Modern Data Warehouse for Sales Analytics

Sreelakshmi Vattapparambil Gopakumar
Dept. of Computer Science
Indiana University
Indianapolis, USA

Anagha Pradeep
Dept. of Computer Science
Indiana University
Indianapolis, USA

***Abstract*—This paper presents the design and implementation of a modern data warehouse for sales analytics built on Microsoft SQL Server using the Medallion (Bronze–Silver–Gold) architecture. The source environment consists of fragmented CSV exports from customer relationship management (CRM) and enterprise resource planning (ERP) systems, with no single source of truth for customers, products, or sales transactions. We implement a layered extract–transform–load (ETL) pipeline in T-SQL that ingests raw CSV files into a Bronze landing layer, applies cleansing and conformance in a Silver layer, and exposes analytics-ready star-schema views in a Gold layer. The resulting solution supports self-service business intelligence through Power BI, improves data quality via systematic validation checks, and provides an extensible template for similar sales analytics scenarios. We discuss design decisions, implementation details, evaluation outcomes, and potential extensions such as incremental loading and slowly changing dimensions.**

***Index Terms*—Data warehouse, Medallion architecture, SQL Server, ETL, Business intelligence, Star schema, Sales analytics**

I. Introduction

A. Motivation

Organizations increasingly depend on data-driven decision-making to remain competitive in dynamic markets [1]. In sales and marketing contexts, this requires timely and reliable insights into customer behavior, product performance, and revenue trends. However, in practice, operational data is often dispersed across multiple systems such as CRM and ERP platforms. A common pattern is that each team periodically exports CSV files, which are then manually merged in spreadsheet tools to prepare reports. This manual approach is error-prone, difficult to reproduce, and does not scale with growing data volumes or new analytical requirements.

In the environment targeted by this project, sales-related data is scattered across several CSV exports. Customer master data, product information, and transactional sales records originate from a CRM system, while extended customer attributes, location details, and product category hierarchies are maintained in an ERP system. There is no integrated data model that brings these sources together. Business users must repeatedly perform ad-hoc joins, filters, and transformations in Excel or other tools, leading to inconsistent definitions and long delays between data updates and decision-making.

B. Problem Statement

The absence of a modern data management solution creates multiple challenges:

- **Data fragmentation:** Related entities such as customers and products appear in multiple CSV files with different identifiers and attribute sets.
- **Manual integration:** Analysts manually join and clean data each time they produce a report, which is both time-consuming and non-repeatable.
- **Inconsistent metrics:** Different teams may implement slightly different logic when calculating core metrics such as total revenue or active customers, undermining trust in the data.
- **Limited historical analysis:** Because data is not systematically persisted in a structured warehouse, conducting longitudinal analyses and trend monitoring is difficult.

Consequently, decision-makers lack a single, authoritative view of sales performance and customer behavior. The goal of this project is to address these challenges by introducing a well-designed data warehouse that consolidates and structures the data for analytics.

The primary objective of this work is to design and implement a modern sales analytics data warehouse using SQL Server and the Medallion architecture [3]. More specifically, the project aims to:

- Integrate CRM and ERP sales-related CSV data into a unified, consistent data mart.
- Apply systematic cleansing, normalization, and conformance across sources.
- Expose a star-schema model in a Gold layer optimized for analytical queries and self-service BI.
- Implement an end-to-end ETL pipeline using T-SQL stored procedures and views.
- Provide a foundation for advanced analytics and future enhancements such as incremental loading.

C. Objectives

The primary objective of this work is to design and implement a modern sales analytics data warehouse using SQL Server and the Medallion architecture. More specifically, the project aims to:

- Integrate CRM and ERP sales-related CSV data into a unified, consistent data mart.
- Apply systematic cleansing, normalization, and conformance across sources.
- Expose a star-schema model in a Gold layer optimized for analytical queries and self-service BI.

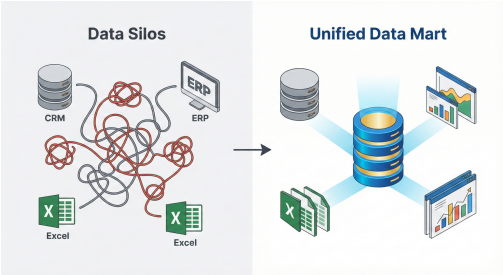

Fig. 1. From messy, disconnected data silos (CRM, ERP, Excel) to a single unified data mart powering clean, consistent reporting and analytics.

