

## 1/1 MongoDB feladatok

1. Készítsen lekérdezést, amely csak az user\_id, firstName és lastname oszlopokat jeleníti meg!

Query	Result
<pre>1 db.collection.find({}, 2 { 3   "user_id": 1, 4   "firstName": 1, 5   "lastName": 1, 6   "_id": 0 7 })</pre>	<pre>[   {     "firstName": "Grace",     "lastName": "Hopper",     "user_id": 2   },   {     "firstName": "Caderyn",     "user_id": 1   } ]</pre>

2. kérdezze le a Grace keresztnévű felhasználó email-címét és jelszavát (csak ez a két mező jelenjen meg)

Query	Result
<pre>1 db.collection.find({ 2   "firstName": "Grace" 3 }, 4 { 5   "email": 1, 6   "password": 1, 7   "_id": 0 8 })</pre>	<pre>[   {     "email": "grace@navy.mil",     "password": "C8w4\u0026CWC^egwecwoWei79chwF"   } ]</pre>

3. Az előző feladatban kiválasztott grades gyűjteményből kérdezze le a 339-es azonosítójú osztály eredményeit!
  - a. A listában csak azok a dokumentumok jelenjenek meg, ahol a tanuló azonosítója 100 alatt van!

<b>FILTER</b>	{class_id: 339, student_id: {\$lt: 100}}
<b>PROJECT</b>	{ field: 0 }
<b>SORT</b>	{ field: -1 }
<b>COLLATION</b>	{ locale: 'simple' }

4. kérdezze le az előző feladatban létrehozott receptek gyűjtemény azon dokumentumait, amelyre teljesül:
- A lájkok száma több, mint 2!
  - A lista legyen sorbarendezve a főzési idő szerint csökkenő sorrendben!  
(A rendezés funkció az Options gomb lenyomása után érhető el)
  - A listában ne jelenjenek meg az ingredients és a rating mezők (Project szakasznál kell beállítani)!



5. készítsen új indexet a 7. feladatban importált receptek gyűjteményhez az Indexes rész Create Index funkciójának segítségével!
- Az index neve legyen `i_title`, és a `title` mező szerint csökkenő legyen
  - Az index egyedi (unique) legyen (Options rész)!

## MongoShell

6. kérdezzük le, hogy a receptek gyűjteményben mely dokumentumoknál szerepel a recept nevében (`title`) a Tacos szó!

```
Atlas atlas-br9unk-shard-0 [primary] myFirstDatabase> use gyak_compass
switched to db gyak_compass
Atlas atlas-br9unk-shard-0 [primary] gyak_compass> db.receptek.find({"title": {$regex: /Tacos/}}, {}).pretty()
```

7. kérdezzük le, hogy recept típusonként (`type`) mennyi a főzési idők (`cook_time`) összege!

```
Atlas atlas-br9unk-shard-0 [primary] gyak_compass>
Atlas atlas-br9unk-shard-0 [primary] gyak_compass> db.receptek.aggregate([
... {$group:
... {_id: "$type", total: {$sum: "$cook_time"}}
... }
... ])
```

8. kérdezzük le, hogy a receptek gyűjteményben **hány olyan dokumentum van**, ahol:
- A recept 4 főre szól (servings) ÉS
  - A tag-ek között szerepel a "quick" vagy az "easy" (legalább az egyik)

```
Atlas atlas-br9unk-shard-0 [primary] gyak_compass>
```

```
Atlas atlas-br9unk-shard-0 [primary] gyak_compass> db.receptek.find({$and:  
... [{servings: 4}, {tags: {$in: ["quick", "easy"]}}]}, {})_
```

→ .count()

```
Atlas atlas-br9unk-shard-0 [primary] gyak_compass> db.receptek.find({ $and: [{ servings: 4  
}, { tags: { $in: ["quick", "easy"] } } ] }, {}).count()  
1
```

9. a receptek gyűjteményben a ObjectId("5e878f5220a4f574c0aa56db") azonosítójú dokumentum esetén módosítsuk a főzési időt (cook\_time) 33 percre!

```
Atlas atlas-br9unk-shard-0 [primary] gyak_compass>
```

```
Atlas atlas-br9unk-shard-0 [primary] gyak_compass> db.receptek.updateOne(  
... {"_id": ObjectId("5e878f5220a4f574c0aa56db")},  
... {$set: {"cook_time": 33}}  
... )
```

Ell.:

```
Atlas atlas-br9unk-shard-0 [primary] gyak_compass> db.receptek.find(  
... {"_id": ObjectId("5e878f5220a4f574c0aa56db")}, {})_
```

