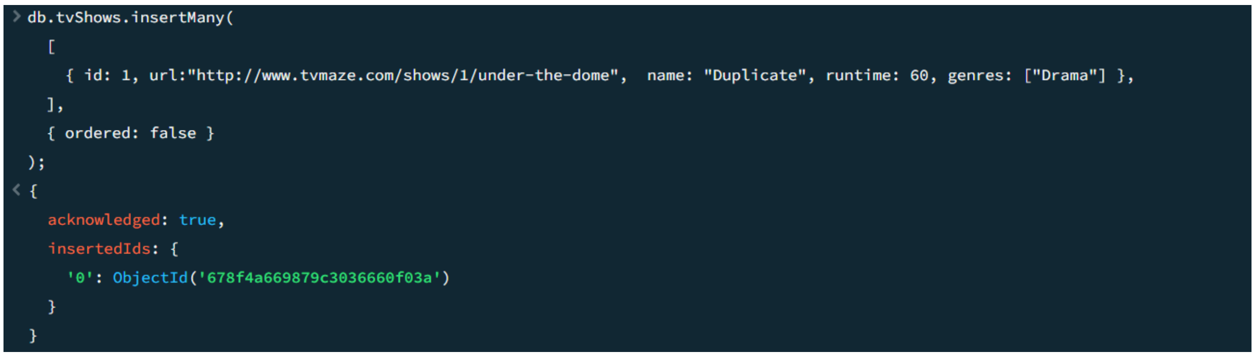
Assignment 3

Zekenov Meiirzhan SE-2319

Validation schema

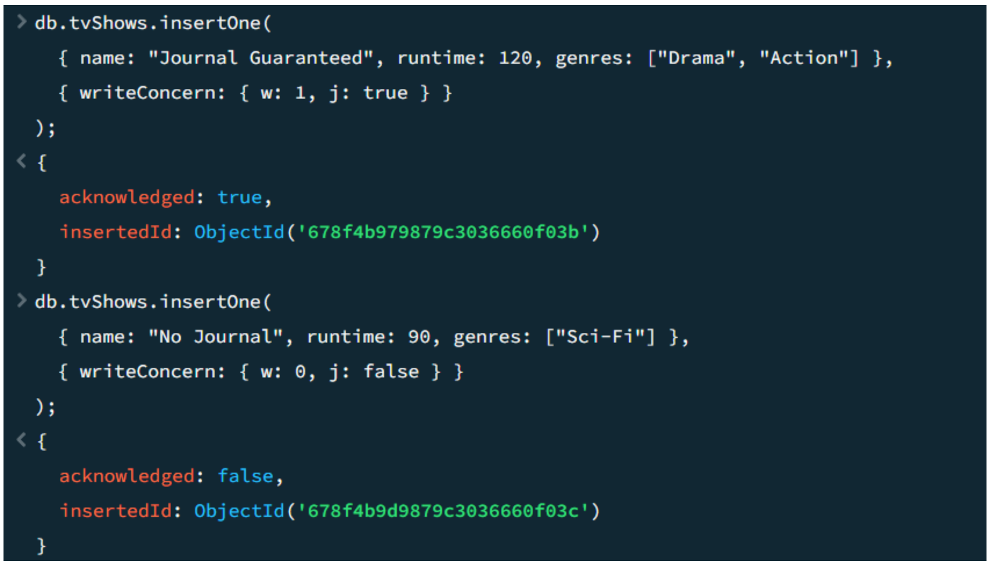
```
{
 $jsonSchema: {
   bsonType: 'object',
   required: [
     'id',
     'url',
     'name',
     'type',
     'language',
     'genres',
     'status',
     'runtime',
     'premiered',
     'officialSite',
     'schedule',
     'rating',
     'weight',
     'network',
     'webChannel',
     'externals',
     'image',
     'summary',
     'updated',
     '_links'
   ],
   properties: {
     id: {
       bsonType: 'int',
       minimum: 0
     },
     url: {
       bsonType: 'string',
       pattern: '^http://'
     },
     name: {
       bsonType: 'string'
     },
type: {
  bsonType: 'string'
},
language: {
  bsonType: 'string'
},
genres: {
  bsonType: 'array',
  items: {
    bsonType: 'string'
  }
},
status: {
  bsonType: 'string'
},
runtime: {
  bsonType: 'int',
  minimum: 0
},
premiered: {
  bsonType: 'string',
  pattern: '^[0-9]{4}-[0-9]{2}-[0-9]{2}$'
},
officialSite: {
  bsonType: [
    'string',
    'null'
  ],
  pattern: '^http://'
},
schedule: {
  bsonType: 'object',
  required: [
    'time',
    'days'
  ],
  properties: {
    time: {
      bsonType: [
        'string',
        'null'
      ],
      pattern: '^([01]\\d|2[0-3]):([0-5]\\d)$'
    },
    days: {
      bsonType: 'array',
      items: {
        bsonType: 'string'
      }
    }
  }
},
rating: {
  bsonType: 'object',
  properties: {
    average: {
      bsonType: [
        'int',
        'double',
        'null'
      ]
    }
  }
},
weight: {
  bsonType: 'int',
  minimum: 0
},
network: {
  bsonType: 'object',
  required: [
    'id',
    'name',
    'country'
  ],
  properties: {
    id: {
      bsonType: 'int',
      minimum: 0
    },
    name: {
      bsonType: 'string'
    },
    country: {
      bsonType: 'object',
      required: [
        'name',
        'code',
        'timezone'
      ],
      properties: {
        name: {
          bsonType: 'string'
        },
        code: {
          bsonType: 'string'
        },
        timezone: {
          bsonType: 'string'
        }
      }
    }
  }
},
webChannel: {
  bsonType: [
    'object',
    'null'
  ]
},
externals: {
  bsonType: 'object',
  required: [
    'tvrage',
    'thetvdb',
    'imdb'
  ],
  properties: {
    tvrage: {
      bsonType: [
        'int',
        'null'
      ],
      minimum: 0
    },
    thetvdb: {
      bsonType: [
        'int',
        'null'
      ],
      minimum: 0
    },
    imdb: {
      bsonType: 'string',
      pattern: '^tt[0-9]{7}$'
    }
  }
},
image: {
  bsonType: 'object',
  required: [
    'medium',
    'original'
  ],
  properties: {
    medium: {
      bsonType: 'string'
    },
    original: {
      bsonType: 'string'
    }
  }
},
summary: {
  bsonType: 'string'
},
updated: {
  bsonType: [
    'int',
    'date'
  ]
},
_links: {
  bsonType: 'object',
        required: [
          'self',
          'previousepisode'
        ],
        properties: {
          self: {
            bsonType: 'object',
            required: [
              'href'
            ],
            properties: {
              href: {
                bsonType: 'string'
              }
            }
          },
          previousepisode: {
            bsonType: 'object',
            required: [
              'href'
            ],
            properties: {
              href: {
                bsonType: 'string'
              }
            }
          }
        }
      }
    }
  }
}
```

