Module 1: Real-Time Market Data Ingestion

1. Time-series Database

2. Reasons:  
It is designed to handle large amounts of time-stamped data efficiently. It supports fast inserts, efficient time-based queries, and can store years of historical data while keeping good performance.

3. Structure  
A table that stores the timestamp, stock symbol, data source, and related values like price and volume.   
The database will organize the data by time to make queries and storage more efficient.

4. Why the particular structure?  
This structure makes it easy to search by time and symbol, which matches how we want to analyze market data. It also helps the database manage and compress the data well for long-term storage.

Module 2: Client Portfolio Management  
  
Database Type: RDBMS (e.g., PostgreSQL or MySQL)  
  
Reason:   
ACID compliance is essential to maintain data integrity during financial transactions. Complex joins are required to analyze client portfolios from multiple perspectives, and the database must support GDPR-compliant audit trails to meet regulatory requirements.  
  
Suggested Structure:   
Relational tables organized as follows:  
  
Client Transactions Table: Stores detailed records of financial transactions, including timestamps, client IDs, transaction amounts, and transaction types.  
  
Portfolio Performance Table: Contains historical data of portfolio values, returns, and key performance indicators.  
  
Justification: RDBMSs like PostgreSQL are well-suited for financial applications due to their strong consistency, support for complex queries, and ability to maintain transaction logs. The structure allows for efficient querying and data integrity management.

Module 3: Research and Analytics  
Database Type: NoSQL - MongoDB  
Suggested Structure:

1.Research Reports Collection:  
Unique identifier for each report  
Report title and full content

2.Analyst Ratings Collection  
Stock symbol/identifier  
Rating category (Buy/Sell/Hold)  
Target price

3. Alternative Data Collection  
Data source identifier  
Flexible data storage section for:  
Web scraping results  
Data quality metrics

Justification:  
1.Schema Requirements  
Dynamic schema essential for alternative data sources  
Research reports need flexible document structure

2.Built-in text search capabilities for research content  
Horizontal scaling for growing data volumes  
Efficient handling of both structured and unstructured data

3. Technical Integration  
Native support for Python/R through robust drivers  
Easy integration with ML frameworks  
JSON-like structure ideal for modern applications

[4.Business](https://4.Business) Requirements  
Quick retrieval of research documents  
Efficient storage and querying of time-series data  
Support for complex queries across different data types

5.Scalability  
Sharding capabilities for large datasets  
Replication for high availability  
Handles concurrent read/write operations effectively

Module 4: Regulatory Reporting  
  
Database Type: Data Lake (AWS S3) or Data Warehouse (Snowflake)  
  
Suggested structure:

1. Raw ingestion of data (batch CSV, Fatca logs, and AML alerts) in the data lake.

2. AML alerts can be ingested via streaming tools in micro batches.  
3. The data can be processed and converted to parquet files for efficient querying.  
4. ETL pipelines can be setup to transform the data as required.  
5. Load the transformed data into snowflake.  
6. Users can access data using BI tools.  
  
Why this structure?  
Data lake stores raw data cheaply at scale. No schema enforcement is needed.  
ETL pipelines clean, validate and convert raw data into columnar formats and load them into the warehouse  
Data warehouse uses partitioning and clustering to ensure fast query responses  
This can support overngith batch sumbissions efficiently.

Module 5:  
  
Module 5 deals with fraud detection. When detecting fraud, we are handling trade recon data, dark web feed data, and employee access pattern data. Because this module needs to handle massive volumes and might need to make connections between suspicious transactions and employee activity, the best data model to persist information would be a graph-based data model.  
  
DB choices: Neo4j, Amazon Neptune  
  
For example, in fraud detection, two accounts might appear unrelated if stored relationally. However, if they interact with the same suspicious group of users or systems, these accounts should be flagged/alerted.  
  
How the overall module can be structured:  
  
Trade recon logs: RDBMS

* Typically involves structured data with defined relationships
* i.e. transaction, timestamp, party involved, trade details

Dark web scraping feeds and employee access patterns: Graph DB

* Will have interconnected data, such as user relationships, site access patterns, and communication between entities