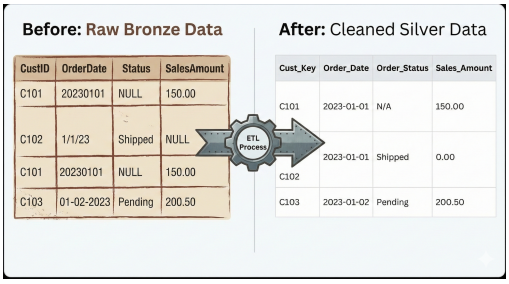

Fig. 2. Silver Layer: Cleansing Standardization

- Implement an end-to-end ETL pipeline using TSQL stored procedures and views.
- Provide a foundation for advanced analytics and future enhancements such as incremental loading.

D. Contributions

The contributions of this paper are threefold:

- It demonstrates how the Medallion (Bronze–Silver–Gold) architecture [4], typically associated with data lakehouse platforms, can be effectively realized in a traditional relational database environment using SQL Server.
- It presents a complete, reproducible ETL design that moves from raw CSV ingestion to a robust star schema, including a discussion of design trade-offs and implementation details.
- It integrates data quality checks and documentation into the pipeline, offering a practical template for similar sales analytics projects that need traceability and trust in their metrics.

The remainder of the paper is organized as follows. Section II reviews related work in data warehousing and Medallion architectures. Section III describes the data sources, users, and overall architecture. Section IV details the Medallion design and schema organization. Sections V–VII describe the implementation of the Bronze, Silver, and Gold layers. Section VIII discusses data quality and validation. Section IX presents reporting and analytics use cases. Section X reports evaluation outcomes. Section XI summarizes design lessons and contributions, and Section XII concludes with future work.

II. Related Work

A. Classical Data Warehousing Paradigms

Classical data warehouse design has been significantly influenced by two main schools of thought. The first is the dimensional modeling paradigm, popularized by Kimball [1], which focuses on building subject-oriented data marts using star and snowflake schemas. In this approach, business processes are modeled as fact tables that store measurements, while descriptive entities such as customers, products, and dates are represented as dimension tables. Dimensional models are highly optimized for analytical queries and are widely used as the basis for business intelligence (BI) and reporting systems.

The second paradigm, associated with Inmon [2], advocates a centralized, normalized enterprise data warehouse (EDW). In this methodology, data from operational systems is integrated into a third normal form (3NF) model, and downstream data marts are derived as needed. This results in a layered architecture where an EDW serves as the single integrated source of truth, and multiple data marts provide specialized views for particular analytical domains.

Both paradigms highlight the importance of integration, data quality, and consistent definitions. However, they were originally conceived in an era of on-premises relational databases and may not directly address challenges associated with semi-structured data, large-scale distributed processing, or new cloud-native architectures.

B. Lakehouse and Medallion Architectures

More recently, the data management ecosystem has seen the emergence of the lakehouse paradigm, which combines elements of data lakes and data warehouses. In lakehouse architectures, data is typically stored in low-cost object storage using open formats, while transactional semantics and analytical performance are provided by specialized engines. A common organizing principle within lakehouse systems is the Medallion architecture [4], [5].

The Medallion architecture divides data into three conceptual layers:

- **Bronze layer:** Contains raw, ingested data as close as possible to the original source format.
- **Silver layer:** Holds cleansed, standardized, and conformed data that can be reused across multiple downstream applications.
- **Gold layer:** Provides business-ready data models tailored to analytics, reporting, or machine learning workloads.

This layered approach emphasizes progressive refinement of data quality and structure. It allows data engineers to separate concerns: ingestion logic can be modified without impacting business models, and analytical views can evolve independently of low-level data ingestion details.

C. SQL-Based Medallion Implementations

Although the Medallion architecture is frequently associated with big data stacks, its principles are largely technology-agnostic [6]. Several practitioners have demonstrated that a

similar layered design can be implemented on traditional relational database platforms such as SQL Server or PostgreSQL [3]. In such implementations, schemas or databases are used to represent Bronze, Silver, and Gold layers, with stored procedures and views implementing the ETL logic.

