

NoSQL Assignment

Roll No: 18

Smit Joshi | NoSQL | 23/08/2023

[](https://github.com/smit-joshi814)

# 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

  }

]

# SET 3

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

##### db.createCollection(“Product”);

product\_id, product\_name, product\_type: more than 1 type is possible, cost\_unit, qty\_in\_stock

### Inserting Records.

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

    {

      "product\_id": 1,

      "product\_name": "Pencil",

      "product\_type": ["Stationary"],

      "cost\_unit": 50,

      "qty\_in\_stock": 100

    },

    {

      "product\_id": 2,

      "product\_name": "Notebook",

      "product\_type": ["Stationary"],

      "cost\_unit": 100,

      "qty\_in\_stock": 50

    },

    {

      "product\_id": 3,

      "product\_name": "Laptop",

      "product\_type": ["Electronics"],

      "cost\_unit": 60000,

      "qty\_in\_stock": 20

    },

    {

      "product\_id": 4,

      "product\_name": "Tablet",

      "product\_type": ["Electronics", "Computer"],

      "cost\_unit": 40000,

      "qty\_in\_stock": 50

    },

    {

      "product\_id": 5,

      "product\_name": "Chair",

      "product\_type": ["Home Decor"],

      "cost\_unit": 5000,

      "qty\_in\_stock": 100

    },

    {

      "product\_id": 6,

      "product\_name": "Book",

      "product\_type": ["Stationary", "Literature"],

      "cost\_unit": 300,

      "qty\_in\_stock": 100

    },

    {

      "product\_id": 7,

      "product\_name": "Mobile Phone",

      "product\_type": ["Electronics"],

      "cost\_unit": 10000,

      "qty\_in\_stock": 50

    },{

      "product\_id": 8,

      "product\_name": "Sofa",

      "product\_type": ["Home Decor"],

      "cost\_unit": 10000,

      "qty\_in\_stock": 100

    }

  ]);

{

    acknowledged: true,

    insertedIds: {

      '0': ObjectId("64e60c583c1de080a4aa72fe"),

      '1': ObjectId("64e60c583c1de080a4aa72ff"),

      '2': ObjectId("64e60c583c1de080a4aa7300"),

      '3': ObjectId("64e60c583c1de080a4aa7301"),

      '4': ObjectId("64e60c583c1de080a4aa7302"),

      '5': ObjectId("64e60c583c1de080a4aa7303"),

      '6': ObjectId("64e60c583c1de080a4aa7304"),

      '7': ObjectId("64e60c583c1de080a4aa7305")

    }

}

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

1. Display those products which are electronics and having price lower than 15k, with only product\_id, product\_name, cost\_unit, qty\_in\_stock

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

    $and: [

        { "product\_type": { $eq: "Electronics" } },

        { "cost\_unit": { $lt: 15000 } }

    ]

}, { "product\_id": 1, "product\_name": 1, "cost\_unit": 1, "qty\_in\_stock": 1, \_id: 0 });

[

    {

      product\_id: 7,

      product\_name: 'Mobile Phone',

      cost\_unit: 10000,

      qty\_in\_stock: 50

    }

]

1. Display only first 2 records whose product type is Stationary.

Enterprise **NoSQL\_Collage**> db.Product.find({ product\_type: "Stationary" }).limit(2);

[

  {

    \_id: ObjectId("64e60c583c1de080a4aa72fe"),

    product\_id: 1,

    product\_name: 'Pencil',

    product\_type: [ 'Stationary' ],

    cost\_unit: 50,

    qty\_in\_stock: 100

  },

  {

    \_id: ObjectId("64e60c583c1de080a4aa72ff"),

    product\_id: 2,

    product\_name: 'Notebook',

    product\_type: [ 'Stationary' ],

    cost\_unit: 100,

    qty\_in\_stock: 50

  }

]

1. Update the cost\_unit of products of Home decor category.

Enterprise **NoSQL\_Collage**> db.Product.updateMany({ product\_type: "Home Decor" },

