

MongoDB

ביטויים רגולריים - המשך

Conditions

Projection

sort

skip and limit

aggregation

סיכום

TomerBu

חלוקת לקבוצות:



REGULAR EXPRESSION

`: / ^([a-z]+)([A-Z]+)$` אחד או יותר +

TEST STRING

happyCODING

הקבוצה יוצרת חלוקה - שאפשר להשתמש בה בסע

חלוקת לקבוצות:

REGULAR EXPRESSION

```
: / ^(\w+) (@) (\w+) (. ) (\w+) $
```

TEST STRING

```
hello@gmail.com|
```

וילדציה לשיסמא:

REGULAR EXPRESSION

⋮ / ^.{8,16}\$

כלתו למעט ירידת שורה

TEST STRING

{8,16}

מונה בין 8 ל-16

abcdefghijklMNPQRSTUVWXYZ

ולידציה לסיסמה:

כלתו למעט ירידת שורה

REGULAR EXPRESSION

```
: / ^(?=.*)[a-zA-Z].{8,16}$
```

{8,16}

מונה בין 8 ל 16

TEST STRING

```
AAAAAAAAAAaAAA|
```

?=

Positive Look Ahead

הסתכל קדימה וחפש בכל המחרוזת את הביטוי הבא

* . כלתו

* אפס או יותר פעמים

[a-zA-Z]

אם לא תמצא אותה קטנה
המחרוזת לא מותאמת ל regex

ולידציה לסיסמה:

REGULAR EXPRESSION

```
:/ ^.{8,16}$
```

TEST STRING

```
abcdefghSDFSDFSD
```

עד 16 תווים

REGULAR EXPRESSION

```
:/ ^(?=.*[a-z]).{8,16}$
```

TEST STRING

```
AAAAAAAAAAaAAA|
```

אותיות קטנות

REGULAR EXPRESSION

```
:/ ^(?=.*[a-z])(?=.*[A-Z]).{8,16}$
```

TEST STRING

```
AAAAAAAAAAAAAAa|
```

אותיות גדולות

REGULAR EXPRESSION

```
:/ ^(?=.*[a-z])(?=.*[A-Z])(?=.*.*[0-9]).{8,16}$
```

TEST STRING

```
AAAAAAAAAAAAAAa1|
```

מספרים

ולידציה לסיסמה:

REGULAR EXPRESSION

```
:/^(?=.*[a-z])(?=.*[A-Z])(?=.*[0-9])(?=.*[@#$%^&*]).{8,16}$
```

TEST STRING

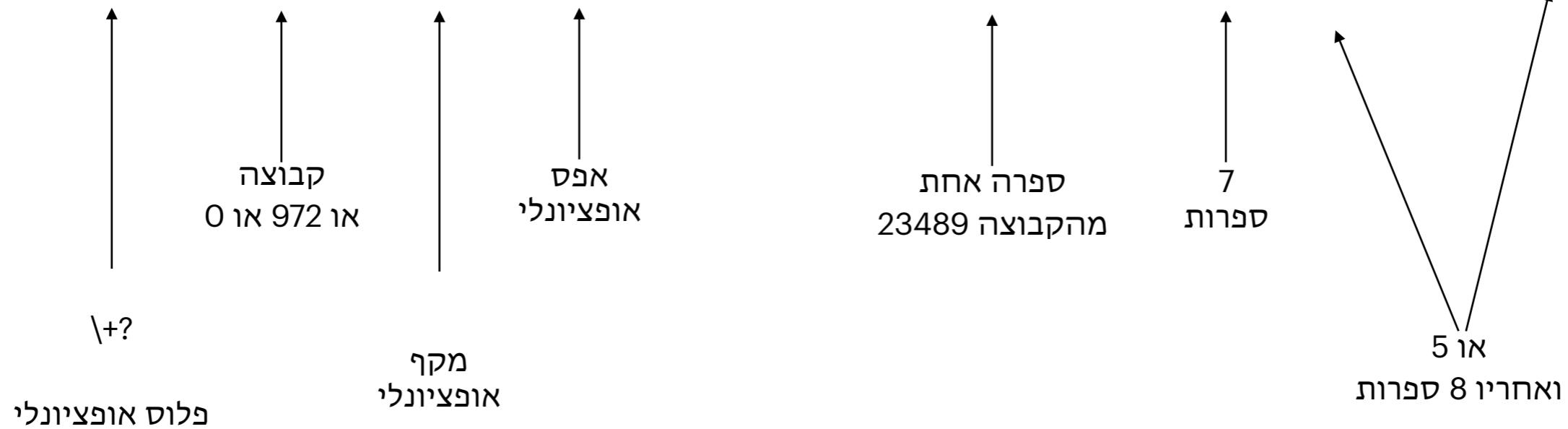
```
AAAAAAAAAA!AAa1
```

חוiot ממש משמש נוחה - לעתים נקריב את הכללים עבור UX/ואו
במערכות שמאפשרות להקל

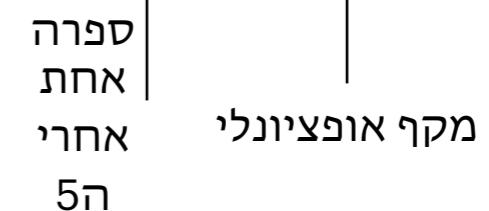
במערכות שאפשר להקל - נדרש את כל החוקים.

עבור מספר טלפון ישראלי regex

/^\\+?(972|0)(\\-)?0?(([23489]{1}\\d{7})|[5]{1}\\d{8})\$/



^\\+?(972|0)(\\-)?0?(([23489]{1}-?\\d{7})|[5]{1}\\d{1}-?\\d{7})\$



עברית מספר טלפון ישראלי regex

טלפון קוי בנפרד:

REGULAR EXPRESSION

```
:/ ^(\+972|0)[23489]-?\d{7}$
```

```
^(\+972|0)[23489]-?\d{7}$
```

TEST STRING

```
09-2312312
```

טלפון נייד:

REGULAR EXPRESSION

```
:/ ^(\+972|0)5\d{1}-?\d{7}$
```

```
^(\+972|0)5\d{1}-?\d{7}$
```

TEST STRING

```
054-4757575
```

עבור מספר טלפון ישראלי regex

`^(^(\+972|0)[23489]-?\d{7}\$)|(^(\+972|0)5\d{1}-?\d{7}\$)$`

↑
קובוצה
מספר קוי

↑
או

↑
קובוצה
מספר נייד

Regex In JS

demo.js

```
// our phone regex:  
const regex = /^(\+972|05\d{1}-?\d{7})|(\+972|0[23489]-?\d{7})$/;  
  
//test array:  
const phoneNumbers = ["08-2632213", "050-5632213"];  
  
//loop over all phoneNumbers:  
phoneNumbers.forEach((p) => {  
  const match = regex.exec(p);  
  
  if (match) {  
    console.log(match);  
  }  
});
```

1) אפשר להעתיק הכל ל[console](#) של כרום
זה יירוץ

2) אפשר להריץ את הפקודה הבאה בטרמינל:

node demo.js

הרצה של ts עם node

//: התקנה זו רציפה

// windows:

