(A) Import Restaurant.json into MongoDb and perform the following queries
1. Write a MongoDB query to display all the documents in the collection restaurants.
2. Write a MongoDB query to display the fields , restaurant_id, name, borough and cuisine for all the documents in the collection restaurant.
3. Write a MongoDB query to display the fields restaurant_id, name, borough and cuisine, but exclude the field _id for all the documents in the` collection restaurant
4. Write a MongoDB query to display the fields restaurant_id, name, borough and zip code, but exclude the field _id for all the documents in the collection restaurant
5. Write a MongoDB query to display all the restaurant which is in the borough Bronx

Sure, assuming you have a MongoDB database named `test` and a collection named `restaurants`, here are the MongoDB queries for the specified tasks:

Test Database -

```
db.restaurants.insertMany([
  { _id: 1, name: "Restaurant A", borough: "Bronx", cuisine: "Italian" },
  { _id: 2, name: "Restaurant B", borough: "Manhattan", cuisine: "American" },
  { _id: 3, name: "Restaurant C", borough: "Queens", cuisine: "Chinese" },
])
```

1. Display all documents in the collection `restaurants`:

```mongodb
db.restaurants.find({})
```

2. Display the fields `restaurant_id`, `name`, `borough`, and `cuisine` for all documents:

```mongodb
db.restaurants.find({}, { restaurant_id: 1, name: 1, borough: 1, cuisine: 1 })
```

3. Display the fields `restaurant_id`, `name`, `borough`, and `cuisine`, excluding the `_id` field:

```mongodb
db.restaurants.find({}, { restaurant_id: 1, name: 1, borough: 1, cuisine: 1, _id: 0 })
```

4. Display the fields `restaurant_id`, `name`, `borough`, and `zip code`, excluding the `_id` field:

```mongodb
db.restaurants.find({}, { restaurant_id: 1, name: 1, borough: 1, "address.zipcode": 1, _id: 0 })
```

Note: The assumption here is that the zip code information is nested under the `address` field. If it's not the case in your data structure, you may need to adjust the field path accordingly.

5. Display all restaurants in the borough `Bronx`:

```mongodb
db.restaurants.find({ borough: "Bronx" })
```

Make sure to replace `test` and `restaurants` with your actual database and collection names if they are different.

(A) Import Restaurant.json into MongoDb and perform the following queries
1. Write a MongoDB query to find the restaurant name, borough,longitude and lattitude and cuisine for those restaurants which contains 'mon' as three letters somewhere in its name
2. Write a MongoDB query to find the restaurant name, borough, longitude and latitude and cuisine for those restaurants which contain 'Mad' as first three letters of its name
3. Write a MongoDB query to find the restaurants which locate in latitude value less than -95.754168
4. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which contain 'Wil' as first three letters for its name.
5. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which contain 'ces' as last three letters for its name.

Test Database -

```
db.restaurants.insertMany([
  { _id: 1, name: "Mondo's Pizza", borough: "Bronx", cuisine: "Italian", "address": { "coord":
[40.123, -73.456] } },
  { _id: 2, name: "Madison Cafe", borough: "Manhattan", cuisine: "American", "address": {
"coord": [40.789, -73.987] } },
  { _id: 3, name: "Willow's Diner", borough: "Queens", cuisine: "Chinese", "address": { "coord":
[40.567, -73.222] } },
  { _id: 4, name: "ABC Bistro", borough: "Brooklyn", cuisine: "French", "address": { "coord":
[40.999, -73.111] } },
  { _id: 5, name: "Oceanic Seafood", borough: "Bronx", cuisine: "Seafood", "address": { "coord":
[41.234, -73.789] } }
])
```

1. Find restaurant name, borough, longitude, latitude, and cuisine for restaurants with 'mon' as three letters somewhere in the name:

```mongodb
db.restaurants.find(
  { name: { $regex: /mon/i } },
  { name: 1, borough: 1, "address.coord": 1, cuisine: 1, _id: 0 }
)
```

2. Find restaurant name, borough, longitude, latitude, and cuisine for restaurants with 'Mad' as the first three letters of the name:

```mongodb
db.restaurants.find(
  { name: { $regex: /^Mad/i } },
  { name: 1, borough: 1, "address.coord": 1, cuisine: 1, _id: 0 }
)
```

