MAKERERE UNIVERSITY BUSINESS SCHOOL

Business Intelligence and Data Warehousing

TOPIC 4

DATA WAREHOUSE IMPLEMENTATION

BUC 2225

Topic Outline

- Analyze the data warehouse development life cycle and best practices.
- Evaluate and compare different data warehouse architectures.
- Understand the different data warehouse components.
- Understand the challenges and considerations for successful data warehouse implementation

Introduction to Data Warehousing

- Data warehouses play a critical role in modern organizations, providing a centralized repository for storing, integrating, and analyzing data from multiple sources.
- By consolidating data in a structured and accessible format, data warehouses empower organizations to gain valuable insights, make informed decisions, and drive business growth.
- Data warehousing refers to the process of designing, building, and maintaining a centralized repository of integrated data from one or more disparate sources.
- The primary objective of a data warehouse is to facilitate efficient querying, analysis, and reporting to support decision-making processes within an organization

Components of Data Warehouse

- Data Sources
  - Operational systems, databases, files, and external sources from which data is extracted.
- ETL (Extract, Transform, Load) Processes
  - Extracts data from source systems, transforms it into a consistent format, and loads it into the data warehouse.
- Staging Area:
  - Intermediate storage for raw data before it is loaded into the data warehouse.
- Data Storage:
  - Fact tables: Store quantitative data (e.g., sales, revenue).
  - Dimension tables: Store descriptive attributes (e.g., product, customer).
  - Star Schema: Simple dimensional model with fact table surrounded by dimension tables.
  - Snowflake Schema: Normalized form of star schema with dimension tables further normalized.
  - Hybrid Schema: Combination of star and snowflake schemas for flexibility.

Components of Data Warehouse

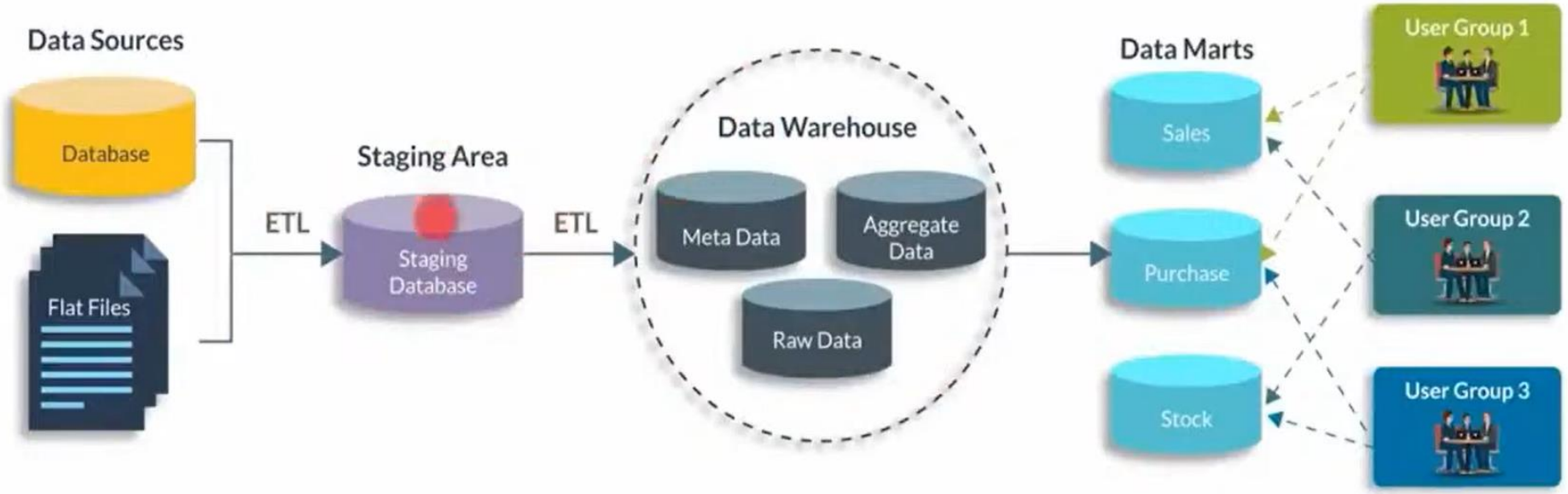
- Metadata Repository:
  - Stores metadata such as data definitions, transformation rules, and data lineage for documentation and understanding.
- Query and Analysis Tools
  - SQL interfaces, OLAP cubes, reporting tools, and BI dashboards for querying and analyzing data.
- Data Mart:
  - Subset of the data warehouse focused on a specific business function, department, or user group.
- Security and Access Controls:
  - Measures to ensure data security, privacy, and compliance with access controls, encryption, and authentication.
- Monitoring and Management Tools:
  - Tools for monitoring system performance, data quality, and user activity, and managing resources and configurations.

