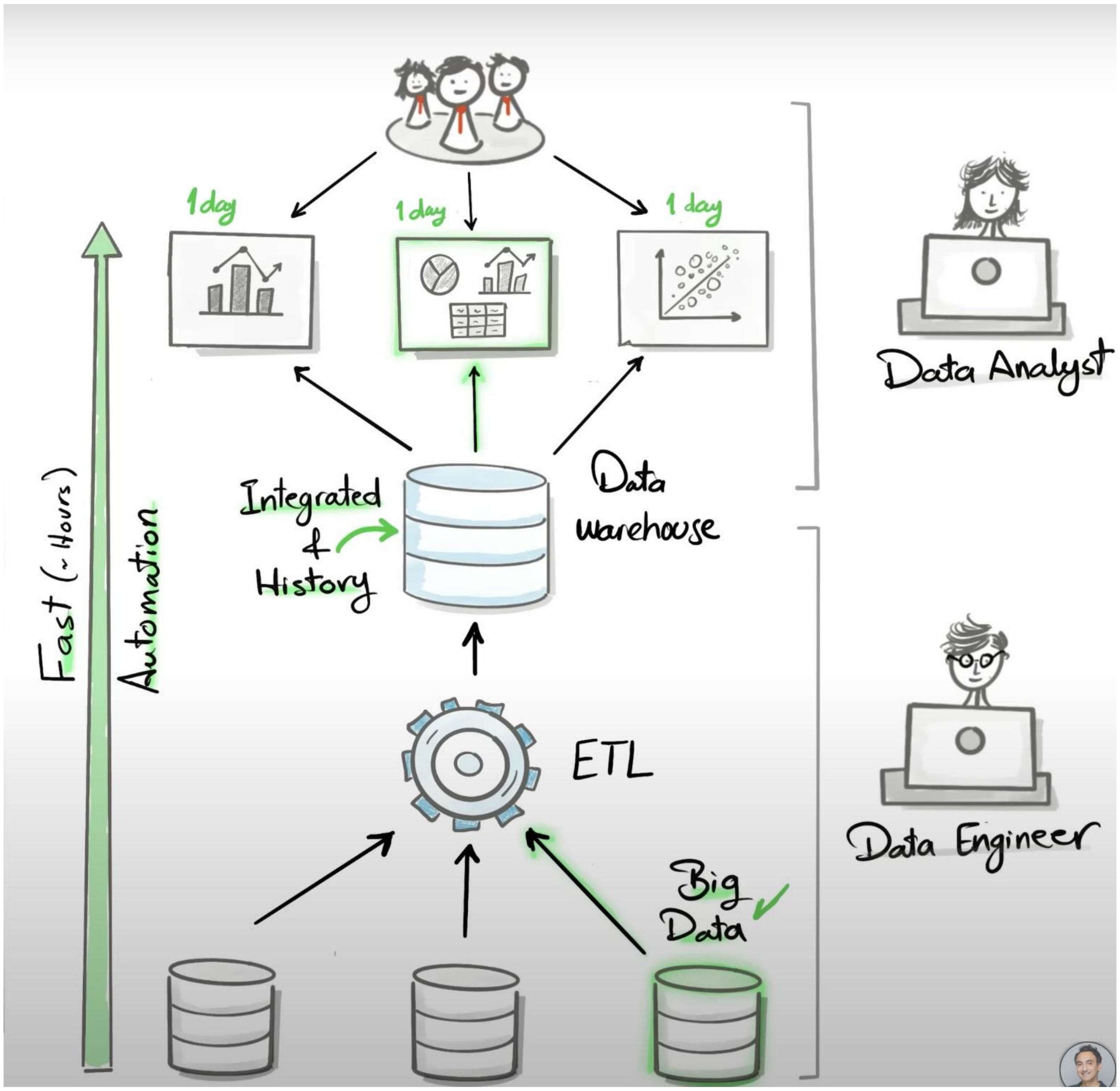
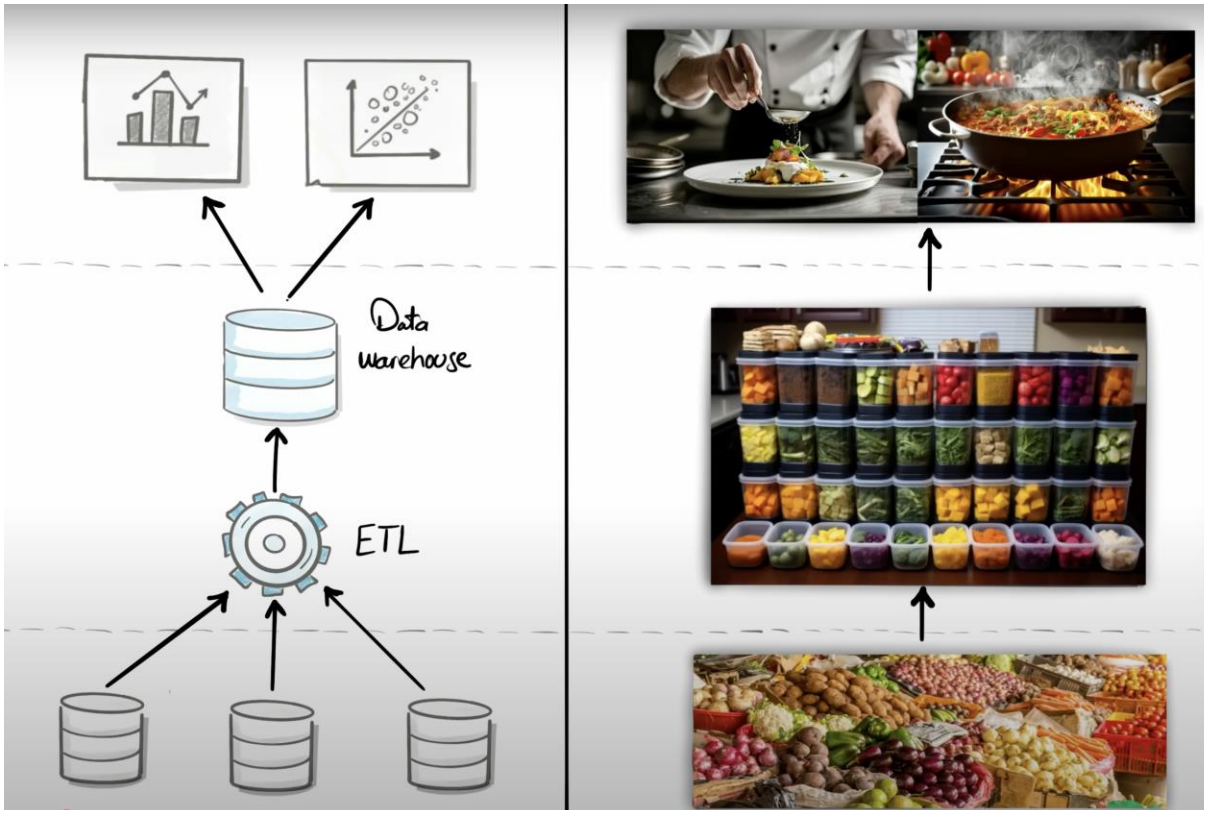
Data Warehouse project



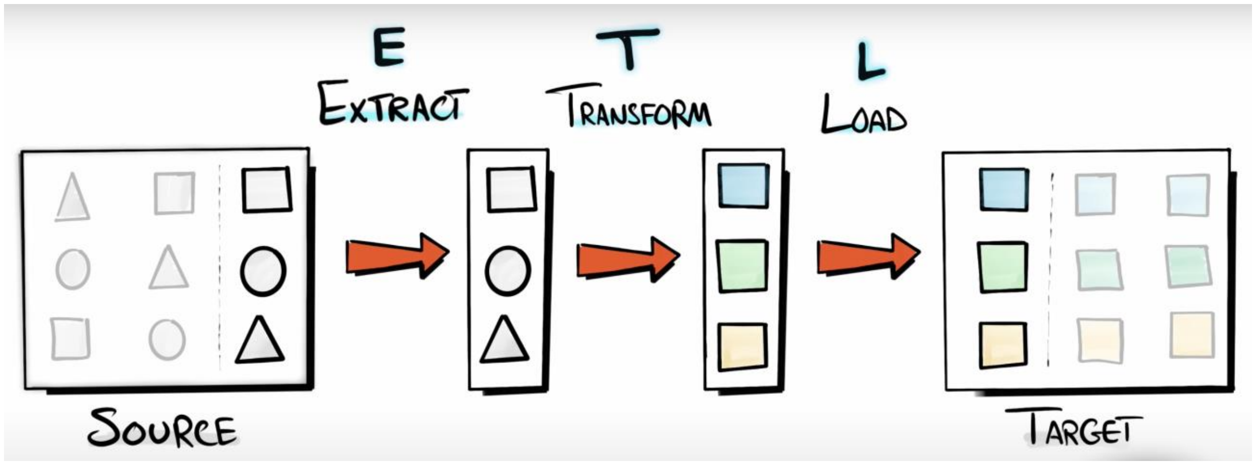
- Here we can pull the data into multiple sources and using ETL process we can store them in a Data warehouse.
- If we have a data warehouse in cloud platform, then we can easily handle even big data easily.

The example of a data warehouse:

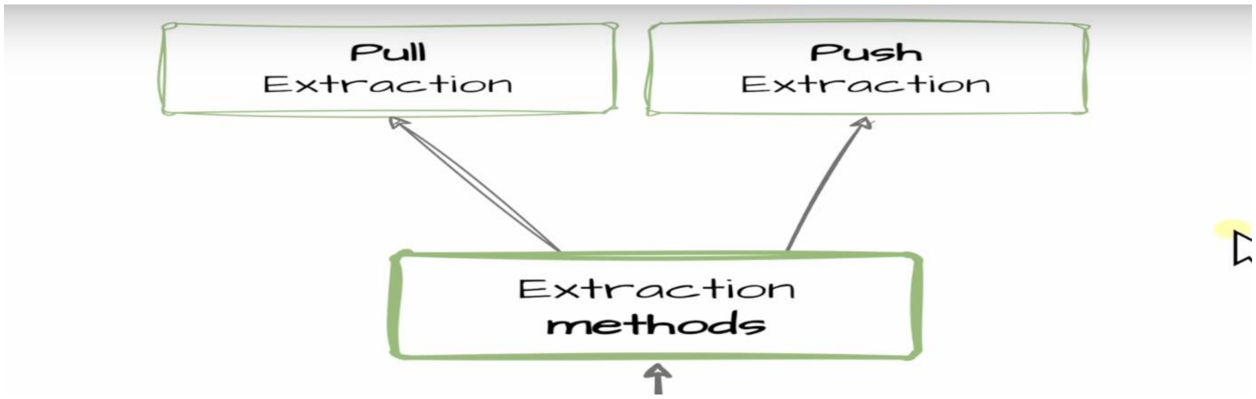


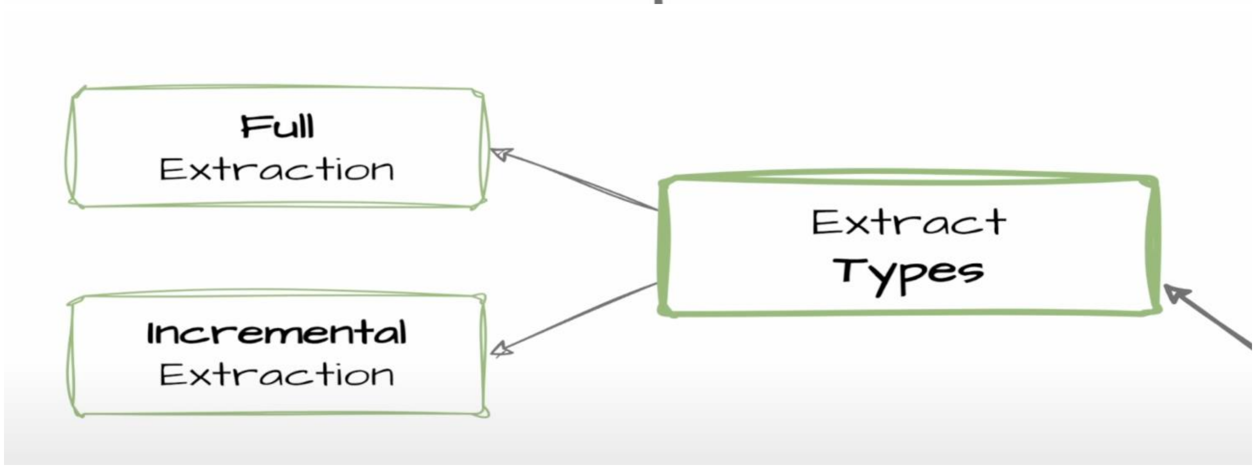
What is ETL?

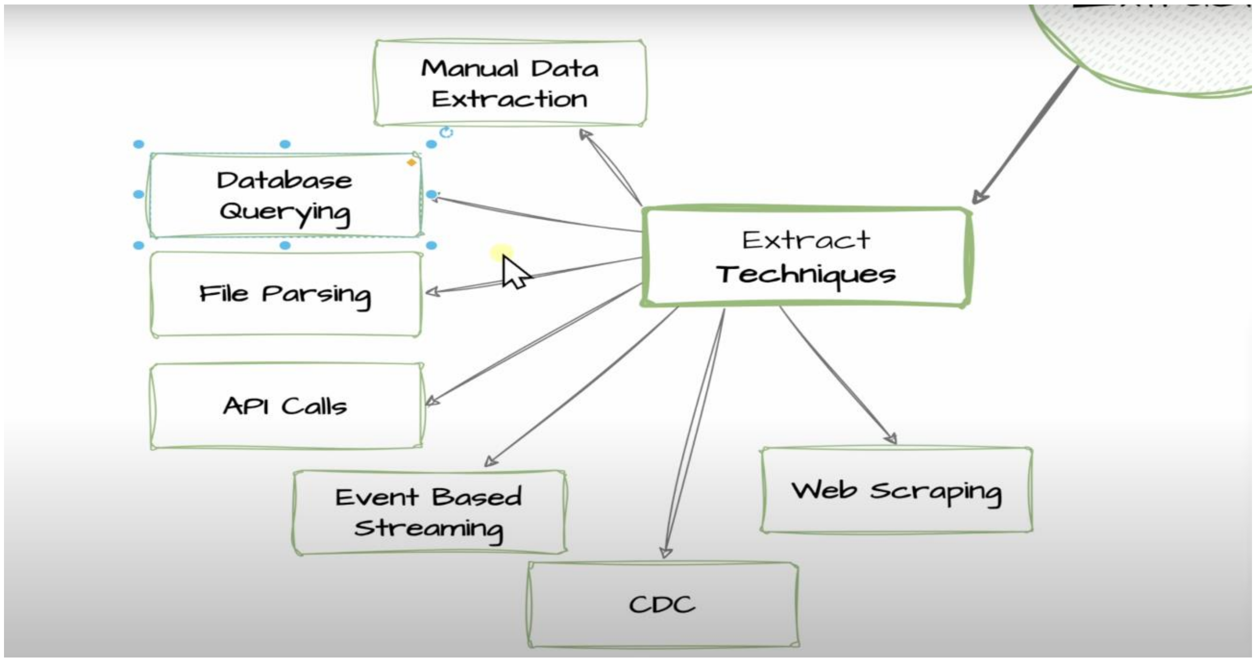
Ans: The process of collecting data from to store it into a Datawarehouse is called ETL.



