<https://www.mongodb.com/docs/manual/reference/method/js-collection/>

db.collection.insertOne()

db.collection.insertMany()

|  |
| --- |
| db.collection.insertMany( |
| [ <document 1> , <document 2>, ... ], |
| { |
| writeConcern: <document>, |
| ordered: <boolean> |
| } |
| ) |

# db.collection.find()

**find()**Metoda w **MongoDB** jest narzędziem do pobierania dokumentów z kolekcji. Obsługuje różne operatory **zapytań** i umożliwia wykonywanie złożonych zapytań. Umożliwia również wybieranie określonych pól w celu optymalizacji transferu danych i korzysta z automatycznego indeksowania w celu uzyskania lepszej wydajności.

* Jest to **podstawowa** metoda pobierania dokumentów z kolekcji w **[MongoDB](https://www.geeksforgeeks.org/mongodb-tutorial/" \t "_blank)** .
* Obsługuje szeroką gamę operatorów zapytań, w tym operatory [**porównania**](https://www.geeksforgeeks.org/mongodb-comparison-query-operators/) , [**logiczne**](https://www.geeksforgeeks.org/mongodb-logical-query-operators/)i **elementów** , umożliwiając także wykonywanie złożonych zapytań.
* Umożliwia nam określenie, które pola mają zostać uwzględnione lub wykluczone w zestawie wyników i zmniejszenie ilości przesyłanych danych.
* Automatycznie używa **indeksów** w celu optymalizacji wydajności zapytań, a także poprawy ich szybkości i efektywności.

db.Collection\_name.find(kryteria\_wyboru, projekcja,opcje)

**Wyjaśnienie** :

* **db.Collection\_name.find:** Określa bazę danych ( db) i kolekcję ( Collection\_name), do której wykonujemy zapytanie, i rozpoczyna operację wyszukiwania.
* **(selection\_criteria, projection, options):** Są to opcjonalne argumenty, które zawężają wyniki:
  + **selection\_criteria(dokument):** Definiuje, które dokumenty mają zostać znalezione na podstawie określonych warunków. Pusty dokument ( {}) pobiera wszystkie dokumenty.
  + **projection(dokument):** Określa, które pola mają zostać uwzględnione lub wykluczone ze zwróconych dokumentów.
  + **options(dokument):** Umożliwia korzystanie z dodatkowych opcji, takich jak sortowanie lub ograniczanie wyników.
* [**findOne()**](https://www.geeksforgeeks.org/mongodb-findone-method/)w MongoDB jest metodą służącą do pobierania pojedynczego dokumentu ze zbioru, który spełnia określone kryteria.
* Zwraca **tylko jeden dokument** , nawet jeśli kryteria spełniają liczne dokumenty.
* Jeśli nie zostanie znaleziony żaden pasujący dokument, zwracana jest wartość null.
* Opcjonalnie możemy podać **kryteria zapytania** , aby określić, który dokument ma zostać znaleziony.

Zadanie 2: Utworzenie bazy danych i kolekcji

1. Utworzenie bazy danych o nazwie students:

use students;

1. Utworzenie kolekcji students\_info:

W MongoDB kolekcje są tworzone automatycznie przy pierwszym wstawieniu dokumentu. Możesz jednak jawnie utworzyć kolekcję, używając:

db.createCollection("students\_info");

Jeśli chcesz potwierdzić, że kolekcja została utworzona, możesz użyć:

show collections;

Zadanie 3: Wstawienie dokumentów do kolekcji

1. Wstawienie dokumentów do kolekcji students\_info:

db.students\_info.insertMany([

{ "name": "Anna Kowalska", "age": 22, "major": "Computer Science" },

{ "name": "Jan Nowak", "age": 21, "major": "Mathematics" },

{ "name": "Kasia Mazur", "age": 23, "major": "Biology" },

{ "name": "Adam Wróbel", "age": 24, "skills": ["Python", "MongoDB"], "graduated": false },

{ "name": "Ewa Wiśniewska", "age": 22, "graduated": true, "degree": "BSc" }

]);

1. Sprawdzenie wszystkich wstawionych dokumentów:

db.students\_info.find().pretty();

Zadanie 4: Odczytywanie danych (operacje find)

1. Znajdź wszystkich studentów studiujących "Computer Science":

db.students\_info.find({ "major": "Computer Science" }).pretty();

1. Znajdź studentów, którzy mają więcej niż 21 lat:

db.students\_info.find({ "age": { $gt: 21 } }).pretty();

1. Znajdź studentów, którzy posiadają pole "skills":

db.students\_info.find({ "skills": { $exists: true } }).pretty();

1. Znajdź studentów, którzy mają mniej niż 23 lata i są absolwentami (graduated: true):

db.students\_info.find({ "age": { $lt: 23 }, "graduated": true }).pretty();

Zadanie 5: Aktualizacja dokumentów (operacje update)

1. Zaktualizuj wiek Anny Kowalskiej na 23 lata:

db.students\_info.updateOne(

{ "name": "Anna Kowalska" },

{ $set: { "age": 23 } }

);

1. Dodaj nowe pole GPA o wartości 4.5 dla wszystkich studentów, którzy mają pole "skills":

db.students\_info.updateMany(

{ "skills": { $exists: true } },

{ $set: { "GPA": 4.5 } }

);