Typical patterns include:

- Using bulk insert operations to populate Bronze tables from CSV files, preserving raw data for traceability.
- Defining Silver tables as cleansed and conformed entities that can be regenerated from Bronze at any time.
- Exposing Gold-layer star schemas as views built atop Silver tables, reducing duplication and simplifying maintenance.

This project follows and extends these patterns by focusing specifically on a sales analytics use case that integrates CRM and ERP data, and by emphasizing documentation, quality checks, and reproducibility in its design [7], [8].

III. System Overview

A. Data Sources

The source systems for this project are represented by six CSV files, logically grouped into CRM and ERP categories.

1) CRM Data: The CRM system provides three exports:

- **custinfo.csv**: Contains customer master data, including customer identifiers, names, and core attributes.
- **prdinfo.csv**: Holds product master information, such as product IDs, names, and basic descriptive attributes.
- **salesdetails.csv**: Stores transactional sales records, including customer and product references, dates, quantities, and amounts.

Together, these files offer a view of who purchased what, when, and in what quantity.

2) ERP Data: The ERP system contributes three additional files:

- **CUSTAZ12.csv**: Extends customer profiles with additional attributes such as date of birth and gender.
- **LOCA101.csv**: Provides customer location details including city, region, and country.
- **PXCATG1V2.csv**: Represents product category hierarchies that relate products to categories and subcategories.

These ERP files enrich the CRM data with demographic and geographic context, and they introduce a hierarchical product classification that is useful for aggregated analytics.

B. Users and Use Cases

The designed data warehouse is intended to support several types of users:

- **Business analysts:** Build dashboards and ad-hoc reports over the sales data.
- **Sales managers:** Monitor key performance indicators such as revenue by region, top customers, and popular product categories.
- **Data engineers:** Maintain, extend, and optimize the data warehouse and ETL pipelines.

Key use cases include:

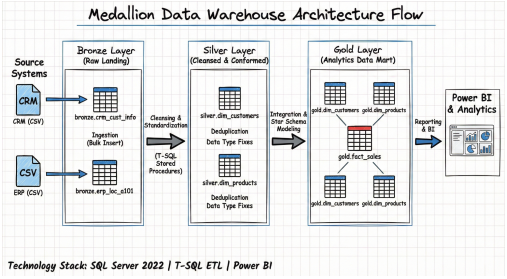

Fig. 3. High-level architecture: CSV sources, Medallion layers in SQL Server, and BI consumption.

- Identifying top customers by revenue and average order value across different time horizons.
- Analyzing product performance by category, market, and customer segment.
- Monitoring trends in sales quantity and revenue over multiple years.
- Supporting customer segmentation and targeted campaigns based on extended attributes and location.

C. High-Level Architecture

Figure **??** illustrates the high-level architecture of the solution. Data flows from the source CSV files into the Bronze schema in SQL Server, then through the Silver schema where cleansing and conformance occur, and finally into the Gold schema where star-schema views are defined. Power BI connects to the Gold layer to deliver interactive visualizations and reports.

D. Technology Stack

The main technologies used in this project are [9]:

- **Microsoft SQL Server 2022:** Serves as the primary data platform for storage, indexing, and query execution.
- **T-SQL stored procedures:** Implement ETL logic for loading, transforming, and publishing data between layers.
- **Power BI Desktop:** Connects to the Gold layer and provides visualization, exploration, and dashboard creation capabilities.

This stack is widely used in industry, making the project relevant and easily transferable to real-world environments.

IV. Medallion Architecture Design

A. Layer Definitions and Responsibilities

The Medallion architecture in this project is realized using three SQL Server schemas: `bronze`, `silver`, and `gold`. Each layer has a distinct purpose and set of responsibilities [5].

- **Bronze:** The Bronze layer acts as a raw landing zone. It receives data from CSV files with minimal transformation, preserving structure and content to maximize traceability.

TABLE I
LOGICAL MEDALLION LAYERS IN SQL SERVER

Layer	Purpose	Implementation
Bronze	Raw ingestion and traceability	Tables in `bronze`
Silver	Cleansing and conformance	Tables in `silver`
Gold	Analytics-ready star schema	Views in `gold`

- **Silver:** The Silver layer contains cleansed, standardized, and conformed data. It applies business-neutral transformations that improve data quality and allow consistent joins across systems.
- **Gold:** The Gold layer exposes analytical data models, primarily in the form of star schemas. It is closely aligned with business use cases and is designed for consumption by BI tools.