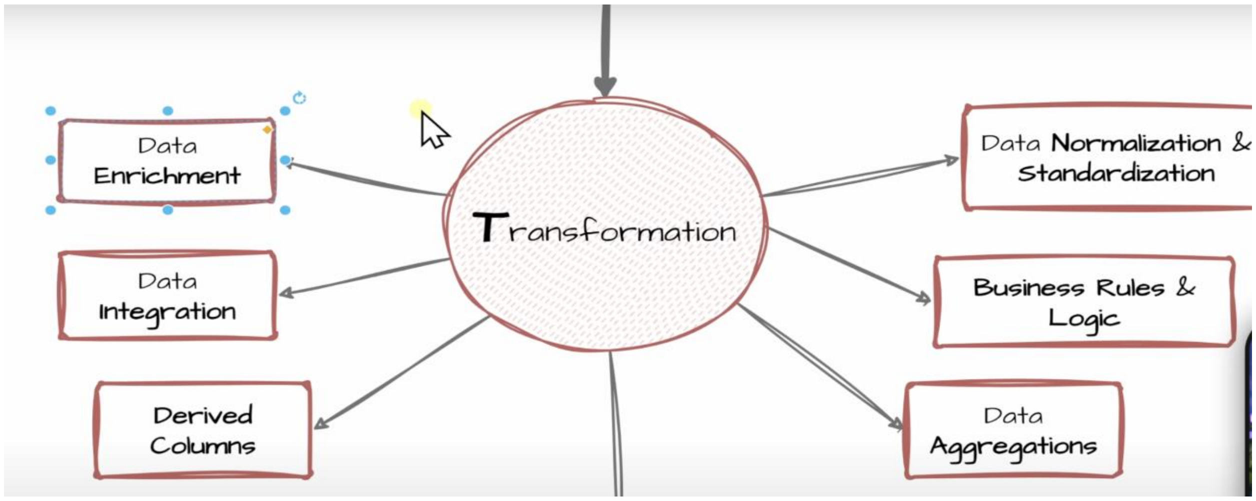
Extract

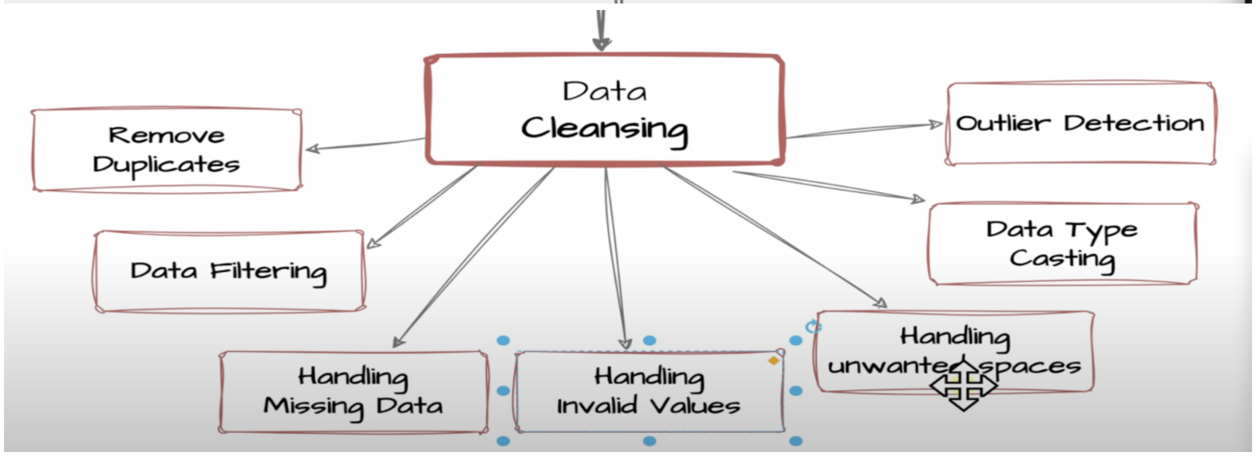




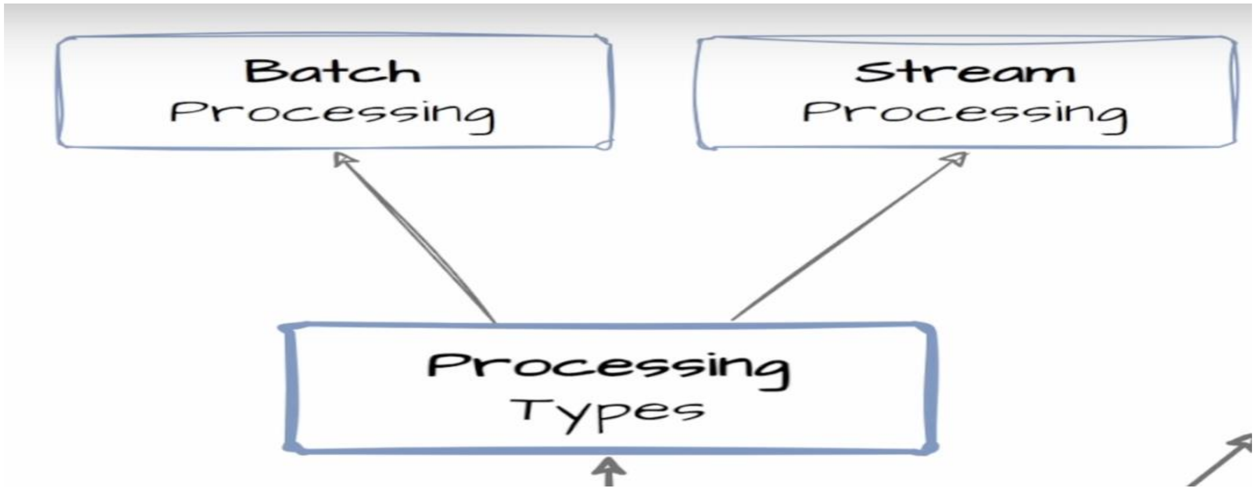


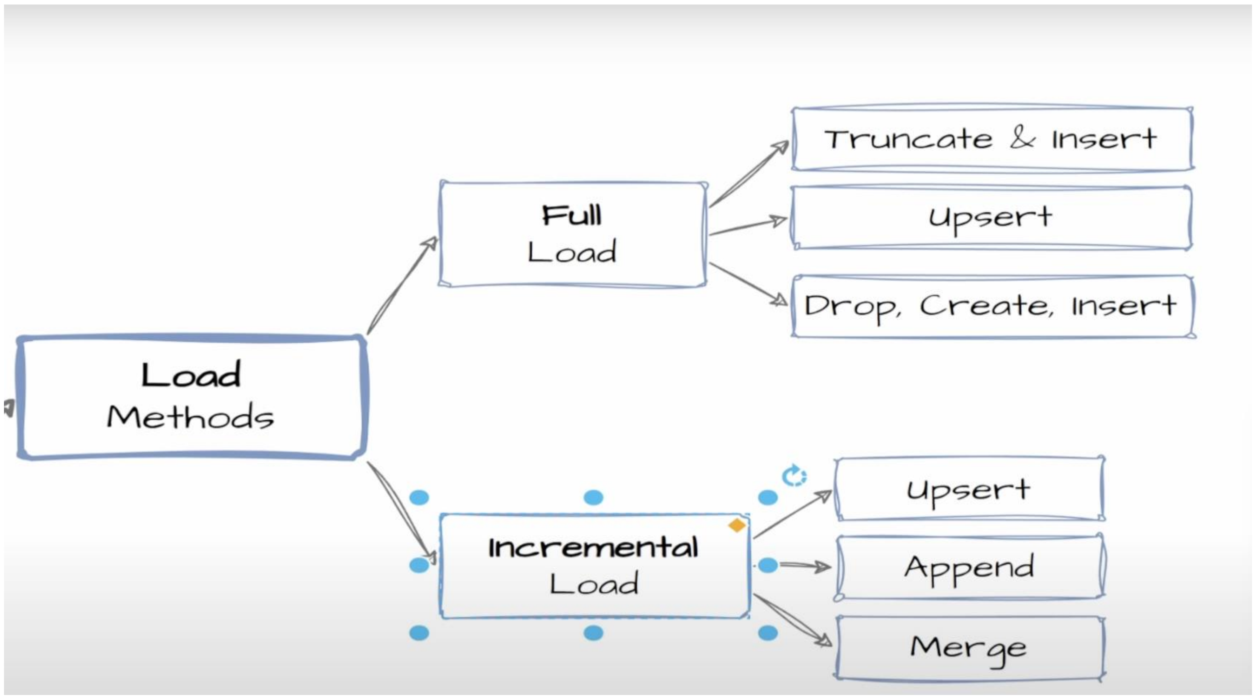
Transformation

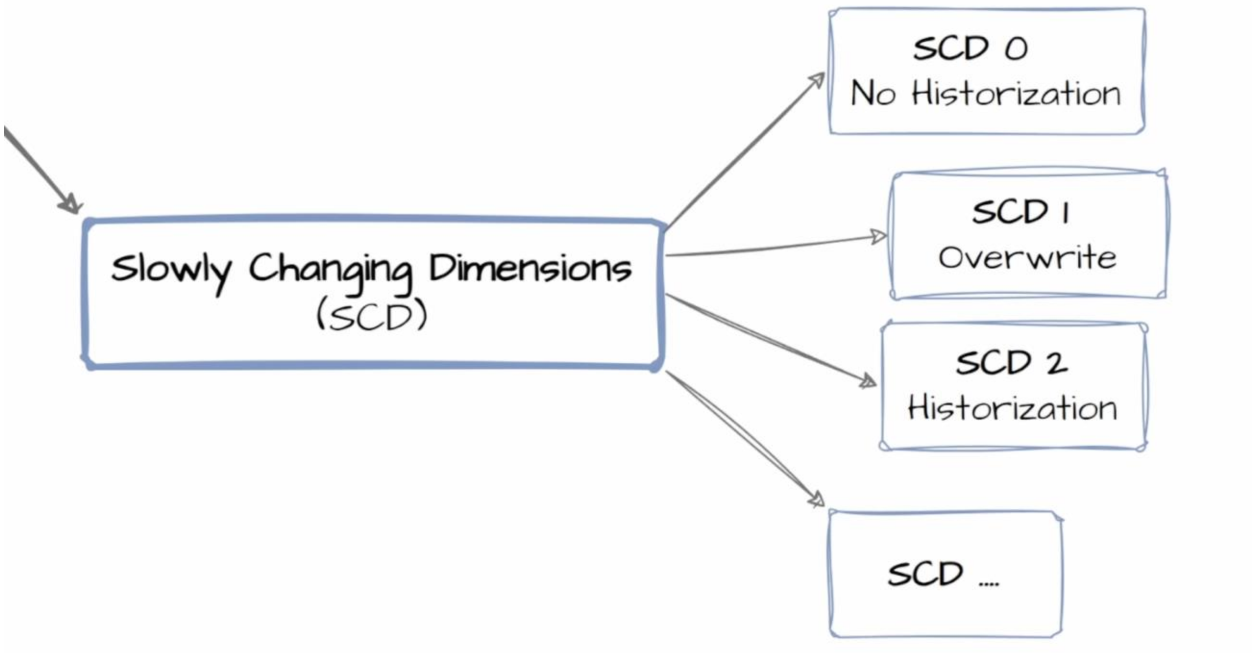

Transformation
Data Enrichment
Data Integration
Derived Columns
Data Normalization &
Standardization
Business Rules & Logic
Data Aggregations


Data Cleansing
Remove Duplicates
Data Filtering
Handling Missing Data
Handling Invalid Values
Handling unwanted spaces
Data Type Casting
Outlier Detection

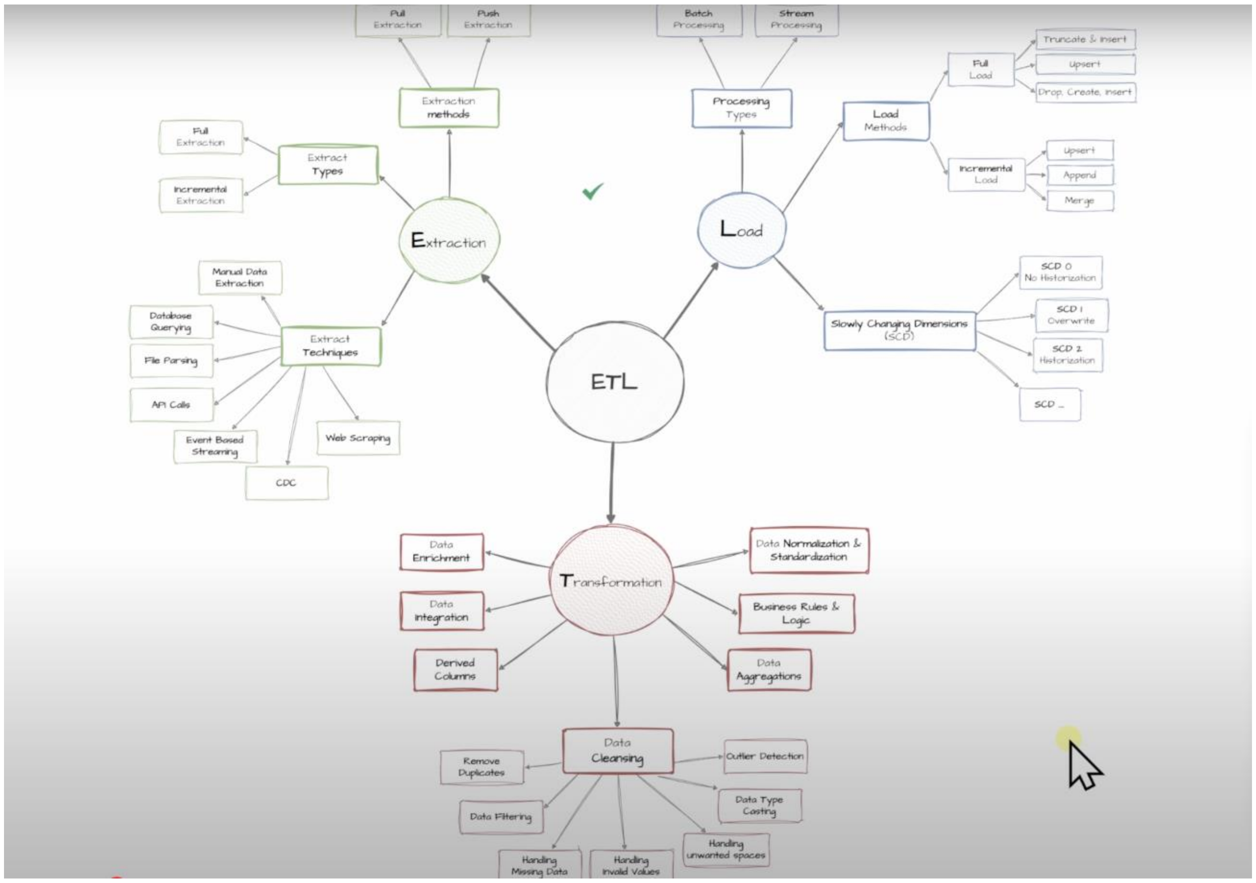
Load


Batch Processing
Stream Processing
Processing Types


Load Methods
Full Load
Truncate & Insert
Upsert
Drop, Create, Insert
Incremental Load
Upsert
Append
Merge


Slowly Changing Dimensions (SCD)
SCD 0 No Historization
SCD 1 Overwrite
SCD 2 Historization
SCD ....

ETL:



- Here in this project, we are going to do Extraction methods such as pull and Extraction type as Full Extraction and Extraction technique as File Parsing.
- As part of transformation, we are going to do all methods of transformation.