10. adjunk hozzá a ObjectId("5e5e9c470d33e9e8e3891b35") azonosítójú dokumentum likes tömbjéhez még egy értéket, mégpedig a 200-at!

```
Atlas atlas-br9unk-shard-0 [primary] gyak_compass>
```

```
Atlas atlas-br9unk-shard-0 [primary] gyak_compass> db.receptek.updateOne(  
... {_id: ObjectId("5e5e9c470d33e9e8e3891b35")},  
... {$push: {likes: 200}}  
... )
```

Ell.:

```
Atlas atlas-br9unk-shard-0 [primary] gyak_compass> db.receptek.find( { _id: ObjectId("5e5e  
9c470d33e9e8e3891b35") }, {})
```

## 1/2 Neo4J feladatok

1. A Neo4J Sandbox Movie adatbázisából kérdezze le azon személyek nevét és születési évét, akik 1964-ben vagy 1965-ben születtek!

```
1 MATCH (p:Person)
2 WHERE p.born = 1964 OR p.born = 1965
3 RETURN p.name, p.born
```

2. kérdezze le azon filmek címét és megjelenési évét, amelyek címe A-betűvel kezdődik! (WHERE, STARTS WITH).

- a. A listát rendezzük a megjelenési év szerint csökkenő sorrendbe (ORDER BY)!

```
1 MATCH (m:Movie)
2 WHERE m.title STARTS WITH 'A'
3 RETURN m.title, m.released
4 ORDER BY m.released DESC
```

3. kérdezze le, hogy milyen filmeket készített (:PRODUCED) Joel Silver!
  - a. Csak a filmek címe jelenjen meg

```
1 MATCH (p:Person)-[:PRODUCED]→(m:Movie)
2 WHERE p.name = 'Joel Silver'
3 RETURN m.title
```

VAGY

```
1 MATCH (p:Person {name: 'Joel Silver'})-[:PRODUCED]→(m:Movie)
2 //WHERE p.name = 'Joel Silver'
3 RETURN m.title
```

4. kérdezze le, hogy melyik rendező hány filmet rendezett! (:DIRECTED).
- a. Csak azokat a rendezőket jelenítsük meg, akik 1-nél több filmet rendeztek! (WHERE)

```
1 MATCH (p:Person)-[:DIRECTED]→(m:Movie)
2 WITH p.name AS `nev`, COUNT(*) AS `db`
3 WHERE db > 1
4 RETURN nev, db
```

→ a WITH használatánál muszáj alias neveket használni!

5. jelenítsük meg azokat a személyeket, akik egyszerre voltak szereplők és rendezők is valamely filmben!

```
1 MATCH (p:Person)-[:ACTED_IN]→(m:Movie)
2 WHERE EXISTS ((p:Person)-[:DIRECTED]→(m:Movie))
3 RETURN (p)
```

6. kérdezze le, hogy mely filmek hány szereplője van!
- a. A lista legyen sorba rendezve a szereplők száma szerint csökkenően, azon belül a film címe szerint növekvően

```
1 MATCH (p:Person)-[:ACTED_IN]→(m:Movie)
2 RETURN m.title, COUNT(*)
3 ORDER BY COUNT(*) DESC, m.title
4 LIMIT 10
```

7. készítsen új indexet i\_person\_born néven!
- a. Az indexelés a Person csúcsokra történjen név és születési idő szerint!

```
1 CREATE INDEX i_person_born
2 FOR (p:Person)
3 ON (p.name, p.born)
```

Ell.: :schema



8. jelenítse meg a személyek nevét és születési évét!

a. A listát szűrje az 1980 és 2000 között született személyekre!

b. Az utasítás végrehajtásával együtt jelenjen meg a végrehajtási terv is!

```
1 PROFILE
2 MATCH (p:Person)
3 WHERE p.born ≥ 1980 AND p.born ≤ 2000
4 RETURN p.name, p.born
```

→csak a végrehajtási terv: *PROFILE* helyett *EXPLAIN*

# MINTAZH

MENT

DATABASES: 11 COLLECTIONS: 25

+ Create Database

🔍 Search Namespaces

▼ Gyak\_compass

receptek

▶ gyak8

▶ sample\_airbnb

▶ sample\_analytics

- ▶ sample\_geospatial

▶ sample\_guides

▶ sample\_mflix

▶ sample\_restaurants

▶ sample\_supplies

- ▶ sample\_training

▶ sample\_weatherdata

Access

ress

A MongoDB Atlas-ban (vagy a Compass-ban) navigáljon a sample\_mflix adatbázishoz, és kérdezze le az embedded\_movies gyűjteményt az alábbiak szerint:

- A műfaj besorolások közül a legelső Akció (Action) legyen ÉS
- Az év (year) 1975 utáni
- Az írók, a szereplők és a rendezők ne jelenjenek meg
- Rendezzük sorban a filmeket futási idő (runtime) szerint növekvően, azon belül év szerint csökkenően

[Hátralé](#)

Beadandó: a Filter, Project és a Sort részbe írt kódok

Filter 

```
{ genres:"Action",year:{$gt: 1975}}
```

Project

```
{writers:0, directors:0, cast:0}
```

Sort

```
{runtime: -1}
```

Collation

```
{ locale: 'simple' }
```

A MongoDB shell-ben csatlakozzon a sample\_mflix adatbázishoz, és kérdezze le a movies gyűjteményt az alábbiak szerint:

- Csak a film címe, megjelenés éve és a szereplők jelenjenek meg
- A szereplők között legyen "Billie Dove"

Beadandó: a megfelelő utasítás

```
db.movies.find(
  { "cast": { "$in": ["Billie Dove"] } },
  { "title": 1, "year": 1, "cast": 1, "_id": 0 }
)
```

```
Atlas atlas-3961vs-shard-0 [primary] sample_mflix> db.movies.find(
...   { "cast": { "$in": ["Billie Dove"] } },
...   { "title": 1, "year": 1, "cast": 1, "_id": 0 }
... )
```



A MongoDB shell-ben csatlakozzon a sample\_mflix adatbázishoz, és kérdezze le a movies gyűjteményt az alábbiak szerint:

- A lekérdezés feleljen meg a következő SQL lekérdezésnek:

```
SELECT cast, AVG(num_mflix_comments)
FROM movies
GROUP BY cast
HAVING AVG(num_mflix_comments) > 3
ORDER BY AVG(num_mflix_comments)
```

A megoldáshoz a \$group előtt át kell alakítani a cast tömböt a következő módon: {\$unwind: "\$cast"}

Beadandó a lekérdező utasítás



```
db.movies.aggregate([
  { $unwind: "$cast" },
  { $group: {
    _id: "$cast",
    avgComments: { $avg: "$num_mflix_comments" }
  }},
  { $match: {
    avgComments: { $gt: 3 }
  }},
  { $sort: {
    avgComments: 1
  }}
])
```

```
Atlas atlas-3961vs-shard-0 [primary] sample_mflix> db.movies.aggregate([
...   { $unwind: "$cast" },
...   { $group: {
...     _id: "$cast",
...     avgComments: { $avg: "$num_mflix_comments" }
...   }},
...   { $match: {
...     avgComments: { $gt: 3 }
...   }},
...   { $sort: {
...     avgComments: 1
...   }}
... ])
```

Csatlakozzon a Movies adatbázishoz a Neo4J Sandbox-ban (vagy a Desktop-ban), és kérdezze le a következőket:

- Melyik filmben hányan szerepeltek?
- Csak a filmek címe és a szereplők száma jelenjen meg
- Rendezzük a listát a szereplők száma szerint csökkenően
- Csak a legelső 5 jelenjen meg
- Hagyjuk ki azokat a filmeket, amelyek címében a "Speed" szó benne van!

Beadandó: a megfelelő utasítás

```
MATCH (p:Person)-[:ACTED_IN]→(m:Movie)
WHERE NOT m.title CONTAINS 'Speed'
WITH m.title AS Movie, count(p) AS `Number of Actors`
RETURN Movie, `Number of Actors`
ORDER BY `Number of Actors` DESC
LIMIT 5
```

A neo4j Sandbox-ban (vagy a Desktop-ban) csatlakozzon a Movies adatbázishoz, és kérdezze le a következő SQL-lekérdezésnek megfelelő adatokat:

```
SELECT p.name, COUNT(*)
FROM Movies m JOIN Person p ON m.writer_id = p.id
WHERE p.name LIKE 'L%'
GROUP BY p.name
HAVING COUNT(*) > 1
```

```
MATCH (p:Person)-[:WROTE]→(m:Movie)
WHERE p.name STARTS WITH 'L'
WITH p.name AS Writer, count(m) AS MoviesCount
WHERE MoviesCount > 1
RETURN Writer, MoviesCount
ORDER BY MoviesCount DESC
```