Table I summarizes key characteristics of each layer.

B. Schema Organization and Naming Conventions

Clear naming conventions enhance maintainability and communication among team members. The following conventions are used:

- Bronze tables mirror source files and include prefixes indicating the source, e.g., `bronze.crm_custinfo`, `bronze.erp_loca101`.
- Silver tables are named after logical entities, such as `silver.customers`, `silver.products`, and `silver.sales`.
- Gold views follow dimensional modeling conventions, such as `gold.dim_customers`, `gold.dim_products`, and `gold.fact_sales`.

This organization makes it easier to understand where a particular table belongs in the pipeline and what transformations have been applied to it.

C. ETL Strategy

The ETL strategy is based on full refreshes using truncate-and-load for simplicity and reproducibility. For each layer transition:

- Bronze tables are truncated and reloaded from CSV using bulk operations.
- Silver tables are truncated and repopulated from Bronze via transformation queries.
- Gold views are defined as virtual objects on top of Silver tables and therefore automatically reflect the latest Silver data.

While incremental loading strategies can be more efficient for large datasets, a full-refresh approach is adequate for the scope of this project and simplifies reasoning about data state and lineage. Future work may introduce incremental and change-data-capture (CDC) mechanisms.

V. BRONZE LAYER: RAW INGESTION

A. Design Goals

The Bronze layer is responsible for ingesting source data from CSV files into SQL Server. Its primary design goals are:

- **Preservation:** Maintain data as close as possible to its original representation.
- **Traceability:** Enable tracing any downstream issue back to the raw source record.
- **Performance:** Support efficient bulk loading for repeatable ingestion.

B. Table Structures

For each CSV file, a corresponding table is created in the `bronze` schema. Columns generally mirror those in the CSV, including names and data types where feasible. Additional metadata columns (such as load timestamps or file identifiers) can be added to support auditing and lineage.

As an example, a simplified version of the Bronze customer table might include:

- `CustomerID`: string identifier from CRM.
- `FirstName`, `LastName`: customer name fields.
- `MaritalStatus`, `Gender`: demographic attributes.
- `CreateDate`: date when the record was created in the CRM system.

C. Load Process

The load process for Bronze follows a consistent pattern:

1) Truncate the target Bronze table to remove previous data.
2) Use a bulk insert operation to load data from the corresponding CSV file into the table.
3) Record load statistics (such as row counts and timestamps) in a control table.

This process is encapsulated in stored procedures, allowing all Bronze tables to be loaded through a single orchestration procedure or a scheduled job. By encapsulating the logic, the project ensures that changes in file paths or formats can be localized without affecting downstream queries.

VI. SILVER LAYER: CLEANSING AND CONFORMANCE

A. Objectives

The Silver layer refines raw Bronze data into reliable, reusable entities. Its key objectives are:

- **Data cleansing:** Remove or correct invalid, inconsistent, or duplicate data.
- **Standardization:** Ensure consistent formats for dates, text, and numeric values.
- **Conformance:** Align keys and attributes across CRM and ERP sources.

B. Cleansing Operations

Typical cleansing operations applied in the Silver layer include [10]:

- **Date normalization:** Converting dates from various input formats into a single standard type, such as `DATE`, using a consistent representation.

- **Whitespace handling:** Trimming leading and trailing spaces and normalizing inner spaces in text fields to reduce unintended duplicates.
- **Case standardization:** Converting textual fields to consistent case (e.g., upper case for country codes) to simplify comparisons.
- **Handling missing values:** Replacing missing values with appropriate defaults or flags, or excluding records when necessary.

These operations are implemented in T-SQL queries that select from Bronze tables and insert into Silver tables.

C. Conformance and Integration

Conformance aligns related records across systems so that they can be joined and analyzed consistently. For the customer entity, this includes:

- Matching CRM customer identifiers with ERP customer rows using shared business keys.
- Integrating demographic attributes from ERP with master data from CRM.
- Combining location information to create a consolidated customer profile.

