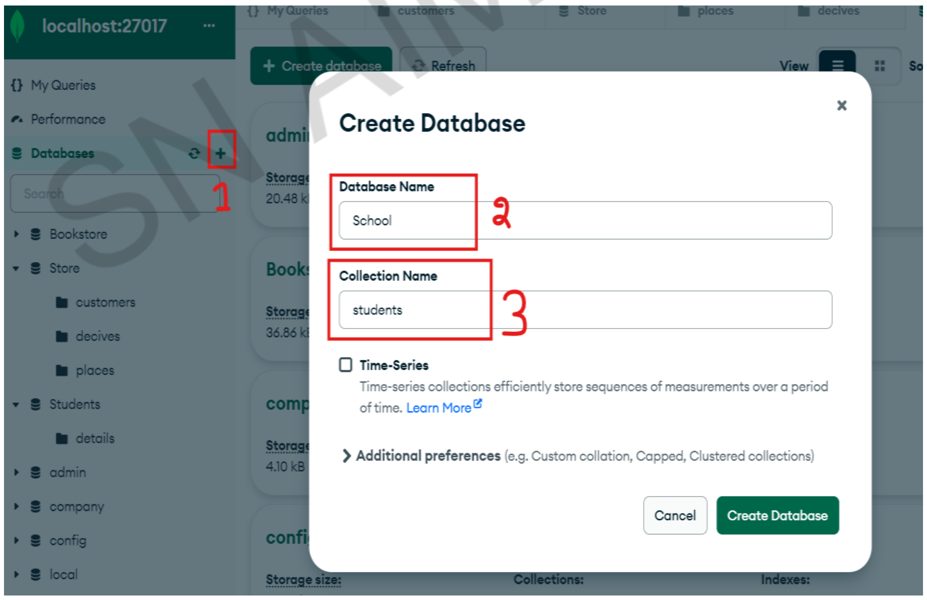
Program 4

Create and demonstrate how projection operators ($, $elematch and $slice) would be used in the MongoDB.

$elemMatch: The $elemMatch operator is used to match documents that contain an array field with at least one element that matches all the specified query criteria.

$slice: The $slice projection operator is used within the projection document to limit the number of elements returned from an array field.

Create a database **School** and collection **students** in Mongo DB IDE**.**



Create Database

Database Name: School

Collection Name: students

☐ Time-Series
Time-series collections efficiently store sequences of measurements over a period of time. Learn More

> Additional preferences (e.g. Custom collation, Capped, Clustered collections)

Cancel | Create Database

Add the following documents in the **details collection** in MongoDB IDE.

