

NoSQL Assignment

Enrollment No: MSCIT23B18

Roll No: 18

Smit Joshi | NoSQL | 29/07/2023

# SET 1

## Create a collection named "book" and insert 5 records with following document schema:

##### db.createCollection(“Book”);

## Book\_code, Book\_name, Author: more than 1 author is possible, Publisher\_name, Year\_of\_publication, type\_of\_book: [textbook, reference, periodicals], Cost

### Inserting Records

Enterprise **NoSQL\_Collage**> db.Books.insertMany([

  {

    "Book\_Code": "1",

    "Book\_name": "Head First Design Patterns",

    "Author": ["Eric Freeman","Elisabeth Robson","Bert bates","Kathy Sierra"],

    "Publisher\_name": "O'Reilly Media",

    "Year\_Of\_Publication": "2020",

    "type\_of\_book": "Textbook",

    "Cost": 2878

  },

  {

    "Book\_Code": "2",

    "Book\_name": "Data Structures And Algorithms in Python,3rd Edition",

    "Author": ["Michael T.Goodrich","Roberto Tamassia","Michael H.Goldwasser"],

    "Publisher\_name": "Peasron",

    "Year\_Of\_Publication": "2020",

    "type\_of\_book": "Textbook",

    "Cost": 350

  },

  {

    "Book\_Code": "3",

    "Book\_name": "The Elements of Style",

    "Author": ["William Strunk Jr.","E.B. White"],

    "Publisher\_name": "Allyn & Bacon",

    "Year\_Of\_Publication": "2018",

    "type\_of\_book": "Reference",

    "Cost": 1072

  },

  {

    "Book\_Code": "4",

    "Book\_name": "Cracking the Coding Interview: 189 Programming Questions and Solutions",

    "Author": ["Gayle Laakmann McDowell"],

    "Publisher\_name": "CareerCup",

    "Year\_Of\_Publication": "2020",

    "type\_of\_book": "Reference",

    "Cost": 2313

  },

  {

    "Book\_Code": "5",

    "Book\_name": "The Economist: The World in 2023",

    "Author": ["The Economist"],

    "Publisher\_name": "The Economist",

    "Year\_Of\_Publication": "2021",

    "type\_of\_book": "Periodicals",

    "Cost": 879

  },

  {

    "Book\_Code": "6",

    "Book\_name": "Scientific American: 50th Anniversary Edition",

    "Author": ["Scientific American"],

    "Publisher\_name": "Scientific American",

    "Year\_Of\_Publication": "2022",

    "type\_of\_book": "Periodicals",

    "cost": 1623

  }

]);

{

acknowledged: true,

insertedIds: {

'0': ObjectId("64e459bce552bdf2a9761023"),

'1': ObjectId("64e459bce552bdf2a9761024"),

'2': ObjectId("64e459bce552bdf2a9761025"),

'3': ObjectId("64e459bce552bdf2a9761026"),

'4': ObjectId("64e459bce552bdf2a9761027"),

'5': ObjectId("64e459bce552bdf2a9761028")

}

}

## Based on the above collection, write a mongodb query for the following:

### Display all the documents of the collection Book with only Book\_code, Book\_name, and author and cost fields.

Enterprise NoSQL\_Collage> db.Books.find({},

{ "Book\_Code": 1, "Book\_name": 1, "Author": 1, "Cost": 1, \_id: 0 });

[

  {

    Book\_Code: '1',

    Book\_name: 'Head First Design Patterns',

    Author: ['Eric Freeman', 'Elisabeth Robson', 'Bert bates', 'Kathy Sierra' ],

    Cost: 2878

  },

  {

    Book\_Code: '2',

    Book\_name: 'Data Structures And Algorithms in Python,3rd Edition',

    Author: [ 'Michael T.Goodrich', 'Roberto Tamassia', 'Michael H.Goldwasser' ],

    Cost: 240

  },

  {

    Book\_Code: '3',

    Book\_name: 'The Elements of Style',

    Author: [ 'William Strunk Jr.', 'E.B. White' ],

    Cost: 1072

  },

  {

    Book\_Code: '4',

    Book\_name: 'Cracking the Coding Interview: 189 Programming Questions and Solutions',