Components of Data Warehouse

- Backup and Recovery:
  - Processes and mechanisms for backing up and restoring data to ensure data integrity and continuity of operations.
- Data Governance Framework:
  - Policies, procedures, and processes for managing data assets, ensuring data quality, and compliance with regulations and standards.
- Scalability and High Availability Features:
  - Ability to scale resources and ensure continuous operation with features such as clustering, replication, and failover mechanisms.

Which architecture is this DW ?

Data Warehouse Architecture



Data Warehouse Architecture

- **Inmon Architecture:**
  - The emphasis is on building a centralized data warehouse as the single source of truth for the organization's data.
  - Data is first integrated into a normalized data model, reducing redundancy and ensuring data consistency.
  - This architecture prioritizes data integration and consistency over immediate access to data for end-users.
  - Data marts in an Inmon architecture are derived from the centralized data warehouse and serve specific business functions or departments.

Data Warehouse Architecture

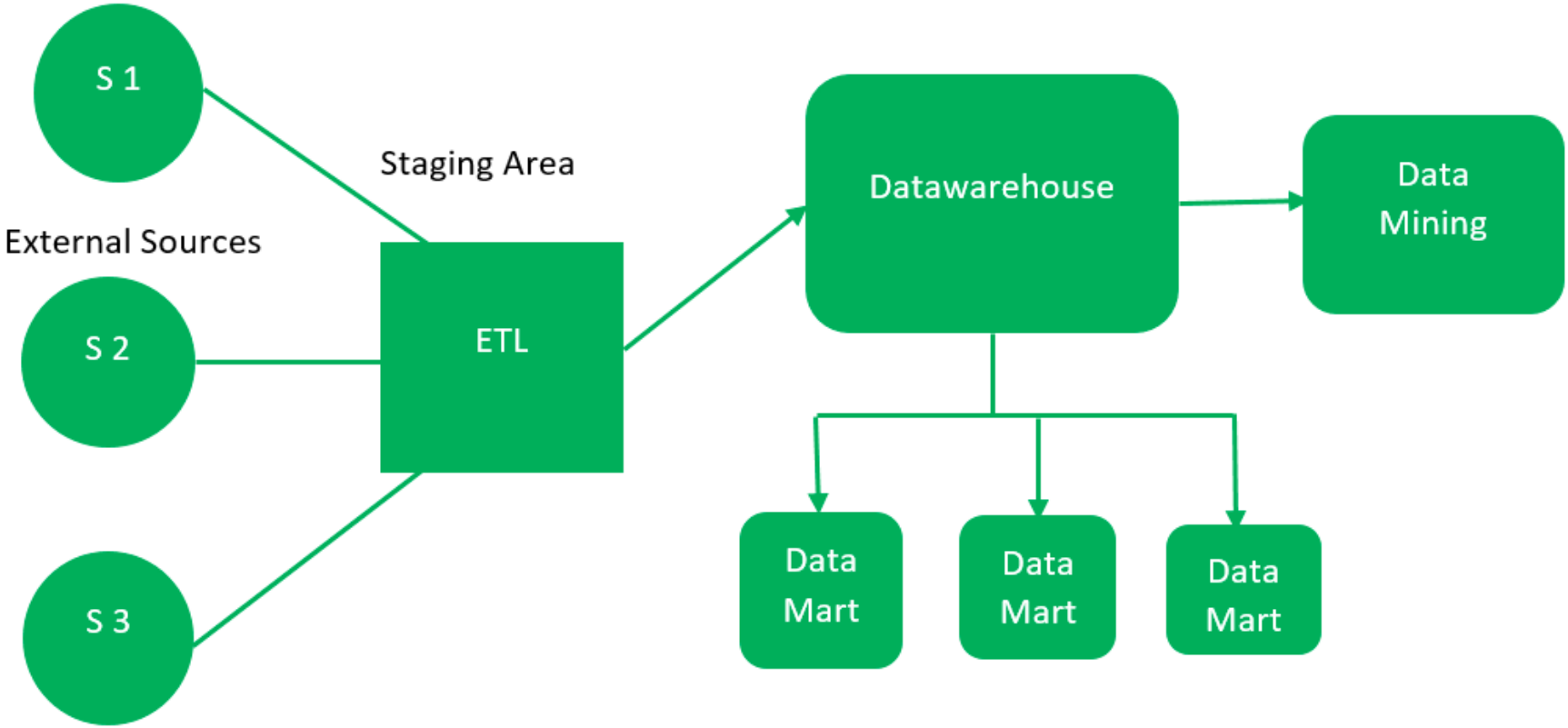


Image Source: Data Warehouse Architecture - GeeksforGeeks

Data Warehouse Architecture

- **Kimball Architecture:**
  - The Kimball architecture focuses on creating data marts first, which are then integrated to form a comprehensive data warehouse.
  - Data marts in Kimball architecture are designed to be business-centric and contain denormalized data optimized for end-user queries and reporting.
  - This architecture prioritizes rapid access to actionable data for business users, allowing for quicker implementation and delivery of analytical solutions.
  - The data warehouse in Kimball architecture is essentially a collection of integrated data marts.