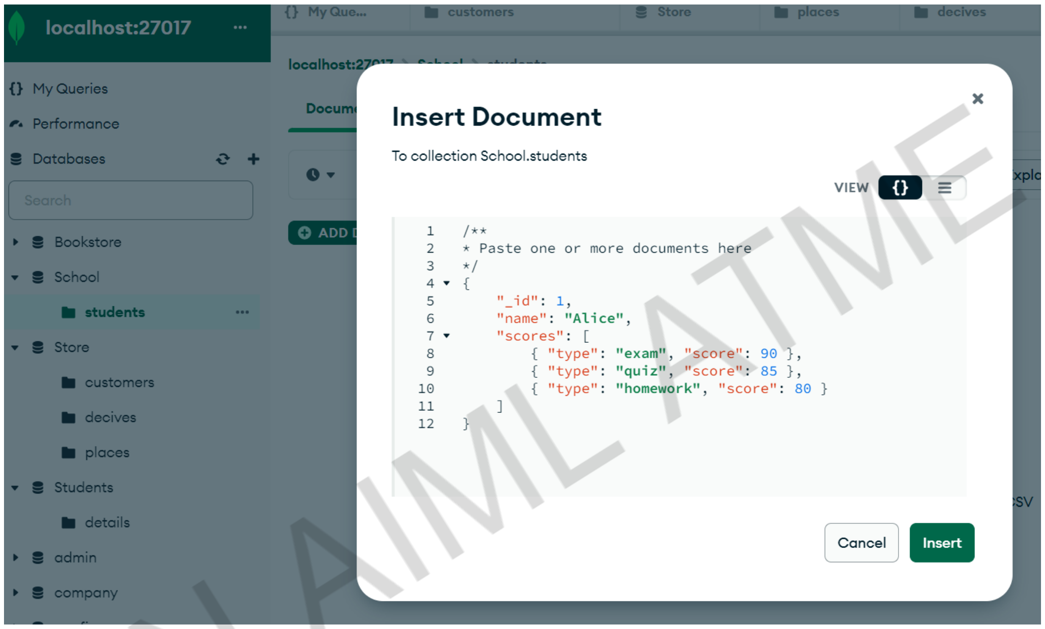
{

"_id": 1,

"name": "Alice",

"scores": [{ "type": "exam", "score": 90 }, { "type": "quiz", "score": 85 }, { "type": "homework", "score": 80 }]

}



```
/**
* Paste one or more documents here
*/
{
    "_id": 1,
    "name": "Alice",
    "scores": [
        { "type": "exam", "score": 90 },
        { "type": "quiz", "score": 85 },
        { "type": "homework", "score": 80 }
    ]
}
```