Task 2



```
> db.tvShows.insertMany(
    [
      { id: 1, url:"http://www.tvmaze.com/shows/1/under-the-dome",  name: "Duplicate", runtime: 60, genres: ["Drama"] },
    ],
    { ordered: false }
  );
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('678f4a669879c3036660f03a')
    }
  }
```

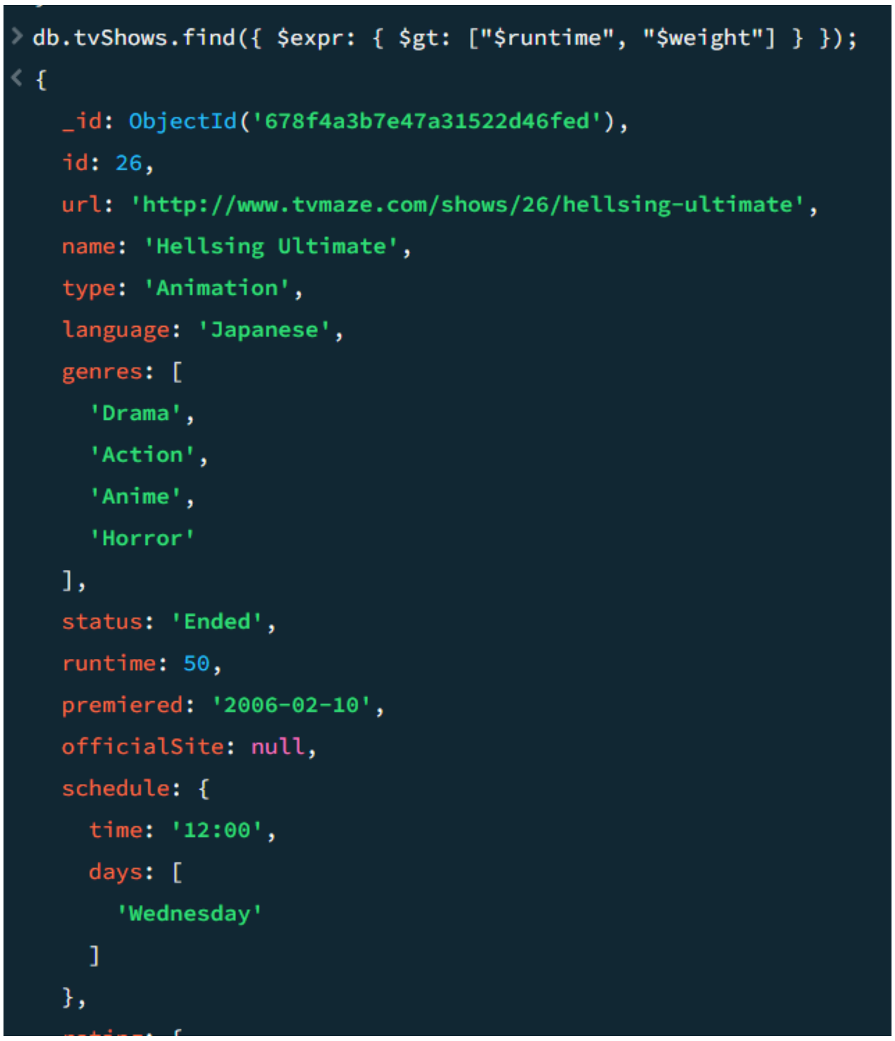
Task 3



```
> db.tvShows.insertOne(
    { name: "Journal Guaranteed", runtime: 120, genres: ["Drama", "Action"] },
    { writeConcern: { w: 1, j: true } }
  );
< {
    acknowledged: true,
    insertedId: ObjectId('678f4b979879c3036660f03b')
  }
> db.tvShows.insertOne(
    { name: "No Journal", runtime: 90, genres: ["Sci-Fi"] },
    { writeConcern: { w: 0, j: false } }
  );
< {
    acknowledged: false,
    insertedId: ObjectId('678f4b9d9879c3036660f03c')
  }
```

