2024202006_Lab8
Lakshay Baijal MongoDB

1. Using the sales collection, find the total revenue generated by each product using an aggregation pipeline.

```
schoolDB> db.sales.aggregate([
...   {
...     $group: {
...       _id: "$product_id",
...       totalRevenue: { $sum: { $multiply: ["$quantity", "$price"] } }
...     }
...   }
... ]);
[
  { _id: 'A123', totalRevenue: 479.70000000000005 },
  { _id: 'B456', totalRevenue: 699.3 }
]
```

2. Count the number of books authored by Alice in the books collection.

```
schoolDB> db.books.aggregate([
...   { $match: { author_name: "Alice" } },
...   { $count: "numberOfBooks" }
... ]);
[ { numberOfBooks: 2 } ]
```

3. Find all distinct product IDs from the sales collection.

```
schoolDB> db.sales.distinct("product_id");
[ 'A123', 'B456' ]
```

4. Using the students collection, find the average marks scored by each student.

```
schoolDB> db.students.aggregate([
...   {
...     $group: {
...       _id: "$Name",
...       averageMarks: { $avg: "$Marks" }
...     }
...   }
... ]);
[
  { _id: 'John', averageMarks: 87.5 },
  { _id: 'Emily', averageMarks: 80 }
]
```

5. Count how many active authors are there in the books collection.

```
schoolDB> db.books.aggregate([
...   { $match: { status: "active" } },
...   { $group: { _id: "$author_name" } },
...   { $count: "activeAuthors" }
... ]);
[ { activeAuthors: 1 } ]
```

6. Find the distinct authors from the books collection.

```
schoolDB> db.books.aggregate([
...   { $group: { _id: null, authors: { $addToSet: "$author_name" } } },
...   { $project: { _id: 0, authors: 1 } }
... ]);
[ { authors: [ 'Bob', 'Alice' ] } ]
```

7. Perform a word count on the text field of the documents collection.

```
schoolDB> db.documents.aggregate([
...   {
...     $project: {
...       wordsArray: {
...         $split: [{ $toLower: "$text" }, " "]
...       }
...     }
...   },
...   { $unwind: "$wordsArray" },
...   {
...     $group: {
...       _id: "$wordsArray",
...       occurrences: { $sum: 1 }
...     }
...   },
...   { $sort: { occurrences: -1 } }
... ]);
[
  { _id: 'is', occurrences: 2 },
  { _id: 'mongodb', occurrences: 2 },
  { _id: 'aggregation', occurrences: 1 },
  { _id: 'a', occurrences: 1 },
  { _id: 'powerful', occurrences: 1 },
  { _id: 'large', occurrences: 1 },
  { _id: 'frameworks', occurrences: 1 },
  { _id: 'and', occurrences: 1 },
  { _id: 'for', occurrences: 1 },
  { _id: 'tool', occurrences: 1 },
  { _id: 'engaging.', occurrences: 1 },
  { _id: 'provide', occurrences: 1 },
  { _id: 'learning', occurrences: 1 },
  { _id: 'fun', occurrences: 1 },
  { _id: 'in', occurrences: 1 },
  { _id: 'data', occurrences: 1 },
  { _id: 'sets.', occurrences: 1 },
  { _id: 'mapreduce', occurrences: 1 },
  { _id: 'flexibility.', occurrences: 1 }
]
```

8. Perform a MapReduce operation to count the number of active books for each author.

```
schoolDB> var mapFunction = function() {
...     if (this.status === "active") {
...       emit(this.author_name, 1);
...     }
... };

schoolDB> var reduceFunction = function(key, values) {
...     return Array.sum(values);
... };

schoolDB> db.books.mapReduce(
...     mapFunction,
...     reduceFunction,
...     {
...       out: "active_books_count"
...     }
... );
{ result: 'active_books_count', ok: 1 }
schoolDB> db.active_books_count.find();
[ { _id: 'Alice', value: 2 } ]
```