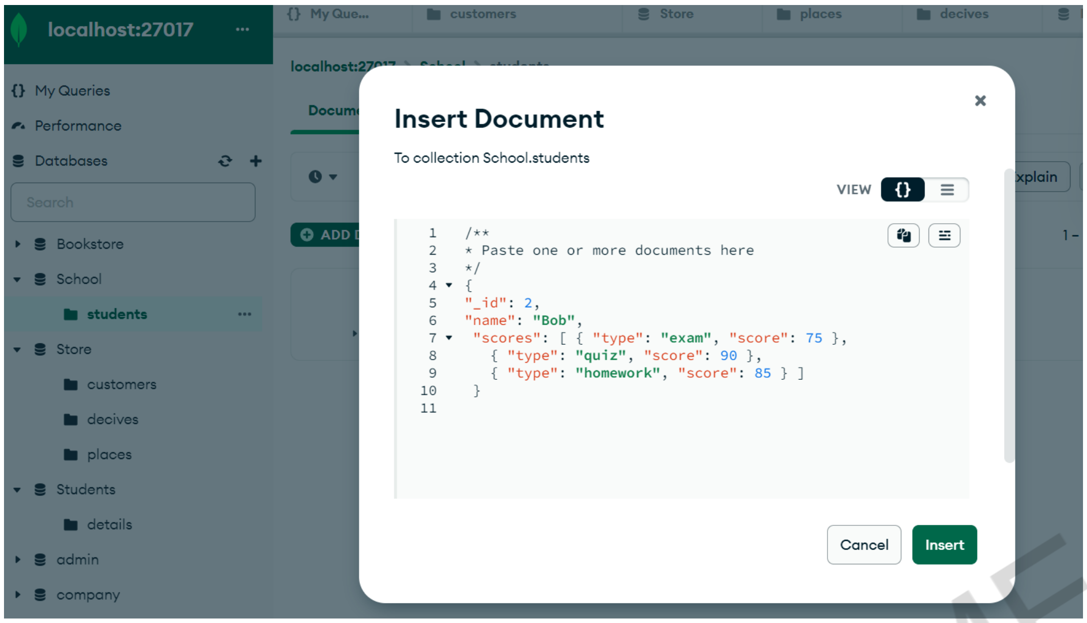
{

"_id": 2,

"name": "Bob",

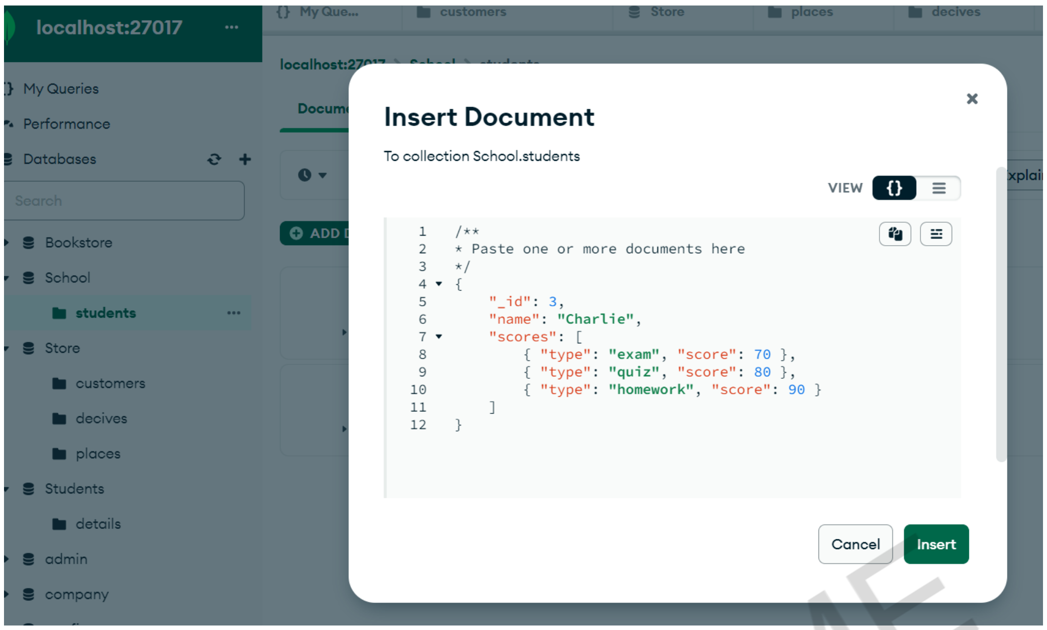
"scores": [{ "type": "exam", "score": 75 }, { "type": "quiz", "score": 90 }, { "type": "homework", "score": 85 }]

}



```
/**
* Paste one or more documents here
*/
{
"_id": 2,
"name": "Bob",
 "scores": [ { "type": "exam", "score": 75 },
   { "type": "quiz", "score": 90 },
   { "type": "homework", "score": 85 } ]
 }
```

```
{
  "_id": 3,
  "name": "Charlie",
  "scores": [
    { "type": "exam", "score": 70 },
    { "type": "quiz", "score": 80 },
    { "type": "homework", "score": 90 }
  ]
}
```