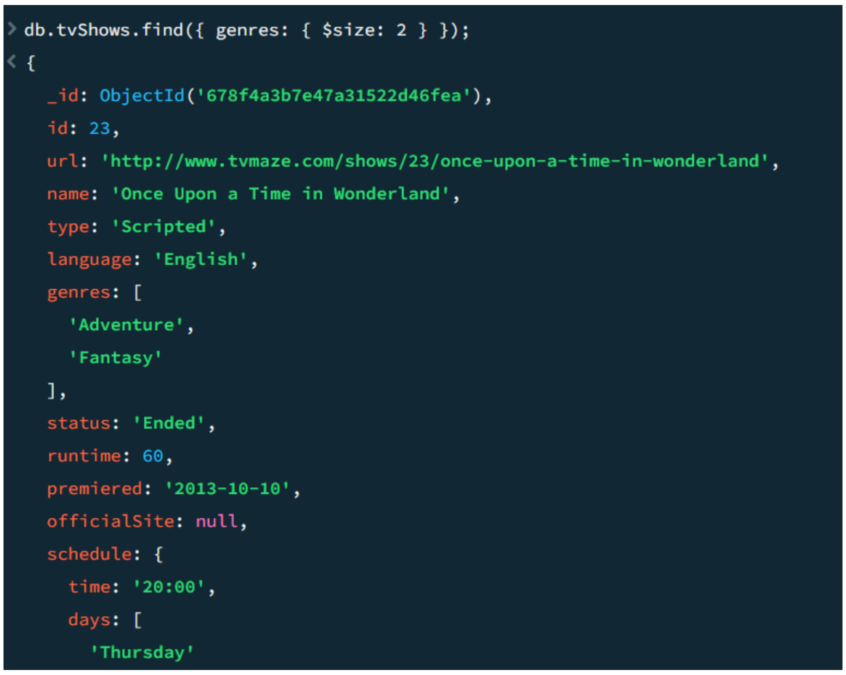
Journaling ensures data durability by writing to the journal before acknowledging the write. Without it, acknowledgment may occur before writing.

