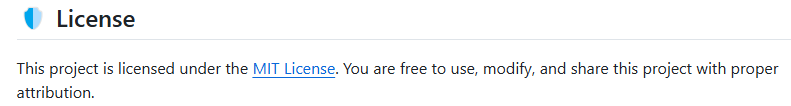
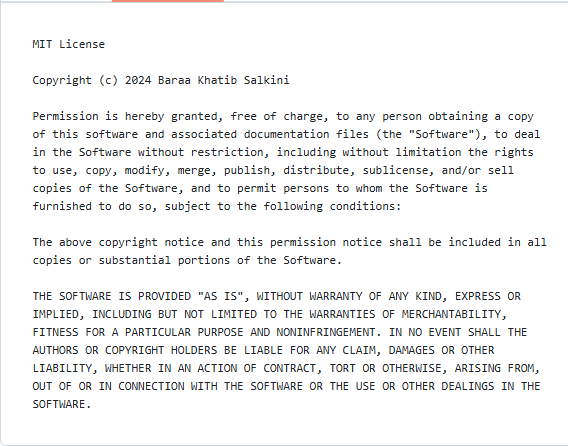
My Walkthrough steps following MIT License: Data With Baraa, Data Warehouse project: Made by Luke Yeo

**Note:**

* Dataset: <https://datawithbaraa.substack.com/p/build-a-data-warehouse-from-scratch>
* Notion Project Plan: <https://thankful-pangolin-2ca.notion.site/SQL-Data-Warehouse-Project-16ed041640ef80489667cfe2f380b269> (I am not the owner of this)
* Note: This is to learn and have an experience for building a data warehouse





* This project uses **Medallion Architecture**:
* **Bronze Layer:** Raw data is ingested as-is from CSV files into SQL Server.
* **Silver Layer:** Data is cleaned, standardized, and normalized for analysis.
* **Gold Layer:** Business-ready data is modeled in a star schema for reporting and analytics.
* What I learn from this project:
* **Data Architecture:** Design a modern Data Warehouse with the Bronze, Silver, and Gold layers.
* **ETL Pipelines:** Extract, transform, and load data into a structured warehouse.
* **Data Modeling:** Build fact and dimension tables for analytics.
* **Reporting:** Create SQL-based dashboards and insights that businesses can act on.
* ETL (Extract – Transform – Load): extract the data from the internet), transform it (data cleaning, formatting, ect) and load the data into the warehouse.
* Data Architecture: have different layers e.g. Source -> L1 -> L2 -> L3 -> L4
* Layer 1: ET
* Layer 2: ETL
* Layer 3: TL
* Layer 4: L (Where this is the target)

 = What we will be using in this project

* **Extraction**:

1. Methods: Pull, Push
2. Extract: Full, Incremental
3. Technique: Manual Data, Database Querying, File Parsing, API calls, Event Base Streaming, CDC (Change data capture), Web Scraping

* **Transformation**: 

1. Data Enrichment
2. Data integration
3. Derived Columns
4. Data Normalization and Standardization
5. Business Rules and logic
6. Data Aggregation
7. Data Cleansing

* **Load**:

1. Processing: Batch, Stream
2. Load: Full  (insert, upsert, create, drop, truncate), increment (upsert, append, merge)
3. Slowly Changing Dimensions (SCD): SCD0-No Historization, SCD1- Overwrite, SCD2- Historization, SCD

**Project requirements**

1. Building the Data Warehouse (Data Engineering)

* Develop a modern data warehouse using SQL Server to consolidate sales data , enabling analytical report and informed decision-making.
* Specifications:
* Data Source: Import data from 2 sources (ERP and CRM) provided as CSV file
* Data Quality: Cleanse and resolve data quality issues prior to analysis
* Integration: Combine both sources into a single friendly user model designed
* Scope: Focus on the latest dataset only, historization is not required
* Documentation: Provide clear documentation of the data model to support both business shareholder and analytics team

1. Power BI Analytics and reporting (Data Analysis)

* Developed SQL-based analytic to deliver insights into:
* Customer Behaviour
* Product Performance
* Sales Trends

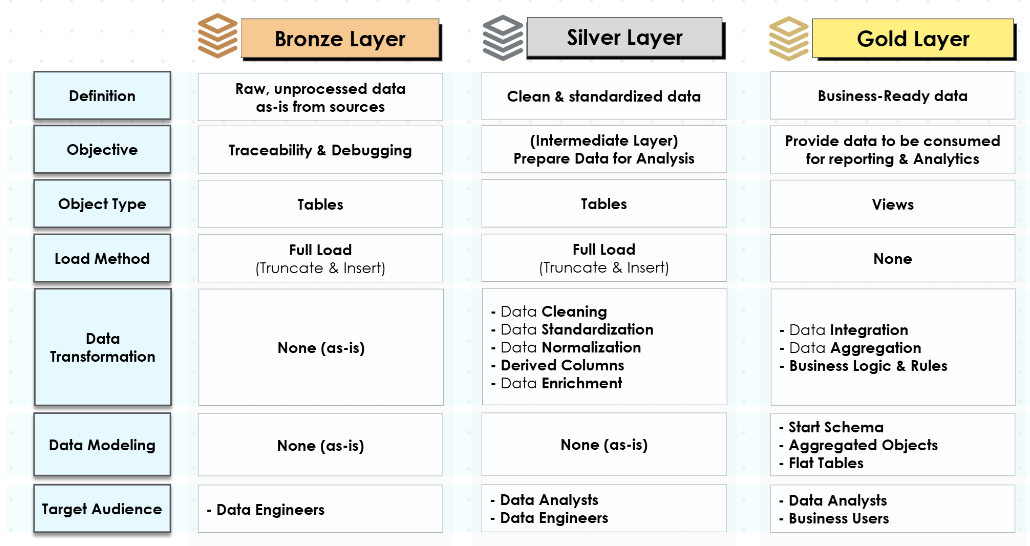
**Data Architecture**

1. Data Warehouse 
2. Data Lake
3. Data Lakehouse
4. Data Mesh

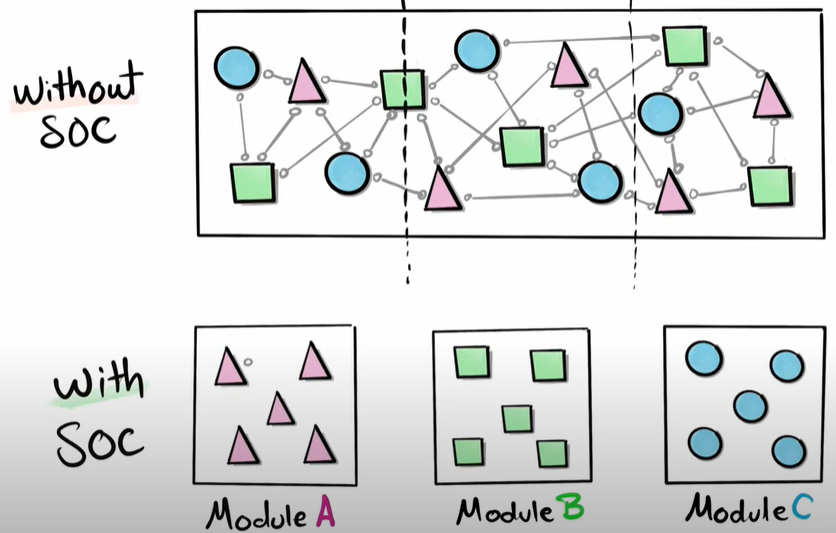
Approaches:

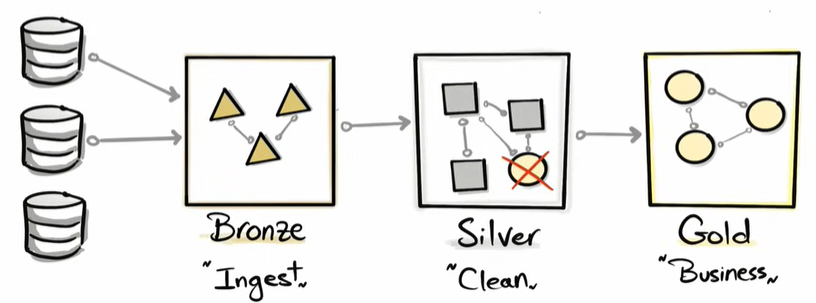
1. Inmon: Sources -> Stage -> EDW(3NF) -> Data Marts -> PowerBI
2. Kimball: Source ->Stage -> Data Marts -> PowerBI
3. Data Vault: Source -> Stage -> Raw vault -> Business Vault -> Data Mart -> PowerBI
4. Medallion Arch: Source -> Bronze -> Silver -> Gold -> PowerBI

**Layers:**

****

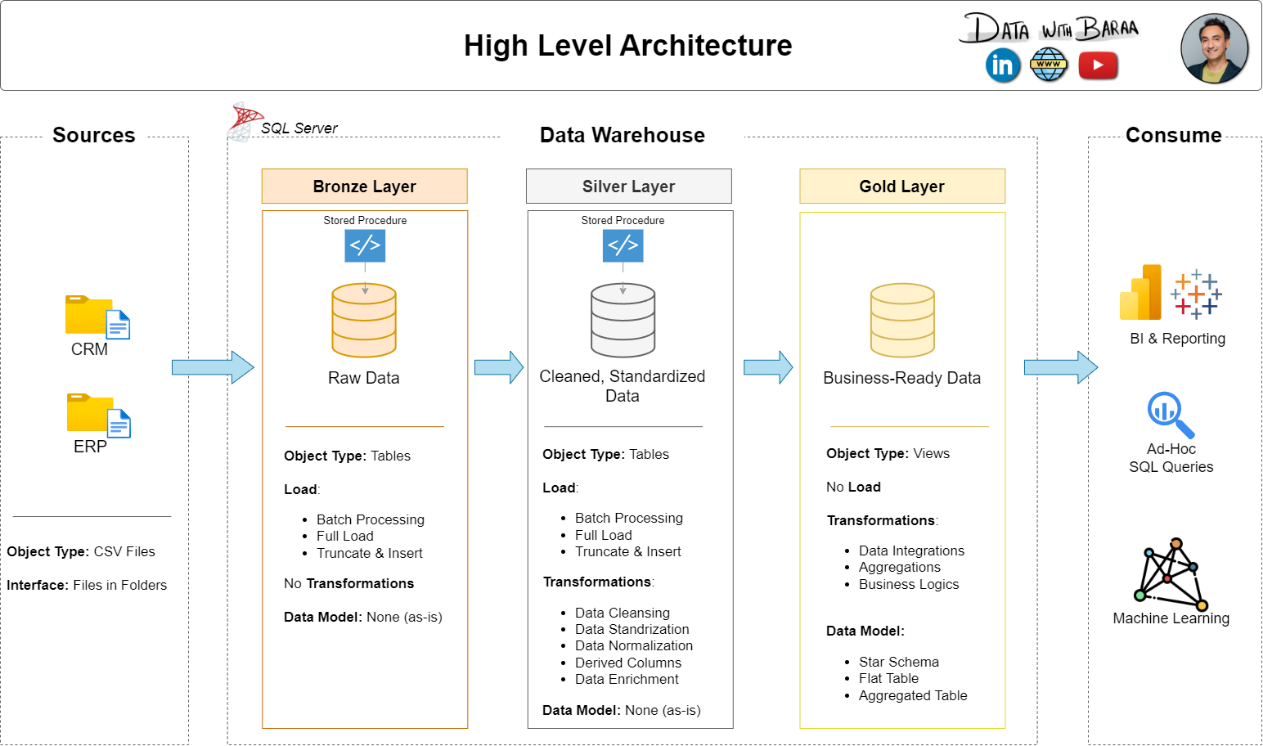
Separation of Concerns (SOC):



