QUERIES EXECUTADAS SEM ÍNDICE:

1. busca: todos os filmes.
   comando: db.filmes.find().explain("executionStats")
   nReturned: 10206,
   executionTimeMillis: 3

2. busca: todos os filmes que foram lançados em 2024
   comando: db.filmes.find({"ano": 2024}).explain("executionStats")
   nReturned: 900,
   executionTimeMillis: 5

3. busca: todos os filmes que o ator Anthony Anderson participou como coadjuvante.
   comando: db.filmes.find({"atoresCoadjuvantes": { "$in": ["Anthony Anderson"]
}}).explain("executionStats")
   nReturned: 14,
   executionTimeMillis: 9

4. busca: todos os filmes que algum 'DiCaprio', tenha feito como ator principal.
   comando: db.filmes.find({"atorPrincipal.nome": {$regex: "DiCaprio", $options:
"i"}}).explain("executionStats")
   nReturned: 27,
   executionTimeMillis: 11,

5. busca: todos os filmes que o ator principal seja Canadense ou Australiano
   comando: db.filmes.find({
                $or: [
                  { "atorPrincipal.nacionalidade": "Canadense" },
                  { "atorPrincipal.nacionalidade": "Australiano" }
                ]
            }).explain("executionStats")
   nReturned: 5071,
   executionTimeMillis: 7

6. busca: todos os filmes do gênero "Action" com nota maior que 7.5
   comando: db.filmes.find({
                genero: "Action",
                nota: { $gt: 7.5 }
            }).explain("executionStats")
   nReturned: 136,
   executionTimeMillis: 7

7. busca: todos os filmes dos gêneros "Action" ou "Drama"
   comando: db.filmes.find({
                genero: { $in: ["Action", "Drama"] }
            }).explain("executionStats")
   nReturned: 3752,
   executionTimeMillis: 6

8. busca: todos os filmes com nota entre 5 e 8
   comando: db.filmes.find({
            nota: { $gte: 5, $lte: 8 }
        }).explain("executionStats")
   nReturned: 9058,
   executionTimeMillis: 8

9. busca: todos os filmes cujo diretor começa com "S"
   comando: db.filmes.find({
            diretor: { $regex: "^S", $options: "i" }
        }).explain("executionStats")
   nReturned: 872,
   executionTimeMillis: 7

10. busca: todos os filmes lançados em 2023
   comando: db.filmes.find({
            dataLancamento: { $regex: "^2023" }
        }).explain("executionStats")
   nReturned: 557,
   executionTimeMillis: 8

11. busca: todos os filmes onde o ator principal tem mais de 40 anos
   comando: db.filmes.find({
            "atorPrincipal.idade": { $gt: 40 }
        }).explain("executionStats")
   nReturned: 6655,
   executionTimeMillis: 6

12. busca: todos os filmes com mais de 3 atores coadjuvantes
   comando: db.filmes.find({
            $expr: { $gt: [{ $size: "$atoresCoadjuvantes" }, 3] }
        }).explain("executionStats")
   nReturned: 9861,
   executionTimeMillis: 8

13. busca: todos os filmes lançados em 2020 ou 2021
   comando: db.filmes.find({
            $or: [
               { ano: 2020 },
               { ano: 2021 }
            ]
        }).explain("executionStats")
   nReturned: 649,
   executionTimeMillis: 5

14. busca: Busca: todos os filmes que não são do gênero "Comedy"
   comando: db.filmes.find({

```
            genero: { $ne: "Comedy" }
        }).explain("executionStats")
nReturned: 8883,
executionTimeMillis: 6
```

15. busca: contar o número de filmes por gênero

```
comando:    db.runCommand({
            explain: {
                aggregate: "filmes",
                pipeline: [
                    { $group: { _id: "$genero", total: { $sum: 1 } } }
                ],
                cursor: {}
            },
            verbosity: "executionStats"
        });
nReturned: 20,
executionTimeMillis: 13
```

16. busca: encontrar filmes dirigidos por alguém com "Chad" no nome e "Wick" no título, ordenados por ano decrescente

```
comando:    db.runCommand({
            explain: {
                aggregate: "filmes",
                pipeline: [
                { $match: {
                    diretor: { $regex: /Chad/i },
                    titulo:  { $regex: "Wick" }
                    }
                },
                { $sort: { ano: -1 } }
                ],
                cursor: {}
            },
            verbosity: "executionStats"
        });
nReturned: 5,,
executionTimeMillis: 8
```

17. busca: encontrar os 100 filmes com as menores notas, pulando os primeiros 8000

```
comando:    db.runCommand({
            explain: {
                aggregate: "filmes",
                pipeline: [
                { $sort: { nota: 1 } },
                { $skip: 8000 },
                { $limit: 100 }
                ],
```

```
            cursor: {}
        },
        verbosity: "executionStats"
    });
nReturned: 100,
executionTimeMillis: 13,
```