3. Find restaurants located in latitude values less than -95.754168:

```mongodb
db.restaurants.find(
  { "address.coord.1": { $lt: -95.754168 } }
)
```

4. Find restaurant Id, name, borough, and cuisine for restaurants with 'Wil' as the first three letters of the name:

```mongodb
db.restaurants.find(
  { name: { $regex: /^Wil/i } },
  { restaurant_id: 1, name: 1, borough: 1, cuisine: 1, _id: 0 }
)
```

5. Find restaurant Id, name, borough, and cuisine for restaurants with 'ces' as the last three letters of the name:

```mongodb
db.restaurants.find(
  { name: { $regex: /ces$/i } },
  { restaurant_id: 1, name: 1, borough: 1, cuisine: 1, _id: 0 }
)
```

Make sure to replace `test` and `restaurants` with your actual database and collection names if they are different. Additionally, note that these queries assume specific field names based on a common structure; you might need to adjust them according to your actual data structure.

(A) Import Restaurant.json into MongoDb and perform the following queries
1. Write a MongoDB query to find the restaurants that do not prepare any cuisine of 'American' and their grade score more than 70 and latitude less than -65.754168
2. Write a MongoDB query to find the restaurants which do not prepare any cuisine of 'American' and achieved a score more than 70 and located in the longitude less than -65.754168.
3. Write a MongoDB query to find the restaurants which do not prepare any cuisine of 'American ' and achieved a grade point 'A' does not belong to the borough Brooklyn. The document must be displayed according to the cuisine in descending order
4. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which contain 'Wil' as first three letters for its name.
5. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which contain 'ces' as last three letters for its name.

(B) Write a MongoDB query to create a backup of existing database and also create a backup of the existing database.

Test Database -

```
use test

db.restaurants.insertMany([
 {
   _id: 1,
   name: "Pizza Place",
   borough: "Bronx",
   cuisine: "Italian",
   grades: [{ grade: "A", score: 75 }],
   address: { coord: [40.123, -65.0] }
 },
 {
   _id: 2,
   name: "Burger Haven",
   borough: "Manhattan",
   cuisine: "American",
   grades: [{ grade: "B", score: 80 }],
   address: { coord: [41.0, -66.0] }
 },
 {
   _id: 3,
   name: "Sushi Delight",
   borough: "Queens",
   cuisine: "Japanese",
   grades: [{ grade: "A", score: 90 }],
   address: { coord: [39.0, -67.0] }
 },
 {
   _id: 4,
   name: "Wok Express",
   borough: "Brooklyn",
   cuisine: "Chinese",
   grades: [{ grade: "A", score: 85 }],
   address: { coord: [38.0, -68.0] }
 },
 {
   _id: 5,
   name: "Wilma's Cafe",
   borough: "Bronx",
   cuisine: "Diner",
   grades: [{ grade: "A", score: 75 }],
   address: { coord: [37.0, -69.0] }
 }
])
```

1. Find restaurants that do not prepare any cuisine of 'American', have a grade score more than 70, and latitude less than -65.754168:

```mongodb
db.restaurants.find({
  cuisine: { $ne: 'American' },
  "grades.score": { $gt: 70 },
  "address.coord.1": { $lt: -65.754168 }
})
```

2. Find restaurants that do not prepare any cuisine of 'American', have a score more than 70, and are located in longitude less than -65.754168:

```mongodb
db.restaurants.find({
  cuisine: { $ne: 'American' },
  "grades.score": { $gt: 70 },
  "address.coord.0": { $lt: -65.754168 }
})
```

3. Find restaurants that do not prepare any cuisine of 'American', achieved a grade point 'A', and do not belong to the borough Brooklyn. Display the documents according to cuisine in descending order:

```mongodb
db.restaurants.find({
  cuisine: { $ne: 'American' },
  "grades.grade": 'A',
  borough: { $ne: 'Brooklyn' }
}).sort({ cuisine: -1 })
```

4. Find restaurant Id, name, borough, and cuisine for restaurants with 'Wil' as the first three letters of the name:

```mongodb
db.restaurants.find(
  { name: { $regex: /^Wil/i } },
  { restaurant_id: 1, name: 1, borough: 1, cuisine: 1, _id: 0 }
)
```