{ $inc: { cost\_unit: 100 } });

{

  acknowledged: true,

  insertedId: null,

  matchedCount: 2,

  modifiedCount: 2,

  upsertedCount: 0

}

1. Delete product of Electronics category

Enterprise **NoSQL\_Collage**> db.Product.deleteMany({ product\_type: "Electronics" });

{ acknowledged: true, deletedCount: 3 }

1. Add new key named reorder\_qty whose cost\_unit range in between 10k to 15k.

Enterprise **NoSQL\_Collage**> db.Product.updateMany({ cost\_unit: { $gt: 10000, $lt: 15000 } }, { $set: { "reorder\_qty": 500 } });

{

  acknowledged: true,

  insertedId: null,

  matchedCount: 1,

  modifiedCount: 1,

  upsertedCount: 0

}

# SET 4

Create a collection named “Tour\_package” and insert 5 records with following document schema:

##### db.createCollection(“Tour\_package”);

### tour\_code, tour\_codename, source station, destination station, type\_of\_package: [pilgrimage, romantic,

### group], category: [Premium, Deluxe and Normal], total\_fare, total\_kms\_covered.

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

    {

        "tour\_code": 1,

        "tour\_codename": "Ladakh Tour",

        "source\_station": "Leh",

        "destination\_station": "Srinagar, Manali",

        "type\_of\_package": ["pilgrimage", "group"],

        "category": "Premium",

        "total\_fare": 150000,

        "total\_kms\_covered": 4000

    },

    {

        "tour\_code": 2,

        "tour\_codename": "Kashmir Tour",

        "source\_station": "Srinagar",

        "destination\_station": "Pahalgam, Gulmarg, Sonmarg",

        "type\_of\_package": ["romantic", "group"],

        "category": "Deluxe",

        "total\_fare": 95000,

        "total\_kms\_covered": 3500

    },

    {

        "tour\_code": 3,

        "tour\_codename": "North East Tour",

        "source\_station": "Guwahati",

        "destination\_station": "Shillong, Cherrapunji, Kaziranga",

        "type\_of\_package": ["group"],

        "category": "Normal",

        "total\_fare": 70000,

        "total\_kms\_covered": 2000

    },

    {

        "tour\_code": 4,

        "tour\_codename": "Kerala Hill Stations Tour",

        "source\_station": "Trivandrum",

        "destination\_station": "Munnar, Thekkady, Kovalam",

        "type\_of\_package": ["group"],

        "category": "Normal",

        "total\_fare": 65000,

        "total\_kms\_covered": 1800

    },

    {

        "tour\_code": 5,

        "tour\_codename": "Tamil Nadu Tour",

        "source\_station": "Chennai",

        "destination\_station": "Madurai, Trichy, Pondicherry",

        "type\_of\_package": ["group"],

        "category": "Normal",

        "total\_fare": 55000,

        "total\_kms\_covered": 1500

    },

    {

        "tour\_code": 6,

        "tour\_codename": "Andhra Pradesh Tour",

        "source\_station": "Hyderabad",

        "destination\_station": "Tirupati, Vijayawada, Hampi",

        "type\_of\_package": ["group"],

        "category": "Normal",

        "total\_fare": 45000,

        "total\_kms\_covered": 1200

    },

    {

        "tour\_code": 7,

        "tour\_codename": "Karnataka Tour",

        "source\_station": "Bangalore",

        "destination\_station": "Mysore, Hampi, Coorg",

        "type\_of\_package": ["group"],

        "category": "Normal",

        "total\_fare": 25000,

        "total\_kms\_covered": 1000

    }

]);

{

    acknowledged: true,

    insertedIds: {

      '0': ObjectId("64e617f03c1de080a4aa7310"),

      '1': ObjectId("64e617f03c1de080a4aa7311"),

      '2': ObjectId("64e617f03c1de080a4aa7312"),

      '3': ObjectId("64e617f03c1de080a4aa7313"),

      '4': ObjectId("64e617f03c1de080a4aa7314"),

      '5': ObjectId("64e617f03c1de080a4aa7315"),

      '6': ObjectId("64e617f03c1de080a4aa7316")

    }

}

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

1. Delete the first document of the collection having fields tour\_code, tour\_codename, source station and destination station.

