Create a collection City (cityname, area, population). Write a MapReduce function to find the number of cities having populationless than 10 Lakhs and cities with poulation greater than 10 Lakhs and categorize them into categories HIGHLY POPULATED AND LOW POPULATED.

```
var mapFunction = function() {
  if (this.Subject === "DBMS") {
    if (this.Marks > 70) {
      emit("Highscores", 1);
    } else if (this.Marks > 40) {
      emit("Average scores", 1);
    } else {
      emit("Failed", 1);
    }
  }
};
var reduceFunction = function(key, values) {
  return Array.sum(values);
};
db.student.mapReduce(
  mapFunction,
  reduceFunction,
  { out: "score_summary" }
);
db.score_summary.find();
```

ANS=

Step 1: Define the Map Function

The map function will emit different categories based on the population of each city.

Step 2: Define the Reduce Function

The reduce function will aggregate the counts emitted by the map function.

Step 3: Execute the MapReduce Query

Here's the code:

javascript

Copy code

```
// Define the map function
var mapFunction = function() {
  if (this.population < 1000000) { // Population less than 10 Lakhs
    emit("LOW POPULATED", 1);
  } else { // Population greater than or equal to 10 Lakhs
    emit("HIGHLY POPULATED", 1);
  }
};

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

// Execute the MapReduce operation
db.City.mapReduce(
  mapFunction,
  reduceFunction,
  { out: "city_population_summary" } // This will store the results in a new collection
);
```

Explanation:

1. **Map Function**:
    - Checks the population of each city.
    - Emits a key-value pair based on the population:
        - "LOW POPULATED" with a value of 1 for cities with a population of less than 10 Lakhs.
        - "HIGHLY POPULATED" with a value of 1 for cities with a population of 10 Lakhs or more.

2. **Reduce Function**:
    - Takes the emitted key and an array of values (counts) and sums them to get the total count for each category.
3. **MapReduce Execution**:
    - Executes the mapReduce function on the City collection.
    - Outputs the result into a new collection called city_population_summary.

Result Verification

To see the results, you can run:

javascript

Copy code

```
db.city_population_summary.find();
```

This will display the count of cities categorized as "HIGHLY POPULATED" and "LOW POPULATED".

Example Data Entry

If you haven't already added data to your City collection, you can insert some example documents like this:

javascript

Copy code

```
db.City.insertMany([
 { cityname: "City A", area: "Area 1", population: 500000 }, // Low populated
 { cityname: "City B", area: "Area 2", population: 1200000 }, // Highly populated
 { cityname: "City C", area: "Area 3", population: 800000 }, // Low populated
 { cityname: "City D", area: "Area 4", population: 2500000 }, // Highly populated
 { cityname: "City E", area: "Area 5", population: 300000 }  // Low populated
]);
```