5. Find restaurant Id, name, borough, and cuisine for restaurants with 'ces' as the last three letters of the name:

```mongodb
db.restaurants.find(
  { name: { $regex: /ces$/i } },
  { restaurant_id: 1, name: 1, borough: 1, cuisine: 1, _id: 0 }
)
```

(A) Import Restaurant.json into MongoDb and perform the following queries
1. Write a MongoDB query to display all the documents in the collection of restaurants.
2. Write a MongoDB query to display the fields, restaurant_id, name, borough and cuisine for all the documents in the collection restaurant.
3. Write a MongoDB query to display the fields restaurant_id, name, borough and cuisine, but exclude the field _id for all the documents in the collection restaurant
4. Write a MongoDB query to display the fields restaurant_id, name, borough and zip code, but exclude the field _id for all the documents in the collection restaurant
5. Write a MongoDB query to display all the restaurant which is in the borough Bronx

Test Database -

```
db.restaurants.insertMany([
 {
  _id: 1,
   name: "Pizzeria Bella",
   borough: "Bronx",
   cuisine: "Italian",
   address: { zipcode: "10453" }
 },
 {
   _id: 2,
   name: "Burger King",
   borough: "Manhattan",
   cuisine: "American",
   address: { zipcode: "10001" }
 },
 {
   _id: 3,
   name: "Sushi Palace",
   borough: "Queens",
   cuisine: "Japanese",
   address: { zipcode: "11368" }
 },
 {
   _id: 4,
   name: "Wok Express",
   borough: "Brooklyn",
   cuisine: "Chinese",
   address: { zipcode: "11201" }
 },
 {
   _id: 5,
   name: "Bronx Grill",
   borough: "Bronx",
   cuisine: "Steakhouse",
   address: { zipcode: "10467" }
 }
])
```

1. Display all documents in the collection `restaurants`:

```mongodb
db.restaurants.find({})
```

2. Display the fields `restaurant_id`, `name`, `borough`, and `cuisine` for all documents:

```mongodb
db.restaurants.find({}, { restaurant_id: 1, name: 1, borough: 1, cuisine: 1 })
```

3. Display the fields `restaurant_id`, `name`, `borough`, and `cuisine`, excluding the `_id` field:

```mongodb
db.restaurants.find({}, { restaurant_id: 1, name: 1, borough: 1, cuisine: 1, _id: 0 })
```

4. Display the fields `restaurant_id`, `name`, `borough`, and `zip code`, excluding the `_id` field:

```mongodb
db.restaurants.find({}, { restaurant_id: 1, name: 1, borough: 1, "address.zipcode": 1, _id: 0 })
```

Note: The assumption here is that the zip code information is nested under the `address` field. If it's not the case in your data structure, you may need to adjust the field path accordingly.

5. Display all restaurants in the borough `Bronx`:

```mongodb
db.restaurants.find({ borough: "Bronx" })
```

Make sure to replace `test` and `restaurants` with your actual database and collection names if they are different. Also, adjust the field paths according to your actual data structure.**(A) Import Restaurant.json into MongoDb and perform the following queries**

1. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which contain 'Reg' as three letters somewhere in its name
2. Write a MongoDB query to find the restaurants which belong to the borough Bronx and prepared either American or Chinese dish.
3. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which belong to the borough Staten Island or Queens or Bronxor Brooklyn
4. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which are not belonging to the borough Staten Island or Queens or Bronxor Brooklyn.
5. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which achieved a score which is not more than 10.

Test Database -

```
db.restaurants.insertMany([
  {
    _id: 1,
    name: "Regina's Pizza",
    borough: "Bronx",
    cuisine: "Italian",
    grades: [{ score: 85 }]
  },
  {
    _id: 2,
    name: "Bella Regal",
    borough: "Brooklyn",
    cuisine: "American",
    grades: [{ score: 92 }]
  },
  {
    _id: 3,
    name: "Szechuan Delight",
    borough: "Bronx",
    cuisine: "Chinese",
    grades: [{ score: 78 }]
  },
  {
    _id: 4,
    name: "Golden Regal Diner",
    borough: "Queens",
    cuisine: "Diner",
    grades: [{ score: 95 }]
  },
  {
    _id: 5,
    name: "Brooklyn Regency",
    borough: "Brooklyn",
    cuisine: "American",
    grades: [{ score: 70 }]
  }
])
```

