Mastering find :

db.collection.find(query, projection)

- **query**: Specifies the filter criteria.
- **projection**: Specifies the fields to include or exclude in the result.

```
project> db.unicorns.find({},{name:1});//getting specific field
[
  { _id: ObjectId('674f9f57c1687a95decdcdfb'), name: 'Horny' },
  { _id: ObjectId('674f9f57c1687a95decdcdfc'), name: 'Aurora' },
  { _id: ObjectId('674f9f57c1687a95decdcdfd'), name: 'Unicrom' },
  { _id: ObjectId('674f9f57c1687a95decdcdfe'), name: 'Roooooodles' },
  { _id: ObjectId('674f9f59c1687a95decdcdff'), name: 'Solnara' }
]
```

```
project> db.unicorns.find({},{name:1,weight:1});
[
  {
    _id: ObjectId('674f9f57c1687a95decdcdfb'),
    name: 'Horny',
    weight: 700
  },
  {
    _id: ObjectId('674f9f57c1687a95decdcdfc'),
    name: 'Aurora',
    weight: 450
  },
  {
    _id: ObjectId('674f9f57c1687a95decdcdfd'),
    name: 'Unicrom',
    weight: 984
  },
  {
    _id: ObjectId('674f9f57c1687a95decdcdfe'),
    name: 'Roooooodles',
    weight: 575
  },
  {
    _id: ObjectId('674f9f59c1687a95decdcdff'),
    name: 'Solnara',
    weight: 550
  }
]
```

```
project> db.unicorns.find({},{_id:0,name:1,weight:1});
[
  { name: 'Horny', weight: 700 },
  { name: 'Aurora', weight: 450 },
  { name: 'Unicrom', weight: 984 },
  { name: 'Roooooodles', weight: 575 },
  { name: 'Solnara', weight: 550 }
]
```

```
project> db.unicorns.find({ $or: [{ gender: 'm' }, { weight: { $lt: 300 } }] });
[
  {
    _id: ObjectId('674f9f57c1687a95decdcdfb'),
    name: 'Horny',
    dob: ISODate('1992-03-13T02:17:00.000Z'),
    loves: [ 'carrot', 'papaya', 'manogo' ],
    weight: 700,
    gender: 'm',
    vampires: 63,
    wight: 700
  },
  {
    _id: ObjectId('674f9f57c1687a95decdcdfd'),
    name: 'Unicrom',
    dob: ISODate('1973-02-09T16:40:00.000Z'),
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId('674f9f57c1687a95decdcdfe'),
    name: 'Roooooodles',
    dob: ISODate('1979-08-18T13:14:00.000Z'),
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  }
]
```

Sort :

```
project> db.unicorns.find({},{name:1,_id:0,weight:1}).sort({weight: 1})//ascending orc
[
  { name: 'Aurora', weight: 450 },
  { name: 'Solnara', weight: 550 },
  { name: 'Roooooodles', weight: 575 },
  { name: 'Horny', weight: 700 },
  { name: 'Unicrom', weight: 984 }
]
```

```
project> db.unicorns.find({},{name:1,_id:0,weight:1}).sort({weight: -1})//descending order
[
  { name: 'Unicrom', weight: 984 },
  { name: 'Horny', weight: 700 },
  { name: 'Roooooodles', weight: 575 },
  { name: 'Solnara', weight: 550 },
  { name: 'Aurora', weight: 450 }
]
```

```
project> db.unicorns.find({},{name:1,_id:0,weight:1,vampires:1}).sort({weight: -1,vampires:1})//descending order
[
  { name: 'Unicrom', weight: 984, vampires: 182 },
  { name: 'Horny', weight: 700, vampires: 63 },
  { name: 'Roooooodles', weight: 575, vampires: 99 },
  { name: 'Solnara', weight: 550, vampires: 80 },
  { name: 'Aurora', weight: 450, vampires: 43 }
]
```

 Primary Sorting (weight: -1):

- The unicorns are primarily sorted by weight in descending order: 500 > 400 > 300.

 Secondary Sorting (vampires: 1):

- Within the group of unicorns with the same weight (e.g., 500), they are further sorted by the vampires field in ascending order.

Projection Output:

- The output only includes the name, weight, and vampires fields, excluding _id.

Limit :

```
project> db.unicorns.find({},{name:1,_id:0,weight:1}).sort({weight: -1,vampires:1}).limit(1)//limit to top record
[ { name: 'Unicrom', weight: 984 } ]
```

```
project> db.unicorns.find({},{name:1,_id:0,weight:1}).sort({weight: -1,vampires:1}).limit(2)//limit to top record
[ { name: 'Unicrom', weight: 984 }, { name: 'Horny', weight: 700 } ]
```

```
project> db.unicorns.find({},{name:1,_id:0,weight:1}).sort({weight: -1,vampires:1}).skip(2)//skip k records and give from k+1 here k=2 1 indexing
[
  { name: 'Roooooodles', weight: 575 },
  { name: 'Solnara', weight: 550 },
  { name: 'Aurora', weight: 450 }
]
```

```
project> db.unicorns.find({},{name:1,_id:0,weight:1}).sort({weight: -1,vampires:1}).limit(1).skip(1)
[ { name: 'Horny', weight: 700 } ]
```

Query Explanation:

db.unicorns.find({}, { name: 1, _id: 0, weight: 1 }).sort({ weight: -1, vampires: 1 }).limit(1).skip(1);

This query performs the following operations:

---

1. find({}):

- **No filter criteria**: All documents in the unicorns collection are considered.

