

Ejercicios de Práctica de MongoDB



Linkia FP Madrid | Centro de Estudios

Ana Chun Gómez de Castro 2ºDAM

ÍNDICE

EJERCICIOS 1

1. [Creación y manipulación de documentos](#)
2. [Actualización de documentos](#)
3. [Eliminación de documentos](#)
4. [Filtros complejos](#)
5. [Ordenación y paginación](#)
6. [Práctica con consultas avanzadas](#)

CREACIÓN DE LA BASE DE DATOS COMPLETA

1. [Crear y poblar la colección libros](#)
2. [Crear y poblar la colección usuarios](#)
3. [Crear y poblar la colección préstamos](#)

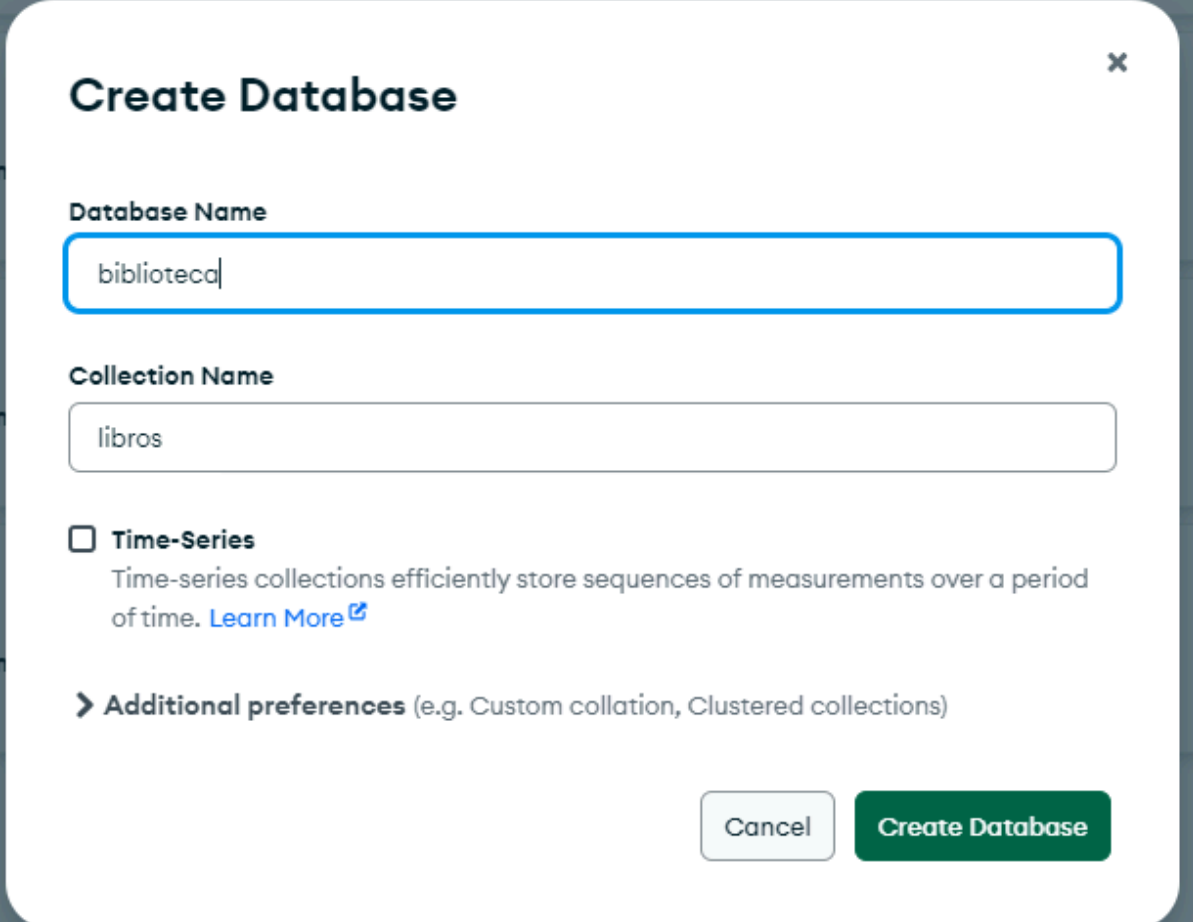
EJERCICIOS 2 PRÁCTICOS AVANZADOS

1. [Índices en MongoDB](#)
2. [Agregaciones](#)
3. [Uniones entre colecciones con \\$lookup](#)
4. [Almacenamiento de resultados con \\$merge](#)

EJERCICIO 1

1. Creación y manipulación de documentos

1.1. Crea una base de datos llamada biblioteca y cambia a ella. 1.2. Crea una colección llamada libros y añade los siguientes documentos:



The screenshot shows the 'Create Database' dialog box in MongoDB Compass. It has a title bar with a close button (X). The dialog contains two input fields: 'Database Name' with the text 'biblioteca' and 'Collection Name' with the text 'libros'. Below these fields is a checkbox labeled 'Time-Series' which is unchecked. To the right of the checkbox is a description: 'Time-series collections efficiently store sequences of measurements over a period of time. [Learn More](#)'. Below this is a section titled 'Additional preferences (e.g. Custom collation, Clustered collections)' with a right-pointing arrow. At the bottom right are two buttons: 'Cancel' and 'Create Database'.

Create Database

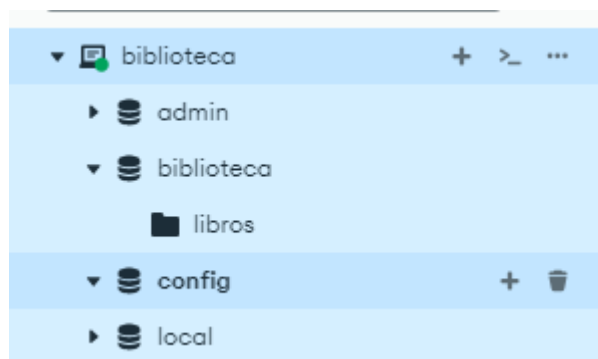
Database Name
biblioteca

Collection Name
libros

☐ **Time-Series**
Time-series collections efficiently store sequences of measurements over a period of time. [Learn More](#)

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

Cancel Create Database



>_MONGOSH

```
> db.libros.insertMany([
  { titulo: "1984", autor: "George Orwell", paginas: 328, año: 1949,
    genero: "Ficción" },
  { titulo: "Cien años de soledad", autor: "Gabriel García Márquez",
    paginas: 417, año: 1967, genero: "Realismo mágico" },
  { titulo: "El Hobbit", autor: "J.R.R. Tolkien", paginas: 310, año:
    1937, genero: "Fantasía" },
  { titulo: "Don Quijote de la Mancha", autor: "Miguel de Cervantes",
    paginas: 863, año: 1605, genero: "Novela" },
  { titulo: "Orgullo y prejuicio", autor: "Jane Austen", paginas:
    279, año: 1813, genero: "Romance" },
  { titulo: "Fahrenheit 451", autor: "Ray Bradbury", paginas: 249,
    año: 1953, genero: "Ciencia ficción" },
  { titulo: "Moby Dick", autor: "Herman Melville", paginas: 635, año:
    1851, genero: "Aventura" },
  { titulo: "Ulises", autor: "James Joyce", paginas: 730, año: 1922,
    genero: "Modernismo" }
])
< {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('67ab0f58d15d407fd61bd742'),
    '1': ObjectId('67ab0f58d15d407fd61bd743'),
    '2': ObjectId('67ab0f58d15d407fd61bd744'),
    '3': ObjectId('67ab0f58d15d407fd61bd745'),
    '4': ObjectId('67ab0f58d15d407fd61bd746'),
    '5': ObjectId('67ab0f58d15d407fd61bd747'),
    '6': ObjectId('67ab0f58d15d407fd61bd748'),
    '7': ObjectId('67ab0f58d15d407fd61bd749')
  }
}
biblioteca> |
```

