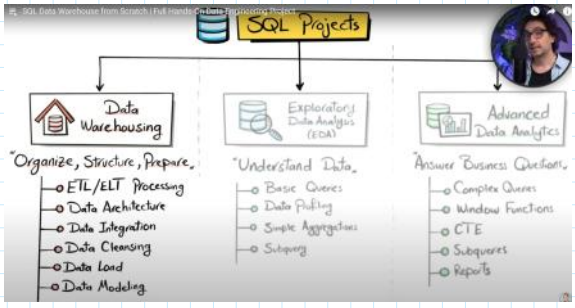
SQL Data Warehouse Project from Scratch from Data with Baraa

20 April 2025 15:10

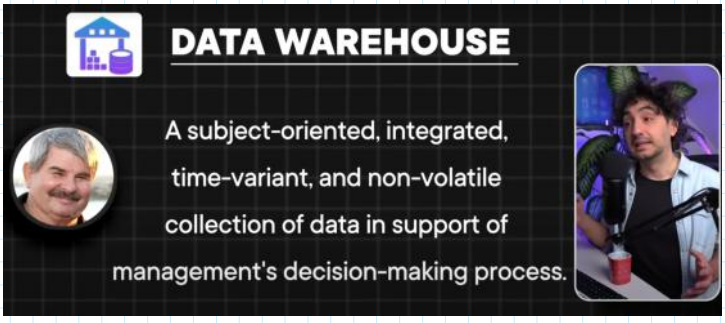
https://github.com/DataWithBaraa/sql-ultimate-course

https://github.com/DataWithBaraa/sql-data-warehouse-project

Using SQL Server

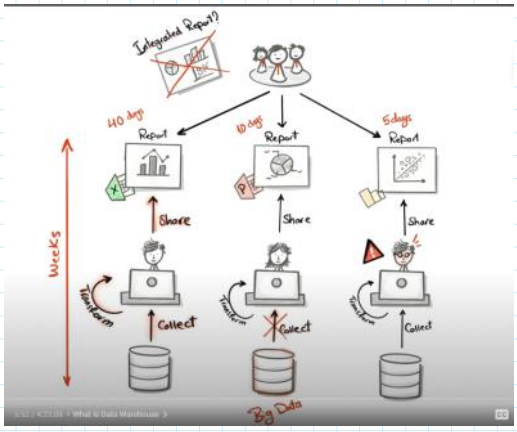


What is Data Warehouse?

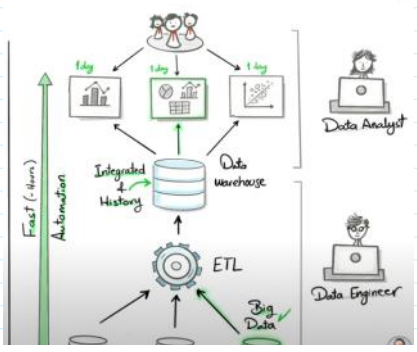


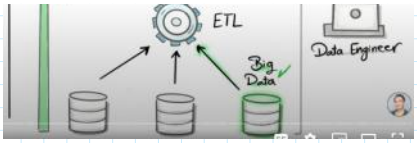
Dumb way of ETL process by multiple data analysts

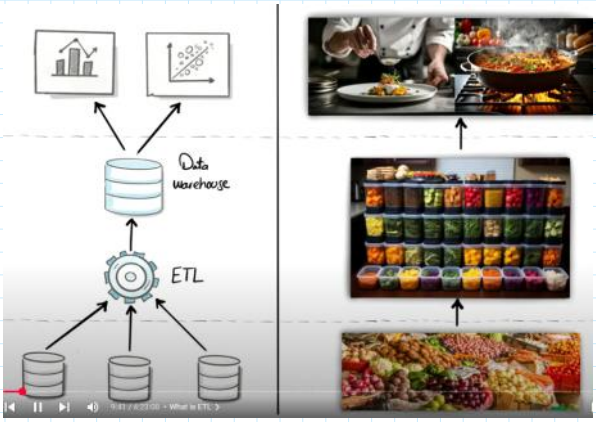
Problems - Cannot integrate, Cannot use big data. Clumsy



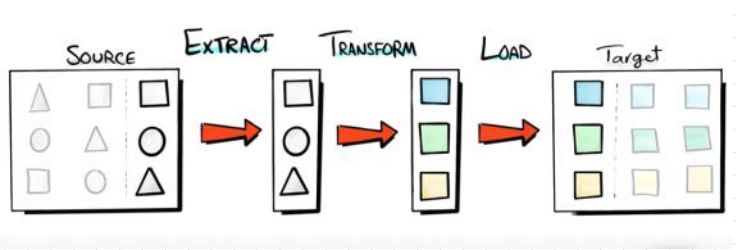
Simple Data warehouse architecture and advantages



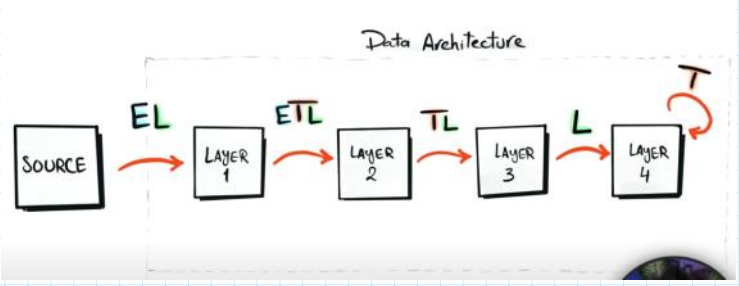




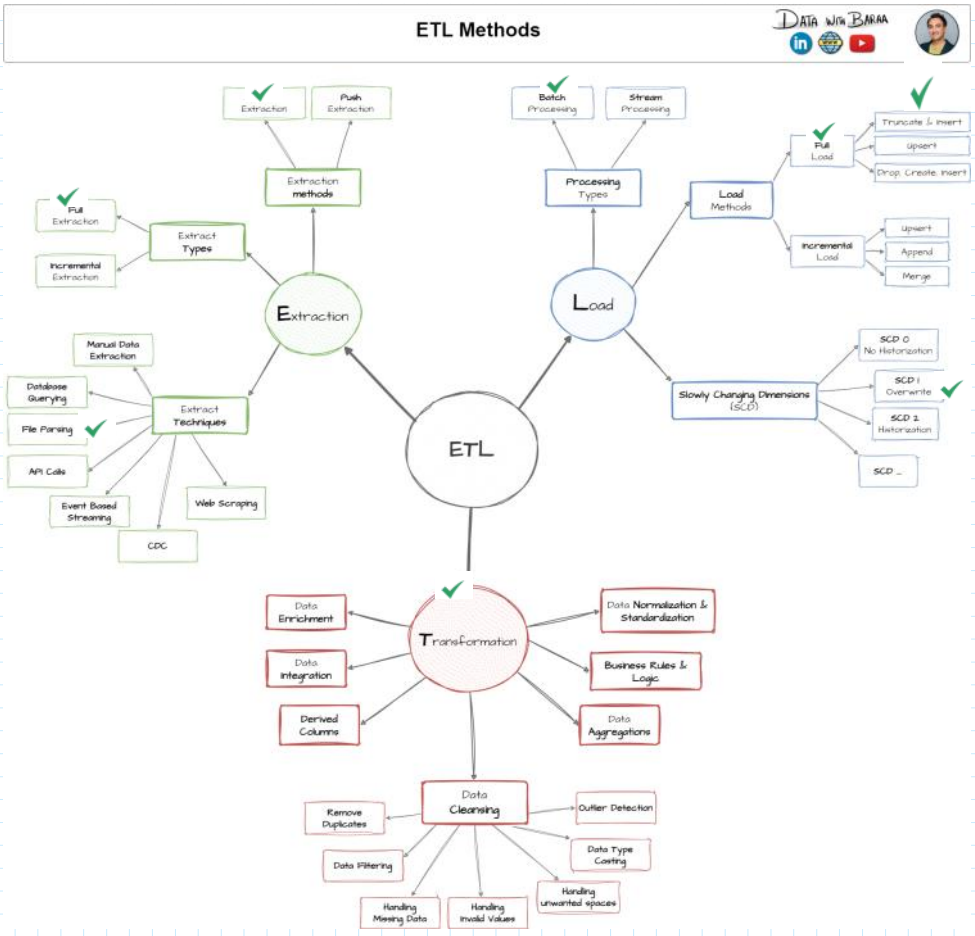
What is ETL?



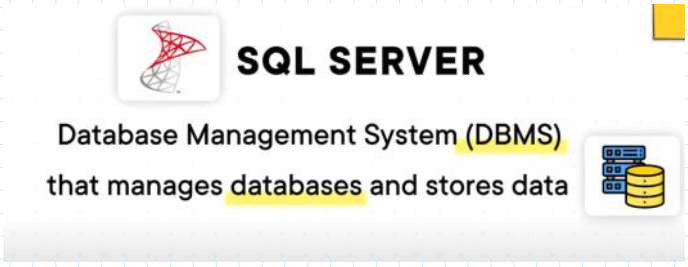
What are layers?

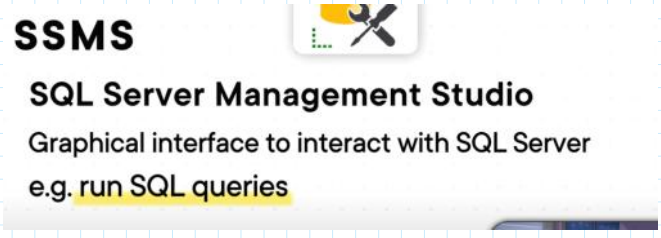


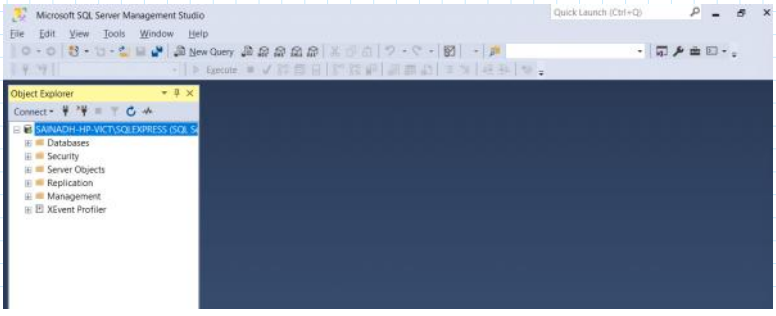
Different methods / techniques in ETL



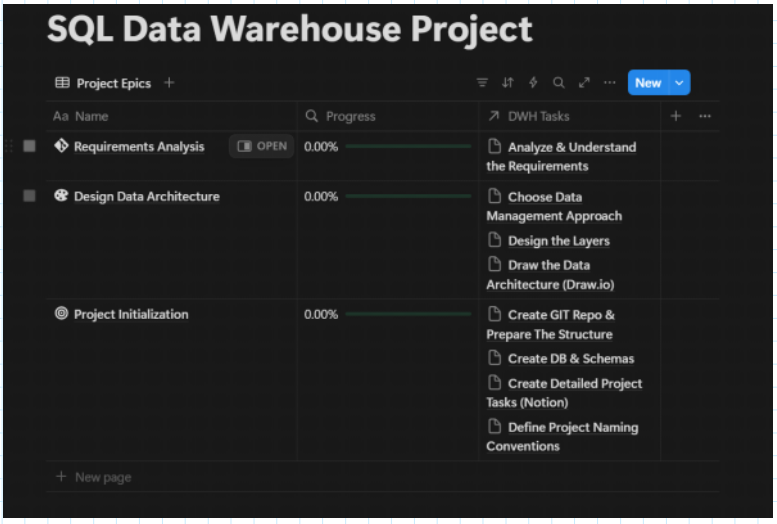
Download SQL Server, SSMS, NOTION, DRAW.io

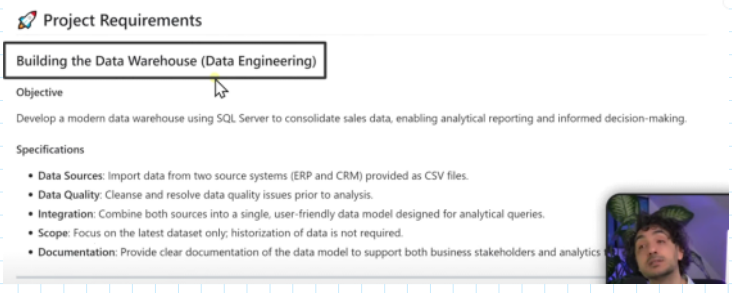


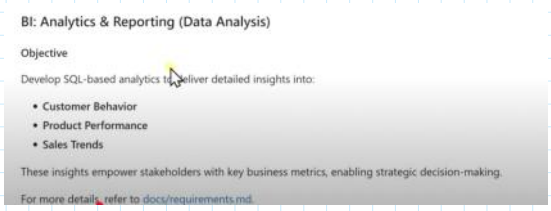




Install Notion and Create SQL Data Warehouse Project Page

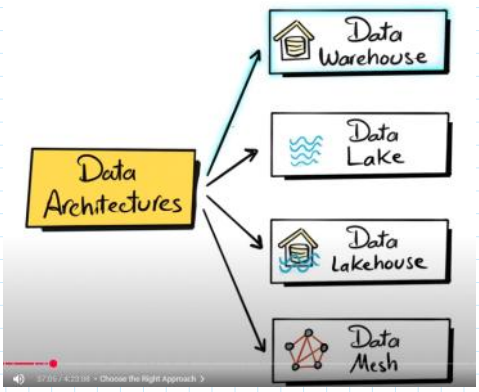




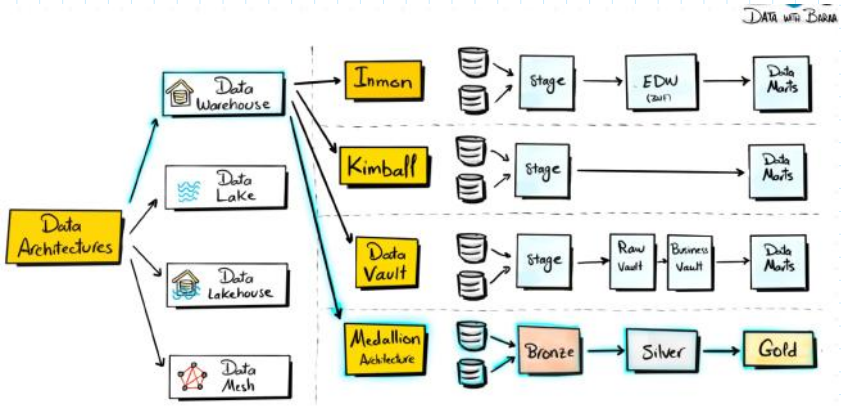


What Data Architect do?

Create Data pipeline for ETL which is scalable and robust

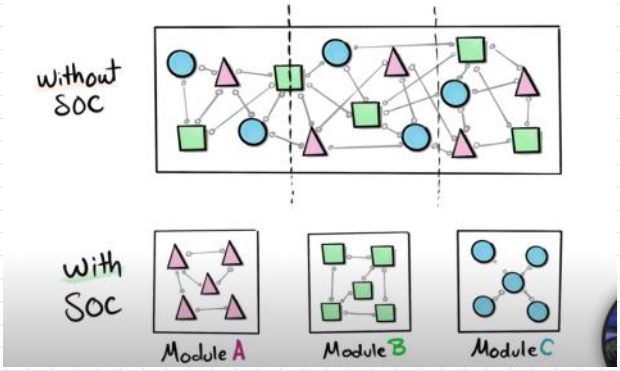


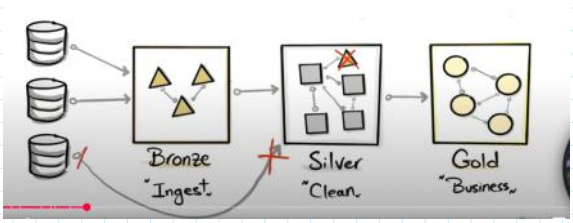
How to build Data Warehouse?