1. Find restaurant Id, name, borough, and cuisine for restaurants with 'Reg' as three letters somewhere in the name:

```mongodb
db.restaurants.find(
  { name: { $regex: /Reg/i } },
  { restaurant_id: 1, name: 1, borough: 1, cuisine: 1, _id: 0 }
)
```

2. Find restaurants in the borough Bronx that prepare either American or Chinese dish:

```mongodb
db.restaurants.find(
  { borough: "Bronx", cuisine: { $in: ["American", "Chinese"] } },
  { restaurant_id: 1, name: 1, borough: 1, cuisine: 1, _id: 0 }
)
```

3. Find restaurant Id, name, borough, and cuisine for restaurants in the borough Staten Island, Queens, Bronx, or Brooklyn:

```mongodb
db.restaurants.find(
  { borough: { $in: ["Staten Island", "Queens", "Bronx", "Brooklyn"] } },
  { restaurant_id: 1, name: 1, borough: 1, cuisine: 1, _id: 0 }
)
```

4. Find restaurant Id, name, borough, and cuisine for restaurants not belonging to the borough Staten Island, Queens, Bronx, or Brooklyn:

```mongodb
db.restaurants.find(
  { borough: { $nin: ["Staten Island", "Queens", "Bronx", "Brooklyn"] } },
  { restaurant_id: 1, name: 1, borough: 1, cuisine: 1, _id: 0 }
)
```

5. Find restaurant Id, name, borough, and cuisine for restaurants with a score not more than 10:

```mongodb
db.restaurants.find(
  { "grades.score": { $not: { $gt: 10 } } },
  { restaurant_id: 1, name: 1, borough: 1, cuisine: 1, _id: 0 }
)
```

Make sure to replace `test` and `restaurants` with your actual database and collection names if they are different. Also, adjust the field paths according to your actual data structure.

(A) Import Restaurant.json into MongoDb and perform the following queries

1. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which prepared dish except 'American' and 'Chinees' or restaurant's name begins with letter 'Wil'
2. Write a MongoDB query to find the restaurant Id, name, and grades for those restaurants which achieved a grade of "A" and scored 11 on an ISODate "2014- 08-11T00:00:00Z" among many of survey dates.
3. Write a MongoDB query to find the restaurant Id, name and grades for those restaurants where the 2nd element of grades array contains a grade of "A" and score 9 on an ISODate "2014-08-11T00:00:00Z"
4. Write a MongoDB query to find the restaurant Id, name, address and geographical location for those restaurants where 2nd element of coord array contains a value which is more than 42 and upto 52.
5. Write a MongoDB query to arrange the name of the restaurants in ascending order along with all the columns.

Test Database -

```
db.restaurants.insertMany([
 {
   _id: 1,
   name: "Regina's Pizza",
   borough: "Bronx",
   cuisine: "Italian",
   grades: [
     { grade: "A", score: 85, date: ISODate("2014-08-11T00:00:00Z") },
     { grade: "B", score: 75, date: ISODate("2014-08-12T00:00:00Z") }
   ],
   address: { coord: [40.123, -73.987] }
 },
 {
   _id: 2,
   name: "Bella Regal",
   borough: "Brooklyn",
   cuisine: "American",
   grades: [
     { grade: "A", score: 92, date: ISODate("2014-08-11T00:00:00Z") },
     { grade: "B", score: 80, date: ISODate("2014-08-12T00:00:00Z") }
   ],
   address: { coord: [41.234, -73.222] }
 },
 {
   _id: 3,
   name: "Szechuan Delight",
   borough: "Bronx",
   cuisine: "Chinese",
   grades: [
     { grade: "A", score: 78, date: ISODate("2014-08-11T00:00:00Z") },
     { grade: "B", score: 85, date: ISODate("2014-08-12T00:00:00Z") }
   ],
   address: { coord: [40.567, -73.789] }
 },
 {
   _id: 4,
   name: "Golden Regal Diner",
   borough: "Queens",
   cuisine: "Diner",
   grades: [
     { grade: "A", score: 95, date: ISODate("2014-08-11T00:00:00Z") },
     { grade: "B", score: 70, date: ISODate("2014-08-12T00:00:00Z") }
   ],
   address: { coord: [39.0, -73.456] }
 },
 {
   _id: 5,
   name: "Wilma's Cafe",
   borough: "Brooklyn",
   cuisine: "American",
   grades: [
     { grade: "A", score: 88, date: ISODate("2014-08-11T00:00:00Z") },
     { grade: "B", score: 65, date: ISODate("2014-08-12T00:00:00Z") }
   ],
   address: { coord: [38.0, -73.111] }
 }
])
```