Data Warehouse Architecture

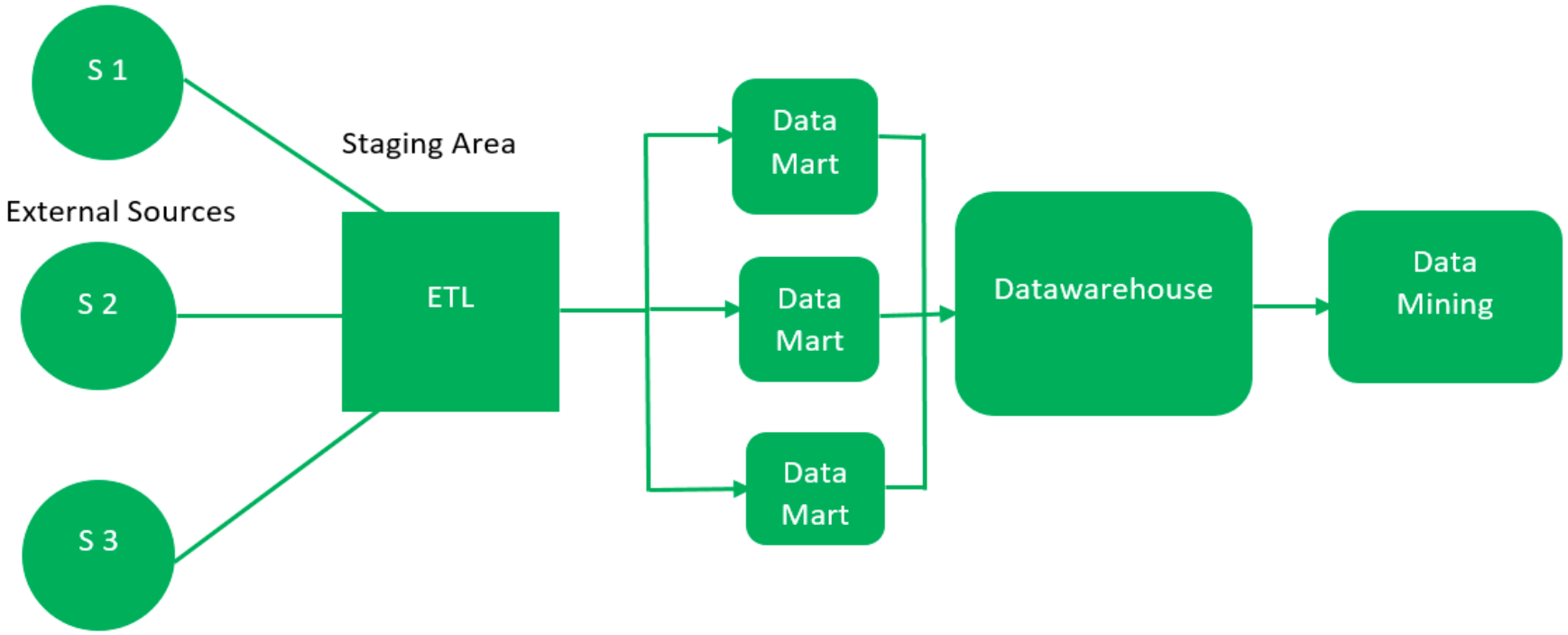


Image Source: Data Warehouse Architecture - GeeksforGeeks

Traditional Vs. Modern Data Warehouses

- Using either of the two DW architectures, you can build either a modern or traditional data warehouse.
- Traditional Data Warehouse
  - Traditional data warehouse architectures typically follow established principles, such as the Inmon or Kimball approaches.
  - They often rely on on-premises infrastructure, relational databases, and batch processing for data integration and analytics.
  - Traditional architectures prioritize structured data and often involve staged ETL processes to transform and load data into a centralized repository.
- Modern Data Warehouse
  - Modern data warehouse architectures embrace newer technologies and paradigms to address evolving business needs and data challenges.
  - They may incorporate cloud-based storage solutions, distributed computing frameworks, and real-time processing capabilities.
  - Modern architectures are more agile, scalable, and flexible, accommodating diverse data types, streaming data, and analytics at scale.

Data Warehouse Development Life Cycle

- Inception:
  - Define Project Scope: Clearly articulate the objectives, scope, and expected outcomes of the data warehouse project. Identify key stakeholders and establish project governance.
  - Establish Objectives: Determine the specific business objectives that the data warehouse aims to address. These objectives should align with the organization's strategic goals and priorities.