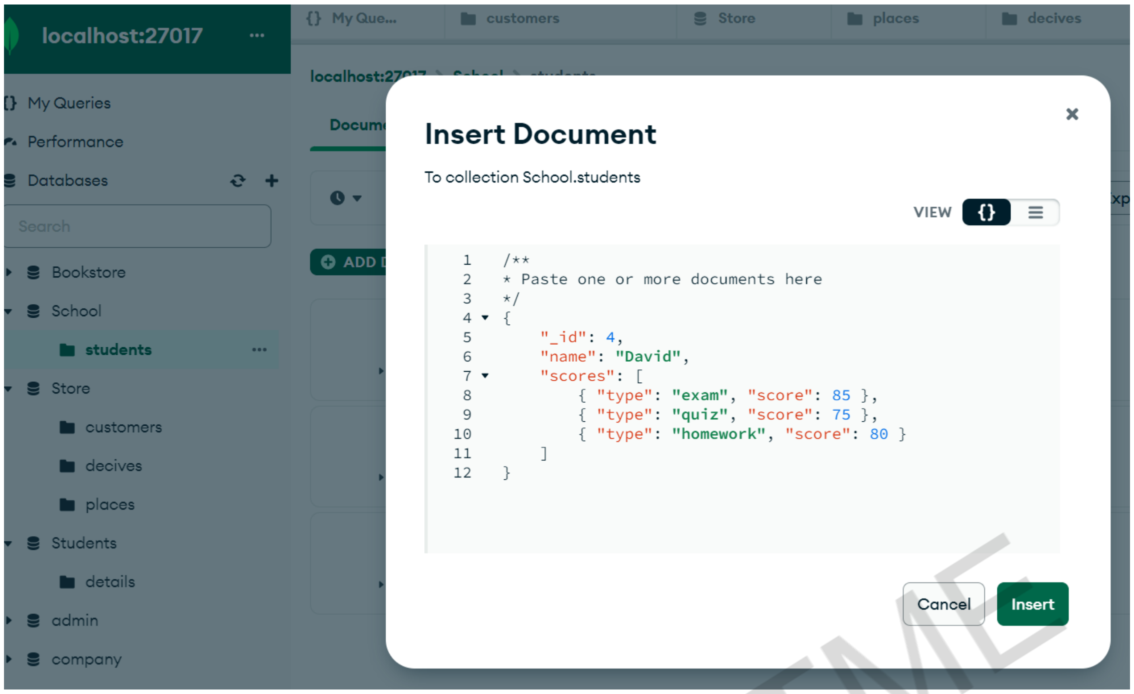


Insert Document

To collection School.students

```
/**
* Paste one or more documents here
*/
{
    "_id": 3,
    "name": "Charlie",
    "scores": [
        { "type": "exam", "score": 70 },
        { "type": "quiz", "score": 80 },
        { "type": "homework", "score": 90 }
    ]
}
```

```
{
  "_id": 4,
  "name": "David",
  "scores": [
    { "type": "exam", "score": 85 },
    { "type": "quiz", "score": 75 },
    { "type": "homework", "score": 80 }
  ]
}
```

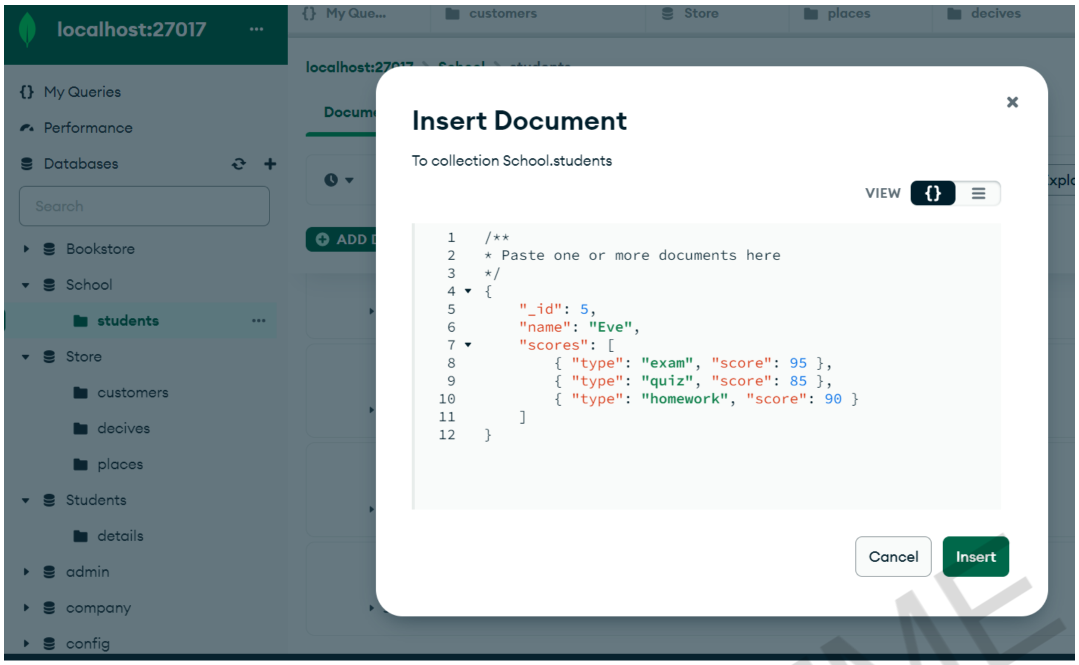


Insert Document

To collection School.students

```
/**
* Paste one or more documents here
*/
{
    "_id": 4,
    "name": "David",
    "scores": [
        { "type": "exam", "score": 85 },
        { "type": "quiz", "score": 75 },
        { "type": "homework", "score": 80 }
    ]
}
```

```
{
   "_id": 5,
   "name": "Eve",
   "scores": [
      { "type": "exam", "score": 95 },
      { "type": "quiz", "score": 85 },
      { "type": "homework", "score": 90 }
   ]
}
```



```
/**
 * Paste one or more documents here
 */
{
    "_id": 5,
    "name": "Eve",
    "scores": [
        { "type": "exam", "score": 95 },
        { "type": "quiz", "score": 85 },
        { "type": "homework", "score": 90 }
    ]
}
```