For products, conformance involves:

- Combining CRM product master data with ERP category hierarchies.
- Establishing a consistent product identifier that can be used across all tables.

The resulting Silver tables represent cleansed and integrated entities such as `silver.customers`, `silver.products`, and `silver.sales`.

D. Silver ETL Patterns

Silver ETL procedures follow a truncate-and-load pattern:

1) Truncate the target Silver table.
2) Insert into the target by selecting from the relevant Bronze tables, applying cleansing and conformance transformations.
3) Optionally record metadata about transformation counts (e.g., rejected or corrected records) for monitoring.

This pattern ensures that Silver data can be fully regenerated from Bronze whenever the transformation logic changes, improving maintainability and reproducibility.

VII. Gold Layer: Star Schema Modeling

A. Dimensional Modeling Approach

The Gold layer presents a star-schema model optimized for analytical queries and BI tools. A star schema consists of [1]:

- A central fact table that stores measurements, such as sales amounts and quantities.
- Multiple dimension tables that store descriptive attributes about customers, products, time, and potentially other entities.

In this project, Gold objects are implemented as views on top of Silver tables rather than as physical tables. This decision simplifies maintenance and allows changes in the Silver layer to be reflected in the Gold layer without additional load steps.

[Image of Star Schema Diagram]

B. Fact View: `gold.fact_sales`

The fact view `gold.fact_sales` provides a unified representation of sales transactions. It typically includes:

- Surrogate keys referencing dimension views, such as `CustomerKey` and `ProductKey`.
- Natural keys (e.g., original customer and product IDs) for traceability.
- Measures such as `SalesAmount`, `Quantity`, and potentially `DiscountAmount`.
- Date fields that can be used to connect to a date dimension.

The view definition joins Silver sales data with Silver customer and product data to ensure that all foreign keys can be resolved.

C. Customer Dimension: `gold.dim_customers`

The customer dimension integrates CRM and ERP customer data into a single, enriched view. Attributes may include:

- Basic identifiers such as `CustomerID`.
- Name fields such as `FirstName` and `LastName`.
- Demographics from ERP, including `Gender` and `DateOfBirth`.
- Location attributes such as `City`, `Region`, and `Country`.

A surrogate key, `CustomerKey`, is introduced to decouple analytical identifiers from operational ones. This design also prepares the model for potential slowly changing dimension (SCD) implementations in the future, where multiple rows could represent different historical versions of a customer.

D. Product Dimension: `gold.dim_products`

The product dimension combines product master data with category hierarchies. Attributes typically include:

- `ProductID` and `ProductName`.
- Category information such as `CategoryName` and `SubcategoryName`.
- Additional product attributes from CRM or ERP, as available.

As with customers, a surrogate `ProductKey` is used for analytical joins. Flattening the category hierarchy into a single dimension row simplifies reporting by allowing analysts to slice and dice sales by category without performing complex hierarchical joins in the BI tool.

E. Advantages of the Gold Design

The Gold-layer design offers several advantages:

- **Performance:** Star schemas are well suited for query optimizers and BI tools, enabling efficient aggregations.
- **Usability:** Analysts can work with meaningful business entities and measures without needing to understand underlying transformation logic.

- **Maintainability:** Implementing the Gold layer as views reduces duplication and centralizes transformation logic in the Silver layer.

VIII. Data Quality and Validation

A. Quality Requirements

Data quality is critical for the trustworthiness of analytical outputs. This project defines quality requirements along three axes:

- **Completeness:** Essential attributes such as customer and product keys must not be missing in Gold views.
- **Uniqueness:** Dimension keys (e.g., `CustomerKey`) must be unique and stable within the dimension.
- **Referential integrity:** Every foreign key in the fact view must correspond to an existing dimension row.

B. Validation Checks

Validation scripts are executed against the Gold layer to verify that these requirements are met. Examples include:

- Queries that count null values in critical columns.
- Checks for duplicate surrogate keys in dimension views.
- Anti-join queries that detect fact rows with missing or invalid dimension references.

If violations are found, they are logged and investigated. In many cases, issues can be traced back to Bronze records, enabling targeted remediation.