Project Tools

- Notion: It helps to organize the ideas, plans,resources into a single place

Steps Of projects (Epics)

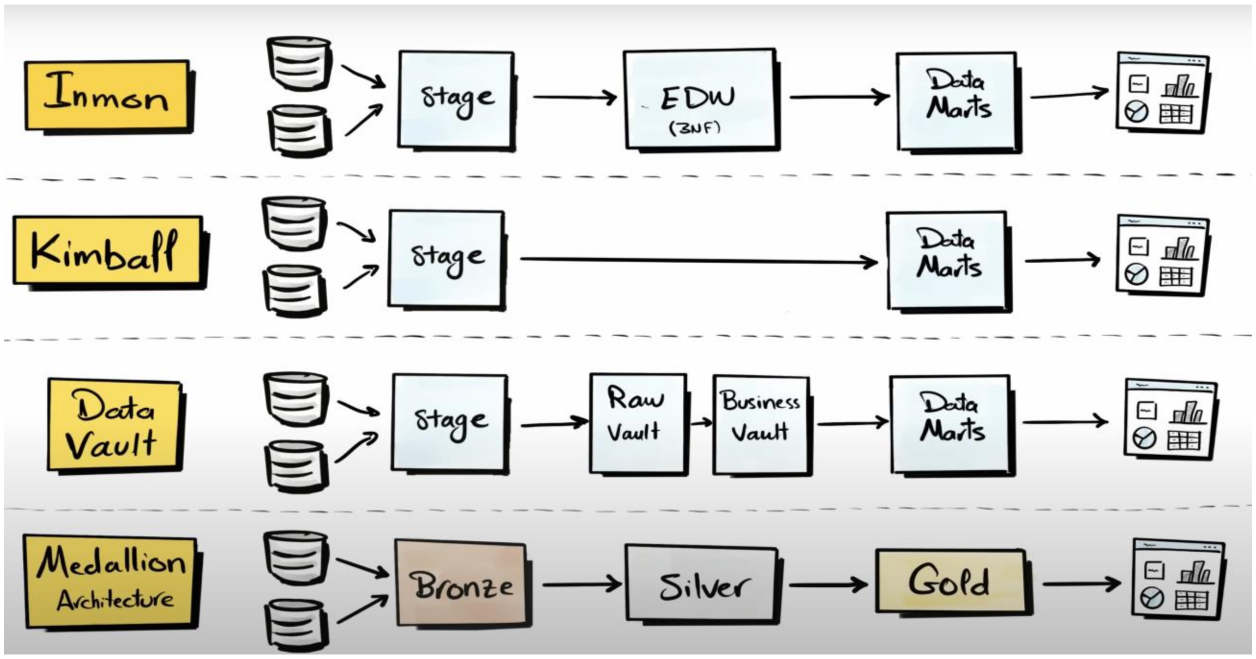
1) Requirement Analysis

- Before starting the work, we should be clear about the requirements of the business.
- We should be clear about the objective

2) Design the Data achitechture
   - Choose the Data management approach
   - Design rough layers of architecture
   - Drawing the data architecture (There is no thumb rule for creating the architecture, according to the type of architecture you are choosing you can draw the diagram.
3) Create a Project plan/Project Initialization
   - Create a detail Project task
   - Define the project naming convention
   - Create a Git Repo for the project
   - Create Database and Schema.
4) Buliding bronze Layer
   - Understand/Analyze the source system
   - Coding: data ingestion
   - Validating: Data completeness and Schema test
   - Document: Draw the data flow
   - Commit the code into Git
5) Building Silver Layer
   - Analyze: Explore and understand the data
   - Coding: Data cleaning (Check Quality, Transformation, Insert into silver
   - Validating: Data Correctness checks
   - Docs: Documenting and versioning to git
6) Building Gold Layer
   - Analyzing: Explore and understand the business logic
   - Coding: Data integration (Build the business object, choose table type (dimension, measure), convert the names into the friendly names)
   - Validation: Data quality checks
   - Docs & Version: Documenting and versioning to git

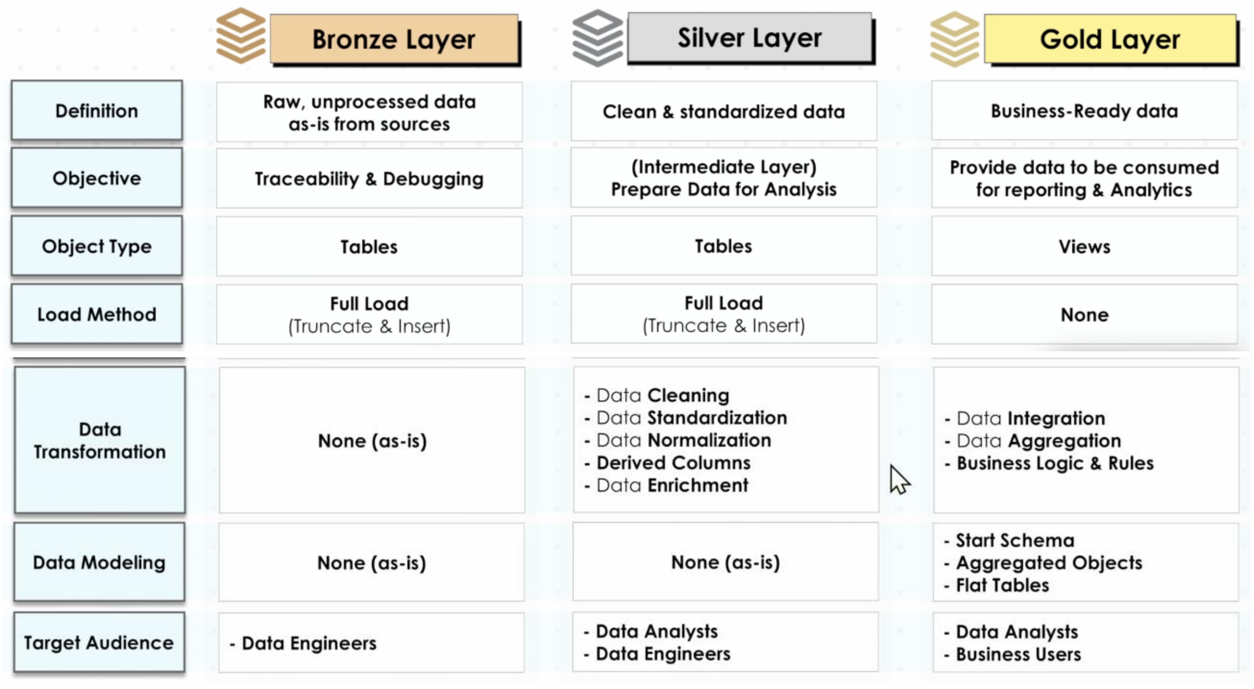
Data Architecture:

Below is the diff type of Data architecture:



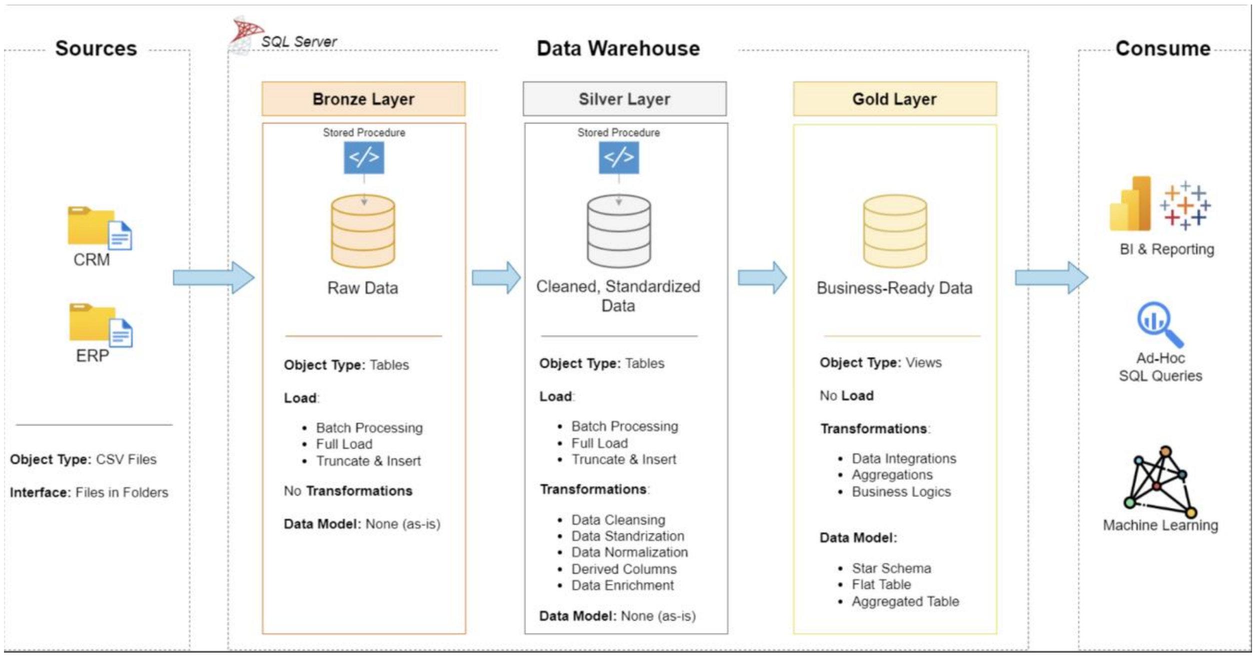
- Here we are selecting Medallion as the type of architecture, so below are the details of Medallion type.

Medallion Approach:



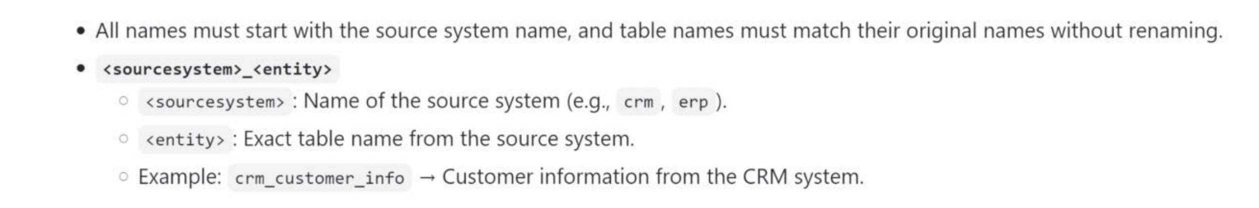
	Bronze Layer	Silver Layer	Gold Layer
Definition	Raw, unprocessed data as-is from sources	Clean & standardized data	Business-Ready data
Objective	Traceability & Debugging	(Intermediate Layer) Prepare Data for Analysis	Provide data to be consumed for reporting & Analytics
Object Type	Tables	Tables	Views
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

High-level Architecture



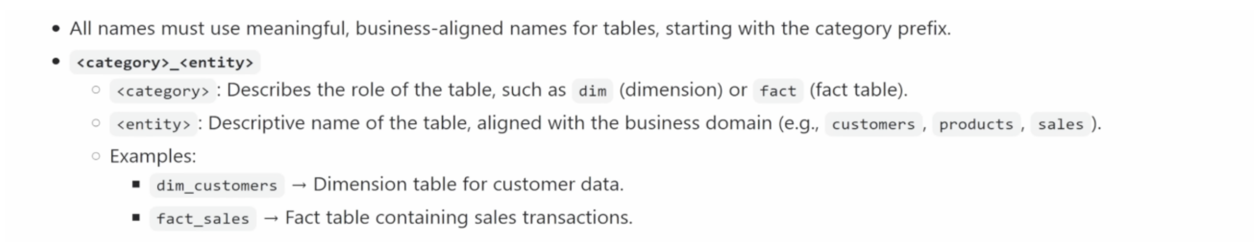
Project Initialization

- After creating the project task, we can go for proper naming convention of each objects (Database, Schema, Tables, Store procedure).
- There are certain rules to name the project (Camel Case, snake case, etc.)
- Bronze and Silver level:



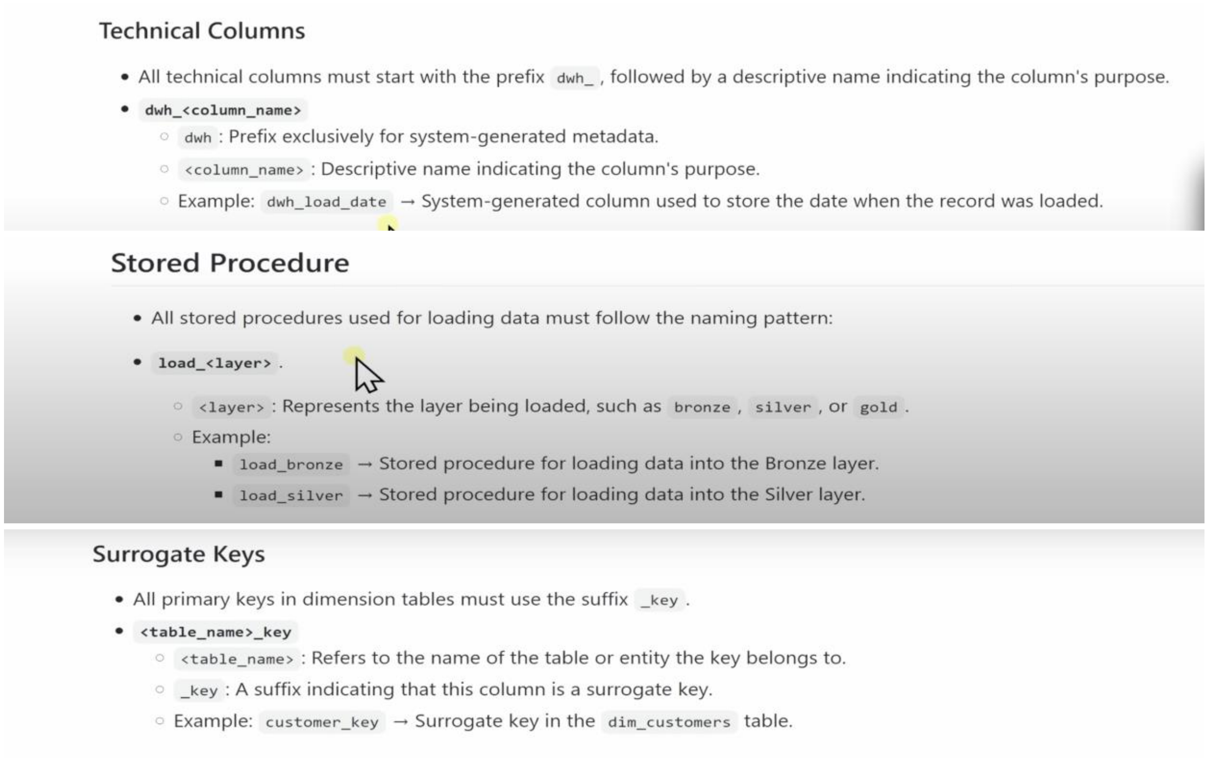
- All names must start with the source system name, and table names must match their original names without renaming.
- `<sourcesystem>_<entity>`
  - `<sourcesystem>` : Name of the source system (e.g., `crm` , `erp`).
  - `<entity>` : Exact table name from the source system.
  - Example: `crm_customer_info` → Customer information from the CRM system.

- Gold Level



- 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.

- Other Naming convention



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

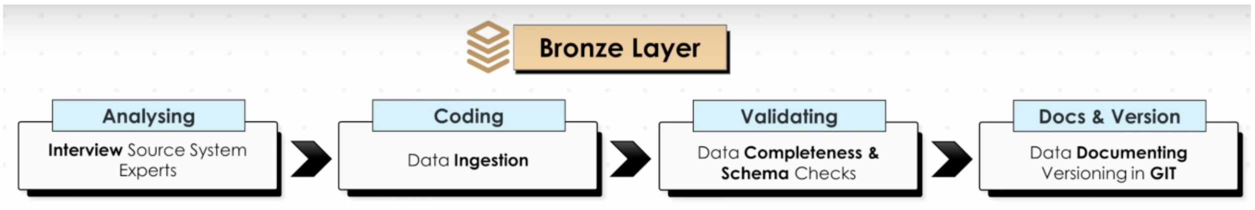
- All stored procedures used for loading data must follow the naming pattern:
- `load_<layer>`.
  - `<layer>` : Represents the layer being loaded, such as `bronze`, `silver`, or `gold`.
  - Example:
    - `load_bronze` → Stored procedure for loading data into the Bronze layer.
    - `load_silver` → Stored procedure for loading data into the Silver layer.

Surrogate Keys

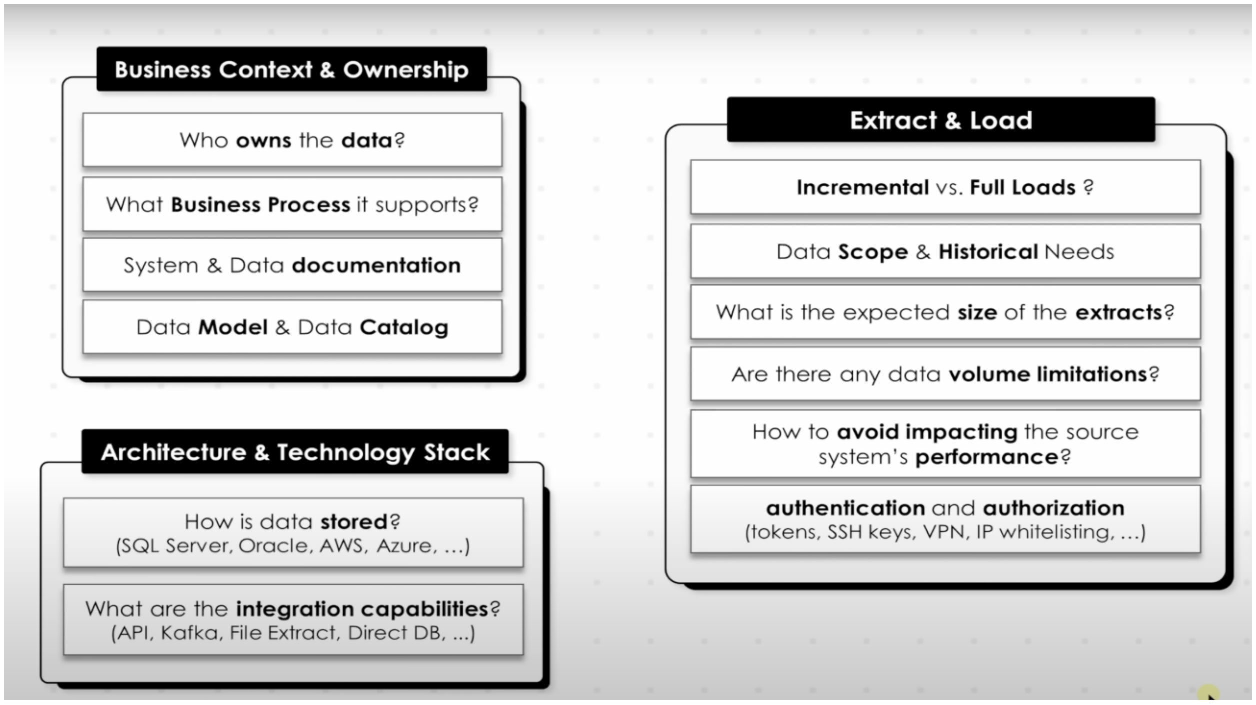
- All primary keys in dimension tables must use the suffix `_key`.
- `<table_name>_key`
  - `<table_name>` : Refers to the name of the table or entity the key belongs to.
  - `_key` : A suffix indicating that this column is a surrogate key.
  - Example: `customer_key` → Surrogate key in the `dim_customers` table.

- Create a proper git folder and prepare the repo structure for the project.
- Create Database in the any DB tool(MS SQL, etc.)
- Create schemas (Gold, Silver, Bronze) under Database.
- Now we should have an empty database and schemas.

Building Bronze layer


Bronze Layer
Analysing
Interview Source System Experts
Coding
Data Ingestion
Validating
Data Completeness & Schema Checks
Docs & Version
Data Documenting Versioning in GIT

1. One should have the proper understating about below question on the source system



Business Context & Ownership

Who **owns** the **data**?

What **Business Process** it supports?

System & Data **documentation**

Data **Model** & Data **Catalog**

Architecture & Technology Stack

How is data **stored**?
(SQL Server, Oracle, AWS, Azure, ...)

What are the **integration capabilities**?
(API, Kafka, File Extract, Direct DB, ...)

Extract & Load

Incremental vs. **Full Loads** ?

Data **Scope** & **Historical** Needs

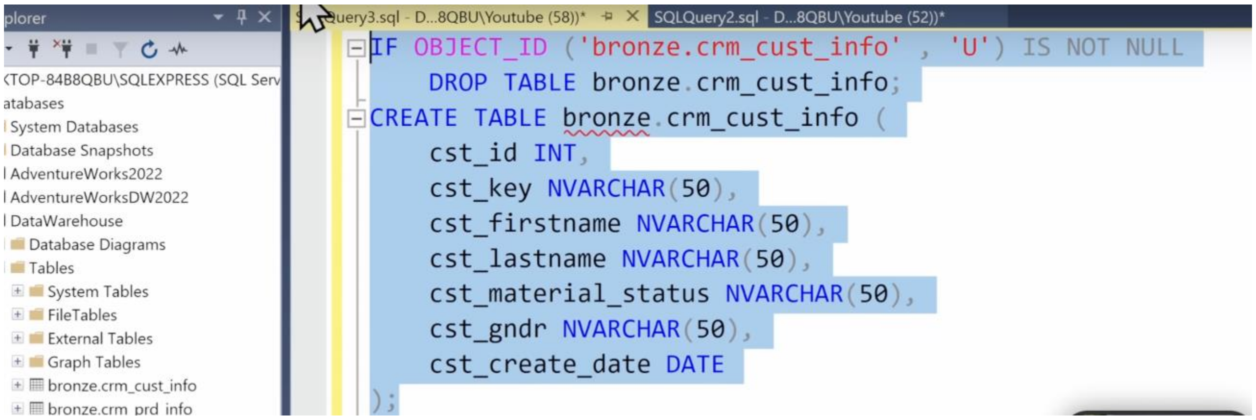
What is the expected **size** of the **extracts**?

Are there any data **volume limitations**?

How to **avoid impacting** the source system's **performance**?

authentication and **authorization**
(tokens, SSH keys, VPN, IP whitelisting, ...)