Task 4



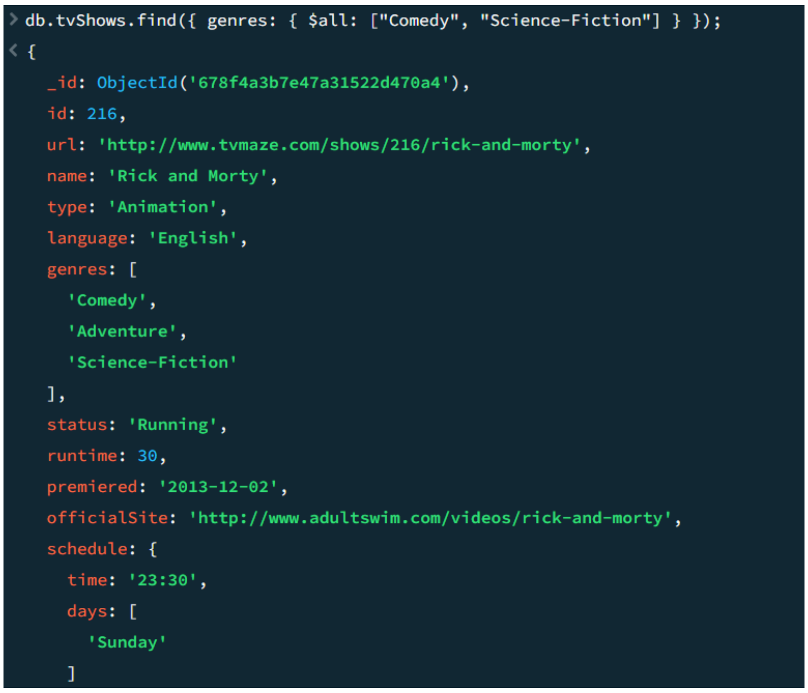
```
> db.tvShows.find({ $expr: { $gt: ["$runtime", "$weight"] } });
< {
    _id: ObjectId('678f4a3b7e47a31522d46fed'),
    id: 26,
    url: 'http://www.tvmaze.com/shows/26/hellsing-ultimate',
    name: 'Hellsing Ultimate',
    type: 'Animation',
    language: 'Japanese',
    genres: [
      'Drama',
      'Action',
      'Anime',
      'Horror'
    ],
    status: 'Ended',
    runtime: 50,
    premiered: '2006-02-10',
    officialSite: null,
    schedule: {
      time: '12:00',
      days: [
        'Wednesday'
      ]
    },
```

Task 5



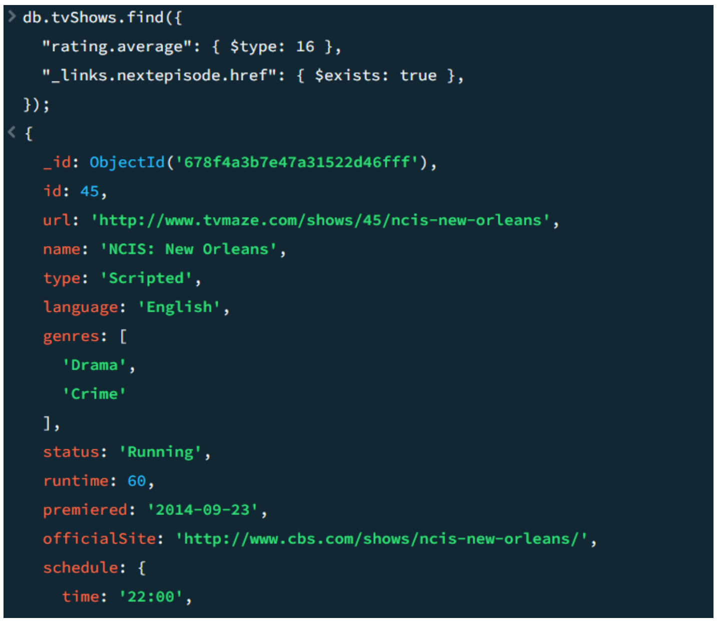
```
> db.tvShows.find({ genres: { $size: 2 } });
< {
    _id: ObjectId('678f4a3b7e47a31522d46fea'),
    id: 23,
    url: 'http://www.tvmaze.com/shows/23/once-upon-a-time-in-wonderland',
    name: 'Once Upon a Time in Wonderland',
    type: 'Scripted',
    language: 'English',
    genres: [
      'Adventure',
      'Fantasy'
    ],
    status: 'Ended',
    runtime: 60,
    premiered: '2013-10-10',
    officialSite: null,
    schedule: {
      time: '20:00',
      days: [
        'Thursday'
```