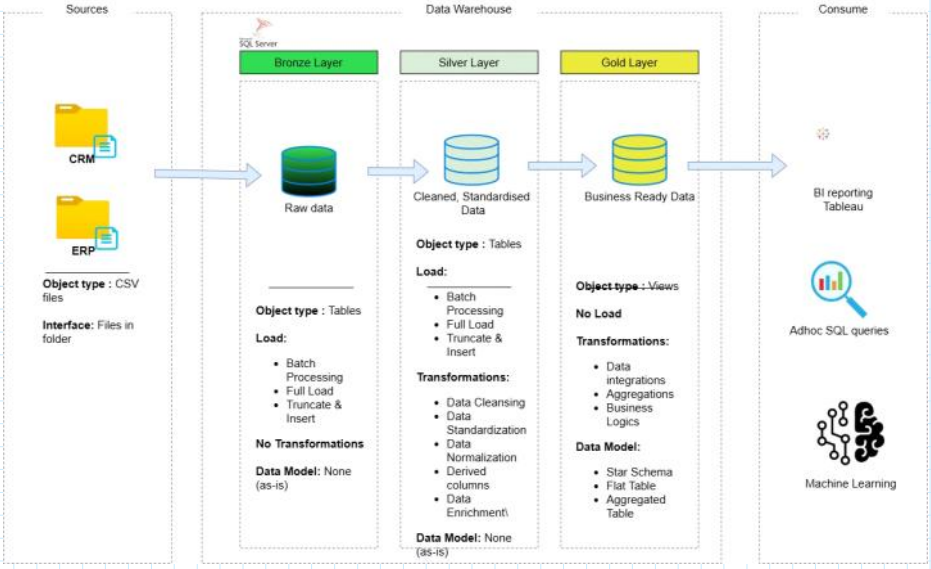
	Bronze Layer	Silver Layer	Gold Layer
Definition	Raw, unprocessed data as-is from sources	Clean & standardized data	Business-Ready data
Objective	Traceability & Debugging	(Intermediate Layer) Prepare Data for Analysis	Provide data to be consumed for reporting & Analytics
Object Type	Tables	Tables	Views
Load Method	Full Load (Truncate & Insert)	Full Load (Truncate & Insert)	None
Data Transformation	None (as-is)	- Data **Cleaning** - Data **Standardization** - Data **Normalization** - **Derived Columns** - Data **Enrichment**	- Data **Integration** - Data **Aggregation** - **Business Logic & Rules**
Data Modeling	None (as-is)	None (as-is)	- **Start Schema** - **Aggregated Objects** - **Flat Tables**
Target Audience	- Data Engineers	- Data Analysts - Data Engineers	- Data Analysts - Business Users

What is Seperation of concerns

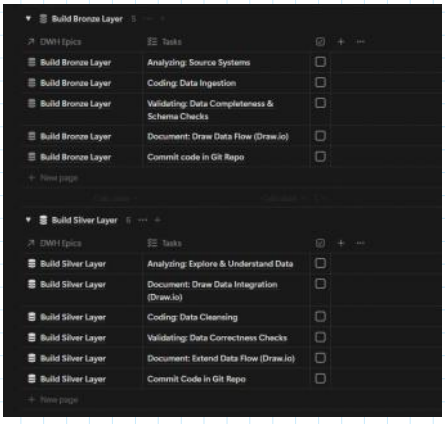




Draw the Architecture in Draw.io

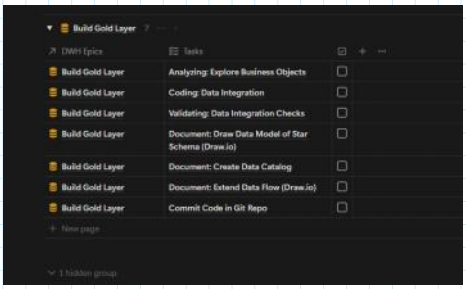


Prepare the detailed Tasks in Notion for All three layers



22-04-2025

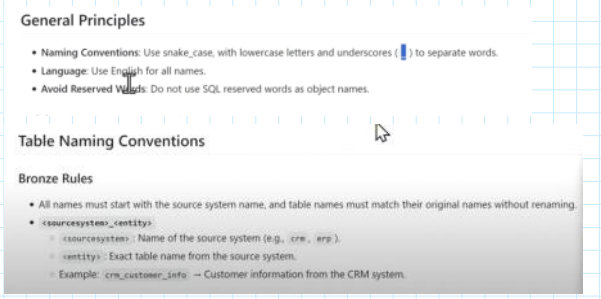
22-04-2025

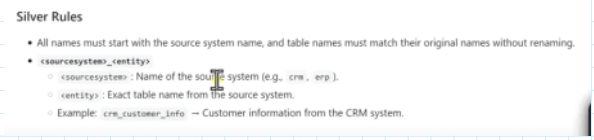


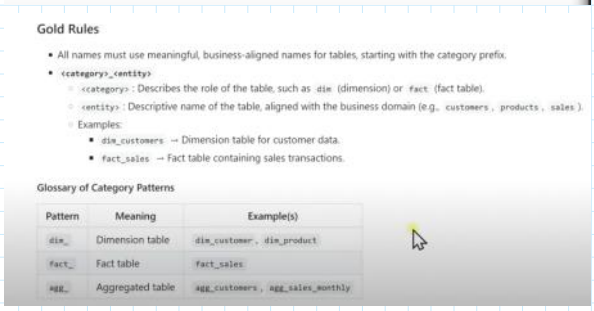

```
# **Naming Conventions**
This document outlines the naming conventions used for schemas, tables, views, columns, and other objects in the data warehouse.
## **Table of Contents**
1. [General Principles](#general-principles)
2. [Table Naming Conventions](#table-naming-conventions)
   - [Bronze Rules](#bronze-rules)
   - [Silver Rules](#silver-rules)
   - [Gold Rules](#gold-rules)
3. [Column Naming Conventions](#column-naming-conventions)
   - [Surrogate Keys](#surrogate-keys)
   - [Technical Columns](#technical-columns)
4. [Stored Procedure](#stored-procedure-naming-conventions)
---
## **General Principles**
- **Naming Conventions**: Use snake_case, with lowercase letters and underscores (`_`) to separate words.
- **Language**: Use English for all names.
- **Avoid Reserved Words**: Do not use SQL reserved words as object names.
## **Table Naming Conventions**
### **Bronze Rules**
- All names must start with the source system name, and table names must match their original names without renaming.
- **`<sourcesystem>_<entity>`**
  - `<sourcesystem>`: Name of the source system (e.g., `crm`, `erp`).
  - `<entity>`: Exact table name from the source system.
  - Example: `crm_customer_info` → Customer information from the CRM system.
### **Silver Rules**
- All names must start with the source system name, and table names must match their original names without renaming.
- **`<sourcesystem>_<entity>`**
  - `<sourcesystem>`: Name of the source system (e.g., `crm`, `erp`).
  - `<entity>`: Exact table name from the source system.
  - Example: `crm_customer_info` → Customer information from the CRM system.
### **Gold Rules**
- All names must use meaningful, business-aligned names for tables, starting with the category prefix.
- **`<category>_<entity>`**
  - `<category>`: Describes the role of the table, such as `dim` (dimension) or `fact` (fact table).
  - `<entity>`: Descriptive name of the table, aligned with the business domain (e.g., `customers`, `products`, `sales`).
  - Examples:
    - `dim_customers` → Dimension table for customer data.
    - `fact_sales` → Fact table containing sales transactions.
#### **Glossary of Category Patterns**
| Pattern     | Meaning                          | Example(s)                                  |
|-------------|----------------------------------|---------------------------------------------|
| `dim_`      | Dimension table                  | `dim_customer`, `dim_product`               |
| `fact_`     | Fact table                       | `fact_sales`                                |
| `report_`   | Report table                     | `report_customers`, `report_sales_monthly`  |
## **Column Naming Conventions**
### **Surrogate Keys**
- All primary keys in dimension tables must use the suffix `_key`.
- **`<table_name>_key`**
  - `<table_name>`: Refers to the name of the table or entity the key belongs to.
  - `_key`: A suffix indicating that this column is a surrogate key.
  - Example: `customer_key` → Surrogate key in the `dim_customers` table.

### **Technical Columns**
- All technical columns must start with the prefix `dwh_`, followed by a descriptive name indicating the column's purpose.
- **`dwh_<column_name>`**
  - `dwh`: Prefix exclusively for system-generated metadata.
  - `<column_name>`: Descriptive name indicating the column's purpose.
```

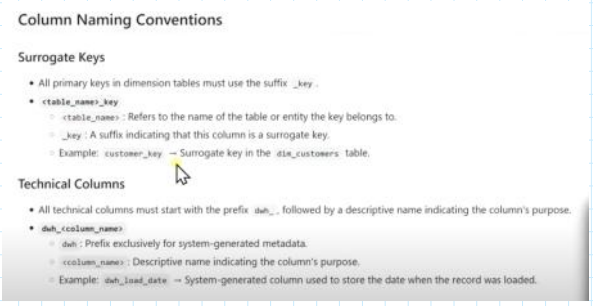
```
  - Example: `dwh_load_date` → System-generated column used to store the date when the record was loaded.

## **Stored Procedure**
- All stored procedures used for loading data must follow the naming pattern:
- **`load_<layer>`**.

  - `<layer>`: Represents the layer being loaded, such as `bronze`, `silver`, or `gold`.
  - Example:
    - `load_bronze` → Stored procedure for loading data into the Bronze layer.
    - `load_silver` → Stored procedure for loading data into the Silver layer.
```

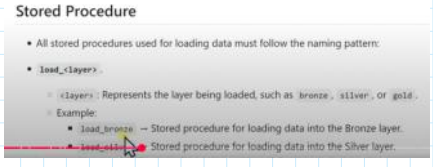
Snake Case -- Eg -- Table name -- customer_infoA











Create Github repo and basic folder structure

Update Readme

SQL Data Warehouse and Analytics Project
This end-to-end portfolio project demonstrates how to build a modern data warehouse and derive actionable insights using **Microsoft SQL Server**. It showcases industry best practices in **data engineering, modeling, and analytics**, aligned with the **Medallion Architecture** (Bronze → Silver → Gold).

Data Architecture – Medallion Layers

![architecture](https://github.com/user-attachments/assets/80ef95db-e901-401d-9c0d-b5e70f63827f)

Bronze Layer
- Ingests **raw data** from CSVs (ERP & CRM systems).
- No transformations applied.
- Acts as the staging layer for traceability.

Silver Layer
- Applies **data cleaning**, **standardization**, and **normalization**.
- Prepares data for analytical processing.

Gold Layer
- Creates **Star Schema** with **Fact and Dimension tables**.
- Final business-ready data for reporting and insights.

Project Overview

This project covers:

- **Data Engineering**: Design and build a SQL Server-based data warehouse.
- **ETL Pipelines**: Extract, clean, and load data through layered SQL workflows.
- **Data Modeling**: Star schema optimized for query performance.
- **Data Analytics**: SQL-based analysis for sales, product, and customer insights.

Skills Demonstrated

- SQL Development & Optimization
- ETL Pipeline Creation
- Data Modeling (Star Schema)
- Medallion Architecture Implementation
- Data Cleansing & Integration
- Analytical Reporting

Project Objectives

Data Warehouse Development
- Import and integrate ERP and CRM CSV datasets.
- Cleanse and resolve inconsistencies before analysis.
- Model unified schema for downstream analytics.
- Scope limited to latest snapshot (no historical tracking).

Analytics & Reporting
- Generate insights on:
  - Customer behavior
  - Product performance
  - Sales trends
- Use SQL queries for decision-ready metrics.

Repository Structure

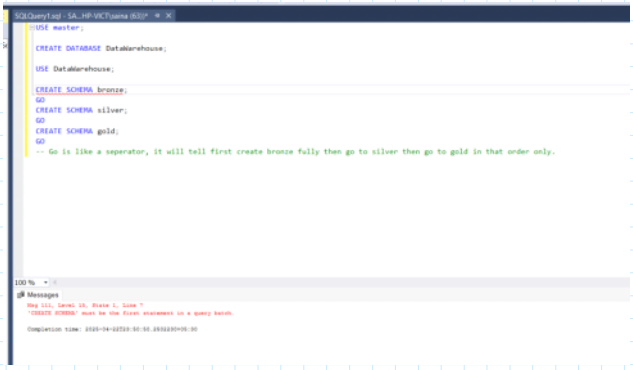
```
data-warehouse-project/
│
├── datasets/                  # Source CSV files (ERP and CRM)
├── docs/                      # Architecture, data flow, and modeling documents
│   ├── etl.drawio
│   ├── data_architecture.drawio
│   ├── data_flow.drawio
│   ├── data_models.drawio
│   ├── data_catalog.md
│   └── naming-conventions.md
│
├── scripts/                   # SQL scripts organized by layer
│   ├── bronze/                  # Raw data ingestion
│   ├── silver/                  # Data cleaning and transformation
│   └── gold/                    # Analytical model creation
│
```

```
├── scripts/                # SQL scripts organized by layer
│   ├── bronze/               # Raw data ingestion
│   ├── silver/               # Data cleaning and transformation
│   └── gold/                 # Analytical model creation
│
├── tests/                  # Data quality and validation scripts
├── README.md                 # Project overview and setup guide
├── LICENSE
├── .gitignore
└── requirements.txt          # (Optional) Any dependencies or tool references
```

🔒 Author

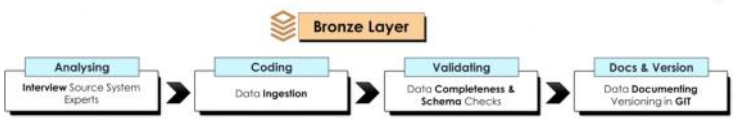
Sainadh Bahadursha
M.Sc in AI & ML | Data Science Enthusiast | Former Assistant Professor
[LinkedIn](https://www.linkedin.com/in/sainadh-bahadursha-67b121171/) | [GitHub](https://github.com/Sainadh-Bahadursha)

Acknowledgements
This project was inspired by the SQL Data Warehouse and Analytics course by [Data with Baraa](https://www.youtube.com/watch?v=9GVqKuTVANE) on YouTube.
Big thanks for making such quality content available to learners!