9. In the documents collection, perform a word count across both the text and title fields, and count how often each word appears. Group the words based on their presence in either text or title.

```
schoolDB> db.documents.aggregate([
...   {
...     $project: {
...       titleWords: {
...         $split: [{ $toLower: "$title" }, " "]
...       },
...       textWords: {
...         $split: [{ $toLower: "$text" }, " "]
...       }
...     }
...   },
...   {
...     $project: {
...       wordSources: {
...         $concatArrays: [
...           {
...             $map: {
...               input: "$titleWords",
...               as: "word",
...               in: { word: "$$word", source: "title" }
...             }
...           },
...           {
...             $map: {
...               input: "$textWords",
...               as: "word",
...               in: { word: "$$word", source: "text" }
...             }
...           }
...         ]
...       }
...     }
...   },
... { $unwind: "$wordSources" },
...   {
...     $group: {
...       _id: {
...         word: "$wordSources.word",
...         source: "$wordSources.source"
...       },
...       count: { $sum: 1 }
...     }
...   },
...   {
...     $group: {
...       _id: "$_id.word",
...       sources: {
...         $push: {
...           source: "$_id.source",
...           count: "$count"
...         }
...       },
...       totalOccurrences: { $sum: "$count" }
...     }
...   },
```

```
...   },
...   { $sort: { totalOccurrences: -1 } }
... ]);
[
  {
    _id: 'mongodb',
    sources: [ { source: 'title', count: 1 }, { source: 'text', count: 2 } ],
    totalOccurrences: 3
  },
  {
    _id: 'mapreduce',
    sources: [ { source: 'text', count: 1 }, { source: 'title', count: 1 } ],
    totalOccurrences: 2
  },
  {
    _id: 'is',
    sources: [ { source: 'text', count: 2 } ],
    totalOccurrences: 2
  },
  {
    _id: 'frameworks',
    sources: [ { source: 'title', count: 1 }, { source: 'text', count: 1 } ],
    totalOccurrences: 2
  },
  {
    _id: 'aggregation',
    sources: [ { source: 'text', count: 1 }, { source: 'title', count: 1 } ],
    totalOccurrences: 2
  },
  {
    _id: 'powerful',
    sources: [ { source: 'text', count: 1 } ],
    totalOccurrences: 1
  },
  {
    _id: 'sets.',
    sources: [ { source: 'text', count: 1 } ],
    totalOccurrences: 1
  },
  {
    _id: 'understanding',
    sources: [ { source: 'title', count: 1 } ],
    totalOccurrences: 1
  },
  {
```

```
  _id: 'understanding',
  sources: [ { source: 'title', count: 1 } ],
  totalOccurrences: 1
},
{
  _id: 'a',
  sources: [ { source: 'text', count: 1 } ],
  totalOccurrences: 1
},
{
  _id: 'and',
  sources: [ { source: 'text', count: 1 } ],
  totalOccurrences: 1
},
{
  _id: 'large',
  sources: [ { source: 'text', count: 1 } ],
  totalOccurrences: 1
},
{
  _id: 'fun',
  sources: [ { source: 'text', count: 1 } ],
  totalOccurrences: 1
},
{
  _id: 'learning',
  sources: [ { source: 'text', count: 1 } ],
  totalOccurrences: 1
},
{
  _id: 'for',
  sources: [ { source: 'text', count: 1 } ],
  totalOccurrences: 1
},
{
  _id: 'tool',
  sources: [ { source: 'text', count: 1 } ],
  totalOccurrences: 1
},
{
  _id: 'basics',
  sources: [ { source: 'title', count: 1 } ],
  totalOccurrences: 1
},
{
```

```
  {
    _id: 'basics',
    sources: [ { source: 'title', count: 1 } ],
    totalOccurrences: 1
  },
  {
    _id: 'engaging.',
    sources: [ { source: 'text', count: 1 } ],
    totalOccurrences: 1
  },
  {
    _id: 'in',
    sources: [ { source: 'text', count: 1 } ],
    totalOccurrences: 1
  },
  {
    _id: 'provide',
    sources: [ { source: 'text', count: 1 } ],
    totalOccurrences: 1
  },
  {
    _id: 'flexibility.',
    sources: [ { source: 'text', count: 1 } ],
    totalOccurrences: 1
  }
]
Type "it" for more
schoolDB> it
[
  {
    _id: 'data',
    sources: [ { source: 'text', count: 1 } ],
    totalOccurrences: 1
  }
]
```

10. In the books collection, group the books by author_name and status (active/inactive) and calculate the total number of books each author has written and the sum of price for the books.

```
schoolDB> db.books.aggregate([
...   {
...     $group: {
...       _id: {
...         author: "$author_name",
...         status: "$status"
...       },
...       numberOfBooks: { $sum: 1 },
...       totalPrice: { $sum: "$price" }
...     }
...   },
...   {
...     $project: {
...       _id: 0,
...       author: "$_id.author",
...       status: "$_id.status",
...       numberOfBooks: 1,
...       totalPrice: 1
...     }
...   },
...   { $sort: { author: 1 } }
... ]);
[
  {
    numberOfBooks: 2,
    totalPrice: 54.98,
    author: 'Alice',
    status: 'active'
  },
  {
    numberOfBooks: 1,
    totalPrice: 39.99,
    author: 'Bob',
    status: 'inactive'
  }
]
```

