⭐ ⭐ ⭐ MONGODB MAP-REDUCE PRACTICAL (FINAL PERFECT ANSWER) ⭐ ⭐ ⭐

🎯 AIM

To perform Map-Reduce operations in MongoDB for:

1. Calculating average marks of students
2. Performing word count from text documents

This demonstrates how MongoDB processes large datasets using map (key–value generation) and reduce (aggregation).

📚 THEORY

✓ What is Map-Reduce?

Map-Reduce is a data processing technique used in MongoDB for large-scale operations.

✓ How It Works?

Phase	Meaning
Map	Processes each document and emits (key, value) pairs
Reduce	Groups values by key and applies an aggregate function (sum, avg, count, etc.)

✓ Why Map-Reduce?

- Works on big datasets
- Performs distributed processing
- Helps in aggregation, counting, averaging, grouping, etc.

✓ Where Used?

- Log analysis
- Word counting
- Big data processing
- Student performance analysis

- Analytics for e-commerce

🟩 TASK 1: Calculate Average Marks of Students

🟦 FULL CLEAN CODE (Without Explanation — For Notebook)

```
use schoolDB


db.createCollection("students")


db.students.insertMany([
{ name: "Amit", subject: "Math", marks: 85 },
{ name: "Amit", subject: "Science", marks: 90 },
{ name: "Riya", subject: "Math", marks: 78 },
{ name: "Riya", subject: "Science", marks: 88 },
{ name: "John", subject: "Math", marks: 92 },
{ name: "John", subject: "Science", marks: 81 }
])


db.students.find().pretty()


var mapFunction = function() {
 emit(this.name, this.marks);
};


var reduceFunction = function(key, values) {
 return Array.avg(values);
};
```

```
db.students.mapReduce(
 mapFunction,
 reduceFunction,
 { out: "average_marks" }
)
```

```
db.average_marks.find().pretty()
```

🟦 CODE WITH EXPLANATION (POINT-WISE)

🔵 1️⃣ Create Database

```
use schoolDB
```

✔ Explanation

1. Creates or switches to the database.
2. All collections will be stored inside this.

🔵 2️⃣ Create Collection

```
db.createCollection("students")
```

✔ Explanation

1. Creates a collection named “students”.
2. Stores each student’s subject and marks.

🔵 3️⃣ Insert Student Mark Records

```
db.students.insertMany([
{ name:"Amit", subject:"Math", marks:85 },
{ name:"Amit", subject:"Science", marks:90 },
{ name:"Riya", subject:"Math", marks:78 },
{ name:"Riya", subject:"Science", marks:88 },

{ name:"John", subject:"Math", marks:92 },

{ name:"John", subject:"Science", marks:81 }

])
```

✓ Explanation

1. Each document contains name, subject, marks.
2. We insert multiple documents at once.

🔵 4️⃣ Check Inserted Data

```
db.students.find().pretty()
```

✓ Explanation

1. Shows all documents in neat, readable format.

🔵 5️⃣ Define MAP Function

```
var mapFunction = function() {

 emit(this.name, this.marks);

};
```

✓ Explanation

1. this.name → student's name as key
2. this.marks → each subject's marks
3. emit(key, value) generates pairs like:
   - ("Amit", 85)
   - ("Amit", 90)
4. Map emits multiple marks for each student.

🔵 6️⃣ Define REDUCE Function

```
var reduceFunction = function(key, values) {
 return Array.avg(values);
};
```

✓ Explanation

1. key = student name
2. values = list of marks for that student
3. Array.avg(values) → calculates average marks

Example for Amit:
values = [85, 90] → avg = 87.5

● 7️⃣ Run Map Reduce

```
db.students.mapReduce(
 mapFunction,
 reduceFunction,
 { out: "average_marks" }
)
```

✓ Explanation

1. MongoDB processes MAP and REDUCE functions.
2. Results stored in average_marks collection.
3. Automatically groups by student name.

● 8️⃣ View Results

```
db.average_marks.find().pretty()
```

✓ Expected Output

```
{ "_id":"Amit", "value":87.5 }
{ "_id":"Riya", "value":83 }
{ "_id":"John", "value":86.5 }
```

🌟 TASK 1 DONE!

