**Step 1: Import zip.json into MongoDB**

First, ensure you have zip.json located in a directory, for example, /path/to/your/data/zip.json.

1. **Import the zip.json file**:

Bash code:

mongoimport --db library --collection zip --file /path/to/your/data/zip.json --jsonArray

* + **Explanation**: This command imports the data from zip.json into the zip collection of the library database.
  + **Path Example**: If your file is located in C:\data\zip.json on Windows, your command would be:

Bash code:

mongoimport --db library --collection zip --file C:\data\zip.json --jsonArray

**Step 2: Create Indexes in the zip Collection**

1. **Create a Single Field Index on Population**:

Code:

use library

db.zip.createIndex({ population: 1 })

* + **Explanation**: This index helps speed up queries that filter by the population field.

1. **Create a Composite Index on State and City**:

Code:

db.zip.createIndex({ state: 1, city: 1 })

* + **Explanation**: The composite index helps improve performance for queries that filter by both state and city.

1. **Create a MultiKey Index on Location**:

Code:

db.zip.createIndex({ location: 1 })

* + **Explanation**: Use this if the location field contains an array of values. This index allows efficient querying on array fields.

**Step 3: Execute Queries Using the Indexes**

1. **Query to Display All Cities with Population Above 1600**:

Code:

db.zip.find({ population: { $gt: 1600 } }).pretty()

* + **Analysis**: This query uses the single field index on population to quickly retrieve cities with a population greater than 1600.

1. **Query to Display All Cities in the State “KS”**:

Code:

db.zip.find({ state: "KS" }).pretty()

* + **Analysis**: The composite index on state and city makes this query more efficient.

1. **Query to Display the Location of the City "TIMKEN"**:

Code:

db.zip.find({ city: "TIMKEN" }, { location: 1, \_id: 0 }).pretty()

* + **Analysis**: The composite index assists in filtering by city, and the query only returns the location field.

**Step 4: Performance Analysis**

Use the explain() method to analyze how indexes are being used for query optimization:

Code:

db.zip.find({ population: { $gt: 1600 } }).explain("executionStats")

* **Explanation**: This command shows the query execution plan, including whether indexes were used and how much time was saved.

**Notes**

* **File Path Example**: Adjust /path/to/your/data/zip.json to match the actual file path on your system.
* **Windows Example**: If you are on Windows and the file is at C:\data\zip.json, use C:\data\zip.json in your mongoimport command.

By using indexes strategically, you can optimize your queries and improve the performance of your MongoDB database.