18. busca: contar o número total de gêneros distintos

```
comando:    db.runCommand({
        explain: {
            aggregate: "filmes",
            pipeline: [
                { $group: { _id: "$genero" } },
                { $group: { _id: null, totalGeneros: { $sum: 1 } } }
            ],
            cursor: {}
        },
        verbosity: "executionStats"
    });
nReturned: 1,
executionTimeMillis: 6
```

19. busca: calcular a média de idade dos atores principais com idade maior ou igual a 18

```
comando:    db.runCommand({
        explain: {
            aggregate: "filmes",
            pipeline: [
                { $match: { "atorPrincipal.idade": { $gte: 18 } } },
                { $group: { _id: null, mediaIdade: { $avg: "$atorPrincipal.idade" } } }
            ],
            cursor: {}
        },
        verbosity: "executionStats"
    });
nReturned: 1,
executionTimeMillis: 21
```

20. busca: somar as notas de todos os filmes do gênero "Drama"

```
comando:    db.runCommand({
        explain: {
            aggregate: "filmes",
            pipeline: [
                { $match: { genero: "Drama" } },
                { $group: { _id: null, somaNotas: { $sum: "$nota" } } }
            ],
            cursor: {}
        },
        verbosity: "executionStats"
```

```
        });
  nReturned: 1,
  executionTimeMillis: 9
```

COMPARAÇÃO DOS TEMPOS DE EXECUÇÃO DAS 5 QUERIES MAIS LENTAS, SEM INDÍCE E COM ÍNDICE:

query: 19 tempo: 21 ms, tempoComIndice: 14 ms
O índice em atorPrincipal.idade permitiu usar IXSCAN no $match, filtrando menos documentos antes do $group e reduzindo a carga.

query: 15, tempo: 13 ms, tempoComIndice: 7 ms
O índice em genero foi escaneado em vez da coleção completa (COLLSCAN), acelerando a leitura mesmo sem filtro prévio.

query: 17 tempo: 13 ms, tempoComIndice: 6 ms
O índice em nota devolve os documentos já ordenados, eliminando o sort em memória e otimizando o skip/limit.

query: 4 tempo: 11 ms, tempoComIndice: 8 ms
O índice de texto em atorPrincipal.nome acelerou a localização dos termos, mas regex ainda faz verificação extra em cada documento.

query: 3 tempo: 9 ms, tempoComIndice: 1 ms
O índice multikey em atoresCoadjuvantes localiza direto os documentos com o valor, reduzindo drasticamente o escaneamento.

query: 20 tempo: 9 ms, tempoComIndice: 6 ms
O índice em gênero agilizou o filtro inicial, diminuindo o número de documentos agrupados e acelerando o cálculo do $sum.

Conclusão sobre os índices:
A análise das consultas executadas sem índices no MongoDB mostrou tempos de execução variando entre 3 e 21 ms, sendo as mais lentas aquelas que envolvem agregações complexas, ordenações ou filtragens em grandes volumes de dados. As cinco queries mais demoradas sem índice foram:

Query 19 (média de idade dos atores principais ≥ 18): 21 ms
Query 15 (contar filmes por gênero): 13 ms
Query 17 (100 filmes com menores notas, pulando 8000): 13 ms
Query 4 (filmes com "DiCaprio" como ator principal): 11 ms
Query 3 (filmes com Anthony Anderson como coadjuvante): 9 ms
(Query 20, soma de notas de filmes "Drama", também com 9 ms, foi analisada na comparação.)
Com a introdução de índices apropriados, os tempos de execução dessas queries foram significativamente reduzidos:

Query 19: de 21 ms para 14 ms (↓ 33,3%)
Índice em atorPrincipal.idade otimizou o $match e reduziu a carga no $group.
Query 15: de 13 ms para 7 ms (↓ 46,2%)
Índice em genero substituiu o COLLSCAN, acelerando a leitura.
Query 17: de 13 ms para 6 ms (↓ 53,8%)
Índice em nota eliminou a ordenação em memória e otimizou skip/limit.
Query 4: de 11 ms para 8 ms (↓ 27,3%)
Índice de texto em atorPrincipal.nome agilizou a busca, apesar da verificação extra por regex.
Query 3: de 9 ms para 1 ms (↓ 88,9%)
Índice multikey em atoresCoadjuvantes localizou diretamente os documentos, reduzindo drasticamente o escaneamento.
Query 20: de 9 ms para 6 ms (↓ 33,3%)
Índice em genero acelerou o filtro inicial e o cálculo do $sum.
Os resultados destacam que os índices são essenciais para melhorar o desempenho em MongoDB, especialmente em consultas com filtragem, ordenação e agregação. A maior redução ocorreu na Query 3 (88,9%), enquanto a menor foi na Query 4 (27,3%), devido à limitação do regex. Assim, a criação de índices bem planejados é uma estratégia fundamental para otimizar consultas, garantindo eficiência e escalabilidade em grandes conjuntos de dados.