Task 6



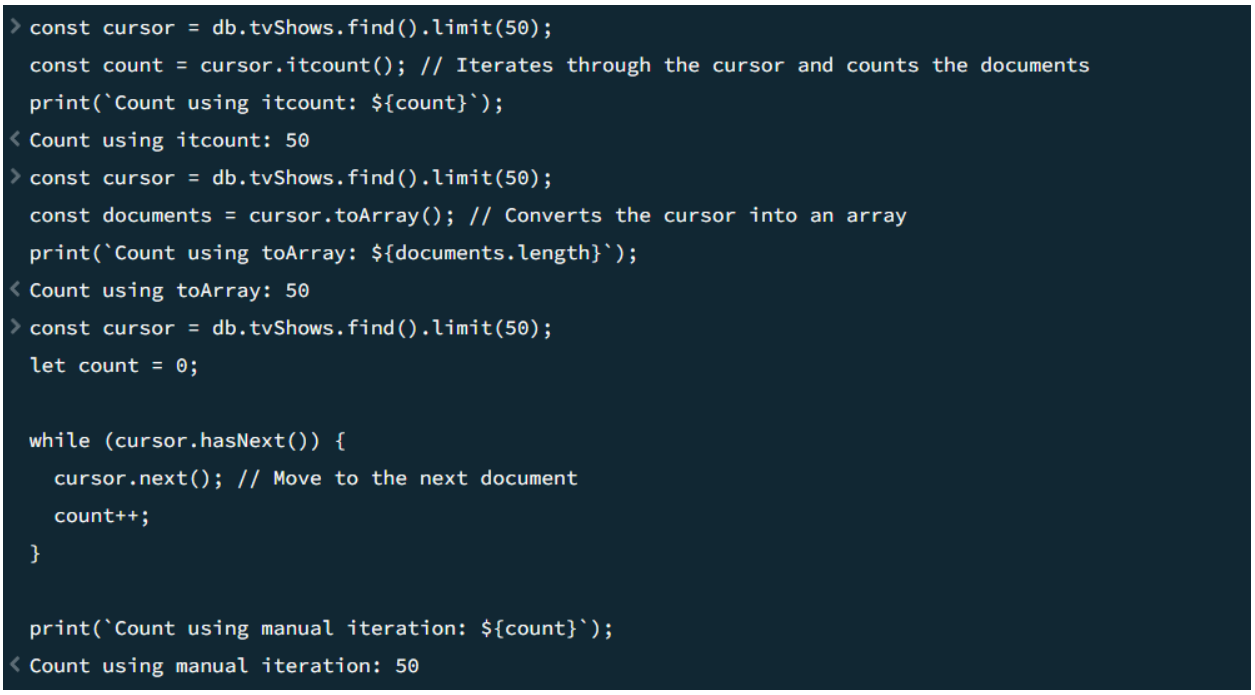
```
> db.tvShows.find({ genres: { $all: ["Comedy", "Science-Fiction"] } });
< {
    _id: ObjectId('678f4a3b7e47a31522d470a4'),
    id: 216,
    url: 'http://www.tvmaze.com/shows/216/rick-and-morty',
    name: 'Rick and Morty',
    type: 'Animation',
    language: 'English',
    genres: [
      'Comedy',
      'Adventure',
      'Science-Fiction'
    ],
    status: 'Running',
    runtime: 30,
    premiered: '2013-12-02',
    officialSite: 'http://www.adultswim.com/videos/rick-and-morty',
    schedule: {
      time: '23:30',
      days: [
        'Sunday'
      ]
```

Task 7



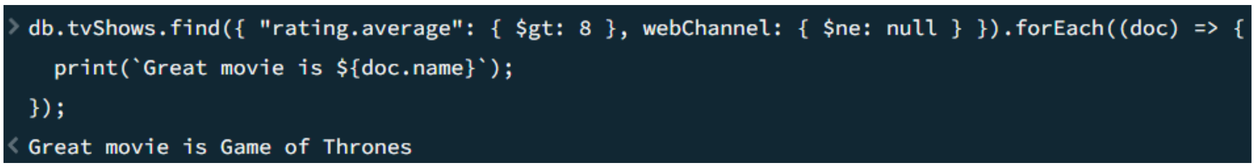
```
> db.tvShows.find({
    "rating.average": { $type: 16 },
    "_links.nextepisode.href": { $exists: true },
  });
< {
    _id: ObjectId('678f4a3b7e47a31522d46fff'),
    id: 45,
    url: 'http://www.tvmaze.com/shows/45/ncis-new-orleans',
    name: 'NCIS: New Orleans',
    type: 'Scripted',
    language: 'English',
    genres: [
      'Drama',
      'Crime'
    ],
    status: 'Running',
    runtime: 60,
    premiered: '2014-09-23',
    officialSite: 'http://www.cbs.com/shows/ncis-new-orleans/',
    schedule: {
      time: '22:00',
```

