이 MongoDB 쿼리들은 다양한 `update`, `replace`, `aggregate` 연산을 보여줍니다. 각 섹션에 대해 자세히 설명하겠습니다.

---

1. 단일 문서 업데이트

```
db.inventory.updateOne(
   { item: "paper" },
   {
     $set: { "size.uom": "cm", status: "P" },
     $currentDate: { lastModified: true }
   }
)
```

- **설명**: `item` 필드가 `"paper"`인 문서를 하나 찾아서 업데이트합니다.
  - `size.uom`을 `"cm"`로 변경하고, `status`를 `"P"`로 설정합니다.
  - 문서의 `lastModified` 필드를 현재 날짜로 업데이트합니다.

2. 여러 문서 업데이트

```
db.inventory.updateMany(
   { "qty": { $lt: 50 } },
   {
     $set: { "size.uom": "in", status: "P" },
     $currentDate: { lastModified: true }
   }
)
```

- **설명**: `qty` 필드가 50 미만인 모든 문서를 업데이트합니다.
  - `size.uom`을 `"in"`로, `status`를 `"P"`로 설정합니다.
  - `lastModified` 필드를 현재 날짜로 설정합니다.

3. 문서 교체

```
db.inventory.replaceOne(
   { item: "paper" },
   { item: "paper", instock: [ { warehouse: "A", qty: 60 }, { warehouse:
"B", qty: 40 } ] }
)
```

- **설명**: `item` 필드가 `"paper"`인 문서를 찾아 전체 문서를 새 문서로 교체합니다.
  - 기존 문서를 삭제하고, `item: "paper"`와 `instock` 배열이 포함된 새 문서를 삽입합니다.

4. 집계 파이프라인 - 단일 필드 업데이트

```
db.students.updateOne( { _id: 3 }, [ { $set: { "test3": 98, modified:
"$$NOW"} } ] )
```

- **설명**: `_id`가 `3`인 문서의 `test3` 필드를 `98`로 설정하고, `modified` 필드를 현재 날짜로 설정합니다.
  - `$set`과 함께 집계 파이프라인이 사용되어 더 복잡한 연산이 가능합니다.

5. 집계 파이프라인 - 여러 필드 업데이트

```
db.students2.updateMany( {},
  [
    { $replaceRoot: { newRoot:
       { $mergeObjects: [ { quiz1: 0, quiz2: 0, test1: 0, test2: 0 },
"$$ROOT" ] }
    } },
    { $set: { modified: "$$NOW"}  }
  ]
)
```

- **설명**: 모든 문서를 대상으로 `quiz1`, `quiz2`, `test1`, `test2` 필드를 추가하거나 기본값으로 설정합니다. 기존 문서 내용은 유지하면서 새로운 필드를 병합합니다.
  - `modified` 필드를 현재 날짜로 설정합니다.

6. 집계 파이프라인 - 평균 및 등급 계산

```
db.students3.updateMany(
   { },
   [
     { $set: { average : { $trunc: [ { $avg: "$tests" }, 0 ] }, modified:
"$$NOW" } },
     { $set: { grade: { $switch: {
                           branches: [
                               { case: { $gte: [ "$average", 90 ] }, then:
"A" },
                               { case: { $gte: [ "$average", 80 ] }, then:
"B" },
                               { case: { $gte: [ "$average", 70 ] }, then:
```

```
"C" },
                                { case: { $gte: [ "$average", 60 ] }, then:
"D" }
                            ],
                            default: "F"
      } } } }
    ]
)
```

- **설명**: 각 학생 문서에서 `tests` 배열의 평균 점수를 계산하고, 이를 `average` 필드에 저장합니다.
  - `average` 값을 기반으로 등급을 계산하고 `grade` 필드에 저장합니다.
  - `modified` 필드를 현재 날짜로 설정합니다.

7. 배열 필드 업데이트

```
db.students4.updateOne( { _id: 2 },
  [ { $set: { quizzes: { $concatArrays: [ "$quizzes", [ 8, 6 ]  ] } } } ]
)
```

- **설명**: `_id` 가 `2` 인 문서의 `quizzes` 배열에 `[8, 6]` 을 추가합니다.
  - `concatArrays` 를 사용하여 기존 배열에 새로운 요소를 결합합니다.

8. $addFields가 포함된 updateMany

```
db.temperatures.updateMany( { },
  [
    { $addFields: { "tempsF": {
          $map: {
             input: "$tempsC",
             as: "celsius",
             in: { $add: [ { $multiply: ["$$celsius", 9/5 ] }, 32 ] }
          }
    } } }
  ]
)
```

- **설명**: 모든 문서에 대해 `tempsC` 배열에 있는 값을 화씨(`Fahrenheit`)로 변환하여 `tempsF` 필드를 추가합니다.
  - 각 섭씨 값을 화씨로 변환하기 위해 `$map` 과 `$add` 연산을 사용합니다.

9. $let 변수를 사용한 업데이트

```
db.cakeFlavors.updateOne(
   {
      $expr: { $eq: [ "$flavor", "$$targetFlavor" ] }
   },
   [
      {
         $set: { flavor: "$$newFlavor" }
      }
   ],
   {
      let: { targetFlavor: "cherry", newFlavor: "orange" }
   }
)
```

- **설명**: `flavor`가 `"cherry"`인 문서를 찾아 `flavor` 필드를 `"orange"`로 업데이트합니다.
  - `$let` 변수를 사용하여 조건과 업데이트 값을 동적으로 설정합니다.

이 쿼리들은 MongoDB에서 다양한 방식으로 데이터를 업데이트, 교체, 또는 집계하는 방법을 보여주며, 특히 집계 파이프라인을 사용하여 복잡한 업데이트 작업을 수행하는 방법을 강조합니다.