UNIVERSITY OF MSILA
COMPUTER SCIENCE DEPARTMENT

Master 1 AI
Big Data
Final Exam Correction

18/01/2026
2 H

Student Information	Final Mark
First Name:	
Last Name:	
Group:	

Exam Instructions

- Smartphones and smartwatches must be **OFF** and put away.
- Read the case study carefully before answering.
- each correct answer= 0.5 / except for part A Q 19, 20, 21, 22 =0.25
-

Case Study: NovaRetail Analytics Platform

NovaRetail is a fast-growing multi-channel data sources retail company operating physical stores and an international e-commerce platform. Every day, the company generates large volumes of data from point-of-sale systems, online transactions, mobile applications, customer feedback, marketing campaigns, and IoT sensors deployed in warehouses.

Initially, NovaRetail relied on a traditional relational data warehouse for reporting and analytics. As the company expanded, data arrival rates became unpredictable, and peak workloads increasingly overloaded the system. Batch processing jobs, real-time dashboards, and ad-hoc analytical queries had to run on the same infrastructure, often competing for shared resources.

To address these challenges, NovaRetail transitioned to a distributed big data architecture. Raw data is ingested continuously and stored before transformation, allowing future recomputation if analytical logic changes. Different teams analysts, data scientists, and operational systems access the same datasets but with different performance and latency requirements.

The platform emphasizes horizontal scalability, fault tolerance through task re-execution, and flexible data models. User profiles and session data are stored in a document-oriented NoSQL database to support low-latency access and evolving schemas. The system prioritizes availability and scalability, even during partial network failures.

Part A

1. What characteristic of NovaRetail's workload caused the traditional warehouse to become insufficient?
   - ☐ Strict transactional requirements
   - ☐ Stable and predictable query patterns
   - ■ **Sudden workload spikes and variable ingestion rates**
   - ☐ Limited storage capacity

2. Why does NovaRetail retain raw data before applying transformations?
   - ☐ To reduce ingestion latency
   - ☐ To enforce strict data validation
   - ■ **To allow future reprocessing when requirements evolve**
   - ☐ To minimize storage usage

3. Running dashboards and historical analytics on the same data platform mainly requires careful management of:
   - ☐ Data compression formats
   - ■ **Resource scheduling and sharing**
   - ☐ Network topology
   - ☐ Index structures

4. What risk emerges when multiple teams query shared datasets simultaneously?
   - ☐ Data inconsistency
   - ■ **Resource contention affecting performance**
   - ☐ Schema incompatibility
   - ☐ Excessive replication

5. Recomputing analytical results reliably depends primarily on:
   - ☐ Strong transactional guarantees
   - ■ **Deterministic processing logic**
   - ☐ Centralized execution control
   - ☐ High replication factors

6. How does the system remain resilient to hardware failures?
   - ☐ By preventing node failures
   - ☐ By duplicating all computations
   - ■ **By re-executing failed tasks transparently**
   - ☐ By enforcing strict consistency

7. Compared to the old warehouse, the new platform accepts greater variability in:
   - ☐ Data correctness
   - ■ **Query response times**
   - ☐ Security enforcement
   - ☐ Storage costs

8. Near-real-time dashboards mainly constrain:
   - ☐ Data modeling choices
   - ■ **Delay between data arrival and visibility**
   - ☐ Storage redundancy
   - ☐ Schema complexity

9. Why can a shared data repository become problematic?
   - ☐ Data replication increases
   - ☐ Metadata is centralized
   - ■ **Different workloads have conflicting latency needs**
   - ☐ Schemas are flexible

10. Traditional data warehouses are best suited when:
    - ☐ Workloads are elastic
    - ☐ Schemas change frequently
    - ■ **Source structures are stable over time**
    - ☐ Data arrives continuously

11. When analytics logic evolves frequently, which capability becomes essential?
    - ☐ Query optimization
    - ■ **Data lineage tracking**
    - ☐ Compression efficiency
    - ☐ Access control

12. Why is delaying schema design useful for exploratory analytics?
    - ☐ It reduces storage costs
    - ■ **Data meaning is not fully known in advance**
    - ☐ Queries become simpler
    - ☐ Indexing is unnecessary

13. Which workload is least tolerant of delays caused by shared execution resources?

- ■ **Scheduled reporting**
- ☐ Ad-hoc analytical exploration
- ☐ Archival storage
- ☐ Historical backfills

14. Combining structured and semi-structured data mainly increases complexity at which level?
    - ☐ Network routing
    - ■ **Logical interpretation of data**
    - ☐ Disk allocation
    - ☐ CPU scheduling

15. Why is horizontal scaling preferred in NovaRetail's architecture?
    - ☐ It eliminates failures
    - ■ **It aligns costs with gradual growth**
    - ☐ It guarantees low latency
    - ☐ It simplifies schemas

16. Separating ingestion from analytics processing allows teams to:
    - ☐ Share execution engines
    - ■ **Evolve pipelines independently**
    - ☐ Enforce stronger consistency
    - ☐ Reduce data volume

17. Why is a document database suitable for user profiles?
    - ☐ Queries are analytical
    - ■ **User attributes vary across individuals**
    - ☐ Data is rarely updated
    - ☐ Joins are required

18. Prioritizing low-latency access for active sessions often requires relaxing:
    - ■ **Immediate durability guarantees**
    - ☐ Replication
    - ☐ Query complexity
    - ☐ Metadata accuracy

19. Why is NovaRetail's NoSQL system designed to remain available during network issues?
    - ☐ Hardware upgrades are frequent
    - ■ **Temporary node isolation can occur**
    - ☐ Queries are complex
    - ☐ Schemas change often