Task 8



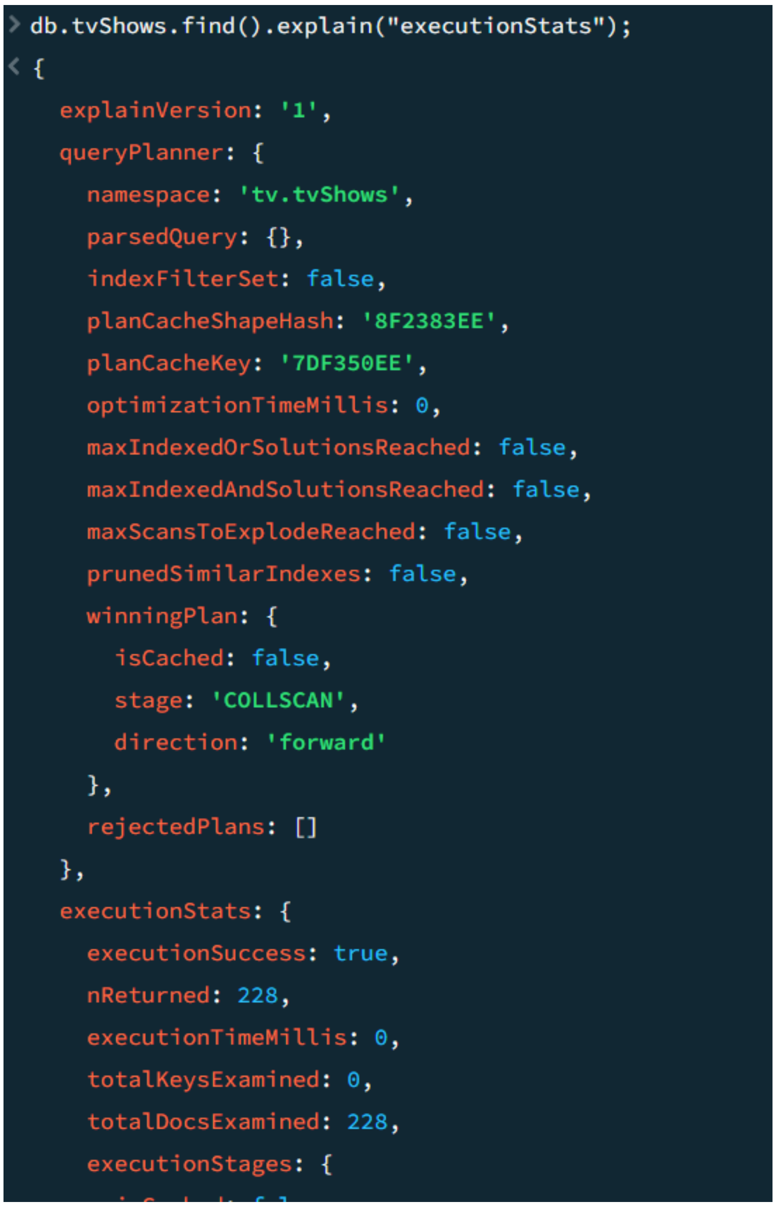
```
> const cursor = db.tvShows.find().limit(50);
  const count = cursor.itcount(); // Iterates through the cursor and counts the documents
  print(`Count using itcount: ${count}`);
< Count using itcount: 50
> const cursor = db.tvShows.find().limit(50);
  const documents = cursor.toArray(); // Converts the cursor into an array
  print(`Count using toArray: ${documents.length}`);
< Count using toArray: 50
> const cursor = db.tvShows.find().limit(50);
  let count = 0;

  while (cursor.hasNext()) {
    cursor.next(); // Move to the next document
    count++;
  }

  print(`Count using manual iteration: ${count}`);
< Count using manual iteration: 50
```

Task 9



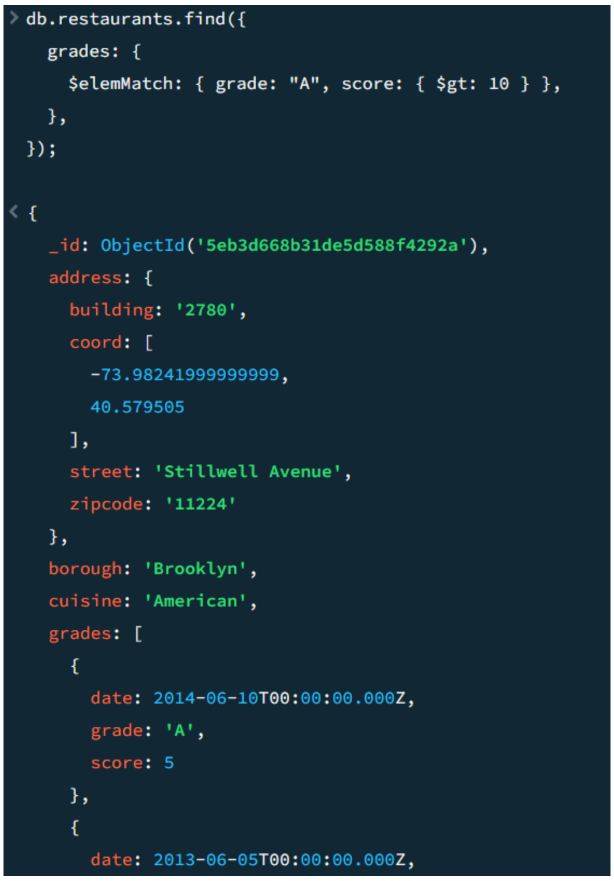
```
> db.tvShows.find({ "rating.average": { $gt: 8 }, webChannel: { $ne: null } }).forEach((doc) => {
    print(`Great movie is ${doc.name}`);
  });
< Great movie is Game of Thrones
```