1. Find restaurant Id, name, borough, and cuisine for restaurants that prepared a dish other than 'American' and 'Chinese' or have a name starting with the letter 'Wil':

```mongodb
db.restaurants.find({
  $or: [
    { cuisine: { $nin: ['American', 'Chinese'] } },
    { name: /^Wil/ }
  ]
}, { restaurant_id: 1, name: 1, borough: 1, cuisine: 1, _id: 0 })
```

2. Find restaurant Id, name, and grades for those restaurants that achieved a grade of "A" and scored 11 on an ISODate "2014-08-11T00:00:00Z":

```mongodb
db.restaurants.find({
  "grades": {
    $elemMatch: {
      grade: "A",
      score: 11,
      date: ISODate("2014-08-11T00:00:00Z")
    }
  }
}, { restaurant_id: 1, name: 1, grades: 1, _id: 0 })
```

3. Find restaurant Id, name, and grades for those restaurants where the 2nd element of grades array contains a grade of "A" and score 9 on an ISODate "2014-08-11T00:00:00Z":

```mongodb
db.restaurants.find({
  "grades.1": { grade: "A", score: 9, date: ISODate("2014-08-11T00:00:00Z") }
}, { restaurant_id: 1, name: 1, grades: 1, _id: 0 })
```

4. Find restaurant Id, name, address, and geographical location for those restaurants where the 2nd element of coord array contains a value between 42 and 52:

```mongodb
db.restaurants.find({
  "address.coord.1": { $gt: 42, $lte: 52 }
}, { restaurant_id: 1, name: 1, address: 1, "address.coord": 1, _id: 0 })
```

5. Arrange the name of the restaurants in ascending order along with all the columns:

```mongodb
db.restaurants.find({}, { _id: 0 }).sort({ name: 1 })
```

Make sure to replace `test` and `restaurants` with your actual database and collection names if they are different. Also, adjust the field paths according to your actual data structure.

(A) Import Restaurant.json into MongoDb and perform the following queries

1. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which prepared dish except 'American' and 'Chinees' or restaurant's name begins with letter 'Wil'
2. Write a MongoDB query to find the restaurant Id, name, and grades for those restaurants which achieved a grade of "A" and scored 11 on an ISODate "2014- 08-11T00:00:00Z" among many of survey dates.
3. Write a MongoDB query to find the restaurant Id, name and grades for those restaurants where the 2nd element of grades array contains a grade of "A" and score 9 on an ISODate "2014-08-11T00:00:00Z"
4. Write a MongoDB query to find the restaurant Id, name, address and geographical location for those restaurants where 2nd element of coord array contains a value which is more than 42 and upto 52.
5. Write a MongoDB query to arrange the name of the restaurants in ascending order along with all the columns.