  - Initial Requirements: Gather preliminary requirements from stakeholders to understand their needs and expectations. These requirements will serve as the foundation for further analysis and design.

- Requirements Gathering:
  - Business Requirements: Conduct workshops, interviews, and surveys with business users to elicit their requirements. Document functional and non-functional requirements, including data sources, data types, frequency of updates, and performance expectations.
  - Data Sources Analysis: Identify and analyze existing data sources, both internal and external, that will feed into the data warehouse. Assess the quality, structure, and accessibility of these sources.
  - Data Profiling: Perform data profiling to understand the characteristics and quality of the source data. Identify any anomalies, inconsistencies, or gaps that may need to be addressed during the ETL process.

- Design:
  - Data Warehouse Schema: Design the logical and physical schema of the data warehouse, including fact tables, dimension tables, and relationships between them. Choose appropriate modeling techniques, such as star schema or snowflake schema, based on the analytical needs of the business.
  - ETL Processes: Design the ETL (Extract, Transform, Load) processes that will extract data from source systems, transform it into a consistent format, and load it into the data warehouse. Define data transformation rules, mappings, and workflows.
  - Metadata Structures: Define metadata structures to document data definitions, transformations, lineage, and dependencies. Establish standards and conventions for metadata management to ensure consistency and usability.