Task 10

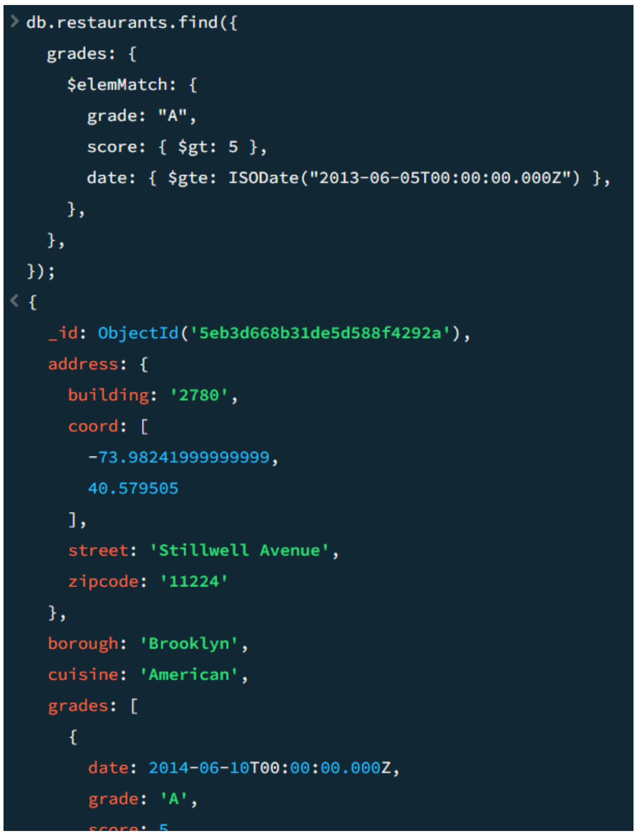


```
> db.tvShows.find().explain("executionStats");
< {
    explainVersion: '1',
    queryPlanner: {
      namespace: 'tv.tvShows',
      parsedQuery: {},
      indexFilterSet: false,
      planCacheShapeHash: '8F2383EE',
      planCacheKey: '7DF350EE',
      optimizationTimeMillis: 0,
      maxIndexedOrSolutionsReached: false,
      maxIndexedAndSolutionsReached: false,
      maxScansToExplodeReached: false,
      prunedSimilarIndexes: false,
      winningPlan: {
        isCached: false,
        stage: 'COLLSCAN',
        direction: 'forward'
      },
      rejectedPlans: []
    },
    executionStats: {
      executionSuccess: true,
      nReturned: 228,
      executionTimeMillis: 0,
      totalKeysExamined: 0,
      totalDocsExamined: 228,
      executionStages: {
```

Task 11

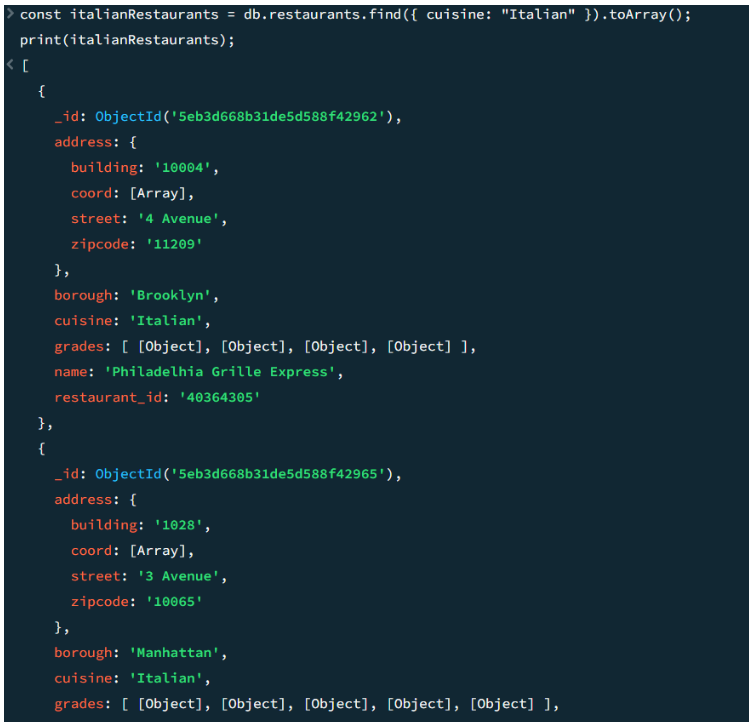


```
> db.restaurants.find({
    grades: {
      $elemMatch: { grade: "A", score: { $gt: 10 } },
    },
  });

< {
    _id: ObjectId('5eb3d668b31de5d588f4292a'),
    address: {
      building: '2780',
      coord: [
        -73.98241999999999,
        40.579505
      ],
      street: 'Stillwell Avenue',
      zipcode: '11224'
    },
    borough: 'Brooklyn',
    cuisine: 'American',
    grades: [
      {
        date: 2014-06-10T00:00:00.000Z,
        grade: 'A',
        score: 5
      },
      {
        date: 2013-06-05T00:00:00.000Z,
```