1.3 Agrega un nuevo libro titulado “Crimen y castigo” de Fiódor Dostoyevski con 671 páginas, publicado en 1866.

Para añadir, se usa el insertOne ya que nos interesa añadir solo un libro.

```
>_ mongosh: biblioteca +

>_MONGOSH

> db.libros.insertOne({titulo: "Crimen y castigo", autor: "Fiódor Dostoyevski", paginas: 671, año: 1866})
< {
  acknowledged: true,
  insertedId: ObjectId('67ab10fdd15d407fd61bd74a')
}
biblioteca>
```

1.4. Consulta todos los documentos de la colección libros.

```
> db["libros"].find()
```

```
>_MONGOSH
use biblioteca
switched to db biblioteca
> db["libros"].find()
< [
  {
    _id: ObjectId('67ab0d7829f4cf74558f8be1'),
    titulo: '1984',
    autor: 'George Orwell',
    paginas: 328,
    año: 1949,
    genero: 'Ficción'
  },
  {
    _id: ObjectId('67ab0d7829f4cf74558f8be2'),
    titulo: 'Cien años de soledad',
    autor: 'Gabriel García Márquez',
    paginas: 417,
    año: 1967,
    genero: 'Realismo mágico'
  },
  {
    _id: ObjectId('67ab0d7829f4cf74558f8be3'),
    titulo: 'El Hobbit',
    autor: 'J.R.R. Tolkien',
    paginas: 310,
    año: 1937,
    genero: 'Fantasía'
  },
  {
    _id: ObjectId('67ab0d7829f4cf74558f8be4'),
    titulo: 'Don Quijote de la Mancha',
    autor: 'Miguel de Cervantes',
    paginas: 863,
    año: 1605,
    genero: 'Novela'
  },
  {
    _id: ObjectId('67ab0d7829f4cf74558f8be5'),
    titulo: 'Orgullo y prejuicio',
    autor: 'Jane Austen',
    paginas: 279,
    año: 1813,
    genero: 'Romance'
  },
  {
    _id: ObjectId('67ab0d7829f4cf74558f8be6'),
    titulo: 'Fahrenheit 451',
    autor: 'Ray Bradbury',
    paginas: 249,
    año: 1953,
    genero: 'Ciencia ficción'
  },
  {
    _id: ObjectId('67ab0d7829f4cf74558f8be7'),
    titulo: 'Moby Dick',
    autor: 'Herman Melville',
    paginas: 635,
    año: 1851,
    genero: 'Aventura'
  },
  {
    _id: ObjectId('67ab0d7829f4cf74558f8be8'),
    titulo: 'Ulises',
    autor: 'James Joyce',
    paginas: 720,
    año: 1922,
    genero: 'Modernismo'
  }
]
```

```
{
  _id: ObjectId('67ab0c4dce74b5f663390818'),
  titulo: 'Crimen y castigo',
  autor: 'Fiódor Dostoyevski',
  paginas: 671,
  año: 1866
}
biblioteca>
```

1.5. Encuentra los libros cuyo autor sea "George Orwell".

```
>_ mongosh: biblioteca +
```

```
>_MONGOSH
```

```
> db.libros.find({autor: "George Orwell"})
```

```
< {
```

```
  _id: ObjectId('67ab0f58d15d407fd61bd742'),
```

```
  titulo: '1984',
```

```
  autor: 'George Orwell',
```

```
  paginas: 328,
```

```
  'año': 1949,
```

```
  genero: 'Ficción'
```

```
}
```

```
biblioteca> |
```

1.6. Encuentra los libros publicados después del año 1950.

```
>_ mongosh: biblioteca +
```

```
>_MONGOSH
```

```
> db.libros.find({año:{$gt: 1950}})
```

```
< {
```

```
  _id: ObjectId('67ab0f58d15d407fd61bd743'),
```

```
  titulo: 'Cien años de soledad',
```

```
  autor: 'Gabriel García Márquez',
```

```
  paginas: 417,
```

```
  'año': 1967,
```

```
  genero: 'Realismo mágico'
```

```
}
```

```
{
```

```
  _id: ObjectId('67ab0f58d15d407fd61bd747'),
```

```
  titulo: 'Fahrenheit 451',
```

```
  autor: 'Ray Bradbury',
```

```
  paginas: 249,
```

```
  'año': 1953,
```

```
  genero: 'Ciencia ficción'
```

```
}
```

```
}
```

```
biblioteca> |
```

2. Actualización de documentos

2.1 Modifica el número de páginas del libro Hobbit a 320 páginas

```
> db.libros.find({titulo: "El Hobbit"})
< {
  _id: ObjectId('67ab0f58d15d407fd61bd744'),
  titulo: 'El Hobbit',
  autor: 'J.R.R. Tolkien',
  paginas: 310,
  'año': 1937,
  genero: 'Fantasía'
}
```

```
> db.libros.updateOne({titulo: "El Hobbit"}, {$set: {paginas: 320}})
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
> db.libros.find({titulo: "El Hobbit"})
< {
  _id: ObjectId('67ab0f58d15d407fd61bd744'),
  titulo: 'El Hobbit',
  autor: 'J.R.R. Tolkien',
  paginas: 320,
  'año': 1937,
  genero: 'Fantasía'
}
```

2.2. Incrementa en 10 páginas todos los libros publicados antes del año 1950

Primero muestro los libros publicados antes del año de 1950

```
>_MONGOSH
> db.libros.find({año: {$lte: 1950}})
< {
  _id: ObjectId('67ad15d57bde80176200d6c9'),
  título: '1984',
  autor: 'George Orwell',
  páginas: 328,
  'año': 1949,
  genero: 'Ficción'
}
{
  _id: ObjectId('67ad15d57bde80176200d6cb'),
  título: 'El Hobbit',
  autor: 'J.R.R. Tolkien',
  páginas: 320,
  'año': 1937,
  genero: 'Fantasía'
}
{
  _id: ObjectId('67ad15d57bde80176200d6cc'),
  título: 'Don Quijote de la Mancha',
  autor: 'Miguel de Cervantes',
  páginas: 863,
  'año': 1605,
  genero: 'Novela'
}
{
  _id: ObjectId('67ad15d57bde80176200d6cd'),
  título: 'Orgullo y prejuicio',
  autor: 'Jane Austen',
  páginas: 279,
  'año': 1813,
  genero: 'Romance'
}
{
  _id: ObjectId('67ad15d57bde80176200d6cf'),
  título: 'Moby Dick',
  autor: 'Herman Melville',
  páginas: 635,
  'año': 1851,
  genero: 'Aventura'
}
{
  _id: ObjectId('67ad15d57bde80176200d6d0'),
  título: 'Ulises',
  autor: 'James Joyce',
  páginas: 730,
  'año': 1922,
  genero: 'Modernismo'
}
{
  _id: ObjectId('67ad16f17bde80176200d6d1'),
  título: 'Crimen y castigo',
  autor: 'Fiódor Dostoyevski',
  páginas: 671,
  'año': 1866
}
biblioteca>
```


Después incrementamos 10 páginas a todos

```
> db.libros.updateMany({año: {$lte: 1950}}, {$inc: {paginas: 10}})
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 7,
  modifiedCount: 7,
  upsertedCount: 0
}
```

