

  
  
Écrivez une requête MongoDB pour trouver tous les documents dans la collection "employees".



Écrivez une requête pour trouver tous les documents où l'âge est supérieur à 33.



Écrivez une requête pour trier les documents dans la collection "employees" par salaire décroissant.



Écrivez une requête pour sélectionner uniquement le nom et le job de chaque document.



Écrivez une requête pour compter le nombre d'employés par poste.

Écrivez une requête pour mettre à jour le salaire de tous les développeurs à 80000.



Exercice 1

Affichez l’identifiant et le nom des salles qui sont des SMAC.

db.salles.find({ smac: true },{ \_id: 1, nom: 1 })

Exercice 2

Affichez le nom des salles qui possèdent une capacité d’accueil strictement supérieure à 1000 places.

db.salles.find({ capacite: { $gt: 1000 } },{ \_id: 0, nom: 1 })

Exercice 3

Affichez l’identifiant des salles pour lesquelles le champ adresse ne comporte pas de numéro.

db.salles.find({ "adresse.numero": { $exists: false } },{ \_id: 1 })

Exercice 4

Affichez l’identifiant puis le nom des salles qui ont exactement un avis.

db.salles.find({ avis: { $size: 1 } },{ \_id: 1, nom: 1 })

Exercice 5

Affichez tous les styles musicaux des salles qui programment notamment du blues.

db.salles.find({ styles: "blues" },{ \_id: 0, styles: 1 })

Exercice 6

Affichez tous les styles musicaux des salles qui ont le style « blues » en première position dans leur tableau styles.

db.salles.find({ "styles.0": "blues" },{ \_id: 0, styles: 1 })

Exercice 7

Affichez la ville des salles dont le code postal commence par 84 et qui ont une capacité strictement inférieure à 500 places (pensez à utiliser une expression régulière).

db.salles.find({ "adresse.codePostal": /^84/, capacite: { $lt: 500 } },{ \_id: 0, "adresse.ville": 1 })

Exercice 8

Affichez l’identifiant pour les salles dont l’identifiant est pair ou le champ avis est absent.

db.salles.find({ $or: [ { \_id: { $mod: [ 2, 0 ] } }, { avis: { $exists: false } } ] },{ \_id: 1 })

Exercice 9

Affichez le nom des salles dont au moins un des avis comporte une note comprise entre 8 et 10 (tous deux inclus).

db.salles.find({ avis: { $elemMatch: { note: { $gte: 8, $lte: 10 } } } },{ \_id: 0, nom: 1 })

Exercice 10

Affichez le nom des salles dont au moins un des avis comporte une date postérieure au 15/11/2019 (pensez à utiliser le type JavaScript Date).

db.salles.find({ "avis.date": { $gt: new Date('2019-11-15') } },{ \_id: 0, nom: 1 })

Exercice 11

Affichez le nom ainsi que la capacité des salles dont le produit de la valeur de l’identifiant par 100 est strictement supérieur à la capacité.

db.salles.find({ $expr: { $gt: [{ $multiply: ["$\_id", 100] }, "$capacite"] } },{ \_id: 0, nom: 1, capacite: 1 })

Exercice 12

Affichez le nom des salles de type SMAC programmant plus de deux styles de musiques différents en utilisant l’opérateur $where qui permet de faire usage de JavaScript.

Exercice 13

Affichez les différents codes postaux présents dans les documents de la collection salles.

db.salles.distinct("adresse.codePostal")

Exercice 14

Mettez à jour tous les documents de la collection salles en rajoutant 100 personnes à leur capacité actuelle.

db.salles.updateMany({}, { $inc: { capacite: 100 } })

Exercice 15

Ajoutez le style « jazz » à toutes les salles qui n’en programment pas.

db.salles.updateMany({ styles: { $exists: false } },{ $set: { styles: ["jazz"] } })

Exercice 16

Retirez le style «funk» à toutes les salles dont l’identifiant n’est égal ni à 2, ni à 3.

db.salles.updateMany({ \_id: { $nin: [2, 3] } },{ $pull: { styles: "funk" } })

Exercice 17

Ajoutez un tableau composé des styles «techno» et « reggae » à la salle dont l’identifiant est 3.

db.salles.updateOne({ \_id: 3 },{ $push: { styles: { $each: ["techno", "reggae"] } } })

