1. Embed vs Reference?

Embed for bounded one-to-few data read together. Reference for large/unbounded or shared datasets.

---

2. How to choose shard key?

High cardinality, evenly distributed, frequently queried, avoid monotonic growth.

---

3. What causes hot shards?

Sequential shard keys like timestamps or auto-increment IDs.

---

4. Replica set purpose?

High availability via automatic failover and replication.

---

5. Write concern levels?

w:1, majority, all – tradeoff between latency and durability.

---

6. Read concern levels?

local, majority, linearizable – tradeoff between speed and consistency.

---

7. How MongoDB handles failover?

Automatic election promotes a secondary to primary.

8. Aggregation framework?

Pipeline-based transformation and analytics engine.

---

9. Compound index rule?

Follows prefix rule; order of fields matters.

---

10. TTL index usage?

Auto-delete expired documents like sessions/logs.

---

11. Capped collection?

Fixed-size collection for logs/events.

---

12. Avoid collection scan?

Create proper indexes and use projection.

---

13. Transactions in MongoDB?

Multi-document ACID transactions in replica sets/sharded clusters.

---

14. Replication lag causes?

Heavy writes, slow disks, network delay.

---

15. Schema evolution?

Flexible schema; gradual app-level rollout.

---

16. Backup strategy?

Snapshots, mongodump, point-in-time recovery.

---

17. Multi-tenant design?

tenant_id with compound indexes or separate DB per tenant.

---

18. Write amplification?

Data + index + journal + replication writes.

---

19. Avoid large documents?

Keep under 16MB; avoid excessive embedding.

---

20. Index types?

Single, compound, multikey, text, geospatial, hashed, TTL.

21. Hashed shard key benefit?

Prevents hot partitions.

---

22. Aggregation vs MapReduce?

Aggregation faster; MapReduce for custom logic.

---

23. Performance monitoring?

Slow query log, CPU, memory, replication lag.

---

24. Prevent hot partitions?

Use hashed keys or distribute writes.

---

25. Design audit logging?

Append-only collection, partition by time.

---

26. Horizontal scaling?

Add shards and rebalance chunks.

---

27. Data archiving strategy?

Move old data to cold storage collections.

---

28. When not to use MongoDB?

Complex relational joins & strict ACID financial workloads.

---

29. Highly available architecture?

Sharded cluster + replica sets + backups.

---

30. Large e-commerce design?

Shard by user/order, index product/user fields, cache catalog.

Difficult MongoDB Aggregation & Query Examples

1. Top 5 products by revenue

```
db.orders.aggregate([
  { $unwind: "$items" },
  { $group: { _id: "$items.productId",
              revenue: { $sum: { $multiply: ["$items.price", "$items.qty"] } } } },
  { $sort: { revenue: -1 } },
  { $limit: 5 }
])
```

2. Users with more than 10 orders

```
db.orders.aggregate([
  { $group: { _id: "$userId", orderCount: { $sum: 1 } } },
  { $match: { orderCount: { $gt: 10 } } }
])
```

3. Monthly revenue report

```
db.orders.aggregate([
  { $group: {
      _id: { month: { $month: "$createdAt" } },
      totalRevenue: { $sum: "$amount" }
  }}
])
```

4. Find duplicate emails

```
db.users.aggregate([
  { $group: { _id: "$email", count: { $sum: 1 } } },
  { $match: { count: { $gt: 1 } } }
])
```

5. Rolling average order value

```
db.orders.aggregate([
  { $setWindowFields: {
      partitionBy: "$userId",
      sortBy: { createdAt: 1 },
      output: {
        rollingAvg: { $avg: "$amount",
                      window: { documents: [-5, 0] } }
      }
  }}
])
```

6. Lookup (Join) example

```
db.orders.aggregate([
  { $lookup: {
      from: "users",
      localField: "userId",
      foreignField: "_id",
      as: "userDetails"
  }}
])
```

7. Count active users last 7 days

```
db.users.countDocuments({
  lastLogin: { $gte: new Date(Date.now() - 7*24*60*60*1000) }
})
```

8. Text search query

```
db.products.find(
  { $text: { $search: "wireless headphones" } },
  { score: { $meta: "textScore" } }
).sort({ score: { $meta: "textScore" } })
```

9. Geo query example

```
db.stores.find({
  location: {
    $near: {
      $geometry: { type: "Point", coordinates: [77.5946, 12.9716] },
      $maxDistance: 5000
    }
  }
})
```

10. Delete old logs using TTL alternative

```
db.logs.deleteMany({
  createdAt: { $lt: new Date(Date.now() - 90*24*60*60*1000) }
})
```