Este código es para comprobar que se ha incrementado

```
> db.libros.find({año: {$lte: 1950}})
< {
  _id: ObjectId('67ad15d57bde80176200d6c9'),
  titulo: '1984',
  autor: 'George Orwell',
  paginas: 338,
  'año': 1949,
  genero: 'Ficción'
}
{
  _id: ObjectId('67ad15d57bde80176200d6cb'),
  titulo: 'El Hobbit',
  autor: 'J.R.R. Tolkien',
  paginas: 330,
  'año': 1937,
  genero: 'Fantasía'
}
{
  _id: ObjectId('67ad15d57bde80176200d6cc'),
  titulo: 'Don Quijote de la Mancha',
  autor: 'Miguel de Cervantes',
  paginas: 873,
  'año': 1605,
  genero: 'Novela'
}
{
  _id: ObjectId('67ad15d57bde80176200d6cd'),
  titulo: 'Orgullo y prejuicio',
  autor: 'Jane Austen',
  paginas: 289,
  'año': 1813,
  genero: 'Romance'
}
{
  _id: ObjectId('67ad15d57bde80176200d6cf'),
  titulo: 'Moby Dick',
  autor: 'Herman Melville',
  paginas: 645,
  'año': 1851,
  genero: 'Aventura'
}
{
  _id: ObjectId('67ad15d57bde80176200d6d0'),
  titulo: 'Ulises',
  autor: 'James Joyce',
  paginas: 740,
  'año': 1922,
  genero: 'Modernismo'
}
{
  _id: ObjectId('67ad16f17bde80176200d6d1'),
  titulo: 'Crimen y castigo',
  autor: 'Fiódor Dostoyevski',
  paginas: 681,
  'año': 1866
}
```

2.3. Cambia el campo año por fecha_publicacion en todos los documentos

```
> db["libros"].find()
< {
  _id: ObjectId('67ad15d57bde88176288d6c9'),
  título: '1984',
  autor: 'George Orwell',
  páginas: 338,
  'año': 1949,
  genero: 'Ficción'
}
{
  _id: ObjectId('67ad15d57bde88176288d6ca'),
  título: 'Cien años de soledad',
  autor: 'Gabriel García Márquez',
  páginas: 417,
  'año': 1957,
  genero: 'Realismo mágico'
}
{
  _id: ObjectId('67ad15d57bde88176288d6cb'),
  título: 'El Hobbit',
  autor: 'J.R.R. Tolkien',
  páginas: 338,
  'año': 1937,
  genero: 'Fantasía'
}
{
  _id: ObjectId('67ad15d57bde88176288d6cc'),
  título: 'Don Quijote de la Mancha',
  autor: 'Miguel de Cervantes',
  páginas: 873,
  'año': 1605,
  genero: 'Novela'
}
{
  _id: ObjectId('67ad15d57bde88176288d6cd'),
  título: 'Orgullo y prejuicio',
  autor: 'Jane Austen',
  páginas: 289,
  'año': 1813,
  genero: 'Romance'
}
{
  _id: ObjectId('67ad15d57bde88176288d6ce'),
  título: 'Fahrenheit 451',
  autor: 'Ray Bradbury',
  páginas: 249,
  'año': 1953,
  genero: 'Ciencia ficción'
}
{
  _id: ObjectId('67ad15d57bde88176288d6cf'),
  título: 'Moby Dick',
  autor: 'Herman Melville',
  páginas: 645,
  'año': 1851,
  genero: 'Aventura'
}
{
  _id: ObjectId('67ad15d57bde88176288d6d0'),
  título: 'Ulises',
  autor: 'James Joyce',
  páginas: 748,
  'año': 1922,
  genero: 'Modernismo'
}
```

Vemos que ahora está el campo año.

Para cambiarlo se usa el rename, y dejando el hueco {} que muestra los campos.

```
> db.libros.updateMany({}, {$rename: {"año": "fecha_publicacion"}})
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 9,
  modifiedCount: 9,
  upsertedCount: 0
}
```

```

> db["libros"].find()
< {
  _id: ObjectId('67ad15d57bde80176200d6c9'),
  titulo: '1984',
  autor: 'George Orwell',
  paginas: 338,
  genero: 'Ficción',
  fecha_publicacion: 1949
}
{
  _id: ObjectId('67ad15d57bde80176200d6ca'),
  titulo: 'Cien años de soledad',
  autor: 'Gabriel García Márquez',
  paginas: 417,
  genero: 'Realismo mágico',
  fecha_publicacion: 1967
}
{
  _id: ObjectId('67ad15d57bde80176200d6cb'),
  titulo: 'El Hobbit',
  autor: 'J.R.R. Tolkien',
  paginas: 330,
  genero: 'Fantasía',
  fecha_publicacion: 1937
}
{
  _id: ObjectId('67ad15d57bde80176200d6cc'),
  titulo: 'Don Quijote de la Mancha',
  autor: 'Miguel de Cervantes',
  paginas: 873,
  genero: 'Novela',
  fecha_publicacion: 1605
}
{
  _id: ObjectId('67ad15d57bde80176200d6cd'),
  titulo: 'Orgullo y prejuicio',
  autor: 'Jane Austen',
  paginas: 289,
  genero: 'Romance',
  fecha_publicacion: 1813
}

```

Comprobamos que en los campos de páginas se ha incrementado las páginas.

3. Eliminación de documentos

3.1 Elimina el libro cuyo título sea “1984”

Buscamos el título con el nombre 1984

```
> db.libros.find({titulo: "1984"})  
< {  
  _id: ObjectId('67ad15d57bde80176200d6c9'),  
  titulo: '1984',  
  autor: 'George Orwell',  
  paginas: 338,  
  genero: 'Ficción',  
  fecha_publicacion: 1949  
}
```

Para eliminarlo, usamos el deleteOne.

```
> db.libros.deleteOne({titulo: "1984"})  
< {  
  acknowledged: true,  
  deletedCount: 1  
}
```

Y comprobamos que se ha eliminado.

```
>_MONGOSH
```

```
> db.libros.find({titulo: "1984"})
```

```
<
```

```
biblioteca> |
```

3.2 Elimina todos los libros con menos de 350 páginas.

Primero consultamos los libros que tienen menos de 350 páginas

```
> db.libros.find({paginas: {$lte: 350}})
< {
  _id: ObjectId('67ad15d57bde80176200d6cb'),
  titulo: 'El Hobbit',
  autor: 'J.R.R. Tolkien',
  paginas: 330,
  genero: 'Fantasía',
  fecha_publicacion: 1937
}
{
  _id: ObjectId('67ad15d57bde80176200d6cd'),
  titulo: 'Orgullo y prejuicio',
  autor: 'Jane Austen',
  paginas: 289,
  genero: 'Romance',
  fecha_publicacion: 1813
}
{
  _id: ObjectId('67ad15d57bde80176200d6ce'),
  titulo: 'Fahrenheit 451',
  autor: 'Ray Bradbury',
  paginas: 249,
  genero: 'Ciencia ficción',
  fecha_publicacion: 1953
}
biblioteca>
```

Ahora eliminamos los libros con menos de 350 páginas

```
> db.libros.deleteMany({paginas: {$lte: 350}})
< {
  acknowledged: true,
  deletedCount: 3
}
biblioteca> |
```

Comprobamos que se ha eliminado y vemos que no hay datos.

```
> db.libros.find({paginas: {$lte: 350}})
<
biblioteca> |
```

4. Filtros complejos

4.1 Encuentra los libros cuyo título sea “El Hobbit” o “Cien años de soledad”

No devuelve el título “El Hobbit” porque en el anterior ejercicio me pedía que eliminase los libros con menos de 350 páginas.

```
> _MONGOSH
> db.libros.find({titulo: {$in: ["El Hobbit", "Cien años de soledad"]}})
< {
  _id: ObjectId('67ad15d57bde80176200d6ca'),
  titulo: 'Cien años de soledad',
  autor: 'Gabriel García Márquez',
  paginas: 417,
  genero: 'Realismo mágico',
  fecha_publicacion: 1967
}
biblioteca>
```

