**Plan to optimise data mart**

Estimated time to implementation: 9 weeks

Stage 1: Performance Diagnosis

Time: week 1 - week 2

Goals: identify and diagnose key efficiencies and performance issues causing delays in data mart

Tasks:

1. Query logging and analysis (week 1 - 3 to 5 days)

* Team: senior data analyst, BI analyst
* Set up query logs to track query execution times and analyse query plans
* Indexing audit: evaluate current indexing strategies
* Buffer for: troubleshooting logging configuration
* Potential bottlenecks: full table scans, redundant joins, suboptimal indexing, data redundancy(denormalized columns causing inefficient operations/data scans)

1. ETL monitoring (week 1&2 - 5 to 8 days)

* Team: ETL engineer
* Monitor ETL processes runtime and resource consumption to identify potential bottlenecks
* Documentation of monitoring process to guide automation of monitoring alerts in the later stage
* Buffer for: issues detected → additional data validation required
* Potential bottlenecks: inefficient ETL processes (the 3 hour ETL batch could cause delays in the data availability)

1. Workload and resource monitoring (week 1&2 - 5 to 8 days)

* Team: cloud architect, database administrator/architect
* Use cloud monitoring tools: AWS cloudwatch to gather performance metrics
* Performance metrics to collect: (CPU, RAM, I/O) during peak usage
* Buffer for: unexpected tool configurations, access issues
* Potential bottlenecks: concurrency issue (too many people using queries)

Deliverables:

* Performance analysis report
* Outlines the primary performance issues and bottlenecks
* Layout action plan for the next stage
* Details software and hardware used in the process for future reference
* Details improvements made to the monitoring process for future improvement

Stage 2: Schema and Query optimization

Time: week 3 - week 5

Goals: refine database schema and optimise queries

Tasks:

1. Schema redesign and column pruning (week 3 - 5 to 10 days)

* Team: senior data analyst, database architect
* Remove redundant columns
* Redesign schema (potentially use star/snowflake models for faster joins and improved query performance. Reduce the number of redundant or unused columns in tables.)
* Buffer for: test the schema changes in a staging environment to ensure there are no query disruptions.

1. Indexing strategy update (week 3 - 5 to 10 days)

* Team: database architects
* Implementing new indexing strategies, Ensure high impact queries are benefited
  + Compound index on columns used in frequently run queries and foreign keys to speed up JOIN operations
  + Use bitmap indexes for columns with low cardinality
* Enable query caching for repeated queries to store results temporarily, reducing the processing load.
* Buffer for: validate the impact of new indexes on live query performance.

1. Partitioning tables (week 3&4 - 5 to 10 days)

* Team: database architects
* Implement table partitioning (need to decide what is the best way to partition, ie by period) to improve query performance by limiting the amount of data being scanned
* Buffer for: Allow 1 week to migrate partitions and test partition performance under load.

1. Materialised views creation (week 4 - 3 to 5 days)

* Team: BI analysts
* Implement materialised views for frequently accessed summary tables (e.g., transaction summaries, aggregated reports).
* Buffer for: set up automatic refresh schedules for materialised views.

Deliverables:

* Report detailing each solution applied to respective bottlenecks
* Rationale behind each solution
* Bottlenecks/troubleshooting issues during the process
* Performance tests
* Room for potential improvement
* Visible improvement in query performance

Stage 3: ETL process optimisation

Time: week 5 - week 7

Goals: Streamline the ETL process to enable near real-time updates without overwhelming the system.

Tasks:

1. Switch to Incremental ETL (week 5&6 - 5 to 10 days):

* Team: ETL engineer
* Shift from full-batch ETL processing to incremental ETL or Change Data Capture (CDC). This ensures only new or changed data is processed, improving ETL efficiency and reducing resource consumption.
* CDC tools: AWS DMS, Azure data factory minimally affect the source database by efficiently tracking only the delta changes
  + Perform CDC in real-time, to minimise the lag between updates without triggering full scans of large tables.
* Buffer: testing incremental ETL in a parallel environment to ensure data accuracy.
* Before deploying incremental ETL into production, set up a parallel testing environment that mirrors production. This allows us to correct the change tracking and validate data integrity without affecting operations.
  + Break the ETL jobs into parallel workflows by processing data in chunks (e.g., per region or customer segment) to distribute the workload across multiple nodes, improving ETL speed.

Deliverables:

* Incremental ETL Setup: CDC or log-based incremental ETL with changes properly tracked.
* Test Results: Documentation of test results in the parallel environment, showing data consistency, ETL runtime improvements, and system resource usage.
* Monitoring Plan: Active monitoring configuration to track incremental ETL performance and resource consumption post-implementation.

2. Implement Micro-Batching or Streaming ETL (week 5,6,7 - 5 to 15 days):

* Team: ETL Engineer, Cloud Architect
* 2.1 Micro-batching:
* Shift from large batch ETL runs to micro-batching or stream-based ETL (e.g., every 5-10 minutes) using tools like AWS Kinesis, Apache Kafka, or Azure Stream Analytics.
* How to avoid impacting performance:Introduce micro-batching gradually in non-peak hours to avoid overwhelming the system with new ETL processes.
* Buffer: test micro-batching in production and for training team members on managing streaming workloads.
* 2.2 Real-Time Streaming:
* Implement real-time streaming ETL using technologies like AWS Kinesis, Apache Kafka, Azure Stream Analytics.
* How to avoid impacting performance:Streaming ETL can be implemented alongside micro-batching to handle high-priority, time-sensitive data (eg, customer transactions may be streamed in real-time while less time-sensitive data is micro-batched.)
  + Ensure proper configuration of backpressure handling to avoid overloading the system with high-speed incoming data.
* 2.3 Testing and Training:
* Perform load tests in a staging environment that mimics the actual data flow.
* Test how the system handles both peak and off-peak data loads.
* Train team members on managing streaming workloads, including how to monitor real-time data flows and troubleshoot any lag or failure.
* Buffer: Allocate 1-2 weeks to allow for additional testing and staff training, ensuring smooth transitions