---

2. Projection ({ name: 1, _id: 0, weight: 1 }):

- **Included Fields**:
  - name: The name of each unicorn is included.
  - weight: The weight of each unicorn is included.
- **Excluded Fields**:
  - _id: The _id field is excluded from the result.

---

3. Sorting (sort({ weight: -1, vampires: 1 })):

- The results are sorted in the following order:

- **Primary Sort (weight: -1)**: Unicorns are sorted by weight in descending order (heaviest first).
- **Secondary Sort (vampires: 1)**: If multiple unicorns have the same weight, they are sorted by vampires in ascending order (fewest vampires first).

---

4. Pagination:

- **skip(1)**:
  - Skips the first document in the sorted results.
- **limit(1)**:
  - Limits the output to a single document after skipping.

Execution Flow:

1. **Step 1**: Retrieve all documents from the collection.
2. **Step 2**: Apply the projection to include only the name and weight fields, excluding _id.
3. **Step 3**: Sort the documents by weight in descending order, and for ties, by vampires in ascending order.
4. **Step 4**: Skip the first document in the sorted list.
5. **Step 5**: Return the next document after skipping, limiting the result to one document.

Count:

```
project> db.unicorns.find().count()//count number of records
5
```

```
project> db.unicorns.find({weight:{$gte:700}}).count()//count number of records
2
```

Your query:

javascript

Copy code

db.unicorns.find({weight: {$gte: 700}}, {name: 1, _id: 0}).count();

Explanation:

This query retrieves and counts the number of records in the unicorns collection that meet the following conditions:

---

1. Filter Criteria ({weight: {$gte: 700}}):

- **Condition**: weight must be greater than or equal to 700.

---

2. Projection ({name: 1, _id: 0}):

- **Included Field**: Only the name field is included in the results.
- **Excluded Fields**:
    - _id: The default _id field is excluded.

---

3. Counting Records (.count()):

- Instead of returning the matching documents, the .count() method calculates the total number of documents that satisfy the filter criteria.

Important Notes:

1. The **projection** ({name: 1, _id: 0}) does not affect the count. Only the filter criteria (weight: {$gte: 700}) are considered for counting.
2. **Deprecated .count()**:
    - In modern MongoDB versions, .count() is deprecated. You should use .countDocuments() or .estimatedDocumentCount() for counting:
        - Use .countDocuments() when using a filter.
        - Example:

javascript

Copy code

db.unicorns.countDocuments({weight: {$gte: 700}});

        - This method provides more accurate results.

Sum :

```
project> db.unicorns.aggregate([{$group:{_id:null,tw:{$sum:"$weight"}}}])//calculate weight
[ { _id: null, tw: 3259 } ]
```

 $aggregate: Initiates the aggregation pipeline.

 $group: Groups documents together.

- **_id: null**: Groups all documents into a single group.
- **tw: { $sum: "$weight" }**: Calculates the total sum of the weight field and stores it in a field named tw.

Average :

```
project> db.unicorns.aggregate([{$group:{_id:null,aw:{$avg:"$weight"}}}])//calculate average weight
[ { _id: null, aw: 651.8 } ]
```

```
project> db.unicorns.aggregate([
...   {
...     $group: {
...       _id: "$gender",
...       aw: { $avg: "$weight" }
...     }
...   }
... ]);
[ { _id: 'm', aw: 753 }, { _id: 'f', aw: 500 } ]
```

 $aggregate: Starts the aggregation pipeline.

 $group: Groups documents for aggregation.

- **_id: null**: Groups all documents into a single group (no subdivision).
- **aw: { $avg: "$weight" }**: Calculates the average of the weight field and assigns the result to aw.

Min and max :

```
project> db.unicorns.aggregate([
... {
... $group:{
... _id:null,
... minvalue:{$min:"$weight"},
... maxvalue:{$max:"$weight"}
... }
... }
... ]);
[ { _id: null, minvalue: 450, maxvalue: 984 } ]
```

```
project> db.unicorns.aggregate([
... {
... $group:{
... _id:"$gender",
... minvalue:{$min:"$weight"},
... maxvalue:{$max:"$weight"}
... }
... }
... ])
[
  { _id: 'm', minvalue: 575, maxvalue: 984 },
  { _id: 'f', minvalue: 450, maxvalue: 550 }
]
```

```
project> db.unicorns.aggregate([
...   { $match: { weight: { $gte: 700 } } },
...   { $count: "unicornsAbove700" }
... ]);
[ { unicornsAbove700: 2 } ]
```

Breakdown of the query:

1. **$match**: This stage filters the documents where the weight is greater than or equal to 700 (weight: { $gte: 700 }). It narrows down the set of unicorns to those that have a weight of 700 or more.
2. **$count**: After filtering, the $count stage counts how many documents match the condition and assigns the result to a field called unicornsAbove700.

$push

$addToSet

$first and $last

$mergeObjects

$stdDevPop and $stdDevSamp

$lookup

$unwind

$project

$facet