Design and Develop MongoDB Queries using Aggregation operations:
Create Employee collection by considering following Fields:
i. Emp_id : Number
ii. Name: Embedded Doc (FName, LName)
iii. Company Name: String
iv. Salary: Number
v. Designation: String
vi. Age: Number
vii. Expertise: Array
viii. DOB: String or Date
ix. Email id: String
x. Contact: String
xi. Address: Array of Embedded Doc (PAddr, LAddr)
Insert at least 5 documents in collection by considering above attribute and execute following:
1. Using aggregation Return separates value in the Expertise array and return sum of each element of array.
2. Using Aggregate method return Max and Min Salary for each company.
3. Using Aggregate method find Employee with Total Salary for Each City with Designation="DBA".
4. Using aggregation method Return separates value in the Expertise array for employee name where Swapnil Jadhav
5. To Create Compound Indexes on Name: 1, Age: -1
6. Create an Index on Emp_id field , compare the time require to search Emp_id before and after creating an index. (Hint Add at least 10000 Documents)
7. Return a List of Indexes on created on employee Collection.

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

// Insert sample entries into the "Employee" collection
db.Employee.insert([
    {
        "Emp_id": 1,
        "Name": { "FName": "John", "LName": "Doe" },
        "Company_Name": "Infosys",
        "Salary": 60000,
        "Designation": "DBA",
        "Age": 28,
        "Expertise": ["MongoDB", "SQL"],
        "DOB": "1995-01-15",
        "Email_id": "john.doe@example.com",
        "Contact": "9876543210",
        "Address": [{ "PAddr": "123 Main St", "LAddr": "Apt 45" }]
    },
    // Insert four more documents
    // ...

    // Aggregation Queries:

    // Query 1: Return separates value in the Expertise array and return sum of each element of the array.
```

```
    db.Employee.aggregate([
        {
            $unwind: "$Expertise"
        },
        {
            $group: {
                _id: "$Expertise",
                total: { $sum: 1 }
            }
        }
    ])

    // Query 2: Return Max and Min Salary for each company.
    db.Employee.aggregate([
        {
            $group: {
                _id: "$Company_Name",
                maxSalary: { $max: "$Salary" },
                minSalary: { $min: "$Salary" }
            }
        }
    ])

    // Query 3: Find Employee with Total Salary for Each City with
Designation="DBA".
    db.Employee.aggregate([
        {
            $match: { "Designation": "DBA" }
        },
        {
            $group: {
                _id: { City: "$Address.PAddr" },
                totalSalary: { $sum: "$Salary" }
            }
        }
    ])

    // Query 4: Return separates value in the Expertise array for employee name
where Name.
    //FName is "Swapnil" and Name.LName is "Jadhav".
    db.Employee.aggregate([
        {
            $match: { "Name.FName": "Swapnil", "Name.LName": "Jadhav" }
        },
        {
            $unwind: "$Expertise"
        },
        {
            $group: {
                _id: "$Name",
                Expertise: { $push: "$Expertise" }
            }
        }
    ])
```

```
    // Index Creation Queries:

    // Query 5: Create Compound Indexes on Name: 1, Age: -1
    db.Employee.createIndex({ "Name.FName": 1, "Name.LName": 1, "Age": -1 })

    // Query 6: Create an Index on Emp_id field, compare the time required to
search Emp_id before and after creating an index.
    // (Hint: Add at least 10000 Documents)

    // Add 10000 documents
    for (let i = 0; i < 10000; i++) {
        db.Employee.insert({ "Emp_id": i, /* Other fields */ })
    }

    // Time to search Emp_id before creating an index
    const startTimeWithoutIndex = new Date()
    db.Employee.find({ "Emp_id": 5000 })
    const endTimeWithoutIndex = new Date()

    // Create an index on Emp_id
    db.Employee.createIndex({ "Emp_id": 1 })

    // Time to search Emp_id after creating an index
    const startTimeWithIndex = new Date()
    db.Employee.find({ "Emp_id": 5000 })
    const endTimeWithIndex = new Date()

    // Calculate time differences
    const timeDifferenceWithoutIndex = endTimeWithoutIndex -
startTimeWithoutIndex
    const timeDifferenceWithIndex = endTimeWithIndex - startTimeWithIndex

    // Print time differences
    print(`Time without index: ${timeDifferenceWithoutIndex} ms`)
    print(`Time with index: ${timeDifferenceWithIndex} ms`)

    // Query 7: Return a List of Indexes created on the employee Collection.
    db.Employee.getIndexes()
])
```