Module – 1

Real-Time Market Data Ingestion

1. Database Chosen

- **Primary Storage: Apache Kafka + Apache Druid**
- **Complementary Systems:**
    - **Apache Cassandra** for high-availability time-series storage.
    - **Amazon S3 or HDFS** for long-term cold storage.
    - **Elasticsearch** for social media sentiment indexing and analysis.

2. Explanation for the Database Choice

- **Apache Kafka** is used for ingesting and buffering the high-throughput data streams reliably with fault tolerance.
- **Apache Druid** supports sub-second OLAP-style queries on streaming and batch data with real-time ingestion capabilities.
- **Apache Cassandra** offers high write throughput and is a great fit for durable time-series storage with predictable latency.
- **S3/HDFS** supports cost-effective long-term data retention for historical market data needed for backtesting or compliance.
- **Elasticsearch** is used for indexing and querying semi-structured social sentiment data at scale with flexible schema handling.

4. Justification for the Structure

- **Kafka** allows decoupling producers and consumers, buffering spikes, and supports reprocessing via topic replays.
- **Druid** offers millisecond-latency aggregation and filtering for real-time dashboards and alerts, critical for trade execution.
- **Cassandra's wide-row model** suits append-heavy time-series writes and ensures linear scalability.
- **S3/HDFS** is cost-efficient and integrates with big data engines for retrospective analysis over the 5-year data window.
- **Elasticsearch** fits the search-heavy use case for unstructured/semi-structured social data.

5. Trade-Offs

Consideration	Benefit	Trade-Off
Kafka + Druid	Ultra-fast ingestion & query latency	Operational complexity; requires tuning memory and shards
Cassandra	Highly scalable, fault-tolerant write performance	Poor for complex ad-hoc queries or joins
Elasticsearch	Fast text search, great for JSON	High resource usage, not ideal for heavy analytics
S3/HDFS	Cheap, long-term retention	High query latency unless paired with compute engines

Real-time vs Cost	Millisecond alerts = more RAM, SSD, CPU	Expensive to maintain low-latency infra

Module - 2

Data Source:

Structured client transaction logs(ACID) - PostgresSQL --> good for structured transactional data

Portfolio performance reports - Azure Blob Storage --> cost effective and scalable for semi-structured files

SEC fillings - Azure Data Lake Storage -->Ideal for large, varied, regulatory documents

Module - 3
Elasticsearch for Unstructured Text
Provides fast full-text search and relevance scoring for research reports without upfront schema constraints.
Scales horizontally for large corpus of documents.
Integrates via REST API easily into Python/R pipelines.
PostgreSQL for Structured Ratings
ACID compliance ensures integrity of analyst ratings and relationships.
Mature ecosystem (ODBC, JDBC) for BI tools and Python/R connectors.
Fixed schema simplifies joins and aggregations.
Trade-off: less flexible for schema evolution but ratings schema is stable.
MongoDB for Alternative Data
Handles evolving, semi-structured data (shipping logs, image metadata) without schema migrations.
Dynamic document model supports varied fields per record.
Trade-off: eventual consistency and higher storage overhead vs. flexibility.
Snowflake as Central Data Warehouse
Consolidates curated data for analytics, ML, and reporting.
Supports ANSI SQL; integrates seamlessly with Python/R connectors.
Scales compute and storage independently, optimizing cost vs. performance.
Trade-off: data latency from micro-batch ingestion, which is acceptable for analytics.
Overall Trade-offs
Cost vs. Scalability: Leveraging cloud-native systems (Elasticsearch Service, Snowflake) incurs managed service fees but reduces operational overhead and ensures elastic scaling.

Flexibility vs. Consistency: Mix of NoSQL and RDBMS balances evolving data ingestion with data integrity requirements.
Query Performance vs. Storage Efficiency: Specialized stores (Elasticsearch for search, Snowflake for analytics) optimize query performance at the cost of data duplication and ETL overhead.

Module 5

Data source:

-Trade reconcilation logs : Relational Database (SQL) -> As large data size and structural data

-Dark web scraping feeds: Graph like database(neo4j) to model relationship between user and post

-Employee access patterns: Time-based database ex)Timescale DB