MongoDB Library Collection: Detailed Q&A

Q1. Insert 5 Documents into library Collection

Task:
Insert five student documents into a collection named 'library', where each document represents a student and (optionally) the book they have borrowed.

Operations:

```
db.library.insertOne({
  student_id: "S1",
  name: "Anu",
  department: "CS",
  issued: "Y",
  book: { title: "Python Basics", author: "James", price: 250, currency: "INR" },
  borrow_days: 10
});

db.library.insertOne({
  student_id: "S2",
  name: "Ravi",
  department: "IT",
  issued: "N",
  book: { title: "Node.js Guide", author: "Alex", price: 300, currency: "USD" },
  borrow_days: 0
});

db.library.insertOne({
  student_id: "S3",
  name: "Megha",
  department: "CS",
  issued: "Y",
  book: { title: "Java Fundamentals", author: "Sam", price: 180, currency: "INR" },
  borrow_days: 7
});

db.library.insertOne({
  student_id: "S4",
  name: "Kiran",
  department: "EC",
  issued: "Y",
  book: { title: "DBMS", author: "Joseph", price: 150, currency: "INR" },
  borrow_days: 5
});
```

```
db.library.insertOne({
  student_id: "S5",
  name: "Divya",
  department: "IT",
  issued: "N",
  book: { title: "C++ Programming", author: "Steve", price: 275, currency: "USD" },
  borrow_days: 0
});
```

Explanation:
Each insertOne call creates (if needed) the 'library' collection and inserts one document. Fields include student_id, name, department, issued flag (Y/N), an embedded book sub-document, and borrow_days.
After these calls, the library collection holds exactly five documents.

Q2. Retrieve All Documents Where Book Title Contains "Java"

Query:
```
db.library.find({ "book.title": /Java/ });
```

Explanation:
- "book.title": /Java/ uses a regular expression to match any document where the book sub-document's title contains 'Java'.
- Matches examples like 'Java Fundamentals', 'Advanced Java Programming', etc.
- Result (from our inserted data) returns the document where student_id is 'S3'.

Q3. Retrieve All Documents Where borrow_days > 5

Query:
```
db.library.find({ borrow_days: { $gt: 5 } });
```

Explanation:
- $gt means 'greater than'.
- Filters documents where borrow_days is strictly greater than 5.
- In our data, that matches student_id 'S1' (borrow_days 10) and 'S3' (borrow_days 7).

Q4. Retrieve All Documents Where book.author Is "James" or "Sam"

Query:
```
db.library.find({ "book.author": { $in: ["James", "Sam"] } });
```

Explanation:
- $in checks if book.author is any value in the array ["James", "Sam"].
- Equivalent to SQL: author = 'James' OR author = 'Sam'.
- Matches inserted documents for 'S1' (author James) and 'S3' (author Sam).

Q5. Retrieve All Students Whose book.price Is Between 150 and 300 (Inclusive)

Query:
db.library.find({ "book.price": { $gte: 150, $lte: 300 } });

Explanation:
- $gte means 'greater than or equal to' and $lte means 'less than or equal to'.
- Filters documents where 150 <= book.price <= 300.
- In our data, every book price (250, 300, 180, 150, 275) lies between 150 and 300. All five documents match.

Q6. Retrieve Students Who Have issued: "Y" and department Is Not "CS"

Query:
```
db.library.find({
  issued: "Y",
  department: { $ne: "CS" }
});
```

Explanation:
- issued: "Y" filters students who have borrowed a book.
- department: { $ne: "CS" } means department is not equal to 'CS'.
- Among inserted docs, only student_id 'S4' (department EC, issued Y) meets both criteria.

Q7. Retrieve All Students Whose book.title Does Not Contain "Programming"

Query:
db.library.find({ "book.title": { $not: /Programming/ } });

Explanation:
- $not: /Programming/ matches titles that do NOT contain the substring 'Programming'.
- Checks each title: 'Python Basics', 'Node.js Guide', 'Java Fundamentals', 'DBMS' (all match). 'C++ Programming' does not match and is excluded.
- Result returns four documents (all except the one with 'C++ Programming').