20. Session data accessed using a known identifier follows which access pattern?
    - ☐ Full dataset scans
    - ■ **Direct record lookup**
    - ☐ Long-term aggregation
    - ☐ Cross-entity joins

21. Storing related data together to avoid joins mainly increases the importance of:
    - ☐ Network bandwidth
    - ■ **Careful document design**
    - ☐ Query planners
    - ☐ Replication strategies

22. When multiple teams reuse the same datasets, what ensures consistent understanding?
    - ☐ Physical separation
    - ■ **Clear metadata definitions**
    - ☐ Higher replication
    - ☐ Automatic query rewriting

Part B

1. NovaRetail ingests 2 TB per day; only 35% is used for analytics. Why keep the remaining 65%?
    - ☐ Reduces daily storage growth
    - ■ **Enables future recomputation and new use cases**
    - ☐ Improves ingestion throughput
    - ☐ Guarantees higher data quality

2. A batch job processes 10 TB in 5 hours on 50 nodes. how much time it takes if 100 nodes are used?
    - ☐ 5 hours
    - ☐ 4 hours
    - ■ **2.5 hours**
    - ☐ 3.125 hour

3. A near-real-time pipeline introduces 3 seconds per stage; 4 sequential stages. Minimum expected latency?
    - ☐ 3 s
    - ☐ 6 s
    - ■ **12 s**
    - ☐ 15 s

4. Data is replicated 3 times across nodes. If one node fails, what fraction of the data remains available?
   - ☐ One-half
   - ■ **Two-thirds**
   - ☐ Three-quarters
   - ☐ All

5. Query scans 1 TB of data; disk throughput 200 MB/s/node; 10 nodes. Approximate scan time?
   - ☐ 50 s
   - ☐ 100 s
   - ■ **500 s**
   - ☐ 1,000 s

6. Document DB stores sessions averaging 2 KB; 5M sessions. Memory needed to cache all?
   - ☐ 1 GB
   - ☐ 5 GB
   - ■ **10 GB**
   - ☐ 20 GB

7. When write latency grows because of coordination overhead, the best way to reduce latency as the system scales is to:
   - ☐ Increase replication factor
   - ■ **Reduce distributed coordination**
   - ☐ Enforce strict consistency
   - ☐ Normalize the data model

8. A cluster has 90% CPU use during batch jobs and 40% during interactive queries. The best approach to improve overall utilization is:
   - ☐ Static resource partitioning
   - ■ **Dynamically schedule resources based on workload**
   - ☐ Add more nodes
   - ☐ Limit interactive queries

9. A system stores 100 TB of data with a replication factor of 3. If storage costs $20 per TB per month, what is the monthly cost?
   - ☐ $2,000
   - ☐ $3,000
   - ■ **$6,000**
   - ☐ $9,000

10. System tolerates node failures via recomputation instead of replication. Main trade-off?
    - ■ **Compute cost for storage savings**
    - ☐ Latency for availability
    - ☐ Consistency for durability
    - ☐ Throughput for scalability

Part C: Correction

1. **Active users in Paris with exactly two purchases**

```
users.find({
  "city": "Paris",
  "status": "active",
  "purchases": {"$size": 2}
})
```

2. **First purchase for Lyon users whose first purchase ¿ 50**

```
pipeline = [
  {"$match": {"city": "Lyon"}},
  {"$project": {
      "user_id": 1,
      "first_purchase": {"$arrayElemAt": ["$purchases", 0]}
  }},
  {"$match": {"first_purchase": {"$gt": 50}}}
]
users.aggregate(pipeline)
```

3. **Inactive users located in Paris or Lyon**

```
users.find({
  "status": "inactive",
  "city": {"$in": ["Paris", "Lyon"]}
})
```

4. **Total purchase amount for users with more than two purchases**

```
pipeline = [
  {"$project": {
      "user_id": 1,
      "nb_purchases": {"$size": "$purchases"},
      "total_purchase": {"$sum": "$purchases"}
  }},
  {"$match": {"nb_purchases": {"$gt": 2}}}
]
users.aggregate(pipeline)
```

5. **First inactive user in Lyon**

```
users.find_one({
  "status": "inactive",
  "city": "Lyon"
})
```

6. **Users having at least one purchase greater than 200**

```
users.find({
  "purchases": {"$gt": 200}
})
```

7. **Number of active users per city**

```
pipeline = [
  {"$match": {"status": "active"}},
  {"$group": {
      "_id": "$city",
      "count": {"$sum": 1}
  }}
]
users.aggregate(pipeline)
```

8. **Users with at least two purchases strictly greater than 150**

```
pipeline = [
  {"$project": {
      "user_id": 1,
      "high_purchases": {
        "$filter": {
          "input": "$purchases",
          "as": "p",
          "cond": {"$gt": ["$$p", 150]}
        }
      }
  }},
  {"$match": {"high_purchases": {"$size": 2}}}
]
users.aggregate(pipeline)
```

9. **Average total purchase per city (users with purchase ¿ 100)**

```
pipeline = [
  {"$match": {"purchases": {"$gt": 100}}},
  {"$project": {
      "city": 1,
      "total_purchase": {"$sum": "$purchases"}
  }},
  {"$group": {
      "_id": "$city",
      "avg_total_purchase": {"$avg": "$total_purchase"}
  }}
]
users.aggregate(pipeline)
```

10. **Users whose total purchases exceed the global average**

```
pipeline = [
  {"$project": {
      "user_id": 1,
      "total_purchase": {"$sum": "$purchases"}
  }}
]
totals = list(users.aggregate(pipeline))
avg = sum(u["total_purchase"] for u in totals) / len(totals)
[u for u in totals if u["total_purchase"] > avg]
```