Perfect! Let's create a **MongoDB query practice set** from **basic → advanced**, including **CRUD, filtering, aggregation, joins (lookup), and indexing**, step by step. I'll include **English + Hinglish explanations**.

Tables / Collections for practice

users

```
[
 { "_id": 1, "name": "Sourabh", "age": 22, "city": "Mumbai" },
 { "_id": 2, "name": "Ravi", "age": 25, "city": "Delhi" },
 { "_id": 3, "name": "Ankit", "age": 28, "city": "Bangalore" }
]
```

orders

```
[
 { "_id": 1, "user_id": 1, "product": "Laptop", "amount": 50000, "order_date": "2025-01-01" },
 { "_id": 2, "user_id": 1, "product": "Mouse", "amount": 500, "order_date": "2025-01-05" },
 { "_id": 3, "user_id": 2, "product": "Keyboard", "amount": 1500, "order_date": "2025-02-10" },
 { "_id": 4, "user_id": 3, "product": "Monitor", "amount": 12000, "order_date": "2025-03-15" }
]
```

products

```
[
 { "_id": 1, "name": "Laptop", "category": "Electronics", "price": 50000 },
 { "_id": 2, "name": "Mouse", "category": "Electronics", "price": 500 },
 { "_id": 3, "name": "Keyboard", "category": "Electronics", "price": 1500 },
 { "_id": 4, "name": "Monitor", "category": "Electronics", "price": 12000 }
]
```

1️⃣ Basic Queries

1. Find all users

```
db.users.find()
```

- English: Retrieve all documents from users.
- Hinglish: users collection ke sabhi documents fetch karo.

2. Find users older than 24

```
db.users.find({ age: { $gt: 24 } })
```

- English: Filter users with age > 24.
- Hinglish: Age > 24 wale users fetch karo.

3. Find specific fields

```
db.users.find({}, { name: 1, city: 1, _id: 0 })
```

- English: Retrieve only name and city, exclude _id.
- Hinglish: Sirf name aur city dikhao, _id exclude karo.

4. Sort users by age descending

```
db.users.find().sort({ age: -1 })
```

- English: Sort users from oldest to youngest.
- Hinglish: Users ko age ke descending order me fetch karo.

2️⃣ Insert / Update / Delete

1. Insert a new user

```
db.users.insertOne({ _id: 4, name: "Neha", age: 26, city: "Pune" })
```

- English: Add a new document to users.
- Hinglish: users me naya document add karo.

2. Update user age

```
db.users.updateOne({ name: "Sourabh" }, { $set: { age: 23 } })
```

- English: Update age of Sourabh.
- Hinglish: Sourabh ka age update karo.

3. Delete a user

```
db.users.deleteOne({ name: "Ravi" })
```

- English: Delete user named Ravi.
- Hinglish: Ravi naam ka user delete karo.

3️⃣ Filtering Queries

1. Find users in Mumbai or Delhi

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

- English: Filter users in specific cities.
- Hinglish: Mumbai ya Delhi wale users fetch karo.

2. Find users not in Bangalore

```
db.users.find({ city: { $ne: "Bangalore" } })
```

- English: Users excluding Bangalore.
- Hinglish: Bangalore ke alawa users fetch karo.

3. Find users between 22 and 26

```
db.users.find({ age: { $gte: 22, $lte: 26 } })
```

- English: Users with age between 22 and 26.
- Hinglish: Age 22 se 26 ke beech wale users fetch karo.

4️⃣ Aggregation Queries

1. Total amount spent by each user

```
db.orders.aggregate([
 { $group: { _id: "$user_id", totalSpent: { $sum: "$amount" } } }
])
```

- English: Sum of order amounts per user.
- Hinglish: Har user ka total spend calculate karo.

2. Average order amount

```
db.orders.aggregate([
 { $group: { _id: "$user_id", avgAmount: { $avg: "$amount" } } }
])
```

- English: Average spending per user.
- Hinglish: Har user ka average order amount calculate karo.

3. Maximum order amount

```
db.orders.aggregate([
 { $group: { _id: null, maxAmount: { $max: "$amount" } } }
])
```

- English: Find largest order amount.
- Hinglish: Sabse bada order amount fetch karo.

5️⃣ Join Queries (Lookup)

1. Join orders with users

```
db.orders.aggregate([
 {
  $lookup: {
   from: "users",
   localField: "user_id",
   foreignField: "_id",
   as: "userInfo"
  }
 }
])
```

- English: Add user info to each order.
- Hinglish: Har order me corresponding user ka info join karo.

2. Join orders with products

```
db.orders.aggregate([
 {
  $lookup: {
   from: "products",
   localField: "product",
   foreignField: "name",
   as: "productInfo"
  }
 }
])
```

- English: Add product details to orders.
- Hinglish: Har order me product details join karo.

3. Users with their orders

```
db.users.aggregate([
 {
  $lookup: {
   from: "orders",
   localField: "_id",
   foreignField: "user_id",
   as: "userOrders"
  }
 }
])
```

- English: Add orders array to each user.
- Hinglish: Har user ke document me unke orders array add karo.

6️⃣ Advanced Aggregation + Filtering

1. Users who spent more than 5000

```
db.orders.aggregate([
 { $group: { _id: "$user_id", totalSpent: { $sum: "$amount" } } },
 { $match: { totalSpent: { $gt: 5000 } } }
])
```

- English: Filter users based on sum of orders.
- Hinglish: Total orders 5000 se zyada wale users fetch karo.

2. Orders sorted by amount descending

```
db.orders.aggregate([
 { $sort: { amount: -1 } }
])
```

- English: Sort orders from largest to smallest.
- Hinglish: Orders ko descending amount ke order me fetch karo.

3. Count of orders per product

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

- English: Count how many times each product was ordered.
- Hinglish: Har product kitni baar order hua, count karo.

7️⃣ Indexing & Optimization

1. Create an index on user_id

```
db.orders.createIndex({ user_id: 1 })
```

- English: Speeds up queries filtering by user_id.
- Hinglish: user_id field ke queries fast karne ke liye index banao.

2. Create a compound index

```
db.orders.createIndex({ user_id: 1, amount: -1 })
```

- English: Optimize queries filtering by user_id and sorting by amount.
- Hinglish: user_id filter aur amount sort fast karne ke liye compound index.