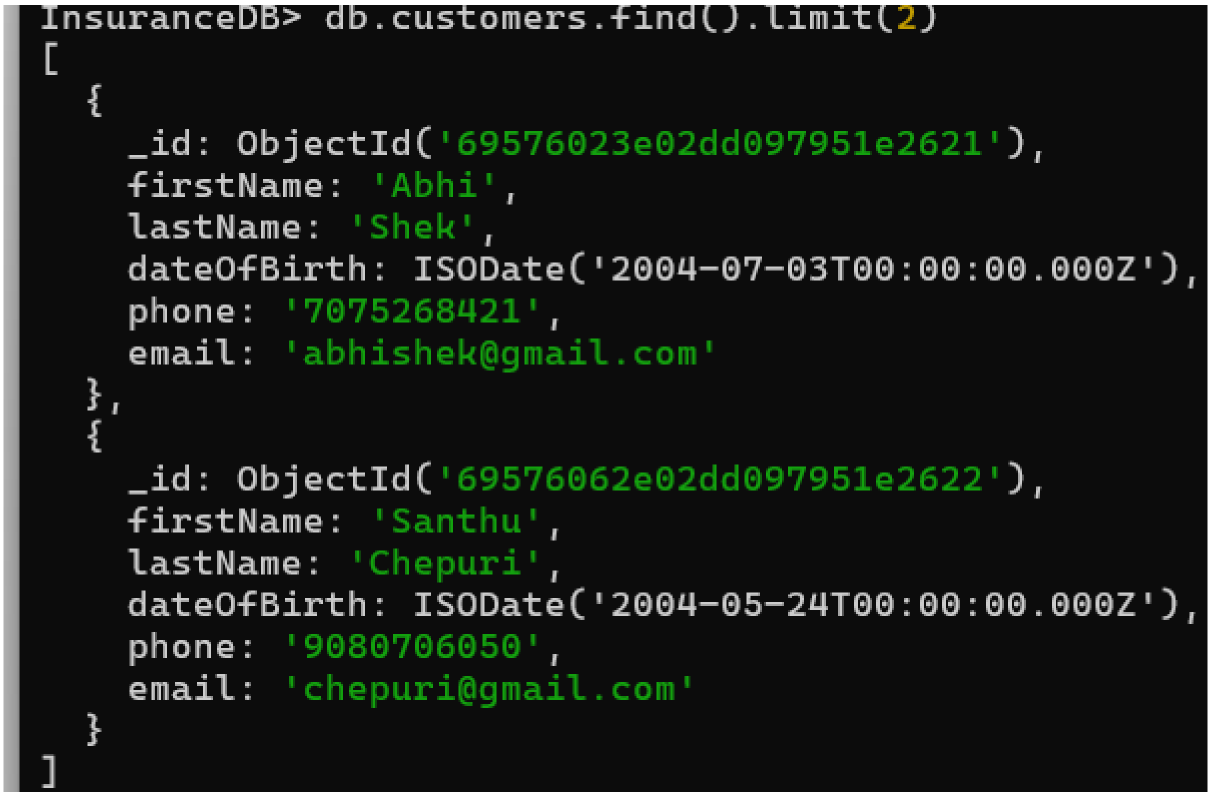
db.policies.find().sort({ premiumAmount: -1 })



```
InsuranceDB> db.policies.find().sort({ premiumAmount: -1 })
[
  {
    _id: ObjectId('69576209e02dd097951e262e'),
    policyName: 'Family Health',
    policyType: 'Health',
    premiumAmount: 20000,
    durationYear: 1
  },
  {
    _id: ObjectId('69576209e02dd097951e262a'),
    policyName: 'Life Term',
    policyType: 'Life',
    premiumAmount: 18000.5,
    durationYear: 25
  },
```

Displaying top 2 from collection:

