1. **Write Mongo query to retrieve the unique citye's from the buyers address as "_id".**

ANS : db.buyers.aggregate([

{ $group: { _id: "$address.city" } }

]);

2. **Write Mongo query to retrieve the unique zip from the buyers address as "_id".**

ANS : db.buyers.aggregate([

{ $group: { _id: "$address.zip" } }

]);

3. **Write Mongo query to retrieve the unique order_id in ascending order from the order_details.**

ANS : db.order_details.aggregate([

{ $group: { _id: "$order_id" } },

{ $sort: { _id: 1 } }

]);

4. **Write Mongo query to retrieve the unique customer_id from the orders.**

ANS : db.orders.aggregate([

{ $group: { _id: "$customer_id" } }

]);

5. **Write Mongo query to retrieve the unique paymentMethod's from the payments collection as "_id".**

ANS : db.payments.aggregate([

{ $group: { _id: "$paymentMethod" } }

]);

6. **Write Mongo query to retrieve the unique paymentstatus's from the payments collection as "_id".**

ANS : db.payments.aggregate([

{ $group: { _id: "$paymentstatus" } }

]);

7. **Write Mongo query to retrieve the unique category_id product from products.**

ANS : db.products.aggregate([

{ $group: { _id: "$category_id" } }

```
]);
```

8. **Write a MongoDB query to aggregate the total sales per customer and list the top 5 customers by total sales amount. Include the customer's ID and their total sales in the output.**

ANS :
```
db.orders.aggregate([
 { $group: { _id: "$customer_id", totalSales: { $sum: "$total" } } },
 { $sort: { totalSales: -1 } },
 { $limit: 5 }
]);
```

9. **Aggregate the orders to count how many there are per status and show only the first 3 statuses based on the aggregated count.**

ANS :
```
db.orders.aggregate([
 { $group: { _id: "$status", count: { $sum: 1 } } },
 { $sort: { count: -1 } },
 { $limit: 3 }
]);
```

10. **Write a MongoDB query to calculate the total amount of payments that have a success status.**

ANS :
```
db.payments.aggregate([
 { $match: { paymentstatus: "success" } },
 { $group: { _id: null, totalAmount: { $sum: "$amount" } } },
 { $project: { _id: 0, totalAmount: 1 } }
]);
```

11. **Aggregate suppliers to find the one with the highest total quantity of products supplied, filtering to only include suppliers with a total product quantity greater than 100.**

ANS :
```
db.products.aggregate([
 { $group: { _id: "$supplier_id", totalQuantity: { $sum: "$quantity" } } },
 { $match: { totalQuantity: { $gt: 100 } } },
 { $sort: { totalQuantity: -1 } },
 { $limit: 1 },
 {
```

```
  $lookup: {
    from: "suppliers",
    localField: "_id",
    foreignField: "_id",
    as: "supplier"
  }
 },
 { $unwind: "$supplier" },
 { $project: { supplier: { name: 1, phone: 1 }, totalQuantity: 1 } }
]);
```

12. Write a MongoDB query to find the top-selling product category based on total sales revenue.

ANS :

```
db.order_details.aggregate([
 { $lookup: { from: "products", localField: "product_id", foreignField: "_id", as: "product" } },
 { $unwind: "$product" },
 { $group: { _id: "$product.category_id", totalRevenue: { $sum: { $multiply: ["$quantity", "$price"] } } } },
 { $sort: { totalRevenue: -1 } },
 { $limit: 1 },
 {
  $lookup: {
    from: "categories",
    localField: "_id",
    foreignField: "_id",
    as: "category"
  }
 },
 { $unwind: "$category" },
 { $project: { category: { name: 1 }, totalRevenue: 1 } }
]);
```