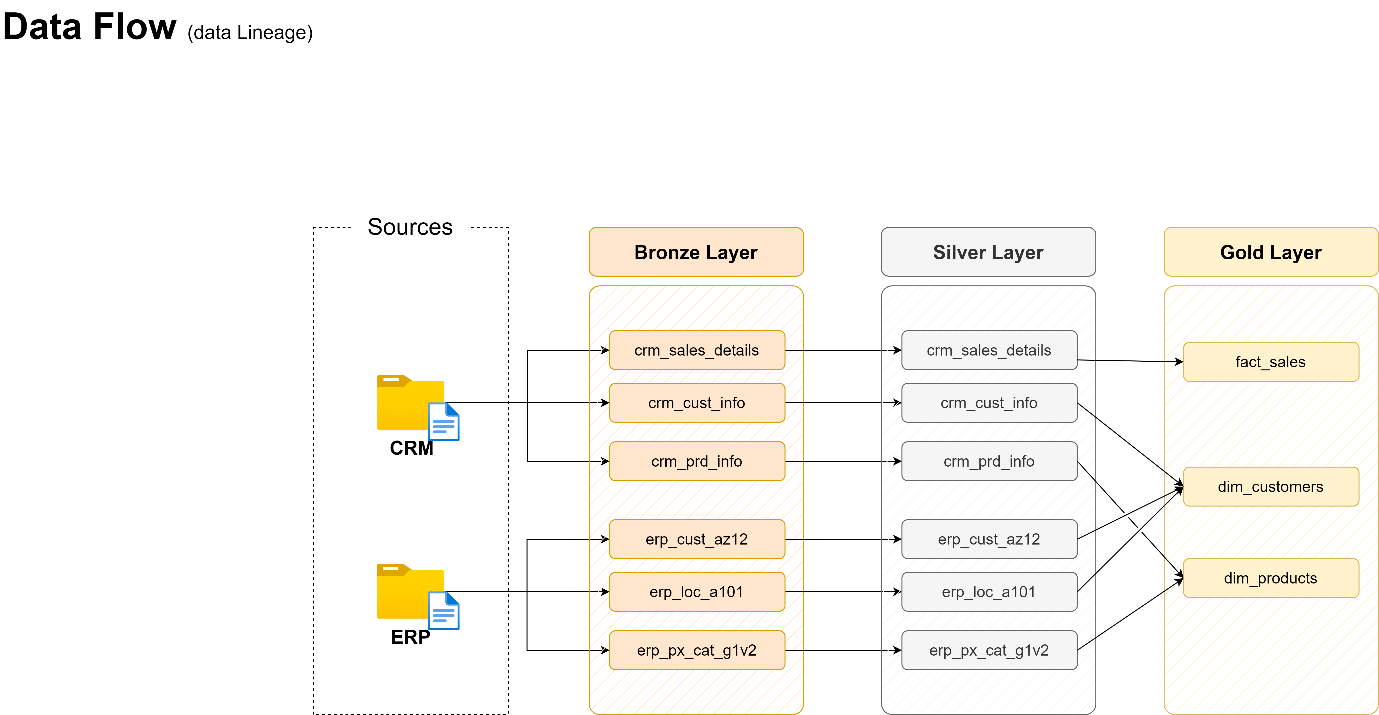


Must not be duplicated or intertwine/overlapping ^

Drawing the data Architecture in Draw.io:



Data Flow:



**Naming Conventions:**

* Set of Rules or Guidelines for naming anything in the project: Database, Schema, Tables, Store Procedures
* Use the snake case “ \_“: basically the underscore
* Language: use English (avoid SQL Reserve word like Table)

Table names:

1. Bronze Rule

* All names must start with the source system name and table names must match their original name without renaming
* **<sourcesystem>\_<entity>**: e.g. crm\_customer\_info where <crm>\_<customer\_info>

1. Silver Rule:

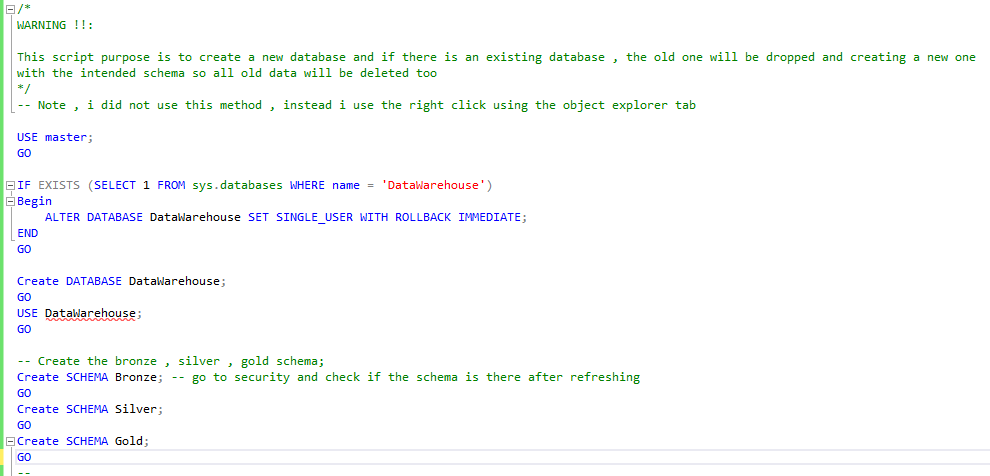
* same as Bronze rule

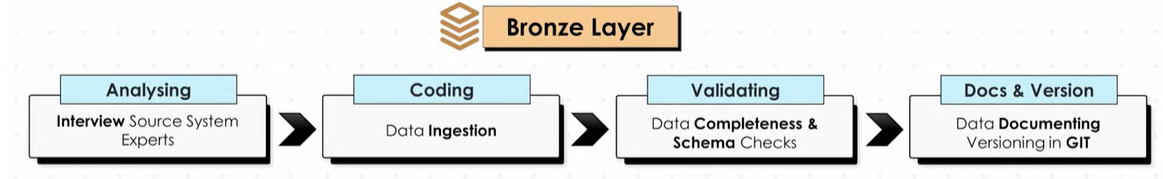
1. Gold Rule:

* All names must use meaningful, business-aligned name for tables, starting with the category prefix
* **<category>\_<entity**>: e.g. dim\_customer or fact\_sales

1. Surrogate keys: **<table\_name>\_key**
2. Technical columns: **dwh\_<column\_name>**
3. Store procedure: **load\_<layer>**

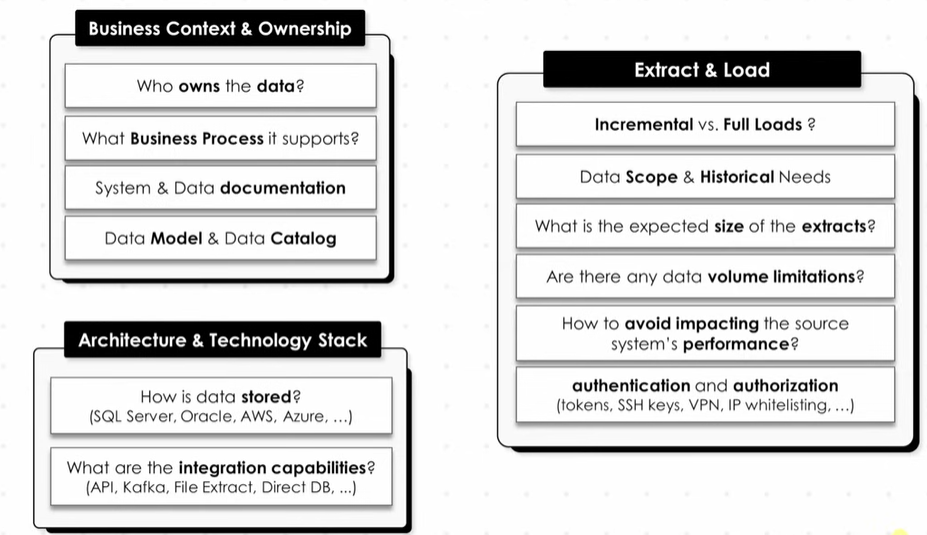
**Creating the Database and its Schema in SSMS:**



**Bronze layer**

* Do this by talking to the Source System Experts (Maybe Data engineer)

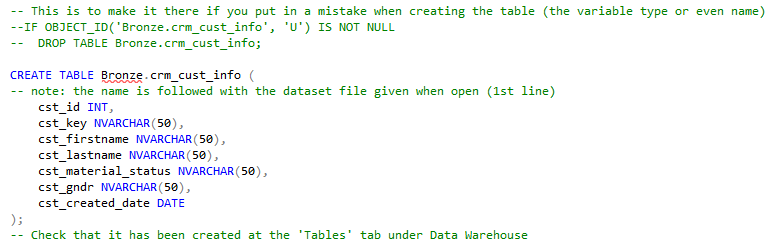
Questions to ask:



1. Creating the table in the Bronze Layer:

In the dataset of cst\_indo (customer\_info)