//npm i -g ts-node

// mac:

//sudo npm i -g ts-node

// run a file:

//ts-node demo.ts

Regex In JS

demo.js

```
// our phone regex:  
const regex = /^(\+972|05\d{1}-?\d{7})|(\+972|0[23489]-?\d{7})$/;  
  
//test array:  
const phoneNumbers = ["050-5632213"];  
  
//loop over all phoneNumbers:  
phoneNumbers.forEach((p) => {  
  
    const match = regex.exec(p);  
  
    if (match) {  
        //valid phone  
        console.log(`The phone you provided ${p} is valid`);  
    } else {  
        console.log(`The phone you provided ${p} does not match`)  
    }  
});
```

Regex In JS

demo.js

```
// our phone regex:  
const regex = /^(\+972|05\d{1}-?\d{7})|(\+972|0[23489]-?\d{7})$/;  
  
//test array:  
const phoneNumbers = ["08-5632213"];  
  
//loop over all phoneNumbers:  
phoneNumbers.forEach((p) => {  
    const match = regex.exec(p);  
    if (match) {  
        //valid phone  
        if (match[1]) {  
            console.log(`The phone you provided ${p} is a mobile phone`);  
        } else if (match[2]) {  
            console.log(`The phone you provided ${p} is a landline phone`);  
        }  
    } else {  
        console.log(`The phone you provided ${p} does not match`);  
    }  
});
```

אם יש קבוצות בregex
אפשר לבדוק במערך - באיזו קבוצה לפי אינדקס

דטה בוייס לשיעור:

use lec3

db.ads.insertMany([...])

```
[  
  {  
    "title": "POP show in Tel Aviv",  
    "description": "An article demo for show in tel aviv",  
    "tags": ["pop", "rock", "cool shows"],  
    "showDate": "2023-11-26T14:13:53.933Z",  
    "ticketPrice": 25,  
    "place": "Nokia palace tel aviv"  
  },  
  {  
    "title": "Shanti banti drums",  
    "description": "Text foo bla bla shanti banti drums",  
    "tags": ["shanti", "drums"],  
    "showDate": "2023-11-26T14:13:53.933Z",  
    "ticketPrice": 10,  
    "place": "Galil area"  
  },  
  {  
    "title": "Infected mushroom return to the sauce",  
    "description": "Infected mushroom best show ever",  
    "tags": ["psy", "electric", "pop"],  
    "showDate": "2023-11-26T14:13:53.933Z",  
    "ticketPrice": 63,  
    "place": "Haoman 17 jerusalem"  
  },  
  {  
    "title": "Rock with metallica",  
    "description": "Best rock show in tel aviv",  
    "tags": ["rock", "pop", "metal", "havy metal"],  
    "showDate": "2023-11-26T14:13:53.933Z",  
    "ticketPrice": 52,  
    "place": "Yarkon tel aviv"  
  }]
```

Conditions - find

And / Or

מחיר כרטיס - בין 20 כולל ל 50 כולל

גם גדול מ 20 וגם קטן מ 55

```
db.ads.find({ticketPrice: {$gte: 20, $lte: 55}})
```

```
db.ads.find({ticketPrice: {$gte: 20, $lte: 55}})
```

או מחיר 20 או שם מתחילה ב Shanti

```
db.ads.find({$or: [{ticketPrice: 20}, {title: /^Shanti/i}]})
```

```
db.ads.find({$or: [{ticketPrice: 20}, {title: /^Shanti/i}]})
```

חפש הופעות שהשם מכיל שנטי
וגם מחיר הכרטיס 50

```
db.ads.find({title: /Shanti/i, ticketPrice: 50})
```

חיפוש במערכות:

כך לא נחפש טווח במערך

```
db.inventory.insertMany([  
  { item: "journal", qty: 25, tags: ["blank", "red"], dim_cm: [ 14, 21 ] },  
  { item: "notebook", qty: 50, tags: ["red", "blank"], dim_cm: [ 14, 21 ] },  
  { item: "paper", qty: 100, tags: ["red", "blank", "plain"], dim_cm: [ 14, 21 ] },  
  { item: "planner", qty: 75, tags: ["blank", "red"], dim_cm: [ 22.85, 30 ] },  
  { item: "postcard", qty: 45, tags: ["blue"], dim_cm: [ 10, 15.25 ] }  
]);
```

dim_cm
הוא מערך
להבדיל מהדוגמה הקודמת

```
lec3 > db.inventory.find({dim_cm: {$gte: 15, $lte: 20}})
```

```
db.inventory.find({dim_cm: {$gte: 15, $lte: 20}})
```

השאילה מוצאת כל אלמנט שקטן מ-20
וגם כל אלמנט שגדול מ-20

חיפוש במערכים:

חיפוש אחר אלמנט שמקיים את כל התנאים:

```
db.inventory.insertMany([
  { item: "journal", qty: 25, tags: ["blank", "red"], dim_cm: [ 14, 21 ] },
  { item: "notebook", qty: 50, tags: ["red", "blank"], dim_cm: [ 14, 21 ] },
  { item: "paper", qty: 100, tags: ["red", "blank", "plain"], dim_cm: [ 14, 21 ] },
  { item: "planner", qty: 75, tags: ["blank", "red"], dim_cm: [ 22.85, 30 ] },
  { item: "postcard", qty: 45, tags: ["blue"] }, dim_cm: [ 10, 15.25 ]
]);|
```

```
db.inventory.find({dim_cm: {$elemMatch: {$gte: 15, $lte: 20}}})
```

```
{
  _id: ObjectId("6565b2a983d29d14e6eb32fb"),
  item: 'postcard',
  qty: 45,
  tags: [
    'blue'
  ],
  dim_cm: [
    10,
    15.25
  ]
}
```

```
db.inventory.find({dim_cm: {$elemMatch: {$gte: 15, $lte: 20}}})
```

<https://www.mongodb.com/docs/manual/tutorial/query-arrays/>

Projection:

```
SELECT *  
FROM Person
```

שאילתת שמחזירה את כל העמודות:
אין Projection - אין בחירת עמודות.
(כל העמודות)

הMETHOD find מקבלת 2 פרמטרים - filter, projection

```
> db.inventory.find({}, {item: 1})
```

↑ ↑
פילטר projection
(הקרנה מההתוצאה)

```
SELECT firstName  
FROM Person
```

בחירה - אילו עמודות

Projection:

ברירת מחדל - `_id` תמיד מוצג

1 - להציג

0 - לא להציג

אם רוצים להסתיר את `_id` חיבבים לציין זאת:

```
db.inventory.find({}, {item: 1, _id: 0})
```