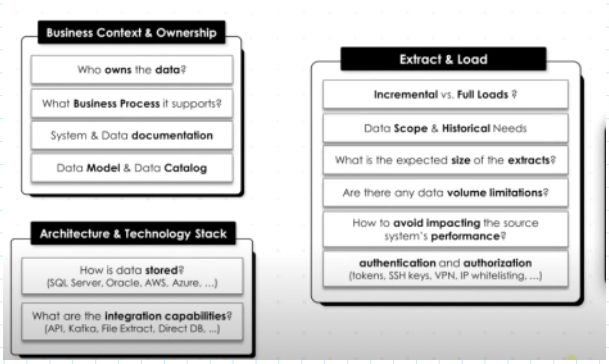



```
/*
=============================================================
Create Database and Schemas
=============================================================
Script Purpose:
    This script creates a new database named 'DataWarehouse' after checking if it already exists.
    If the database exists, it is dropped and recreated. Additionally, the script sets up three schemas
    within the database: 'bronze', 'silver', and 'gold'.

WARNING:
    Running this script will drop the entire 'DataWarehouse' database if it exists.
    All data in the database will be permanently deleted. Proceed with caution
    and ensure you have proper backups before running this script.
*/

USE master;
GO

-- Drop and recreate the 'DataWarehouse' database
IF EXISTS (SELECT 1 FROM sys.databases WHERE name = 'DataWarehouse')
BEGIN
    ALTER DATABASE DataWarehouse SET SINGLE_USER WITH ROLLBACK IMMEDIATE;
    DROP DATABASE DataWarehouse;
END;
GO

-- Create the 'DataWarehouse' database
CREATE DATABASE DataWarehouse;
GO

USE DataWarehouse;
GO

-- Create Schemas
CREATE SCHEMA bronze;
GO

CREATE SCHEMA silver;
GO

CREATE SCHEMA gold;
GO
```

Build the Bronze Layer



Ask expert many questions about SOURCE



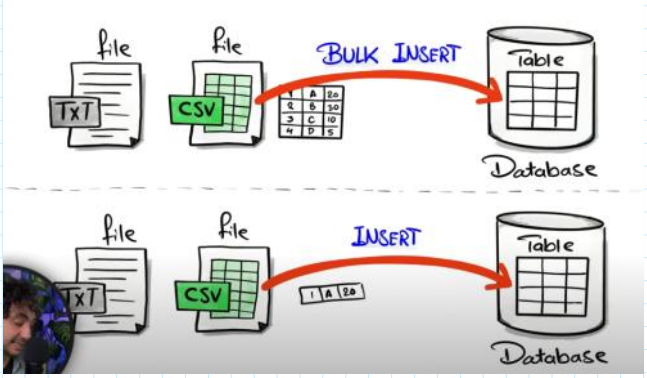
Data Ingestion (DDL)_

Create Table metadata

```
/*
===============================================================================
DDL Script: Create Bronze Tables
===============================================================================
Script Purpose:
    This script creates tables in the 'bronze' schema, dropping existing tables
    if they already exist.
        Run this script to re-define the DDL structure of 'bronze' Tables
===============================================================================
*/

IF OBJECT_ID('bronze.crm_cust_info', 'U') IS NOT NULL
    DROP TABLE bronze.crm_cust_info;
GO

CREATE TABLE bronze.crm_cust_info (
    cst_id              INT,
    cst_key             NVARCHAR(50),
    cst_firstname       NVARCHAR(50),
    cst_lastname        NVARCHAR(50),
    cst_marital_status  NVARCHAR(50),
    cst_gndr            NVARCHAR(50),
    cst_create_date     DATE
);
GO

IF OBJECT_ID('bronze.crm_prd_info', 'U') IS NOT NULL
    DROP TABLE bronze.crm_prd_info;
GO

CREATE TABLE bronze.crm_prd_info (
    prd_id       INT,
    prd_key      NVARCHAR(50),
    prd_nm       NVARCHAR(50),
    prd_cost     INT,
    prd_line     NVARCHAR(50),
    prd_start_dt DATETIME,
    prd_end_dt   DATETIME
);
GO

IF OBJECT_ID('bronze.crm_sales_details', 'U') IS NOT NULL
    DROP TABLE bronze.crm_sales_details;
GO

CREATE TABLE bronze.crm_sales_details (
    sls_ord_num  NVARCHAR(50),
    sls_prd_key  NVARCHAR(50),
    sls_cust_id  INT,
    sls_order_dt INT,
    sls_ship_dt  INT,
    sls_due_dt   INT,
    sls_sales    INT,
    sls_quantity INT,
    sls_price    INT
);
GO

IF OBJECT_ID('bronze.erp_loc_a101', 'U') IS NOT NULL
    DROP TABLE bronze.erp_loc_a101;
GO

CREATE TABLE bronze.erp_loc_a101 (
    cid    NVARCHAR(50),
    cntry  NVARCHAR(50)
);
GO

IF OBJECT_ID('bronze.erp_cust_az12', 'U') IS NOT NULL
    DROP TABLE bronze.erp_cust_az12;
GO

CREATE TABLE bronze.erp_cust_az12 (
    cid    NVARCHAR(50),
    bdate  DATE,
    gen    NVARCHAR(50)
);
GO
```

```
CREATE TABLE bronze.erp_cust_az12 (
    cid    NVARCHAR(50),
    bdate  DATE,
    gen    NVARCHAR(50)
);
GO

IF OBJECT_ID('bronze.erp_px_cat_g1v2', 'U') IS NOT NULL
    DROP TABLE bronze.erp_px_cat_g1v2;
GO

CREATE TABLE bronze.erp_px_cat_g1v2 (
    id          NVARCHAR(50),
    cat         NVARCHAR(50),
    subcat      NVARCHAR(50),
    maintenance NVARCHAR(50)
);
GO
```

Insert the data



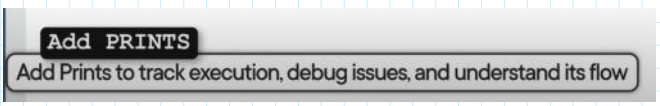
Basic code to data ingestion

```
-- Truncate and Insert (Full load)
-- crm_cust_info
TRUNCATE TABLE bronze.crm_cust_info;
BULK INSERT bronze.crm_cust_info
FROM 'C:\Users\saina\Desktop\DS_ML_AI\Projects
\SQL_Data_warehouse_project_Data_with_Baraa\sql-
data-warehouse-project\datasets\source_crm
\cust_info.csv'
WITH (
    FIRSTROW = 2,
    FIELDTERMINATOR = ',',
    TABLOCK
);

-- crm_prd_info
TRUNCATE TABLE bronze.crm_prd_info;
BULK INSERT bronze.crm_prd_info
FROM 'C:\Users\saina\Desktop\DS_ML_AI\Projects
\SQL_Data_warehouse_project_Data_with_Baraa\sql-
data-warehouse-project\datasets\source_crm\prd_info.csv'
WITH (
    FIRSTROW = 2,
    FIELDTERMINATOR = ',',
    TABLOCK
);

-- crm_sales_details
TRUNCATE TABLE bronze.crm_sales_details;
BULK INSERT bronze.crm_sales_details
FROM 'C:\Users\saina\Desktop\DS_ML_AI\Projects
\SQL_Data_warehouse_project_Data_with_Baraa\sql-
data-warehouse-project\datasets\source_crm
\sales_details.csv'
WITH (
    FIRSTROW = 2,
    FIELDTERMINATOR = ',',
    TABLOCK
);

-- erp_cust_az12
TRUNCATE TABLE bronze.erp_cust_az12;
BULK INSERT bronze.erp_cust_az12
FROM 'C:\Users\saina\Desktop\DS_ML_AI\Projects
\SQL_Data_warehouse_project_Data_with_Baraa\sql-
data-warehouse-project\datasets\source_erp
\CUST_AZ12.csv'
WITH (
    FIRSTROW = 2,
    FIELDTERMINATOR = ',',
    TABLOCK
);

-- erp_loc_a101
TRUNCATE TABLE bronze.erp_loc_a101;
BULK INSERT bronze.erp_loc_a101
FROM 'C:\Users\saina\Desktop\DS_ML_AI\Projects
\SQL_Data_warehouse_project_Data_with_Baraa\sql-
data-warehouse-project\datasets\source_erp
\LOC_A101.csv'
WITH (
    FIRSTROW = 2,
```

```
    FIRSTROW = 2,
    FIELDTERMINATOR = ',',
    TABLOCK
);

-- erp_loc_px_cat_g1v2
TRUNCATE TABLE bronze.erp_px_cat_g1v2;
BULK INSERT bronze.erp_px_cat_g1v2
FROM 'C:\Users\saina\Desktop\DS_ML_AI\Projects
\SQL_Data_warehouse_project_Data_with_Baraa\sql-
data-warehouse-project\datasets\source_erp
\PX_CAT_G1V2.csv'
WITH (
    FIRSTROW = 2,
    FIELDTERMINATOR = ',',
    TABLOCK
);
```

Create Stored procedure for total data ingestion

```
CREATE OR ALTER PROCEDURE bronze.load_bronze
AS
BEGIN
    -- Truncate and Insert (Full load)
    -- crm_cust_info
    TRUNCATE TABLE bronze.crm_cust_info;
    BULK INSERT bronze.crm_cust_info
    FROM 'C:\Users\saina\Desktop\DS_ML_AI\Projects
    \SQL_Data_warehouse_project_Data_with_Baraa\sql-
    data-warehouse-project\datasets\source_crm
    \cust_info.csv'
    WITH (
        FIRSTROW = 2,
        FIELDTERMINATOR = ',',
        TABLOCK
    );

    -- crm_prd_info
    TRUNCATE TABLE bronze.crm_prd_info;
    BULK INSERT bronze.crm_prd_info
    FROM 'C:\Users\saina\Desktop\DS_ML_AI\Projects
    \SQL_Data_warehouse_project_Data_with_Baraa\sql-
    data-warehouse-project\datasets\source_crm
    \prd_info.csv'
    WITH (
        FIRSTROW = 2,
        FIELDTERMINATOR = ',',
        TABLOCK
    );

    -- crm_sales_details
    TRUNCATE TABLE bronze.crm_sales_details;
    BULK INSERT bronze.crm_sales_details
    FROM 'C:\Users\saina\Desktop\DS_ML_AI\Projects
    \SQL_Data_warehouse_project_Data_with_Baraa\sql-
    data-warehouse-project\datasets\source_crm
    \sales_details.csv'
    WITH (
        FIRSTROW = 2,
        FIELDTERMINATOR = ',',
        TABLOCK
    );

    -- erp_cust_az12
    TRUNCATE TABLE bronze.erp_cust_az12;
    BULK INSERT bronze.erp_cust_az12
    FROM 'C:\Users\saina\Desktop\DS_ML_AI\Projects
    \SQL_Data_warehouse_project_Data_with_Baraa\sql-
    data-warehouse-project\datasets\source_erp
    \CUST_AZ12.csv'
    WITH (
        FIRSTROW = 2,
        FIELDTERMINATOR = ',',
        TABLOCK
    );

    -- erp_loc_a101
    TRUNCATE TABLE bronze.erp_loc_a101;
    BULK INSERT bronze.erp_loc_a101
    FROM 'C:\Users\saina\Desktop\DS_ML_AI\Projects
    \SQL_Data_warehouse_project_Data_with_Baraa\sql-
    data-warehouse-project\datasets\source_erp
    \LOC_A101.csv'
    WITH (
        FIRSTROW = 2,
        FIELDTERMINATOR = ',',
        TABLOCK
    );

    -- erp_loc_px_cat_g1v2
    TRUNCATE TABLE bronze.erp_px_cat_g1v2;
    BULK INSERT bronze.erp_px_cat_g1v2
    FROM 'C:\Users\saina\Desktop\DS_ML_AI\Projects
    \SQL_Data_warehouse_project_Data_with_Baraa\sql-
    data-warehouse-project\datasets\source_erp
    \PX_CAT_G1V2.csv'
    WITH (
        FIRSTROW = 2,
        FIELDTERMINATOR = ',',
        TABLOCK
    );
END
```




```
    EXEC bronze.load_bronze
```

Total Data ingestion into bronze layer

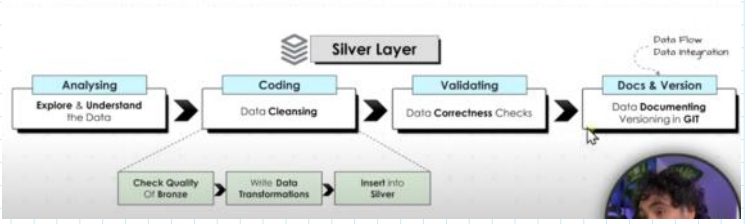
```
EXEC bronze.load_bronze;

/*
===============================================
===============================================
Stored Procedure: Load Bronze Layer (Source -> Bronze)
===============================================
===============================================
Script Purpose:
    This stored procedure loads data into the 'bronze'
schema from external CSV files.
    It performs the following actions:
    - Truncates the bronze tables before loading data.
    - Uses the `BULK INSERT` command to load data
from csv Files to bronze tables.

Parameters:
    None.
        This stored procedure does not accept any parameters
     or return any values.

Usage Example:
    EXEC bronze.load_bronze;
===============================================
===============================================
*/
CREATE OR ALTER PROCEDURE bronze.load_bronze
AS
BEGIN
    DECLARE @start_time DATETIME, @end_time
    DATETIME, @batch_start_time DATETIME,
    @batch_end_time DATETIME;
    BEGIN TRY
        -- Truncate and Insert (Full load)
        -- crm_cust_info
        SET @batch_start_time = GETDATE();
            PRINT
            '================================
            ====='
            PRINT 'Loading Bronze Layer'
            PRINT
            '================================
            ====='

            PRINT '---------------------------------------'
            PRINT 'Loading CRM Tables'
            PRINT '---------------------------------------'

            SET @start_time = GETDATE();
                PRINT '>> Truncating Table:
                bronze.crm_cust_info';
                TRUNCATE TABLE
                bronze.crm_cust_info;

                PRINT '>> Inserting Data Into:
                bronze.crm_cust_info';
                BULK INSERT bronze.crm_cust_info
                FROM 'C:\Users\saina\Desktop\DS_ML_AI
                \Projects
                \SQL_Data_warehouse_project_Data_with_
                Baraa\sql-data-warehouse-project\datasets
                \source_crm\cust_info.csv'
                WITH (
                    FIRSTROW = 2,
                    FIELDTERMINATOR = ',',
                    TABLOCK
                );
            SET @end_time = GETDATE();
            PRINT '>> Load Duration: ' +
            CAST(DATEDIFF(second, @start_time,
            @end_time) AS NVARCHAR) + ' seconds'

            SET @start_time = GETDATE();
                -- crm_prd_info
                PRINT '>> Truncating Table:
                bronze.crm_prd_info';
                TRUNCATE TABLE bronze.crm_prd_info;

                PRINT '>> Inserting Data Into:
                bronze.crm_prd_info';
                BULK INSERT bronze.crm_prd_info
                FROM 'C:\Users\saina\Desktop\DS_ML_AI
                \Projects
                \SQL_Data_warehouse_project_Data_with_
                Baraa\sql-data-warehouse-project\datasets
                \source_crm\prd_info.csv'
                WITH (
                    FIRSTROW = 2,
                    FIELDTERMINATOR = ',',
                    TABLOCK
                );
            SET @end_time = GETDATE();
            PRINT '>> Load Duration: ' +
            CAST(DATEDIFF(second, @start_time,
            @end_time) AS NVARCHAR) + ' seconds'


            SET @start_time = GETDATE();
                -- crm_sales_details
                PRINT '>> Truncating Table:
                bronze.crm_sales_details';
                TRUNCATE TABLE
                bronze.crm_sales_details;

                PRINT '>> Inserting Data Into:
                bronze.crm_sales_details';
                BULK INSERT bronze.crm_sales_details
                FROM 'C:\Users\saina\Desktop\DS_ML_AI
                \Projects
```

```
            PRINT '>> Inserting Data Into:
            bronze.crm_sales_details';
            BULK INSERT bronze.crm_sales_details
            FROM 'C:\Users\saina\Desktop\DS_ML_AI
            \Projects
            \SQL_Data_warehouse_project_Data_with_
            Baraa\sql-data-warehouse-project\datasets
            \source_crm\sales_details.csv'
            WITH (
                FIRSTROW = 2,
                FIELDTERMINATOR = ',',
                TABLOCK
            );
        SET @end_time = GETDATE();
        PRINT '>> Load Duration: ' +
        CAST(DATEDIFF(second, @start_time,
        @end_time) AS NVARCHAR) + ' seconds'

        PRINT '----------------------------------------'
        PRINT 'Loading ERP Tables'
        PRINT '----------------------------------------'

        SET @start_time = GETDATE();
            -- erp_cust_az12
            PRINT '>> Truncating Table:
            bronze.erp_cust_az12';
            TRUNCATE TABLE
            bronze.erp_cust_az12;

            PRINT '>> Inserting Data Into:
            bronze.erp_cust_az12';
            BULK INSERT bronze.erp_cust_az12
            FROM 'C:\Users\saina\Desktop\DS_ML_AI
            \Projects
            \SQL_Data_warehouse_project_Data_with_
            Baraa\sql-data-warehouse-project\datasets
            \source_erp\CUST_AZ12.csv'
            WITH (
                FIRSTROW = 2,
                FIELDTERMINATOR = ',',
                TABLOCK
            );
        SET @end_time = GETDATE();
        PRINT '>> Load Duration: ' +
        CAST(DATEDIFF(second, @start_time,
        @end_time) AS NVARCHAR) + ' seconds'

        SET @start_time = GETDATE();
        -- erp_loc_a101
            PRINT '>> Truncating Table:
            bronze.erp_loc_a101';
            TRUNCATE TABLE bronze.erp_loc_a101;

            PRINT '>> Inserting Data Into:
            bronze.erp_loc_a101';
            BULK INSERT bronze.erp_loc_a101
            FROM 'C:\Users\saina\Desktop\DS_ML_AI
            \Projects
            \SQL_Data_warehouse_project_Data_with_
            Baraa\sql-data-warehouse-project\datasets
            \source_erp\LOC_A101.csv'
            WITH (
                FIRSTROW = 2,
                FIELDTERMINATOR = ',',
                TABLOCK
            );
        SET @end_time = GETDATE();
        PRINT '>> Load Duration: ' +
        CAST(DATEDIFF(second, @start_time,
        @end_time) AS NVARCHAR) + ' seconds'

        SET @start_time = GETDATE();
            -- erp_px_cat_g1v2
            PRINT '>> Truncating Table:
            bronze.erp_px_cat_g1v2';
            TRUNCATE TABLE
            bronze.erp_px_cat_g1v2;

            PRINT '>> Inserting Data Into:
            bronze.erp_px_cat_g1v2';
            BULK INSERT bronze.erp_px_cat_g1v2
            FROM 'C:\Users\saina\Desktop\DS_ML_AI
            \Projects
            \SQL_Data_warehouse_project_Data_with_
            Baraa\sql-data-warehouse-project\datasets
            \source_erp\PX_CAT_G1V2.csv'
            WITH (
                FIRSTROW = 2,
                FIELDTERMINATOR = ',',
                TABLOCK
            );
        SET @end_time = GETDATE();
        PRINT '>> Load Duration: ' +
        CAST(DATEDIFF(second, @start_time,
        @end_time) AS NVARCHAR) + ' seconds'
    SET @batch_end_time = GETDATE();
    PRINT
    '======================================
    =='
    PRINT 'Loading Bronze Layer is Completed'
    PRINT ' - Total Load Duration: '+
    CAST(DATEDIFF(second,@batch_start_time,
    @batch_end_time) AS NVARCHAR) + ' seconds';
    PRINT
    '======================================
    =='
END TRY
-- CATCH will execute only if TRY fails to execute
BEGIN CATCH
    PRINT
    '======================================
    =='
    PRINT 'ERROR OCCURED DURING LOADING
    BRONZE LAYER'
```