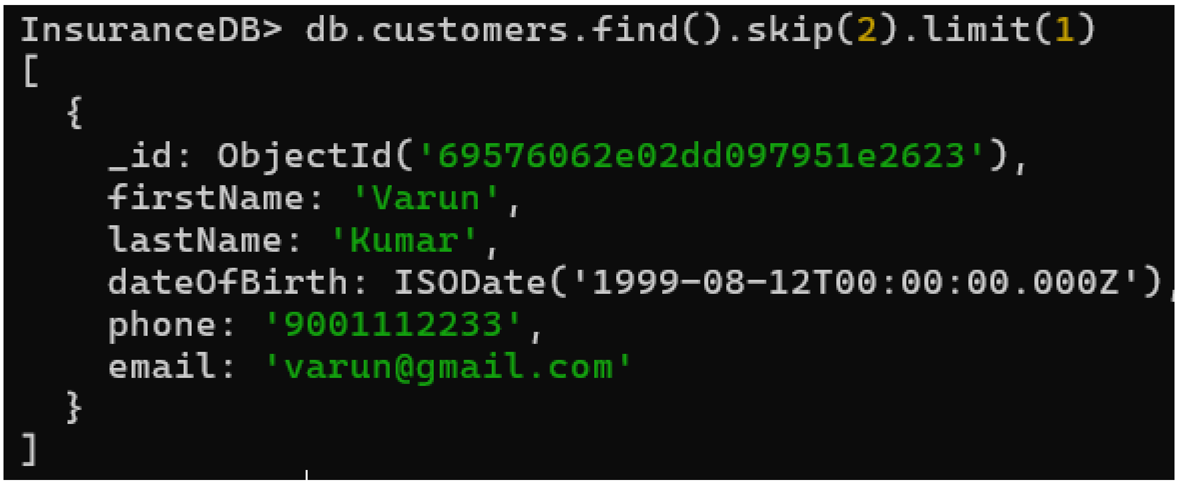
db.customers.find().limit(2)



```
InsuranceDB> db.customers.find().limit(2)
[
  {
    _id: ObjectId('69576023e02dd097951e2621'),
    firstName: 'Abhi',
    lastName: 'Shek',
    dateOfBirth: ISODate('2004-07-03T00:00:00.000Z'),
    phone: '7075268421',
    email: 'abhishek@gmail.com'
  },
  {
    _id: ObjectId('69576062e02dd097951e2622'),
    firstName: 'Santhu',
    lastName: 'Chepuri',
    dateOfBirth: ISODate('2004-05-24T00:00:00.000Z'),
    phone: '9080706050',
    email: 'chepuri@gmail.com'
  }
]
```

Skipping some and displaying top

db.customers.find().skip(2).limit(1)



```
InsuranceDB> db.customers.find().skip(2).limit(1)
[
  {
    _id: ObjectId('69576062e02dd097951e2623'),
    firstName: 'Varun',
    lastName: 'Kumar',
    dateOfBirth: ISODate('1999-08-12T00:00:00.000Z')
    phone: '9001112233',
    email: 'varun@gmail.com'
  }
]
```

Update collections:

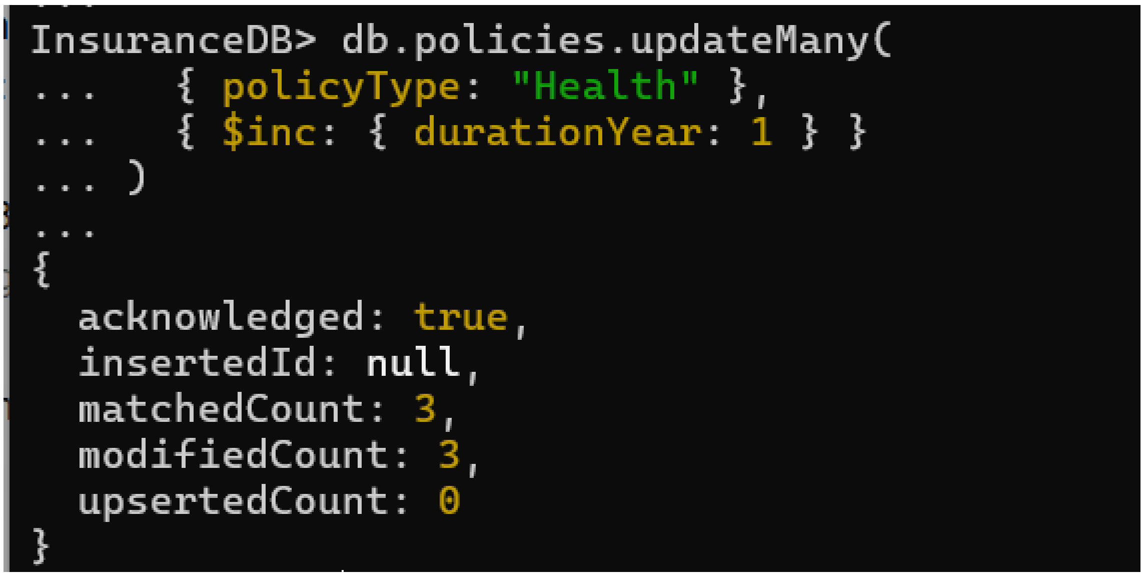
Updateone:

```
db.customers.updateOne(
...   { email: "abhishek@gmail.com" },
...   { $set: { phone: "8888888888" } }
... )
...
```