פילטר - כל המסמכים באוסף

להציג את התכונה `item`

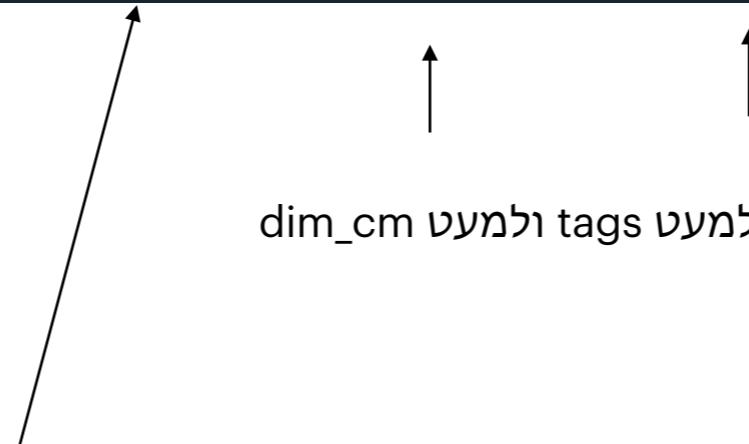
לא להציג את התכונה `_id`

Projection with Filter

```
db.inventory.find({item: /^p/i}, {item: 1, qty: 1, _id: 0})
```

Exclusion Projection

```
db.inventory.find({}, {tags: 0, dim_cm: 0})
```



first	middle	last	id
a	a	a	1
b	b	b	2
c	c	c	3

```
SELECT middle  
FROM Person  
WHERE id < 3
```

מיני תרגיל

**הציגו מוצריים מה`inventory`
ששמם מתחילה באות ז**

הציגו רק את הגודל `dim_cm` ואת שם ה`item`

```
db.inventory.find({item: /^ז/i}, {dim_cm:1, item:1, _id: 0})
```

Skip And Limit

משמש לpagination - מחזיר רק חלק מהתוצאות

skip - כמות המסמכים שעלייהם נרצה לדלג

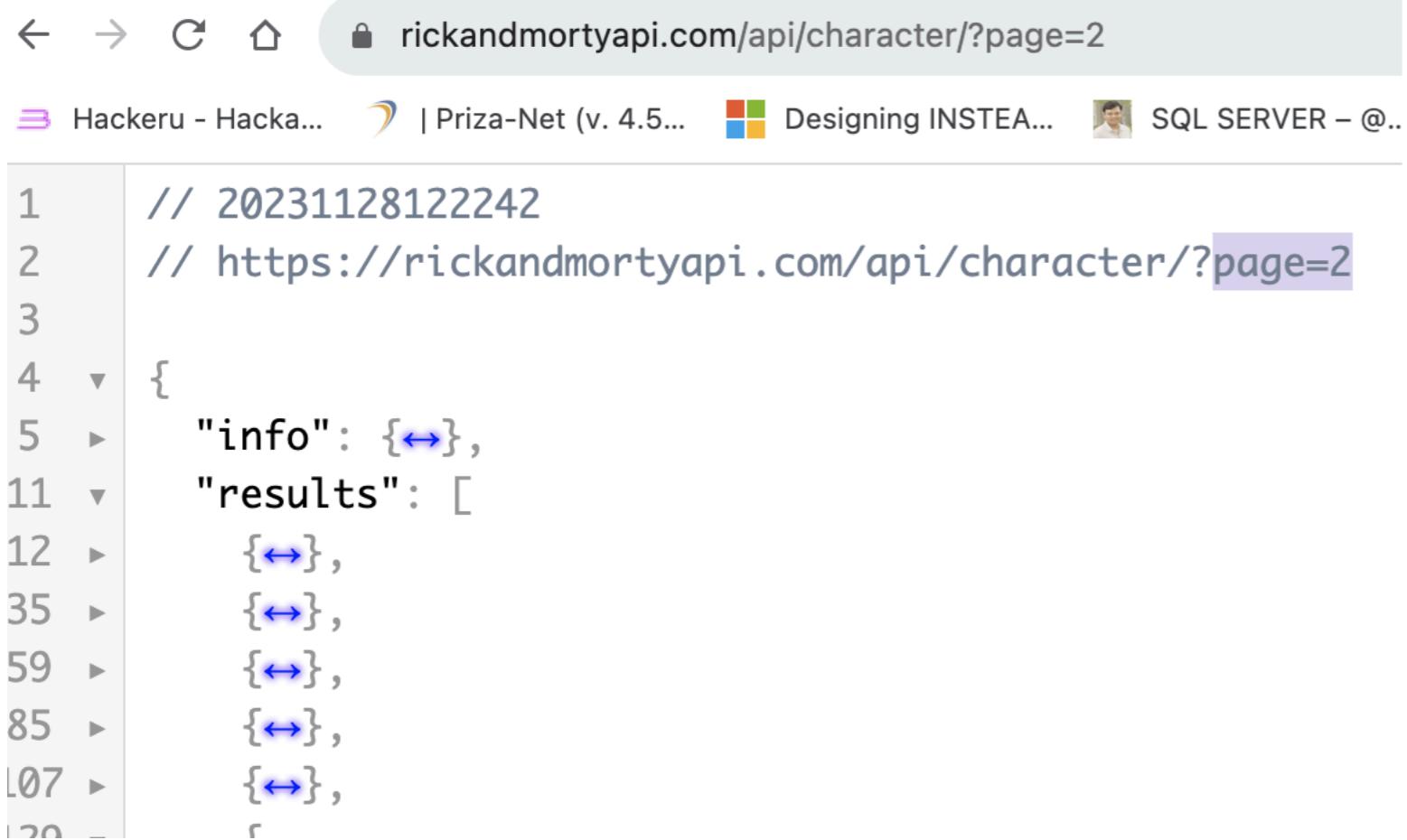
limit - כמות המסמכים שנרצה להציג

Skip And Limit

```
> db.ads.find().skip(2).limit(1)
```

```
< {  
    _id: ObjectId("6565b0b083d29d14e6eb32f5") ,  
    title: 'Infected mushroom return to the sauce' ,  
    description: 'Infected mushroom best show ever' ,  
    tags: [  
        'psy' ,  
        'electric' ,  
        'pop'  
    ] ,  
    showDate: '2023-11-26T14:13:53.933Z' ,  
    ticketPrice: 63 ,  
    place: 'Haoman 17 jerusalem'  
}
```

Skip And Limit



A screenshot of a web browser displaying a JSON response from the Rick and Morty API. The URL in the address bar is `rickandmortyapi.com/api/character/?page=2`. The JSON structure shows the first 10 characters from the second page. The code block below shows the raw JSON output.

```
1 // 20231128122242
2 // https://rickandmortyapi.com/api/character/?page=2
3
4 {
5   "info": { },
11   "results": [
12     { },
35     { },
59     { },
85     { },
L07     { },
120     { }
  ]
```

<https://rickandmortyapi.com/api/character/?page=2>

חושך תעבורה

Sort

מיון התוצאות לפי שדה מסוים:

1 סדר עולה
ASC
DESC - 1 סדר יורד

```
db.ads.find().sort({ticketPrice: 1})
```

↑
filter
projection

↑
מיון לפי מחיר בסדר עולה

Aggregations

SQL Statement:

```
SELECT city, COUNT(*) as totalCustomers  
FROM Customers  
GROUP BY City  
ORDER BY COUNT(*) DESC;
```

Edit the SQL Statement, and click "Run SQL" to see the result.

