```
// ====================================
// ✅ USE DATABASE
// ====================================
use te31339_db



// ====================================
// ✅ SAMPLE DATA (Accounts Collection)
// ====================================
db.accounts.drop()

db.accounts.insertMany([
   { accountNo: 1001, name: "Rahul", city: "Mumbai", balance: 30000 },
   { accountNo: 1002, name: "Priya", city: "Pune", balance: 40000 },
   { accountNo: 1003, name: "Arjun", city: "Hyderabad", balance: 90000 },
   { accountNo: 1004, name: "Riya", city: "Pune", balance: 25000 }
])



// ====================================
// ✅ MAP-REDUCE 1: TOTAL BALANCE (one output)
// ====================================

// MAP
var mapTotalBalance = function () {
   emit(null, this.balance);   // key = null because single result
```

```
};

// REDUCE
var reduceTotalBalance = function (key, values) {
   return Array.sum(values);   // sum of all balances
};

// RUN
db.accounts.mapReduce(
   mapTotalBalance,
   reduceTotalBalance,
   { out: { inline: 1 } }
);




// ====================================
// ✅ MAP-REDUCE 2: TOTAL BALANCE PER CITY
// ====================================

// MAP
var mapBalancePerCity = function () {
   emit(this.city, this.balance);   // key = city, value = balance
};

// REDUCE
```

```
var reduceBalancePerCity = function (city, balances) {
   return Array.sum(balances);      // adds all balances per city
};

// RUN
db.accounts.mapReduce(
   mapBalancePerCity,
   reduceBalancePerCity,
   { out: { inline: 1 } }
);




// ====================================
// ✅ MAP-REDUCE 3: COUNT ACCOUNTS PER CITY (easy one)
// ====================================

// MAP
var mapCountCity = function () {
   emit(this.city, 1);   // count each account as 1
};

// REDUCE
var reduceCountCity = function (city, values) {
   return Array.sum(values);   // sum of 1s = number of accounts
};
```

```
// RUN
db.accounts.mapReduce(
   mapCountCity,
   reduceCountCity,
   { out: { inline: 1 } }
);
```

Here is a **simple, clear, viva-perfect explanation** of **why we use Map-Reduce in MongoDB**.

You can speak EXACTLY these lines in your practical.

✅ Why We Use Map-Reduce in MongoDB (Simple Explanation)

Map-Reduce is used in MongoDB to perform **complex data processing** that cannot be done easily using normal queries.

It is mainly used for:

✅ 1. To process large amounts of data efficiently

When the dataset is very large, normal queries may be slow.
Map-Reduce splits the work into two steps:

- **Map:** Process each document
- **Reduce:** Combine the results

This makes it fast and scalable.

✅ 2. To perform computations that are not possible with simple queries

Examples:

- Calculating averages

- Calculating ratios
- Custom mathematical operations
- Combining multiple outputs
- Custom grouping logic

Aggregation covers many cases, but Map-Reduce is more **flexible and programmable**.

✅ 3. To run custom JavaScript logic on data

MongoDB Map-Reduce allows us to use **JavaScript functions**:

- Writing custom logic
- Applying conditions
- Doing loops or advanced calculations

This is useful when requirements are complex.

✅ 4. To perform group operations (like GROUP BY in SQL)

Map-Reduce is perfect for:

- Sum per category
- Count per category
- Maximum/minimum per category
- Average per category

Example:
Sum of balance per city → Map groups, Reduce calculates.

✅ 5. To generate summarized data

You can use Map-Reduce to:

- Generate monthly reports
- Summarize sales

- Summarize user activity
- Calculate totals, counts, averages

✅ 6. Used when Aggregation pipeline is not enough

Aggregation is faster, but sometimes:

- The logic is too complex
- You need conditional computations
- You need customized emit behavior
- You want to write your own JavaScript functions

In these cases, Map-Reduce is the solution.

✅ Short Viva Answer (Use This):

"We use Map-Reduce in MongoDB for performing complex data processing, such as sums, averages, or counts on large datasets. It allows us to write custom JavaScript logic using map and reduce functions. Map-Reduce is helpful when normal queries or simple aggregation cannot handle the required computation."