Deliverables:

* Micro-batching ETL Process: Configured to run smaller batches every 5-10 minutes.
* Streaming Pipeline: Real-time data pipeline setup for critical, high-frequency data (e.g., transactions, customer updates).
* Load Testing Results: Documentation of the system’s performance under micro-batching and streaming loads, confirming minimal impact on overall system resources.
* Training Materials: Staff training documentation for managing real-time streaming workloads.

Stage 4: Data Governance and Quality Management

Time: week 3 - week 8

Goal: Implement data quality checks and establish governance frameworks to ensure data consistency and security.

Tasks:

1. Data profiling and validation rules (ensuring data quality) (week 3,4,5 - 5 to 12 days)

* Team: Data Governance Specialist, ETL engineer
* Implement automated data profiling to detect and correct incomplete or inconsistent data in the ETL stage before loading data into the data mart. Run profiling checks during each ETL cycle and flag data quality issues.
* Set up data validation tools at the ETL stage to ensure data consistency and accuracy
* Introduce an master data management MDM system to standardise and centralise key customer, transaction, and enquiry data, ensuring accuracy and minimising duplication across the data mart.
* Buffer: 5 days to set up/fine-tune automated quality checks and handle any edge cases discovered.

1. Data stewardship and access control implementation (data governance) (week 5,6,7 - 5 to 12 days)

* Team: Data Governance Specialist
* Assign data stewards for each data domain (e.g., enquiries, transactions) and define access control policies to ensure data privacy and security.
* Apply role-based access controls (RBAC) and track user activities to ensure proper usage of data resources and prevent unauthorised access.
* Buffer: Add 1-2 weeks to establish and train the data stewardship team.

1. Data Catalog Setup (data governance) (week 7&8 - 5 to 10 days)

* Team: BI Analysts, Data Governance Specialist
* Implement a data catalogue (e.g., AWS Glue Data Catalog or Alation) to ensure all data assets are documented and searchable, which improves discoverability and compliance.
* Ensure metadata accuracy
* Buffer: Add 1 week to ensure metadata accuracy and catalogue maintenance.

Deliverables:

* Automated quality checks
* Master data management
* Data stewards
* Role based access controls (RBAC)
* Fully implemented data catalogue, providing documentation and searchable data sets which will improve transparency and cohesiveness.

Stage 5: Monitoring and Maintenance

Time: week 9 onwards

Goal: Set up automated monitoring tools and implement regular maintenance processes to ensure long-term efficiency

Tasks:

1. Performance Monitoring Setup (week 9 - 5 to 8 days)

* *Team*: Cloud Architect, DBA, ETL Engineer
* Set up automated performance monitoring with alerts (e.g., AWS CloudWatch, Azure Monitor) for key metrics such as:
  + query latency:
    - time taken for queries to execute
    - slow query log (must define latency threshold of ‘slow’)
    - check when full table scans occur
  + resource usage on the cloud infrastructure
    - CPU usage: monitor CPU utilisation (amazon redshift → amazon cloudwatch, Google BigQuery → Google cloud monitoring, Azure SQL warehouse → monitor the data warehouse unit via Azure monitor
    - Memory utilisation: memory bottlenecks force system to use slower disk-base storage.
    - Disk I/O: monitor read and write operations to see if slow disk access is contributing to delays
    - Network throughput: measure data transfer rate, network bottlenecks can slow query execution times in distributed cloud data platforms.
  + ETL job performance
* Monitor user queries and workloads to understand the frequency and complexity of queries run during peak and non-peak hours.
  + Real-time Performance Dashboards: Set up dashboards using monitoring tools (e.g., AWS CloudWatch, GCP Stackdriver, or Datadog) to track real-time performance metrics such as query latency, resource utilisation, and ETL success/failure rates.
  + Usage Analytics: Continuously track query patterns and user behaviour to identify and optimise slow-performing queries or resource-intensive processes.
* Continuously monitor ETL job execution times and failure rates, setting up alerts for job failures or delays.
* Buffer: Allow 2-3 days to configure custom thresholds and alerting mechanisms.

1. Regular maintenance (ongoing)
   1. Regular index and partition maintenance (monthly meeting to access the bottle necks and update the necessary changes for the coming months)
      1. Rebuild and optimise indexes and update table partitions based on data growth and query changes.
   2. Data archiving (week 9,10,11)
      1. Archive older, infrequently accessed data to a cold storage layer (e.g., AWS S3 or Azure Blob Storage), leaving only frequently accessed data in the main tables to reduce query times.
      2. Buffer: Allow 1-2 weeks for archiving implementation and testing.
   3. Infrastructure Auto-scaling:

* Set up auto-scaling policies on cloud resources, scaling up during peak hours and scaling down during off-hours to optimise costs.

Deliverables:

* Create dashboards in AWS Cloudwatch, Google data studio, or Azure Monitor to provide real-time visibility into each of the key resources
* Set up alerts in cloud monitoring tools to notify your team of potential resource issues
* Use historical data collected through monitoring to adjust the system
* Regular maintenance tasks like index optimization and ETL auditing