[Run SQL »](#)

Result:

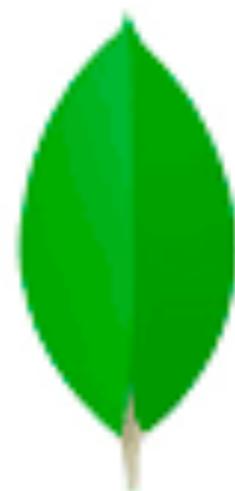
Number of Records: 69

city	totalCustomers
London	6
México D.F.	5
São Paulo	4
Buenos Aires	3

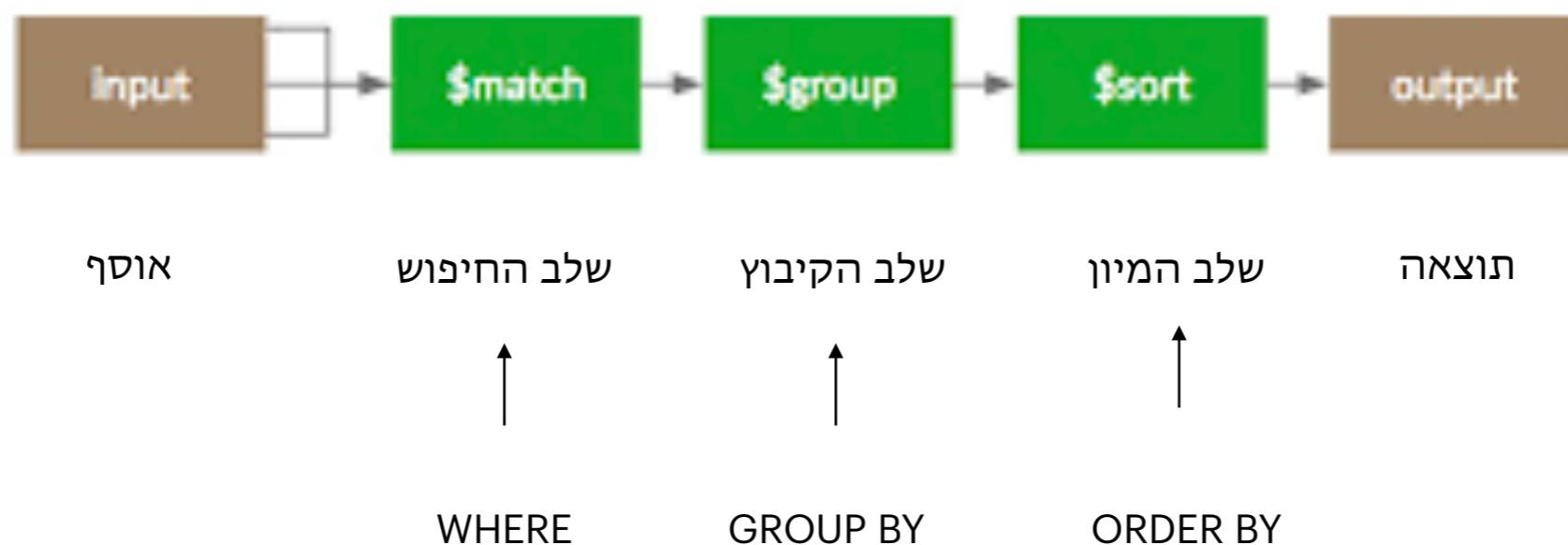
Aggregations

קיבוץ מידע ממספר מסמכים לפי מכנה משותף

חישובים לפי המידע שנוצר



mongoDB
Aggregation Framework



Aggregations

הפעולה של אוסףים - aggregate

המетодה מקבלת מערך של שלבים
כל שלב הוא אובייקט

```
{ db.ads.aggregate([]) }
```

אם נעביר מערך ריק - המетодה מתנהגת כמו find()

Aggregations

שלב `$match`

שלב של סינון ראשוני - מתנהג כמו הfilter שאנחנו מכירים מ`find`

```
db.users.aggregate([{$match: {name: /^c/i}}])
```



שלבים - אובייקטים במערך:



`db.users.aggregate([{$match: {name: /^c/i}}])`

Aggregations

שלב ה \$match

שלב של סינון ראשוני - מתנהג כמו ה filter שאנו מכירים מ find()

```
[  
  {  
    "id": 1,  
    "name": "Leanne Graham",  
    "username": "Bret",  
    "email": "Sincere@april.biz",  
    "address": {  
      "street": "Kulas Light",  
      "suite": "Apt. 556",  
      "city": "Gwenborough",  
      "zipcode": "92998-3874",  
      "geo": {  
        "lat": "-37.3159",  
        "lng": "81.1496"  
      }  
    },  
    "phone": "1-770-736-8031 x56442",  
    "website": "hildegard.org",  
    "company": {  
      "name": "Romaguera-Crona",  
      "catchPhrase": "Multi-layered client-server neural-net",  
      "bs": "harness real-time e-markets"  
    }  
  },  
  {
```

Aggregations

שלב `$group`

קיבוץ לפי מכנה משותף:

נשאיר רק את התכונות שאיתן רוצים לחשב:

```
db.users.aggregate([
    {$match: {}},
    {$group: {_id: "$address.city"}}
])
```

→ כל המטמכים

→ בחירת המכנה המשותף

```
SELECT city, COUNT(*) as totalCustomers
FROM Customers
GROUP BY City
ORDER BY COUNT(*) DESC;
```

Aggregations

שלב `$group`
עם הפעלה של פונקציית ארגזית:

```
db.users.aggregate([  
  {$match: {}},  
  {$group: {  
    _id: "$address.city", totalUsers: {$sum: 1}  
  }}  
])
```

לפי מה לקבץ

תcona מהושבת
עם פונקציית ארגזית

```
SELECT city, COUNT(*) as totalCustomers  
FROM Customers  
GROUP BY City  
ORDER BY COUNT(*) DESC;
```

אוסף חדש לדוגמאות הבאות:

```
db.orders.insertMany( [  
  { _id: 0, name: "Pepperoni", size: "small", price: 19,  
    quantity: 10, date: ISODate( "2021-03-13T08:14:30Z" ) },  
  { _id: 1, name: "Pepperoni", size: "medium", price: 20,  
    quantity: 20, date : ISODate( "2021-03-13T09:13:24Z" ) },  
  { _id: 2, name: "Pepperoni", size: "large", price: 21,  
    quantity: 30, date : ISODate( "2021-03-17T09:22:12Z" ) },  
  { _id: 3, name: "Cheese", size: "small", price: 12,  
    quantity: 15, date : ISODate( "2021-03-13T11:21:39.736Z" ) },  
  { _id: 4, name: "Cheese", size: "medium", price: 13,  
    quantity:50, date : ISODate( "2022-01-12T21:23:13.331Z" ) },  
  { _id: 5, name: "Cheese", size: "large", price: 14,  
    quantity: 10, date : ISODate( "2022-01-12T05:08:13Z" ) },  
  { _id: 6, name: "Vegan", size: "small", price: 17,  
    quantity: 10, date : ISODate( "2021-01-13T05:08:13Z" ) },  
  { _id: 7, name: "Vegan", size: "medium", price: 18,  
    quantity: 10, date : ISODate( "2021-01-13T05:10:13Z" ) }]  
]
```