Test Database -

```
db.restaurants.insertMany([
 {
  restaurant_id: 1,
  name: "Regina's Pizza",
  borough: "Bronx",
  cuisine: "Italian",
  grades: [
   { grade: "A", score: 85, date: ISODate("2014-08-11T00:00:00Z") },
   { grade: "B", score: 75, date: ISODate("2014-08-12T00:00:00Z") }
  ],
  address: { coord: [40.123, -73.987] }
 },
 {
  restaurant_id: 2,
  name: "Bella Regal",
  borough: "Brooklyn",
  cuisine: "American",
  grades: [
   { grade: "A", score: 92, date: ISODate("2014-08-11T00:00:00Z") },
   { grade: "B", score: 80, date: ISODate("2014-08-12T00:00:00Z") }
  ],
  address: { coord: [41.234, -73.222] }
 },
 {
  restaurant_id: 3,
  name: "Szechuan Delight",
  borough: "Bronx",
  cuisine: "Chinese",
  grades: [
   { grade: "A", score: 78, date: ISODate("2014-08-11T00:00:00Z") },
   { grade: "B", score: 85, date: ISODate("2014-08-12T00:00:00Z") }
  ],
  address: { coord: [40.567, -73.789] }
 },
 {
  restaurant_id: 4,
  name: "Golden Regal Diner",
  borough: "Queens",
  cuisine: "Diner",
  grades: [
   { grade: "A", score: 95, date: ISODate("2014-08-11T00:00:00Z") },
   { grade: "B", score: 70, date: ISODate("2014-08-12T00:00:00Z") }
  ],
  address: { coord: [39.0, -73.456] }
 },
 {
  restaurant_id: 5,
  name: "Wilma's Cafe",
  borough: "Brooklyn",
  cuisine: "American",
  grades: [
   { grade: "A", score: 88, date: ISODate("2014-08-11T00:00:00Z") },
   { grade: "B", score: 65, date: ISODate("2014-08-12T00:00:00Z") }
  ],
  address: { coord: [38.0, -73.111] }
 }
])
```

1. Find restaurant Id, name, borough, and cuisine for restaurants that prepared a dish other than 'American' and 'Chinese' or have a name starting with the letter 'Wil':

```mongodb
db.restaurants.find({
  $or: [
    { cuisine: { $nin: ['American', 'Chinese'] } },
    { name: /^Wil/ }
  ]
}, { restaurant_id: 1, name: 1, borough: 1, cuisine: 1, _id: 0 })
```

2. Find restaurant Id, name, and grades for those restaurants that achieved a grade of "A" and scored 11 on an ISODate "2014-08-11T00:00:00Z":

```mongodb
db.restaurants.find({
  "grades": {
    $elemMatch: {
      grade: "A",
      score: 11,
      date: ISODate("2014-08-11T00:00:00Z")
    }
  }
}, { restaurant_id: 1, name: 1, grades: 1, _id: 0 })
```

3. Find restaurant Id, name, and grades for those restaurants where the 2nd element of grades array contains a grade of "A" and score 9 on an ISODate "2014-08-11T00:00:00Z":

```mongodb
db.restaurants.find({
  "grades.1": { grade: "A", score: 9, date: ISODate("2014-08-11T00:00:00Z") }
}, { restaurant_id: 1, name: 1, grades: 1, _id: 0 })
```

4. Find restaurant Id, name, address, and geographical location for those restaurants where the 2nd element of coord array contains a value between 42 and 52:

```mongodb
db.restaurants.find({
  "address.coord.1": { $gt: 42, $lte: 52 }
}, { restaurant_id: 1, name: 1, address: 1, "address.coord": 1, _id: 0 })
```

5. Arrange the name of the restaurants in ascending order along with all the columns:

```mongodb
db.restaurants.find({}, { _id: 0 }).sort({ name: 1 })
```

Make sure to replace `test` and `restaurants` with your actual database and collection names if they are different. Also, adjust the field paths according to your actual data structure.

Create a JSON file and persist it in any database.
To create a JSON file and persist it in a MongoDB database, follow these steps:

1. Create a JSON File in Notepad and save with extension .json:

```
// Sample JSON data
[
 {
   "name": "John Doe",
   "age": 30,
   "city": "New York"
 },
 {
   "name": "Jane Smith",
   "age": 25,
   "city": "Los Angeles"
 },
 {
   "name": "Bob Johnson",
   "age": 35,
   "city": "Chicago"
 }
]
```

Save the above JSON data in a file, let's say `people.json`.

2. Import JSON Data into MongoDB:

Assuming you have a MongoDB database named `test` and a collection named `people` in mongodb shell, you can use the `mongoimport` command in the command prompt to import the data:

```
mongoimport --db test --collection people --file people.json --jsonArray
```

This command imports the data from the `people.json` file into the `people` collection in the `test` database.

3. Verify Data in MongoDB:

You can use the **MongoDB shell** to verify that the data has been imported successfully:

```
mongo
use test
db.people.find()
```

This will show the documents in the `people` collection.

That's it! You have created a JSON file and persisted it in a MongoDB database. Adjust the database and collection names as needed.