Enterprise **NoSQL\_Collage**> db.Tour\_package.deleteOne({

    $and: [

        { "tour\_code": { $exists: true } },

        { "tour\_codename": { $exists: true } },

        { "source\_station": { $exists: true } },

        { "destination\_station": { $exists: true } }

    ]

});

{ acknowledged: true, deletedCount: 1 }

1. Update the tour\_package collection, by updating the source station of all tours to Delhi. After updating display the records in ascending order.

Enterprise **NoSQL\_Collage**> db.Tour\_package.updateMany({}, { $set: { source\_station: "Delhi" } });

{

  acknowledged: true,

  insertedId: null,

  matchedCount: 6,

  modifiedCount: 6,

  upsertedCount: 0

}

Enterprise **NoSQL\_Collage**> db.Tour\_package.find().sort({ tour\_code: 1 });

[ { \_id: ObjectId("64e617f03c1de080a4aa7311"),

    tour\_code: 2,

    tour\_codename: 'Kashmir Tour',

    source\_station: 'Delhi',

    destination\_station: 'Pahalgam, Gulmarg, Sonmarg',

    type\_of\_package: [ 'romantic', 'group' ],

    category: 'Deluxe',

    total\_fare: 95000,

    total\_kms\_covered: 3500

  },{

    \_id: ObjectId("64e617f03c1de080a4aa7312"),

    tour\_code: 3,

    tour\_codename: 'North East Tour',

    source\_station: 'Delhi',

    destination\_station: 'Shillong, Cherrapunji, Kaziranga',

    type\_of\_package: [ 'group' ],

    category: 'Normal',

    total\_fare: 70000,

    total\_kms\_covered: 2000

  },

  {

    \_id: ObjectId("64e617f03c1de080a4aa7313"),

    tour\_code: 4,

    tour\_codename: 'Kerala Hill Stations Tour',

    source\_station: 'Delhi',

    destination\_station: 'Munnar, Thekkady, Kovalam',

    type\_of\_package: [ 'group' ],

    category: 'Normal',

    total\_fare: 65000,

    total\_kms\_covered: 1800

  },

  {

    \_id: ObjectId("64e617f03c1de080a4aa7314"),

    tour\_code: 5,

    tour\_codename: 'Tamil Nadu Tour',

    source\_station: 'Delhi',

    destination\_station: 'Madurai, Trichy, Pondicherry',

    type\_of\_package: [ 'group' ],

    category: 'Normal',

    total\_fare: 55000,

    total\_kms\_covered: 1500

  },

  {

    \_id: ObjectId("64e617f03c1de080a4aa7315"),

    tour\_code: 6,

    tour\_codename: 'Andhra Pradesh Tour',

    source\_station: 'Delhi',

    destination\_station: 'Tirupati, Vijayawada, Hampi',

    type\_of\_package: [ 'group' ],

    category: 'Normal',

    total\_fare: 45000,

    total\_kms\_covered: 1200

  },

  {

    \_id: ObjectId("64e617f03c1de080a4aa7316"),

    tour\_code: 7,

    tour\_codename: 'Karnataka Tour',

    source\_station: 'Delhi',

    destination\_station: 'Mysore, Hampi, Coorg',

    type\_of\_package: [ 'group' ],

    category: 'Normal',

    total\_fare: 25000,

    total\_kms\_covered: 1000

  }

]

1. Display three costliest tours.

Enterprise **NoSQL\_Collage**> db.Tour\_package.find().sort({ totle\_fare: -1 }).limit(3);