    Author: [ 'Gayle Laakmann McDowell' ],

    Cost: 2313

  },

  {

    Book\_Code: '5',

    Book\_name: 'The Economist: The World in 2023',

    Author: [ 'The Economist' ],

    Cost: 879

  },

  {

    Book\_Code: '6',

    Book\_name: 'Scientific American: 50th Anniversary Edition',

    Author: [ 'Scientific American' ],

    Cost: 1623

  }

]

### Update Book collection whose cost is greater than 300, update to 240 Indian rupees of Pearson publication.

Enterprise **NoSQL\_Collage**>  db.Books.updateMany({

    $and: [{ Publisher\_name: { $eq: "Peasron" } }, { Cost: { $gt: 300 } }]

}, { $set: { Cost: 240 } });

{

  acknowledged: true,

  insertedId: null,

  matchedCount: 1,

  modifiedCount: 1,

  upsertedCount: 0

}

### Display the third costlier book from the collection.

Enterprise **NoSQL\_Collage**> db.Books.find().sort({"Cost":-1}).skip(2).limit(1);

[

  {

    \_id: ObjectId("64e459bce552bdf2a9761028"),

    Book\_Code: '6',

    Book\_name: 'Scientific American: 50th Anniversary Edition',

    Author: [ 'Scientific American' ],

    Publisher\_name: 'Scientific American',

    Year\_Of\_Publication: '2022',

    type\_of\_book: 'Periodicals',

    Cost: 1623

  }

]

### Display the unique list of periodicals in chronologic order.

Enterprise **NoSQL\_Collage**> db.Books.find(

{ type\_of\_book: "Periodicals" }).sort({ Book\_name: 1 });

[

  {

    \_id: ObjectId("64e459bce552bdf2a9761028"),

    Book\_Code: '6',

    Book\_name: 'Scientific American: 50th Anniversary Edition',

    Author: [ 'Scientific American' ],

    Publisher\_name: 'Scientific American',

    Year\_Of\_Publication: '2022',

    type\_of\_book: 'Periodicals',

    Cost: 1623

  },

  {

    \_id: ObjectId("64e459bce552bdf2a9761027"),

    Book\_Code: '5',

    Book\_name: 'The Economist: The World in 2023',

    Author: [ 'The Economist' ],

    Publisher\_name: 'The Economist',

    Year\_Of\_Publication: '2021',

    type\_of\_book: 'Periodicals',

    Cost: 879

  }

]

### Rename Cost key to Price key for book published in year 2021.

Enterprise **NoSQL\_Collage**> db.Books.updateMany({ Year\_Of\_Publication: "2021" },

{ $rename: { "Cost": "Price" } });

{

  acknowledged: true,

  insertedId: null,

  matchedCount: 1,

  modifiedCount: 1,

  upsertedCount: 0

}

# SET 2

Create a collection named “Car” and insert 5 records with following document schema:

##### db.createCollection(“Car”);

### Model\_id, Model\_name, Brand\_name, Type\_of\_car: SUV, Sedan, XUV and Motor, Dimensions: it contains color, height, width and weight, Price\_on\_road

### Inserting Records

 Enterprise **NoSQL\_Collage**> db.Car.insertMany([

    {

      "Model\_id": "1",

      "Model\_name": "Rav4",

      "Brand\_name": "Toyota",

      "Type\_of\_car": "SUV",

      "Dimensions": {"color": "Blue","height": "68 inches","width": "73 inches","weight": 1588 },

      "Price\_on\_road": 2250000

    },

    {

      "Model\_id": "2",

      "Model\_name": "Civic",

      "Brand\_name": "Honda",

      "Type\_of\_car": "Sedan",

      "Dimensions": {"color": "Silver","height": "57 inches","width": "70 inches","weight": 1270 },

      "Price\_on\_road": 1875000

    },

    {

      "Model\_id": "3",

      "Model\_name": "X5",

      "Brand\_name": "BMW",

      "Type\_of\_car": "XUV",

      "Dimensions": { "color": "Black", "height": "68 inches", "width": "76 inches", "weight": 1905 },

      "Price\_on\_road": 3375000

    },

    {

      "Model\_id": "4",

      "Model\_name": "Mustang",

      "Brand\_name": "Ford",

      "Type\_of\_car": "Sedan",

      "Dimensions": {"color": "Red","height": "54 inches","width": "75 inches","weight": 1451 },