In Mongo DB Shell

>use School

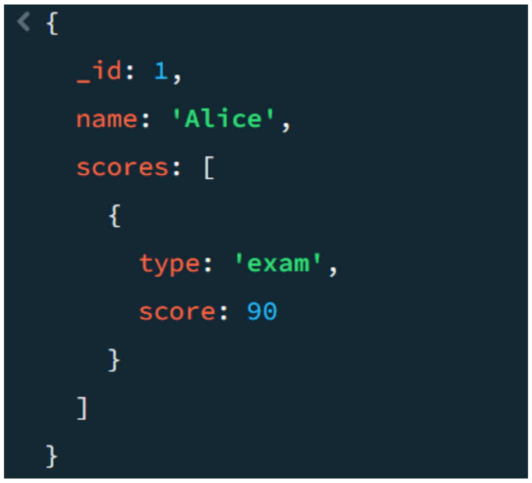
1. $ Operator

The $ operator is used to project a single element from an array that matches a specified condition. For instance, to find the exam score of Alice, you would use:

// To project only the first element in the grades array that is greater than or equal to 85, we can use the following query:

```
db.students.find(
   { "name": "Alice", "scores.type": "exam" },
   { "name": 1, "scores.$": 1 }
)
```

Output:



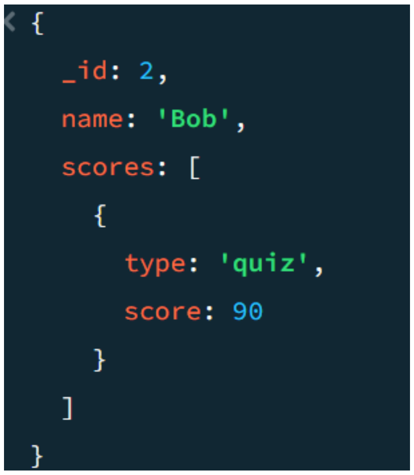
```
{
  _id: 1,
  name: 'Alice',
  scores: [
    {
      type: 'exam',
      score: 90
    }
  ]
}
```

2. $elemMatch Operator

The **$elemMatch** operator is used to project the first matching element from an array. To get the quiz score of Bob, you would use:

```
>db.students.find(
   { "name": "Bob" },
   { "name": 1, "scores": { $elemMatch: { "type": "quiz" } } }
)
```

Output:



```
< {
    _id: 2,
    name: 'Bob',
    scores: [
      {
        type: 'quiz',
        score: 90
      }
    ]
  }
```

3. $slice Operator

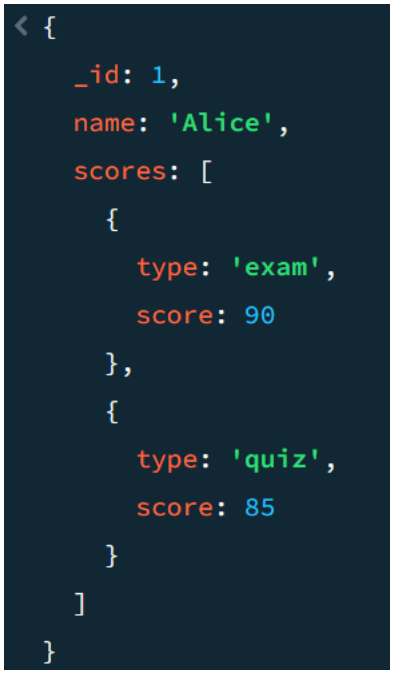
The $slice operator limits the number of array elements included in the query result.