🟥 TASK 2: WORD COUNT USING MAP-REDUCE

🟦 FULL CLEAN CODE (Notebook Version)

```
use textDB


db.createCollection("texts")


db.texts.insertMany([
{ line: "MongoDB map reduce example" },
{ line: "map reduce is powerful" }
])


var mapFunction = function() {
 this.line.split(" ").forEach(function(word) {
  emit(word, 1);
 });
};


var reduceFunction = function(key, values) {
 return Array.sum(values);
};


db.texts.mapReduce(
 mapFunction,
 reduceFunction,
 { out: "word_counts" }
)
```

```
db.word_counts.find().pretty()
```

🟦 CODE WITH PROPER EXPLANATION

🔵 1️⃣ Create Database

```
use textDB
```

🔵 2️⃣ Create Collection

```
db.createCollection("texts")
```

🔵 3️⃣ Insert Text Data

```
db.texts.insertMany([
{ line: "MongoDB map reduce example" },
{ line: "map reduce is powerful" }
])
```

✔ Explanation

We inserted two sentences to count occurrences of each word.

🔵 4️⃣ Map Function (Word Extraction)

```
var mapFunction = function() {
 this.line.split(" ").forEach(function(word) {
  emit(word, 1);
 });
};
```

✔ Explanation

1. this.line → picks the sentence

2. .split(" ") → breaks sentence into words
3. .forEach() → loops each word
4. emit(word, 1) →
    - Key = the word
    - Value = 1 (word appeared once)

Example Output of Map:
("map",1), ("reduce",1), ("powerful",1)

🔵 5️⃣ Reduce Function (Count Words)

```
var reduceFunction = function(key, values) {
 return Array.sum(values);
};
```

✔ Explanation

1. key = word
2. values = [1, 1, 1, 1...]
3. Array.sum(values) = total number of times word appears

🔵 6️⃣ Run Map-Reduce

```
db.texts.mapReduce(
 mapFunction,
 reduceFunction,
 { out: "word_counts" }
)
```

✔ Explanation

1. Processes all words in all documents.
2. Groups same words together.
3. Counts frequency.
4. Stores results in word_counts.

🔵 7️⃣ View Results

db.word_counts.find().pretty()

✔ Expected Output

```
{ "_id":"MongoDB", "value":1 }

{ "_id":"map", "value":2 }

{ "_id":"reduce", "value":2 }

{ "_id":"example", "value":1 }

{ "_id":"is", "value":1 }

{ "_id":"powerful", "value":1 }
```

🎉 TASK 2 DONE

🟧 EXTRA IMPORTANT QUESTIONS FOR WRITING / ORAL

✔ What is Map function?

A function that emits (key, value) pairs for each document.

✔ What is Reduce function?

A function that aggregates values with the same key.

✔ Why use Map-Reduce?

For processing large-scale data, analytics, grouping, counting, averaging.

✔ What is emit()?

A MongoDB method that sends key-value pairs to reducer.

✔ What is Array.sum()?

Built-in function to add numbers in an array.

✔ What is Array.avg()?

Built-in function to calculate average.

✔ What does { out: "collection_name" } do?

Stores the final results in a collection.

🎤 VIVA QUESTIONS (WITH ANSWERS)

1️⃣ What is Map-Reduce?

Map-Reduce is a data processing technique using Map (key-value creation) and Reduce (aggregation).

2️⃣ What does Map do?

Processes each document and emits key-value pairs.

3️⃣ What does Reduce do?

Groups values by key and performs aggregation (sum, average, count).

4️⃣ What is emit()?

Function inside Map to output (key, value).

5️⃣ Why use Map-Reduce in MongoDB?

Used for complex aggregation and large-scale data processing.

6️⃣ What is stored in the output collection?

Final processed results (key → aggregated value).

7️⃣ What does Array.sum() do?

Returns sum of all values in an array.

8️⃣ What does Array.avg() do?

Returns average of values in array.

9️⃣ Can Map-Reduce store results?

Yes, using { out: "collection_name" }.

🔟 What is the difference between Map-Reduce and Aggregation?

Aggregation is faster; Map-Reduce is more flexible for complex logic.

✓ CONCLUSION

In this practical, Map-Reduce was successfully used to calculate student average marks and perform word counting. The map functions emitted key-value pairs, and

the reduce functions aggregated the corresponding values. Results were stored in new collections for further analysis.