      "Price\_on\_road": 2625000

    },

    {

      "Model\_id": "5",

      "Model\_name": "Model 3",

      "Brand\_name": "Tesla",

      "Type\_of\_car": "Motor",

      "Dimensions": {"color": "White","height": "56 inches","width": "73 inches","weight": 1588 },

      "Price\_on\_road": 3750000

    },

    { "Model\_id": "6",

      "Model\_name": "Creta",

      "Brand\_name": "Hyundai",

      "Type\_of\_car": "SUV",

      "Dimensions": {"color": "Grey","height": "65 inches","width": "70 inches","weight": 1361 },

      "Price\_on\_road": 1580000

    },

    {

      "Model\_id": "7",

      "Model\_name": "Verna",

      "Brand\_name": "Hyundai",

      "Type\_of\_car": "Sedan",

      "Dimensions": {"color": "White","height": "56 inches","width": "69 inches","weight": 1315 },

      "Price\_on\_road": 1450000

    },

    {

      "Model\_id": "8",

      "Model\_name": "Alto",

      "Brand\_name": "Maruti Suzuki",

      "Type\_of\_car": "Hatchback",

      "Dimensions": {"color": "Red","height": "53 inches","width": "58 inches","weight": 725 },

      "Price\_on\_road": 550000

    },

    {

      "Model\_id": "9",

      "Model\_name": "Kwid",

      "Brand\_name": "Renault",

      "Type\_of\_car": "Hatchback",

      "Dimensions": {"color": "White","height": "55 inches","width": "59 inches","weight": 816 },

      "Price\_on\_road": 620000

    },

    {

      "Model\_id": "10",

      "Model\_name": "Santro",

      "Brand\_name": "Hyundai",

      "Type\_of\_car": "Hatchback",

      "Dimensions": {"color": "Blue","height": "56 inches","width": "63 inches","weight": 862 },

      "Price\_on\_road": 680000

    },

    {

      "Model\_id": "11",

      "Model\_name": "Tucson",

      "Brand\_name": "Hyundai",

      "Type\_of\_car": "SUV",

      "Dimensions": {"color": "Black","height": "67 inches","width": "73 inches","weight": 1950 },

      "Price\_on\_road": 2890000

    },

    {

      "Model\_id": "012",

      "Model\_name": "Creta Plus",

      "Brand\_name": "Hyundai",

      "Type\_of\_car": "SUV",

      "Dimensions": {"color": "Black","height": "66 inches","width": "72 inches","weight": 1588 },

      "Price\_on\_road": 1050000

    }

  ]);

{

    acknowledged: true,

    insertedIds: {

      '0': ObjectId("64e4d4e9d8f9ad3cd60d99fc"),

      '1': ObjectId("64e4d4e9d8f9ad3cd60d99fd"),

      '2': ObjectId("64e4d4e9d8f9ad3cd60d99fe"),

      '3': ObjectId("64e4d4e9d8f9ad3cd60d99ff"),

      '4': ObjectId("64e4d4e9d8f9ad3cd60d9a00"),

      '5': ObjectId("64e4d4e9d8f9ad3cd60d9a01"),

      '6': ObjectId("64e4d4e9d8f9ad3cd60d9a02"),

      '7': ObjectId("64e4d4e9d8f9ad3cd60d9a03"),

      '8': ObjectId("64e4d4e9d8f9ad3cd60d9a04"),

      '9': ObjectId("64e4d4e9d8f9ad3cd60d9a05"),

      '10': ObjectId("64e4d4e9d8f9ad3cd60d9a06"),

      '11': ObjectId("64e4d4e9d8f9ad3cd60d9a07")

    }

}

## Based on the above collection, write a mongodb query for the following:

### Display all the documents in the collection with brand name, type of car and price of road fields.

Enterprise **NoSQL\_Collage**> db.Car.find({},

    { Brand\_name: 1, Type\_of\_car: 1, Price\_on\_road: 1, \_id: 0 });