עוד דוגמא: ספירה של פיצות לפי גודל

```
db.orders.aggregate([{$group: {  
    _id: "$size",  
    totalOrders: {$sum: 1}  
}}])
```

עוד דוגמא: ספירה של פיצות לפי סוג וגודל

קיבוץ לפי יותר מתכונה אחת:

```
db.orders.aggregate([{$group: {  
    _id: {size: "$size", name: "$name"},  
    totalOrders: {$sum: 1}  
}}])
```

```
< {  
    _id: {  
        size: 'small',  
        name: 'Cheese'  
    },  
    totalOrders: 1  
}
```

Aggregation Accumulators

\$min

\$max

\$count

\$sum

\$avg

Aggregation Accumulators

האופרטור `$sum`
מאפשר גם לסקום

```
db.orders.aggregate([{$group: {  
    _id: '$size',  
    totalQuantity: {$sum: '$quantity'}  
}}])
```

Aggregation Accumulators

האופרטור `$sum`
מאפשר גם לסקום וגם למספר

```
db.orders.aggregate([{$group: {  
    _id: '$size',  
    totalOrders: {$sum: 1},  
    totalQuantity: {$sum: '$quantity'}  
}}])
```

Aggregation Accumulators

האופרטור \$avg
чисוב ממוצע

```
db.orders.aggregate([{$group: {  
    _id: '$size',  
    avarageQuantity: {$avg: '$quantity'}  
}}])
```

Aggregation Accumulators

האופרטור `$min`
הזמןות עם `quantity` הבי נמור:

```
db.orders.aggregate([{$group: {  
    _id: '$size',  
    avarageQuantity: {$min: '$quantity'}  
}}])
```

האופרטור `$max`
הזמןות עם `quantity` הבי גדול:

```
db.orders.aggregate([{$group: {  
    _id: '$size',  
    maxQuantity: {$max: '$quantity'}  
}}])
```

Aggregation Accumulators

האופרטור \$count
ספרה

```
db.orders.aggregate([{$group: {  
    _id: '$size',  
    totalOrders: {$count: {}}  
}}])
```

Syntax

\$count syntax:

```
{ $count: { } }
```

אגרציה: שלב \$sort

```
db.orders.aggregate([
  {$match: {}},
  {$group: {
    _id: '$size',
    totalOrders: {$count: {}}
  }},
  {$sort: {_id: -1}}
])
```

בשלב `sort`
נתיחס לשמות שנוצרו בשלב הקודם:

קוראים לקבוצה `$_id` - לפיה קיבצנו.

אגרציה: שלב \$sort

```
db.orders.aggregate([
  {$match: {}},
  {$group: {
    _id: {name: "$name"},
    totalOrders: {$count: {}}
  }},
  {$sort: {'_id.name': -1}}
])
```



שלב \$sort

מתיחס לשמות המפתחות שנוצרו בשלב group

Schemas

בmongoo - אין חובה לסדר את הנתונים לפי סכמה.

בשלב ההתחלתי - לא נוח לעבוד עם סכמה.

כיצד ישתנו עד שנגיע למבנה הנכון מבחינת התוכנה שלנו.

סכמות נתנות יתרון - כל הנתונים ישמרו על אחידות.

<https://www.mongodb.com/docs/manual/core/schema-validation/>

Schemas

```
db.createCollection("students", {  
  validator: {  
    $jsonSchema: {  
      required: ["name"],  
      properties: {  
        name: {  
          bsonType: "string",  
          description: "'name' must be a string and is required"  
        },  
      }  
    }  
  }  
})  
  
db.students.insertOne({foo: 'bar'})
```

CREATE TABLE students(name VARCHAR(50) NOT NULL)

Schemas

```
db.createCollection("students", {  
  validator: {  
    $jsonSchema: {  
      required: ["firstName", "lastName"],  
      properties: {  
        firstName: {  
          bsonType: "string",  
          description: "'firstName' must be a string and is required"  
        },  
        lastName: {  
          bsonType: "string",  
          description: "'lastName' must be a string and is required"  
        },  
      }  
    }  
  }  
})
```

Schemas

הגדרת ערך מספרי בטוחה:

```
db.createCollection("students", {  
  validator: {  
    $jsonSchema: {  
      required: ["firstName", "lastName", "year"],  
      properties: {  
        firstName: {  
          bsonType: "string",  
          description: "'firstName' must be a string and is required"  
        },  
        lastName: {  
          bsonType: "string",  
          description: "'lastName' must be a string and is required"  
        },  
        year: {  
          bsonType: "int",  
          description: "Birth Year",  
          minimum: 1900,  
          maximum: 2100  
        }  
      }  
    }  
  }  
})
```

מספר עשרוני - double
מספר שלם - int

```
| db.students.insertOne({firstName: 'moe', lastName: 'green', year: 2022.2})
```

```
| db.students.insertOne({firstName: 'moe', lastName: 'green', year: 1871})
```

Schemas

אורך מינמלי של מחuzeות:

```
db.createCollection("students", {  
  validator: {  
    $jsonSchema: {  
      required: ["firstName", "lastName", "year"],  
      properties: {  
        firstName: {  
          bsonType: "string",  
          description: "'firstName' must be a string and is required",  
          minLength: 2  
        },  
        lastName: {  
          bsonType: "string",  
          description: "'lastName' must be a string and is required"  
        },  
        year: {  
          bsonType: "int",  
          description: "Birth Year",  
          minimum: 1900,  
          maximum: 2100  
        }  
      }  
    }  
  }  
})
```

Schemas

Enums

```
db.createCollection("students", {  
    validator: {  
        $jsonSchema: {  
            required: ["firstName", "lastName", "year"],  
            properties: {  
                firstName: {  
                    bsonType: "string",  
                    description: "'firstName' must be a string and is required",  
                    minLength: 2  
                },  
                lastName: {  
                    bsonType: "string",  
                    description: "'lastName' must be a string and is required"  
                },  
                year: {  
                    bsonType: "int",  
                    description: "Birth Year",  
                    minimum: 1900,  
                    maximum: 2100  
                },  
                country: {  
                    enum: ["UK", "USA"],  
                    description: "Must be UK or USA"  
                }  
            }  
        }  
    }  
}) db.students.insertOne({firstName: 'moe', lastName: 'green', year: 2022, country: "UK"})
```

Schemas

אסור להוסיף שדות:

```
db.createCollection("students", {  
    validator: {  
        $jsonSchema: {  
            required: ["firstName", "lastName", "year", "country", "_id"],  
            properties: {  
                _id: {  
                    bsonType: "objectId"  
                },  
                firstName: {  
                    bsonType: "string",  
                    description: "'firstName' must be a string and is required",  
                    minLength: 2  
                },  
                lastName: {  
                    bsonType: "string",  
                    description: "'lastName' must be a string and is required"  
                },  
                year: {  
                    bsonType: "int",  
                    description: "Birth Year",  
                    minimum: 1900,  
                    maximum: 2100  
                },  
                country: {  
                    enum: ["UK", "USA"],  
                    description: "Must be UK or USA"  
                }  
            },  
            additionalProperties: false  
        }  
    }  
})
```