C. Monitoring and Governance

While the project primarily focuses on technical implementation, it also highlights the importance of monitoring and governance. For production environments, quality checks should be integrated into orchestration workflows so that failures can trigger alerts and prevent bad data from being published to dashboards. Over time, a more comprehensive data quality framework can be adopted to manage rules, thresholds, and exceptions.

IX. Reporting and Analytics

A. Power BI Integration

Power BI Desktop is used to build dashboards and reports on top of the Gold layer. The BI model mirrors the star schema: the fact and dimension views are imported or accessed in DirectQuery mode, and relationships are defined according to the surrogate keys. Measures are defined in DAX or precomputed in SQL as needed.

B. Analytical Scenarios

Several analytical scenarios are implemented to illustrate the capabilities of the data warehouse:

- **Top customers by revenue:** A report that lists customers ranked by total sales, with slicers for time period and geography.
- **Product performance by category:** Visualizations that show revenue and quantity by product category and subcategory.
- **Time-series analysis:** Line charts and bar charts that plot sales over months or years to identify trends and seasonality.

These scenarios leverage the integrated customer and product dimensions to provide richer insights than would be possible using only individual CSV files.

C. Example Measures

Common measures used in the dashboards include:

- Total sales amount.
- Total quantity sold.
- Average order value per customer.
- Revenue contribution percentages by category or region.

These measures can be implemented with either DAX expressions in Power BI or as derived columns and aggregated views in SQL Server, depending on performance and governance considerations.

X. Results

A. Functional Outcomes

The implemented solution meets the core objectives outlined in Section I. In particular:

- Sales-related data from CRM and ERP sources is integrated into a single, coherent data mart.
- A layered ETL pipeline enables repeatable, auditable transformations from raw CSV files to analytics-ready views.
- Business users can consume a star-schema model that abstracts away technical complexity and provides a clear semantic layer for reporting.

B. Qualitative Evaluation

Qualitatively, the solution provides significant improvements over the previous manual reporting process. Analysts benefit from:

- Reduced time-to-insight because complex joins and cleansing steps are performed once in the pipeline rather than repeatedly in spreadsheets.
- Greater trust in metrics due to centralized logic and validation checks.
- Increased analytical richness through integrated customer and product attributes that were previously scattered across files.

C. Performance Considerations

Given the moderate size of the project dataset, full-refresh ETL and relational query execution are sufficient to deliver good performance. Fact and dimension views can be indexed via their underlying Silver tables, and Power BI can leverage import mode to pre-aggregate data for interactive exploration. For larger datasets, additional performance tuning and partitioning might be required, but the current design provides a solid baseline.

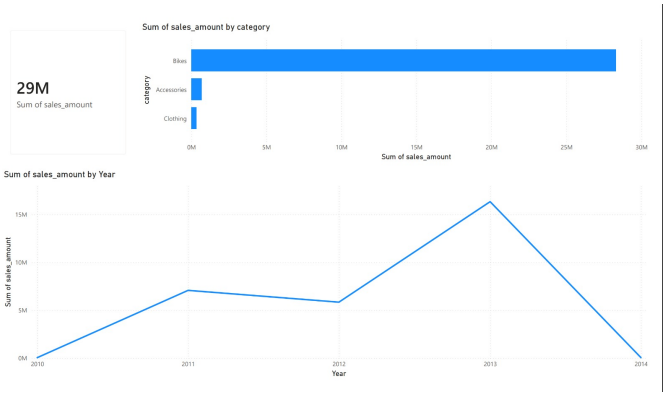

Fig. 4. Power BI dashboard built on the Gold layer, showing total sales, sales by category, and sales trends by year.

D. Power BI Dashboard Insights

To validate the usefulness of the Gold layer, an interactive Power BI dashboard was built on top of the `gold.fact_sales` and dimension views. Figure 4 shows a sample report containing a KPI card, a bar chart by product category, and a time-series chart by year.

Revenue overview: The dashboard tracks a total historical revenue of approximately 29 million currency units, serving as a high-level KPI for overall business performance. Category dominance: The Bikes category is identified as the primary revenue driver, generating significantly higher sales than Accessories or Clothing. Growth trends: The time-series analysis reveals a strong upward sales trajectory peaking in 2013, allowing stakeholders to easily track year-over-year growth patterns and identify periods of maximum performance.