4.2 Encuentra los libros que no fueron escritos por “Gabriel García Márquez.”

Realiza una consulta en la colección libros para obtener todos los documentos donde el campo autor no sea igual a "Gabriel García Márquez". Utiliza el \$ne que significa no igual para filtrar los resultados.

```
>_MONGOSH
> db.libros.find({autor: {$ne: "Gabriel García Márquez"}})
< {
  _id: ObjectId('67ad15d57bde80176200d6cc'),
  titulo: 'Don Quijote de la Mancha',
  autor: 'Miguel de Cervantes',
  paginas: 873,
  genero: 'Novela',
  fecha_publicacion: 1605
}
{
  _id: ObjectId('67ad15d57bde80176200d6cf'),
  titulo: 'Moby Dick',
  autor: 'Herman Melville',
  paginas: 645,
  genero: 'Aventura',
  fecha_publicacion: 1851
}
{
  _id: ObjectId('67ad15d57bde80176200d6d0'),
  titulo: 'Ulises',
  autor: 'James Joyce',
  paginas: 740,
  genero: 'Modernismo',
  fecha_publicacion: 1922
}
{
  _id: ObjectId('67ad16f17bde80176200d6d1'),
  titulo: 'Crimen y castigo',
  autor: 'Fiódor Dostoyevski',
  paginas: 681,
  fecha_publicacion: 1866
}
biblioteca> |
```

4.3 Encuentra los libros con más de 300 páginas y que hayan sido publicados después de 1950.

Usamos el `$gte` para consultar mayores o iguales a 300 páginas y lo mismo con la `fecha_publicacion`

```
>_MONGOSH
```

```
> db.libros.find({ paginas: {$gte: 300}, fecha_publicacion: {$gte: 1950} })
```

```
< {
```

```
  _id: ObjectId('67ad15d57bde80176200d6ca'),
```

```
  titulo: 'Cien años de soledad',
```

```
  autor: 'Gabriel García Márquez',
```

```
  paginas: 417,
```

```
  genero: 'Realismo mágico',
```

```
  fecha_publicacion: 1967
```

```
}
```

```
biblioteca> |
```


5. Ordenación y paginación

5.1. Obtén una lista de libros ordenados por título en orden alfabético.

Para ordenarlo se usa el `sort`, con el nombre del campo y el 1

```
>_MONGOSH
> db.libros.find().sort({titulo: 1})
< {
  _id: ObjectId('67ad15d57bde80176200d6ca'),
  título: 'Cien años de soledad',
  autor: 'Gabriel García Márquez',
  paginas: 417,
  genero: 'Realismo mágico',
  fecha_publicacion: 1967
}
{
  _id: ObjectId('67ad16f17bde80176200d6d1'),
  título: 'Crimen y castigo',
  autor: 'Fiódor Dostoyevski',
  paginas: 681,
  fecha_publicacion: 1866
}
{
  _id: ObjectId('67ad15d57bde80176200d6cc'),
  título: 'Don Quijote de la Mancha',
  autor: 'Miguel de Cervantes',
  paginas: 873,
  genero: 'Novela',
  fecha_publicacion: 1605
}
{
  _id: ObjectId('67ad15d57bde80176200d6cf'),
  título: 'Moby Dick',
  autor: 'Herman Melville',
  paginas: 645,
  genero: 'Aventura',
  fecha_publicacion: 1851
}
{
  _id: ObjectId('67ad15d57bde80176200d6d0'),
  título: 'Ulises',
  autor: 'James Joyce',
  paginas: 740,
  genero: 'Modernismo',
  fecha_publicacion: 1922
}
biblioteca>
```

5.2 Obtén los libros ordenados por fecha_publicacion en orden descendente.

Igual con el `sort` pero como es descendente se pone el `-1`

```
>_MONGOSH
> db.libros.find().sort({fecha_publicacion: -1})
< {
  _id: ObjectId('67ad15d57bde80176200d6ca'),
  titulo: 'Cien años de soledad',
  autor: 'Gabriel García Márquez',
  paginas: 417,
  genero: 'Realismo mágico',
  fecha_publicacion: 1967
}
{
  _id: ObjectId('67ad15d57bde80176200d6d0'),
  titulo: 'Ulises',
  autor: 'James Joyce',
  paginas: 740,
  genero: 'Modernismo',
  fecha_publicacion: 1922
}
{
  _id: ObjectId('67ad16f17bde80176200d6d1'),
  titulo: 'Crimen y castigo',
  autor: 'Fiódor Dostoyevski',
  paginas: 681,
  fecha_publicacion: 1866
}
{
  _id: ObjectId('67ad15d57bde80176200d6cf'),
  titulo: 'Moby Dick',
  autor: 'Herman Melville',
  paginas: 645,
  genero: 'Aventura',
  fecha_publicacion: 1851
}
{
  _id: ObjectId('67ad15d57bde80176200d6cc'),
  titulo: 'Don Quijote de la Mancha',
  autor: 'Miguel de Cervantes',
  paginas: 873,
  genero: 'Novela',
  fecha_publicacion: 1605
}
biblioteca>|
```

5.3. Devuelve solo los primeros dos libros de la colección.

Como queremos solo los dos primeros, usamos el `limit`.

```
>_MONGOSH
```

```
> db.libros.find().limit(2)
```

```
< {
```

```
  _id: ObjectId('67ad15d57bde80176200d6ca'),
```

```
  titulo: 'Cien años de soledad',
```

```
  autor: 'Gabriel García Márquez',
```

```
  paginas: 417,
```

```
  genero: 'Realismo mágico',
```

```
  fecha_publicacion: 1967
```

```
}
```

```
{
```

```
  _id: ObjectId('67ad15d57bde80176200d6cc'),
```

```
  titulo: 'Don Quijote de la Mancha',
```

```
  autor: 'Miguel de Cervantes',
```

```
  paginas: 873,
```

```
  genero: 'Novela',
```

```
  fecha_publicacion: 1605
```

```
}
```

```
biblioteca>|
```

5.4. Omite los dos primeros libros y devuelve los siguientes.

Para omitirlo, se usa el `skip`.

```
>_MONGOSH
```

```
> db.libros.find().skip(2)
```

```
< {
  _id: ObjectId('67ad15d57bde80176200d6cf'),
  titulo: 'Moby Dick',
  autor: 'Herman Melville',
  paginas: 645,
  genero: 'Aventura',
  fecha_publicacion: 1851
}
{
  _id: ObjectId('67ad15d57bde80176200d6d0'),
  titulo: 'Ulises',
  autor: 'James Joyce',
  paginas: 740,
  genero: 'Modernismo',
  fecha_publicacion: 1922
}
{
  _id: ObjectId('67ad16f17bde80176200d6d1'),
  titulo: 'Crimen y castigo',
  autor: 'Fiódor Dostoyevski',
  paginas: 681,
  fecha_publicacion: 1866
}
```

```
biblioteca>
```

6. Práctica con consultas avanzadas

6.1. Encuentra los libros que contienen exactamente el campo autor.

Se muestran libros con autores.

```
>_MONGOSH

> db.libros.find({autor: {$exists: true}})
< {
  _id: ObjectId('67ad15d57bde80176200d6ca'),
  titulo: 'Cien años de soledad',
  autor: 'Gabriel García Márquez',
  paginas: 417,
  genero: 'Realismo mágico',
  fecha_publicacion: 1967
}
{
  _id: ObjectId('67ad15d57bde80176200d6cc'),
  titulo: 'Don Quijote de la Mancha',
  autor: 'Miguel de Cervantes',
  paginas: 873,
  genero: 'Novela',
  fecha_publicacion: 1605
}
{
  _id: ObjectId('67ad15d57bde80176200d6cf'),
  titulo: 'Moby Dick',
  autor: 'Herman Melville',
  paginas: 645,
  genero: 'Aventura',
  fecha_publicacion: 1851
}
{
  _id: ObjectId('67ad15d57bde80176200d6d0'),
  titulo: 'Ulises',
  autor: 'James Joyce',
  paginas: 740,
  genero: 'Modernismo',
  fecha_publicacion: 1922
}
{
  _id: ObjectId('67ad16f17bde80176200d6d1'),
  titulo: 'Crimen y castigo',
  autor: 'Fiódor Dostoyevski',
  paginas: 681,
  fecha_publicacion: 1866
}
biblioteca>
```

6.2. Encuentra los libros donde paginas es mayor que fecha_publicacion.

No aparece porque no hay datos.
El `expr` se usa para compararlo.

```
>_MONGOSH
```

```
> db.libros.find({$expr: {$gte: ["$paginas", "$fecha_publicacion"]}})
<
biblioteca>
```