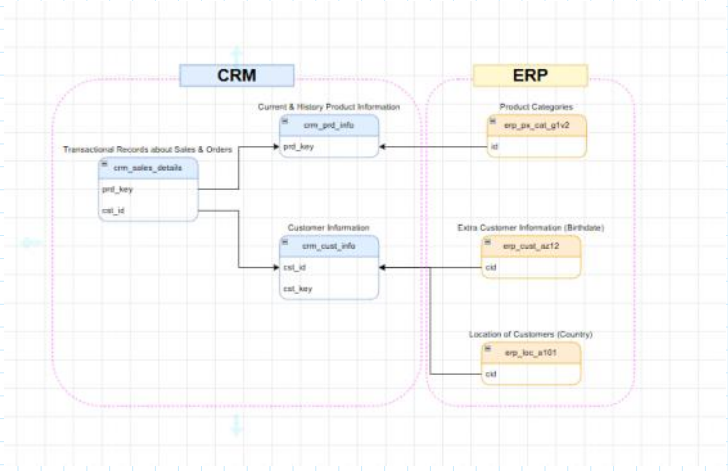
```
        PRINT
        '====================================
        =='
        PRINT 'ERROR OCCURED DURING LOADING
        BRONZE LAYER'
        PRINT 'ERROR MESSAGE' +
        ERROR_MESSAGE();
        PRINT 'ERROR MESSAGE' + CAST
        (ERROR_NUMBER() AS NVARCHAR);
        PRINT 'ERROR MESSAGE' + CAST
        (ERROR_STATE() AS NVARCHAR);
        PRINT
        '====================================
        =='
    END CATCH
END
```

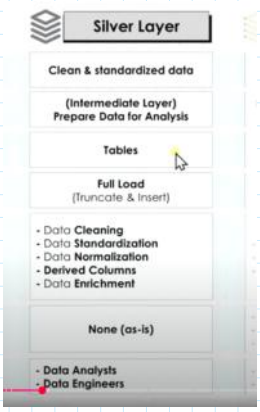
Push to GIT the Bronze layer all code

SILVER LAYER



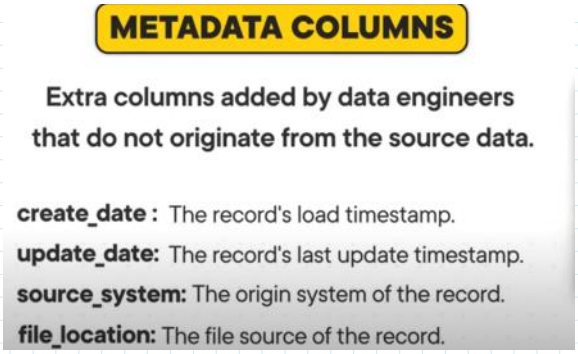
Draw Data integration draw.io





Build The DDL for Silver Layer

```
/*
=============================================
=============================================
DDL Script: Create Silver Tables
=============================================
=============================================
Script Purpose:
    This script creates tables in the 'silver' schema,
dropping existing tables
    if they already exist.
        Run this script to re-define the DDL structure of
    'silver' Tables
=============================================
=============================================
*/

IF OBJECT_ID('silver.crm_cust_info', 'U') IS NOT
NULL
    DROP TABLE silver.crm_cust_info;
GO

CREATE TABLE silver.crm_cust_info (
    cst_id              INT,
    cst_key             NVARCHAR(50),
    cst_firstname       NVARCHAR(50),
    cst_lastname        NVARCHAR(50),
    cst_marital_status  NVARCHAR(50),
    cst_gndr            NVARCHAR(50),
    cst_create_date     DATE
);
GO

IF OBJECT_ID('silver.crm_prd_info', 'U') IS NOT NULL
    DROP TABLE silver.crm_prd_info;
GO

CREATE TABLE silver.crm_prd_info (
    prd_id       INT,
    prd_key      NVARCHAR(50),
    prd_nm       NVARCHAR(50),
    prd_cost     INT,
    prd_line     NVARCHAR(50),
    prd_start_dt DATETIME,
    prd_end_dt   DATETIME
);
GO

IF OBJECT_ID('silver.crm_sales_details', 'U') IS NOT
NULL
    DROP TABLE silver.crm_sales_details;
GO

CREATE TABLE silver.crm_sales_details (
    sls_ord_num  NVARCHAR(50),
    sls_prd_key  NVARCHAR(50),
    sls_cust_id  INT,
    sls_order_dt INT,
    sls_ship_dt  INT,
    sls_due_dt   INT,
    sls_sales    INT,
    sls_quantity INT,
    sls_price    INT
);
GO

IF OBJECT_ID('silver.erp_loc_a101', 'U') IS NOT NULL
    DROP TABLE silver.erp_loc_a101;
GO

CREATE TABLE silver.erp_loc_a101 (
    cid    NVARCHAR(50),
    cntry  NVARCHAR(50)
);
GO

IF OBJECT_ID('silver.erp_cust_az12', 'U') IS NOT
NULL
    DROP TABLE silver.erp_cust_az12;
GO

CREATE TABLE silver.erp_cust_az12 (
    cid    NVARCHAR(50),
    bdate  DATE,
    gen    NVARCHAR(50)
);
GO

IF OBJECT_ID('silver.erp_px_cat_g1v2', 'U') IS NOT
NULL
    DROP TABLE silver.erp_px_cat_g1v2;
GO

CREATE TABLE silver.erp_px_cat_g1v2 (
    id          NVARCHAR(50),
    cat         NVARCHAR(50),
    subcat      NVARCHAR(50),
    maintenance  NVARCHAR(50)
);
GO
```

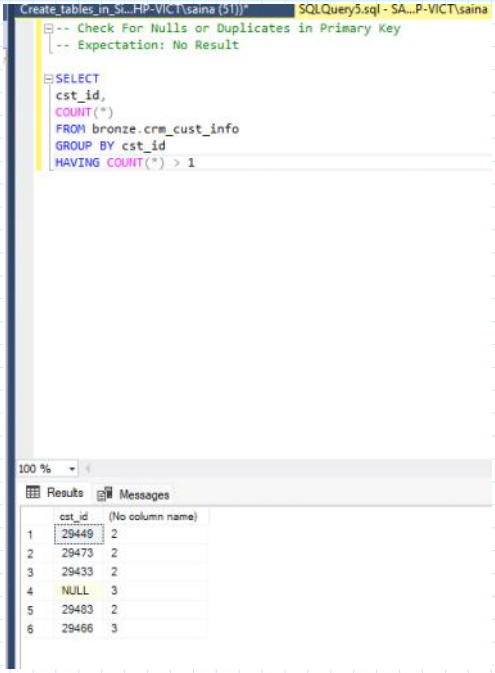


Add meta data to ddl of silver layer

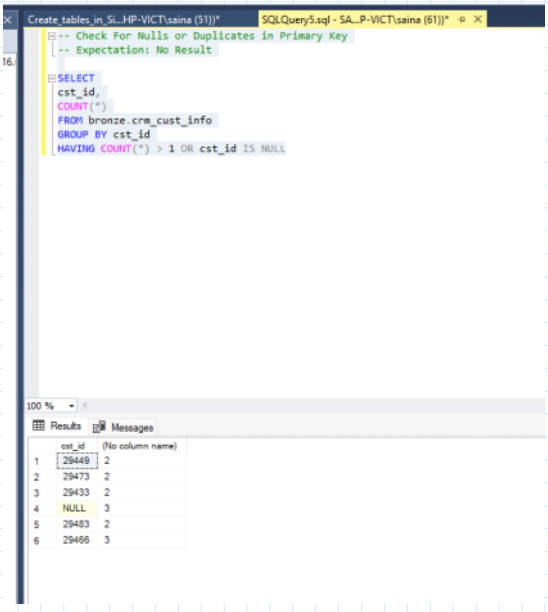
```
/*
=============================================
=====================================
DDL Script: Create Silver Tables
=============================================
=====================================
Script Purpose:
    This script creates tables in the 'silver' schema,
dropping existing tables
    if they already exist.
      Run this script to re-define the DDL structure of
    'silver' Tables
=============================================
=====================================
*/

IF OBJECT_ID('silver.crm_cust_info', 'U') IS NOT
NULL
    DROP TABLE silver.crm_cust_info;
GO

CREATE TABLE silver.crm_cust_info (
    cst_id              INT,
    cst_key             NVARCHAR(50),
    cst_firstname       NVARCHAR(50),
    cst_lastname        NVARCHAR(50),
    cst_marital_status  NVARCHAR(50),
    cst_gndr            NVARCHAR(50),
    cst_create_date     DATE,
     dwh_create_date DATETIME2 DEFAULT
     GETDATE()
);
GO

IF OBJECT_ID('silver.crm_prd_info', 'U') IS NOT NULL
    DROP TABLE silver.crm_prd_info;
GO

CREATE TABLE silver.crm_prd_info (
    prd_id       INT,
    prd_key      NVARCHAR(50),
    prd_nm       NVARCHAR(50),
    prd_cost     INT,
    prd_line     NVARCHAR(50),
    prd_start_dt DATETIME,
    prd_end_dt   DATETIME,
     dwh_create_date DATETIME2 DEFAULT
     GETDATE()
);
GO

IF OBJECT_ID('silver.crm_sales_details', 'U') IS NOT
NULL
    DROP TABLE silver.crm_sales_details;
GO

CREATE TABLE silver.crm_sales_details (
    sls_ord_num  NVARCHAR(50),
    sls_prd_key  NVARCHAR(50),
    sls_cust_id  INT,
    sls_order_dt INT,
    sls_ship_dt  INT,
    sls_due_dt   INT,
    sls_sales    INT,
    sls_quantity INT,
    sls_price    INT,
     dwh_create_date DATETIME2 DEFAULT
     GETDATE()
);
GO

IF OBJECT_ID('silver.erp_loc_a101', 'U') IS NOT NULL
    DROP TABLE silver.erp_loc_a101;
GO

CREATE TABLE silver.erp_loc_a101 (
    cid    NVARCHAR(50),
    cntry  NVARCHAR(50),
     dwh_create_date DATETIME2 DEFAULT
     GETDATE()
);
GO

IF OBJECT_ID('silver.erp_cust_az12', 'U') IS NOT
NULL
    DROP TABLE silver.erp_cust_az12;
GO

CREATE TABLE silver.erp_cust_az12 (
    cid    NVARCHAR(50),
```

```
    cid    NVARCHAR(50),
    bdate  DATE,
    gen    NVARCHAR(50),
     dwh_create_date DATETIME2 DEFAULT
     GETDATE()
);
GO

IF OBJECT_ID('silver.erp_px_cat_g1v2', 'U') IS NOT
NULL
    DROP TABLE silver.erp_px_cat_g1v2;
GO

CREATE TABLE silver.erp_px_cat_g1v2 (
    id          NVARCHAR(50),
    cat         NVARCHAR(50),
    subcat      NVARCHAR(50),
    maintenance NVARCHAR(50),
     dwh_create_date DATETIME2 DEFAULT
     GETDATE()
);
GO
```

Identify and solve Data quality issues



	cst_id	(No column name)
1	29449	2
2	29473	2
3	29433	2
4	NULL	3
5	29483	2
6	29466	3



	cst_id	(No column name)
1	29449	2
2	29473	2
3	29433	2
4	NULL	3
5	29483	2
6	29466	3

To remove duplicates using Rank by cst_create_date

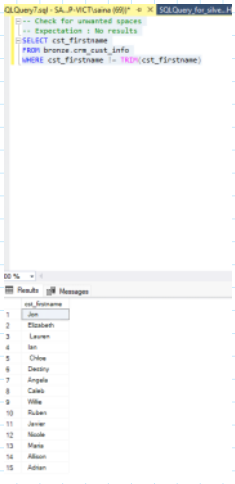
```
SELECT
*
```

To remove duplicates using Rank by cst_create_date

```
SELECT
*
FROM
(
    SELECT
    *,
    ROW_NUMBER() OVER (PARTITION BY cst_id
    ORDER BY cst_create_date DESC) as flag_last
    FROM bronze.crm_cust_info
) AS t
WHERE
    flag_last = 1 AND cst_id = 29466
```

Lot of String columns

Should check unwanted spaces



```
-- Check for unwanted spaces
-- Expectation : No results
SELECT cst_firstname
FROM bronze.crm_cust_info
WHERE cst_firstname != TRIM(cst_firstname);
```

```
-- Check for unwanted spaces
-- Expectation : No results
SELECT cst_lastname
FROM bronze.crm_cust_info
WHERE cst_lastname != TRIM(cst_lastname);
```

```
-- Check for unwanted spaces
-- Expectation : No results
SELECT cst_gndr
FROM bronze.crm_cust_info
WHERE cst_gndr != TRIM(cst_gndr);
```

```
-- Check for unwanted spaces
-- Expectation : No results
SELECT cst_marital_status
FROM bronze.crm_cust_info
WHERE cst_marital_status != TRIM(cst_marital_status);
```

```
-- Check for unwanted spaces
-- Expectation : No results
SELECT cst_key
FROM bronze.crm_cust_info
WHERE cst_key != TRIM(cst_key);
```

\Remove unwanted spaces

```
SELECT
    [cst_id]
    ,[cst_key]
    ,TRIM([cst_firstname])
    ,TRIM([cst_lastname])
    ,[cst_marital_status]
    ,[cst_gndr]
    ,[cst_create_date]
FROM
(
    SELECT
    *,
    ROW_NUMBER() OVER (PARTITION BY cst_id
    ORDER BY cst_create_date DESC) as flag_last
    FROM bronze.crm_cust_info
) AS t
WHERE
    flag_last = 1
```

Data standardization means abbreviating the categorical columns and
Handling null values

```
SELECT
    cst_id,
    cst_key,
    TRIM(cst_firstname) AS cst_firstname,
    TRIM(cst_lastname) AS cst_lastname,
    CASE
        WHEN UPPER(TRIM(cst_marital_status)) = 'S' THEN 'Single'
        WHEN UPPER(TRIM(cst_marital_status)) = 'M' THEN 'Married'
        ELSE 'n/a'
    END AS cst_marital_status, -- Normalize marital status values to
    readable format
    CASE
        WHEN UPPER(TRIM(cst_gndr)) = 'F' THEN 'Female'
        WHEN UPPER(TRIM(cst_gndr)) = 'M' THEN 'Male'
        ELSE 'n/a'
    END AS cst_gndr, -- Normalize gender values to readable format
    cst_create_date
FROM (
    SELECT
        *,
        ROW_NUMBER() OVER (PARTITION BY cst_id ORDER BY
```

```
        cst_create_date DESC) AS flag_last
    FROM bronze.crm_cust_info
    WHERE cst_id IS NOT NULL
) t
WHERE flag_last = 1;
```

