Building a Preliminary Data Warehouse for a D2C-Gifting Brand

Background: A rapidly growing ***D2C (Direct-to-Consumer) gifting brand*** is expanding its digital footprint and generating large volumes of data across multiple sources through e-commerce transactions, customer interactions, product inventories, marketing campaigns, and user feedback. The brand wants to evaluate the feasibility of a full-scale data warehouse by constructing a preliminary version using a sample chunk of its data.

Objective:
The primary goal is to design and implement a ***simple data warehouse architecture*** that efficiently stores, segregates, organizes, and processes data from multiple sources. This will help the company assess scalability, performance, and ease of data retrieval before committing to a full-scale deployment.

Tools Used

Python (Pandas and Numpy Libraries), MySQL Workbench, Microsoft Power BI and Microsoft Excel.

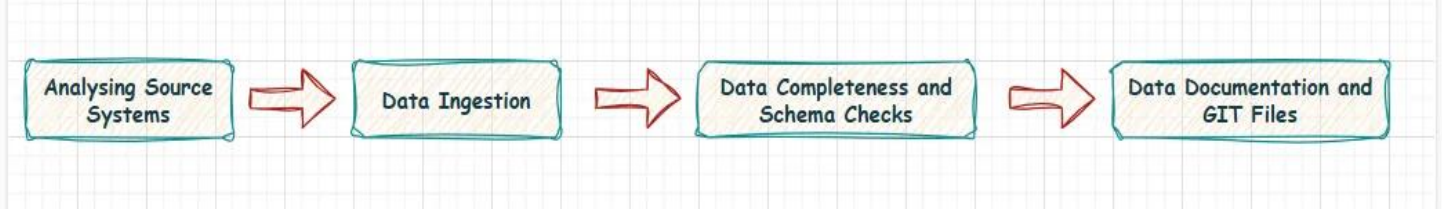
Role as a Data Analyst

As a Data Analyst, my responsibility is to propose and design the architecture of the preliminary data warehouse, ensuring that it aligns with the company's business goals. It includes:

- Identify key data sources (e.g., customer databases, marketing platforms).
- Design an optimized schema that balances performance and storage efficiency.
- Select appropriate database technologies (e.g., SQLServer, Snowflake, or BigQuery).
- Develop ETL (Extract, Transform, Load) processes to ingest and clean data.
- Analyze query performance to determine how well the warehouse handles large datasets. Provide insights and recommendations on scaling to a full-fledged warehouse.

Key Learnings 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: Fact and dimension tables for analytics.
Reporting: Create SQL and Power BI based dashboard to generate meaningful insights that businesses can act on.

