Ministry of Higher Education and Scientific Research
University of Abdelhafid Boussouf - Mila
Institute of Mathematics and Computer Science
Department of Computer Science
Master 2 I2A – Big Data
2025/2026

Directed Works TD 8 – Aggregation Framework & Indexing Strategies

Exercise 1: The Pipeline Logic (Mental Compilation)

Aggregation is a "Conveyor Belt". Documents enter Stage 1, get transformed, and pass to Stage 2.

Input Data (Collection: `orders`):

```
{ "_id": 1, "client": "Amine", "status": "completed", "items": ["Laptop", "Mouse"], "total": 60000 }
{ "_id": 2, "client": "Sara",  "status": "pending",   "items": ["Tablet"],          "total": 30000 }
{ "_id": 3, "client": "Omar",  "status": "completed", "items": ["Laptop", "HDMI"], "total": 70000 }
```

The Pipeline:

```
db.orders.aggregate([
  // Stage 1
  { $match: { status: "completed" } },

  // Stage 2
  { $unwind: "$items" },

  // Stage 3
  { $group: {
      _id: "$items",
      count: { $sum: 1 },
      avg_order_val: { $avg: "$total" }
    }
  },
  // Stage 4
  { $sort: { count: -1 } }
])
```

Questions:

1. **Filtering:** Which document(s) are discarded at **Stage 1**?
2. **Explosion:** How many documents exist in memory immediately after **Stage 2**? List them roughly (e.g., `{item: "Laptop", ...}`).
3. **Grouping:** What is the final output of this aggregation? Draw the result table.
4. **Logical Analysis:** Notice we calculated `avg_order_val`. Does this metric represent the "Average Price of the Item" or something else? Is it a useful metric here?

Exercise 2: Constructing Advanced Pipelines

Using the **TechStore** database schema, construct pipelines for the following scenarios.

1. Analytics: Monthly Revenue Report We need a report showing sales trends.

- **Requirement:** Group orders by **Year-Month** (e.g., "2024-01") and calculate the total revenue.
- *Hint:* Use `$project` first to create a formatted date string, or use a composite `_id` in `$group`: `_id: { year: { $year: "$date" }, month: { $month: "$date" } }`.

2. Segmentation: Customer Value Categories We want to label our customers based on their spending.

- **Requirement:** For each order, project a new field `category`.
- If `total > 50000`, label is "Premium", Otherwise, label is "Standard".

- *Hint:* Use the `$cond` (Conditional) operator: `{ $cond: { if: { $gt: [...] }, then: "...", else: "..." } }`.

3. Joins: Average Spending by City This requires data from two collections: `users` (contains `address.city`) and `orders` (contains `total_amount`).

- **Requirement:** Calculate the average order total for each **City**.
- *Challenge:* You start from `orders`. You must `$lookup` the user to get their city, `$unwind` the user array (because lookup returns an array), and then `$group` by the city field.

Exercise 3: Indexing & The ESR Rule

We run this query 1,000 times per minute:

```
db.products.find({
  "category": "Laptops",
  "price": { $gt: 50000 }
}).sort({ "price": -1 })
```

Part A: Selecting the Best Index Analyze these three index candidates using the ESR Rule (Equality, Sort, Range).

1. { "price": 1 }
2. { "category": 1, "price": -1 }
3. { "price": -1, "category": 1 }

Question: Which index is optimal? Walk through the ESR logic to justify your answer.

Part B: The "Covered Query" A "Covered Query" is the holy grail of performance: the database answers using *only* the Index (RAM), never touching the Documents (Disk).

Question: If we have the optimal index from Part A, how must we modify the `.find()` command above to make it a **Covered Query**? (*Hint:* Think about the `projection` argument. What fields are in the index? What fields are returned by default?)

Exercise 4: Schema Design & The 16MB Limit

Scenario: Product Reviews. We decide to embed reviews inside the Product document to avoid Joins.

```
{
  "_id": "prod_1",
  "name": "Gaming Laptop",
  "reviews": [
    // Each review object is approx 500 bytes
    { "user": "Ali", "text": "Excellent...", "rating": 5, "date": ... },
    ...
  ]
}
```

Questions:

1. Do the Math: If each review is 500 bytes (0.5 KB), and a MongoDB document has a hard limit of 16 MB, what is the maximum number of reviews a single product can store?
- *Calculation:* `16,000,000 bytes / 500 bytes = ?`
2. Failure Scenario: What happens when the 32,001st customer tries to post a review? Does the database crash? Does the insert fail?
3. The Fix (Subset Pattern): Propose a schema that keeps the "Top 5 Reviews" embedded (for fast loading) but stores the rest indefinitely without hitting the limit.