11. In the students collection, group student marks by Name and Subject, and calculate the total marks and the number of entries for each subject per student.

```
schoolDB> db.students.aggregate([
...   {
...     $group: {
...       _id: {
...         student: "$Name",
...         subject: "$Subject"
...       },
...       totalMarks: { $sum: "$Marks" },
...       entryCount: { $sum: 1 }
...     }
...   },
...   {
...     $project: {
...       _id: 0,
...       student: "$_id.student",
...       subject: "$_id.subject",
...       totalMarks: 1,
...       entryCount: 1
...     }
...   },
...   { $sort: { student: 1, subject: 1 } }
... ]);
[
  {
    totalMarks: 160,
    entryCount: 2,
    student: 'Emily',
    subject: 'Science'
  },
  { totalMarks: 175, entryCount: 2, student: 'John', subject: 'Math' }
]
```

12. In the sales collection, calculate a 3-day moving average of the total_sales for each product. You need to emit sales values by date and calculate a rolling average of the last 3 sales days.

```
schoolDB> db.sales.aggregate([
...   {
...     $addFields: {
...       saleDate: {
...         $dateFromString: { dateString: "$date", format: "%Y-%m-%d" }
...       },
...       totalSaleAmount: {
...         $multiply: ["$quantity", "$price"]
...       }
...     }
...   },
...   {
...     $sort: { product_id: 1, saleDate: 1 }
...   },
...   {
...     $setWindowFields: {
...       partitionBy: "$product_id",
...       sortBy: { saleDate: 1 },
...       output: {
...         threeDayMovingAvg: {
...           $avg: "$totalSaleAmount",
...           window: { documents: [-2, 0] }
...         }
...       }
...     }
...   },
...   {
...     $project: {
...       _id: 0,
...       product_id: 1,
...       date: "$saleDate",
...       movingAverage: "$threeDayMovingAvg"
...     }
...   }
... ]);
```

```
[
  {
    product_id: 'A123',
    date: ISODate('2024-10-01T00:00:00.000Z'),
    movingAverage: 159.9
  },
  {
    product_id: 'A123',
    date: ISODate('2024-10-02T00:00:00.000Z'),
    movingAverage: 119.92500000000001
  },
  {
    product_id: 'A123',
    date: ISODate('2024-10-03T00:00:00.000Z'),
    movingAverage: 122.58999999999999
  },
  {
    product_id: 'A123',
    date: ISODate('2024-10-04T00:00:00.000Z'),
    movingAverage: 106.60000000000001
  },
  {
    product_id: 'B456',
    date: ISODate('2024-10-01T00:00:00.000Z'),
    movingAverage: 199.8
  },
  {
    product_id: 'B456',
    date: ISODate('2024-10-02T00:00:00.000Z'),
    movingAverage: 174.825
  },
  {
    product_id: 'B456',
    date: ISODate('2024-10-03T00:00:00.000Z'),
    movingAverage: 199.79999999999998
  },
  {
    product_id: 'B456',
    date: ISODate('2024-10-04T00:00:00.000Z'),
    movingAverage: 166.5
  }
]
```

13. In the sales collection, group sales by category and calculate the total revenue, total profit, and the average profit margin for each category.

```
schoolDB> db.sales.aggregate([
...   {
...     $addFields: {
...       revenue: { $multiply: ["$quantity", "$price"] },
...       profit: {
...         $multiply: ["$quantity", { $subtract: ["$price", "$cost"] }]
...       },
...       profitMargin: {
...         $cond: {
...           if: { $eq: ["$price", 0] },
...           then: 0,
...           else: { $divide: [{ $subtract: ["$price", "$cost"] }, "$price"] }
...         }
...       }
...     }
...   },
...   {
...     $group: {
...       _id: "$category",
...       totalRevenue: { $sum: "$revenue" },
...       totalProfit: { $sum: "$profit" },
...       averageProfitMargin: { $avg: "$profitMargin" }
...     }
...   },
...   {
...     $sort: { totalRevenue: -1 }
...   }
... ]);
[
  {
    _id: 'Clothing',
    totalRevenue: 699.3,
    totalProfit: 349.3,
    averageProfitMargin: 0.4994994994994995
  },
  {
    _id: 'Electronics',
    totalRevenue: 479.70000000000005,
    totalProfit: 179.70000000000002,
    averageProfitMargin: 0.3746091307066917
  }
]
```

14. In the students collection, compute the average marks for each student and subject. Group the data by student and subject, and return the average marks for each.

```
schoolDB> db.students.aggregate([
...   {
...     $group: {
...       _id: {
...         studentName: "$Name",
...         subject: "$Subject"
...       },
...       averageMarks: { $avg: "$Marks" }
...     }
...   },
...   {
...     $project: {
...       _id: 0,
...       student: "$_id.studentName",
...       subject: "$_id.subject",
...       averageMarks: 1
...     }
...   },
...   { $sort: { student: 1, subject: 1 } }
... ]);
[
  { averageMarks: 80, student: 'Emily', subject: 'Science' },
  { averageMarks: 87.5, student: 'John', subject: 'Math' }
]
```