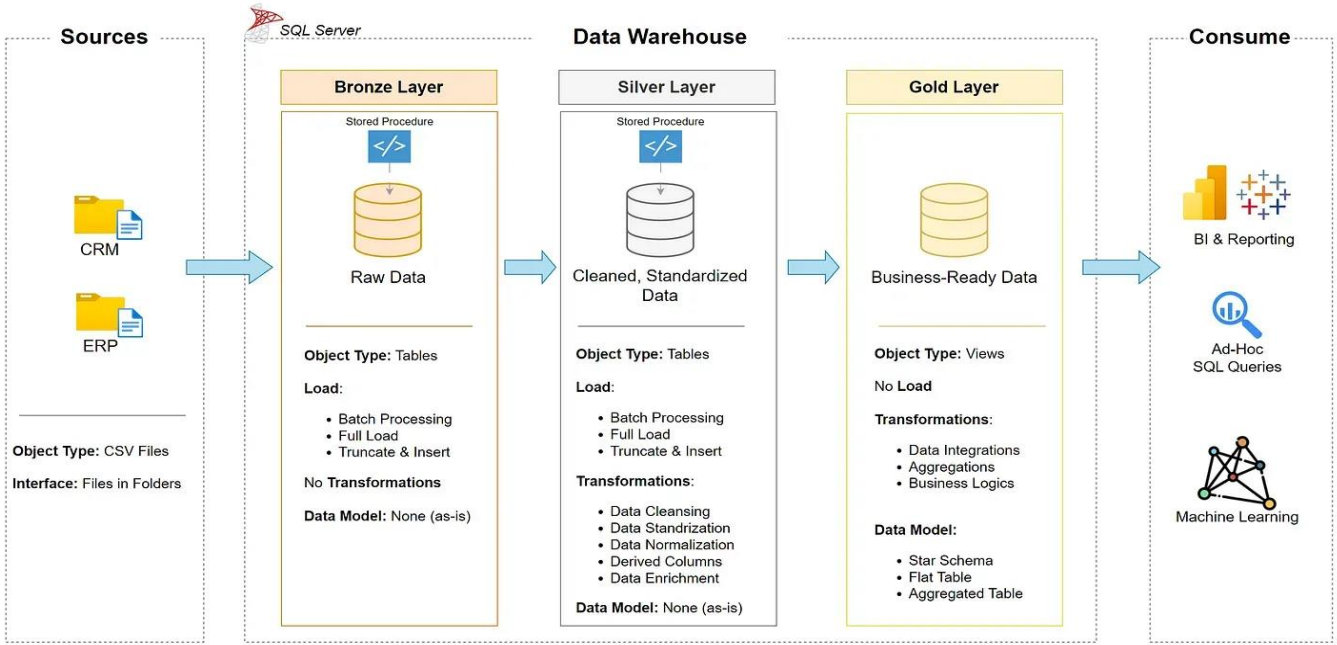


SQL Concepts Leveraged

- Aggregate Functions
- Window Functions
- Stored Procedures
- CASE Statements
- JOIN Operations
- Built-in STRING Functions
- Data Modelling

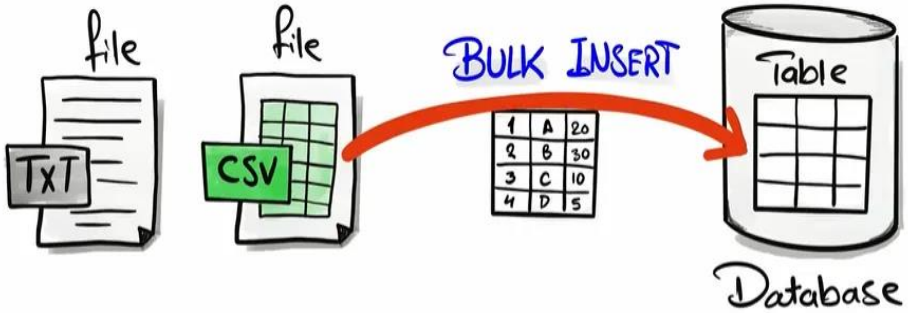
Architectural Setup of the Warehouse

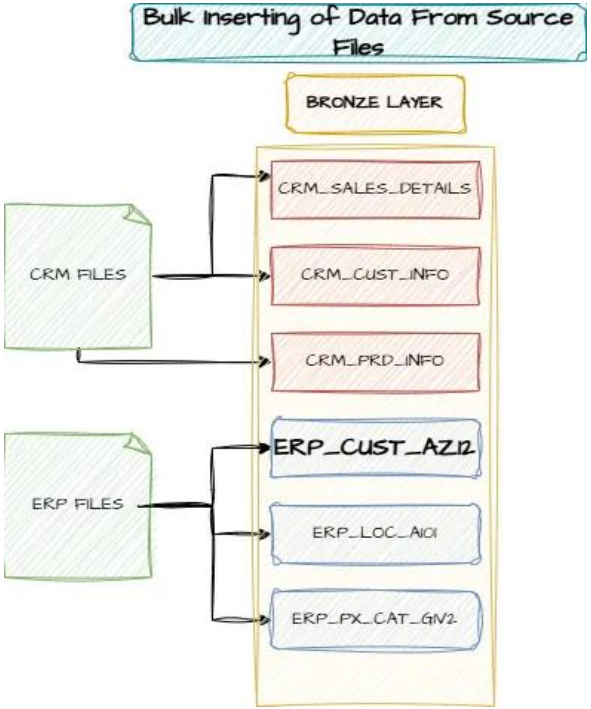
- **Bronze Layer:** Raw data is ingested as-it-is from CSV files into SQL Server.
- **Silver Layer:** Data is cleaned, standardized, and normalized for analysis.
- **Gold Layer:** Business-ready data is modelled in a star schema for reporting and analytics.
- **Documentation and Analysis:** Provide clear documentation of the data model to support both business stakeholders and analytics.