[

  { Brand\_name: 'Toyota', Type\_of\_car: 'SUV', Price\_on\_road: 2250000 },

  { Brand\_name: 'Honda', Type\_of\_car: 'Sedan', Price\_on\_road: 1875000 },

  { Brand\_name: 'BMW', Type\_of\_car: 'XUV', Price\_on\_road: 3375000 },

  { Brand\_name: 'Ford', Type\_of\_car: 'Sedan', Price\_on\_road: 2625000 },

  { Brand\_name: 'Tesla', Type\_of\_car: 'Motor', Price\_on\_road: 3750000 },

  { Brand\_name: 'Hyundai', Type\_of\_car: 'SUV', Price\_on\_road: 1580000 },

  { Brand\_name: 'Hyundai', Type\_of\_car: 'Sedan', Price\_on\_road: 1450000 },

  { Brand\_name: 'Maruti Suzuki', Type\_of\_car: 'Hatchback', Price\_on\_road: 550000 },

  { Brand\_name: 'Renault', Type\_of\_car: 'Hatchback', Price\_on\_road: 620000 },

  { Brand\_name: 'Hyundai', Type\_of\_car: 'Hatchback', Price\_on\_road: 680000 },

  { Brand\_name: 'Hyundai', Type\_of\_car: 'SUV', Price\_on\_road: 2890000 },

  { Brand\_name: 'Hyundai', Type\_of\_car: 'SUV', Price\_on\_road: 1050000 }

]

### Display top three costliest Car of brand Hyundai.

Enterprise NoSQL\_Collage> db.Car.find({ Brand\_name: "Hyundai" }).sort({ Price\_on\_road: -1 }).limit(3);

[

  { \_id: ObjectId("64e4d4e9d8f9ad3cd60d9a06"),

    Model\_id: '11',

    Model\_name: 'Tucson',

    Brand\_name: 'Hyundai',

    Type\_of\_car: 'SUV',

    Dimensions: { color: 'Black', height: '67 inches', width: '73 inches', weight: 1950 },

    Price\_on\_road: 2890000

  },

  {

    \_id: ObjectId("64e4d4e9d8f9ad3cd60d9a01"),

    Model\_id: '6',

    Model\_name: 'Creta',

    Brand\_name: 'Hyundai',

    Type\_of\_car: 'SUV',

    Dimensions: { color: 'Grey', height: '65 inches', width: '70 inches', weight: 1361 },

    Price\_on\_road: 1580000

  },

  {

    \_id: ObjectId("64e4d4e9d8f9ad3cd60d9a02"),

    Model\_id: '7',

    Model\_name: 'Verna',

    Brand\_name: 'Hyundai',

    Type\_of\_car: 'Sedan',

    Dimensions: { color: 'White', height: '56 inches', width: '69 inches', weight: 1315 },

    Price\_on\_road: 1450000

  }

]

1. Delete all the documents whose dimensional weight is greater than 200kgs and colour black.

Enterprise NoSQL\_Collage> db.Car.deleteMany({

$and: [

{ "Dimensions.color": { $eq: "Black" } },

{ "Dimensions.weight": { $gt: 200 } }

] });

{ acknowledged: true, deletedCount: 3 }

1. Update car details for all Sedan cars.

Enterprise NoSQL\_Collage> db.Car.updateMany(

    { Type\_of\_car: "Sedan" },

    { $inc: { Price\_on\_road: 10000 } }

);

{

  acknowledged: true,

  insertedId: null,

  matchedCount: 3,

  modifiedCount: 3,

  upsertedCount: 0

}

1. Display all SUV cars whose price of road is greater than 10 lacs.

Enterprise **NoSQL\_Collage**> db.Car.find({

    $and: [

        { Type\_of\_car: { $eq: "SUV" } },

        { Price\_on\_road: { $gt: 1000000 } }

    ]

});

[

  {

    \_id: ObjectId("64e4d4e9d8f9ad3cd60d99fc"),

    Model\_id: '1',

    Model\_name: 'Rav4',

    Brand\_name: 'Toyota',

    Type\_of\_car: 'SUV',

    Dimensions: { color: 'Blue', height: '68 inches', width: '73 inches', weight: 1588 },

    Price\_on\_road: 2250000

  },

  {

    \_id: ObjectId("64e4d4e9d8f9ad3cd60d9a01"),

    Model\_id: '6',

    Model\_name: 'Creta',

    Brand\_name: 'Hyundai',

    Type\_of\_car: 'SUV',

    Dimensions: { color: 'Grey', height: '65 inches', width: '70 inches', weight: 1361 },

    Price\_on\_road: 1580000

  }

]