**MongoDB Project**

Based on the columns in superstore.csv dataset, we define which entities will make a database collection and their respective data (key: value pairs). we are dealing with order and product data in an e-commerce context. To design an Entity-Relationship Diagram (ERD) for MongoDB database, we must consider the relationships between the entities and how they interact. Since MongoDB is a NoSQL database, the relationships will be based on data embedding and referencing. In this task , we are using referencing to represent relationships between entities. This design allows us to efficiently query and retrieve data while maintaining the integrity of relationships.

**Entities:**

* Customer
* Order
* Product

**We will design 3 collections with above names in ecommerce database**

**Relationships:**

**One-to-Many Relationship: Customer to Orders and Customers to Products**

A customer can place multiple orders, but each order belongs to only one customer. An customer can purchase multiple products as well. We consider unique Customer ID as a foreign key in other collections

**ERD Design:**

**A screenshot of a computer

Description automatically generated**

This ERD design aims to reflect the relationships between customers, orders, and products in E-commerce context.

**Entity : Customer**

+------------------------+

\_id

Customer ID

Customer Name

Segment

Country

City

State

Postal Code

Region

+------------------------+

**Entity : Order**

+------------------------+

\_id

Order ID

Order Date

Ship Date

Ship Mode

Customer ID (Reference to Customer)

Sales

Quantity

Discount

Profit

**Entity :Product**

+------------------------+

Product ID

Customer ID (Reference to Customer)

Category

Sub-Category

Product Name

+------------------------+

Based on your provided ERD and considering the relationships between entities, we can design our collections in MongoDB as follows:

**Customer Collection :**

We create a collection named "customers" to store customer-related information. Each customer will have a unique "\_id" field generated by MongoDB. The structure of the collection could be:

{

"\_id": ObjectId("..."),

"customer\_id": "some\_customer\_id",

"customer\_name": "Customer Name",

"segment": "Segment",

"country": "Country",

"city": "City",

"state": "State",

"postal\_code": "Postal Code",

"region": "Region"

}

**Order Collection:** Create a collection named "orders" to store order-related information. Each order will have a unique "\_id" field generated by MongoDB. The "customer\_id" field will reference the "customer\_id" from the "customers" collection. The "products" field will contain an array of embedded products within each order. The structure of the collection could be:

{

"\_id": ObjectId("..."),

"order\_id": "some\_order\_id",

"order\_date": ISODate("..."),

"ship\_date": ISODate("..."),

"ship\_mode": "Ship Mode",

"customer\_id": ObjectId("..."), // Reference to Customer

"sales": some numbers,

"quantity": some numbers,

"discount": some numbers,

"profit": some numbers

}

**product Collection:**

{

"\_id": ObjectId("..."),

"product\_id": "some\_product\_id",

"customer\_id": ObjectId("..."), // Reference to Customer

"category": "Category",

"sub\_category": "Sub-Category",

"product\_name": "Product Name"

}

We will create appropriate indexes on fields that we will frequently query or use for filtering to ensure optimal query performance. To create an index use the command: db.collection.createIndex()