**Practice Questions**

1. **Find Events Happening in a Specific Month**
   * **Question**: Write a query to find all events happening in March 2024.
   * **Hint**: Use the $month operator in combination with $match.
2. **Calculate the End Time of Each Event**
   * **Question**: Write an aggregation query to calculate the end time of each event by adding durationHours to the event's start time.
   * **Hint**: Use $add operator in the $project stage.
3. **Find Events Happening on Weekends**
   * **Question**: Write a query to find all events that are scheduled on weekends (Saturday and Sunday).
   * **Hint**: Use the $dayOfWeek operator to filter by the day of the week.
4. **Count Events Grouped by Month**
   * **Question**: Write an aggregation query to count the number of events happening each month.
   * **Hint**: Use $group by month extracted from the date field.
5. **Find the Longest Event**
   * **Question**: Write a query to find the event with the longest duration.
   * **Hint**: Use $sort to sort by durationHours in descending order and limit the result to one.
6. **Find Events Happening in the Next 30 Days**
   * **Question**: Write a query to find all events that are scheduled within the next 30 days from today.
   * **Hint**: Use $gte and $lte to filter dates within the specified range.
7. **Convert Event Dates to Local Time Zones**
   * **Question**: Write an aggregation query to convert event dates to their respective local time zones. For example, convert the date to "America/New\_York" for events in New York.
   * **Hint**: Use $dateToString with the timezone option.
8. **Calculate the Total Duration of All Events**
   * **Question**: Write an aggregation query to calculate the total duration of all events combined.
   * **Hint**: Use $sum in the $group stage.
9. **Find Events Before a Specific Date**
   * **Question**: Write a query to find all events that are scheduled before June 1, 2024.
   * **Hint**: Use $lt to filter events before the specified date.
10. **Create a New Field for Event Day Names**
    * **Question**: Write a query to add a new field that displays the day of the week (e.g., "Monday", "Tuesday") for each event.
    * **Hint**: Use $dayOfWeek and map the numbers to their corresponding day names.

In MongoDB, dates are stored as **BSON Date objects**, which are based on the number of **milliseconds** since the Unix epoch (January 1, 1970, 00:00:00 UTC). This is also known as **Unix time** or **Unix timestamp**.

**Explanation:**

1. **Unix Epoch**:
   * MongoDB represents dates as the number of milliseconds that have elapsed since the Unix epoch (January 1, 1970, UTC).
   * For example, the Unix timestamp 0 corresponds to 1970-01-01T00:00:00.000Z.
2. **Date Storage in Milliseconds**:
   * When you insert a date into MongoDB, the system internally converts that date into the number of milliseconds since the Unix epoch and stores it as a Date object in BSON.
   * For example:

javascript

Copy code

ISODate("2024-03-15T09:00:00Z")

Internally, this will be stored as the number of milliseconds that have passed since January 1, 1970.

1. **Why Milliseconds?**:
   * Milliseconds provide high precision, allowing MongoDB to represent and compare dates down to the millisecond level.
   * Most programming languages and systems also use the Unix time format (milliseconds or seconds since epoch), which ensures easy compatibility between MongoDB and other systems.

**Working with Date in Milliseconds:**

When you perform date calculations or comparisons in MongoDB, the system uses the internal millisecond representation. For example, if you add or subtract time, you’re actually manipulating the number of milliseconds.

**Example 1: Adding Hours to a Date**

* To add 5 hours to a date in MongoDB, you need to convert 5 hours into milliseconds:
  + **1 hour = 60 minutes = 3600 seconds = 3600 \* 1000 milliseconds = 3,600,000 milliseconds.**
  + So, adding 5 hours means adding 5 \* 3,600,000 = 18,000,000 milliseconds.

**Example 2: Date Comparison**

* When comparing two dates in MongoDB, it compares the underlying millisecond values.
  + Example: If dateA = ISODate("2024-01-01T00:00:00Z") and dateB = ISODate("2024-01-01T05:00:00Z"), then dateB is 18,000,000 milliseconds (5 hours) later than dateA.

**Example of How Dates are Stored and Used in MongoDB:**

javascript

Copy code

db.events.insert({

eventName: "New Year Party",

date: ISODate("2024-01-01T00:00:00Z")

});

Internally, MongoDB stores the date as the number of milliseconds from the Unix epoch, but when queried, it provides the date in a human-readable format (e.g., ISODate).

**Adding 5 Hours to the Event Date:**

javascript

Copy code

db.events.aggregate([

{

$addFields: {

endTime: {

$add: [

"$date",

{ $multiply: [5, 60 \* 60 \* 1000] } // Adding 5 hours in milliseconds

]

}

}

}

]);

* Here, MongoDB is adding 5 hours (in milliseconds) to the event's date, which is stored internally as milliseconds since the epoch.

**Summary:**

* MongoDB stores Date objects as **milliseconds** since the Unix epoch (1970-01-01T00:00:00Z).
* When performing calculations (like adding hours or days), you need to convert the time into milliseconds.
* Storing dates in this format allows MongoDB to handle date comparisons and arithmetic with high precision.