Step 1: Create Database & Collection

```
use insuranceDB

db.policies.insertMany([
  { policyNo: 101, customerName: "Amit", policyType: "Health", premium: 12000, city: "Delhi", active: true },
  { policyNo: 102, customerName: "Neha", policyType: "Life", premium: 18000, city: "Mumbai", active: true },
  { policyNo: 103, customerName: "Rahul", policyType: "Car", premium: 9000, city: "Delhi", active: false },
  { policyNo: 104, customerName: "Sneha", policyType: "Health", premium: 15000, city: "Chennai", active: true },
  { policyNo: 105, customerName: "Karan", policyType: "Life", premium: 22000, city: "Mumbai", active: false }
])
```

1. Display all documents

```
db.policies.find()
```

Output

```
{ policyNo: 101, customerName: "Amit", policyType: "Health", premium: 12000, city: "Delhi", active: true }

...
```

---

2. Display only Health policies

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

Output

```
{ policyNo: 101, customerName: "Amit", policyType: "Health", premium: 12000, city: "Delhi", active: true }

{ policyNo: 104, customerName: "Sneha", policyType: "Health", premium: 15000, city: "Chennai", active: true }
```

---

3. Policies from Mumbai

```
db.policies.find({ city: "Mumbai" })
```

Output

```
{ policyNo: 102, customerName: "Neha", policyType: "Life", premium: 18000 }

{ policyNo: 105, customerName: "Karan", policyType: "Life", premium: 22000 }
```

PROJECTIONS

4. Show only customerName and policyType

```
db.policies.find({}, { customerName: 1, policyType: 1 })
```

Output

{ customerName: "Amit", policyType: "Health" }

{ customerName: "Neha", policyType: "Life" }

---

5. Exclude _id

db.policies.find({}, { _id: 0, customerName: 1, premium: 1 })

Output

{ customerName: "Amit", premium: 12000 }

{ customerName: "Neha", premium: 18000 }

---

6. Show all fields except premium

db.policies.find({}, { premium: 0 })

Output

{ policyNo: 101, customerName: "Amit", policyType: "Health", city: "Delhi", active: true }

CONDITIONAL QUERIES

7. Premium greater than 15000

db.policies.find({ premium: { $gt: 15000 } })

Output

{ policyNo: 102, customerName: "Neha", premium: 18000 }

{ policyNo: 105, customerName: "Karan", premium: 22000 }

---

8. Active policies only

db.policies.find({ active: true })

Output

{ policyNo: 101, customerName: "Amit" }

{ policyNo: 102, customerName: "Neha" }

{ policyNo: 104, customerName: "Sneha" }

CREATE (INSERT)

9. Insert a new policy

db.policies.insertOne({

```
  policyNo: 106,
  customerName: "Rohit",
  policyType: "Bike",
  premium: 5000,
  city: "Pune",
  active: true
})
```

Output

```
{ acknowledged: true, insertedId: ObjectId("...") }
```

UPDATE

10. Update premium of policyNo 101

```
db.policies.updateOne(
  { policyNo: 101 },
  { $set: { premium: 13000 } }
)
```

Output

```
{ acknowledged: true, matchedCount: 1, modifiedCount: 1 }
```

---

11. **Mark all Life policies as inactive**

```
db.policies.updateMany(
  { policyType: "Life" },
  { $set: { active: false } }
)
```

Output

```
{ matchedCount: 2, modifiedCount: 2 }
```

DELETE

12. Delete policy with policyNo 103

```
db.policies.deleteOne({ policyNo: 103 })
```

Output

```
{ deletedCount: 1 }
```

---

13. Delete all inactive policies

db.policies.deleteMany({ active: false })

Output

{ deletedCount: 3 }

COMBINATION QUERIES

14. Active Health policies with projection

db.policies.find(

{ policyType: "Health", active: true },

{ _id: 0, customerName: 1, premium: 1 }

)

Output

{ customerName: "Amit", premium: 13000 }

{ customerName: "Sneha", premium: 15000 }

---

15. Premium between 10000 and 20000

db.policies.find(

{ premium: { $gte: 10000, $lte: 20000 } },

{ customerName: 1, premium: 1 }

)

Output

{ customerName: "Amit", premium: 13000 }

{ customerName: "Sneha", premium: 15000 }

More on Find & Filter

16. Policies with premium ≥ 15000 in Mumbai

db.policies.find({ premium: { $gte: 15000 }, city: "Mumbai" })

Output

{ customerName: "Neha", premium: 18000, city: "Mumbai" }

---

17. Policies NOT from Delhi

db.policies.find({ city: { $ne: "Delhi" } })