2. Create tables according to the source system data and load data

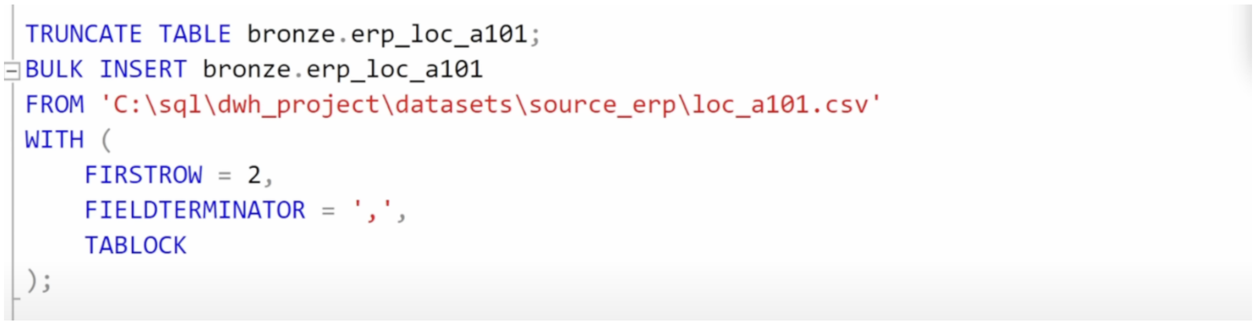


```
IF OBJECT_ID ('bronze.crm_cust_info' , 'U') IS NOT NULL
    DROP TABLE bronze.crm_cust_info;
CREATE TABLE bronze.crm_cust_info (
    cst_id INT,
    cst_key NVARCHAR(50),
    cst_firstname NVARCHAR(50),
    cst_lastname NVARCHAR(50),
    cst_material_status NVARCHAR(50),
    cst_gndr NVARCHAR(50),
    cst_create_date DATE
);
```

(Above query for creating table)

- Create the same DDL for all the available tables, then verify all the source system tables that have been created in the database.

- Then By bulk import method we can load multiple data from the source system to the database.
- Write the query for all the table

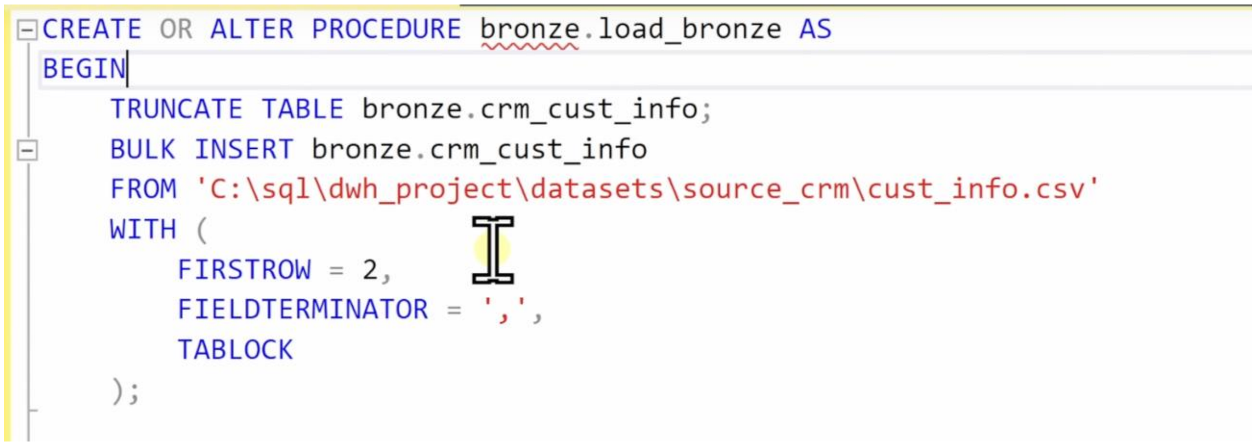


```sql
TRUNCATE TABLE bronze.erp_loc_a101;
BULK INSERT bronze.erp_loc_a101
FROM 'C:\sql\dwh_project\datasets\source_erp\loc_a101.csv'
WITH (
    FIRSTROW = 2,
    FIELDTERMINATOR = ',',
    TABLOCK
);
```

(Sample query for loading the data)

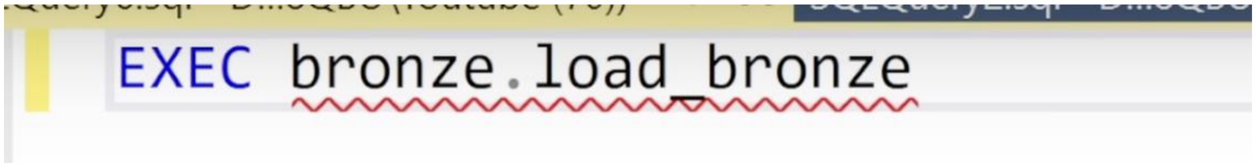
3. If you need to need to run the code on daily basis or frequently then better to write a store Procedure for this.

(Sample for store procedure)



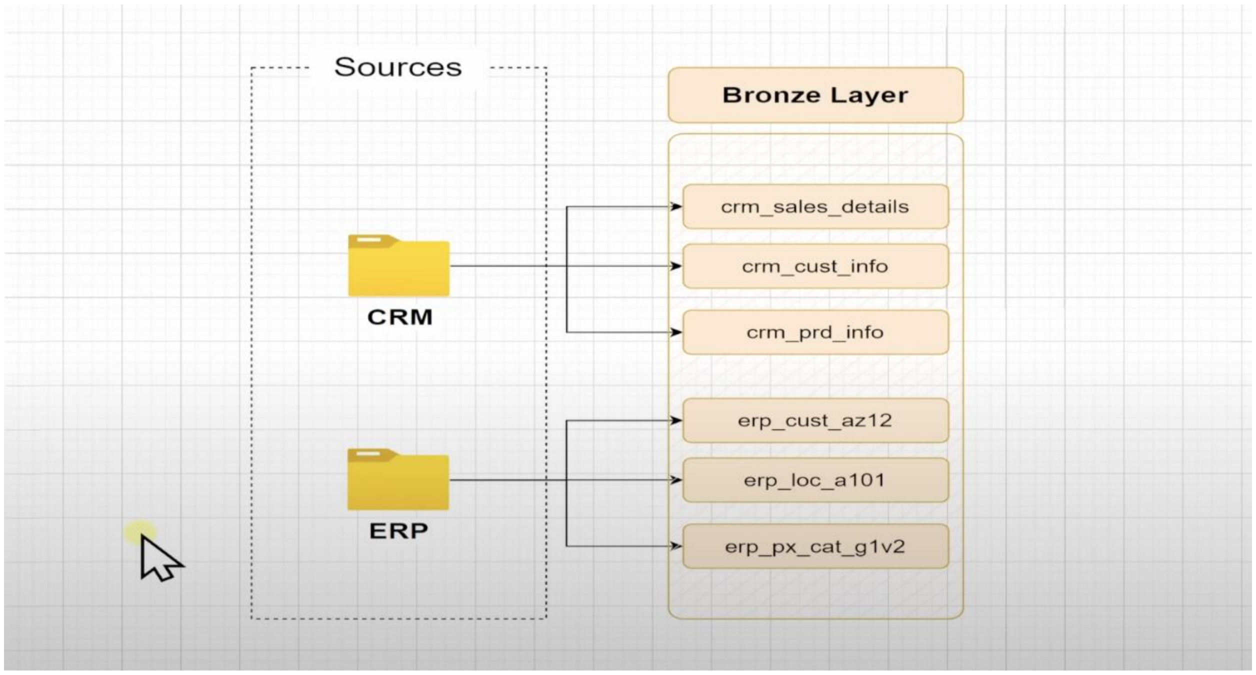
```sql
CREATE OR ALTER PROCEDURE bronze.load_bronze AS
BEGIN
    TRUNCATE TABLE bronze.crm_cust_info;
    BULK INSERT bronze.crm_cust_info
    FROM 'C:\sql\dwh_project\datasets\source_crm\cust_info.csv'
    WITH (
        FIRSTROW = 2,
        FIELDTERMINATOR = ',',
        TABLOCK
    );
```

Then we need to run only the below query, no need to run all the table query frequently

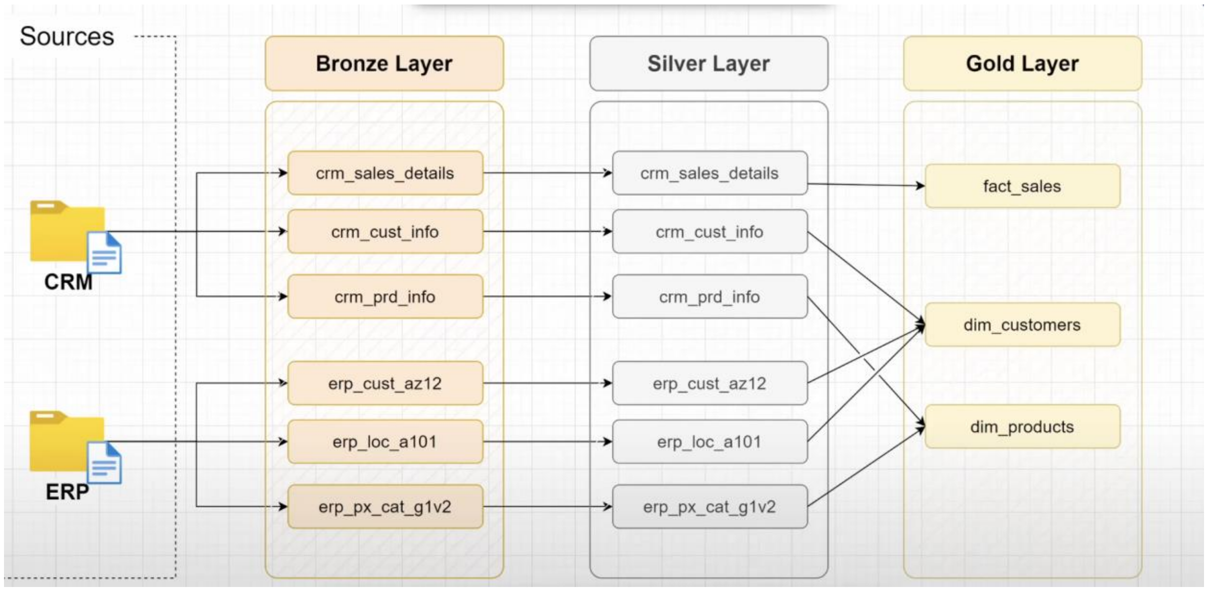


```sql
EXEC bronze.load_bronze
```

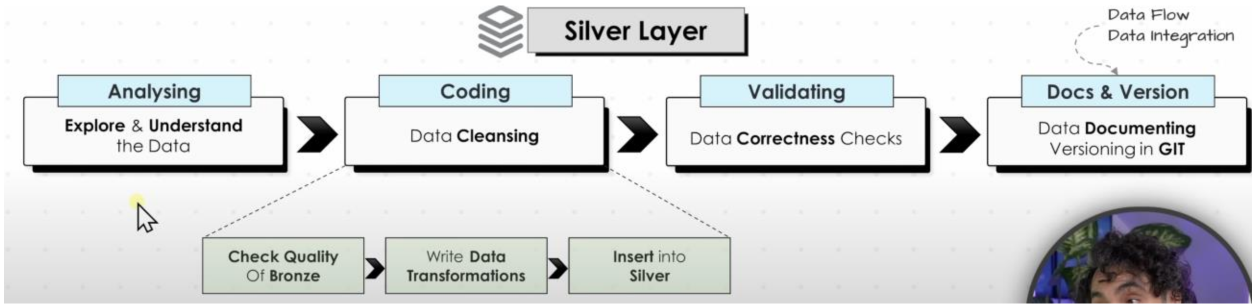
4. Data Flow



(Data flow for all the layers)



