**Problem Statement-1**

Write a MapReduce program to compute the average grade of each student from a large dataset containing student IDs, subjects, and grades. Additionally, identify the highest and lowest grades for each subject.

**Step 1: Create a Sample Collection and Insert Sample Data**

use schoolDatabase;

db.grades.drop();

db.grades.insertMany([

{ studentId: 1, subject: "Math", grade: 85 },

{ studentId: 1, subject: "Science", grade: 92 },

{ studentId: 2, subject: "Math", grade: 78 },

{ studentId: 2, subject: "Science", grade: 88 },

{ studentId: 3, subject: "Math", grade: 90 },

{ studentId: 3, subject: "Science", grade: 80 },

{ studentId: 1, subject: "Math", grade: 95 },

{ studentId: 2, subject: "Math", grade: 70 }

]);

**Step 2: MapReduce Program for Average Grade and Highest/Lowest Grades by Subject**

1. **Map Function** (for average grade per student and highest/lowest grades per subject):
   * Emit the student ID for average calculation and the subject for highest/lowest grade determination.
2. **Reduce Function** (computes average, highest, and lowest grades):
   * For average, it accumulates grades and divides by count.
   * For highest/lowest, it keeps track of minimum and maximum for each subject.

// Map Function

const mapFunction = function() {

// Emit for average grade per student

emit({ type: "student", id: this.studentId }, { grade: this.grade, count: 1 });

// Emit for highest and lowest grades per subject

emit({ type: "subject", id: this.subject }, { grade: this.grade });

};

// Reduce Function

const reduceFunction = function(key, values) {

if (key.type === "student") {

// Calculate average grade for students

let totalGrades = 0;

let totalCount = 0;

values.forEach(value => {

totalGrades += value.grade;

totalCount += value.count;

});

return { avgGrade: totalGrades / totalCount };

} else if (key.type === "subject") {

// Calculate highest and lowest grade for subjects

let minGrade = Infinity;

let maxGrade = -Infinity;

values.forEach(value => {

minGrade = Math.min(minGrade, value.grade);

maxGrade = Math.max(maxGrade, value.grade);

});

return { minGrade: minGrade, maxGrade: maxGrade };

}

};

// Execute MapReduce

db.grades.mapReduce(mapFunction, reduceFunction, { out: "gradeAnalysis" });

// View Results

db.gradeAnalysis.find().pretty();

**Problem Statement-2**

Write a MapReduce program to analyze a large dataset of banking transactions. The program should compute the total transaction amount per customer, as well as identify the highest and lowest transaction amounts for each customer. This analysis will help the bank understand individual customer spending patterns and detect potential anomalies. Assumne neccesary field in collection.

**Step 1: Create a Sample Collection and Insert Sample Data**

db.transactions.drop();

db.transactions.insertMany([

{ customerId: "C1", transactionAmount: 200 },

{ customerId: "C1", transactionAmount: 500 },

{ customerId: "C2", transactionAmount: 300 },

{ customerId: "C2", transactionAmount: 150 },

{ customerId: "C3", transactionAmount: 700 },

{ customerId: "C3", transactionAmount: 300 },

{ customerId: "C1", transactionAmount: 400 },

{ customerId: "C2", transactionAmount: 50 }

]);

**Step 2: MapReduce Program for Total, Highest, and Lowest Transaction per Customer**

1. **Map Function** (for total transaction, highest, and lowest transaction):
   * Emit each transaction amount by customer.
2. **Reduce Function** (computes total, highest, and lowest):
   * Accumulates transaction totals, finds minimum, and maximum per customer.

// Map Function

const mapFunction = function() {

emit(this.customerId, { total: this.transactionAmount, max: this.transactionAmount, min: this.transactionAmount });

};

// Reduce Function

const reduceFunction = function(customerId, values) {

let totalAmount = 0;

let minAmount = Infinity;

let maxAmount = -Infinity;

values.forEach(value => {

totalAmount += value.total;

minAmount = Math.min(minAmount, value.min);

maxAmount = Math.max(maxAmount, value.max);

});

return { total: totalAmount, max: maxAmount, min: minAmount };

};

// Execute MapReduce

db.transactions.mapReduce(mapFunction, reduceFunction, { out: "transactionAnalysis" });

// View Results

db.transactionAnalysis.find().pretty();

**Problem Statement-3**

Write a MapReduce program to analyze a large dataset of e-commerce sales. The program should calculate the total number of purchases per customer and determine the maximum and minimum purchase values made by each customer. This will help the e-commerce platform gain insights into customer purchase behavior and identify potential loyal customers or high spenders

**Step 1: Create a Sample Collection and Insert Sample Data**

use ecommerceDatabase;

db.sales.drop();

db.sales.insertMany([

{ customerId: "C1", purchaseAmount: 150 },

{ customerId: "C1", purchaseAmount: 200 },

{ customerId: "C2", purchaseAmount: 300 },

{ customerId: "C2", purchaseAmount: 80 },

{ customerId: "C3", purchaseAmount: 220 },

{ customerId: "C3", purchaseAmount: 450 },

{ customerId: "C1", purchaseAmount: 100 },

{ customerId: "C2", purchaseAmount: 120 }

]);

**Step 2: MapReduce Program for Total Purchases, Maximum, and Minimum Purchase per Customer**

1. **Map Function** (for total purchases, max, and min purchase amounts):
   * Emit purchase amount per customer.
2. **Reduce Function** (calculates total purchases, max, and min purchase amounts):
   * Accumulates total purchases and determines max and min per customer.

// Map Function

const mapFunction = function() {

emit(this.customerId, { totalPurchases: 1, maxPurchase: this.purchaseAmount, minPurchase: this.purchaseAmount });

};

// Reduce Function

const reduceFunction = function(customerId, values) {

let totalPurchases = 0;

let maxPurchase = -Infinity;

let minPurchase = Infinity;

values.forEach(value => {

totalPurchases += value.totalPurchases;

maxPurchase = Math.max(maxPurchase, value.maxPurchase);

minPurchase = Math.min(minPurchase, value.minPurchase);

});

return { totalPurchases: totalPurchases, maxPurchase: maxPurchase, minPurchase: minPurchase };

};

// Execute MapReduce

db.sales.mapReduce(mapFunction, reduceFunction, { out: "salesAnalysis" });

// View Results

db.salesAnalysis.find().pretty();