Architecture & ETL Data Flow (Redis → MongoDB)

- DATA SOURCE INGESTED USING ETL SCRIPTS (EXTRACT, TRANSFORM, LOAD)
- REDIS ACTS AS A HIGH-SPEED CACHING LAYER FOR RAPID DATA RETRIEVAL
- MONGODB DATA LAKE STORES PROCESSED AND HISTORICAL DATA
- DASHBOARD TOOLS CONNECT FOR ANALYTICS AND VISUALIZATION

Lessons Learned from Extending Cached Database Pipeline

- Redis significantly improved query response times and scalability.
- MongoDB's flexible schema allowed integration of diverse data sources.
- ETL automation ensured consistent, reliable data ingestion.
- Monitoring Redis TTL and eviction policies was critical for accuracy.
- Streamlined analytics improved data-driven decision-making.

Challenges What Worked Well? What Did Not Work Well?

- What Worked Well:
- Caching reduced latency and optimized data retrieval.
- ETL automation improved consistency and reduced manual errors.

- What Did Not Work Well:
- Cache invalidation required manual intervention at times.
- Synchronizing Redis and MongoDB was complex during heavy loads.
- Memory management in Redis needed careful tuning.

- 💡 Future Improvements:
- Introduce automated cache refresh logic.
- Implement monitoring dashboards for better system insights.