E. Limitations

The project also has limitations:

- ETL is implemented as full refresh, which may become inefficient for very large datasets.
- Slowly changing dimensions are not yet implemented, limiting the ability to analyze historical attribute changes.
- The solution is deployed on a single SQL Server instance without horizontal scaling.

These limitations inform the roadmap for future work described in Section XII.

XI. Discussion and Contributions

A. Design Trade-Offs

Several design trade-offs were made during implementation. Using views for Gold objects simplifies maintenance and reduces data duplication, but it also means that some computation occurs at query time. Similarly, adopting full-refresh ETL simplifies pipeline logic but may not be efficient for very large or frequently changing datasets. These trade-offs are acceptable for the current context but should be revisited if the warehouse is extended to handle larger volumes or more complex requirements.

B. Practical Lessons Learned

From a practical standpoint, several lessons emerged:

- Clear separation of concerns across Bronze, Silver, and Gold schemas greatly simplifies reasoning about data state and lineage.
- Preserving raw data in Bronze is invaluable for debugging transformation issues and for auditing.
- Integrating data quality checks into the pipeline early helps prevent bad data from affecting dashboards and builds user trust.

C. Summary of Contributions

Overall, this project demonstrates that [11]:

- Medallion architecture principles can be successfully applied to a traditional SQL Server environment for sales analytics.
- A layered ETL design can move from raw CSV exports to a robust star schema while preserving traceability and maintainability.
- Combining dimensional modeling with layered data management yields a practical and extensible solution for real-world business intelligence needs.

XII. Conclusion and Future Work

A. Conclusion

This paper has presented the design and implementation of a modern sales analytics data warehouse on SQL Server using the Medallion architecture. By integrating CRM and ERP CSV exports into a structured Bronze–Silver–Gold pipeline and exposing a star-schema Gold layer, the solution addresses data fragmentation, reduces manual reporting effort, and improves the reliability of sales metrics. The architecture is flexible, extensible, and aligned with contemporary data engineering practices.

B. Future Work

Several directions for future work are planned:

- **Incremental loading:** Implement incremental and CDC-based loading strategies to support more frequent updates and reduce the overhead of full refreshes.
- **Slowly changing dimensions:** Introduce SCD patterns for key dimensions, particularly customers and products, to allow analysis of historical attribute changes.
- **Advanced analytics:** Build machine learning models for tasks such as churn prediction, cross-sell recommendation, or demand forecasting on top of the Gold layer.
- **Cloud migration:** Evaluate migrating the solution to a cloud-based platform that supports elastic scaling and additional services such as orchestration and monitoring.

By following these directions, the system can evolve from a single-domain sales analytics warehouse into a broader enterprise data platform.

REFERENCES

[1] R. Kimball and M. Ross, *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling*, 3rd ed. Wiley, 2013.

[2] W. H. Inmon, *Building the Data Warehouse*, 4th ed. Wiley, 2005.

[3] S. Paudel, "Building a modern data warehouse in SQL Server with Medallion Architecture," *Dev.to*, Jul. 2025. [Online]. Available: https://tinyurl.com/bddefvk9

[4] Databricks, "What is the Medallion Architecture?" [Online]. Available: https://www.databricks.com/glossary/medallion-architecture

[5] Microsoft, "Implement Medallion Lakehouse architecture in Microsoft Fabric," *Microsoft Learn*. [Online]. Available: https://learn.microsoft.com/en-us/fabric/analytics-engineer/medallion-lakehouse-architecture

[6] P. Spati, "Medallion architecture: A complete guide," *Medium*, 2023.

[7] B. Zini, "SQL data warehouse from scratch — Full hands-on data engineering project," *YouTube*, Feb. 2025. [Online]. Available: https://www.youtube.com/watch?v=9GVqKuTVANE

[8] B. Zini, "Data warehouse project repository," *GitHub*. [Online]. Available: https://github.com/barazini/sql-data-warehouse-project

[9] Microsoft, "Modern data warehouse architecture," *Azure Architecture Center*. [Online]. Available: https://learn.microsoft.com/en-us/azure/architecture/solution-ideas/articles/modern-data-warehouse

[10] Intellify, "A guide to the Medallion Architecture," *Intellify Blog*, 2024.

[11] Amazon Web Services, "Evolving the data warehouse," *AWS Whitepapers*.