University of Saida Dr Moulay Tahar
Faculty of M.I.T
Computer Science Department

Module: G.A.M.D
Promotion: 2^{nd} Year Master AI
Duration: 1h 30

FINAL EXAM

Exercise 1 (5 Points):

1 Short Answer Questions

1.1 Q1: Define Big Data from 2025 Perspective [1 Point]

Big Data = extracting **actionable value** from diverse, high-volume, high-velocity data.

5V Framework (volume alone insufficient):

- **Velocity**: Speed of data processing (real-time vs. batch)
- **Variety**: Multiple data types (70% unstructured: images, video, text, IoT)
- **Veracity**: Data quality & trustworthiness
- **Value**: Business impact (revenue, cost, risk mitigation)

1.2 Q2: Veracity vs. Value [1 Point]

Veracity = Data quality (accurate, reliable, complete)

- Example: Hospital with 20–30% missing patient demographics → inaccurate ML predictions

Value = Business impact (revenue, cost savings, competitive advantage)

- Example: Netflix recommendations = $1B annual customer retention value

1.3 Q3: MongoDB vs. Relational Database [1 Point]

MongoDB is better because:

- **Schema-less**: Store different attributes per user
- **No NULL waste**: Only store what exists
- **No downtime**: Add new attributes without ALTER TABLE
- **Agile**: Perfect for evolving startups

1.4 Q4: NoSQL Database for Financial Institution [1 Point]

Cassandra (primary): Time-series, write-heavy (100M transactions/day), cross-region replication

Redis (cache): Sub-millisecond fraud detection lookups

Snowflake (archive): Regulatory storage (7+ years), compressed

1.5 Q5: Primary Use Case for Graph Databases [1 Point]

Relationship-heavy systems:

- Social networks (friends-of-friends, community detection)
- Recommendation systems (product relationships)
- Fraud detection (suspicious transaction networks)
- Knowledge graphs (semantic relationships)

Key advantage: Index-free adjacency = constant-time graph traversal

2 Multiple Choice Answers

Q#	Answer	Explanation
Q6	**C**	Column-family (Cassandra): time-series IoT data, write-heavy, append-only, optimized for aggregations
Q7	**C**	Predictive analytics: forecasts future customer behavior
Q8	**A & B**	map(), filter() are lazy transformations; Spark builds DAG before execution
Q9	**C**	Apache Flink: 100ms latency (Spark: 1–2s, MapReduce: minutes)
Q10	**C**	BSON/JSON: MongoDB's native document format
Q11	**C**	.collect(): action that brings data to driver; map/filter are lazy

Table 1: Multiple Choice Answers

3 Scenario Exercise 1 [6 Points]

3.1 Part a: NoSQL Database Choice (3 Points)

Primary: Time-Series Database / Column-Family Store (Cassandra or HBase) (70% of transactions)
- **Why**: Transaction data is append-only, write-heavy, and time-indexed
Optimizations:
- Cassandra excels at 100M+ writes/day across EU data centers
- Native replication across regions (ensures EU compliance)
- Optimized for time-series queries (aggregations by hour, day)
- Example: Store as (customer$_id, timestamp, amount, merchant) per time interval$

2. **Secondary: Key-Value Store (Redis)** (real-time caching)
- **Why:** Real-time fraud detection needs sub-millisecond lookups
- **Data:** Cache recent customer transactions, spending patterns, anomaly flags
- **Benefits:** In-memory, sub-millisecond latency for fraud scoring
- **TTL:** Auto-expire old cache entries (24-48 hours)

3. **Archive: Cloud Data Warehouse**
- **Why:** Long-term storage for regulatory reporting (7+ years)
- **Cost-efficient:** Compressed columnar format reduces storage 10x
- **Query:** Run daily regulatory reports without impacting transactional system

Avoid: MongoDB (document database unsuitable for time-series); graph databases (not applicable); MapReduce (batch, not real-time).

3.2 Part b: Hybrid Processing Architecture (2 Points)

Data Flow:

```
Transaction Stream (Kafka)
    |
Real-time: Flink (100ms) -> Redis cache ->
           Fraud rules -> Alerts
    |
Batch: Spark daily (00:00) -> ML retraining ->
       Update Redis
    |
Storage: Cassandra (transactions) ->
         Snowflake (archive)
```

Flink: Real-time fraud scoring (lookup Redis, apply rules)
Spark: Daily batch (aggregations, ML model retraining, regulatory reports)
Cost: Use spot instances for Spark (~70% savings)

3.3 Part c: Cloud Deployment Model (1 Points)

Hybrid Cloud:

- **Private**: On-premises Cassandra
- **Public**: AWS for compute (Kafka MSK, Flink EMR, Spark EMR, Snowflake)

Compliance: EU servers only

4 Scenario Exercise 2 [6 Points]

4.1 Part a: Complete Data Pipeline (2 Points)

Pipeline:

```
Data Sources (EHR, images, notes, labs)
    |
Kafka (ingestion)
    |
Storage: MongoDB (patient records),
         HBase (time-series labs),
         Elasticsearch (text search),
         S3/MinIO (images)
    |
Processing: Spark (aggregation),
            TensorFlow (image analysis),
            NLP (clinical notes)
    |
Output: Risk scores, preventive care
        recommendations, dashboards
```

4.2 Part b: Data Quality Issues (2 Points)

Issue	**Solution**	
Missing values	Imputation, mean/median fill	
Inconsistent formats	ETL standardization	
Unstructured text	NLP preprocessing, entity extraction	
Outliers	Statistical detection (IQR, Z-score)	
Duplicates	Record linkage, deduplication	
Image quality	Quality scoring filters, preprocessing	
Class imbalance	SMOTE oversampling, class weights	

Table 2: Data Quality Issues and Solutions

4.3 Part c: GDPR & Security Measures (2 Points)

Control	**Implementation**	
Encryption	AES-256 at rest, TLS 1.3 in transit	
Access Control	RBAC (role-based), VPC isolation	
De-identification	Hash patient IDs, age ranges, remove dates	
Data Retention	Auto-delete after 7 years	
Right to Erasure	Workflow to delete patient data on request	
Audit Logging	CloudTrail logs for all data access	
Anonymization	k-anonymity ($k \geq 5$ for research data)	
DPA	Data Processing Agreement with cloud providers	

Table 3: GDPR & Security Controls