MongoDB Aggregation Framework

The MongoDB Aggregation Framework is a powerful tool for performing complex data transformations and analyses directly within MongoDB databases. It is designed to aggregate data from multiple documents and return computed results, allowing for operations like data summarization, transformation, and complex filtering.

Understanding the Aggregation Pipeline

1. **Basics of the Aggregation Pipeline**:
    - The aggregation pipeline is a framework that processes data records and returns computed results.
    - It operates through a series of stages, each performing an operation on the data (e.g., filtering, grouping, projecting).
    - Data flows sequentially from one stage to the next, with the output of one stage serving as the input for the subsequent stage.
2. **Core Components and Operations**:
    - **$match**: Filters the data to pass only the documents that match specified conditions to the next pipeline stage, similar to the `find` method but used within the aggregation framework.
    - **$project**: Reshapes documents by including, excluding, or adding new fields. It can also be used to compute new values and assign them to fields.
    - **Starting an Aggregation Query**

```
db.collectionName.aggregate([{},{},{}])
```

This initiates an aggregation with an array representing the pipeline stages.

- **Dummy Data for Practice**: Download JSON

Practical Use Cases and Examples

1. **Filtering Documents**:
   - Retrieves users aged between 18 and 60:

```
db.Customers.aggregate([
  {$match: {$and: [{age: {$gte: 18}}, {age: {$lte: 6
0}}]}}
])
```

2. **Projecting Specific Fields**:
   - Include only the `first_name` and `salary` fields in the output:

```
db.Customers.aggregate([
  {$project: {first_name: 1, salary: 1}}
])
```

   - Transform `age` from years to months and include `first_name`:

```
db.Customers.aggregate([
  {$project: {NewAge: {$multiply: ["$age", 12]}, first_
name: 1}}
])
```

3. **Combining Fields**:
   - Concatenate `first_name` and `last_name` into a full name:

```
db.Customers.aggregate([
  {$project: {Name: {$concat: ["$first_name", " ", "$la
```

```
st_name"]}, age: 1}}
])
```

4. **Filter by Age and Project Names**

- Retrieve the first and last name of customers older than 20.

```
jsxCopy code
db.Customers.aggregate([
  {$match: {age: {$gt: 20}}},
  {$project: {first_name: 1, last_name: 1}}
])
```

Advanced Usage: Updating Nested Fields

- **Scenario**: Updating the `zipcode` in a user's address.
  This operation demonstrates how to target and update nested fields within documents, showcasing the flexibility of MongoDB in handling complex data structures.

```
db.users.updateOne(
  { "username": "john_doe" },
  { $set: { "address.zipcode": "12345" }}
)
```

Student Activity

- Task: Convert salary to thousands and append "k" (e.g., 35k for a salary of 35167).
- This activity encourages the application of multiple operations like division, rounding, and string manipulation within an aggregation pipeline.

Further Learning Resources

- **Aggregation Pipeline**: Detailed exploration of how MongoDB processes data through multiple stages.

  MongoDB Aggregation Pipeline

- **MongoDB Aggregation Framework**: Aggregation Introduction
- **Pipeline Stages**: Comprehensive list and explanation of available stages and their functions.

  Pipeline Stages and Operators

- **$project**: Projection Operators
- **Update Operations**: Guide on updating document fields, including nested fields and arrays.

  Update Operators