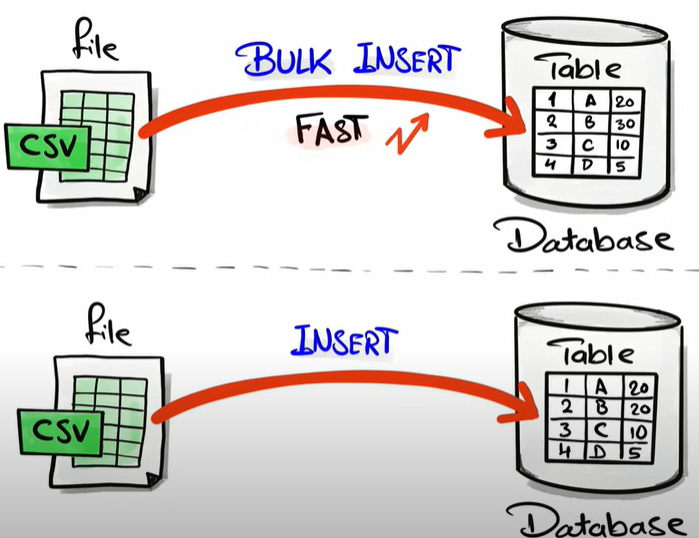


1. Do the same for all other dataset, **ALL CRM dataset** and **ALL ERP dataset.**

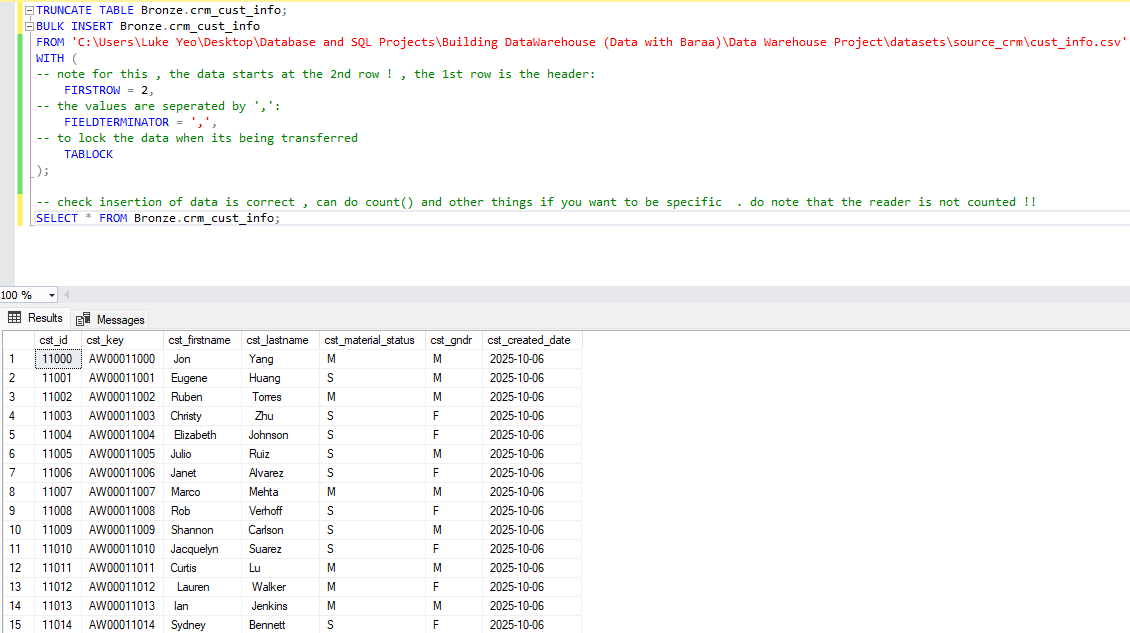
**Note**: I mis-spelled marital as material!!, FOR THE DROP TABLE FOR CHECKING, I DID NOT DO IT FOR EVERY TABLE THOUGH I SHOULD !!!

**Inserting database**

* we will be using bulk insert (a whole chunk of data) than normal insert (row by row)

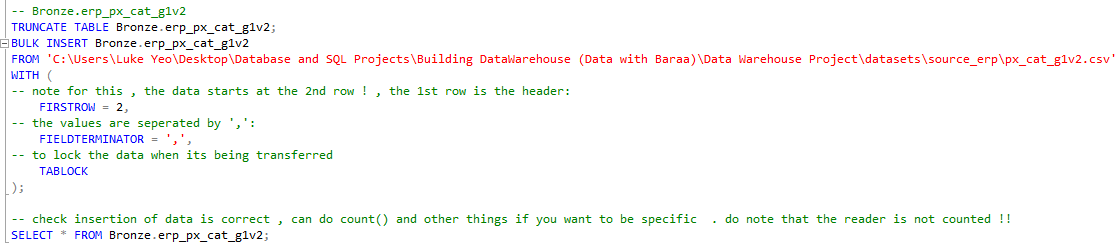


1. Bulk inserting the crm\_cust\_info

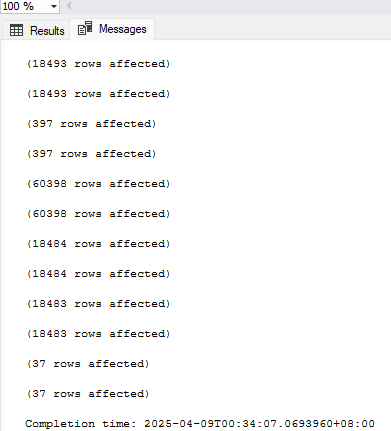


1. Do the same for all the other csv file.

* An example for erp csv file:

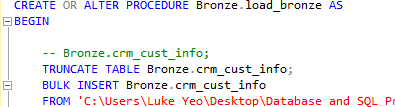


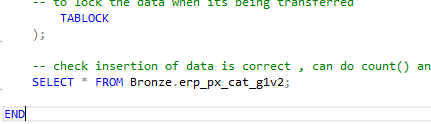
1. Final result, compare it to the number in csv files:



**Create Stored Procedure**

* Use when it is something you have to do frequently, maybe even on a daily basis.
* The SQL for it:





* Check on the tab, Programmability -> store procedures:



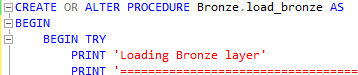
* Take note, when run, the “message” of the code as seen above is by “rows affected” is not clear. which is not a good thing for ETL so we can use the PRINT function:

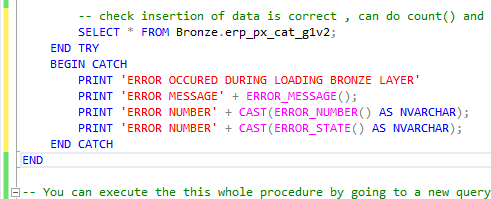
(I DID NOT DO THIS FOR ALL , JUST 1 AS AN EXAMPLE)



**Error catching:**

Using “Try and Catch” at the start and end:

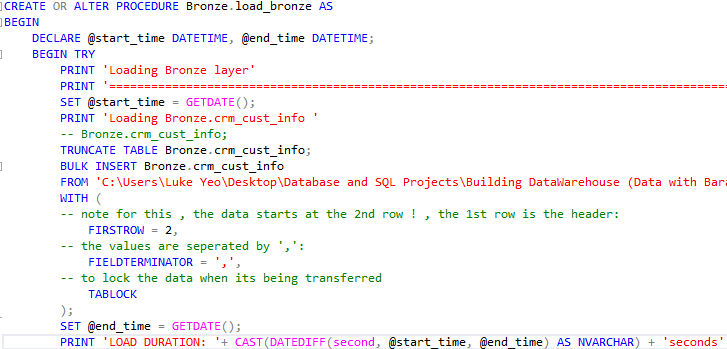




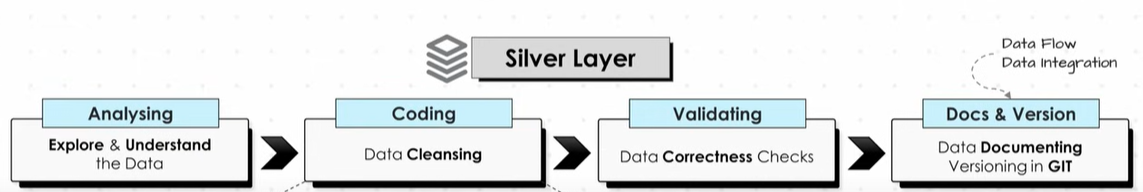
**ETL Duration:**

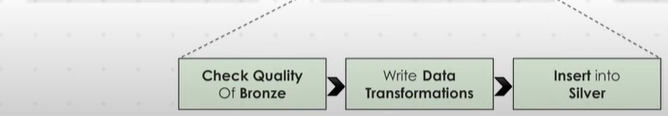
Helps to identify bottlenecks, optimize performance, monitor trends, detect issues

* For timing, use @start\_time and @end\_time
* Print out the difference



**Silver Layer:**

****

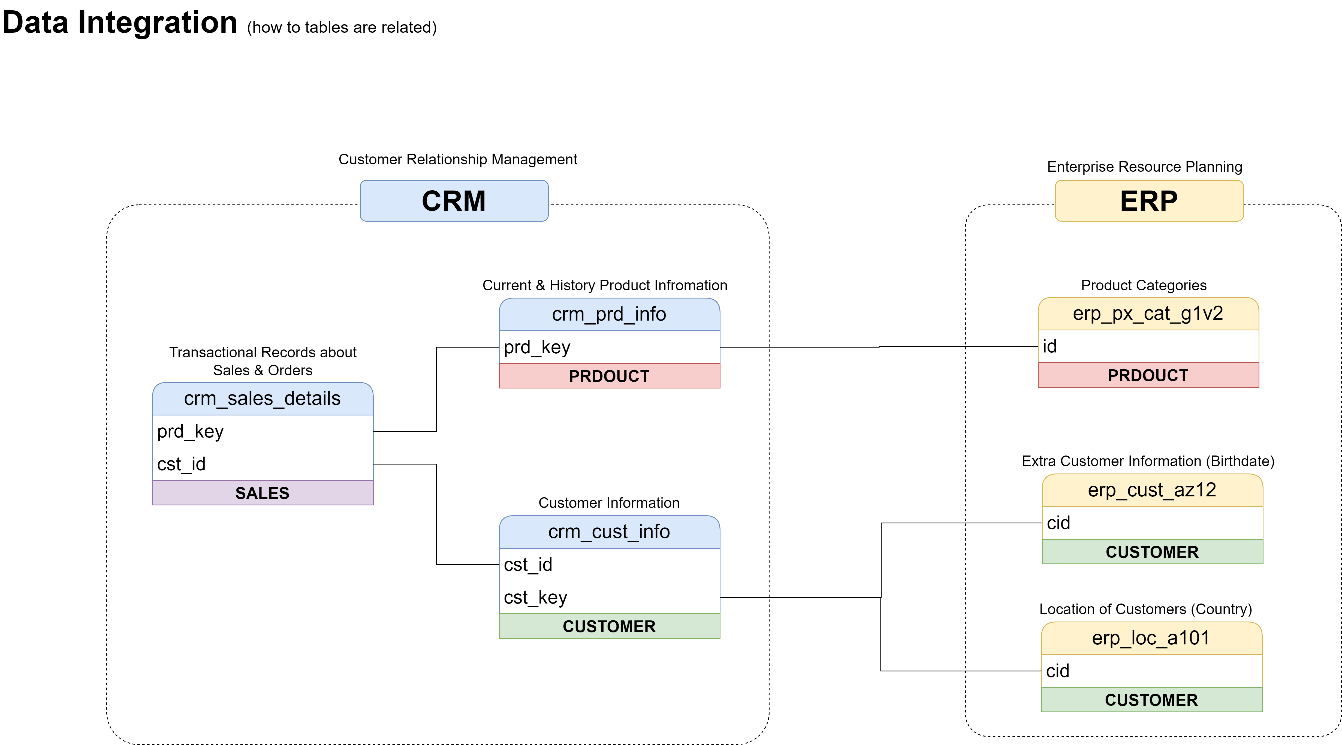
****

**Exploring and understanding the data:**

1. Visually looking at the different datasets:

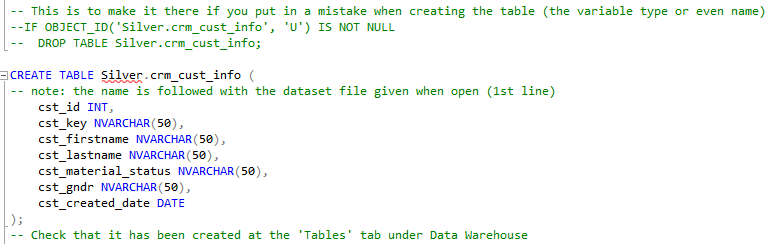
* Go to the tab, Table -> “Bronze.crm\_cust\_info”, right click and select top 1000 rows. Do the same for all the others in the Table
* See how the tables are set and such and draw a integration model, see what we can use as a primary – foreign key to other datasets

**Relations of Tables and Info:**

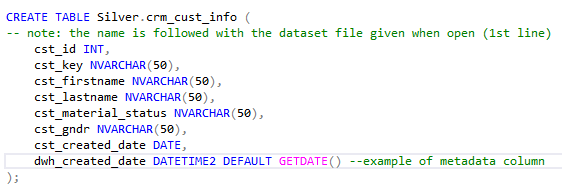


**Create DDL for Tables:**

1. Copy paste the same thing as in SQL for Bronze Layer
2. Find and Replace in the edit tab : “Bronze” to “Silver”



Note (Meta Data Column): Added by data engineers that do not originate from the source data but is to provide with more information that maybe useful like “update date” , “file location” , ect .

