Database 2 Final project

Queries(MongoDB):

1)

```
db.Customers.aggregate([
{
$lookup: {
from: "Orders",
localField: "CUSTOMER_ID",
foreignField: "CUSTOMER_ID",
as: "orders"
}
},
{
$match: {
orders: { $size: 0 }
}
}
])
```

2)

```
db.orders.aggregate([
{
$lookup: {
from: "customers",
localField: "CUSTOMER_ID",
foreignField: "CUSTOMER_ID",
as: "customer"
}
},
{
$match: {
"customer.AGE": { $gte: 30 }
}
```

```
},
{
$group: {
_id: null,
totalPrice: { $sum: "$PRICE" }
}
}
])
```

3)

```
db.orders.aggregate([
{
$group: {
_id: "$CUSTOMER_ID",
totalOrderPrice: { $sum: "$PRICE" }
}
},
{
$sort: {
totalOrderPrice: -1
}
},
{
$limit: 5
},
{
$lookup: {
from: "Customers",
localField: "_id",
foreignField: "CUSTOMER_ID",
as: "customer"
}
},
{
$project: {
"customer.CUSTOMER_ID": 1,
"customer.FULL_NAME": 1,
```

```
"customer.ADDRESS": 1,
"customer.AGE": 1,
totalOrderPrice: 1
}
}
])
```

4)

```
db.clothes.aggregate([
  {
   $match: {
    CSIZE: "M",
    DISCOUNT: { $gte: 22 }
   }
  },
  {
   $group: {
    _id: "$PNAME",
    AVERAGEPRICE: { $avg: "$PRICE" }
   }
  },
  {
   $sort: {
    TOTALREVENUE: -1
   }
  },
  {
   $limit: 5
  }
])
```

5)

```
db.delivery.aggregate([
  {
   $group: {
     _id: "$CUSTOMER_ID",
     AVERAGEDELIVERYCOST: { $avg: "$PRICE" }
   }
  },
  {
   $sort: {
     AVERAGEDELIVERYCOST: 1
   }
  }
])
```

6)

```
db.orders.aggregate([
  {
   $match: {
     ORDER_TYPE: "goods",
     PRICE: { $gt: 9000 }
   }
  },
  {
   $project: {
     _id: 0,
     ORDER_ID: 1,
     CUSTOMER_ID: 1,
     ORDER_TYPE: 1,
     ODATE: 1,
     PRICE: 1
   }
  },
  {
```

```
    $sort: {
      PRICE: -1
    }
  }
])
```

Thus, the result of executing this code will be a list of the first 10 documents from the "services" collection that meet the price condition, sorted in descending order of price, and containing only the SERVISE_ID, SNAME, and PRICE fields.

7)

```
db.services.aggregate([
  {
    $match: {
      PRICE: { $gte: 50 } // greater than or equal to 50
    }
  },
  {
    $sort: {
      PRICE: -1 // descending order
    }
  },
  {
    $limit: 10
  },
  {
    $project: {
      _id: 0,
```

```
    SERVISE_ID: 1,
    SNAME: 1,
    PRICE: 1
   }
  }
])
```

8)

This query will return a single document containing the name of the youngest customer, the name of the oldest customer, the average age of all customers and the total number of customers.

```
db.customers.aggregate([
  {
   $sort: {
    AGE: 1
   }
  },
  {
   $group: {
    _id: null,
    youngestCustomer: { $first: "$FULLNAME" },
    oldestCustomer: { $last: "$FULLNAME" },
    averageAge: { $avg: "$AGE" },
    count: { $sum: 1 }
   }
  },
  {
   $project: {
    _id: 0,
    youngestCustomer: 1,
```

```
    oldestCustomer: 1,
    averageAge: 1,
    count: 1
   }
  }
])
```

9)

This query will return one document containing an array of vendor names (excluding the first two vendors in the sorted list) and the total number of vendors.

```
db.supplier.aggregate([
  {
   $sort: {
    NAME: 1 // Sort by name in ascending order
   }
  },
  {
   $skip: 2 // Skip the first 2 documents
  },
  {
   $group: {
    _id: null,
    suppliers: { $push: "$SNAME" }, // ARRAY
    count: { $sum: 1 } // Counter
   }
```

```
  },
  {
   $project: {
    _id: 0,
    suppliers: 1,
    count: 1
   }
  }
])
```

10)

```
db.customers.aggregate([
 {
  $lookup: {
   from: "orders",
   localField: "CUSTOMER_ID",
   foreignField: "CUSTOMER_ID",
   as: "orders"
  }
 },
 {
  $project: {
   CUSTOMER_ID: 1,
   AGE: 1,
   FULLNAME: 1,
   ADDRESS: 1,
   order_count: { $size: "$orders" }
  }
 }
])
```

11)

```
db.customers.aggregate([
 {
```

```
    $lookup: {
      from: "orders",
      localField: "CUSTOMER_ID",
      foreignField: "CUSTOMER_ID",
      as: "orders"
    }
  },
  {
    $project: {
      CUSTOMER_ID: 1,
      FULLNAME: 1,
      ORDER_COUNT: { $size: "$orders" }
    }
  },
  {
    $sort: {
      ORDER_COUNT: -1
    }
  },
  {
    $limit: 5
  }
])
```

12)

```
db.orders.aggregate([
  {
    $group: {
      _id: null,
      total_price: { $sum: "$PRICE" }
    }
  }
])
```

13)

```
db.customers.aggregate([
  {
    $match: { CUSTOMER_ID: 1 }
  },
  {
    $lookup: {
      from: "orders",
      localField: "CUSTOMER_ID",
      foreignField: "CUSTOMER_ID",
      as: "orders"
    }
  }
])
```

To see what kind of orders a customer has done

14)

```
db.sportshoes.aggregate([
  {
    $addFields: {
      priceAfterDiscount: {
        $multiply: [
          "$PRICE",
          {
            $subtract: [
              1,
```

```
            {
              $divide: ["$DISCOUNT", 100]
            }
          ]
        }
      ]
    }
  }
 },
 {
  $project: {
    _id: 0,
    PNAME: 1,
    PRICE: 1,
    DISCOUNT: 1,
    priceAfterDiscount: 1
  }
 }
])
```

To see price of shoe after discount

15)

```
db.sportnutrition.aggregate([
 {
  $addFields: {
    priceAfterDiscount: {
      $multiply: [
        "$PRICE",
        {
          $subtract: [
            1,
            {
              $divide: ["$DISCOUNT", 100]
            }
```

```
          ]
        }
      ]
    }
  }
 },
 {
  $project: {
    _id: 0,
    GOODS_ID: 1,
    PNAME: 1,
    CSIZE: 1,
    PRICE: 1,
    DISCOUNT: 1,
    priceAfterDiscount: 1
  }
 }
])
```

The same but for nutrition