# 1. Architecture & ETL Data Flow (Redis → MongoDB)

- Redis serves as the in-memory cache for fast data retrieval.
- ETL pipeline extracts data from Redis, transforms it into structured JSON, and loads it into MongoDB.
- MongoDB acts as the persistent NoSQL storage for analytics and backup.
- Scheduler (e.g., cron/worker) triggers ETL jobs periodically.
- Data Flow: Application → Redis (cache) → ETL Script → MongoDB (storage).

# 2. Lessons Learned

- Importance of managing cache expiration and consistency.
- Proper schema design in MongoDB improved query performance.
- Logging and monitoring were essential for debugging ETL jobs.
- Automated ETL improved reliability and reduced manual sync tasks.
- Understanding data serialization formats (JSON, BSON) was key.

# 3. Challenges

✅ What Worked Well:

- Redis–MongoDB connection stability.
- ETL scripts executed efficiently for small datasets.
- MongoDB aggregation pipeline supported fast analytics.

⚠️ What Did Not Work Well:

- Handling large cache invalidations slowed ETL performance.
- Occasional data duplication during concurrent ETL runs.
- Needed better error handling for failed transfers.