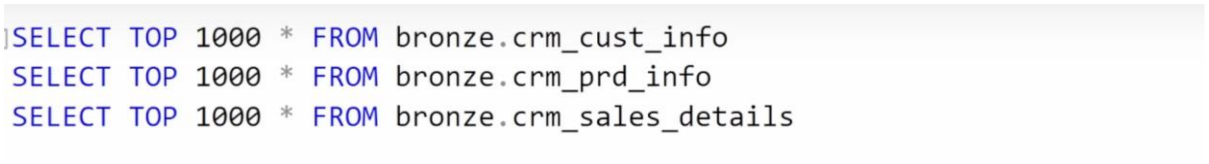
Building Silver layer



1. Analyzing (Explore & understand the data)

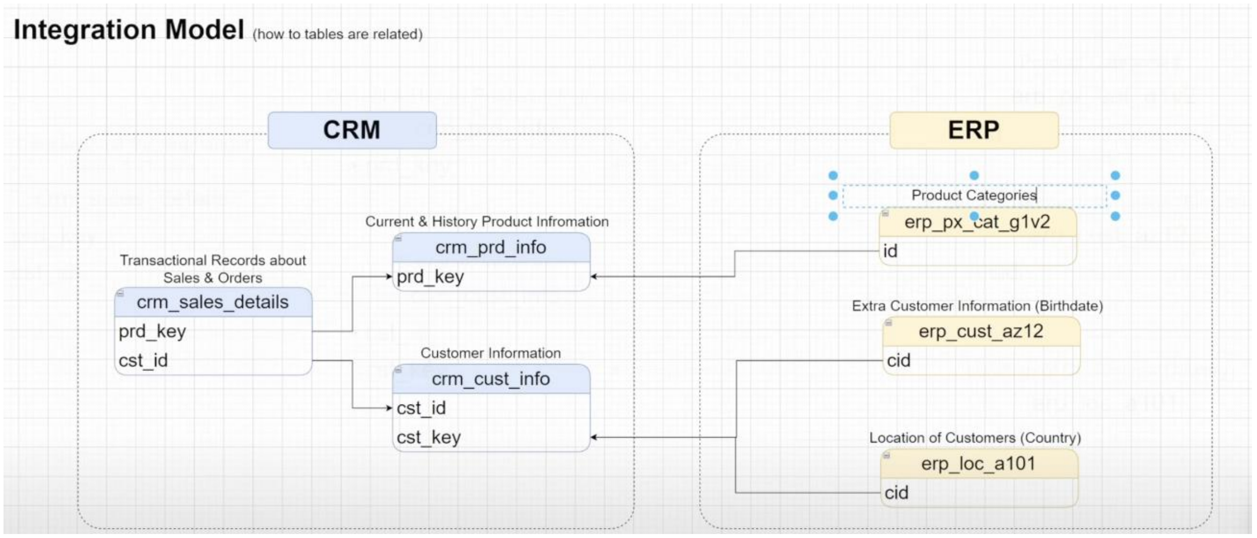
- Query to see the data of all the table

(Like this we can see the data)



```
SELECT TOP 1000 * FROM bronze.crm_cust_info
SELECT TOP 1000 * FROM bronze.crm_prd_info
SELECT TOP 1000 * FROM bronze.crm_sales_details
```

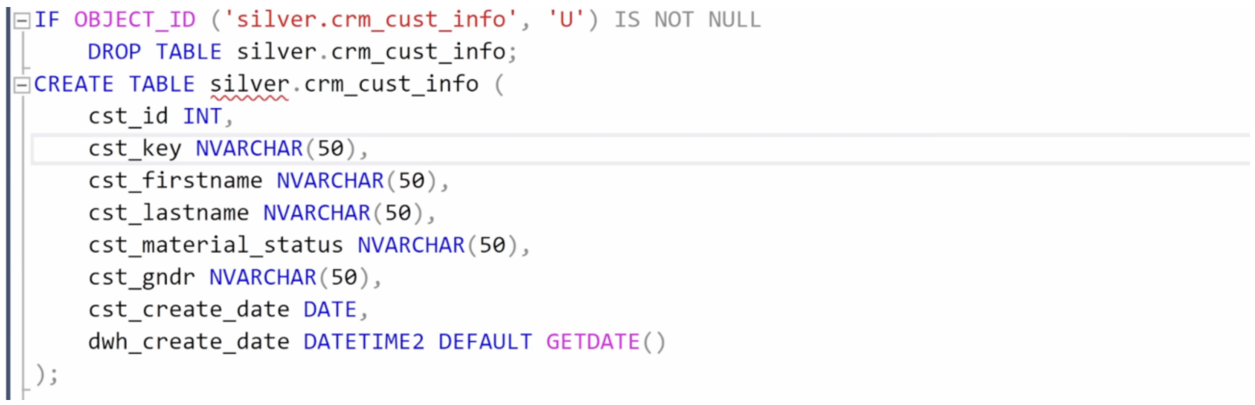
- Create data model like this


Integration Model (how to tables are related)
CRM
ERP
Product Categories
erp_px_cat_g1v2
id
Current & History Product Infromation
crm_prd_info
prd_key
Transactional Records about Sales & Orders
crm_sales_details
prd_key
cst_id
Extra Customer Information (Birthdate)
erp_cust_az12
cid
Customer Information
crm_cust_info
cst_id
cst_key
Location of Customers (Country)
erp_loc_a101
cid

2.Coding (Data cleansing)

- Writing Store procedure for silver layer to load the table in silver layer

(below is the sample query for the one of the table)



```
IF OBJECT_ID ('silver.crm_cust_info', 'U') IS NOT NULL
    DROP TABLE silver.crm_cust_info;
CREATE TABLE silver.crm_cust_info (
    cst_id INT,
    cst_key NVARCHAR(50),
    cst_firstname NVARCHAR(50),
    cst_lastname NVARCHAR(50),
    cst_material_status NVARCHAR(50),
    cst_gndr NVARCHAR(50),
    cst_create_date DATE,
    dwh_create_date DATETIME2 DEFAULT GETDATE()
);
```

- Then we can go start cleaning the data from the bronze layer and then add in the silver layer.
    1. Check for Nulls or duplicates in primary key
    2. Replace Invalid value
    3. Check for unwanted space
    4. Data standardization and consistency
    5. Convert the low cardinality column into meaningful column
    6. Convert to date time function
- We must perform these above steps for all the table present in the source
- After cleaning the data, we can perform the data transform in the tables
- Then we can load the data into the silver layer. (Here we are doing fool load)
  (Use truncate before running the insert query)

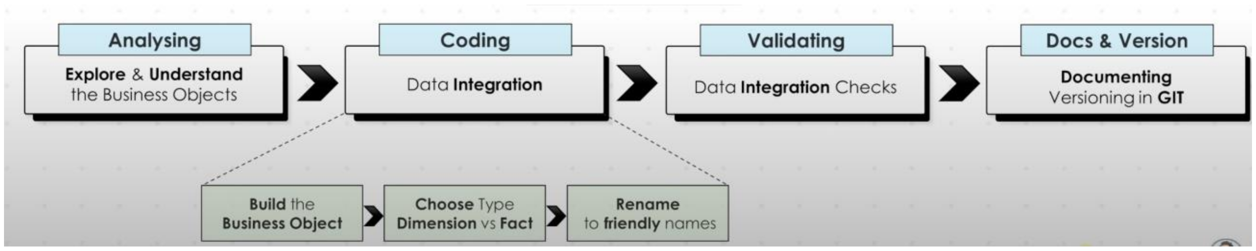
3. Validating (data correctness check)

- While writing the store procedure for silver layer, make sure we are keeping all our code in the try catch block
- We need to give proper print statement to check all the tables get inserted or not
- We can also add a couple of methods like time, etc. to check the total time required to execute the procedure.
- We must use error handling as well

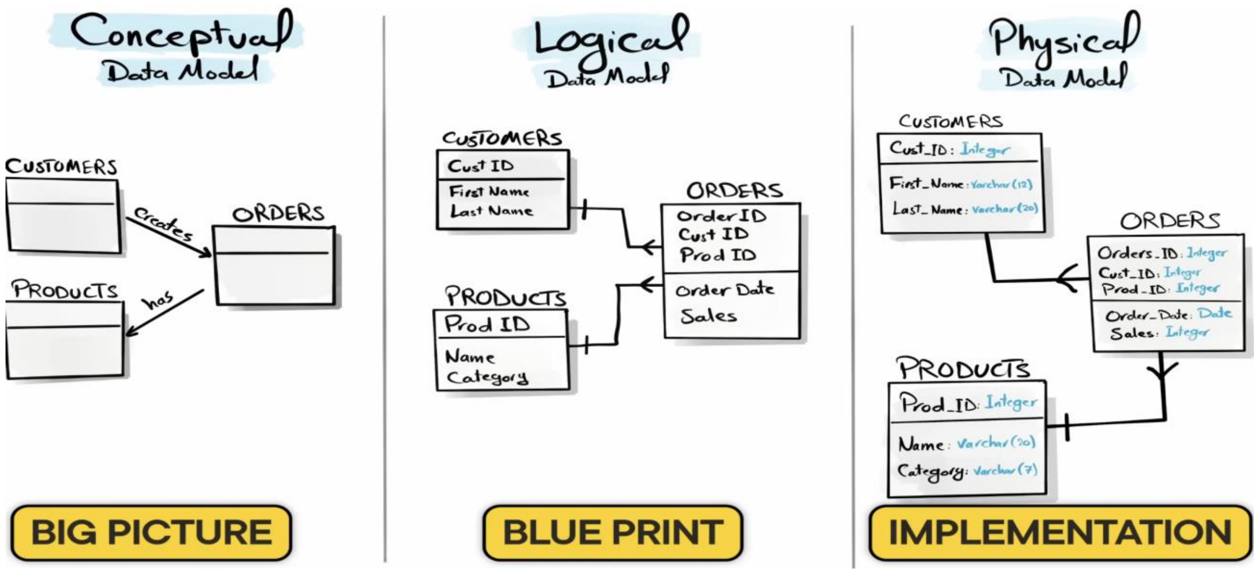
4. Doc & Version (Data documenting and pushing into git)