Example

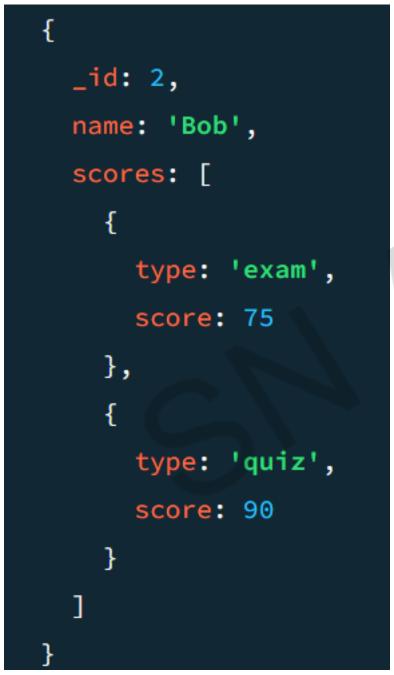
Query to find students with the first two score entries:

```
> db.students.find(
  {},
  { "name": 1, "scores": { $slice: 2 } }
)
```

Output:



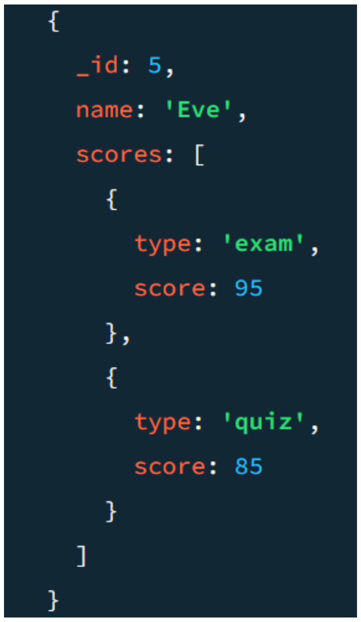
```
< {
    _id: 1,
    name: 'Alice',
    scores: [
      {
        type: 'exam',
        score: 90
      },
      {
        type: 'quiz',
        score: 85
      }
    ]
  }
```



```
  {
    _id: 2,
    name: 'Bob',
    scores: [
      {
        type: 'exam',
        score: 75
      },
      {
        type: 'quiz',
        score: 90
      }
    ]
  }
```

```
{
  _id: 3,
  name: 'Charlie',
  scores: [
    {
      type: 'exam',
      score: 70
    },
    {
      type: 'quiz',
      score: 80
    }
  ]
}
```

```
{
  _id: 4,
  name: 'David',
  scores: [
    {
      type: 'exam',
      score: 85
    },
    {
      type: 'quiz',
      score: 75
    }
  ]
}
```


```
{
  _id: 5,
  name: 'Eve',
  scores: [
    {
      type: 'exam',
      score: 95
    },
    {
      type: 'quiz',
      score: 85
    }
  ]
}
```

Alternatively, you can use negative values with **$slice** to get elements from the end of the array.

Query to find students with the last score entry:

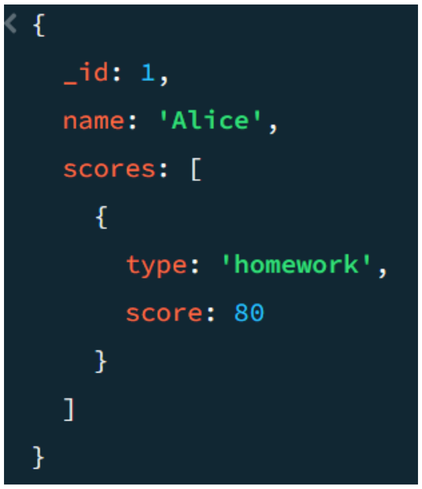
db.students.find(

{},

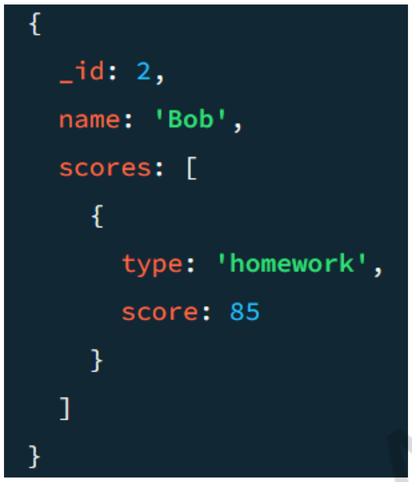
{ "name": 1, "scores": { $slice: -1 } }

)