Data Warehouse Development Life Cycle

- Implementation:
  - Develop Components: Develop the components of the data warehouse, including databases, ETL scripts, data integration tools, and metadata repositories. Implement data quality checks, error handling mechanisms, and data validation routines.
  - Customization: Customize the data warehouse components to meet specific business requirements and technical constraints. Fine-tune performance optimizations, such as indexing, partitioning, and caching, to enhance query performance.

- Testing:
  - Functionality Testing: Verify that the data warehouse components function as intended, including ETL processes, data loading, and metadata management. Conduct unit tests, integration tests, and system tests to validate functionality.
  - Performance Testing: Evaluate the performance of the data warehouse under different conditions, such as varying data volumes, user concurrency, and query complexity. Identify and address any bottlenecks or performance issues.
  - Data Quality Testing: Assess the quality of the data in the data warehouse, including accuracy, completeness, consistency, and timeliness. Perform data validation, reconciliation, and profiling to ensure data integrity.

Data Warehouse Development Life Cycle

- Deployment:
  - Production Deployment: Deploy the data warehouse to the production environment, following established change management processes and procedures. Coordinate with system administrators, database administrators, and other stakeholders to ensure a smooth transition.
  - User Training: Provide training and support to end-users to familiarize them with the data warehouse and its capabilities. Offer documentation, tutorials, and hands-on workshops to empower users to leverage the data warehouse effectively.

- Maintenance:
  - Monitoring: Monitor the performance and usage of the data warehouse on an ongoing basis. Implement monitoring tools and dashboards to track system health, resource utilization, and user activity. Set up alerts and notifications to proactively identify and address issues.
  - Maintenance Tasks: Perform routine maintenance tasks, such as data backups, software updates, and database optimizations, to ensure the ongoing reliability and availability of the data warehouse. Address any issues or anomalies detected during monitoring.

Data Warehouse Development Life Cycle

- Inception:
  - Define project scope, goals, and initial requirements through stakeholder collaboration.
  - Establish project governance and identify key team members.
- Requirements Gathering:
  - Gather and analyze business requirements from stakeholders to understand their data needs.
  - Identify and assess data sources to determine data availability, quality, and relevance.
- Design:
  - Design the data warehouse schema, including fact tables, dimension tables, and relationships.
  - Define ETL processes to extract, transform, and load data into the warehouse.
  - Establish metadata structures to document data definitions, transformations, and lineage.
- 4. Implementation:
  - Develop data warehouse components, including databases, ETL scripts, and metadata repositories.
  - Customize components to align with specific business requirements and technical constraints.

Data Warehouse Development Life Cycle

- Testing:
  - Conduct functional testing to verify data warehouse components' behavior and functionality.
  - Perform performance testing to ensure the warehouse can handle expected data volumes and user loads.
  - Assess data quality through data validation and profiling to maintain accuracy and consistency.

- Deployment:
  - Deploy the data warehouse to the production environment following established change management processes.
  - Provide user training and support to ensure effective utilization of the warehouse.

- Maintenance:
  - Monitor the performance and health of the data warehouse, addressing any issues promptly.
  - Conduct routine maintenance tasks, such as backups, updates, and optimizations, to sustain optimal performance.
  - Continuously refine and enhance the warehouse to accommodate evolving business needs and technological advancements.

Challenges in DW Implementation

- Data Quality Issues:
  - Ensuring data accuracy, completeness, consistency, and timeliness is crucial but challenging due to data discrepancies, errors, and inconsistencies across disparate source systems.
- Data Integration Complexity:
  - Integrating data from multiple sources with varying formats, structures, and semantics can be complex and time-consuming, requiring extensive transformation and cleansing.
- Scalability:
  - Designing the data warehouse architecture to handle increasing data volumes and user concurrency while maintaining performance can be challenging, especially as data grows over time.
- Performance Optimization:
  - Optimizing query performance and response times, especially for complex analytical queries, requires careful indexing, partitioning, and tuning of database systems and ETL processes.
- Resource Constraints:
  - Limited budget, expertise, and resources can hinder data warehouse development, implementation, and maintenance efforts, leading to delays and compromises in quality.

