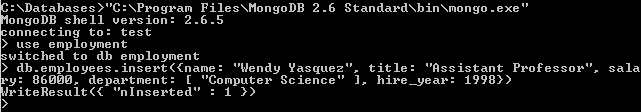
1. Create a new MongoDB database called employment.

|  |
| --- |
| use employment |

2. Insert a new record for Wendy Yasquez into the database and into a collection called employees.

|  |
| --- |
| db.employees.insert({name: "Wendy Yasquez", title: "Assistant Professor", salary: 86000, department: [ "Computer Science" ], hire\_year: 1998}) |

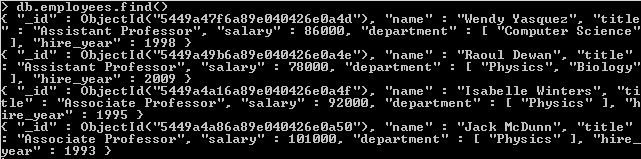


3. Write a JavaScript function to insert new professors into the employees collection.

|  |
| --- |
| function insertProfessor(name, title, salary, departments, hireyear) {  return db.employees.insert({name: name, title: title, salary: salary, department: departments, hire\_year: hireyear});  } |

4. Use this function to insert the records for Raoul Dewan, Isabelle Winters, and Jack McDunn.

|  |
| --- |
| insertProfessor("Raoul Dewan", "Assistant Professor", 78000, ["Physics", "Biology"], 2009)  insertProfessor("Isabelle Winters", "Associate Professor", 92000, ["Physics"], 1995)  insertProfessor("Jack McDunn", "Associate Professor", 101000, ["Physics"], 1993) |

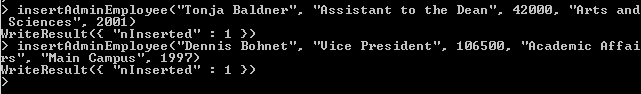


5. Write a JavaScript function to insert new administrative employees into the employees collection.

|  |
| --- |
| function insertAdminEmployee(name, title, salary, division, location, hireyear) {  return db.employees.insert({name: name, title: title, salary: salary, division: division, location: location, hire\_year: hireyear});  } |

6. Use this function to insert the records for Tonja Baldner and Dennis Bohnet.

|  |
| --- |
| insertAdminEmployee("Tonja Baldner", "Assistant to the Dean", 42000, "Arts and Sciences", null, 2001)  insertAdminEmployee("Dennis Bohnet", "Vice President", 106500, "Academic Affairs", "Main Campus", 1997) |

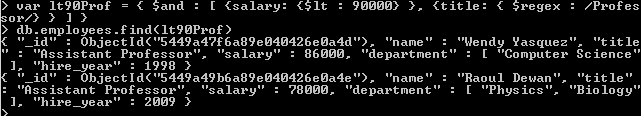


7. Show the code that will return all employees with salaries less than $90,000.

|  |
| --- |
| var lt90 = {salary: {$lt:90000}}  db.employees.find(lt90) |

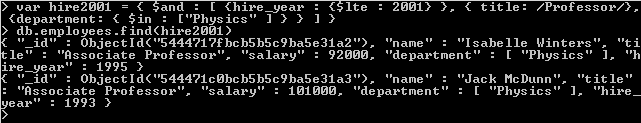
8. Show the code that will return all professors with salaries less than $90,000.

|  |
| --- |
| var lt90Prof = { $and : [ {salary: {$lt : 90000} }, {title: { $regex : /Professor/} } ] }  db.employees.find(lt90Prof) |



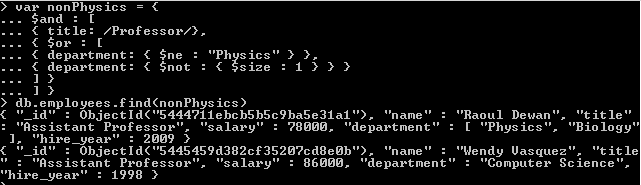
9. Show the code that will return all Physics professors hired before 2001.

|  |
| --- |
| var hire2001 = { $and : [ {hire\_year : {$lte : 2001} }, { title: /Professor/}, {department: { $in : [“Physics” ] } } ] }  db.employees.find(hire2001) |



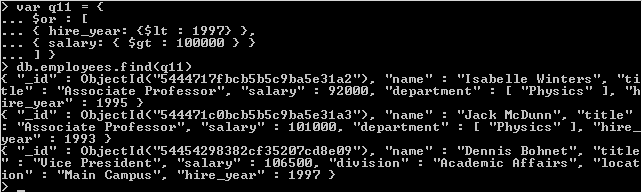
10. Show the code that will return all professors who teach for departments other than Physics. (This should include professors who teach for Physics and also other departments.)

|  |
| --- |
| var nonPhysics = {  $and : [  { title: /Professor/},  { $or : [  { department: { $ne : "Physics" } },  { department: { $not : { $size : 1 } } }  ] }  ] }  db.employees.find(nonPhysics) |



11. Show the code that will return all employees who were either hired before 1997 or who have salaries greater than $100,000.

|  |
| --- |
| var q11 = {  $or : [  { hire\_year: {$lt : 1997} },  { salary: { $gt : 100000 } }  ] }  db.employees.find(q11) |

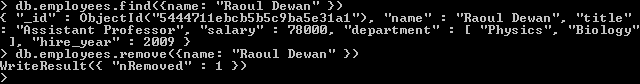


12. Suppose Tonja Baldner has been given a 10% raise. Show the code that will update her salary correctly.

|  |
| --- |
| var salinc10 = { $mul : { salary : 1.10 } }  db.employees.update({ \_id: ObjectId ("54454404382cf35207cd8e0a") }, salinc10) |

13. Professor Dewan has been offered a job at another university. Show the code that would remove his record from the database.

|  |
| --- |
| db.employees.remove({name: "Raoul Dewan" }) |



14. Instead of removing Professor Dewan’s record, we might prefer to create a new collection called pastemployees and move his record there. Show the code that will move his record to the new collection and add a departyear value of 2014 to his record. (You can do it in two steps).

|  |
| --- |
| var dewan = db.employees.findAndModify({query: { name: "Raoul Dewan"}, remove: true } )  db.pastemployees.update(dewan, {$set: {departyear: 2014}}, {upsert: true}) |