M ake sure date are datetime dtypes

Different transformation done

Similar checks should be done on prd_info table also

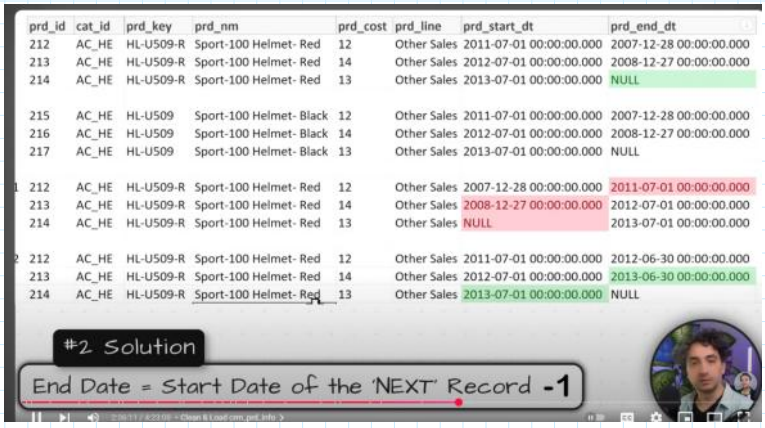
Code for getting cat_id

```
SELECT [prd_id]
      ,[prd_key]
      ,REPLACE(SUBSTRING(prd_key,1,5),'-','_') AS
cat_id
       ,[prd_nm]
      ,[prd_cost]
      ,[prd_line]
      ,[prd_start_dt]
      ,[prd_end_dt]
  FROM [DataWarehouse].[bronze].[crm_prd_info]
  WHERE REPLACE(SUBSTRING(prd_key,1,5),'-','_')
NOT IN
  (SELECT distinct id FROM bronze.erp_px_cat_g1v2)
```

Code join prd_info and sales_details

Cleaning the prd_key

```
SELECT [prd_id]
      ,[prd_key]
      ,REPLACE(SUBSTRING(prd_key,1,5),'-','_') AS
cat_id
       ,SUBSTRING(prd_key,7,LEN(prd_key)) AS
      prd_key
       ,[prd_nm]
      ,[prd_cost]
      ,[prd_line]
      ,[prd_start_dt]
      ,[prd_end_dt]
  FROM [DataWarehouse].[bronze].[crm_prd_info]
  WHERE SUBSTRING(prd_key,7,LEN(prd_key)) NOT
IN (
 SELECT sls_prd_key FROM bronze.crm_sales_details
WHERE sls_prd_key LIKE 'FK-16%')
```



Modified silver ddl

```
IF OBJECT_ID('silver.crm_prd_info', 'U') IS NOT NULL
    DROP TABLE silver.crm_prd_info;
GO

CREATE TABLE silver.crm_prd_info (
    prd_id       INT,
     cat_id              NVARCHAR(50),
    prd_key      NVARCHAR(50),
    prd_nm       NVARCHAR(50),
    prd_cost     INT,
    prd_line     NVARCHAR(50),
    prd_start_dt DATE,
    prd_end_dt   DATE,
     dwh_create_date DATETIME2 DEFAULT
     GETDATE()
);
GO
```

Code for stored procedure till now

```
SELECT [prd_id]
      ,[prd_key]
      ,REPLACE(SUBSTRING(prd_key,1,5),'-','_') AS
cat_id
       ,SUBSTRING(prd_key,7,LEN(prd_key)) AS
      prd_key
       ,[prd_nm]
      ,ISNULL([prd_cost],0) AS prd_cost
      ,[prd_line]
       ,CASE UPPER(TRIM(prd_line))
             WHEN 'M' THEN 'Mountain'
             WHEN 'R' THEN 'Road'
             WHEN 'S' THEN 'Other Sales'
             WHEN 'T' THEN 'Touring'
             ELSE 'n/a'
        END AS prd_line
      ,CAST(prd_start_dt AS DATE) AS prd_start_dt
       ,CAST(LEAD(prd_start_dt) OVER(PARTITION BY
      prd_key ORDER BY prd_start_dt)-1 AS DATE) AS
      prd_end_dt
```

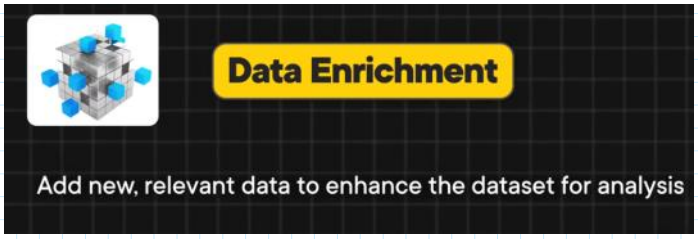
```
            ELSE 'n/a'
        END AS prd_line
    ,CAST(prd_start_dt AS DATE) AS prd_start_dt
     ,CAST(LEAD(prd_start_dt) OVER(PARTITION BY
    prd_key ORDER BY prd_start_dt)-1 AS DATE) AS
    prd_end_dt
FROM [DataWarehouse].[bronze].[crm_prd_info]
```

All Data quality checks for Silver

```
-- Check for Nulls or Duplicates in Primary Key
-- Expectation: No Result

SELECT
prd_id,
COUNT(*)
FROM silver.crm_prd_info
GROUP BY prd_id
HAVING COUNT(*) > 1 OR prd_id is NULL;


-- Check for unwanted spaces
-- Expectation: No result
SELECT prd_nm
FROM silver.crm_prd_info
WHERE prd_nm != TRIM(prd_nm);

-- Check for NULLs or Negative numbers
-- Expectation: No Results
SELECT prd_cost
FROM silver.crm_prd_info
WHERE prd_cost < 0 OR prd_cost IS NULL;

--Data standardization & consistency

SELECT Distinct prd_line
FROM silver.crm_prd_info

--Check for Invalid Date Orders
SELECT *
FROM silver.crm_prd_info
WHERE prd_end_dt < prd_start_dt

-- Final look
SELECT * FROM silver.crm_prd_info
```

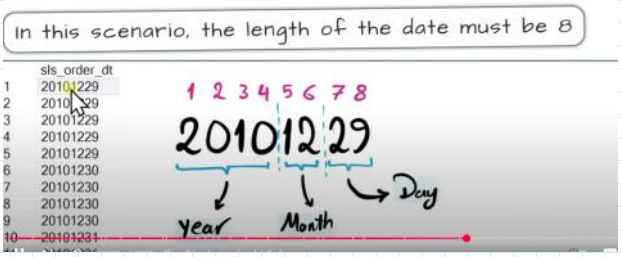
Data ingestion into crm_prd_info for silver

```
INSERT INTO silver.crm_prd_info(
    prd_id,
    cat_id,
    prd_key,
    prd_nm,
    prd_cost,
    prd_line,
    prd_start_dt,
    prd_end_dt
    )

SELECT [prd_id]
    ,REPLACE(SUBSTRING(prd_key,1,5),'-','_') AS
cat_id
     ,SUBSTRING(prd_key,7,LEN(prd_key)) AS
    prd_key
     ,[prd_nm]
    ,ISNULL([prd_cost],0) AS prd_cost
     ,CASE UPPER(TRIM(prd_line))
            WHEN 'M' THEN 'Mountain'
            WHEN 'R' THEN 'Road'
            WHEN 'S' THEN 'Other Sales'
            WHEN 'T' THEN 'Touring'
            ELSE 'n/a'
        END AS prd_line
    ,CAST(prd_start_dt AS DATE) AS prd_start_dt
     ,CAST(LEAD(prd_start_dt) OVER(PARTITION BY
    prd_key ORDER BY prd_start_dt)-1 AS DATE) AS
    prd_end_dt
  FROM [DataWarehouse].[bronze].[crm_prd_info]
```

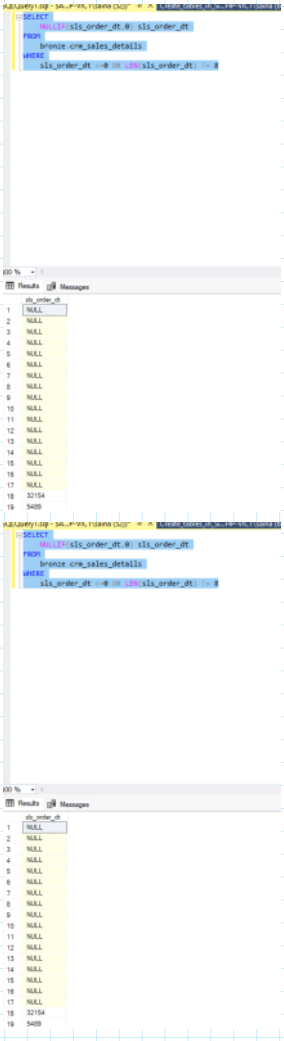


Crm_sales_details

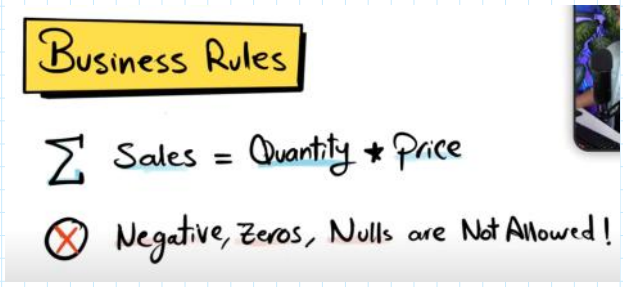
Convert the integer to date in sls_order_dt column

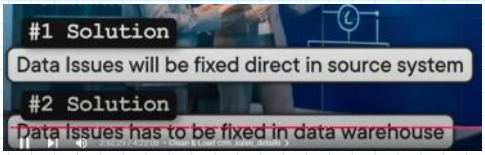


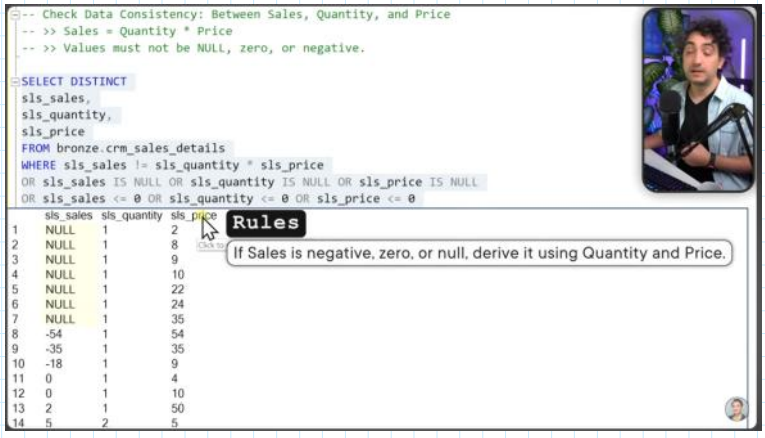
Length of date should be 8



Order Date must always be earlier than the Shipping Date or Due Date



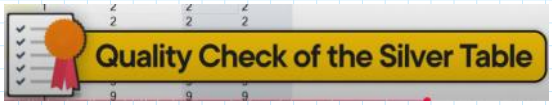




	sls_sales	sls_quantity	sls_price
1	NULL	1	2
2	NULL	1	8
3	NULL	1	9
4	NULL	1	10
5	NULL	1	22
6	NULL	1	24
7	NULL	1	35
8	-54	1	54
9	-35	1	35
10	-18	1	9
11	0	1	4
12	0	1	10
13	2	1	50
14	5	2	5

```
TRUNCATE TABLE silver.crm_sales_details;
INSERT INTO silver.crm_sales_details (
    sls_ord_num,
    sls_prd_key,
    sls_cust_id,
    sls_order_dt,
    sls_ship_dt,
    sls_due_dt,
    sls_sales,
    sls_quantity,
    sls_price
)
SELECT
    sls_ord_num,
    sls_prd_key,
    sls_cust_id,
    CASE
        WHEN sls_order_dt = 0 OR LEN(sls_order_dt) !=
        8 THEN NULL
        ELSE CAST(CAST(sls_order_dt AS VARCHAR)
        AS DATE)
    END AS sls_order_dt,
    CASE
        WHEN sls_ship_dt = 0 OR LEN(sls_ship_dt) != 8
        THEN NULL
        ELSE CAST(CAST(sls_ship_dt AS VARCHAR)
        AS DATE)
    END AS sls_ship_dt,
    CASE
        WHEN sls_due_dt = 0 OR LEN(sls_due_dt) != 8
        THEN NULL
        ELSE CAST(CAST(sls_due_dt AS VARCHAR)
        AS DATE)
    END AS sls_due_dt,
    CASE
        WHEN sls_sales IS NULL OR sls_sales <= 0 OR
        sls_sales != sls_quantity * ABS(sls_price)
            THEN sls_quantity * ABS(sls_price)
        ELSE sls_sales
    END AS sls_sales, -- Recalculate sales if original
    value is missing or incorrect
    sls_quantity,
    CASE
```

```
        ELSE sls_sales
    END AS sls_sales, -- Recalculate sales if original
    value is missing or incorrect
    sls_quantity,
    CASE
        WHEN sls_price IS NULL OR sls_price <= 0
            THEN sls_sales / NULLIF(sls_quantity, 0)
        ELSE sls_price  -- Derive price if original value is
        invalid
    END AS sls_price
FROM bronze.crm_sales_details;
```



Create silver erp_cust_az

```
SELECT
cid,
bdate,
gen
FROM bronze.erp_cust_az12
WHERE cid LIKE '%AW00011000%'

SELECT * FROM [silver].[crm_cust_info];
```

	cst_id	cst_key	cst_firstname	cst_lastname	cst_marital_status	cst_gndr	cst_create_date	dwh_
5	NAS	AW00011004	1979-08-05	Female				
6	NAS	AW00011005	1976-08-01	Male				
7	NAS	AW00011006	1976-12-02	Female				
8	NAS	AW00011007	1969-11-06	Male				
1	11000	AW00011000	Jon	Yang	Married	Male	2025-10-06	2024
2	11001	AW00011001	Eugene	Huang	Single	Male	2025-10-06	2024
3	11002	AW00011002	Ruben	Torres	Married	Male	2025-10-06	2024
4	11003	AW00011003	Christy	Zhu	Single	Female	2025-10-06	2024
5	11004	AW00011004	Elizabeth	Johnson	Single	Female	2025-10-06	2024

```
TRUNCATE TABLE silver.erp_cust_az12;
INSERT INTO silver.erp_cust_az12 (
    cid,
    bdate,
    gen
)
SELECT
    CASE
        WHEN cid LIKE 'NAS%' THEN
        SUBSTRING(cid, 4, LEN(cid)) -- Remove 'NAS'
        prefix if present
        ELSE cid
    END AS cid,
    CASE
        WHEN bdate > GETDATE() THEN NULL
        ELSE bdate
    END AS bdate, -- Set future birthdates to NULL
    CASE
        WHEN UPPER(TRIM(gen)) IN ('F', 'FEMALE')
        THEN 'Female'
        WHEN UPPER(TRIM(gen)) IN ('M', 'MALE')
        THEN 'Male'
        ELSE 'n/a'
    END AS gen -- Normalize gender values and handle
    unknown cases
FROM bronze.erp_cust_az12;
```

Create erp_loc_a101

```
TRUNCATE TABLE silver.erp_loc_a101;
INSERT INTO silver.erp_loc_a101 (
    cid,
    cntry
)
SELECT
    REPLACE(cid, '-', '') AS cid,
    CASE
        WHEN TRIM(cntry) = 'DE' THEN 'Germany'
        WHEN TRIM(cntry) IN ('US', 'USA') THEN
        'United States'
        WHEN TRIM(cntry) = '' OR cntry IS NULL
        THEN 'n/a'
        ELSE TRIM(cntry)
    END AS cntry -- Normalize and Handle missing or
    blank country codes
FROM bronze.erp_loc_a101;
```

```
TRUNCATE TABLE silver.erp_px_cat_g1v2;
INSERT INTO silver.erp_px_cat_g1v2 (
    id,
    cat,
    subcat,
    maintenance
)
```

```
)
SELECT
    id,
    cat,
    subcat,
    maintenance
FROM bronze.erp_px_cat_g1v2;
```