[

  {

    \_id: ObjectId("64e617f03c1de080a4aa7311"),

    tour\_code: 2,

    tour\_codename: 'Kashmir Tour',

    source\_station: 'Delhi',

    destination\_station: 'Pahalgam, Gulmarg, Sonmarg',

    type\_of\_package: [ 'romantic', 'group' ],

    category: 'Deluxe',

    total\_fare: 95000,

    total\_kms\_covered: 3500

  },

  {

    \_id: ObjectId("64e617f03c1de080a4aa7312"),

    tour\_code: 3,

    tour\_codename: 'North East Tour',

    source\_station: 'Delhi',

    destination\_station: 'Shillong, Cherrapunji, Kaziranga',

    type\_of\_package: [ 'group' ],

    category: 'Normal',

    total\_fare: 70000,

    total\_kms\_covered: 2000

  },

  {

    \_id: ObjectId("64e617f03c1de080a4aa7313"),

    tour\_code: 4,

    tour\_codename: 'Kerala Hill Stations Tour',

    source\_station: 'Delhi',

    destination\_station: 'Munnar, Thekkady, Kovalam',

    type\_of\_package: [ 'group' ],

    category: 'Normal',

    total\_fare: 65000,

    total\_kms\_covered: 1800

  }

]

1. Display the tour records whose total fare is in a range of 20K-30K and total\_kms\_covered lower than 2000 kms.

Enterprise **NoSQL\_Collage**> db.Tour\_package.find({

$and: [ { total\_fare: { $gt: 20000, $lt: 30000 } }, { total\_kms\_covered: { $lt: 2000 } }]

});

[

  {

    \_id: ObjectId("64e617f03c1de080a4aa7316"),

    tour\_code: 7,

    tour\_codename: 'Karnataka Tour',

    source\_station: 'Delhi',

    destination\_station: 'Mysore, Hampi, Coorg',

    type\_of\_package: [ 'group' ],

    category: 'Normal',

    total\_fare: 25000,

    total\_kms\_covered: 1000

  }

]

1. Display all tour packages who do not belong to Premium category of tour packages.

Enterprise **NoSQL\_Collage**> db.Tour\_package.find({ category: { $not: { $eq: "Premium" } } });

