Referencing in MongoDB:

In MongoDB, database references, also known as DBRefs, are a way to create relationships between documents in different collections. A database reference is essentially a convention for including information about one document in another document using a standardized structure. It includes the collection name and the ObjectId of the referenced document.

MongoDB applications use one of two methods to relate documents:

- **Manual references** save the _id field of one document in another document as a reference. The application runs a second query to return the related data. These references are simple and sufficient for most use cases.
- **DBRefs** are references from one document to another using the value of the first document's _id field, collection name, and, optionally, its database name, as well as any other fields. DBRefs allows easy reference to documents stored in multiple collections or databases
  Format:
  { "$ref" : <value>, "$id" : <value>, "$db" : <value> }
  - ➔ The $ref field holds the name of the collection where the referenced document resides
  - ➔ The $id field contains the value of the _id field in the referenced document.
  - ➔ The $db(optional) field contains the name of the database where the referenced document resides.

The $lookup pipeline can be used to combine documents from one collection with documents from another collection based on a specified condition It pPerforms a left outer join to a collection in the same database to filter in documents from the "joined" collection for processing. The $lookup stage adds a new array field to each input document. The new array field contains the matching documents from the "joined" collection.

One to one relationship with document references:

```
> db.bookIssue.aggregate([{$lookup:{from:"student",localField:"student_id",foreignField:"_id",as:"studentDetails"}}])
< {
    _id: ObjectId("64e473418bd8126e8925b51b"),
    student_id: ObjectId("64e465a68bd8126e8925b515"),
    issueDate: 2023-02-02T00:00:00.000Z,
    dueDate: 2023-04-02T00:00:00.000Z,
    bookId: ObjectId("64e46fd08bd8126e8925b519"),
    studentDetails: [
      {
        _id: ObjectId("64e465a68bd8126e8925b515"),
        address: 'Kritipur',
        email: 'ram@gmail.com',
        name: 'Ram',
        phone: 5555555,
        rollNo: 1
      }
    ]
  }
Atlas atlas-iyexlg-shard-0 [primary] LibraryManagementSystem>
```

One to many relationship with document references:

```
> db.author.aggregate([
    {
      $match: { _id: ObjectId("64e4721a8bd8126e8925b51a") }
    },
    {
      $lookup: {
        from: "books",
        localField: "_id",
        foreignField: "author_id",
        as: "booksDetail"
      }
    }
  ])
< {
    _id: ObjectId("64e4721a8bd8126e8925b51a"),
    name: 'F.Scott',
    birthdate: 1998-02-02T00:00:00.000Z,
    biography: 'A renowed author for his book Begining MongoDB',
    books: [
      {
        object_id: ObjectId("64e46fd08bd8126e8925b519")
      },
      {
        object_id: ObjectId("64e58a508bd8126e8925b51c")
      }
```

```
    booksDetail: [
      {
        _id: ObjectId("64e46fd08bd8126e8925b519"),
        title: 'Begining MongoDB',
        author: 'F.Scott',
        isbn: '973-316-456123-0',
        description: 'MongDB guide for begineer',
        genre: {
          title: 'Programming',
          description: 'Programming languages, frameworks, and methodologies
        },
        author_id: ObjectId("64e4721a8bd8126e8925b51a")
      },
      {
        _id: ObjectId("64e58a508bd8126e8925b51c"),
        title: 'JS',
        author: 'Paulo',
        isbn: '329-645-452152-1',
        description: 'Learn JS',
        author_id: ObjectId("64e4721a8bd8126e8925b51a")
      }
    ]
  }
Atlas atlas-iyexlg-shard-0 [primary] LibraryManagementSystem>
```

The above two examples show how data are retrieved for one-to-one and one-to-many relationships using $lookup.

Managing data without references:

Managing data without using MongoDB references often involves embedding documents . In this approach, you store related data directly within a document as an array or nested subdocuments. This can be suitable when the related data is small and frequently accessed together.

Embedding documents can lead to duplication and increased document size if data is shared among multiple documents. However, it can also improve query performance by reducing the need for joins.

```
{
  "_id": {
    "$oid": "64e46fd08bd8126e8925b519"
  },
  "title": "Begining MongoDB",
  "author": "F.Scott",
  "isbn": "973-316-456123-0",
  "description": "MongDB guide for begineer",
  "genre": {
    "title": "Programming",
    "description": "Programming languages, framewor
  },
  "author_id": {
    "$oid": "64e4721a8bd8126e8925b51a"
  }
}
```

In this example, the genre collection is embedded within the books collection.

Situations when to use and not use references:

Data Integrity and Consistency:
Using references can help maintain data integrity by ensuring that changes to related data are updated in one place.
If not used changes need to be propagated to all instances of duplicated data.

Query Performance:
Using references can require additional queries to fetch related data. However, it allows for more flexible queries and avoids potentially large document sizes when dealing with multiple related entities.
If not used can lead to faster queries because all data is available in a single document. However, as the document size grows due to embedded data, query performance might degrade.

Data Duplication:
Using references reduces data duplication, leading to smaller document sizes. This can be especially important for scenarios with large amounts of related data.
If not used can lead to data duplication. This may increase storage requirements and maintenance efforts.

Complexity of Data Model:
Using references often results in a simpler data model, with data organized more naturally across multiple collections.
It not used can simplify certain queries and reduce the need for joins, but it can also lead to complex documents and potential difficulties when managing relationships.

Updates and Changes:
Using references can simplify updates, as changes are made in one place. However, updating references might require additional steps to ensure data consistency.
If not used, changes need to be propagated across documents.