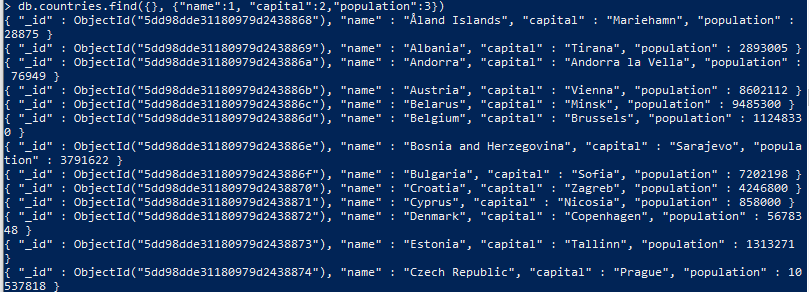
IST769 MongoDB and Redis

Your Name: Tajudeen Abdulazeez   
Your SUID: 69687-7373-0   
Your Email: toabdula@syr.edu   
Date Due:   
Homework #: 8

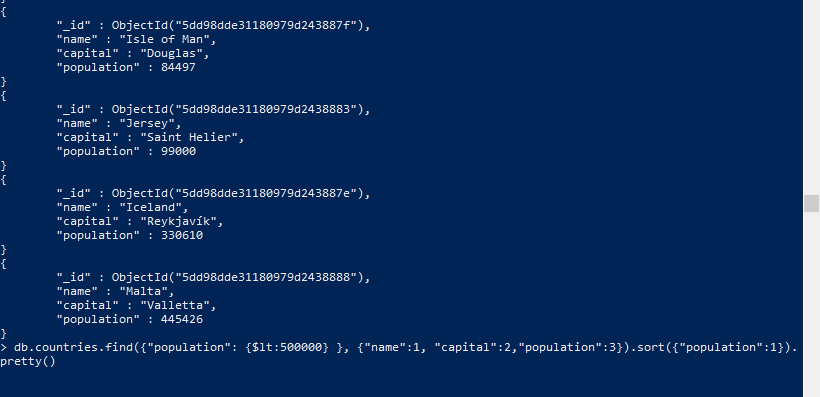
* Write a MongoDB Query to retrieve Country name, population, and capital for all countries in the collection.

db.countries.find({}, {"name":1, "capital":2,"population":3})



* Write a MongoDB Query to retrieve Country name, population, and capital for all countries with a population under 500,000 sorted by population.

db.countries.find({"population": {$lt:500000} }, {"name":1, "capital":2,"population":3}).sort({"population":1}).pretty()



* Use the**. explain(“executionStats”)** method to analyze the query you wrote in the previous step. Write an index to improve the performance of the query, then perform another explain to demonstrate it worked. Include the code of the index you wrote, the and the relevant output of the execution stats which demonstrate the index is being used.

db.countries.find({"population": {$lt:500000} }, {"name":1, "capital":2,"population":3}).sort({"population":1}).explain("executionStats")