[

  {

    \_id: ObjectId("64e617f03c1de080a4aa7311"),

    tour\_code: 2,

    tour\_codename: 'Kashmir Tour',

    source\_station: 'Delhi',

    destination\_station: 'Pahalgam, Gulmarg, Sonmarg',

    type\_of\_package: [ 'romantic', 'group' ],

    category: 'Deluxe',

    total\_fare: 95000,

    total\_kms\_covered: 3500

  },

  {

    \_id: ObjectId("64e617f03c1de080a4aa7312"),

    tour\_code: 3,

    tour\_codename: 'North East Tour',

    source\_station: 'Delhi',

    destination\_station: 'Shillong, Cherrapunji, Kaziranga',

    type\_of\_package: [ 'group' ],

    category: 'Normal',

    total\_fare: 70000,

    total\_kms\_covered: 2000

  },

  {

    \_id: ObjectId("64e617f03c1de080a4aa7313"),

    tour\_code: 4,

    tour\_codename: 'Kerala Hill Stations Tour',

    source\_station: 'Delhi',

    destination\_station: 'Munnar, Thekkady, Kovalam',

    type\_of\_package: [ 'group' ],

    category: 'Normal',

    total\_fare: 65000,

    total\_kms\_covered: 1800

  },

  {

    \_id: ObjectId("64e617f03c1de080a4aa7314"),

    tour\_code: 5,

    tour\_codename: 'Tamil Nadu Tour',

    source\_station: 'Delhi',

    destination\_station: 'Madurai, Trichy, Pondicherry',

    type\_of\_package: [ 'group' ],

    category: 'Normal',

    total\_fare: 55000,

    total\_kms\_covered: 1500

  },

  {

    \_id: ObjectId("64e617f03c1de080a4aa7315"),

    tour\_code: 6,

    tour\_codename: 'Andhra Pradesh Tour',

    source\_station: 'Delhi',

    destination\_station: 'Tirupati, Vijayawada, Hampi',

    type\_of\_package: [ 'group' ],

    category: 'Normal',

    total\_fare: 45000,

    total\_kms\_covered: 1200

  },

  {

    \_id: ObjectId("64e617f03c1de080a4aa7316"),

    tour\_code: 7,

    tour\_codename: 'Karnataka Tour',

    source\_station: 'Delhi',

    destination\_station: 'Mysore, Hampi, Coorg',

    type\_of\_package: [ 'group' ],

    category: 'Normal',

    total\_fare: 25000,

    total\_kms\_covered: 1000

  }

]

# SET 5

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

##### db.createCollection(“Watch”);

model\_id, model\_name, brand\_name, type\_of\_dial: analog, digital, chronograph, dimension: it contains height, width and weight, price

Enterprise **NoSQL\_Collage**> db.Watch.insertMany(

[

    {

      "model\_id": 1,

      "model\_name": "Rolex Submariner",

      "brand\_name": "Rolex",

      "type\_of\_dial": "analog",

      "dimension": { "height": 12, "width": 40, "weight": 150 },

      "price": 66040

    },

    {

      "model\_id": 2,

      "model\_name": "Omega Speedmaster",

      "brand\_name": "Omega",

      "type\_of\_dial": "analog",

      "dimension": { "height": 10, "width": 38, "weight": 120 },

      "price": 50530

    },

    {

      "model\_id": 3,

      "model\_name": "Seiko Prospex",

      "brand\_name": "Seiko",

      "type\_of\_dial": "analog",

      "dimension": { "height": 11, "width": 42, "weight": 130 },

      "price": 33020

    },

    {

      "model\_id": 4,

      "model\_name": "Casio G-Shock",

      "brand\_name": "Casio",

      "type\_of\_dial": "digital",

      "dimension": { "height": 10, "width": 43, "weight": 100 },

      "price": 16510

    },

    {

      "model\_id": 5,

      "model\_name": "Timex Ironman",

      "brand\_name": "Timex",

      "type\_of\_dial": "digital",

      "dimension": { "height": 9, "width": 36, "weight": 80 },

      "price": 8255

    },

    {

      "model\_id": 6,

      "model\_name": "Fossil Grant Chronograph FS4810",

      "brand\_name": "Fossil",

      "type\_of\_dial": "chronograph",

      "dimension": { "height": 12, "width": 44, "weight": 120 },

      "price": 247647

    },

    {

      "model\_id": 7,

      "model\_name": "Fossil Townsman Chronograph FS5433",

      "brand\_name": "Fossil",

      "type\_of\_dial": "chronograph",

      "dimension": { "height": 11, "width": 42, "weight": 100 },

      "price": 206375

    },

    {

      "model\_id": 8,

      "model\_name": "Orient Bambino Ver. 2",

      "brand\_name": "Orient",

      "type\_of\_dial": "analog",

      "dimension": { "height": 12, "width": 40, "weight": 110 },

      "price": 16510

    },

    {

      "model\_id": 9,

      "model\_name": "Citizen Eco-Drive Promaster Diver",

      "brand\_name": "Citizen",

      "type\_of\_dial": "analog",

      "dimension": { "height": 13, "width": 44, "weight": 160 },

      "price": 247647

    }

]);

{

    acknowledged: true,

    insertedIds: {

      '0': ObjectId("64e630724a7006f7140b8aec"),

      '1': ObjectId("64e630724a7006f7140b8aed"),

      '2': ObjectId("64e630724a7006f7140b8aee"),

      '3': ObjectId("64e630724a7006f7140b8aef"),

      '4': ObjectId("64e630724a7006f7140b8af0"),

      '5': ObjectId("64e630724a7006f7140b8af1"),

      '6': ObjectId("64e630724a7006f7140b8af2"),

      '7': ObjectId("64e630724a7006f7140b8af3"),

      '8': ObjectId("64e630724a7006f7140b8af4")

