System Architecture and ETL Data Pipeline

- Diagram showing:
  - Client → Node.js Server → Redis Cache → MongoDB Atlas
  - Data flow for ETL (Extract from Redis, Transform in app logic, Load to MongoDB).

- Redis acts as a fast cache layer to reduce MongoDB query load.

- TTL (Expiration) automatically cleans up stale cache data.

Lessons Learned from Extending Cached Database Pipeline

- Redis dramatically reduces query latency (from ~90 ms to < 5 ms).
- Implementing cache aside, read-through, and TTL requires careful key management.
- TTL testing helped understand automatic cache invalidation.
- Learned how to connect multiple databases (MongoDB Atlas + Redis) in Node.js.

Challenges and What Worked Well

- Worked Well: Connection stability after configuring .env and auth parameters.
- Challenges: Debugging Redis connection timing and MongoDB URI issues.
- Improvement: Add more error logging and monitor cache hit rate.
- Overall: Gained a deeper understanding of cache strategies and real-time performance optimization.