6.3 Encuentra los libros que contienen una clave editorial, incluso si aún no existe en la base de datos.

Este fragmento de código busca los documentos en libros donde el campo editorial exista.

```
>_MONGOSH
```

```
> db.libros.find({ editorial: { $exists: true } })
<
biblioteca> |
```

CREACIÓN DE LA BASE DE DATOS COMPLETA

1. Crear y poblar la colección libros

```
>_MONGOSH

> use biblioteca
< switched to db biblioteca
> db["libros"].find()
<
> db.libros.insertMany([
  { titulo: "1984", autor: "George Orwell", paginas: 328, año: 1949,
    genero: "Ficción" },
  { titulo: "Cien años de soledad", autor: "Gabriel García Márquez",
    paginas: 417, año: 1967, genero: "Realismo mágico" },
  { titulo: "El Hobbit", autor: "J.R.R. Tolkien", paginas: 310, año:
    1937, genero: "Fantasía" },
  { titulo: "Don Quijote de la Mancha", autor: "Miguel de Cervantes",
    paginas: 863, año: 1605, genero: "Novela" },
  { titulo: "Orgullo y prejuicio", autor: "Jane Austen", paginas:
    279, año: 1813, genero: "Romance" },
  { titulo: "Fahrenheit 451", autor: "Ray Bradbury", paginas: 249,
    año: 1953, genero: "Ciencia ficción" },
  { titulo: "Moby Dick", autor: "Herman Melville", paginas: 635, año:
    1851, genero: "Aventura" },
  { titulo: "Ulises", autor: "James Joyce", paginas: 730, año: 1922,
    genero: "Modernismo" }
])
< {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('67b23861baaf179f107bf485'),
    '1': ObjectId('67b23861baaf179f107bf486'),
    '2': ObjectId('67b23861baaf179f107bf487'),
    '3': ObjectId('67b23861baaf179f107bf488'),
    '4': ObjectId('67b23861baaf179f107bf489'),
    '5': ObjectId('67b23861baaf179f107bf48a'),
    '6': ObjectId('67b23861baaf179f107bf48b'),
    '7': ObjectId('67b23861baaf179f107bf48c')
  }
}
biblioteca>
```

2. Crear y poblar la colección usuarios

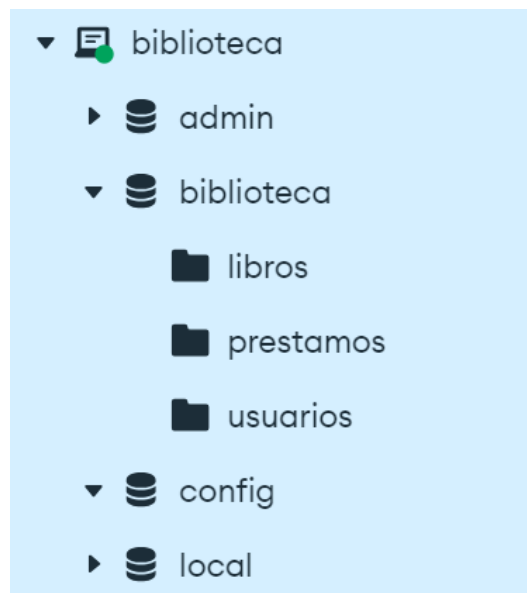
```
>_MONGOSH

> use biblioteca
< switched to db biblioteca
> db["usuarios"].find()
<
> db.usuarios.insertMany([
  { nombre: "Carlos Pérez", direccion: "Calle 1, Madrid", telefono:
"123456789" },
  { nombre: "Ana Gómez", direccion: "Calle 2, Barcelona", telefono:
"987654321" },
  { nombre: "Luis Fernández", direccion: "Calle 3, Valencia",
telefono: "456123789" },
  { nombre: "Marta López", direccion: "Calle 4, Sevilla", telefono:
"321654987" }
])
< {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('67b238b223576e402a7173a8'),
    '1': ObjectId('67b238b223576e402a7173a9'),
    '2': ObjectId('67b238b223576e402a7173aa'),
    '3': ObjectId('67b238b223576e402a7173ab')
  }
}
biblioteca>|
```


3. Crear y poblar la colección prestamos

```
> use biblioteca
< switched to db biblioteca
> db["prestamos"].find()
<
> // Obtener los ObjectId de los libros y usuarios existentes
const librosIds = db.libros.find({}, { _id: 1 }).toArray().map(doc => doc._id);
const usuariosIds = db.usuarios.find({}, { _id: 1 }).toArray().map(doc => doc._id);

// Insertar préstamos vinculando los IDs correctos
db.prestamos.insertMany([
  { usuario_id: usuariosIds[0], libro_id: librosIds[0], fecha_prestamo: "2025-01-15", fecha_devolucion: "2025-02-15" },
  { usuario_id: usuariosIds[1], libro_id: librosIds[1], fecha_prestamo: "2025-02-10", fecha_devolucion: "2025-03-10" },
  { usuario_id: usuariosIds[2], libro_id: librosIds[2], fecha_prestamo: "2025-02-05", fecha_devolucion: "2025-03-05" },
  { usuario_id: usuariosIds[3], libro_id: librosIds[3], fecha_prestamo: "2025-01-20", fecha_devolucion: "2025-02-20" },
  { usuario_id: usuariosIds[0], libro_id: librosIds[4], fecha_prestamo: "2025-02-12", fecha_devolucion: "2025-03-12" },
  { usuario_id: usuariosIds[2], libro_id: librosIds[5], fecha_prestamo: "2025-02-18", fecha_devolucion: "2025-03-18" }
]);
< {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('67b2391a94bf12d9628a8c46'),
    '1': ObjectId('67b2391a94bf12d9628a8c47'),
    '2': ObjectId('67b2391a94bf12d9628a8c48'),
    '3': ObjectId('67b2391a94bf12d9628a8c49'),
    '4': ObjectId('67b2391a94bf12d9628a8c4a'),
    '5': ObjectId('67b2391a94bf12d9628a8c4b')
  }
}
biblioteca>
```



Aquí se muestran que se han creado estas tres colecciones.

EJERCICIOS 2 PRÁCTICOS AVANZADOS

1. Índices en MongoDB

1.1 Crea un índice en el campo direccion de la colección usuarios para optimizar las búsquedas por dirección.

Crea un índice ascendente en direccion y muestra los índices existentes:

```
> db.usuarios.createIndex({direccion: 1})
< direccion_1
> db.usuarios.getIndexes({direccion: 1})
< [
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { direccion: 1 }, name: 'direccion_1' }
]
```

1.2 Crea un índice compuesto en telefono (ascendente) y nombre (descendente).

Crea un índice compuesto en la colección usuarios sobre los campos

```
>_MONGOSH
> db.usuarios.createIndex({telefono: 1, nombre: -1})
< telefono_1_nombre_-1
biblioteca>
```

1.3. Realiza una consulta buscando un usuario por telefono, utilizando hint() para forzar el uso del índice creado en el paso 1.2

Busca un documento por telefono usando un `hint()` específico.

```
>_MONGOSH
> db.usuarios.find({telefono: "123456789"}).hint({telefono: 1, nombre: -1})
< {
  _id: ObjectId('67b238b223576e402a7173a8'),
  nombre: 'Carlos Pérez',
  direccion: 'Calle 1, Madrid',
  telefono: '123456789'
}
biblioteca>|
```

1.4. Lista todos los índices existentes en la colección usuarios.

Para mostrar que se ha creado, se pone el `getIndexes()`

```
>_MONGOSH
> db.usuarios.getIndexes()
< [
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { direccion: 1 }, name: 'direccion_1' },
  {
    v: 2,
    key: { telefono: 1, nombre: -1 },
    name: 'telefono_1_nombre_-1'
  }
]
biblioteca>
```

1.5. Elimina el índice creado en 1.2.

```
>_MONGOSH
```

```
> db.usuarios.dropIndex({telefono: 1, nombre: -1})
< { nIndexesWas: 3, ok: 1 }
biblioteca>
```

1.6. Realiza una consulta en libros buscado por título con el índice aplicado en título.

Se crea el índice si no existe y comprobamos que se haya creado.

```
>_MONGOSH
```

```
> db.libros.createIndex({titulo: 1})
< titulo_1
> db.libros.getIndexes()
< [
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { titulo: 1 }, name: 'titulo_1' }
]
biblioteca> |
```

Ahora ya podemos consultar con el hint

```
> db.libros.find({titulo: "1984"}).hint({titulo: 1})
< {
  _id: ObjectId('67b23861baaf179f107bf485'),
  titulo: '1984',
  autor: 'George Orwell',
  paginas: 328,
  'año': 1949,
  genero: 'Ficción'
}
biblioteca>
```

1.7. Realiza una consulta combinada en usuarios, buscando por nombre y ordenando por edad, asegurando que use el índice en usuario_id.