OUTPUT:

```
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

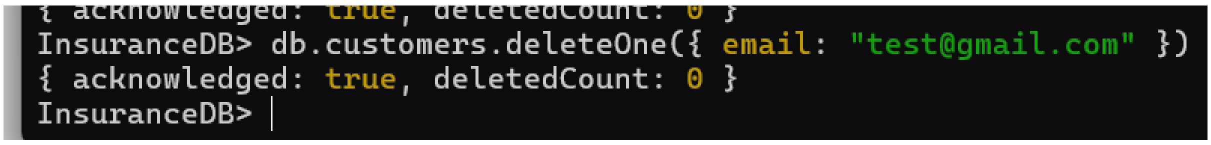
```
db.policies.updateMany(
  { policyType: "Health" },
  { $inc: { durationYear: 1 } })
```



```
InsuranceDB> db.policies.updateMany(
...    { policyType: "Health" },
...    { $inc: { durationYear: 1 } }
... )
...
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 3,
  modifiedCount: 3,
  upsertedCount: 0
}
```

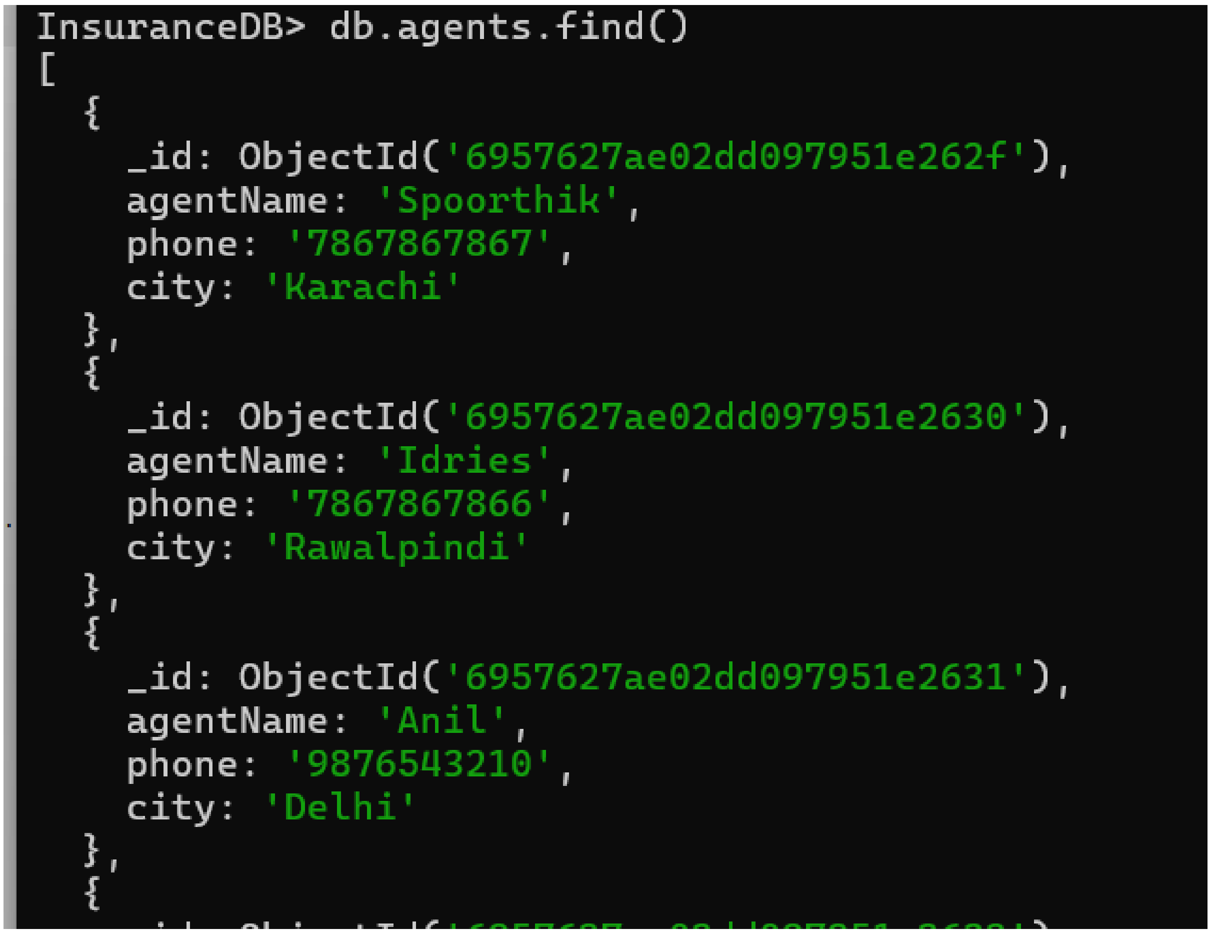
Delete from collections:

Here the email mentioned is not present in collections so we see deletedcount=0



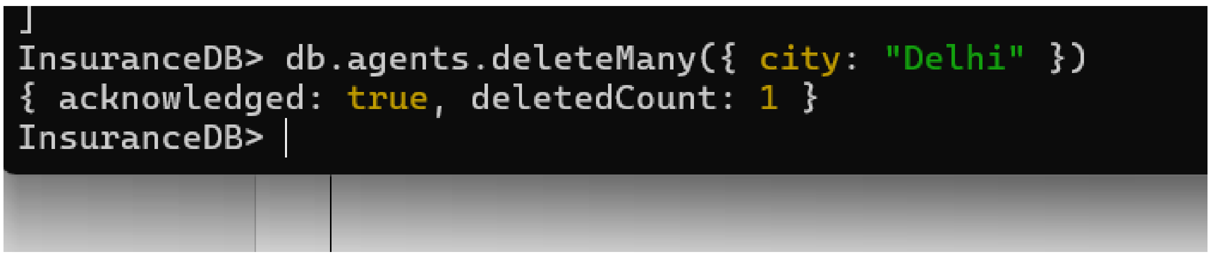
```
{ acknowledged: true, deletedCount: 0 }
InsuranceDB> db.customers.deleteOne({ email: "test@gmail.com" })
{ acknowledged: true, deletedCount: 0 }
InsuranceDB>
```

Checking collection:



```
InsuranceDB> db.agents.find()
[
  {
    _id: ObjectId('6957627ae02dd097951e262f'),
    agentName: 'Spoorthik',
    phone: '7867867867',
    city: 'Karachi'
  },
  {
    _id: ObjectId('6957627ae02dd097951e2630'),
    agentName: 'Idries',
    phone: '7867867866',
    city: 'Rawalpindi'
  },
  {
    _id: ObjectId('6957627ae02dd097951e2631'),
    agentName: 'Anil',
    phone: '9876543210',
    city: 'Delhi'
  },
  {
```

db.agents.deleteMany({ city: "Delhi" })



```
]
InsuranceDB> db.agents.deleteMany({ city: "Delhi" })
{ acknowledged: true, deletedCount: 1 }
InsuranceDB> 
```

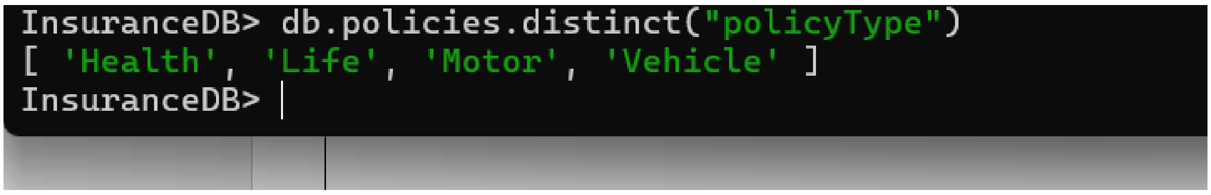
here we had one field it is deleted with deleteMany,incase of multiple all dets deleted

COUNT & DISTINCT:

InsuranceDB> db.customers.countDocuments()

7

db.policies.distinct("policyType")



```
InsuranceDB> db.policies.distinct("policyType")
[ 'Health', 'Life', 'Motor', 'Vehicle' ]
InsuranceDB> 
```