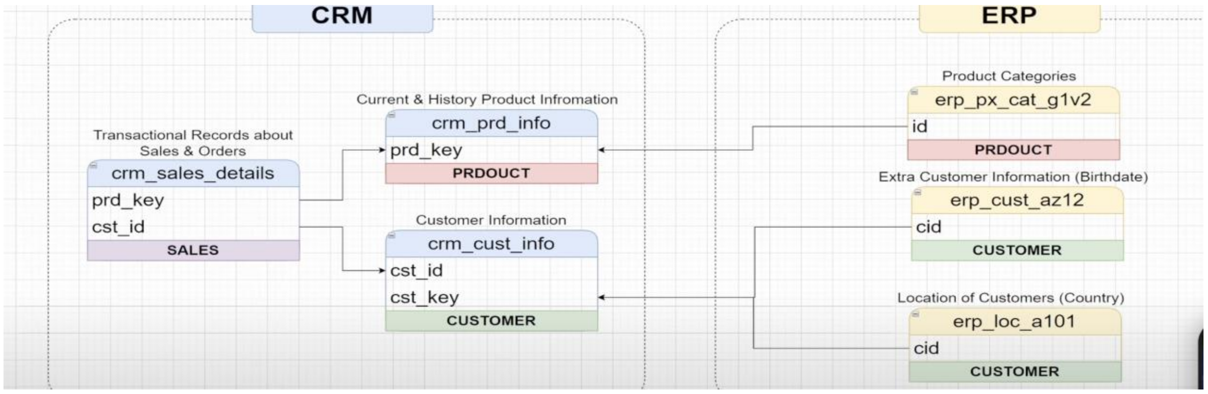
- Create a file under script to upload the Store procedure query
- Create a folder/file name as test to upload all the data cleaning or transformation query that we wrote to clean the tables.

Building Gold layer


Analysing
Explore & Understand the Business Objects
Coding
Data Integration
Validating
Data Integration Checks
Docs & Version
Documenting Versioning in GIT
Build the Business Object
Choose Type Dimension vs Fact
Rename to friendly names

1. Analyzing: Explore & Understand Business

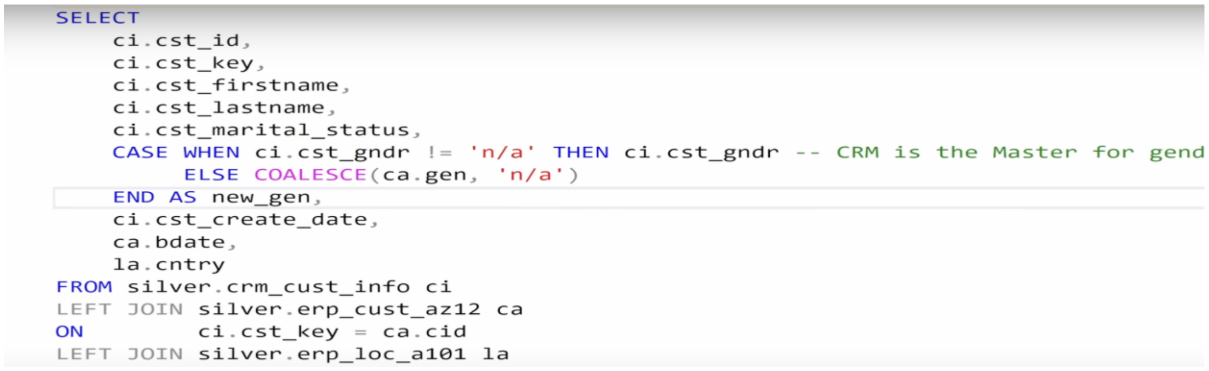
- 

- Above is the type of data model present, but here we are using Logical layer model,
- We are using Star schema here
- 

  Here we are adding the labels at the bottom of the tables to make it easy to understand the tables' content.

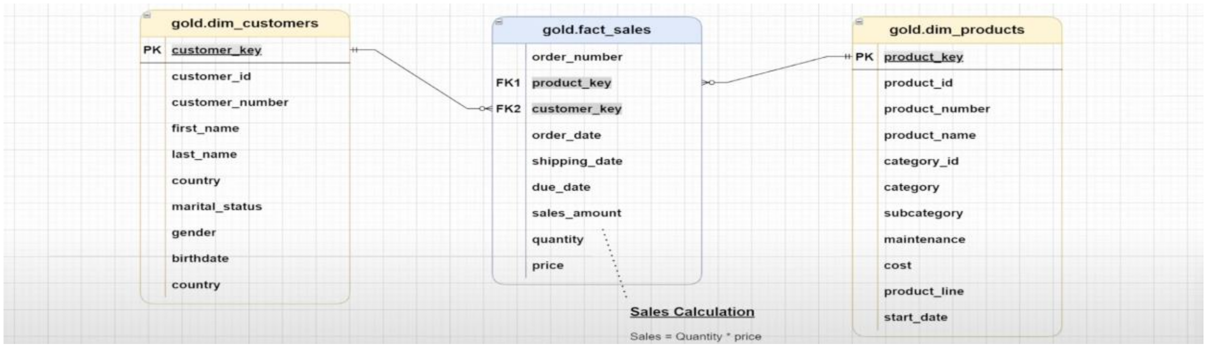
2&3. Data Integration and Validation

- Here we are collecting the customer information from two source system (erp, crm)

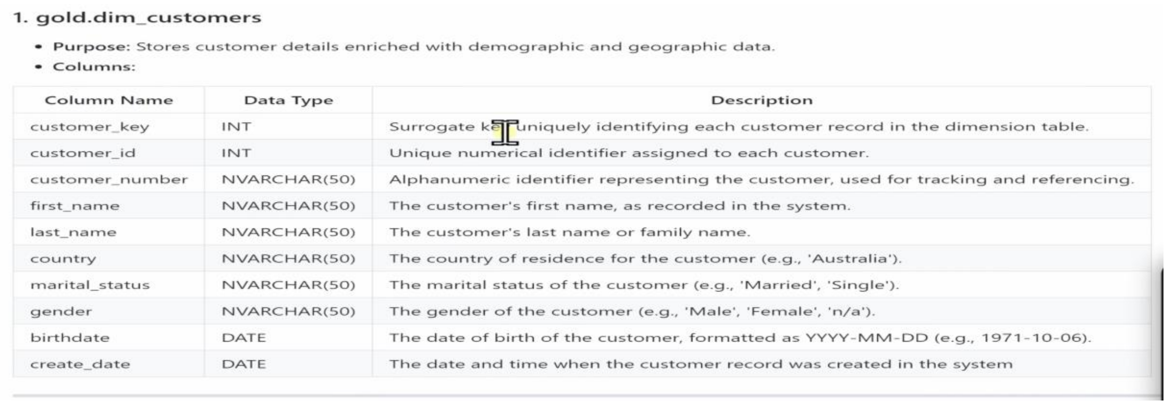


```
SELECT
    ci.cst_id,
    ci.cst_key,
    ci.cst_firstname,
    ci.cst_lastname,
    ci.cst_marital_status,
    CASE WHEN ci.cst_gndr != 'n/a' THEN ci.cst_gndr -- CRM is the Master for gend
         ELSE COALESCE(ca.gen, 'n/a')
    END AS new_gen,
    ci.cst_create_date,
    ca.bdate,
    la.cntry
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
```

- Convert the column name to some easy and understandable name
- Make the proper order of the column.
- Find out whether it is a dimension or a fact table. Then start creating the dimension table and the fact table.
- Make sure we have some primary key present, if not then create it on basis of existing column.
- Then create a view of the gold level.
- Create a data model as per the table (Dimension & Fact) that has been created



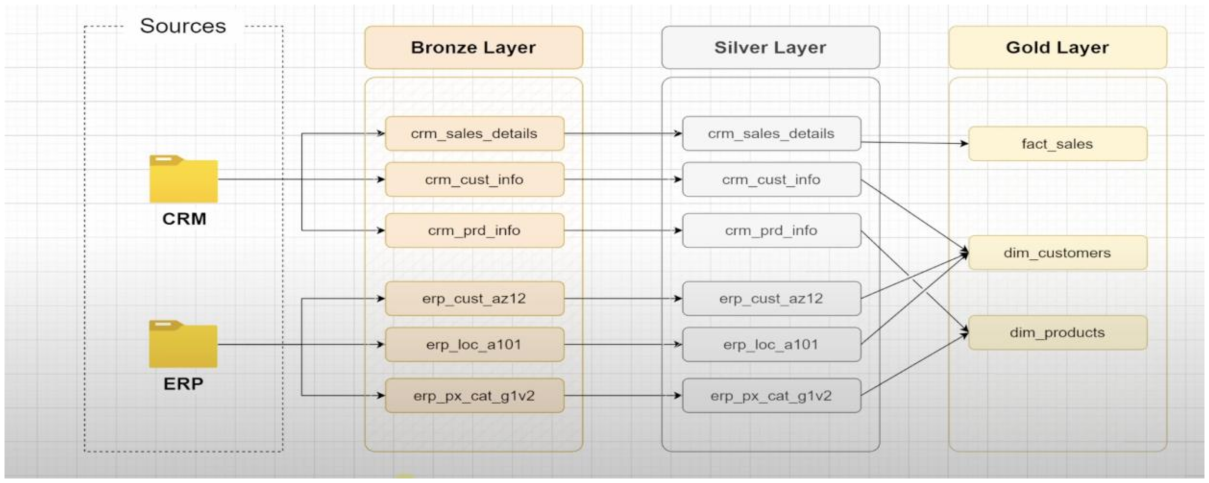
- Create a data catalogue (Catalogue describes the details of the data like column info, connection type, etc.)



1. gold.dim_customers
   - **Purpose:** Stores customer details enriched with demographic and geographic data.
   - **Columns:**

Column Name	Data Type	Description
customer_key	INT	Surrogate key uniquely identifying each customer record in the dimension table.
customer_id	INT	Unique numerical identifier assigned to each customer.
customer_number	NVARCHAR(50)	Alphanumeric identifier representing the customer, used for tracking and referencing.
first_name	NVARCHAR(50)	The customer's first name, as recorded in the system.
last_name	NVARCHAR(50)	The customer's last name or family name.
country	NVARCHAR(50)	The country of residence for the customer (e.g., 'Australia').
marital_status	NVARCHAR(50)	The marital status of the customer (e.g., 'Married', 'Single').
gender	NVARCHAR(50)	The gender of the customer (e.g., 'Male', 'Female', 'n/a').
birthdate	DATE	The date of birth of the customer, formatted as YYYY-MM-DD (e.g., 1971-10-06).
create_date	DATE	The date and time when the customer record was created in the system

- Draw the final data flow diagram



- Then commit all the script to the Git