{ "queryPlanner" : { "plannerVersion" : 1, "namespace" : "demo.countries", "indexFilterSet" : false, "parsedQuery" : { "population" : { "$lt" : 500000 } }, "winningPlan" : { "stage" : "PROJECTION", "transformBy" : { "name" : 1, "capital" : 2, "population" : 3 }, "inputStage" : { "stage" : "SORT", "sortPattern" : { "population" : 1 }, "inputStage" : { "stage" : "SORT\_KEY\_GENERATOR", "inputStage" : { "stage" : "COLLSCAN", "filter" : { "population" : { "$lt" : 500000 } }, "direction" : "forward" } } } }, "rejectedPlans" : [ ] }, "executionStats" : { "executionSuccess" : true, "nReturned" : 14, "executionTimeMillis" : 0, "totalKeysExamined" : 0, "totalDocsExamined" : 53, "executionStages" : { "stage" : "PROJECTION", "nReturned" : 14, "executionTimeMillisEstimate" : 0, "works" : 71, "advanced" : 14, "needTime" : 56, "needYield" : 0, "saveState" : 0, "restoreState" : 0, "isEOF" : 1, "invalidates" : 0, "transformBy" : { "name" : 1, "capital" : 2, "population" : 3 }, "inputStage" : { "stage" : "SORT", "nReturned" : 14, "executionTimeMillisEstimate" : 0, "works" : 71, "advanced" : 14, "needTime" : 56, "needYield" : 0, "saveState" : 0, "restoreState" : 0, "isEOF" : 1, "invalidates" : 0, "sortPattern" : { "population" : 1 }, "memUsage" : 9775, "memLimit" : 33554432, "inputStage" : { "stage" : "SORT\_KEY\_GENERATOR", "nReturned" : 14, "executionTimeMillisEstimate" : 0, "works" : 56, "advanced" : 14, "needTime" : 41, "needYield" : 0, "saveState" : 0, "restoreState" : 0, "isEOF" : 1, "invalidates" : 0, "inputStage" : { "stage" : "COLLSCAN", "filter" : { "population" : { "$lt" : 500000 } }, "nReturned" : 14, "executionTimeMillisEstimate" : 0, "works" : 55, "advanced" : 14, "needTime" : 40, "needYield" : 0, "saveState" : 0, "restoreState" : 0, "isEOF" : 1, "invalidates" : 0, "direction" : "forward", "docsExamined" : 53 } } } } }, "serverInfo" : { "host" : "mongo", "port" : 27017, "version" : "4.0.0", "gitVersion" : "3b07af3d4f471ae89e8186d33bbb1d5259597d51" }, "ok" : 1 }

* Select the most appropriate Redis data structure to store the following information:

|  |  |  |  |
| --- | --- | --- | --- |
| Product ID | Name | Qty On Hand | Unit Price |
| 1 | Apple | 7 | 2.49 |
| 2 | Banana | 12 | 1.99 |
| 3 | Cherry | 9 | 4.99 |

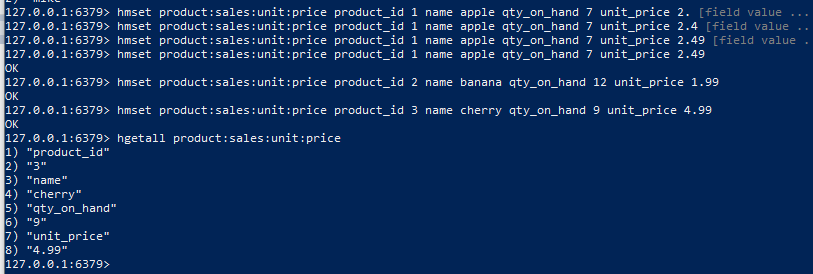
Execute the commands to store this information in Redis. Make sure to namespace your key and each of the fields should be retrievable under the key used.

hmset product:sales:unit:price product\_id 1 name apple qty\_on\_hand 7 unit\_price 2.49

hmset product:sales:unit:price product\_id 2 name banana qty\_on\_hand 12 unit\_price 1.99

hmset product:sales:unit:price product\_id 3 name cherry qty\_on\_hand 9 unit\_price 4.99

hgetall product:sales:unit:price



* Select the most appropriate Redis data structure to store the following information:

The 2018 Golden Snowball Competition for the Upstate NY City with the Highest Snowfall. Scores updated hourly.

|  |  |  |  |
| --- | --- | --- | --- |
| City | Syracuse | Rochester | Buffalo |
| Snowfall Inches | 97 | 68 | 84 |

Execute the commands to store this information in Redis. Make sure to namespace your key and each of the snowfall values should be updatable. For example, you should be able to add 10 inches to Buffalo to make it 94. You should be able to display the information upon request.

zadd golden:snowball:competition 97 syracuse 68 rochester 84 buffallo

zrange golden:snowball:competition 0 -1 WITHSCORES

zadd golden:snowball:competition incr 94 buffallo

zrange golden:snowball:competition 0 -1 WITHSCORES