Exercice 18

Pour les salles dont le nom commence par la lettre P (majuscule ou minuscule), augmentez la capacité de 150 places et rajoutez un champ de type tableau nommé contact dans lequel se trouvera un document comportant un champ nommé telephone dont la valeur sera « 04 11 94 00 10 ».

db.salles.updateMany({ nom: { $regex: /^P/i } },{ $inc: { capacite: 150 },$set: { contact: {telephone: "04 11 94 00 10"}}})

Exercice 19

Pour les salles dont le nom commence par une voyelle (peu importe la casse, là aussi), rajoutez dans le tableau avis un document composé du champ date valant la date courante et du champ note valant 10 (double ou entier). L’expression régulière pour chercher une chaîne de caractères débutant par une voyelle suivie de n’importe quoi d’autre est [^aeiou]+$.

db.salles.updateMany({ nom: { $regex: /^[aeiou]/i } },{ $push: { avis: { $each: [{ date: new Date(), note: 10 }]}}})

Exercice 20

En mode upsert, vous mettrez à jour tous les documents dont le nom commence par un z ou un Z en leur affectant comme nom « Pub Z », comme valeur du champ capacite 50 personnes (type entier et non décimal) et en positionnant le champ booléen smac à la valeur « false ».

db.salles.updateMany({ nom: { $regex: /^[zZ]/ } },{ $set: {nom: "Pub Z", capacite: 50, smac: false}},{ upsert: true })  
  
  
  
  
  
  
Agrégation :

exo 1 :

db.achats.aggregate([{$addFields: {totalAchats: { $sum: "$achats" } ,totalReductions: { $sum: "$reductions" }}},{$addFields: { totalPayed: {$round: { $subtract: ["$totalAchats", "$totalReductions"]}}}}])

exo 2 :  
  
db.ventes.aggregate([{$lookup:{"from":"artistes","localField":"artiste","foreignField":"nom","as":"detail\_artiste"}}])

Agregate :  
  
exo 1 :  
  
var pipeline = [{$addFields :{ville :{$gt: ["$capacite", 50]}}}, {$addFields: {grande: {$gt: ["$capacite", 1000]}}}, {$match: {ville: true}}, {$project:{"adresse.ville":1, "grande": 1, "\_id":0}}]

Exo2 :

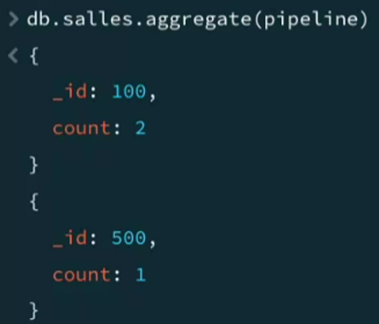
{$addFields:{apres\_extension:{$sum: ["$capacite", 100]}}}

Exo 3:  
  
var pipeline = [{$addFields:{departement:{$substrBytes:["$adresse.codePostal", 0, 2]}}}, { $group : { \_id: "$departement", capaciteT: {$sum : "$capacite"} } }, {$project:{"\_id":0, departements: "$\_id", capaciteT :1}}]

Exo 4 :  
  


Exo 5:

Exo 5 :  
  

  
  
  
  
geo loc :  
  
Exo 1:

var KilToRad = function(kilometres) {var rayonTerrestre = 6371; return kilometres / rayonTerrestre;}

var req = {"adresse.localisation": {$geoWithin: {$centerSphere: [salle.adresse.localisation.coordinates, KilToRad(60)]}}, "styles" :{$in :["blues", "soul"]}}

db.salles.find(req, {"\_id":0, "nom":1})

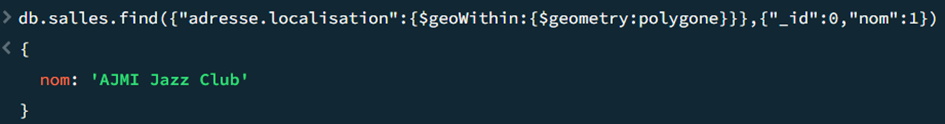
Exo 2 :

var marseille = {"type": "Point", "coordinates": [43.300000, 5.400000]}

db.salles.createIndex({"adresse.localisation": "2dsphere"})

db   
  
db.salles.find({"adresse.localisation": {$nearSphere:{$geometry: marseille, $maxDistance: 100000}}}, {"\_id":0, "adresse.ville":1})

Exo3 :  
  

Create collection.

