NAME: CHANDNI SEHGAL

1. FIND 1 EXAMPLE QUERY FOR WHICH YOU WILL HAVE TO COMBINE MORE THAN COLLECTION AND FOR WHICH THE DATA IS BETTER TO RESIDE IN ONE COLLECTION. EXPLAIN WHY. (20%)

Collections considered: Products and Categories
It is beneficial to create a new collection that contains data from the combination of these two tables because Product collection contains only ID of categories, it does not contain its name or description. Similarly, Categories collection has only id, name and description, but does not contain any price, quantity, etc for a product that could belong to a category. Having a new single collection that contains such relevant and frequently used data would help reduce number of calls made to server.

An example use case:
Find name of categories for which the product price is greater than $50.
In this case, the variables such as price and quantity are present only in Products collection and category name is available only in Categories collection. With the new collection, it would require to search only in one collection and all pieces of information required (name) based on conditions (price) would be feasible.

How such a collection be useful?

1. It can help consumers see which categories are expensive to buy products from, which in turn could help them manage their monthly budget.
2. It could also help a store to manage their inventory by seeing how much quantity of product from each category is available.
3. Lookup (search) on a website for products based on category name.

CREATE A NEW COMBINED TABLE. (20%)

Included in a separate script also

Note: I am not clear if the requirement is to create a collection and use insert statement to populate the table. Hence, the query is added to the script that would create the collection. *

```
db.createCollection("expensiveCategories", { autoIndexId : true } )
```

* However, the records **not** fetched from using this newly created collection.

IMPLEMENT THE QUERY. (20%)

Included in separate script also.

```
db.getCollection('products').aggregate([
        {$match:
                {UnitPrice: {$gte:50}}
        },{$lookup:
                {from: "categories",
                localField: "CategoryID",
                foreignField: "CategoryID",
                as: "results"}
        },{$project:
        {
                _id: 1,ProductID: 1,
                CategoryID: 1,
                ProductName: 1,
                CategoryName: "$results.CategoryName",
                UnitPrice: 1}
        },{$sort:
                {UnitPrice: -1}
        }
]).pretty()
```

2. FIND 1 EXAMPLE QUERY THAT INVOLVE SEARCHING MULTIPLE TABLES BUT YOU THINK YOU WOULD WANT LEAVE IT IN SEPARATE COLLECTIONS.

EXPLAIN WHY. (20%)

It is beneficial to query through multiple collection but keep them separate when the data is changing frequently and reflecting the change would impact more than one collection. A field such as the price of a Freight can change frequently since it can be based on gas prices, currency value, custom taxes/charges, etc. of the country.

<u>An example use case:</u>
Find name of customers who are paying Freights over a $1000.
The Orders collection contains data that is relevant to each order, such as its location, customerID (who ordered it), employeeID (who sold it), Freight, Shipping, Location, etc. on the other hand, the Customers collection only contains its relevant information such as id, name, address, etc. Querying these two collections can help a company such as FedEx to determine where is shipping need to be delivered and what is the name of receiver.

However, it is beneficial to keep both as *separate collections* because the change in Freight would automatically be updated for the customer if he/she changes their location; and it would also automatically update for all customers if Freight prices change for a certain country. Both collections would get update, rather than each collection requiring a manual update.

IMPLEMENT THE QUERY. (20%)

Included in separate script also

```
db.getCollection('orders').aggregate([
        {$match:
                {Freight: { $gt: 1000 }}
        },{$lookup:
                {from: "customers",
                localField: "CustomerID",
                foreignField: "CustomerID",
                as: "results"}
        }, {$project:
                {_id: 0,
                ContactName: "$results.ContactName",
                Country: "$results.Country",Freight: 1}
        }, {$sort:
                {Freight: -1}
        }
]).pretty()
```