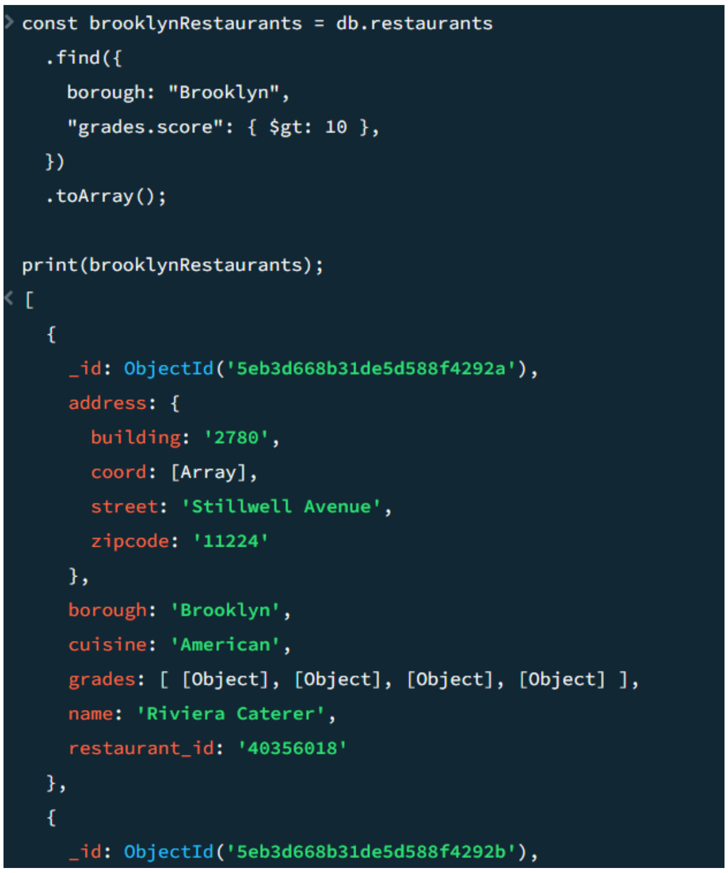


```
> db.restaurants.find({
    grades: {
      $elemMatch: {
        grade: "A",
        score: { $gt: 5 },
        date: { $gte: ISODate("2013-06-05T00:00:00.000Z") },
      },
    },
  });
< {
    _id: ObjectId('5eb3d668b31de5d588f4292a'),
    address: {
      building: '2780',
      coord: [
        -73.98241999999999,
        40.579505
      ],
      street: 'Stillwell Avenue',
      zipcode: '11224'
    },
    borough: 'Brooklyn',
    cuisine: 'American',
    grades: [
      {
        date: 2014-06-10T00:00:00.000Z,
        grade: 'A',
        score: 5
```

Task 12



```
> const italianRestaurants = db.restaurants.find({ cuisine: "Italian" }).toArray();
  print(italianRestaurants);
< [
    {
      _id: ObjectId('5eb3d668b31de5d588f42962'),
      address: {
        building: '10004',
        coord: [Array],
        street: '4 Avenue',
        zipcode: '11209'
      },
      borough: 'Brooklyn',
      cuisine: 'Italian',
      grades: [ [Object], [Object], [Object], [Object] ],
      name: 'Philadelhia Grille Express',
      restaurant_id: '40364305'
    },
    {
      _id: ObjectId('5eb3d668b31de5d588f42965'),
      address: {
        building: '1028',
        coord: [Array],
        street: '3 Avenue',
        zipcode: '10065'
      },
      borough: 'Manhattan',
      cuisine: 'Italian',
      grades: [ [Object], [Object], [Object], [Object], [Object] ],
```



```
> const brooklynRestaurants = db.restaurants
    .find({
      borough: "Brooklyn",
      "grades.score": { $gt: 10 },
    })
    .toArray();

  print(brooklynRestaurants);
< [
    {
      _id: ObjectId('5eb3d668b31de5d588f4292a'),
      address: {
        building: '2780',
        coord: [Array],
        street: 'Stillwell Avenue',
        zipcode: '11224'
      },
      borough: 'Brooklyn',
      cuisine: 'American',
      grades: [ [Object], [Object], [Object], [Object] ],
      name: 'Riviera Caterer',
      restaurant_id: '40356018'
    },
    {
      _id: ObjectId('5eb3d668b31de5d588f4292b'),
```