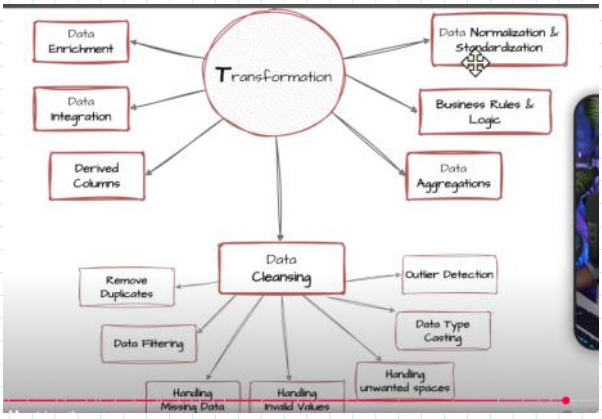

```
/*
===========================================
===================================
DDL Script: Create Silver Tables
===========================================
===================================
Script Purpose:
    This script creates tables in the 'silver' schema,
dropping existing tables
    if they already exist.
        Run this script to re-define the DDL structure of
     'bronze' Tables
===========================================
===================================
*/

IF OBJECT_ID('silver.crm_cust_info', 'U') IS NOT
NULL
    DROP TABLE silver.crm_cust_info;
GO

CREATE TABLE silver.crm_cust_info (
    cst_id              INT,
    cst_key             NVARCHAR(50),
    cst_firstname       NVARCHAR(50),
    cst_lastname        NVARCHAR(50),
    cst_marital_status NVARCHAR(50),
    cst_gndr            NVARCHAR(50),
    cst_create_date     DATE,
    dwh_create_date     DATETIME2 DEFAULT
GETDATE()
);
GO

IF OBJECT_ID('silver.crm_prd_info', 'U') IS NOT NULL
    DROP TABLE silver.crm_prd_info;
GO

CREATE TABLE silver.crm_prd_info (
    prd_id          INT,
    cat_id          NVARCHAR(50),
    prd_key         NVARCHAR(50),
    prd_nm          NVARCHAR(50),
    prd_cost        INT,
    prd_line        NVARCHAR(50),
    prd_start_dt    DATE,
    prd_end_dt      DATE,
    dwh_create_date DATETIME2 DEFAULT
GETDATE()
);
GO

IF OBJECT_ID('silver.crm_sales_details', 'U') IS NOT
NULL
    DROP TABLE silver.crm_sales_details;
GO

CREATE TABLE silver.crm_sales_details (
    sls_ord_num     NVARCHAR(50),
    sls_prd_key     NVARCHAR(50),
    sls_cust_id     INT,
    sls_order_dt    DATE,
    sls_ship_dt     DATE,
    sls_due_dt      DATE,
    sls_sales       INT,
    sls_quantity    INT,
    sls_price       INT,
    dwh_create_date DATETIME2 DEFAULT
GETDATE()
);
GO

IF OBJECT_ID('silver.erp_loc_a101', 'U') IS NOT NULL
```

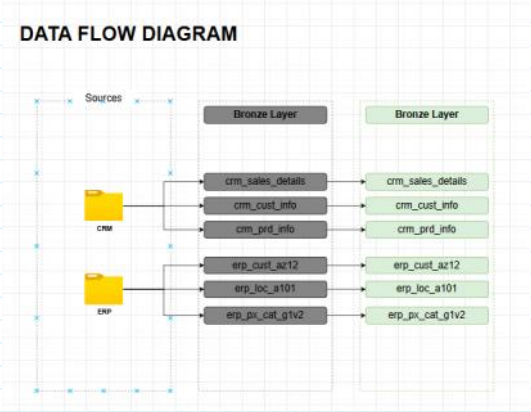
```
    dwh_create_date DATETIME2 DEFAULT
GETDATE()
);
GO

IF OBJECT_ID('silver.erp_loc_a101', 'U') IS NOT NULL
    DROP TABLE silver.erp_loc_a101;
GO

CREATE TABLE silver.erp_loc_a101 (
    cid             NVARCHAR(50),
    cntry           NVARCHAR(50),
    dwh_create_date DATETIME2 DEFAULT
GETDATE()
);
GO

IF OBJECT_ID('silver.erp_cust_az12', 'U') IS NOT
NULL
    DROP TABLE silver.erp_cust_az12;
GO

CREATE TABLE silver.erp_cust_az12 (
    cid             NVARCHAR(50),
    bdate           DATE,
    gen             NVARCHAR(50),
    dwh_create_date DATETIME2 DEFAULT
GETDATE()
);
GO

IF OBJECT_ID('silver.erp_px_cat_g1v2', 'U') IS NOT
NULL
    DROP TABLE silver.erp_px_cat_g1v2;
GO

CREATE TABLE silver.erp_px_cat_g1v2 (
    id              NVARCHAR(50),
    cat             NVARCHAR(50),
    subcat          NVARCHAR(50),
    maintenance     NVARCHAR(50),
    dwh_create_date DATETIME2 DEFAULT
GETDATE()
);
GO




/*
=============================================
=====================================
Stored Procedure: Load Silver Layer (Bronze -> Silver)
=============================================
=====================================
Script Purpose:
    This stored procedure performs the ETL (Extract,
Transform, Load) process to
    populate the 'silver' schema tables from the 'bronze'
schema.
      Actions Performed:
          - Truncates Silver tables.
          - Inserts transformed and cleansed data from
          Bronze into Silver tables.

Parameters:
    None.
       This stored procedure does not accept any parameters
      or return any values.

Usage Example:
    EXEC Silver.load_silver;
=============================================
=====================================
*/

CREATE OR ALTER PROCEDURE silver.load_silver
AS
BEGIN
    DECLARE @start_time DATETIME, @end_time
DATETIME, @batch_start_time DATETIME,
@batch_end_time DATETIME;
    BEGIN TRY
        SET @batch_start_time = GETDATE();
        PRINT
'==========================================
=======';
        PRINT 'Loading Silver Layer';
        PRINT
'==========================================
=======';

            PRINT '----------------------------------------------';
            PRINT 'Loading CRM Tables';
            PRINT '----------------------------------------------';

            -- Loading silver.crm_cust_info
        SET @start_time = GETDATE();
            PRINT '>> Truncating Table:
            silver.crm_cust_info';
            TRUNCATE TABLE silver.crm_cust_info;
            PRINT '>> Inserting Data Into:
            silver.crm_cust_info';
            INSERT INTO silver.crm_cust_info (
                cst_id,
                cst_key,
                cst_firstname,
                cst_lastname,
                cst_marital_status,
                cst_gndr,
                cst_create_date
            )
            SELECT
                cst_id,
                cst_key,
```

```
            cst_key,
            TRIM(cst_firstname) AS cst_firstname,
            TRIM(cst_lastname) AS cst_lastname,
            CASE
                WHEN UPPER(TRIM(cst_marital_status))
                = 'S' THEN 'Single'
                WHEN UPPER(TRIM(cst_marital_status))
                = 'M' THEN 'Married'
                ELSE 'n/a'
            END AS cst_marital_status, -- Normalize
            marital status values to readable format
            CASE
                WHEN UPPER(TRIM(cst_gndr)) = 'F'
                THEN 'Female'
                WHEN UPPER(TRIM(cst_gndr)) = 'M'
                THEN 'Male'
                ELSE 'n/a'
            END AS cst_gndr, -- Normalize gender values
            to readable format
            cst_create_date
        FROM (
            SELECT
                *,
                ROW_NUMBER() OVER (PARTITION
                BY cst_id ORDER BY cst_create_date
                DESC) AS flag_last
            FROM bronze.crm_cust_info
            WHERE cst_id IS NOT NULL
        ) t
        WHERE flag_last = 1; -- Select the most recent
        record per customer
        SET @end_time = GETDATE();
        PRINT '>> Load Duration: ' +
CAST(DATEDIFF(SECOND, @start_time, @end_time)
AS NVARCHAR) + ' seconds';
        PRINT '>> ----------';

        -- Loading silver.crm_prd_info
        SET @start_time = GETDATE();
        PRINT '>> Truncating Table: silver.crm_prd_info';
        TRUNCATE TABLE silver.crm_prd_info;
        PRINT '>> Inserting Data Into:
        silver.crm_prd_info';
        INSERT INTO silver.crm_prd_info (
            prd_id,
            cat_id,
            prd_key,
            prd_nm,
            prd_cost,
            prd_line,
            prd_start_dt,
            prd_end_dt
        )
        SELECT
            prd_id,
            REPLACE(SUBSTRING(prd_key, 1, 5), '-', '_')
            AS cat_id, -- Extract category ID
            SUBSTRING(prd_key, 7, LEN(prd_key)) AS
            prd_key,        -- Extract product key
            prd_nm,
            ISNULL(prd_cost, 0) AS prd_cost,
            CASE
                WHEN UPPER(TRIM(prd_line)) = 'M'
                THEN 'Mountain'
                WHEN UPPER(TRIM(prd_line)) = 'R'
                THEN 'Road'
                WHEN UPPER(TRIM(prd_line)) = 'S'
                THEN 'Other Sales'
                WHEN UPPER(TRIM(prd_line)) = 'T'
                THEN 'Touring'
                ELSE 'n/a'
            END AS prd_line, -- Map product line codes to
            descriptive values
            CAST(prd_start_dt AS DATE) AS
            prd_start_dt,
            CAST(
                LEAD(prd_start_dt) OVER (PARTITION
                BY prd_key ORDER BY prd_start_dt) - 1
                AS DATE
            ) AS prd_end_dt -- Calculate end date as one
            day before the next start date
        FROM bronze.crm_prd_info;
        SET @end_time = GETDATE();
        PRINT '>> Load Duration: ' +
CAST(DATEDIFF(SECOND, @start_time, @end_time)
AS NVARCHAR) + ' seconds';
        PRINT '>> ----------';

        -- Loading crm_sales_details
        SET @start_time = GETDATE();
        PRINT '>> Truncating Table:
        silver.crm_sales_details';
        TRUNCATE TABLE silver.crm_sales_details;
        PRINT '>> Inserting Data Into:
        silver.crm_sales_details';
        INSERT INTO silver.crm_sales_details (
            sls_ord_num,
            sls_prd_key,
            sls_cust_id,
            sls_order_dt,
            sls_ship_dt,
            sls_due_dt,
            sls_sales,
            sls_quantity,
            sls_price
        )
        SELECT
            sls_ord_num,
            sls_prd_key,
            sls_cust_id,
            CASE
                WHEN sls_order_dt = 0 OR
                LEN(sls_order_dt) != 8 THEN NULL
                ELSE CAST(CAST(sls_order_dt AS
                VARCHAR) AS DATE)
```

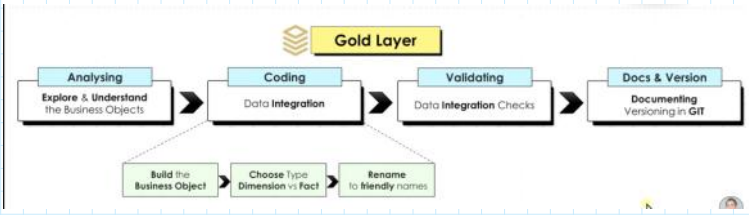
```
                    VARCHAR) AS DATE)
                END AS sls_order_dt,
                CASE
                    WHEN sls_ship_dt = 0 OR
                    LEN(sls_ship_dt) != 8 THEN NULL
                    ELSE CAST(CAST(sls_ship_dt AS
                    VARCHAR) AS DATE)
                END AS sls_ship_dt,
                CASE
                    WHEN sls_due_dt = 0 OR
                    LEN(sls_due_dt) != 8 THEN NULL
                    ELSE CAST(CAST(sls_due_dt AS
                    VARCHAR) AS DATE)
                END AS sls_due_dt,
                CASE
                    WHEN sls_sales IS NULL OR sls_sales <=
                    0 OR sls_sales != sls_quantity *
                    ABS(sls_price)
                        THEN sls_quantity * ABS(sls_price)
                    ELSE sls_sales
                END AS sls_sales, -- Recalculate sales if
                original value is missing or incorrect
                sls_quantity,
                CASE
                    WHEN sls_price IS NULL OR sls_price <=
                    0
                        THEN sls_sales / NULLIF(sls_quantity,
                        0)
                    ELSE sls_price  -- Derive price if original
                    value is invalid
                END AS sls_price
            FROM bronze.crm_sales_details;
            SET @end_time = GETDATE();
            PRINT '>> Load Duration: ' +
        CAST(DATEDIFF(SECOND, @start_time, @end_time)
        AS NVARCHAR) + ' seconds';
            PRINT '>> -------------';

            -- Loading erp_cust_az12
            SET @start_time = GETDATE();
            PRINT '>> Truncating Table:
            silver.erp_cust_az12';
            TRUNCATE TABLE silver.erp_cust_az12;
            PRINT '>> Inserting Data Into:
            silver.erp_cust_az12';
            INSERT INTO silver.erp_cust_az12 (
                cid,
                bdate,
                gen
            )
            SELECT
                CASE
                    WHEN cid LIKE 'NAS%' THEN
                    SUBSTRING(cid, 4, LEN(cid)) -- Remove
                    'NAS' prefix if present
                    ELSE cid
                END AS cid,
                CASE
                    WHEN bdate > GETDATE() THEN NULL
                    ELSE bdate
                END AS bdate, -- Set future birthdates to
                NULL
                CASE
                    WHEN UPPER(TRIM(gen)) IN ('F',
                    'FEMALE') THEN 'Female'
                    WHEN UPPER(TRIM(gen)) IN ('M',
                    'MALE') THEN 'Male'
                    ELSE 'n/a'
                END AS gen -- Normalize gender values and
                handle unknown cases
            FROM bronze.erp_cust_az12;
            SET @end_time = GETDATE();
            PRINT '>> Load Duration: ' +
        CAST(DATEDIFF(SECOND, @start_time, @end_time)
        AS NVARCHAR) + ' seconds';
            PRINT '>> -------------';

            PRINT '------------------------------------------------';
            PRINT 'Loading ERP Tables';
            PRINT '------------------------------------------------';

            -- Loading erp_loc_a101
            SET @start_time = GETDATE();
            PRINT '>> Truncating Table: silver.erp_loc_a101';
            TRUNCATE TABLE silver.erp_loc_a101;
            PRINT '>> Inserting Data Into:
            silver.erp_loc_a101';
            INSERT INTO silver.erp_loc_a101 (
                cid,
                cntry
            )
            SELECT
                REPLACE(cid, '-', '') AS cid,
                CASE
                    WHEN TRIM(cntry) = 'DE' THEN
                    'Germany'
                    WHEN TRIM(cntry) IN ('US', 'USA')
                    THEN 'United States'
                    WHEN TRIM(cntry) = '' OR cntry IS NULL
                    THEN 'n/a'
                    ELSE TRIM(cntry)
                END AS cntry -- Normalize and Handle
                missing or blank country codes
            FROM bronze.erp_loc_a101;
            SET @end_time = GETDATE();
            PRINT '>> Load Duration: ' +
        CAST(DATEDIFF(SECOND, @start_time, @end_time)
        AS NVARCHAR) + ' seconds';
            PRINT '>> -------------';

            -- Loading erp_px_cat_g1v2
            SET @start_time = GETDATE();
            PRINT '>> Truncating Table:
            silver.erp_px_cat_g1v2';
            TRUNCATE TABLE silver.erp_px_cat_g1v2;
```

```
        -- Loading erp_px_cat_g1v2
        SET @start_time = GETDATE();
        PRINT '>> Truncating Table:
        silver.erp_px_cat_g1v2';
        TRUNCATE TABLE silver.erp_px_cat_g1v2;
        PRINT '>> Inserting Data Into:
        silver.erp_px_cat_g1v2';
        INSERT INTO silver.erp_px_cat_g1v2 (
            id,
            cat,
            subcat,
            maintenance
        )
        SELECT
            id,
            cat,
            subcat,
            maintenance
        FROM bronze.erp_px_cat_g1v2;
        SET @end_time = GETDATE();
        PRINT '>> Load Duration: ' +
        CAST(DATEDIFF(SECOND, @start_time,
        @end_time) AS NVARCHAR) + ' seconds';
      PRINT '>> ------------';

        SET @batch_end_time = GETDATE();
        PRINT
        '=================================
        ======'
        PRINT 'Loading Silver Layer is Completed';
      PRINT '   - Total Load Duration: ' +
CAST(DATEDIFF(SECOND, @batch_start_time,
@batch_end_time) AS NVARCHAR) + ' seconds';
        PRINT
        '=================================
        ======'

    END TRY
    BEGIN CATCH
        PRINT
        '=================================
        ======'
        PRINT 'ERROR OCCURED DURING LOADING
        BRONZE LAYER'
        PRINT 'Error Message' + ERROR_MESSAGE();
        PRINT 'Error Message' + CAST
        (ERROR_NUMBER() AS NVARCHAR);
        PRINT 'Error Message' + CAST
        (ERROR_STATE() AS NVARCHAR);
        PRINT
        '=================================
        ======'
    END CATCH
END
```



