NoSQL

Problem Statement 2:

In this assignment, you need to implement a NoSQL database for an e-commerce platform. The database should store product information, customer orders, and transaction history. Design the data schema and implement a query that retrieves the top-selling products for a given time period.

NoSQL Data Model for E-commerce Platform (Using MongoDB)

The schema is designed for efficient querying of top-selling products while ensuring flexibility in handling varied data structures.

1. Products Collection

Stores information about products.

```
{
  "_id": "prod001",
  "name": "Wireless Headphones",
  "category": "Electronics",
  "price": 99.99,
  "stock": 150,
  "description": "High-quality wireless headphones with noise
cancellation",
  "created_at": "2024-01-15T08:00:00Z"
}
```

- Basic product details with a unique `_id`.
- Category and price for filtering and sorting.

2. Customers Collection

Stores customer details.

```
{
  "_id": "cust001",
  "name": "Alice Johnson",
  "email": "alice@example.com",
  "phone": "123-456-7890",
  "address": "123 Main St, Cityville",
  "joined_at": "2024-01-20T09:00:00Z"
}
```

- Stores essential customer information.
- `joined_at` for tracking customer registration.

3. Orders Collection

Stores order and transaction details.

```
{
  "_id": "ord001",
  "customer_id": "cust001",
  "order_date": "2024-02-01T10:00:00Z",
  "total_amount": 199.98,
  "status": "Completed",
  "items": [
    {
      "product_id": "prod001",
      "quantity": 2,
      "price": 99.99
    }
  ],
  "transaction_id": "txn001"
}
```

- Embedded `items` array stores products in the order.
- `transaction_id` links to payment details if necessary.

4. Transactions Collection

Stores transaction details.

```
{
  "_id": "txn001",
  "order_id": "ord001",
  "payment_method": "Credit Card",
  "payment_status": "Successful",
  "transaction_date": "2024-02-01T10:01:00Z"
}
```

- Tracks payment status and method.
- Links to orders for detailed history.

Query: Retrieve Top-Selling Products for a Given Time Period

To find the top-selling products based on quantity sold:

```
db.orders.aggregate([
  {
    $match: {
      order_date: { $gte: ISODate("2024-01-01T00:00:00Z"), $lte:
ISODate("2024-02-01T23:59:59Z") }
    }
  },
  {
    $unwind: "$items"
```

```
  },
  {
    $group: {
      _id: "$items.product_id",
      total_quantity_sold: { $sum: "$items.quantity" }
    }
  },
  {
    $lookup: {
      from: "products",
      localField: "_id",
      foreignField: "_id",
      as: "product_info"
    }
  },
  {
    $unwind: "$product_info"
  },
  {
    $sort: { total_quantity_sold: -1 }
  },
  {
    $project: {
      product_name: "$product_info.name",
      category: "$product_info.category",
      total_quantity_sold: 1
    }
  }
]);
```

Query Explanation

1. Filter Orders by Date: Limits to the specified time range.
2. Unwind Items: Breaks down orders to analyze individual products.
3. Group by Product ID: Calculates total quantity sold per product.
4. Lookup Product Details: Joins with `Products` collection for product names.
5. Sort by Quantity Sold: Orders results by highest sales.
6. Project Relevant Fields: Displays product name, category, and quantity sold.

Indexing Strategy

- Index on `orders.order_date` for efficient date filtering.
- Index on `orders.items.product_id` for quick aggregation.

```
db.orders.createIndex({ order_date: 1 });
db.orders.createIndex({ "items.product_id": 1 });
```

Advantages

- Embedded Documents: Simplifies and speeds up queries.
- Aggregation Pipeline: Efficiently processes data for insights.
- Scalable: Easily adapts to growing data and new features.