Se crea primero el index y se comprueba.

```
> db.usuarios.createIndex({nombre: 1, edad: -1})
< nombre_1_edad_-1
> db.usuarios.getIndexes()
< [
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { direccion: 1 }, name: 'direccion_1' },
  { v: 2, key: { nombre: 1, edad: -1 }, name: 'nombre_1_edad_-1' }
]
biblioteca>
```

```
> db.usuarios.find({ nombre: "Carlos Pérez" }).sort({ edad: -1 }).hint({ nombre: 1, edad: -1 });
< {
  _id: ObjectId('67b238b223576e402a7173a8'),
  nombre: 'Carlos Pérez',
  direccion: 'Calle 1, Madrid',
  telefono: '123456789'
}
biblioteca>
```

1.8. Realiza una consulta en prestamos para obtener los préstamos de un usuario específico, forzando el uso del índice en usuario_id.

Se crea primero el índice de prestamos y se comprueba.

```
>_MONGOSH
> db.prestamos.createIndex({usuario_id: 1})
< usuario_id_1
> db.prestamos.getIndexes()
< [
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { usuario_id: 1 }, name: 'usuario_id_1' }
]
biblioteca>
```

Ahora se realiza una consulta

```
>_MONGOSH
> db.prestamos.find({usuario_id: ObjectId("67b238b223576e402a7173a8")}).hint({usuario_id: 1})
< {
  _id: ObjectId('67b2391a94bf12d9628a8c46'),
  usuario_id: ObjectId('67b238b223576e402a7173a8'),
  libro_id: ObjectId('67b23861baaf179f107bf485'),
  fecha_prestamo: '2025-01-15',
  fecha_devolucion: '2025-02-15'
}
{
  _id: ObjectId('67b2391a94bf12d9628a8c4a'),
  usuario_id: ObjectId('67b238b223576e402a7173a8'),
  libro_id: ObjectId('67b23861baaf179f107bf489'),
  fecha_prestamo: '2025-02-12',
  fecha_devolucion: '2025-03-12'
}
biblioteca>
```

2. Agregaciones

2.1. Usa \$match para obtener todos los usuarios cuyo número de teléfono comienza con “987”

En este ejercicio, se usa la expresión regular (/^987/) para buscar números de teléfono que empiezan exactamente con los dígitos 987 y que el símbolo “^” indica el inicio de esta secuencia.

```
> db.usuarios.aggregate([{$match: {telefono: /^987/}}])
< {
  _id: ObjectId('67b238b223576e402a7173a9'),
  nombre: 'Ana Gómez',
  direccion: 'Calle 2, Barcelona',
  telefono: '987654321'
}
biblioteca> |
```

2.2. Agrupa los usuarios por ciudad y cuenta cuántos hay en cada una.

Se usa el `$group` para agrupar los campos.

```
>_MONGOSH
> db.usuarios.aggregate([{$group: { _id: "$direccion", Total_Usuarios: { $sum: 1 } } }]);
< {
  _id: 'Calle 3, Valencia',
  Total_Usuarios: 1
}
{
  _id: 'Calle 4, Sevilla',
  Total_Usuarios: 1
}
{
  _id: 'Calle 2, Barcelona',
  Total_Usuarios: 1
}
{
  _id: 'Calle 1, Madrid',
  Total_Usuarios: 1
}
biblioteca>|
```

2.3. Ordena los resultados del ejercicio 2.2 en orden descendente según el número de usuarios.

Igual que en el anterior ejercicio solo que se añade el `sort` para ordenar el número de usuarios.

```
> db.usuarios.aggregate([{$group: { _id: "$direccion", Total_Usuarios: { $sum: 1 } } }, {$sort: {Total_Usuarios: -1}}]);
< {
  _id: 'Calle 2, Barcelona',
  Total_Usuarios: 1
}
{
  _id: 'Calle 1, Madrid',
  Total_Usuarios: 1
}
{
  _id: 'Calle 3, Valencia',
  Total_Usuarios: 1
}
{
  _id: 'Calle 4, Sevilla',
  Total_Usuarios: 1
}
biblioteca>|
```

2.4. Proyecta solo los nombres y teléfonos de los usuarios, ocultando el `_id`.

Para ocultar el id, ponemos `_id: 0` y con eso no lo muestra.

```
>_MONGOSH
> db.usuarios.aggregate([{$project: {nombre: 1, telefono: 1, _id: 0}}])
< {
  nombre: 'Carlos Pérez',
  telefono: '123456789'
}
{
  nombre: 'Ana Gómez',
  telefono: '987654321'
}
{
  nombre: 'Luis Fernández',
  telefono: '456123789'
}
{
  nombre: 'Marta López',
  telefono: '321654987'
}
biblioteca> |
```


3. Uniones entre colecciones con \$lookup

3.1. Realiza una unión (\$lookup) entre prestamos y usuarios para mostrar el nombre del usuario en cada registro de préstamo.

from → colección en la que se une
localFields → campo de la colección de préstamo
foreignFields → campo en la colección de usuario
as → nombre del campo resultante

Código que se usa → `db.prestamos.aggregate([{ $lookup: { from: "usuarios", localField: "usuario_id", foreignField: "_id", as: "informacion_usuario" } }])`

```
> MONDRIAN
> db.prestamos.aggregate([ { $lookup: { from: "usuarios", localField: "usuario_id", foreignField: "_id", as: "informacion_usuario" } } ])
< [
  {
    "_id": ObjectId("67b2391a94bf12d9628a8c4b"),
    "usuario_id": ObjectId("67b238b223576e402a7173aa"),
    "libro_id": ObjectId("67b23861baaf179f107bf489"),
    "fecha_prestamo": "2025-01-18",
    "fecha_devolucion": "2025-02-15",
    "informacion_usuario": [
      {
        "_id": ObjectId("67b238b223576e402a7173aa"),
        "nombre": "Carlos Pérez",
        "direccion": "Calle 1, Madrid",
        "telefono": "123456789"
      }
    ]
  },
  {
    "_id": ObjectId("67b2391a94bf12d9628a8c47"),
    "usuario_id": ObjectId("67b238b223576e402a7173aa"),
    "libro_id": ObjectId("67b23861baaf179f107bf486"),
    "fecha_prestamo": "2025-02-18",
    "fecha_devolucion": "2025-03-18",
    "informacion_usuario": [
      {
        "_id": ObjectId("67b238b223576e402a7173aa"),
        "nombre": "Ana Gómez",
        "direccion": "Calle 2, Barcelona",
        "telefono": "987654321"
      }
    ]
  },
  {
    "_id": ObjectId("67b2391a94bf12d9628a8c48"),
    "usuario_id": ObjectId("67b238b223576e402a7173aa"),
    "libro_id": ObjectId("67b23861baaf179f107bf487"),
    "fecha_prestamo": "2025-02-05",
    "fecha_devolucion": "2025-03-05",
    "informacion_usuario": [
      {
        "_id": ObjectId("67b238b223576e402a7173aa"),
        "nombre": "Luis Fernández",
        "direccion": "Calle 3, Valencia",
        "telefono": "456123789"
      }
    ]
  },
  {
    "_id": ObjectId("67b2391a94bf12d9628a8c49"),
    "usuario_id": ObjectId("67b238b223576e402a7173aa"),
    "libro_id": ObjectId("67b23861baaf179f107bf488"),
    "fecha_prestamo": "2025-01-28",
    "fecha_devolucion": "2025-02-28",
    "informacion_usuario": [
      {
        "_id": ObjectId("67b238b223576e402a7173aa"),
        "nombre": "Marta López",
        "direccion": "Calle 4, Sevilla",
        "telefono": "321654987"
      }
    ]
  }
]
```

```
{
  "_id": ObjectId("67b2391a94bf12d9628a8c4a"),
  "usuario_id": ObjectId("67b238b223576e402a7173aa"),
  "libro_id": ObjectId("67b23861baaf179f107bf489"),
  "fecha_prestamo": "2025-02-12",
  "fecha_devolucion": "2025-03-12",
  "informacion_usuario": [
    {
      "_id": ObjectId("67b238b223576e402a7173aa"),
      "nombre": "Carlos Pérez",
      "direccion": "Calle 1, Madrid",
      "telefono": "123456789"
    }
  ]
},
{
  "_id": ObjectId("67b2391a94bf12d9628a8c4b"),
  "usuario_id": ObjectId("67b238b223576e402a7173aa"),
  "libro_id": ObjectId("67b23861baaf179f107bf48a"),
  "fecha_prestamo": "2025-02-18",
  "fecha_devolucion": "2025-03-18",
  "informacion_usuario": [
    {
      "_id": ObjectId("67b238b223576e402a7173aa"),
      "nombre": "Luis Fernández",
      "direccion": "Calle 3, Valencia",
      "telefono": "456123789"
    }
  ]
}
]
biblioteca>
```

