Data 101/CS 187: Data Engineering Fall 2026 Discussion 8 – October 30th, 2026

Semi-Structured Data, MongoDB

Instructor: Aditya Parameswaran *Covers Lecture(s): 16–18*

I Semi-Structured Data [Optional]

1. Use JSON to represent the following two tables. The `players` table has a foreign key on the `team` column referencing the `name` column in the `teams` table. Use this to connect the two tables.

teams

name	state
Warriors	CA
Celtics	MA
Lakers	CA

players

name	age	team	position
Stephen Curry	34	Warriors	point guard
Draymond Green	32	Warriors	forward
Jaylen Brown	25	Celtics	forward
LeBron James	37	Lakers	forward

```
{
    "teams": [
        {
            "name": "Warriors",
            "state": "CA",
            "players": [
                {
                    "name": "Stephen Curry",
                    "age": 34,
                    "position": "point guard"
                },
                {
                    "name": "Draymond Green",
                    "age": 32,
                    "position": "forward"
                }
            ]
        },
        {
            "name": "Celtics",
            "state": "MA",
            "players": [
                {
                    "name": "Jaylen Brown",
                    "age": 25,
                    "position": "forward"
                }
            ]
        },
        {
            "name": "Lakers",
            "state": "CA",
            "players": [
                {
```

```
                    "name": "LeBron James",
                    "age": 37,
                    "position": "forward"
                }
            ]
        }
    ]
}
```

2. When transforming JSON to rectangular data, what value should use for missing attributes?

> We can represent missing values with NULL.

II MongoDB Concepts

3. For each of the following concept in a relational DBMS, what is the corresponding concept in MongoDB?
   (a) Database. database
   (b) Relation. collection
   (c) Row. document
   (d) Column. field

III MQL Queries – Retrieval

```
paintings: {
    "_id": "ObjectId("a1a2b3r4t6x7s7")",
    "paintingId": 2134803,
    "paintingName": "Girl with a Pearl Earring",
    "artistId": "Johannes Vermeer",
    "year": 1665,
    "purchasePrice": 30000000,
    "timePeriod": "Baroque",
    "status": "unappreciated",
    "mediums": [
        "oil",
        "canvas"
    ]
}
```

```
artists: {
    "_id": "ObjectId("a1a2b3r4t6x7s7")",
    "artistId": "Johannes Vermeer",
    "numberOfWorks": 34,
    "backgroundInfo": {
        "birthYear": 1632,
        "country": "Netherlands",
        "style": "Baroque",
        "mentor": null
    }
}
```

Given the **first row** of the MongoDB collections above, provide the corresponding MQL query for each of the following scenarios.

4. Write a query to find all the paintings in the database.

> `db.paintings.find()`
> OR:
> `db.paintings.find({})`

5. Write a query to find all the paintings painted by Paul Cezanne.

```
db.paintings.find({"artistId":"Paul Cezanne"})
```

6. Write a query to find the artists who were born later than the year 1700.

```
db.artists.find({"backgroundInfo.birthYear": {$gt:1700} })
```

7. Write a query to find the paintings that were painted on canvas.

```
db.paintings.find({"mediums": "canvas"})
OR
db.paintings.find({"mediums": {$elemMatch: {$eq: "canvas"}} })
```

8. Write a query to find the paintings from a time period of either the Baroque, Romantic, or Impressionist age.

```
db.paintings.find({$or: [{"timePeriod": "Baroque"}, {"timePeriod":
"Romantic"}, {"timePeriod": "Impressionist"}]})
OR
db.paintings.find({"timePeriod": {$in: ["Baroque", "Romantic",
"Impressionist"]}})
```

IV MQL Queries – Aggregation

```
paintings: {
    "_id": "ObjectId("a1a2b3r4t6x7s7")",
    "paintingId": 2134803,
    "paintingName": "Girl with a Pearl Earring",
    "artistId": "Johannes Vermeer",
    "year": 1665,
    "purchasePrice": 30000000,
    "timePeriod": "Baroque",
    "status": "unappreciated",
    "mediums": [
        "oil",
        "canvas"
    ]
}
```

```
artists: {
    "_id": "ObjectId("a1a2b3r4t6x7s7")",
    "artistId": "Johannes Vermeer",
    "numberOfWorks": 34,
    "backgroundInfo": {
        "birthYear": 1632,
        "country": "Netherlands",
        "style": "Baroque",
        "mentor": null
    }
}
```

Let's revisit the `paintings` and `artists` collections from last week's discussion. Given the **first row** of the MongoDB collections above, provide the corresponding MQL query for each of the following scenarios.

9. Find the average painting purchase price per artist.

```
db.paintings.aggregate([
    {$group: {_id: "$artistId", avgPurchasePrice: {$avg: "$purchasePrice"}} }
])
```

10. Find the artists of paintings with a combined purchase price of at least $100 million and sort by descending order.

```
db.paintings.aggregate([
    {$group: {_id: "$artistId", totalPurchasePrice: {$sum: "$purchasePrice"}}
},
    {$match: {totalPurchasePrice: {$gte: 100000000}} },
    {$sort: {totalPurchasePrice: -1}}
])
```

11. For each artist, find the name of the first painting they created based on the year of the paintings.

```
db.paintings.aggregate([
    {$sort: {year: 1}},
    {$group: { _id: "$artistId", firstPaintingName: {$first: "$paintingName"}
} }
])
```

12. **Challenge:** Write a query to count the number of paintings for each medium and provide the average purchase price of paintings for each medium. Also, showcase the oldest painting's year for each medium.

```
db.paintings.aggregate([
    {$unwind: "$mediums"},
    {$group: { _id: "$mediums",
        count: {$sum: 1},
        avgPurchasePrice: {$avg: "$purchasePrice"},
        oldestYear: {$min: "$year"} } },
    {$project: { _id: 0,
        medium: "$_id",
        count: 1,
        avgPurchasePrice: 1,
        oldestYear: 1 } }
])
```

V MQL Queries – Lookup

13. Find all paintings that were created by artists from the Netherlands, and project the painting name and the corresponding country.

```
db.paintings.aggregate([
    {$lookup: {
        from: "artists",
        localField: "artistId",
        foreignField: "artistId",
        as: "artistInfo"
    } },
    {$unwind: "$artistInfo" },
    {$match: { "artistInfo.backgroundInfo.country": "Netherlands" } },
    {$project: { _id: 0,
        paintingName: 1,
        artistCountry: "$artistInfo.backgroundInfo.country"
    } }
])
```

VI MQL Queries – Update

14. Update the purchase price of a painting to be increased by $1 million if the current purchase price is less then $5 million to account for appreciating value over time, and also set the status of these paintings to "appreciated".

```
db.paintings.updateMany(
   {"purchasePrice":  {$lt:  5000000}},
   {
      $inc:  {"purchasePrice":  1000000},
      $set:  {"status":  "appreciated"}
   }
)
```