Challenges in DW Implementation

- Organizational Alignment:
  - Achieving alignment between business objectives, IT capabilities, and stakeholder expectations requires effective communication, collaboration, and buy-in across departments and functions.
- Data Governance and Security:
  - Implementing robust data governance practices and security measures to protect sensitive data, ensure regulatory compliance, and mitigate risks requires careful planning and ongoing monitoring.
- Change Management:
  - Managing changes to data warehouse structures, processes, and requirements over time can be challenging, requiring flexibility, adaptability, and effective change management practices.
- User Adoption and Training:
  - Ensuring user adoption and proficiency in using the data warehouse requires comprehensive training, documentation, and support to empower users and maximize the warehouse's value.
- Technological Advancements:
  - Keeping up with rapidly evolving technologies and trends in data warehousing, such as cloud computing, big data, and machine learning, requires ongoing learning and adaptation.
- Legacy Systems Integration:
  - Integrating legacy systems and outdated technologies with modern data warehouse environments can present compatibility issues and require careful planning and migration strategies.

Best Practices for DW Implementation

- Clearly Define Objectives:
  - Start by clearly defining the objectives and goals of the data warehouse project. Ensure alignment with the organization's strategic priorities.
- Engage Stakeholders:
  - Involve stakeholders from different business units early in the process to gather requirements, ensure buy-in, and facilitate collaboration.
- Follow a Methodology:
  - Adopt a structured methodology, such as Kimball's Dimensional Modeling or Inmon's Corporate Information Factory (CIF), to guide the design and implementation process.
- Focus on Data Quality:
  - Prioritize data quality assurance from the outset. Implement data profiling, cleansing, and validation processes to maintain high-quality data.
- Design for Scalability:
  - Design the data warehouse architecture with scalability in mind to accommodate growing data volumes and user concurrency.
- Ensure Security and Compliance:
  - Implement robust data governance, security, and compliance measures to protect sensitive data and ensure regulatory compliance.

Best Practices for DW Implementation

- Iterative Development:
  - Adopt an iterative development approach, building and deploying the data warehouse in phases to incorporate feedback and deliver quick wins.
- Promote User Adoption:
  - Provide comprehensive user training, documentation, and support to ensure effective adoption and utilization of the data warehouse.
- Monitor Performance:
  - Implement monitoring tools and processes to track system performance, data quality, and user activity. Address issues promptly to maintain optimal performance.
- Plan for Maintenance:
  - Develop a plan for ongoing maintenance and support, including routine tasks such as backups, updates, and optimizations.
- Embrace Flexibility:
  - Design the data warehouse to be flexible and adaptable to changing business needs and technological advancements.
- Continuous Improvement:
  - Foster a culture of continuous improvement by soliciting feedback, monitoring key performance indicators, and investing in ongoing enhancements and optimizations.

Practical Session

DW Implementation

- 1. Create a folder on the Desktop in your name and registration number
- 2. Copy and paste the Sales.csv and dateDimension.txt files inside your folder.
- 3. Open PowerBI and create a new project. Save it as SalesDataWarehouse.
- 4. Extract the sales data from the Sales.csv file inside your project.
- 5. Carryout the following transformations on the extracted data.
- 6. Rename the Sales table to FactSales.
- 7. Make a copy of the FactSales table into 4 other tables that will be used to make the customer, product and geography dimension tables.
- 8. Rename the created new tables matching the dimension tables needed.
- 9. In the customer dimension, remove all duplicates.
- 10. Remove all the columns save customer id, customer name, email.

DW Implementation

- 11. Get the customer first name and last name from the email column.
- 12. Clean the email to remove the brackets from the email.
- 13. Repeat steps 8 to 10 (for step 10, decide which columns to stay with basing on the dimension you're working on) for the rest of the dimension tables.
- 14. Create a time dimension table using the dateDimension.txt file that was provided.
- 15. Remove the unnecessary columns in the factsales table.