Output

{ customerName: "Neha", city: "Mumbai" }

{ customerName: "Sneha", city: "Chennai" }

---

18. Policies with premium between 10000 and 20000

```
db.policies.find({ premium: { $gt: 10000, $lt: 20000 } })
```

Output

{ customerName: "Amit", premium: 13000 }

{ customerName: "Sneha", premium: 15000 }

---

19. Active policies from Delhi or Mumbai

```
db.policies.find({
  active: true,
  city: { $in: ["Delhi", "Mumbai"] }
})
```

Output

{ customerName: "Amit", city: "Delhi" }

{ customerName: "Neha", city: "Mumbai" }

---

20. Policies where premium is NOT between 10k–20k

```
db.policies.find({
  premium: { $not: { $gte: 10000, $lte: 20000 } }
})
```

Output

{ customerName: "Karan", premium: 22000 }

---

21. Policies with missing active field

```
db.policies.find({ active: { $exists: false } })
```

Output

(no documents)

---

22. Policies sorted by premium (descending)

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

Output

{ customerName: "Karan", premium: 22000 }

---

23. Top 2 highest premium policies

db.policies.find().sort({ premium: -1 }).limit(2)

Output

{ customerName: "Karan", premium: 22000 }

{ customerName: "Neha", premium: 18000 }

---

24. Skip first 2 records

db.policies.find().skip(2)

Output

{ customerName: "Rahul" }

---

25. **Count active policies**

db.policies.countDocuments({ active: true })

Output

3

More on Projections & Expressions

26. Show name, premium, exclude _id

db.policies.find({}, { _id: 0, customerName: 1, premium: 1 })

---

27. Rename fields using projection

```
db.policies.find(
  {},
  { _id: 0, Name: "$customerName", Amount: "$premium" }
)
```

---

28. Show policyType in uppercase

```
db.policies.aggregate([
  { $project: { policyType: { $toUpper: "$policyType" } } }
```

])

---

29. Add GST (18%) to premium (calculated field)

```
db.policies.aggregate([
 {
  $project: {
   customerName: 1,
   totalPremium: { $multiply: ["$premium", 1.18] }
  }
 }
])
```

---

30. Conditional field using $cond

```
db.policies.aggregate([
 {
  $project: {
   customerName: 1,
   category: {
    $cond: [
     { $gt: ["$premium", 15000] },
     "High",
     "Normal"
    ]
   }
  }
 }
])
```

Aggregation & Grouping

31. Total premium collected

```
db.policies.aggregate([
 { $group: { _id: null, totalPremium: { $sum: "$premium" } } }
```

])

---

32. Average premium per policyType

```
db.policies.aggregate([
  { $group: { _id: "$policyType", avgPremium: { $avg: "$premium" } } }
])
```

---

33. Number of policies per city

```
db.policies.aggregate([
  { $group: { _id: "$city", count: { $sum: 1 } } }
])
```

---

34. Highest premium per policyType

```
db.policies.aggregate([
  { $group: { _id: "$policyType", maxPremium: { $max: "$premium" } } }
])
```

---

35. Only cities with more than 1 policy

```
db.policies.aggregate([
  { $group: { _id: "$city", count: { $sum: 1 } } },
  { $match: { count: { $gt: 1 } } }
])
```

---

36. Sort policy types by average premium

```
db.policies.aggregate([
  { $group: { _id: "$policyType", avgPremium: { $avg: "$premium" } } },
  { $sort: { avgPremium: -1 } }
])
```

---

37. Total premium by city for active policies only

```
db.policies.aggregate([
  { $match: { active: true } },
```

```
  { $group: { _id: "$city", total: { $sum: "$premium" } } }
])
```

---

38. Count active vs inactive policies

```
db.policies.aggregate([
  { $group: { _id: "$active", count: { $sum: 1 } } }
])
```

---

39. Find policyType having avg premium > 15000

```
db.policies.aggregate([
  { $group: { _id: "$policyType", avgPremium: { $avg: "$premium" } } },
  { $match: { avgPremium: { $gt: 15000 } } }
])
```

---

40. Add policyCount field to each document

```
db.policies.aggregate([
  {
    $group: {
      _id: "$policyType",
      policies: { $push: "$$ROOT" },
      count: { $sum: 1 }
    }
  }
])
```

Updates & Deletes

41. Increase all Health premiums by 10%

```
db.policies.updateMany(
  { policyType: "Health" },
  { $mul: { premium: 1.1 } }
)
```

---

42. Add lastUpdated field to all documents

```
db.policies.updateMany(
  {},
  { $set: { lastUpdated: new Date() } }
)
```

---

43. Remove city field from inactive policies

```
db.policies.updateMany(
  { active: false },
  { $unset: { city: "" } }
)
```

---

44. Delete policies with premium > 25000

```
db.policies.deleteMany({ premium: { $gt: 25000 } })
```

---

45. Drop entire collection

```
db.policies.drop()
```