1. Find the total revenue (price × quantity) for each item, sorted from highest to lowest.
2. db.sales.aggregate([
3. {
4. $project: {
5. item: 1,
6. revenue: { $multiply: ["$price", "$quantity"] }
7. }
8. },
9. {
10. $group: {
11. \_id: "$item",
12. totalRevenue: { $sum: "$revenue" }
13. }
14. },
15. {
16. $sort: { totalRevenue: -1 }
17. },
18. {
19. $project: {
20. \_id: 0,
21. item: "$\_id",
22. totalRevenue: 1
23. }
24. }
25. ]);

2. Calculate the total quantity sold per month in 2022.

db.sales.aggregate([

{

$match: {

date: {

$gte: ISODate("2022-01-01T00:00:00Z"),

$lt: ISODate("2023-01-01T00:00:00Z")}}

},

{

$group: {

\_id: {

$dateToString: { format: "%Y-%m", date: "$date" }

},

totalQuantity: { $sum: "$quantity" }

}},

{

$sort: { "\_id": 1 }},

{

$project: {

\_id: 0,

month: "$\_id",

totalQuantity: 1

}}

]);

3. Find all items where price is greater than 10 and size is not 'Short'.

db.sales.find({

price: { $gt: 10 },

size: { $ne: "Short" }

});

4. Find all records that do not have the field size..  
db.sales.find({

size: { $exists: false }

});

5. Get all Cappuccino sales with quantity between 10 and 20.  
db.sales.find({

item: "Cappuccino",

quantity: { $gt: 10, $lt: 20 }

});

6. Query to find items where the item name starts with "A".  
db.sales.find({

item: { $regex: /^A/, $options: "i" }

});

7. Find all sales that are either "Grande" or "Tall" but not "Americanos".  
db.sales.find({

size: { $in: ["Grande", "Tall"] },

item: { $ne: "Americanos" }

});

8. Find sales where the quantity is more than twice the price.  
db.sales.find({

$expr: {

$gt: ["$quantity", { $multiply: [2, "$price"] }]

}

});

9. Find all sales where the price is greater than the average price of their respective size.  
db.sales.aggregate([

{

$group: {

\_id: "$size",

avgPrice: { $avg: "$price" }

}

},

{

$lookup: {

from: "sales",

localField: "\_id",

foreignField: "size",

as: "salesWithSameSize"

}

},

{

$unwind: "$salesWithSameSize"

},

{

$replaceRoot: {

newRoot: {

$mergeObjects: [

"$salesWithSameSize",

{ avgPriceBySize: "$avgPrice" }

]

}

}

},

{

$match: {

$expr: { $gt: ["$price", "$avgPriceBySize"] }

}

},

]);

10. Filter sales where the total revenue is even and exceeds 100.  
db.sales.find({

$where: function() {

const total = this.price \* this.quantity;

return total > 100 && total % 2 === 0;

}

})

11. Find Sales Where the Day of Week Matches Quantity's Last Digit [Filter sales where the day of the week (0=Sunday, 1=Monday, etc.) matches the last digit of quantity]  
  
db.sales.find({

$where: function() {

const dayOfWeek = this.date.getUTCDay();

const lastDigit = this.quantity % 10;

return dayOfWeek === lastDigit;

}

});

12. Find Sales Where the Month is Prime and Quantity is Odd [Filter sales where the month (1-12) is a prime number (2,3,5,7,11) AND quantity is odd]  
db.sales.find({

$where: function() {

const month = this.date.getUTCMonth() + 1;

const isPrimeMonth = [2, 3, 5, 7, 11].includes(month);

const isOddQuantity = (this.quantity % 2) === 1;

return isPrimeMonth && isOddQuantity;

}

});

13. Find Sales with "Suspicious Quantities" (Divisible by 5 or 7) [Filter sales where quantity is divisible by 5 or 7]

db.sales.find({

$expr: {

$or: [

{ $eq: [{ $mod: ["$quantity", 5] }, 0] },

{ $eq: [{ $mod: ["$quantity", 7] }, 0] } ]}

});