3.2. Expande los resultados de la consulta anterior usando \$unwind para que cada préstamo muestre los datos del usuario de manera individual.

Lo mismo que en el anterior ejercicio, solo que se añade el unwind que descompone el array en varios documentos.

```
> MONGOSH
> db.prestamos.aggregate([{$lookup: { from: "usuarios", localField: "usuario_id", foreignField: "_id", as: "informacion_usuario" }}, {$unwind: "$informacion_usuario" }])
{
  "_id": ObjectId("67b2391a94bf12d9628a8c46"),
  "usuario_id": ObjectId("67b238b223576e402a7173aa"),
  "libro_id": ObjectId("67b23861baaf179f107bf485"),
  "fecha_prestamo": "2025-01-15",
  "fecha_devolucion": "2025-02-15",
  "informacion_usuario": {
    "_id": ObjectId("67b238b223576e402a7173aa"),
    "nombre": "Carlos Pérez",
    "direccion": "Calle 1, Madrid",
    "telefono": "123456789"
  }
}
{
  "_id": ObjectId("67b2391a94bf12d9628a8c47"),
  "usuario_id": ObjectId("67b238b223576e402a7173aa"),
  "libro_id": ObjectId("67b23861baaf179f107bf485"),
  "fecha_prestamo": "2025-02-10",
  "fecha_devolucion": "2025-03-10",
  "informacion_usuario": [
    {
      "_id": ObjectId("67b238b223576e402a7173aa"),
      "nombre": "Ana Gómez",
      "direccion": "Calle 2, Barcelona",
      "telefono": "987654321"
    }
  ]
}
{
  "_id": ObjectId("67b2391a94bf12d9628a8c48"),
  "usuario_id": ObjectId("67b238b223576e402a7173aa"),
  "libro_id": ObjectId("67b23861baaf179f107bf487"),
  "fecha_prestamo": "2025-02-05",
  "fecha_devolucion": "2025-03-05",
  "informacion_usuario": {
    "_id": ObjectId("67b238b223576e402a7173aa"),
    "nombre": "Luis Fernández",
    "direccion": "Calle 3, Valencia",
    "telefono": "456123789"
  }
}
{
  "_id": ObjectId("67b2391a94bf12d9628a8c49"),
  "usuario_id": ObjectId("67b238b223576e402a7173aa"),
  "libro_id": ObjectId("67b23861baaf179f107bf488"),
  "fecha_prestamo": "2025-01-20",
  "fecha_devolucion": "2025-02-20",
  "informacion_usuario": {
    "_id": ObjectId("67b238b223576e402a7173aa"),
    "nombre": "Marta López",
    "direccion": "Calle 4, Sevilla",
    "telefono": "321654987"
  }
}
{
  "_id": ObjectId("67b2391a94bf12d9628a8c4a"),
  "usuario_id": ObjectId("67b238b223576e402a7173aa"),
  "libro_id": ObjectId("67b23861baaf179f107bf489"),
  "fecha_prestamo": "2025-02-12",
  "fecha_devolucion": "2025-03-12",
  "informacion_usuario": {
    "_id": ObjectId("67b238b223576e402a7173aa"),
    "nombre": "Carlos Pérez",
    "direccion": "Calle 1, Madrid",
    "telefono": "123456789"
  }
}
```

```
{
  "_id": ObjectId('67b2391a94bf12d9628a8c4b'),
  "usuario_id": ObjectId('67b238b223576e402a7173aa'),
  "libro_id": ObjectId('67b23861baaf179f107bf48a'),
  "fecha_prestamo": '2025-02-18',
  "fecha_devolucion": '2025-03-18',
  "informacion_usuario": {
    "_id": ObjectId('67b238b223576e402a7173aa'),
    "nombre": 'Luis Fernández',
    "direccion": 'Calle 3, Valencia',
    "telefono": '456123789'
  }
}
biblioteca>
```

3.3. Realiza una unión entre prestamos y libros para obtener el título del libro prestado en cada registro.

```
> db.prestamos.aggregate([
  {$lookup:
    {from: "libros",
     localField: "libro_id",
     foreignField: "_id",
     as: "titulo_libro" }}, {$unwind: "$titulo_libro" }
])
< [
  {
    _id: ObjectId('67b2391a94bf12d9628a8c46'),
    usuario_id: ObjectId('67b238b223576e402a7173a8'),
    libro_id: ObjectId('67b23861baaf179f107bf485'),
    fecha_prestamo: '2025-01-15',
    fecha_devolucion: '2025-02-15',
    titulo_libro: {
      _id: ObjectId('67b23861baaf179f107bf485'),
      titulo: '1984',
      autor: 'George Orwell',
      paginas: 328,
      'año': 1949,
      genero: 'Ficción'
    }
  },
  {
    _id: ObjectId('67b2391a94bf12d9628a8c47'),
    usuario_id: ObjectId('67b238b223576e402a7173a9'),
    libro_id: ObjectId('67b23861baaf179f107bf486'),
    fecha_prestamo: '2025-02-10',
    fecha_devolucion: '2025-03-10',
    titulo_libro: {
      _id: ObjectId('67b23861baaf179f107bf486'),
      titulo: 'Cien años de soledad',
      autor: 'Gabriel García Márquez',
      paginas: 417,
      'año': 1967,
      genero: 'Realismo mágico'
    }
  },
  {
    _id: ObjectId('67b2391a94bf12d9628a8c48'),
    usuario_id: ObjectId('67b238b223576e402a7173aa'),
    libro_id: ObjectId('67b23861baaf179f107bf487'),
    fecha_prestamo: '2025-02-05',
    fecha_devolucion: '2025-03-05',
    titulo_libro: {
      _id: ObjectId('67b23861baaf179f107bf487'),
      titulo: 'El Hobbit',
      autor: 'J.R.R. Tolkien',
      paginas: 310,
      'año': 1937,
      genero: 'Fantasía'
    }
  },
  {
    _id: ObjectId('67b2391a94bf12d9628a8c49'),
    usuario_id: ObjectId('67b238b223576e402a7173ab'),
    libro_id: ObjectId('67b23861baaf179f107bf488'),
    fecha_prestamo: '2025-01-20',
    fecha_devolucion: '2025-02-20',
    titulo_libro: {
      _id: ObjectId('67b23861baaf179f107bf488'),
      titulo: 'Don Quijote de la Mancha',
      autor: 'Miguel de Cervantes',
      paginas: 863,
      'año': 1605,
      genero: 'Novela'
    }
  }
]
```

```
{
  _id: ObjectId('67b2391a94bf12d9628a8c4a'),
  usuario_id: ObjectId('67b238b223576e402a7173a8'),
  libro_id: ObjectId('67b23861baaf179f107bf489'),
  fecha_prestamo: '2025-02-12',
  fecha_devolucion: '2025-03-12',
  titulo_libro: {
    _id: ObjectId('67b23861baaf179f107bf489'),
    titulo: 'Orgullo y prejuicio',
    autor: 'Jane Austen',
    paginas: 279,
    'año': 1813,
    genero: 'Romance'
  }
}
{
  _id: ObjectId('67b2391a94bf12d9628a8c4b'),
  usuario_id: ObjectId('67b238b223576e402a7173aa'),
  libro_id: ObjectId('67b23861baaf179f107bf48a'),
  fecha_prestamo: '2025-02-18',
  fecha_devolucion: '2025-03-18',
  titulo_libro: {
    _id: ObjectId('67b23861baaf179f107bf48a'),
    titulo: 'Fahrenheit 451',
    autor: 'Ray Bradbury',
    paginas: 249,
    'año': 1953,
    genero: 'Ciencia ficción'
  }
}
biblioteca>
```

