
# NoSQL Assignment

Roll No: 18

Smit Joshi | NoSQL | 23/08/2023

# SET 1

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

- db.createCollection("Book");

1. 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:

1. 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
  }
]
```

2. 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
}
```

3. 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
  }
]
```

4. 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
  }
]
```

5. 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:

1. 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 }
]
```

2. 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
  }
]
```

3. 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 }
```

4. 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
}
```

5. 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
    }
]
```

2. 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
  }
]
```

3. 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
}
```

4. Delete product of Electronics category

```
Enterprise NoSQL_Collage> db.Product.deleteMany({ product_type: "Electronics" });

{ acknowledged: true, deletedCount: 3 }
```

5. 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 }
```

2. 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
  }
]
```

3. 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
  }
]
```

4. 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
  }
]
```

5. 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
  }
]
```

2. 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
  }
]
```

3. 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
  }
]
```

4. 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
  }
]
```

5. 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
  }
]
```