1. Zaktualizuj informacje o Janie Nowaku, dodając pole "skills" z wartością ["Mathematica", "Statistics"]:

db.students\_info.updateOne(

{ "name": "Jan Nowak" },

{ $set: { "skills": ["Mathematica", "Statistics"] } }

);

Zadanie 6: Usuwanie dokumentów (operacje delete)

1. Usuń wszystkich studentów, którzy mają więcej niż 23 lata:

db.students\_info.deleteMany({ "age": { $gt: 23 } });

1. Usuń wszystkie dokumenty, które zawierają pole graduated i mają wartość false:

db.students\_info.deleteMany({ "graduated": false });

Zadanie 7: Indeksowanie kolekcji

1. Utwórz indeks na polu name:

db.students\_info.createIndex({ "name": 1 });

1. Utwórz indeks na polu major w celu optymalizacji zapytań wyszukujących według kierunku studiów:

db.students\_info.createIndex({ "major": 1 });

1. Sprawdzenie istniejących indeksów:

Aby zobaczyć wszystkie utworzone indeksy:

db.students\_info.getIndexes();

**Structure of 'restaurants' collection:**

{

"address": {

"building": "1007",

"coord": [ -73.856077, 40.848447 ],

"street": "Morris Park Ave",

"zipcode": "10462"

},

"borough": "Bronx",

"cuisine": "Bakery",

"grades": [

{ "date": { "$date": 1393804800000 }, "grade": "A", "score": 2 },

{ "date": { "$date": 1378857600000 }, "grade": "A", "score": 6 },

{ "date": { "$date": 1358985600000 }, "grade": "A", "score": 10 },

{ "date": { "$date": 1322006400000 }, "grade": "A", "score": 9 },

{ "date": { "$date": 1299715200000 }, "grade": "B", "score": 14 }

],

"name": "Morris Park Bake Shop",

"restaurant\_id": "30075445"

}

1. Napisz zapytanie MongoDB, aby wyświetlić wszystkie dokumenty w kolekcji "restaurants".
2. Napisz zapytanie MongoDB, aby wyświetlić pola restaurant\_id, name, borough i cuisine dla wszystkich dokumentów w kolekcji "restaurant".
3. Napisz zapytanie MongoDB, aby wyświetlić pola restaurant\_id, name, borough i cuisine, ale wykluczyć pole \_id dla wszystkich dokumentów w kolekcji "restaurant".
4. Napisz zapytanie MongoDB, aby wyświetlić pola restaurant\_id, name, borough i zip code, ale wykluczyć pole \_id dla wszystkich dokumentów w kolekcji "restaurant".
5. Napisz zapytanie MongoDB, aby wyświetlić wszystkie restauracje znajdujące się w dzielnicy "Bronx".
6. Napisz zapytanie MongoDB, aby wyświetlić pierwsze 5 restauracji znajdujących się w dzielnicy "Bronx".
7. Napisz zapytanie MongoDB, aby wyświetlić kolejne 5 restauracji po pominięciu pierwszych 5 znajdujących się w dzielnicy "Bronx".
8. Napisz zapytanie MongoDB, aby znaleźć restauracje, które osiągnęły wynik większy niż 90.
9. Napisz zapytanie MongoDB, aby znaleźć restauracje, które osiągnęły wynik większy niż 80, ale mniejszy niż 100.
10. Napisz zapytanie MongoDB, aby znaleźć restauracje, które znajdują się na szerokości geograficznej mniejszej niż -95.754168.
11. Napisz zapytanie MongoDB, aby znaleźć restauracje, które nie serwują kuchni "American", mają wynik większy niż 70 i znajdują się na szerokości geograficznej mniejszej niż -65.754168.
12. Napisz zapytanie MongoDB, aby znaleźć restauracje, które nie serwują kuchni "American", mają wynik większy niż 70 i znajdują się na długości geograficznej mniejszej niż -65.754168. Uwaga: Wykonaj to zapytanie bez użycia operatora $and.
13. Napisz zapytanie MongoDB, aby znaleźć restauracje, które nie serwują kuchni "American", uzyskały ocenę "A" i nie należą do dzielnicy "Brooklyn". Dokumenty muszą być posortowane według kuchni w porządku malejącym.
14. Napisz zapytanie MongoDB, aby znaleźć restaurant\_id, name, borough i cuisine dla restauracji, których nazwa zaczyna się od pierwszych trzech liter "Wil".
15. Napisz zapytanie MongoDB, aby znaleźć restaurant\_id, name, borough i cuisine dla restauracji, których nazwa kończy się trzema literami "ces".
16. Napisz zapytanie MongoDB, aby znaleźć restaurant\_id, name, borough i cuisine dla restauracji, które mają w nazwie ciąg trzech liter "Reg".
17. Napisz zapytanie MongoDB, aby znaleźć restauracje należące do dzielnicy "Bronx" i serwujące kuchnię "American" lub "Chinese".
18. Napisz zapytanie MongoDB, aby znaleźć restaurant\_id, name, borough i cuisine dla restauracji, które należą do dzielnicy "Staten Island", "Queens", "Bronx" lub "Brooklyn".
19. Napisz zapytanie MongoDB, aby znaleźć restaurant\_id, name, borough i cuisine dla restauracji, które nie należą do dzielnicy "Staten Island", "Queens", "Bronx" lub "Brooklyn".
20. Napisz zapytanie MongoDB, aby znaleźć restaurant\_id, name, borough i cuisine dla restauracji, które uzyskały wynik nie większy niż 10.