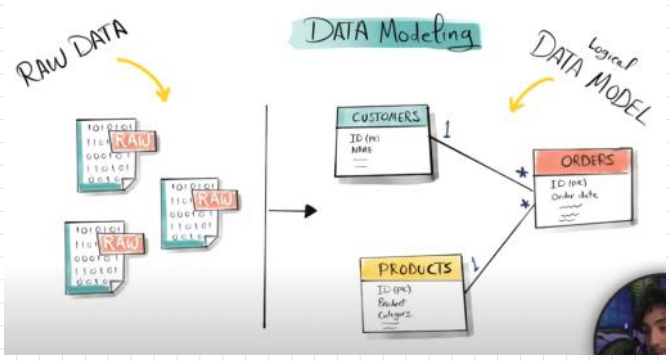


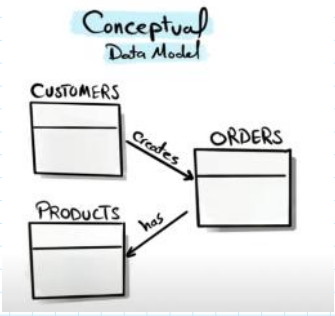
Building GOLD LAYER

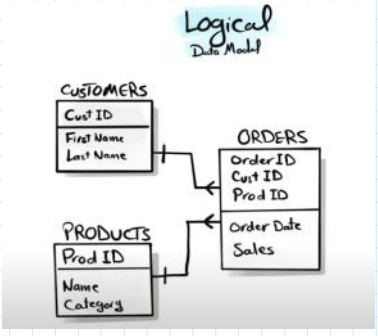


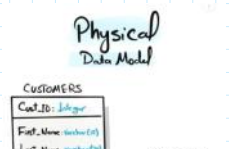
Theory

What is Data Modeling?

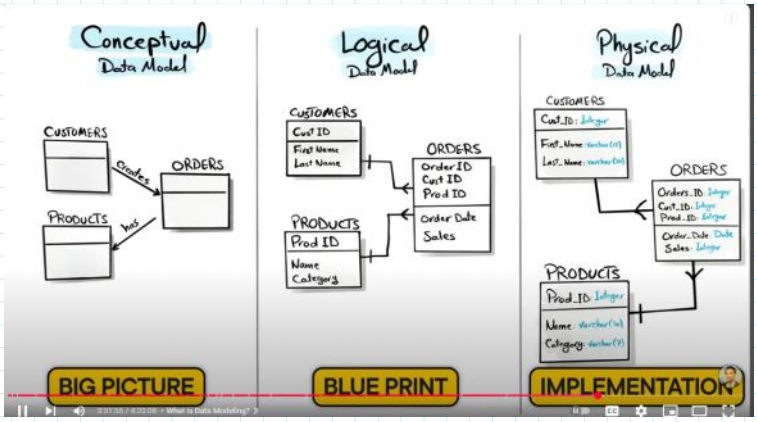




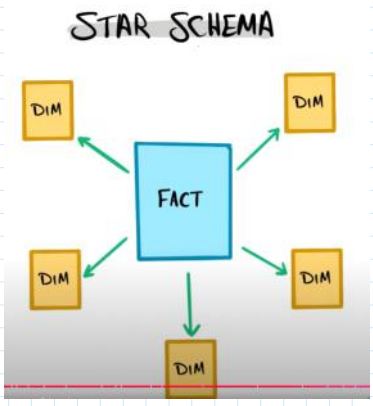


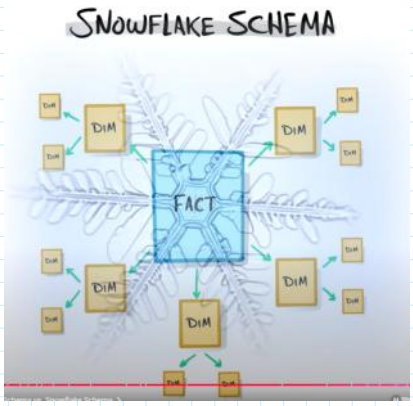


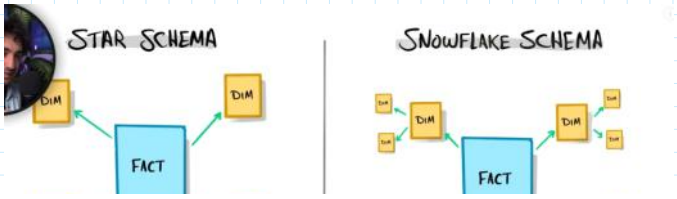

Data Model
CUSTOMERS
Cust_ID: Integer
First_Name: Varchar(10)
Last_Name: Varchar(10)
ORDERS
Orders_ID: Integer
Cust_ID: Integer
Prod_ID: Integer
Order_Date: Date
Sales: Integer
PRODUCTS
Prod_ID: Integer
Name: Varchar(10)
Category: Varchar(9)

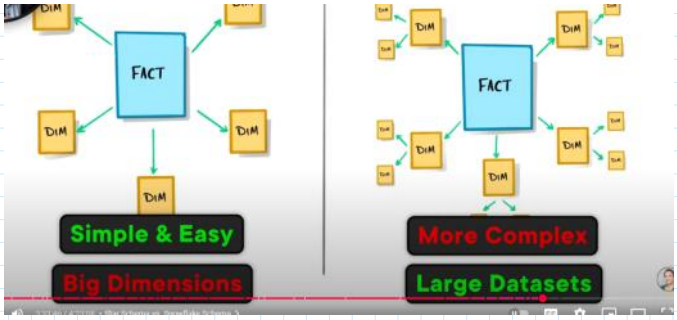

Conceptual Data Model
CUSTOMERS
Creates
ORDERS
PRODUCTS
Has
Logical Data Model
CUSTOMERS
Cust ID
First Name
Last Name
ORDERS
Order ID
Cust ID
Prod ID
Order Date
Sales
PRODUCTS
Prod ID
Name
Category
Physical Data Model
CUSTOMERS
Cust_ID: Integer
First_Name: Varchar(10)
Last_Name: Varchar(10)
ORDERS
Orders_ID: Integer
Cust_ID: Integer
Prod_ID: Integer
Order_Date: Date
Sales: Integer
PRODUCTS
Prod_ID: Integer
Name: Varchar(10)
Category: Varchar(9)
BIG PICTURE
BLUE PRINT
IMPLEMENTATION


OPTIMIZED FOR REPORTING
FLEXIBLE
EASE TO UNDERS
flexible scalable
understand and

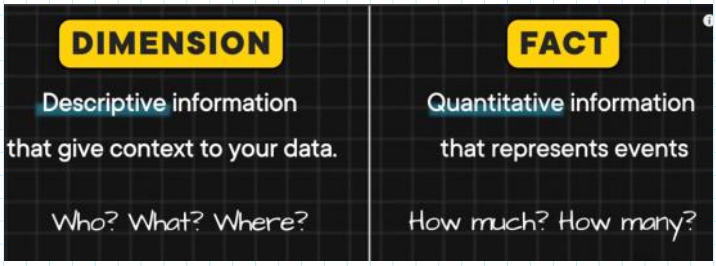

STAR SCHEMA
DIM
DIM
FACT
DIM
DIM
DIM


SNOWFLAKE SCHEMA
DIM
DIM
FACT
DIM
DIM
DIM

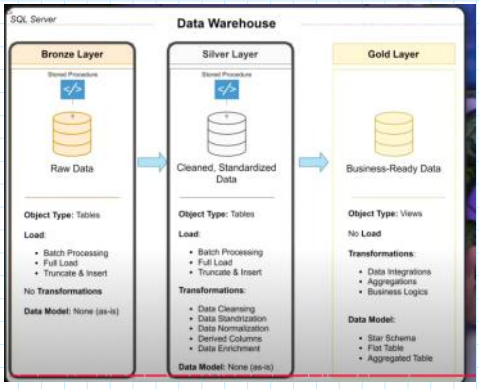

STAR SCHEMA
DIM
DIM
FACT
SNOWFLAKE SCHEMA
DIM
DIM
FACT


FACT
DIM
DIM
DIM
Simple & Easy
Big Dimensions
FACT
DIM
DIM
DIM
DIM
DIM
More Complex
Large Datasets

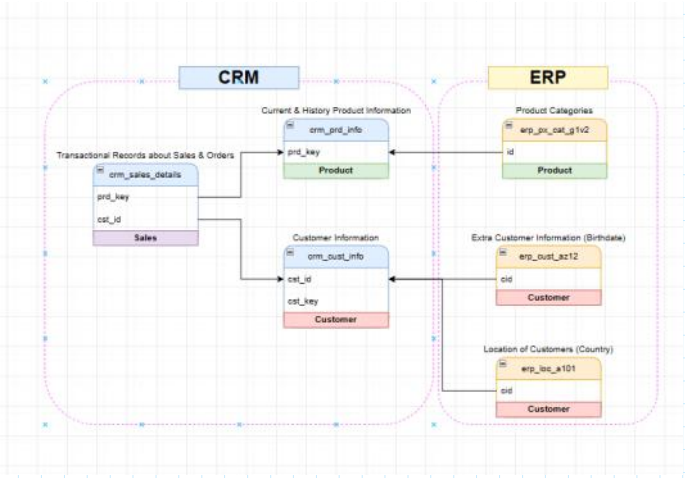

Theory
Dimensions vs Facts

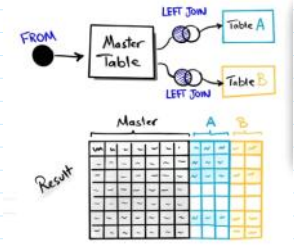

DIMENSION
Descriptive information
that give context to your data.
Who? What? Where?
FACT
Quantitative information
that represents events
How much? How many?


Build Gold Layer
Explore the Business Objects


SQL Server
Data Warehouse
Bronze Layer
Raw Data
Object Type: Tables
Load:
Batch Processing
Full Load
Truncate & Insert
No Transformations
Data Model: None (as-is)
Silver Layer
Cleaned, Standardized Data
Object Type: Tables
Load:
Batch Processing
Full Load
Truncate & Insert
Transformations:
Data Cleansing
Data Standrization
Data Normalization
Derived Columns
Data Enrichment
Data Model: None (as-is)
Gold Layer
Business-Ready Data
Object Type: Views
No Load
Transformations:
Data Integrations
Aggregations
Business Logics
Data Model:
Star Schema
Flat Table
Aggregated Table

Add labels





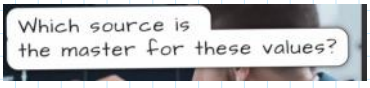

```
SELECT cst_id, COUNT(*) FROM (

    SELECT                                  Check duplicates
        ci.cst_id,
        ci.cst_key,
        ci.cst_firstname,
        ci.cst_lastname,
        ci.cst_marital_status,
        ci.cst_gndr,
        ci.cst_create_date,
        ca.bdate,
```

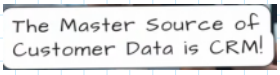
```
        ci.cst_firstname,
        ci.cst_lastname,
        ci.cst_marital_status,
        ci.cst_gndr,
        ci.cst_create_date,
        ca.bdate,
        ca.gen,
        la.cntry
    FROM
        silver.crm_cust_info ci
    LEFT JOIN
        silver.erp_cust_az12 ca
    ON ci.cst_key = ca.cid
    LEFT JOIN
        silver.erp_loc_a101 la
    ON  ci.cst_key = la.cid
) t GROUP BY cst_id
HAVING COUNT(*) > 1
```

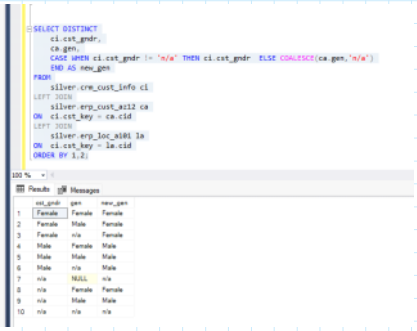
```
SELECT DISTINCT
    ci.cst_gndr,
    ca.gen
FROM silver.crm_cust_info ci
LEFT JOIN silver.erp_cust_az12 ca
ON        ci.cst_key = ca.cid
LEFT JOIN silver.erp_loc_a101 la
ON        ci.cst_key = la.cid
ORDER BY 1,2
```

	cst_gndr	gen
1	Female	Female
2	Female	Male
3	Female	n/a
4	Male	Female
5	Male	Male
6	Male	n/a
7	n/a	NULL
8	n/a	Female
9	n/a	Male

NULLs often come from joined tables!
NULL will appear if SQL finds no match





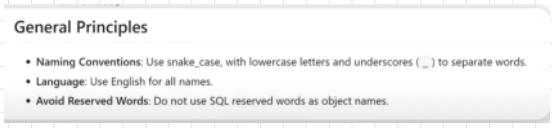


```
SELECT DISTINCT
    ci.cst_gndr,
    ca.gen
FROM
    silver.crm_cust_info ci
LEFT JOIN
    silver.erp_cust_az12 ca
ON ci.cst_key = ca.cid
LEFT JOIN
    silver.erp_loc_a101 la
ON  ci.cst_key = la.cid
ORDER BY 1,2;
```

```
SELECT DISTINCT
    ci.cst_gndr,
    ca.gen,
    CASE WHEN ci.cst_gndr != 'n/a' THEN ci.cst_gndr
    ELSE COALESCE(ca.gen,'n/a')
    END AS new_gen
FROM
    silver.crm_cust_info ci
LEFT JOIN
    silver.erp_cust_az12 ca
ON ci.cst_key = ca.cid
LEFT JOIN
    silver.erp_loc_a101 la
ON  ci.cst_key = la.cid
```

```
ORDER BY 1,2;
```

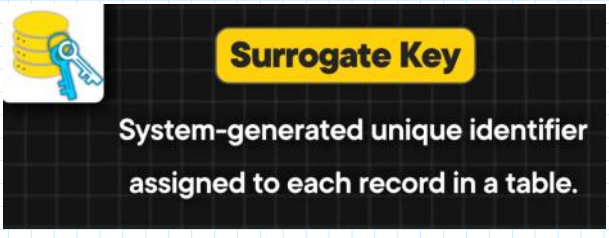
```
SELECT
    ci.cst_id,
    ci.cst_key,
    ci.cst_firstname,
    ci.cst_lastname,
    ci.cst_marital_status,
    CASE WHEN ci.cst_gndr != 'n/a' THEN ci.cst_gndr
    ELSE COALESCE(ca.gen,'n/a')
    END AS new_gen,
    ci.cst_create_date,
    ca.bdate,
    ca.gen,
    la.cntry
FROM
    silver.crm_cust_info ci
LEFT JOIN
    silver.erp_cust_az12 ca
ON ci.cst_key = ca.cid
LEFT JOIN
    silver.erp_loc_a101 la
ON  ci.cst_key = la.cid
```