Schemas

מותר להכניס ערך NULL

```
db.createCollection("students", {
  validator: {
    $jsonSchema: {
      required: ["firstName", "lastName", "year", "country", "_id"],
      properties: {
        _id: {
          bsonType: "objectId"
        },
        firstName: {
          bsonType: "string",
          description: "'firstName' must be a string and is required",
          minLength: 2
        },
        middleName: {
          bsonType: ["string", "null"],
          description: "middleName Must be a string or null"
        },
        lastName: {
          bsonType: "string",
          description: "'lastName' must be a string and is required"
        },
        year: {
          bsonType: "int",
          description: "Birth Year",
          minimum: 1900,
          maximum: 2100
        },
        country: {
          enum: ["UK", "USA"],
          description: "Must be UK or USA"
        }
      },
      additionalProperties: false
    }
  }
})
```

Schemas

אפשר להגדיר תבנית REGEX

```
email: {  
    bsonType: "string",  
    description: "email must end with @mongodb.com",  
    pattern: "@mongodb\\.com$"  
}
```

צפירה בחוקי וlidציה:

מציג את כל האוסףים הקיימים:

```
lec3> db.getCollectionInfos()
```

```
{  
  name: 'students',  
  type: 'collection',  
  options: { validator: [Object] },  
  info: {  
    readOnly: false,  
    uuid: new UUID("c2a6a53b-23cf-4aeb-857a-a35bed1e3da1")  
  },  
  idIndex: { v: 2, key: [Object], name: '_id' }  
},
```

פילטר - רק את students

```
db.getCollectionInfos({name: 'students'})
```

המתודה מוחזירה מערך

צפירה בחוקי וlidציה:

```
db.getCollectionInfos({name: 'students'})[0].options.validator
```

```
db.getCollectionInfos({name: 'students'})[0].options.validator.$jsonSchema
```

גישה לתוכנות של אובייקט:

```
const person = {  
  'firstName': "dave"  
}  
  
console.log(person.firstName)
```

הדרך הרגילה:

```
const person = {  
  'first Name': "dave"  
}  
  
console.log(person["first Name"])
```

דרך נוספת -
במקרה שיש מפתח
עם שם לא סטנדרטי:

Schemas

הגדרת ערך מספרי בטוחה:

BSON Types

BSON is a binary serialization format used to store documents and make remote procedure calls in MongoDB. The BSON specification is located at bsonspec.org.

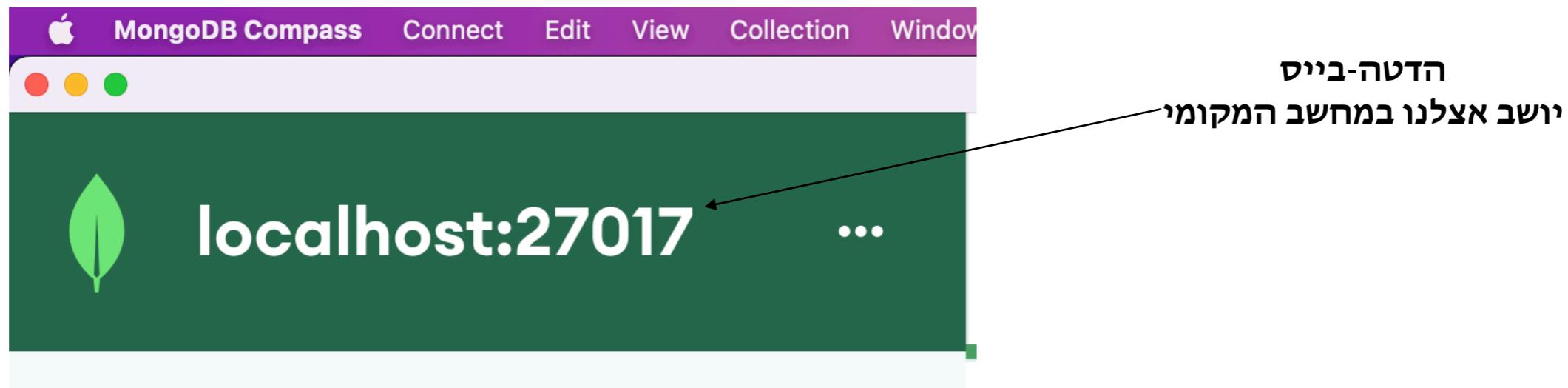
Each BSON type has both integer and string identifiers as listed in the following table:

Type	Number	Alias	Notes
Double	1	"double"	
String	2	"string"	
Object	3	"object"	
Array	4	"array"	
Binary data	5	"binData"	
Undefined	6	"undefined"	Deprecated.
ObjectId	7	"objectId"	
Boolean	8	"bool"	
Date	9	"date"	
Null	10	"null"	
Regular Expression	11	"regex"	
DBPointer	12	"dbPointer"	Deprecated.
JavaScript	13	"javascript"	
Symbol	14	"symbol"	Deprecated.
JavaScript code with scope	15	"javascriptWithScope"	Deprecated in MongoDB 4.4.
32-bit integer	16	"int"	
Timestamp	17	"timestamp"	
64-bit integer	18	"long"	
Decimal128	19	"decimal"	
Min key	-1	"minKey"	
Max key	127	"maxKey"	

הכנה לשיעורים הבאים:

סביבת פיתוח:

סביבת ענן:



מאפשר ליצור דטה-ביס בענן:

<https://www.mongodb.com/atlas/database>

נשמר את הסיסמה:

tomerbu

KD8Q22egbio5AcWE

יצירת שם משתמש וסיסמה:

- i** We autogenerated a username and password for your first database user in this project using your MongoDB Cloud registration information.



Create a database user using a username and password. Users will be given the *read and write to any database privilege* by default. You can update these permissions and/or create additional users later. Ensure these credentials are different to your MongoDB Cloud username and password.

Username

tomerbu

Password

KD8Q22egbio5AcWE

Autogenerate Secure Password

Copy

Create User

מיצאו כתובת IP אפשר להתחבר:

i We added your current IP address. You can connect to your cluster locally from this device. **X**

Add entries to your IP Access List

Only an IP address you add to your Access List will be able to connect to your project's clusters. You can manage existing IP entries via the [Network Access Page](#).

