📊 Walmart Sales Advanced DBMS Project

📌 Overview

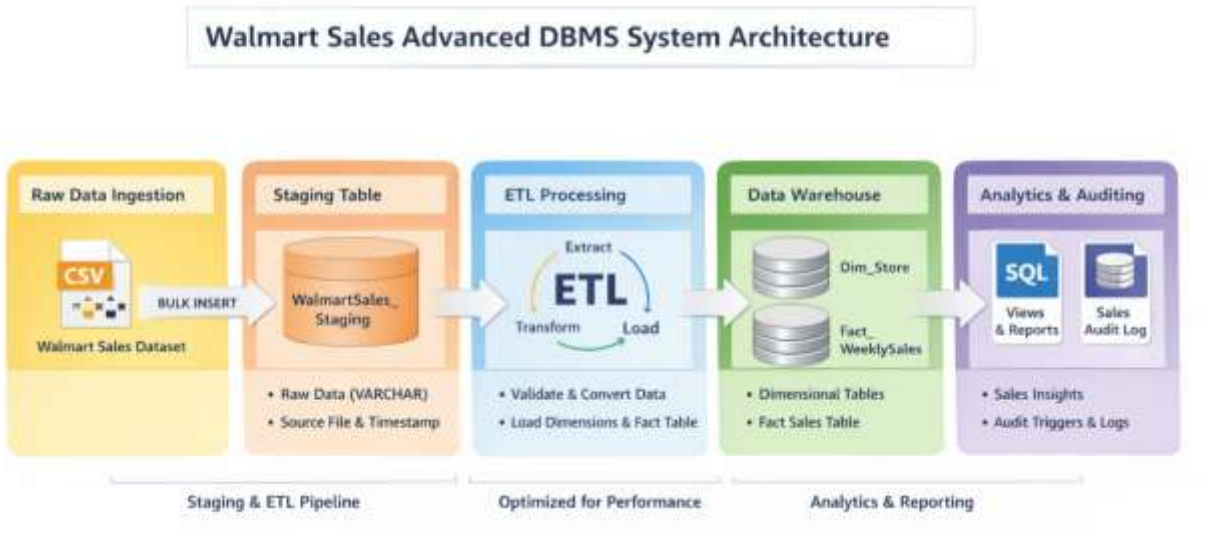
This project implements an enterprise-grade retail sales database system using SQL Server.
It simulates real-world data warehousing workflows, including staging-based ETL, fact–dimension modeling, auditing, performance tuning, and analytics-ready views.
The design mirrors how large-scale retailers (e.g., Walmart) handle high-volume transactional data.

🛠️ Technologies Used

- SQL Server
- SQL Server Management Studio (SSMS)
- T-SQL
- BULK INSERT
- ETL Pipelines
- Indexing & Partitioning (Design)
- Relational & Dimensional Modeling

🧱 System Architecture

CSV Dataset
↓
Raw Staging Table (VARCHAR-based)
↓
Data Validation & Transformation
↓
Dimension Tables
↓
Fact Tables
↓
Indexes & Performance Optimization
↓
Analytics Views



🚀 Key Features

✅ Core Relational Design

- Normalized schema with referential integrity
- Foreign keys and constraints

✅ Staging-Based ETL Pipeline

- Raw CSV ingestion using BULK INSERT
- Data type cleansing using TRY_CONVERT
- Fact table population via controlled ETL

✅ Dimensional Modeling

- Dim_Store dimension
- Fact_WeeklySales fact table
- Star-schema design principles

✅ Auditing & Change Tracking

- INSERT / UPDATE / DELETE triggers
- Centralized audit table for traceability

✅ Performance Optimization

- Indexes on high-usage analytics columns
- Partitioning strategy documented for Enterprise SQL Server

✅ Analytics Layer

- Sales summaries
- Product performance insights
- Time-based analysis readiness

📁 Project Structure

```
sql/
├── 01_Schema.sql
├── 02_Data_Insertion.sql
├── 03_Views.sql
├── 04_Triggers_audits.sql
├── 05_Procedures_udf.sql
├── 06_Cursor.sql
├── 07_Staging_ETL.sql
├── Bulk_Inserting.sql
├── Dimension_table.sql
├── Fact_Table.sql
├── Indexes_Fact_Table.sql
└── 08_indexes_partitioning.sql
```

▶️ How to Run (IMPORTANT)

Step 1 — Create Database

Open SSMS and run:

```
CREATE DATABASE WalmartSalesDB;
GO
USE WalmartSalesDB;
GO
```

Step 2 — Execute SQL Files (STRICT ORDER)

Phase 1 — Core Schema

01_Schema.sql
02_Data_Insertion.sql
03_Views.sql
04_Triggers_audits.sql
05_Procedures_udf.sql
06_Cursor.sql

Phase 2 — ETL Pipeline

07_Staging_ETL.sql
Bulk_Inserting.sql
Dimension_table.sql
Fact_Table.sql

Phase 3 — Optimization

Indexes_Fact_Table.sql
08_indexes_partitioning.sql

📈 Example Analytics Queries

```
SELECT TOP 10 *
FROM dbo.Fact_WeeklySales
ORDER BY Weekly_Sales DESC;

SELECT
    Sale_Date,
    SUM(Weekly_Sales) AS TotalSales
FROM dbo.Fact_WeeklySales
GROUP BY Sale_Date
ORDER BY Sale_Date;
```

🧠 Enterprise Design Notes

- Staging tables isolate dirty raw data
- Fact tables store clean, analytics-ready data
- Indexes improve reporting performance
- Partitioning is documented for enterprise editions
- Triggers provide compliance-grade auditability

🎯 What This Project Demonstrates

- End-to-end SQL data engineering
- Realistic ETL workflows
- Performance-aware schema design
- Production-safe data loading
- Interview-ready enterprise architecture

📌 Author

Rishabh Agrawal

Masters in Information Technology Project Management
Arizona State University