This is a final exam for ADB XXXX. The final will cover MongoDB and will be worth 35% of your total grade. Each task is worth 10 points. Partial credit will be awarded.

Please read the entire questions. If you are asked for multiple parts in an answer (Statement & Results), each part is worth 50% of the total question value.

In the examples below, text in ALL CAPS needs to be replaced with a value.

IE: Replace MANUFACTURER NAME with Ford or Honda in the statement: \_id: "manufacturer:MANUFACTURER NAME"

Text is ALL CAPITALS needs to be replaced with some data of your choice.

Start Exam

0. Start Mongo and create a new database called "adbfp" with a collection called ‘vehicle’.

1. Create documents for car models.

3 car models for 1 manufacturer

2 car models for 1 manufacturer

1 car model for 1 manufacturer

Six car models total. Save the data to the database using a **single statement** for each document. Use the format below. Ex: Manufacturers=Ford, Model=Mustang

Features should be an array of features (strings). Each array should be different and contain 1-3 features.

Ex: ["powerWindows","AC","MP3","sunRoof","moonRoof","automaticTransmission","leather","4wd"]

Paste the 6 statements below.

\_id: MODEL NAME

name: MODEL NAME

manufacturer: MANUFACTURER NAME

type: "auto"

features []

|  |
| --- |
| **db.vehicle.save({name:"918 Spyder",manufacturer:"Porsche",type:"auto",features:["4wd","powerWindows","sunRoof"]})**  **db.vehicle.save({name:"911",manufacturer:"Porsche",type:"auto",features:["AC","moonRoof","leather"]})**  **db.vehicle.save({name:"Cayman",manufacturer:"Porsche",type:"auto",features:["AC","MP3","leather"]})**  **db.vehicle.save({name:"Aventador",manufacturer:"Lamborghini",type:"auto",features:["sunRoof","4wd","AC"]})**  **db.vehicle.save({name:"Huracan",manufacturer:"Lamborghini",type:"auto",features:["MP3","automaticTransmission","leather"]})**  **db.vehicle.save({name:"Agera",manufacturer:"Koenigsegg",type:"auto",features:["MP3","sunRoof","leather"]})** |

2. Create documents for motorcycle models. (If you don’t of know any use Google)

3 motorcycle models for 1 manufacturer

2 motorcycle models for 1 manufacturer

1 motorcycle model for 1 manufacturer

Six motorcycle models total. Create an **object first** in the format below, then **save the object** to the database.(2 steps).

Paste all 12 statements below.

\_id: MODEL NAME

name: MODEL NAME

manufacturer: MANUFACTURER NAME

type: "motorcycle"

|  |
| --- |
| **db.vehicle.save({name:"100 Sport",manufacturer:"Ducati",type:"motorcycle"})**  **db.vehicle.save({name:"821 Dark",manufacturer:"Ducati",type:"motorcycle"})**  **db.vehicle.save({name:"Diavel Carbon",manufacturer:"Ducati",type:"motorcycle"})**  **db.vehicle.save({name:"Ninja H2R",manufacturer:"Kawasaki",type:"motorcycle"})**  **db.vehicle.save({name:"Z800 ABS",manufacturer:"Kawasaki",type:"motorcycle"})**  **db.vehicle.save({name:"Iron 883",manufacturer:"Harley-Davidson",type:"motorcycle"})** |

3. Create an index on the name field. Then run the command to list all of your indexes. Paste both commands and results.

|  |
| --- |
| **db.vehicle.ensureIndex({name:1})**  {  "createdCollectionAutomatically" : false,  "numIndexesBefore" : 1,  "numIndexesAfter" : 2,  "ok" : 1  }  **db.vehicle.getIndexKeys()**  [ { "\_id" : 1 }, { "name" : 1 } ] |

4. Write a command to return one auto document (type=auto) by querying the name field (name = ???). Do not use findOne. Only one document should be returned. Paste command and results.

|  |
| --- |
| **db.vehicle.find({type:"auto"},{name:"918 Spyder"}).limit(1)**  { "\_id" : ObjectId("564e86fc9d8217dc046573c6"), "name" : "918 Spyder" } |

5. Write a command to return the motorcycles of 2 different manufacturers.

IE: Where type = motorcycle and (manufacturer = MANUFACTURER1 or manufacturer = MANUFACTURER2).

