**Aggregation Problems for**users**Collection**

1. **Average Age of Users**: Calculate the average age of all users.
2. **Top 5 Oldest Users**: Find the top 5 oldest users.
3. **Users with Specific Domain Emails**: Count the number of users with email addresses from a specific domain (e.g., example.com).
4. **Recent Logins**: Retrieve the users who have logged in within the last 30 days.
5. **Usernames Starting with a Letter**: Find all usernames that start with a specific letter.
6. **Users by Last Login Year**: Group users by the year of their last login and count the number of users for each year.
7. **Users with Missing Fields**: Identify users who do not have an email address.
8. **Average Age by First Name Initial**: Calculate the average age of users grouped by the first letter of their first name.
9. **Users with Multiple Conditions**: Find users who are older than 30 and have logged in within the last 6 months.

**Aggregation Problems for**products**Collection**

1. **Average Price by Category**: Calculate the average price of products in each category.
2. **Total Inventory Value**: Calculate the total value of all products.
3. **Top 3 Most Expensive Products**: Find the top 3 most expensive products.
4. **Products with Specific Keyword**: Find products whose description contains a specific keyword.
5. **Products by Category Count**: Count the number of products in each category.
6. **Price Increase Simulation**: Calculate the new prices if all products’ prices were increased by 10%.
7. **Products with Multiple Conditions**: Find products that belong to a specific category and have a price above a certain threshold.

**Aggregation Problems for**places**Collection**

1. **Average Rating of Places**: Calculate the average rating of all places.
2. **Places Near a Point**: Find all places within a 10-kilometer radius of a given point.
3. **Top 5 Highest Rated Places**: Find the top 5 highest rated places.
4. **Places by Proximity**: Sort places by their distance from a given point.
5. **Places with Specific Name Keyword**: Find places whose name contains a specific keyword.
6. **Places by Rating Count**: Count the number of places for each rating value.
7. **Places with Multiple Conditions**: Find places that have a rating above 4.5 and are within a certain distance from a given point.

**Additional Information**

**Work with dates.**

How to use Date as a Date and not as a String. And how to obtain the difference between current date and the attribute in days.

db.users.aggregate([{$addFields : {lastLogin : {$toDate : "$lastLogin"}}},{$addFields : {lastLoginDays: {$dateDiff : {startDate: "$lastLogin", endDate : "$$NOW", unit: "day"}}}}])

The **$addFields** operation adds a filed by a name and an expression for its values.

The **$toDate** operation transforms an attribute into a Date type instead of a String type.

The **$dateDiff** operation allows to subtract dates with a specific unit, in this case with “days”.

The **$$NOW** operation returns he current date in Date format.

[For more information about dateDiff go through the link](https://www.mongodb.com/docs/manual/reference/operator/aggregation/dateDiff/#mongodb-expression-exp.-dateDiff)

**Work with existing or not elements**

Use the operator $exists to filter by a field that exits or not.

Example of use:

db.users.aggregate([{$match : {name : {$exists : false}}}])

**Work with distances**

Use the $geoNear operation to query the distance to a point. Example:

db.buildings.aggregate([

{

$geoNear: {

near: { type: "Point", coordinates: [-74.0445, 40.6892] },

maxDistance: 5000,

distanceField: "location",

spherical: true

}

}

])

Details about the parameters:

* near: The point to compare with.
* distanceField: The field in the collection that contains the point.
* maxDistance: The maximum distance in meters.
* spherical: Set to true to consider the curvature of the Earth.

**Note**: If you omit the maxDistance parameter, the query will return all places sorted by distance in meters, in the location attribute.