Assignment 8 - Implement MongoDB Aggregation framework.

1. Explain all the aggregation functions in MongoDB.

Aggregation in MongoDB is nothing but an operation used to process the data that returns the computed results. Aggregation basically groups data from multiple documents and operates in many ways on those grouped data in order to return one combined result. In sql, count(*) and with groupby is an equivalent of MongoDB aggregation. Aggregation groups the records in a collection & can be used to provide total number (sum), average, minimum, maximum etc. of the group selected.

For performing aggregation in MongoDB, aggregate() is the function to be used.

Syntax -

~~aggregation.~~ db.collection_name.aggregate(aggregate_operation)

Different expressions used by aggregate function -

$sum - sums up the defined values from all the documents in a collection.

$avg - calculates the average values from all the documents in a collection

$min - returns the minimum of all values of documents in a collection

$max - Returns the maximum of all values of documents in a collection

$addToSet - Inserts values to an array but no duplicates in the resulting document.

$push - Inserts values to ~~the~~ an array in the resulting document.

$first - Returns the first document from the source document.

$last - Returns the last document from the source document.

2. Explain all the pipeline operators in detail.

In MongoDB, aggregation pipeline is a framework for data aggregation modelled on the concept of data processing pipelines. Documents enter as an input in the multi-stage pipeline which transforms the documents into aggregated results.

MongoDB pipeline consists of various stages. Each stage transforms the documents passing through the pipeline. The pipeline stages can appear multiple times in the pipeline. Following are some pipeline operators -

$project - Used to select some specific fields from a collection.

$match - This is a filtering operation & can reduce the amount of documents that are given as input to the next stage.

$group - This does the actual aggregation.

$sort - Sorts the documents.

2. Explain all the pipeline operators in detail.

$skip - With this, it is possible to skip forward in the list of documents for a given amount of documents.

$limit - This limits the amount of documents to look at, by the given number starting from the current positions.

$unwind - This is used to unwind document that are using arrays. When using an array the data is kind of pre-joined & this operation will be undone with this to have individual documents again. Thus with this stage, we will increase the amount of documents for the next stage.

3. Explain indexing in MongoDB.

Indexes support the efficient resolution of queries. Without indexes, MongoDB must scan every document of a collection to select those documents that match the query statement. The scan is highly inefficient & requires MongoDB to process a large volume of data.

createIndex() method - To create an index, you need to use the createIndex() method of MongoDB.

Syntax -

>db.collection_name.createIndex({key:1})

here key is the name of the field that you want to create index & 1 is for ascending order. To create index in descending order, you need to use -1.

dropIndex() method - You can drop a particular index using the dropIndex() method of MongoDB.

>db.collection_name.dropIndex({key:1})
key is name of the file that you want to create index, 1 is for ascending order. For descending order, you need -1.

dropIndexes() method - This method deletes multiple (specified) indexes on a collection.

>db.collection_name.dropIndexes()

getIndexes() method - It returns the description of all the indexes in the collection.
>db.mycol.getIndexes()
↑
This retrieves all the indexes in the collection mycol.

4. Explain the explain() function in MongoDB.
explain() command provides information on the execution of the following commands - aggregate, count, distinct, find, findAndModify, delete, mapReduce & update. explain() is a method that you can apply to simple querries or cursors to investigate the querry execution plan. The execution plan is how MongoDB

resolves a query. The goal of explain command in ~~Mongo DB~~ & MySQL & MongoDB's explain() method are exactly same.

5. Explain all parameters in detail.
Syntax of explain method is
db.collection.explain()

Parameters

Name	Description	Required/ Optional	Type
verbosity	Specifies the verbosity mode for the explain output. The mode affects the behaviour of explain() and determines the amount of information to return. The possible modes are - "queryPlanner", "executionStats" and "allPlansExecution" The behaviour of db.collection.explain() and the amount of information returned depend on the verbosity mode.	Optional	String

queryPlannerMode - By default db.collection.explain() runs in queryPlanner verbosity mode.

executionStats Mode - MongoDB runs the query optimizer to choose the winning plan, executes the winning plan to completion and returns statistics for the other candidate plans captured during plan selection.

allPlansExecution Mode - MongoDB runs the query optimizer to choose the winning plan & executes the winning plan to completion and returns statistics describing the execution of the winning plan, as well as statistics for the other candidate plans captured during the plan selection.

6. Explain MapReduce() in MongoDB with example.

When to use Map-Reduce → In MongoDB, you can use MapReduce() when your aggregation query is slow because data is present in large amount & the aggregation query is taking more time to process. So using map-reduce, you can perform action faster than aggregation query.

MongoDB provides the mapReduce() function to perform the map-reduce operations. This function has 2 main functions - map function & reduce function. The map function is used to group all the data based on the key-value & the reduce function is used to perform operations on the mapped data. Using Map Reduce, you can perform some aggregation functions like max, avg on the data using some key & it is similar to groupBy in SQL.

Syntax -

```
db.collectionName.mapReduce(
... map(),
... reduce(),
... querry{},
... output{}
);
```

map() function - It uses the emit() function, in which it takes 2 parameters - key & value key. Here the key is on which we make groups like groups in MySQL. Eg - like group by ages or names & the second parameter is on which aggregation is performed like avg(), sum() is calculated on.

reduce() function - It is the step in which perform our aggregate function like avg(), sum().

query - Here we will pass the query to filter the resultset.

output - In this we will specify the collection name where the result is stored.