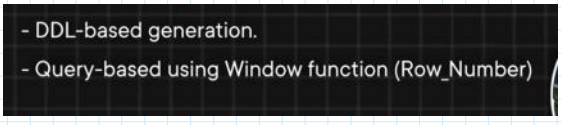



```
SELECT
    ci.cst_id AS customer_id,
    ci.cst_key AS customer_number,
    ci.cst_firstname AS first_name,
    ci.cst_lastname AS last_name,
    la.cntry AS country,
    ci.cst_marital_status AS marital_status,
    CASE WHEN ci.cst_gndr != 'n/a' THEN ci.cst_gndr
    ELSE COALESCE(ca.gen,'n/a')
    END AS gender,
    ca.bdate AS birthdate,
    ci.cst_create_date AS create_date
FROM
    silver.crm_cust_info ci
LEFT JOIN
    silver.erp_cust_az12 ca
ON ci.cst_key = ca.cid
LEFT JOIN
    silver.erp_loc_a101 la
ON  ci.cst_key = la.cid
```

customer_id	customer_number	first_name	last_name	country	marital_status	gender	birthdate	create_date
11000	AW00011000	Jon	Yang	Australia	Married	Male	1971-10-06	2025-10-06
11001	AW00011001	Eugene	Huang	Australia	Single	Male	1976-05-10	2025-10-06
11002	AW00011002	Ruben	Torres	Australia	Married	Male	1971-02-09	2025-10-06
11003	AW00011003	Christy	Zhu	Australia	Single	Female	1973-08-14	2025-10-06
11004	AW00011004	Elizabeth	Johnson	Australia	Single	Female	1979-08-05	2025-10-06
11005	AW00011005	Julio	Ruiz	Australia	Single	Male	1976-08-01	2025-10-06
11006	AW00011006	Janet	Alvarez	Australia	Single	Female	1976-12-02	2025-10-06
11007	AW00011007	Marco	Mehta	Australia	Married	Male	1969-11-06	2025-10-06

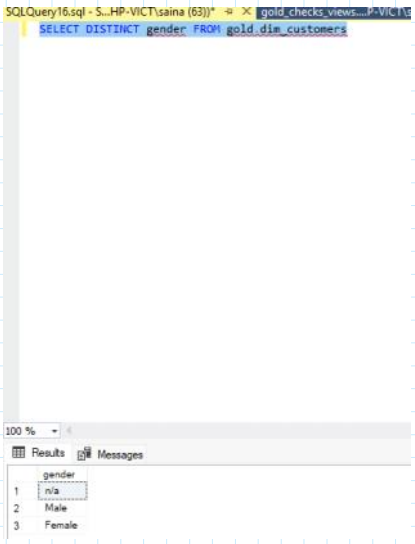
It is dimension table . Check if primary key is present if not create surrogate key.

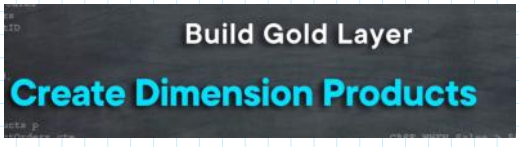




```
SELECT
    ROW_NUMBER() OVER (ORDER BY cst_id) as
    customer_key,
    ci.cst_id AS customer_id,
    ci.cst_key AS customer_number,
    ci.cst_firstname AS first_name,
    ci.cst_lastname AS last_name,
    la.cntry AS country,
    ci.cst_marital_status AS marital_status,
    CASE WHEN ci.cst_gndr != 'n/a' THEN ci.cst_gndr
    ELSE COALESCE(ca.gen,'n/a')
    END AS gender,
    ca.bdate AS birthdate,
    ci.cst_create_date AS create_date
FROM
    silver.crm_cust_info ci
LEFT JOIN
    silver.erp_cust_az12 ca
ON ci.cst_key = ca.cid
LEFT JOIN
    silver.erp_loc_a101 la
ON  ci.cst_key = la.cid
```

```
CREATE VIEW gold.dim_customers AS (
SELECT
    ROW_NUMBER() OVER (ORDER BY cst_id) as
    customer_key,
    ci.cst_id AS customer_id,
    ci.cst_key AS customer_number,
    ci.cst_firstname AS first_name,
    ci.cst_lastname AS last_name,
    la.cntry AS country,
    ci.cst_marital_status AS marital_status,
    CASE WHEN ci.cst_gndr != 'n/a' THEN ci.cst_gndr
    ELSE COALESCE(ca.gen,'n/a')
    END AS gender,
    ca.bdate AS birthdate,
    ci.cst_create_date AS create_date
FROM
    silver.crm_cust_info ci
LEFT JOIN
    silver.erp_cust_az12 ca
ON ci.cst_key = ca.cid
LEFT JOIN
    silver.erp_loc_a101 la
ON  ci.cst_key = la.cid)
```





We may not need historization data. Should remove

```
SELECT
    pn.prd_id,
    pn.cat_id
```

```
SELECT
    pn.prd_id,
    pn.cat_id,
    pn.prd_key,
    pn.prd_nm,
    pn.prd_cost,
    pn.prd_line,
    pn.prd_start_dt
FROM
    silver.crm_prd_info pn
WHERE
    prd_end_dt IS NULL -- filter out historical data
```

```
SELECT
    pn.prd_id,
    pn.cat_id,
    pn.prd_key,
    pn.prd_nm,
    pn.prd_cost,
    pn.prd_line,
    pn.prd_start_dt,
    pc.cat,
    pc.subcat,
    pc.maintenance
FROM
    silver.crm_prd_info pn
LEFT JOIN
    silver.erp_px_cat_g1v2 pc
ON pn.cat_id = pc.id
WHERE
    prd_end_dt IS NULL -- filter out historical data
```

Check for duplicates

```
SELECT prd_key, COUNT(*) FROM (
SELECT
    pn.prd_id,
    pn.cat_id,
    pn.prd_key,
    pn.prd_nm,
    pn.prd_cost,
    pn.prd_line,
    pn.prd_start_dt,
    pc.cat,
    pc.subcat,
    pc.maintenance
FROM
    silver.crm_prd_info pn
LEFT JOIN
    silver.erp_px_cat_g1v2 pc
ON pn.cat_id = pc.id
WHERE
    prd_end_dt IS NULL -- filter out historical data
) t GROUP BY prd_key
HAVING COUNT(*) > 1
```

Rename and reorder the features

```
SELECT
    pn.prd_id AS product_id,
    pn.prd_key AS product_number,
    pn.prd_nm AS product_name,
    pn.cat_id AS category_id,
    pc.cat AS category,
    pc.subcat AS subcategory,
    pc.maintenance,
    pn.prd_cost AS cost,
    pn.prd_line AS product_line,
    pn.prd_start_dt AS start_date
FROM
    silver.crm_prd_info pn
LEFT JOIN
    silver.erp_px_cat_g1v2 pc
ON pn.cat_id = pc.id
WHERE
    prd_end_dt IS NULL -- filter out historical data
```

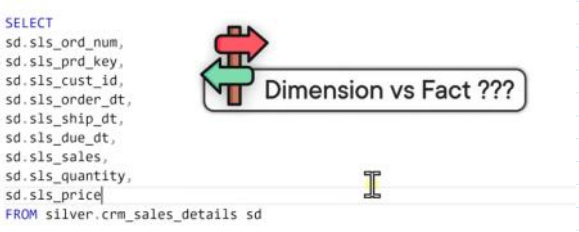
Create surrogate primary key

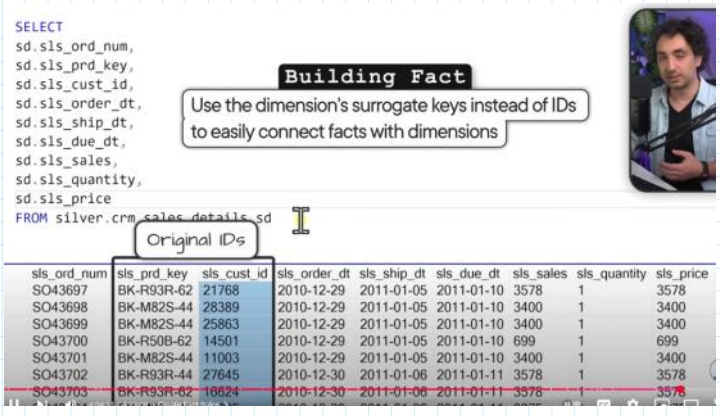
```
SELECT
    ROW_NUMBER() OVER(ORDER BY
    pn.prd_start_dt, pn.prd_key) AS product_key,
    pn.prd_id AS product_id,
    pn.prd_key AS product_number,
    pn.prd_nm AS product_name,
    pn.cat_id AS category_id,
    pc.cat AS category,
    pc.subcat AS subcategory,
    pc.maintenance,
    pn.prd_cost AS cost,
    pn.prd_line AS product_line,
    pn.prd_start_dt AS start_date
FROM
    silver.crm_prd_info pn
LEFT JOIN
    silver.erp_px_cat_g1v2 pc
```

```
ON pn.cat_id = pc.id
WHERE
    prd_end_dt IS NULL -- filter out historical data
```

```
CREATE VIEW gold.dim_products AS (
SELECT
    ROW_NUMBER() OVER(ORDER BY
    pn.prd_start_dt, pn.prd_key) AS product_key,
    pn.prd_id AS product_id,
    pn.prd_key AS product_number,
    pn.prd_nm AS product_name,
    pn.cat_id AS category_id,
    pc.cat AS category,
    pc.subcat AS subcategory,
    pc.maintenance,
    pn.prd_cost AS cost,
    pn.prd_line AS product_line,
    pn.prd_start_dt AS start_date
FROM
    silver.crm_prd_info pn
LEFT JOIN
    silver.erp_px_cat_g1v2 pc
ON pn.cat_id = pc.id
WHERE
    prd_end_dt IS NULL)
```

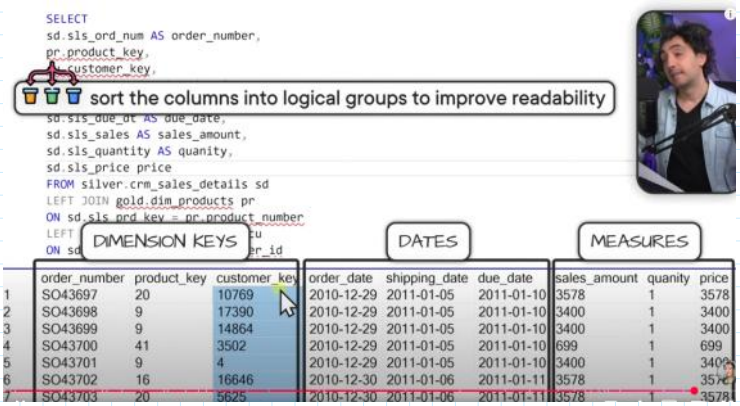
```
SELECT * FROM gold.dim_products
```



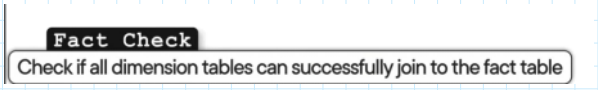


Rename and Reorder

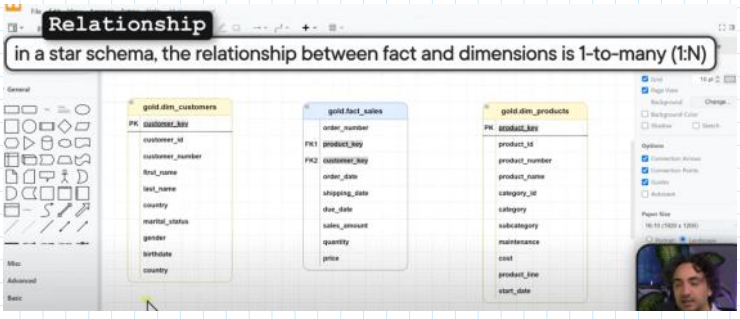
```
SELECT
    sd.sls_ord_num AS order_number,
    pr.product_key,
    cu.customer_key,
    sd.sls_order_dt AS order_date,
    sd.sls_ship_dt AS shipping_date,
    sd.sls_due_dt AS due_date,
    sd.sls_sales AS sales_amount,
    sd.sls_quantity AS quantity,
    sd.sls_price AS price
FROM
    silver.crm_sales_details sd
LEFT JOIN gold.dim_products pr
ON sd.sls_prd_key = pr.product_number
LEFT JOIN gold.dim_customers cu
ON sd.sls_cust_id = cu.customer_id
```

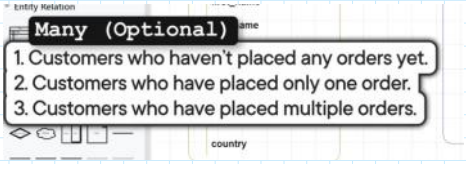


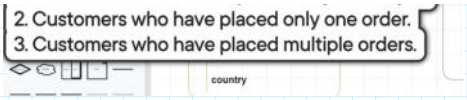

```
CREATE VIEW gold.fact_sales AS (
SELECT
    sd.sls_ord_num AS order_number,
    pr.product_key,
    cu.customer_key,
    sd.sls_order_dt AS order_date,
    sd.sls_ship_dt AS shipping_date,
    sd.sls_due_dt AS due_date,
    sd.sls_sales AS sales_amount,
    sd.sls_quantity AS quantity,
    sd.sls_price AS price
FROM
    silver.crm_sales_details sd
LEFT JOIN gold.dim_products pr
ON sd.sls_prd_key = pr.product_number
LEFT JOIN gold.dim_customers cu
ON sd.sls_cust_id = cu.customer_id)
```



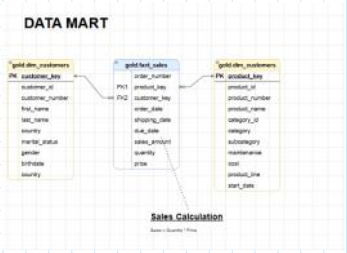

```
-- Foreign key integrity check
SELECT *
FROM gold.fact_sales f
LEFT JOIN gold.dim_customers c
ON c.customer_key = f.customer_key
LEFT JOIN gold.dim_products p
ON p.product_key = f.product_key
WHERE p.product_key IS NULL
```



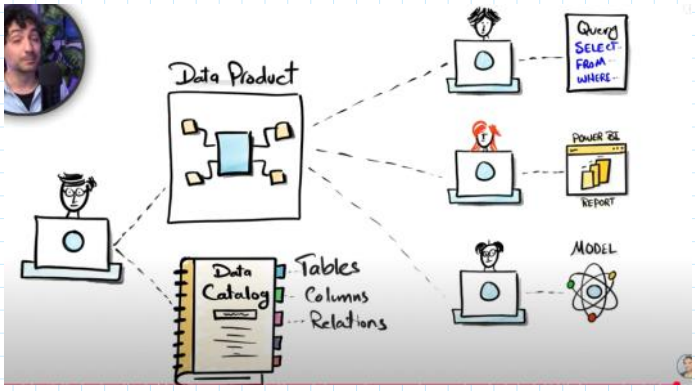


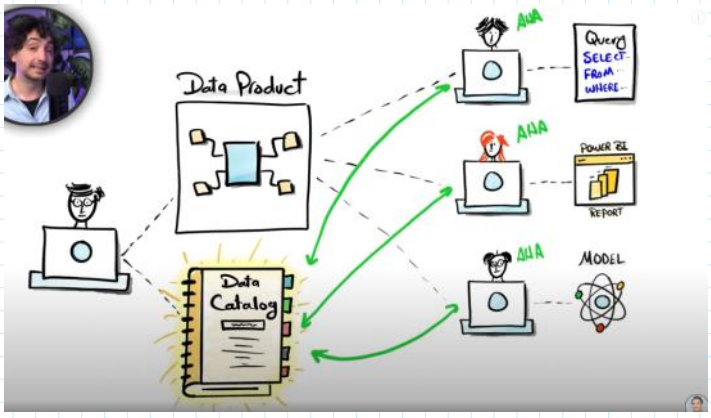


Data model



Data Catologue





Create all docs like draw.io images, naming convention.md, readme.md, data catalog.md etc.,

Upload in git and project of resume