    }

}

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

1. Display model\_name, brand, and price whose price lies in range 15k-20k.

Enterprise **NoSQL\_Collage**> db.Watch.find({ price: { $gt: 15000, $lt: 20000 } }, { model\_name: 1, brand\_name: 1, price: 1, \_id: 0 });

[

  { model\_name: 'Casio G-Shock', brand\_name: 'Casio', price: 16510 },

  {

    model\_name: 'Orient Bambino Ver. 2',

    brand\_name: 'Orient',

    price: 16510

  }

]

1. Update the collection using the price field and display all the fields of document.

Enterprise **NoSQL\_Collage**> db.Watch.updateMany({}, { $inc: { price: 1000 } });

{

  acknowledged: true,

  insertedId: null,

  matchedCount: 9,

  modifiedCount: 9,

  upsertedCount: 0

}

Enterprise **NoSQL\_Collage**> db.Watch.find().pretty();

[

  {

    \_id: ObjectId("64e630724a7006f7140b8aec"),

    model\_id: 1,

    model\_name: 'Rolex Submariner',

    brand\_name: 'Rolex',

    type\_of\_dial: 'analog',

    dimension: { height: 12, width: 40, weight: 150 },

    price: 67040

  },

  {

    \_id: ObjectId("64e630724a7006f7140b8aed"),

    model\_id: 2,

    model\_name: 'Omega Speedmaster',

    brand\_name: 'Omega',

    type\_of\_dial: 'analog',

    dimension: { height: 10, width: 38, weight: 120 },

    price: 51530

  },

  {

    \_id: ObjectId("64e630724a7006f7140b8aee"),

    model\_id: 3,

    model\_name: 'Seiko Prospex',

    brand\_name: 'Seiko',

    type\_of\_dial: 'analog',

    dimension: { height: 11, width: 42, weight: 130 },

    price: 34020

  },{

    \_id: ObjectId("64e630724a7006f7140b8aef"),

    model\_id: 4,

    model\_name: 'Casio G-Shock',

    brand\_name: 'Casio',

    type\_of\_dial: 'digital',

    dimension: { height: 10, width: 43, weight: 100 },

    price: 17510

  },{

    \_id: ObjectId("64e630724a7006f7140b8af0"),

    model\_id: 5,

    model\_name: 'Timex Ironman',

    brand\_name: 'Timex',

    type\_of\_dial: 'digital',

    dimension: { height: 9, width: 36, weight: 80 },

    price: 9255

  },{

    \_id: ObjectId("64e630724a7006f7140b8af1"),

    model\_id: 6,

    model\_name: 'Fossil Grant Chronograph FS4810',

    brand\_name: 'Fossil',

    type\_of\_dial: 'chronograph',

    dimension: { height: 12, width: 44, weight: 120 },

    price: 248647

  },{

    \_id: ObjectId("64e630724a7006f7140b8af2"),

    model\_id: 7,

    model\_name: 'Fossil Townsman Chronograph FS5433',

    brand\_name: 'Fossil',

    type\_of\_dial: 'chronograph',

    dimension: { height: 11, width: 42, weight: 100 },

    price: 207375

  },{

    \_id: ObjectId("64e630724a7006f7140b8af3"),

    model\_id: 8,

    model\_name: 'Orient Bambino Ver. 2',

    brand\_name: 'Orient',

    type\_of\_dial: 'analog',

    dimension: { height: 12, width: 40, weight: 110 },

    price: 17510

  },

  {

    \_id: ObjectId("64e630724a7006f7140b8af4"),

    model\_id: 9,

    model\_name: 'Citizen Eco-Drive Promaster Diver',

    brand\_name: 'Citizen',

    type\_of\_dial: 'analog',

    dimension: { height: 13, width: 44, weight: 160 },

    price: 248647

  }

]