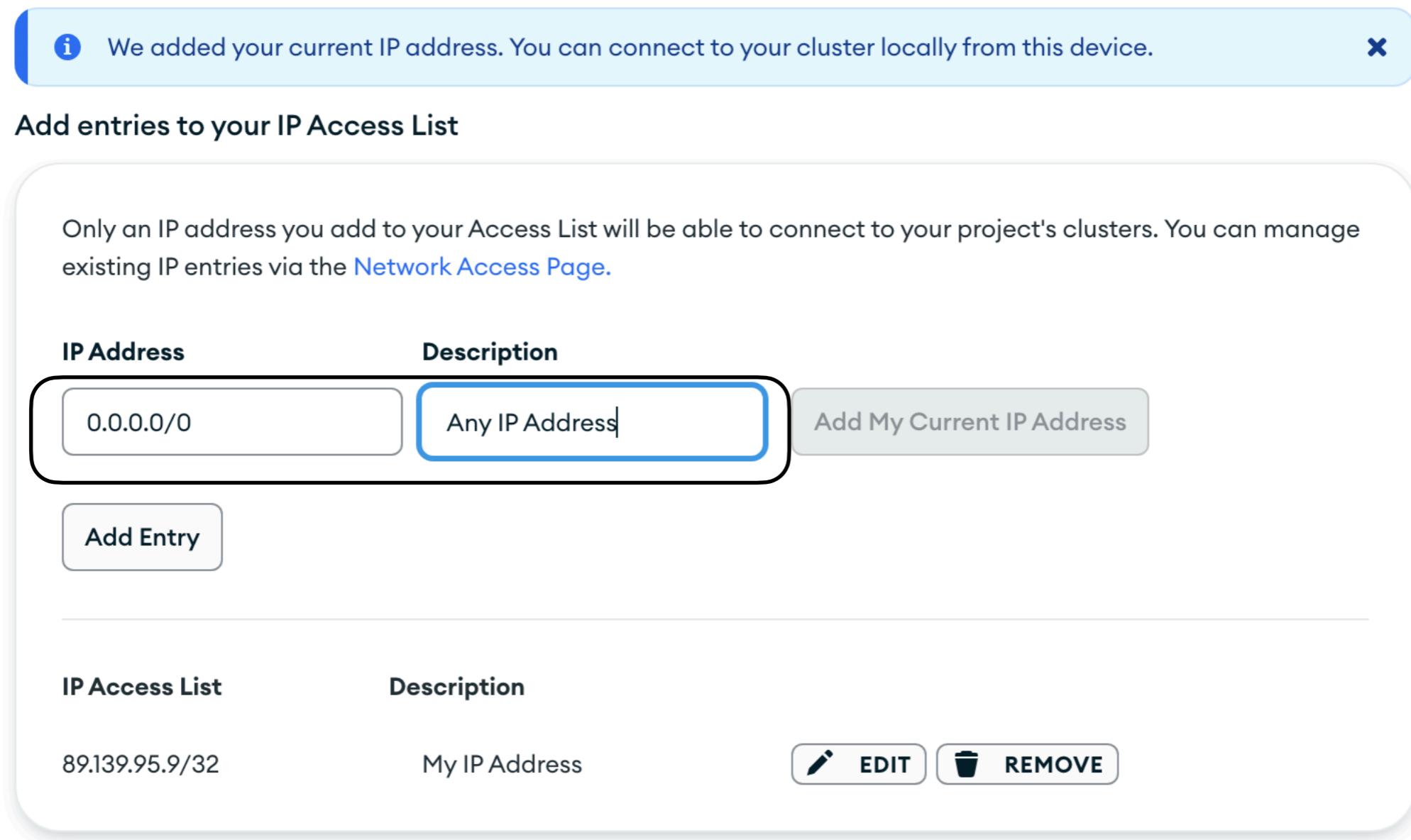
IP Address	Description
0.0.0.0/0	Any IP Address

Add Entry

Add My Current IP Address

IP Access List	Description
89.139.95.9/32	My IP Address

EDIT **REMOVE**



במציאות - נקבעו את כתובת הIP

פרטי התחברות לדטה-בייס:

TOMERBU > LEC3

Overview

Database Deployments



Cluster0

CONNECT

EDIT CONFIGURATION

FREE **SHARED**



Add Data



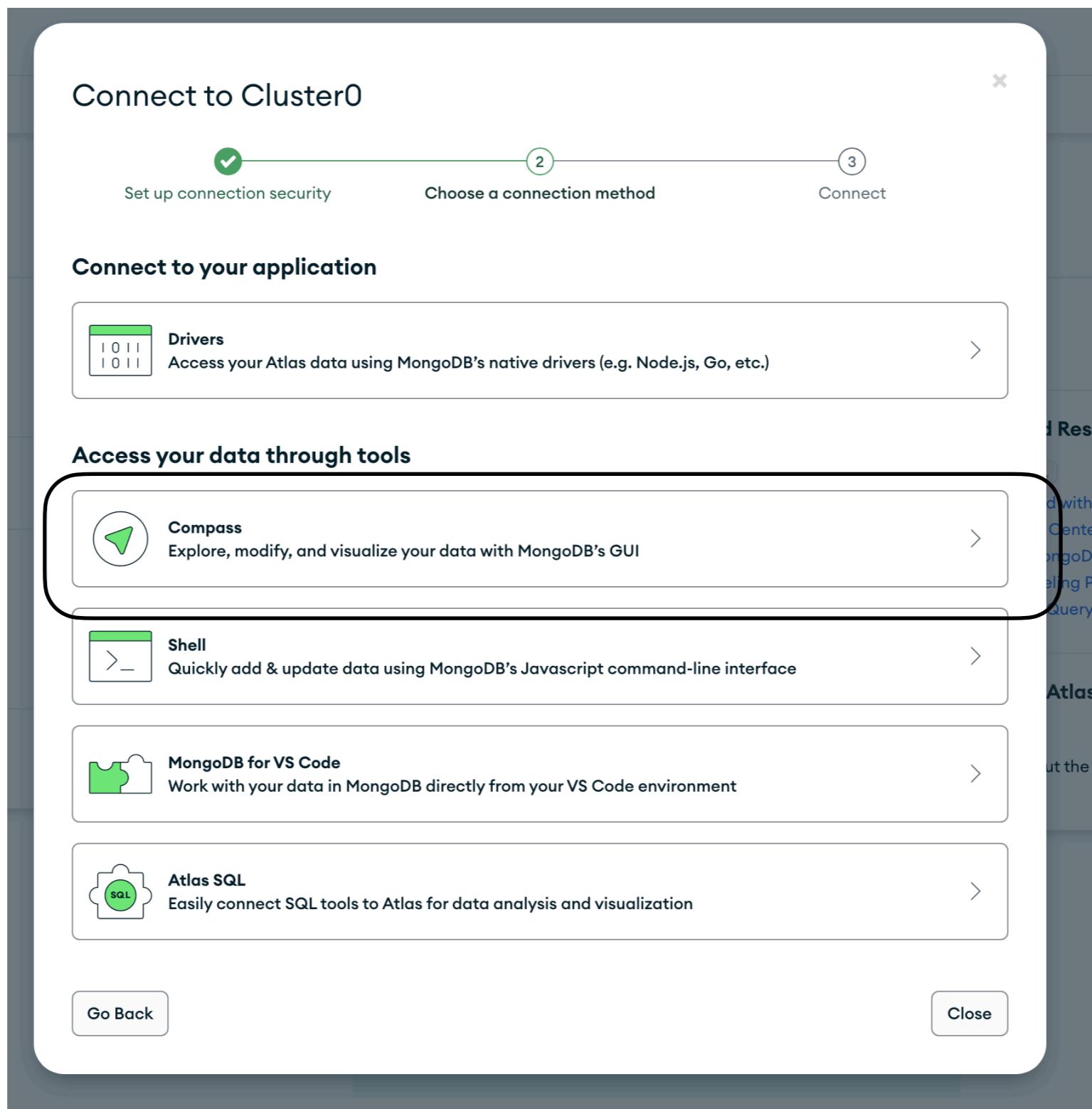
Load Sample Data



Data Modeling Templates

+ Add Tag

פרטי התחברות לדטה-ביס:



Connect to Cluster0

Set up connection security

Choose a connection method

Connect

Connecting with MongoDB Compass

I don't have MongoDB Compass installed

I have MongoDB Compass installed

1. Select your operating system and download MongoDB Compass

macOS 64-bit (10.14+)

[Download Compass \(1.40.4\)](#)

or

[Copy download URL](#)

Compass is an interactive tool for querying, optimizing, and analyzing your MongoDB data.