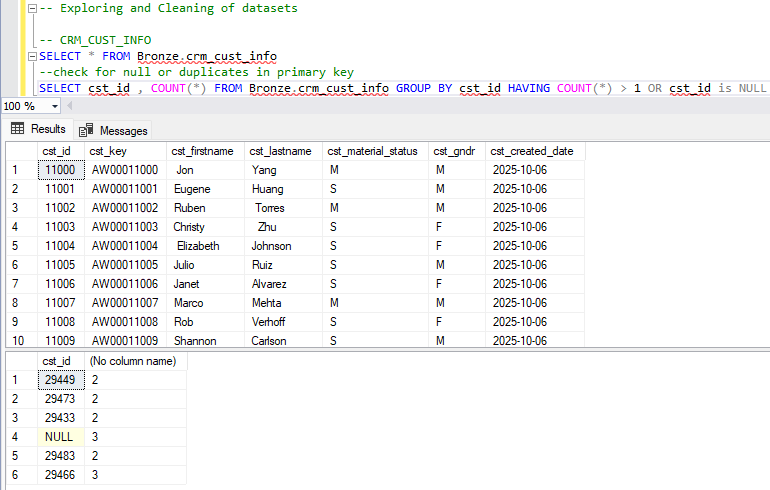


* Add it to all other table in silver layer then execute the query

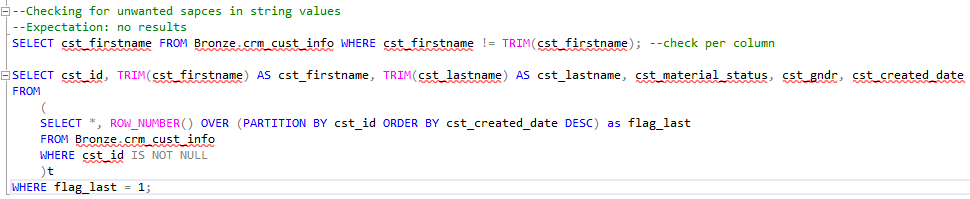
**Cleaning of the datasets:**

* **Crm\_cust\_info**

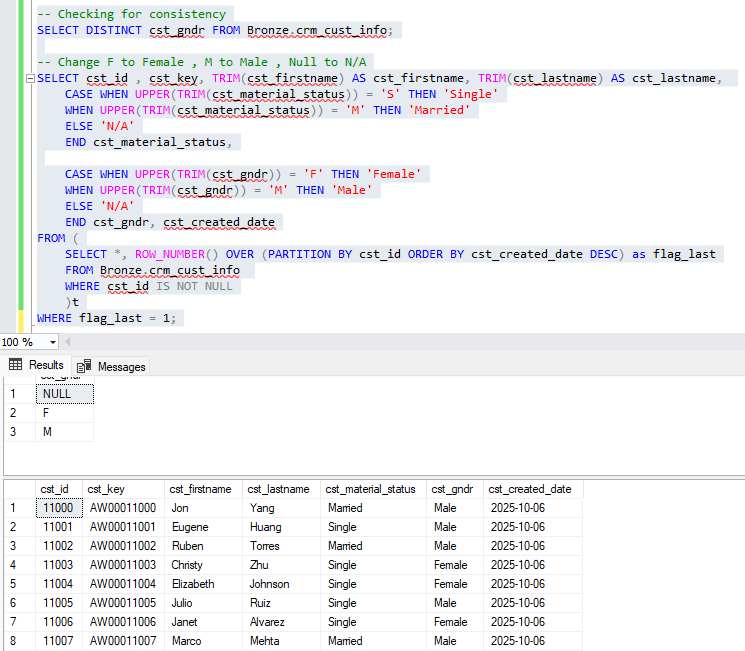
1. Checking for NULL and duplicate values in the primary key

****

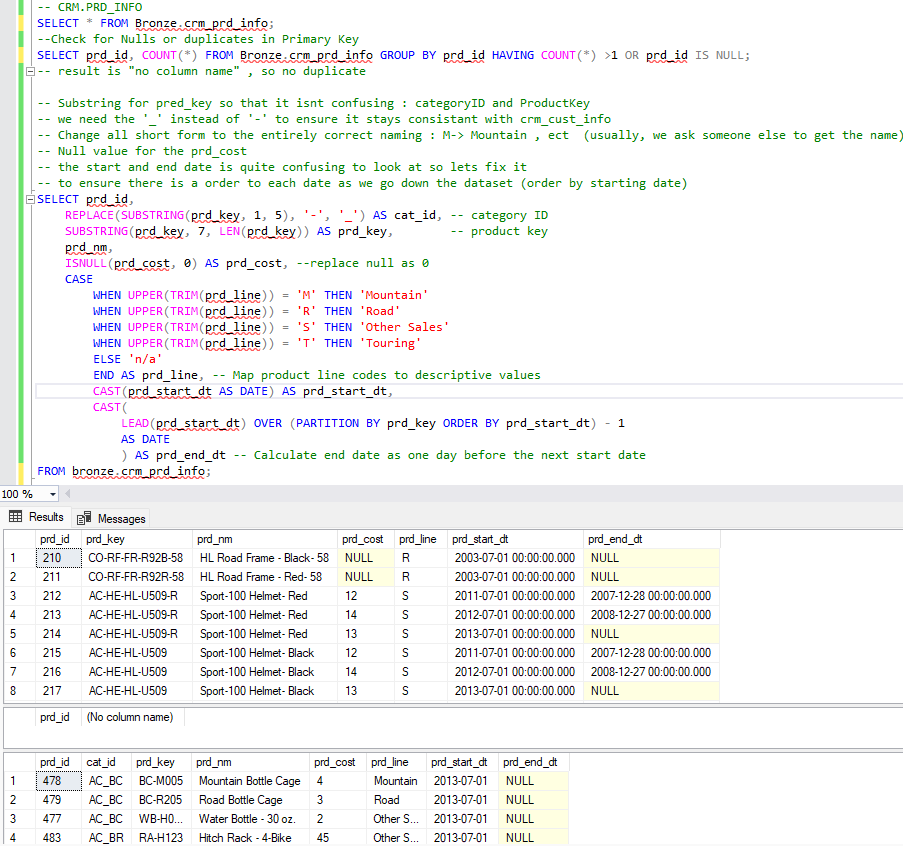
1. Checking for unwanted spaces in the dataset