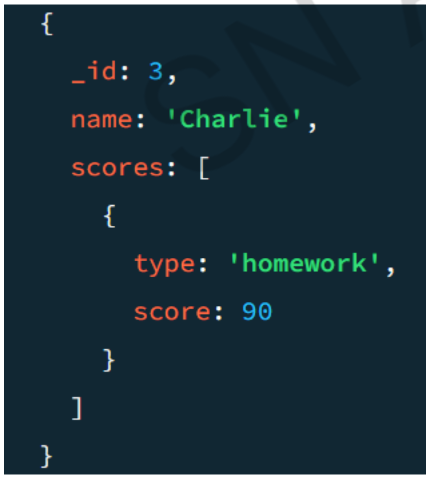
Output:



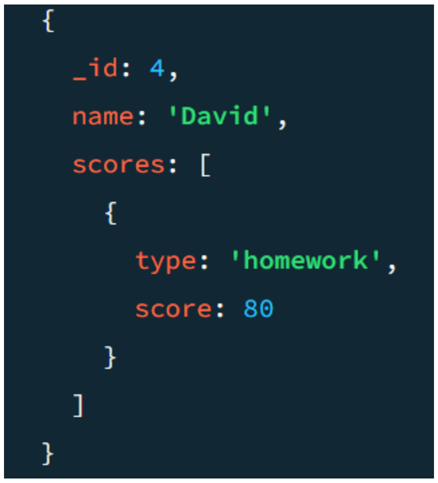
```
{
  _id: 1,
  name: 'Alice',
  scores: [
    {
      type: 'homework',
      score: 80
    }
  ]
}
```



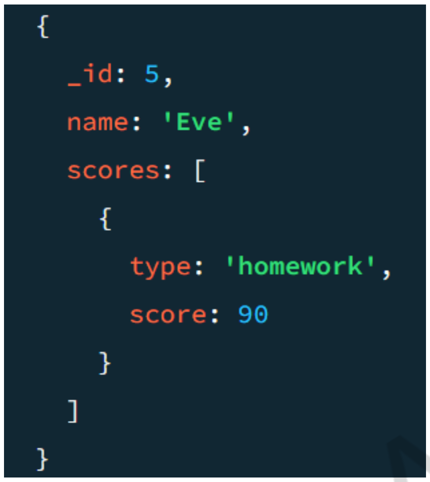
```
{
  _id: 2,
  name: 'Bob',
  scores: [
    {
      type: 'homework',
      score: 85
    }
  ]
}
```



```
{
  _id: 3,
  name: 'Charlie',
  scores: [
    {
      type: 'homework',
      score: 90
    }
  ]
}
```



```
{
  _id: 4,
  name: 'David',
  scores: [
    {
      type: 'homework',
      score: 80
    }
  ]
}
```



```
{
  _id: 5,
  name: 'Eve',
  scores: [
    {
      type: 'homework',
      score: 90
    }
  ]
}
```

Explanation

- **Geospatial Selector**:
    - **$near**: Finds documents near a specified point. Requires a `2dsphere` index on the location field.
    - **$geometry**: Specifies the reference point as a GeoJSON object.
    - **$maxDistance**: Limits the distance from the reference point (in meters).
- **Bitwise Selector**:
    - **$bitsAllSet**: Matches documents where all of the given bit positions are 1.
    - **$bitsAnySet**: Matches documents where any of the given bit positions are 1.

By executing these queries, you can filter documents based on geospatial proximity and bitwise conditions.