NOSQL _ MONGODB Assignment 2

Affan Mohammed N Marikar
281911

Utilise the Aggregation Framework to perform data manipulation and analysis within your game:

- Count the total number of locations in your game world.
  db.locations.aggregate({$count:"total_rows"})

```
adventure_game> db.locations.aggregate({$count:"total_rows"})
[ { total_rows: 4 } ]
adventure_game>
```

- Calculate the average number of exits per location.

```
db.locations.aggregate([
  {
    $project: { numExits: { $size: "$exits" }}
  },
  {
    $group: { _id: null,  Avg_Exits: { $avg: "$numExits" } }
  }
]);
```

```
adventure_game> db.locations.aggregate([
...     {
...         $project: {
...             numExits: { $size: "$exits" }
...         }
...     },
...     {
...         $group: {
...             _id: null,
...             Avg_Exits: { $avg: "$numExits" }
...         }
...     }
... ]);
[ { _id: null, Avg_Exits: 1.5 } ]
adventure_game>
```

- Identify the most prevalent item type using aggregation pipelines.

```
db.items.aggregate([
  {
    $group: { _id: "$type", Most_Prevalent: { $sum: 1 } }
  },
  {
    $sort: { Most_Prevalent: -1 }
  },
  {
    $limit: 1
  }
]);
```

```
adventure_game> db.items.aggregate([
...     {
...         $group: {
...             _id: "$type",
...             Most_Prevalent: { $sum: 1 }
...         }
...     },
...     {
...         $sort: { Most_Prevalent: -1 }
...     },
...     {
...         $limit: 1
...     }
... ]);
[ { _id: 'Tourism', Most_Prevalent: 2 } ]
adventure_game>
```

Implement indexing strategies to optimise query performance in your game:

- Identify frequently used query fields in your game (e.g., location names, item types). Create indexes on these fields within the relevant collections.
  db.items.createIndex({type:1})
  db.locations.createIndex({name:1})
  db.characters.createIndex({name:1})

```
adventure_game> db.characters.getIndexes()
[ { v: 2, key: { _id: 1 }, name: '_id_' } ]
adventure_game> db.characters.createIndex({name:1})
name_1eryHash: 'A2F868FD',
adventure_game> db.characters.getIndexes()
[   maxIndexedOrSolutionsReached: false,
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { name: 1 }, name: 'name_1' }
]   winningPlan: {
adventure_game> db.items.getIndexes()
[     inputStage: {
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { type: 1 }, name: 'type_1' }
]       indexName: 'name_1',
adventure_game> db.locations.getIndexes()
[       multiKeyPaths: { name: [] },
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { name: 1 }, name: 'name_1' }
]       isPartial: false,
adventure_game>
```

- Test the impact of indexes on query speed by comparing performance before and after indexing.
  db.locations.find().explain("executionStats")

  Before:

```
adventure_game> db.locations.find().explain("executionStats")
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'adventure_game.locations',
    indexFilterSet: false,
    parsedQuery: {},
    queryHash: '8880B5AF',
    planCacheKey: '8880B5AF',
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: { stage: 'COLLSCAN', direction: 'forward' },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 4,
    executionTimeMillis: 0,
    totalKeysExamined: 0,
    totalDocsExamined: 4,
    executionStages: {
      stage: 'COLLSCAN',
      nReturned: 4,
      executionTimeMillisEstimate: 0,
      works: 5,
      advanced: 4,
      needTime: 0,
      needYield: 0,
      saveState: 0,
      restoreState: 0,
      isEOF: 1,
      direction: 'forward',
      docsExamined: 4
    }
  },
  command: { find: 'locations', filter: {}, '$db': 'adventure_game' },
  serverInfo: {
    host: '6804e5f8eceb555',
    port: 27017,
    version: '7.0.11',
    gitVersion: 'f451220f0df2b9dfe073f1521837f8ec5c208a8c'
  },
  serverParameters: {
    internalQueryFacetBufferSizeBytes: 104857600,
    internalQueryFacetMaxOutputDocSizeBytes: 104857600,
    internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
    internalDocumentSourceGroupMaxMemoryBytes: 104857600,
    internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
    internalQueryProhibitBlockingMergeOnMongoS: 0,
    internalQueryMaxAddToSetBytes: 104857600,
    internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600,
    internalQueryFrameworkControl: 'trySbeRestricted'
  },
  ok: 1
}
```

After:

```
adventure_game> db.locations.find().explain("executionStats")
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'adventure_game.locations',
    indexFilterSet: false,
    parsedQuery: {},
    queryHash: '8880B5AF',
    planCacheKey: '8880B5AF',
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: { stage: 'COLLSCAN', direction: 'forward' },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 4,
    executionTimeMillis: 0,
    totalKeysExamined: 0,
    totalDocsExamined: 4,
    executionStages: {
      stage: 'COLLSCAN',
      nReturned: 4,
      executionTimeMillisEstimate: 0,
      works: 5,
      advanced: 4,
      needTime: 0,
      needYield: 0,
      saveState: 0,
      restoreState: 0,
      isEOF: 1,
      direction: 'forward',
      docsExamined: 4
    }
  },
  command: { find: 'locations', filter: {}, '$db': 'adventure_game' },
  serverInfo: {
    host: '6804e5f8eceb555',
    port: 27017,
    version: '7.0.11',
    gitVersion: 'f451220f0df2b9dfe073f1521837f8ec5c208a8c'
  },
  serverParameters: {
    internalQueryFacetBufferSizeBytes: 104857600,
    internalQueryFacetMaxOutputDocSizeBytes: 104857600,
    internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
    internalDocumentSourceGroupMaxMemoryBytes: 104857600,
    internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
    internalQueryProhibitBlockingMergeOnMongoS: 0,
    internalQueryMaxAddToSetBytes: 104857600,
    internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600,
    internalQueryFrameworkControl: 'trySbeRestricted'
  },
  ok: 1
}
```