1. Display watch records of brand Fossil and dial type chronographic in ascending order.

Enterprise **NoSQL\_Collage**> db.Watch.find({ $and: [ { brand\_name: { $eq: "Fossil" } }, { type\_of\_dial: { $eq: "chronograph" } }] }).sort({ model\_name: 1 });

[

  {

    \_id: ObjectId("64e630724a7006f7140b8af1"),

    model\_id: 6,

    model\_name: 'Fossil Grant Chronograph FS4810',

    brand\_name: 'Fossil',

    type\_of\_dial: 'chronograph',

    dimension: { height: 12, width: 44, weight: 120 },

    price: 248647

  },

  {

    \_id: ObjectId("64e630724a7006f7140b8af2"),

    model\_id: 7,

    model\_name: 'Fossil Townsman Chronograph FS5433',

    brand\_name: 'Fossil',

    type\_of\_dial: 'chronograph',

    dimension: { height: 11, width: 42, weight: 100 },

    price: 207375

  }

]

1. Delete all the documents of the collection.

Enterprise **NoSQL\_Collage**> db.Tour\_package.find(

{

$and: [

{ total\_fare: { $gt: 20000, $lt: 30000 } },

{ total\_kms\_covered: { $lt: 2000 } }

]

}

);

[

  {

    \_id: ObjectId("64e617f03c1de080a4aa7316"),

    tour\_code: 7,

    tour\_codename: 'Karnataka Tour',

    source\_station: 'Delhi',

    destination\_station: 'Mysore, Hampi, Coorg',

    type\_of\_package: [ 'group' ],

    category: 'Normal',

    total\_fare: 25000,

    total\_kms\_covered: 1000

  }

]

1. Display all watches which do not have chronographic dials.

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

type\_of\_dial: { $not: { $eq: "chronograph" } }

});

[

  {

    \_id: ObjectId("64e630724a7006f7140b8aec"),

    model\_id: 1,

    model\_name: 'Rolex Submariner',

    brand\_name: 'Rolex',

    type\_of\_dial: 'analog',

    dimension: { height: 12, width: 40, weight: 150 },

    price: 67040

  },

  {

    \_id: ObjectId("64e630724a7006f7140b8aed"),

    model\_id: 2,

    model\_name: 'Omega Speedmaster',

    brand\_name: 'Omega',

    type\_of\_dial: 'analog',

    dimension: { height: 10, width: 38, weight: 120 },

    price: 51530

  },

  {

    \_id: ObjectId("64e630724a7006f7140b8aee"),

    model\_id: 3,

    model\_name: 'Seiko Prospex',

    brand\_name: 'Seiko',

    type\_of\_dial: 'analog',

    dimension: { height: 11, width: 42, weight: 130 },

    price: 34020

  },

  {

    \_id: ObjectId("64e630724a7006f7140b8aef"),

    model\_id: 4,

    model\_name: 'Casio G-Shock',

    brand\_name: 'Casio',

    type\_of\_dial: 'digital',

    dimension: { height: 10, width: 43, weight: 100 },

    price: 17510

  },

  {

    \_id: ObjectId("64e630724a7006f7140b8af0"),

    model\_id: 5,

    model\_name: 'Timex Ironman',

    brand\_name: 'Timex',

    type\_of\_dial: 'digital',

    dimension: { height: 9, width: 36, weight: 80 },

    price: 9255

  },

  {

    \_id: ObjectId("64e630724a7006f7140b8af3"),

    model\_id: 8,

    model\_name: 'Orient Bambino Ver. 2',

    brand\_name: 'Orient',

    type\_of\_dial: 'analog',

    dimension: { height: 12, width: 40, weight: 110 },

    price: 17510

  },

  {

    \_id: ObjectId("64e630724a7006f7140b8af4"),

    model\_id: 9,

    model\_name: 'Citizen Eco-Drive Promaster Diver',

    brand\_name: 'Citizen',

    type\_of\_dial: 'analog',

    dimension: { height: 13, width: 44, weight: 160 },

    price: 248647

  }

]