Design and Develop MongoDB Queries using CRUD operations:
Create Employee collection by considering following Fields:
i. Name: Embedded Doc (FName, LName)
ii. Company Name: String
iii. Salary: Number
iv. Designation: String
v. Age: Number
vi. Expertise: Array
vii. DOB: String or Date
viii. Email id: String
ix. Contact: String
x. Address: Array of Embedded Doc (PAddr, LAddr)
Insert at least 5 documents in collection by considering above attribute and execute following queries:
1. Final name of Employee where age is less than 30 and salary more than 50000.
2. Creates a new document if no document in the employee collection contains
{Designation: "Tester", Company_name: "TCS", Age: 25}
3. Selects all documents in the collection where the field age has a value less than 30 or the value of the salary field is greater than 40000.
4. Find documents where Designation is not equal to "Developer".
5. Find _id, Designation, Address and Name from all documents where Company_name is "Infosys".
6. Display only FName and LName of all Employees

```
// Create the "Employee" collection
db.createCollection("Employee")

// Insert sample entries into the "Employee" collection
db.Employee.insert([
    {
        "Name": { "FName": "John", "LName": "Doe" },
        "Company_Name": "Infosys",
        "Salary": 60000,
        "Designation": "Developer",
        "Age": 28,
        "Expertise": ["Java", "Python"],
        "DOB": "1995-01-15",
        "Email_id": "john.doe@example.com",
        "Contact": "9876543210",
        "Address": [{ "PAddr": "123 Main St", "LAddr": "Apt 45" }]
    },
    {
        "Name": { "FName": "Alice", "LName": "Smith" },
        "Company_Name": "TCS",
        "Salary": 35000,
        "Designation": "Tester",
        "Age": 25,
        "Expertise": ["Testing", "Automation"],
        "DOB": "1998-05-22",
        "Email_id": "alice.smith@example.com",
        "Contact": "8765432109",
```

```
        "Address": [{ "PAddr": "456 Oak St", "LAddr": "Suite 12" }]
    },
    // Insert three more documents
    // ...

    // Query 1: Final name of Employee where age is less than 30 and salary more
than 50000.
    db.Employee.findOne({ "Age": { $lt: 30 }, "Salary": { $gt: 50000 } }, {
"Name": 1 })

    // Query 2: Creates a new document if no document in the employee collection
contains
    //{Designation: "Tester", Company_name: "TCS", Age: 25}.
    db.Employee.updateOne(
        { "Designation": "Tester", "Company_Name": "TCS", "Age": 25 },
        { $setOnInsert: { "Designation": "Tester", "Company_Name": "TCS", "Age":
25 } },
        { upsert: true }
    )

    // Query 3: Selects all documents in the collection where the field age has
a value less than 30
    //or the value of the salary field is greater than 40000.
    db.Employee.find({ $or: [{ "Age": { $lt: 30 } }, { "Salary": { $gt: 40000 }
}] })

    // Query 4: Find documents where Designation is not equal to "Developer".
    db.Employee.find({ "Designation": { $ne: "Developer" } })

    // Query 5: Find _id, Designation, Address and Name from all documents where
Company_name is "Infosys".
    db.Employee.find({ "Company_Name": "Infosys" }, { "_id": 1, "Designation":
1, "Address": 1, "Name": 1 })

    // Query 6: Display only FName and LName of all Employees.
    db.Employee.find({}, { "Name.FName": 1, "Name.LName": 1 })
])
```