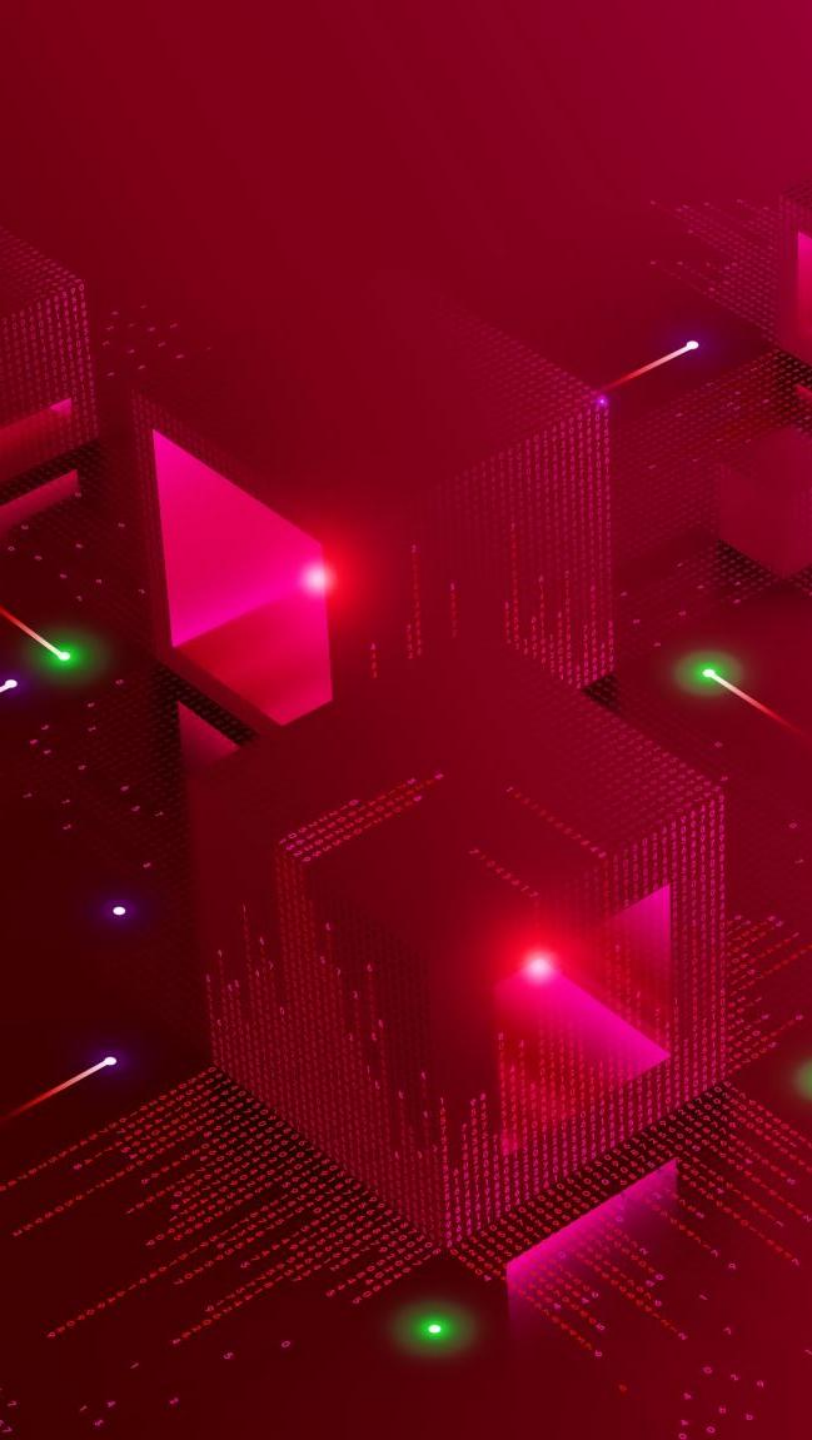

# Architecture & ETL Data Flow (Redis → MongoDB)

App will request data from REDIS

Checks if there is cache or not

Loads the DB

# Lessons Learned from Cached Database Pipeline

- Implementing Redis cache reduced data retrieval time significantly.
- Cache-Aside and Read-Through improved user experience during repeated queries.
- TTL caching ensured stale data was replaced automatically.
- Write-Through and Write-Behind exposed trade-offs between real-time consistency and write speed.
- Benchmarking confirmed faster response times during warm-cache scenarios compared to cold-cache reads.

## Challenges, Observations & Future Improvements

- Cache invalidation was the most difficult aspect to manage.
- Required careful balancing of TTL duration to prevent stale reads.
- Write-Behind introduced risks of temporary inconsistency before flush.
- Monitoring Redis memory usage and key eviction policies was necessary.
- Future plans include implementing LRU cache eviction, real-time monitoring dashboard, and automated performance alerts.