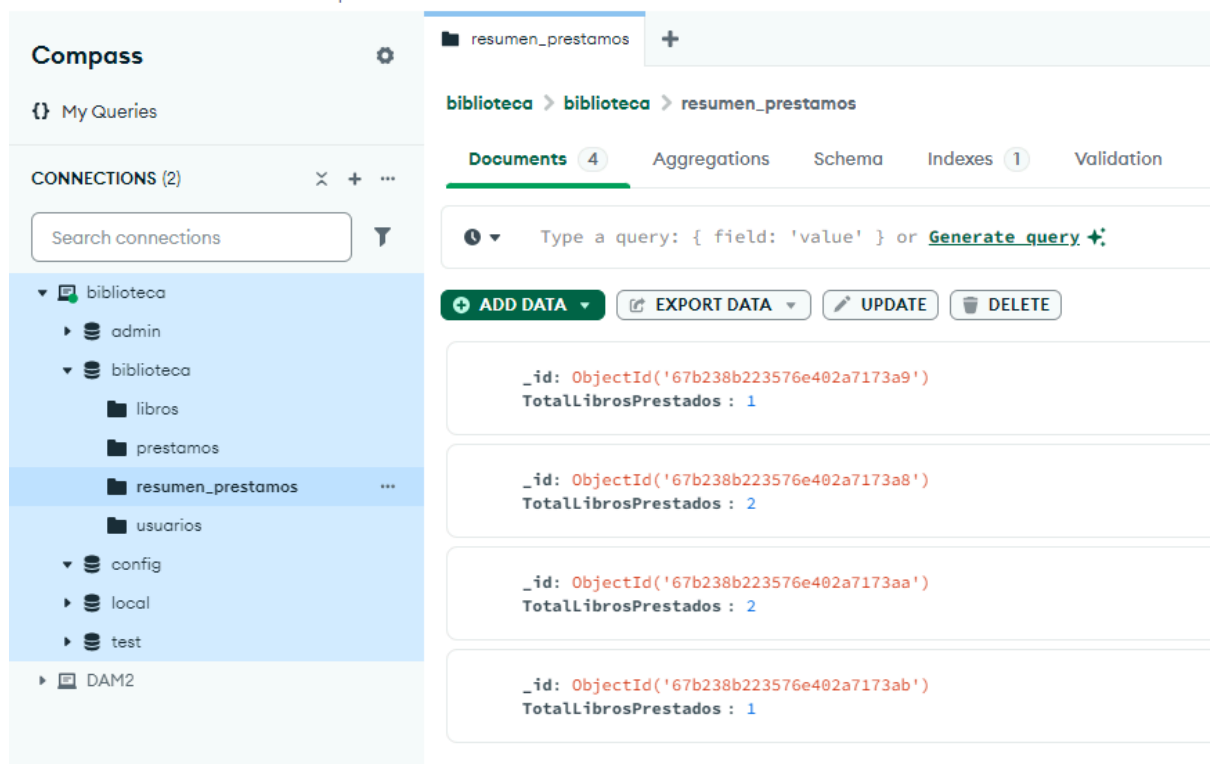
4. Almacenamiento de resultados con \$merge

4.1. Agrupa los préstamos por usuario_id y cuenta cuántos libros tiene cada usuario prestados, almacenando el resultado en resumen_prestamos.

El código agrupa los préstamos por usuario_id y calcula el total de libros prestados, luego usa \$merge para insertar o fusionar los resultados en resumen_prestamos.

```
>_MONGOSH
> db.prestamos.aggregate([
  { $group: { _id: "$usuario_id", TotalLibrosPrestados: { $sum: 1 } } },
  { $merge: { into: "resumen_prestamos", whenMatched: "merge", whenNotMatched: "insert" } }
])
<
biblioteca> |
```

Comprobamos que se ha creado esta colección



The screenshot shows the MongoDB Compass interface. On the left, the 'CONNECTIONS (2)' panel lists the database 'biblioteca' with collections 'libros', 'prestamos', 'resumen_prestamos' (selected), and 'usuarios'. The main panel displays the 'resumen_prestamos' collection with 4 documents. The documents are as follows:

| _id | TotalLibrosPrestados |
|--------------------------------------|----------------------|
| ObjectId('67b238b223576e402a7173a9') | 1 |
| ObjectId('67b238b223576e402a7173a8') | 2 |
| ObjectId('67b238b223576e402a7173aa') | 2 |
| ObjectId('67b238b223576e402a7173ab') | 1 |

4.2. Verifica el contenido de la colección `resumen_prestamos`.

Consultamos que se ha creado.

```
>_MONGOSH  
  
> db.resumen_prestamos.find()  
< {  
  _id: ObjectId('67b238b223576e402a7173a9'),  
  TotalDePrestamos: 1,  
  TotalLibrosPrestados: 1  
}  
{  
  _id: ObjectId('67b238b223576e402a7173a8'),  
  TotalDePrestamos: 2,  
  TotalLibrosPrestados: 2  
}  
{  
  _id: ObjectId('67b238b223576e402a7173aa'),  
  TotalDePrestamos: 2,  
  TotalLibrosPrestados: 2  
}  
{  
  _id: ObjectId('67b238b223576e402a7173ab'),  
  TotalDePrestamos: 1,  
  TotalLibrosPrestados: 1  
}  
biblioteca> |
```

4.3. Modifica la agregación anterior para que, en caso de coincidencia de usuario, reemplace el documento en lugar de hacer merge

Agrupar los documentos de prestamos por usuario_id y calcular el total de libros prestados. Luego, con \$merge, inserta o reemplaza los resultados en la colección resumen_prestamos, dependiendo de si el documento ya existe.

```
> db.prestamos.aggregate([  
  $group: {_id: "$usuario_id", TotalLibrosPrestados: { $sum: 1 } }},  
  {$merge: {into: "resumen_prestamos", whenMatched: "replace", whenNotMatched: "insert"}} ])  
<
```

```
> db.resumen_prestamos.find()  
< {  
  _id: ObjectId('67b238b223576e402a7173a9'),  
  TotalLibrosPrestados: 1  
}  
{  
  _id: ObjectId('67b238b223576e402a7173a8'),  
  TotalLibrosPrestados: 2  
}  
{  
  _id: ObjectId('67b238b223576e402a7173aa'),  
  TotalLibrosPrestados: 2  
}  
{  
  _id: ObjectId('67b238b223576e402a7173ab'),  
  TotalLibrosPrestados: 1  
}  
biblioteca> |
```