****

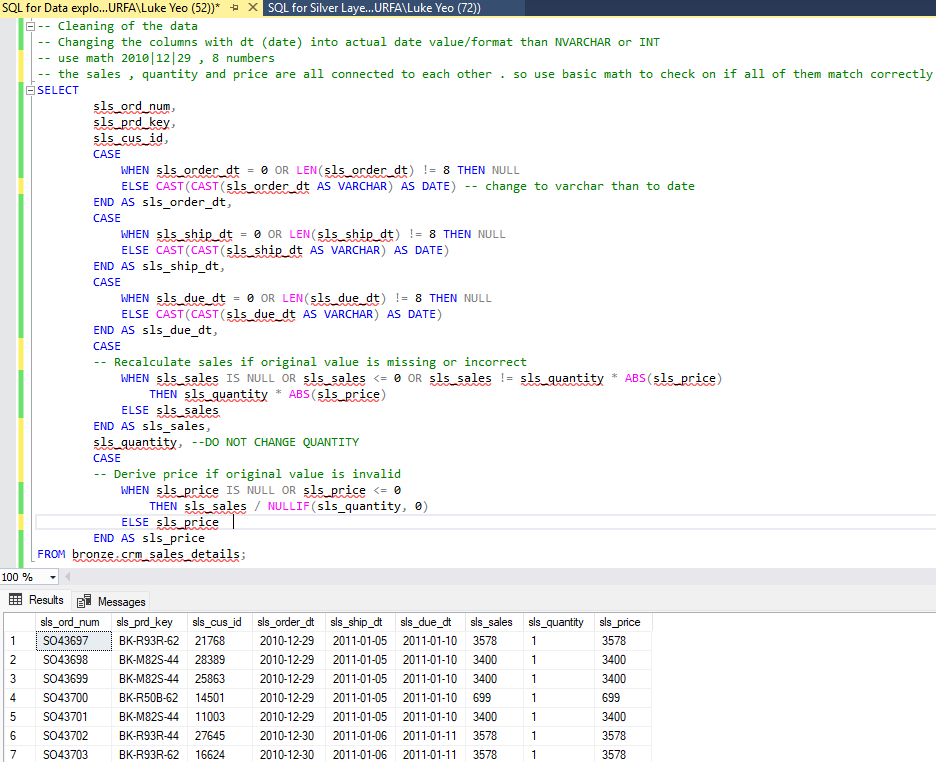
1. Checking the consistency of values: Data Normalization and Standardization



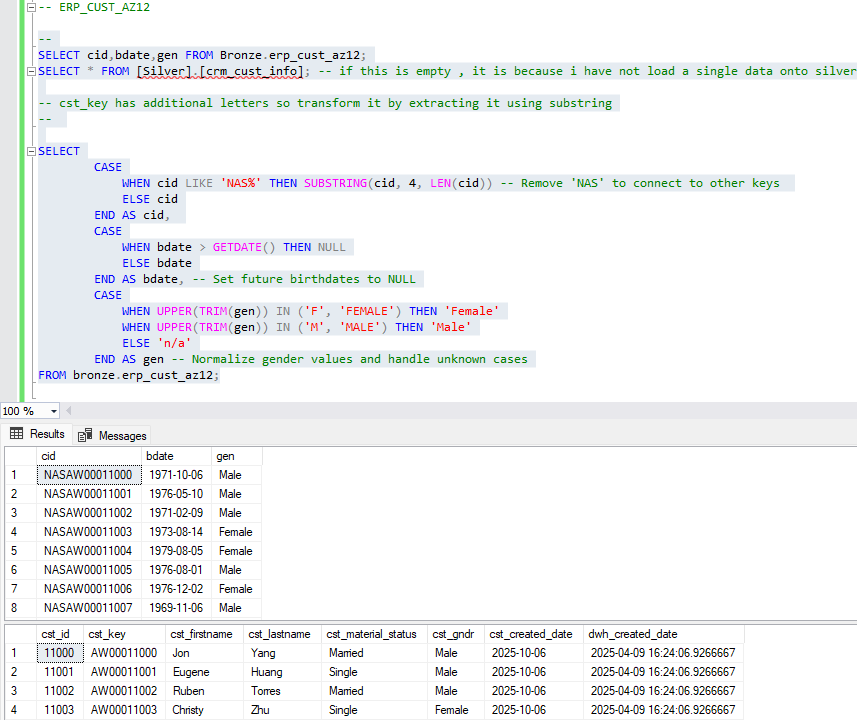
* Crm\_Prd\_info



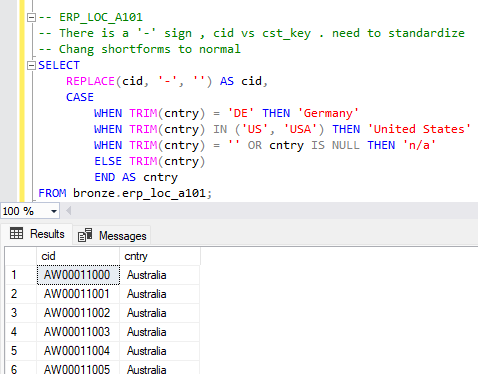
* Crm\_sales\_detail
* Business Rule: Sum of Sales = Quantity \* price.
* No negative or 0 or Null is allowed



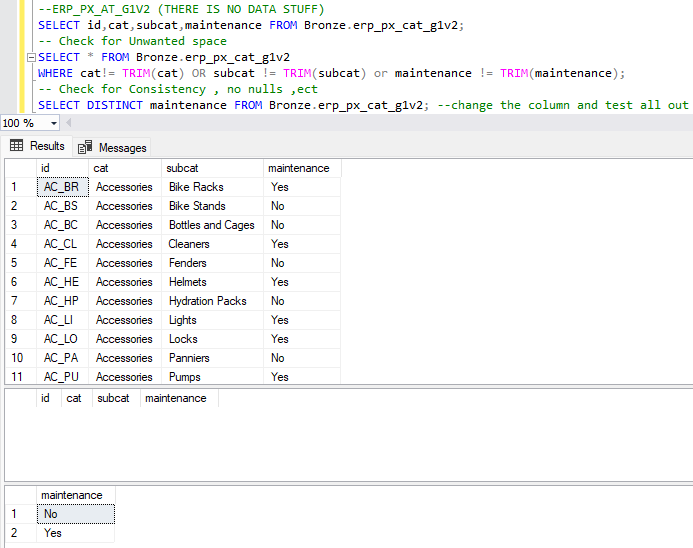
* Erp\_crust\_az12



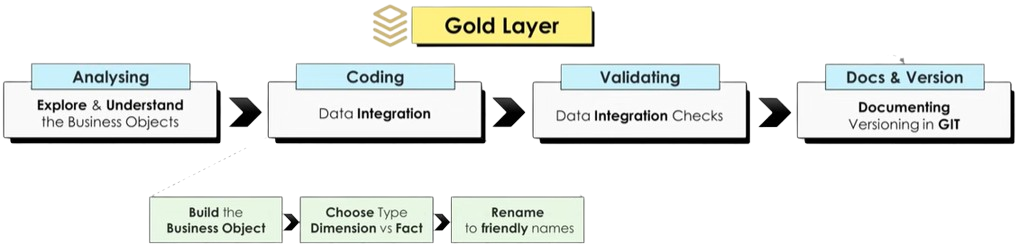
* Erp\_loc\_a101:

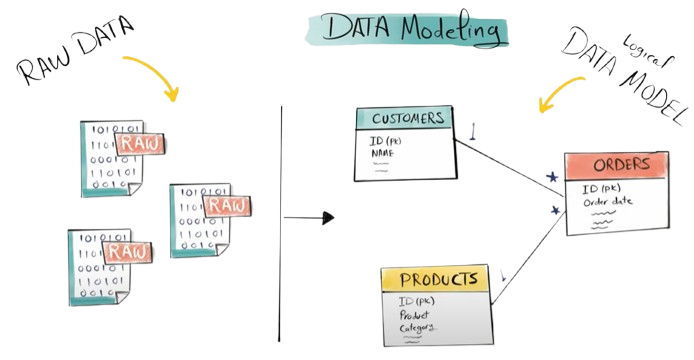


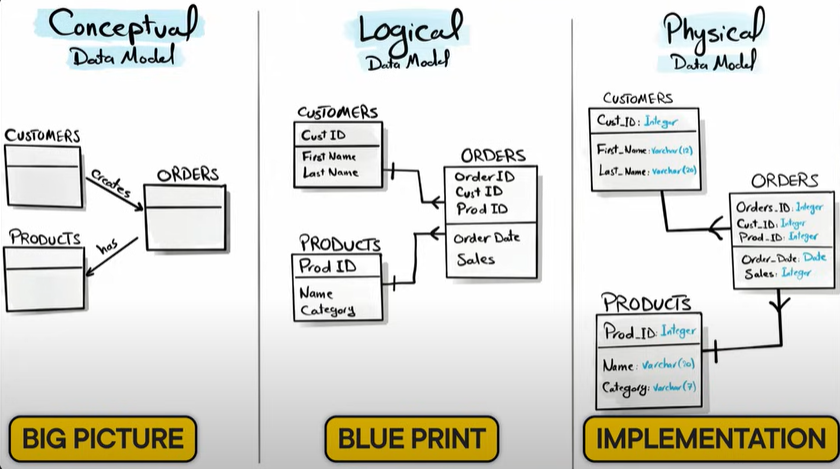
-Erp\_px\_cat\_g1v2



**GOLD LAYER:**

****

****

****

|  |  |
| --- | --- |
| **Star Schema:Checkmark** | **Snowflake Schema:** |

**Dimension:**

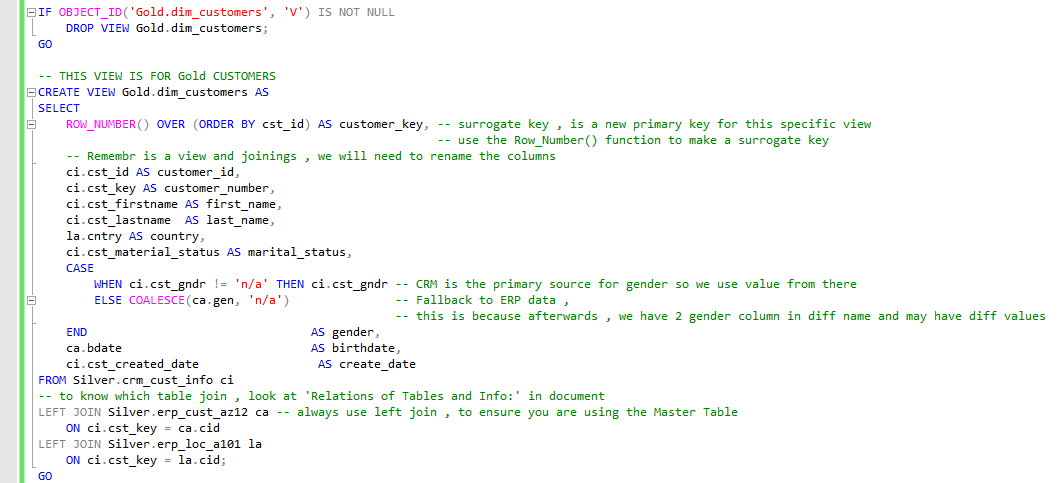
* Descriptive information that gives context to your data
* Who? What? Where?

**Fact:**

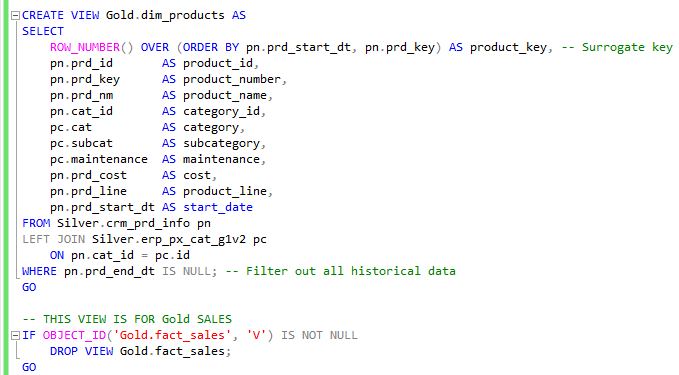
* Quantitative information that represents events
* How much? How many

IN GOLD, WE WILL BE USING VIEWS FROM SILVER DATA-TABLES, NOT TABLES!! (NO NEED TO LOAD ANYTHING)

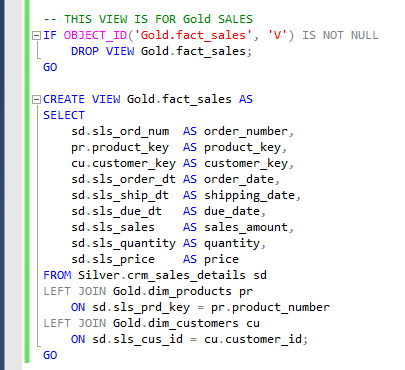
**Create Dimension Customers:**

****

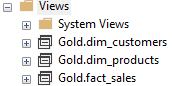
**Create Product info:**

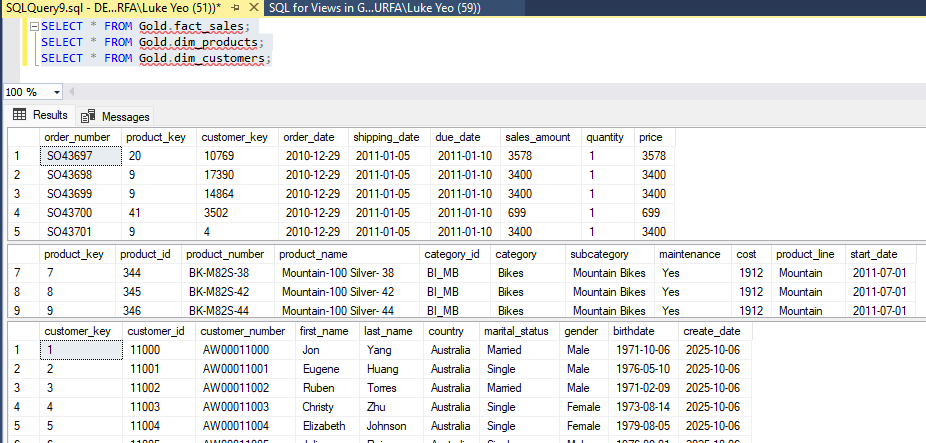
****

**Create Sales detail:**

****

**Note: for some reason, in case the views do not show up in the view tab. So, I used query to check:**

****

****