Building Bronze Layer

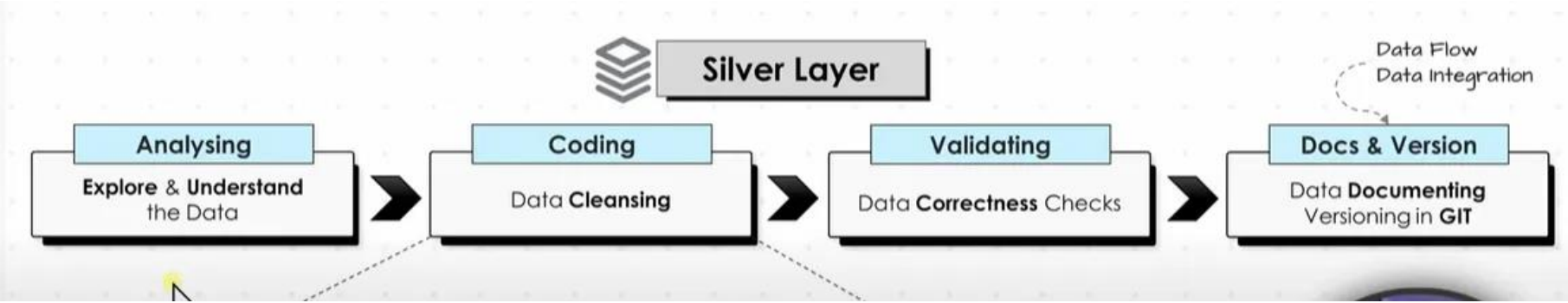
- The main aim behind building the bronze layer is to facilitate the creation of the schemas and perform bulk inserting of data from the source systems.
- In this layer, simply the loading of the data will occur. Any transformations of cleaning part will be done in the **Silver Layer.**





Building Silver Layer

- **Data Understanding:** Analyze the loaded data to gain insights into its structure and key attributes.
- **Inconsistency Check:** Identify any missing, duplicate, or erroneous values that may impact analysis.
- **Data Transformation:** Apply necessary preprocessing techniques to clean and standardize the data for accuracy and consistency.



Exploratory Data Analysis

Identifying the Primary Key Columns For Each table

This task involves identifying those columns which can serve as the primary key and help build ***Primary-Foreign Key relationships*** among tables or help us to join the tables to fetch data.

Table Name	Primary Key
crm_cust_info	(cst_id,cst_key) (composite primary key)
crm_prd_info	(prd_id)
crm_sales_details	(sls_prd_key,sls_cust_id,sls_ord_num) (composite primary key)
erp_loc_a101	(cid)
erp_cust_az12	(cid)
erp_px_cat_g1v2	(id)

Exploratory Data Analysis

Building a Clean and Consolidated Dataset for Product Details

In this task, to fetch the details of all the products, we need to use two tables: ***CRM_prd_info and ERP_PX_CAT_G1V2.***

- To ensure efficient **JOIN operations**, a connection must be established between these tables through **PK**.
- This will be achieved by extracting a substring from **CRM_prd_info.prd_key** and matching it with the entire column **ERP_px_cat_g1v2.id.**

Step 1 Extracted the Category ID: Pulled out the **category_id** from the **prd_key** column and used it as the primary key for the table.

Step 2 Created a New Product Key: Extracted the remaining part of **prd_key**, excluding **category_id**, and used it as the new product key.

Step 3 Fixed Incorrect Date Order: Identified that **prd_end_dt** was earlier than **prd_start_dt**. The two columns were swapped and renamed as **Manufacturing Date** and **Warranty Date.**

Step 4 Handled Warranty Dates: For rows where Manufacturing Date was equal to Warranty Date, the Warranty Date was set to NULL in the subsequent **`SELECT`** statement.

Step 5 Created the Final Table: After performing all the transformations, the final table was created in the Silver Layer and named **Final_Product_Info.**

Exploratory Data Analysis

Fetching Customer Data and Merging for Analysis

- In this task, to fetch the details of all the customers, we need to use three tables: **CRM_cust_info, ERP_LOC_101 and ERP_CUST_AZ12**.
- To perform efficient JOIN operations, a connection must be established between these tables.
- The connection will be established by extracting a substring from **CRM_prd_info.prd_key** and matching it with the entire column **ERP_px_cat_g1v2.id.**
- Following the same JOIN condition, the country corresponding to each customer was also fetched from the **, ERP_LOC_101 table** as the third join.

Step 1: The first and last names were concatenated and trimmed to ensure each customer's full name is displayed without any extra spaces.

Step 2: While joining the tables, gender was assigned to customers whose gender details were available in either of the tables. Those whose gender was not mentioned were labeled as "Not Mentioned.

Step 3: The marital status was standardized and labeled as "Married" and "Single" for better clarity.

Exploring and Pre-Processing the Sales Details

Step 1: The date format for all columns was changed.

Step 2: The quantity in each order was calculated by dividing the sales column by the price column.

Milestones Achieved

- We successfully executed all the necessary steps for loading and preprocessing the data. The three tables within the *silver_layer* schema now contain product, sales, and customer data while maintaining optimal data integrity.
- However, there is currently no built-in technique or helper column available to establish relationships between these datasets, which limits the ability to perform a cohesive analysis.
- From here onwards, the clean data can be extracted by other people in the organization for further analysis.