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 Designation with Total Salary is Above 200000.
2. Using Aggregate method returns names and _id in upper case and in alphabetical order.
3. Using aggregation method find Employee with Total Salary for Each City with Designation="DBA".
4. Create Single Field Indexes on Designation field of employee collection
5. To Create Multikey Indexes on Expertise field of employee collection.
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 Designation with Total Salary above 200000.
    db.Employee.aggregate([
        {
            $group: {
                _id: "$Designation",
                totalSalary: { $sum: "$Salary" }
            }
        },
        {
            $match: { totalSalary: { $gt: 200000 } }
        },
        {
            $project: { _id: 0, Designation: "$_id", totalSalary: 1 }
        }
    ])

    // Query 2: Returns names and _id in upper case and in alphabetical order.
    db.Employee.aggregate([
        {
            $project: {
                _id: 1,
                Name: {
                    $toUpper: { $concat: ["$Name.FName", " ", "$Name.LName"] }
                }
            }
        },
        {
            $sort: { Name: 1 }
        }
    ])

    // 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" }
            }
        }
    ])

    // Index Creation Queries:

    // Query 4: Create Single Field Indexes on Designation field of employee
collection.
    db.Employee.createIndex({ "Designation": 1 })

    // Query 5: To Create Multikey Indexes on Expertise field of employee
collection.
    db.Employee.createIndex({ "Expertise": 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()
])
```