2. Copy the connection string, then open MongoDB Compass

`mongodb+srv://tomerbu:<password>@cluster0.gz2x7od.mongodb.net/`



Replace `<password>` with the password for the `tomerbu` user.

When entering your password, make sure that any special characters are [URL encoded](#).

RESOURCES

[Connect with Compass](#)

[Access your Database Users](#)

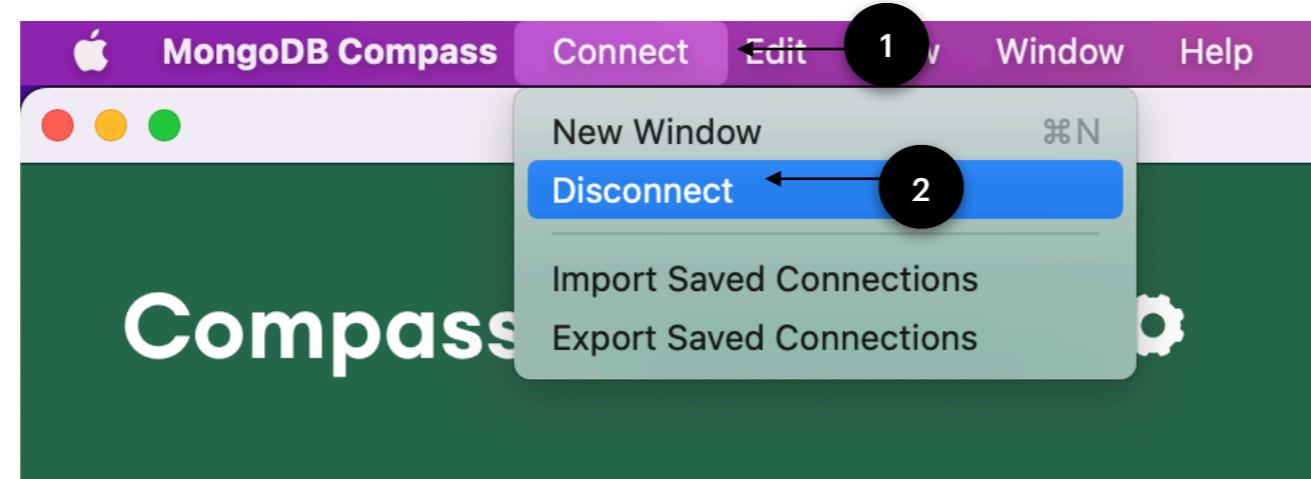
[Import and Export Data](#)

[Troubleshoot Connections](#)

[Go Back](#)

[Close](#)

התרברות לדטה-בייס מהשבד/iisan שלנו:



The screenshot shows the MongoDB Compass connection setup interface. On the left, there's a sidebar with "Compass" branding, a "New connection +" button (numbered 3), and sections for "Saved connections" (containing "Lec3CloudAtlas" with a "Never" status) and "Recents" (listing "localhost:27017" with dates Nov 13, 2023, 7:45 PM and Nov 6, 2023, 9:39 PM). The main area has a "Lec3CloudAtlas" heading, a "URI" field containing "mongodb+srv://tomerbu:*****@cluster0.gz2x7od.mongodb.net/", an "Edit Connection String" toggle switch, and "Save" and "Connect" buttons at the bottom right. A "FAVORITE" star icon is also present. The top of the screen shows a Mac OS X-style menu bar with "MongoDB Compass", "Connect", "Edit", "View", "Window", "Help", and system icons.

שיעור בית

בכל השאלות יש להשתמש בaggregate

```
db.cities.insertMany[
```

```
  {"name": "Seoul", "country": "South Korea", "continent": "Asia", "population": 25.674 },  
  {"name": "Mumbai", "country": "India", "continent": "Asia", "population": 19.980 },  
  {"name": "Lagos", "country": "Nigeria", "continent": "Africa", "population": 13.463 },  
  {"name": "Beijing", "country": "China", "continent": "Asia", "population": 19.618 },  
  {"name": "Shanghai", "country": "China", "continent": "Asia", "population": 25.582 },  
  {"name": "Osaka", "country": "Japan", "continent": "Asia", "population": 19.281 },  
  {"name": "Cairo", "country": "Egypt", "continent": "Africa", "population": 20.076 },  
  {"name": "Tokyo", "country": "Japan", "continent": "Asia", "population": 37.400 },  
  {"name": "Karachi", "country": "Pakistan", "continent": "Asia", "population": 15.400 },  
  {"name": "Dhaka", "country": "Bangladesh", "continent": "Asia", "population": 19.578 },  
  {"name": "Rio de Janeiro", "country": "Brazil", "continent": "South America", "population": 13.293 },  
  {"name": "São Paulo", "country": "Brazil", "continent": "South America", "population": 21.650 },  
  {"name": "Mexico City", "country": "Mexico", "continent": "North America", "population": 21.581 },  
  {"name": "Delhi", "country": "India", "continent": "Asia", "population": 28.514 },  
  {"name": "Buenos Aires", "country": "Argentina", "continent": "South America", "population": 14.967 },  
  {"name": "Kolkata", "country": "India", "continent": "Asia", "population": 14.681 },  
  {"name": "New York", "country": "United States", "continent": "North America", "population": 18.819 },  
  {"name": "Manila", "country": "Philippines", "continent": "Asia", "population": 13.482 },  
  {"name": "Chongqing", "country": "China", "continent": "Asia", "population": 14.838 },  
  {"name": "Istanbul", "country": "Turkey", "continent": "Europe", "population": 14.751 }
```

```
]
```

הציגו רק ערים ביבשת אירופה (\$match 1)

הציגו רק ערים ביבשות אירופה או צפון אמריקה (\$match 2)

שיעור בית

בכל השאלות יש להשתמש בaggregate

3 הציגו את כל הערים לפי גודל האוכלוסייה בסדר יורד (\$sort)

4 הציגו רק ערים באירופה לפי גודל האוכלוסייה בסדר יורד

5 הציגו רק ערים בצפון אמריקה לפי גודל האוכלוסייה בסדר יורד

6 בצעו קיבוץ לפי יבשת - (\$group)

7 בצעו קיבוץ לפי יבשת ומדינה - (group\$)

8 בצעו קיבוץ לפי יבשת
יש להציג את כמות האנשים ביבשת
sum של \$population

9 בצעו קיבוץ לפי יבשת ומדינה - (group\$)
יש להציג את הערך המקסימלי של population
(יציג את כמות התושבים בעיר שבה יש הכى הרובה תושבים לפי יבשת ומדינה)

10 באיזו עיר יש הכى הרובה תושבים (מכל הערים הקיימות)