Paste command and results.

|  |
| --- |
| **db.vehicle.find({type:"motorcycle",$or:[{manufacturer:"Ducati"},{manufacturer:"Kawasaki"}]})**  { "\_id" : ObjectId("564e88c89d8217dc046573cc"), "name" : "100 Sport", "manufacturer" : "Ducati", "type" : "motorcycle" }  { "\_id" : ObjectId("564e89ec9d8217dc046573cd"), "name" : "821 Dark", "manufacturer" : "Ducati", "type" : "motorcycle" }  { "\_id" : ObjectId("564e89ec9d8217dc046573ce"), "name" : "Diavel Carbon", "manufacturer" : "Ducati", "type" : "motorcycle" }  { "\_id" : ObjectId("564e89ec9d8217dc046573cf"), "name" : "Ninja H2R", "manufacturer" : "Kawasaki", "type" : "motorcycle" }  { "\_id" : ObjectId("564e89ec9d8217dc046573d0"), "name" : "Z800 ABS", "manufacturer" : "Kawasaki", "type" : "motorcycle" } |

6. Add a new field called "mpg" to the 8 documents (6 autos & 2 motorcycles) using the update command with $set. These values should not repeat values and must be a numeric. Paste ALL commands below.

mpg: Number between 10-80 (This is the miles per gallon rating.)

|  |
| --- |
| **db.vehicle.update({name:"918 Spyder"},{$set:{"mpg":30}})**  **db.vehicle.update({name:"Agera"},{$set:{"mpg":25}})**  **db.vehicle.update({name:"Huracan"},{$set:{"mpg":28}})**  **db.vehicle.update({name:"Cayman"},{$set:{"mpg":32}})**  **db.vehicle.update({name:"Aventador"},{$set:{"mpg":23}})**  **db.vehicle.update({name:"911"},{$set:{"mpg":35}})**  **db.vehicle.update({name:"100 Sport"},{$set:{"mpg":35}})**  **db.vehicle.update({name:"Ninja H2R"},{$set:{"mpg":40}})** |

7. Write a command to push "hoverCapable" to the features array on one of the auto documents. Paste the command.

|  |
| --- |
| **db.vehicle.update({name:"918 Spyder"},{$push:{"features":"hoverCapable"}})** |

8. Write a command to return the auto with the third highest MPG. Only one document should be returned.

Paste the command and results.

|  |
| --- |
| **db.vehicle.find({type:"auto"}).sort({mpg:-1}).skip(2).limit(1)**  { "\_id" : ObjectId("564e86fc9d8217dc046573c6"), "name" : "918 Spyder", "manufacturer" : "Porsche", "type" : "auto", "features" : [ "4wd", "powerWindows", "sunRoof", "hoverCapable" ], "mpg" : 30 } |

9. Write a command to return the average "mpg" for all autos (type=auto) in the database. Only include documents with an "mpg" field. IE: if there is not an "mpg" field, do not count it as a 0. Paste the command and results.

|  |
| --- |
| **db.vehicle.group({**  **cond: {mpg:{$exists:true}, type:"auto"},**  **initial: {totalMPG:0,count:0},**  **reduce: function(obj,prev){**  **prev.totalMPG += obj.mpg;**  **prev.count++;},**  **finalize: function(out){out.avgMPG = out.totalMPG / out.count;}})**  [ { "totalMPG" : 173, "count" : 6, "avgMPG" : 28.833333333333332 } ] |

10. Write a command to return the count of all documents in your database.

Paste the command and results below.

|  |
| --- |
| **db.vehicle.count()**  12 |

11. Write a command to remove all motorcycles from your database.

Paste the command and results below.

|  |
| --- |
| **db.vehicle.remove({type:"motorcycle"})**  WriteResult({ "nRemoved" : 6 }) |

12. Backup your database. Zip the backup directory.

FINISHING YOUR EXAM

1. Save your files (This word file and zip of your export)