Protocol – Gold Layer Build Database

1. Gold Layer: Goals & Theory

Context: The Gold Layer transforms technical data (Silver) into business-ready data optimized for reporting and analytics.

1.1 Process Overview

1. **Analyze:** Identify business objects (Entities eg. Products,Customers) hidden in source tables.
2. **Integrate:** Combine data from multiple sources into single entities.
3. **Validate:** Ensure connectability and data quality.
4. **Document:** Create data models and data dictionaries.

1.2 Data Modeling Concepts

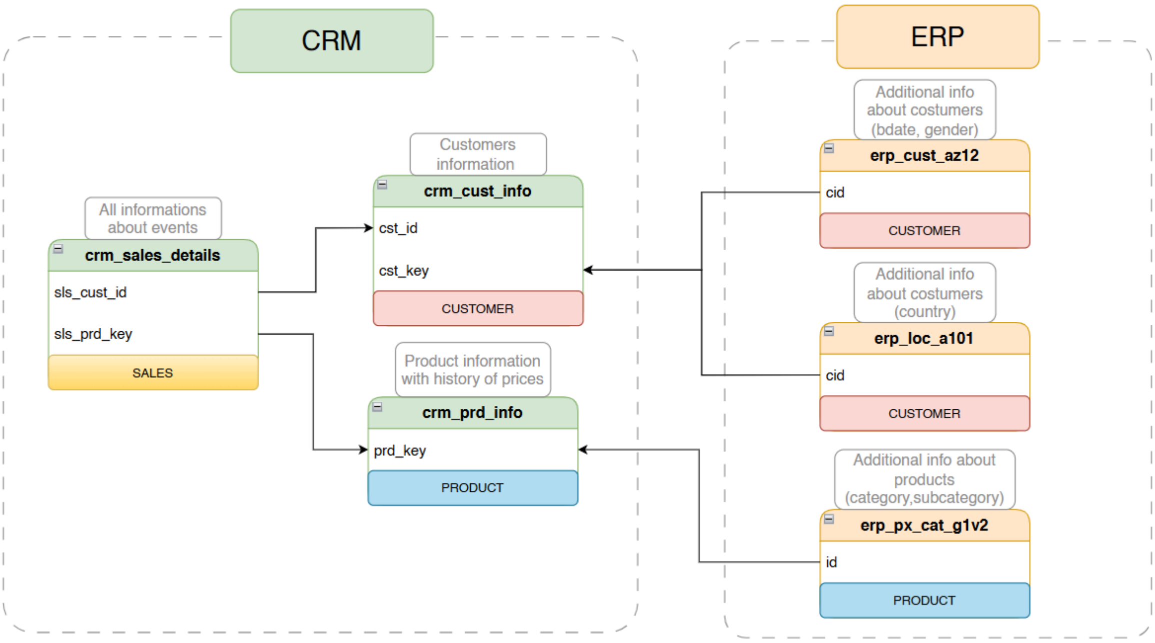
- **Star Schema:** The chosen model for this project.
  - **Center:** Fact Table (Transactions, measures, foreign keys).
  - **Points:** Dimension Tables (Descriptive attributes).
  - *Why?* Simpler for BI tools (like Power BI) and easier for users to query than Snowflake schema.
- **Dimensions (Who, What, Where):** Descriptive data.
  - Example: Customer Name, Product Category, Country.
- **Facts (How much, How many):** Transactions/Events.
  - Contains: Foreign Keys (to dimensions), Dates, and Measures (numbers).

---

2. Identify Business Objects (Analysis)

Action: Before coding, map source tables to business concepts using a diagramming tool (e.g., draw.io).

1. **Label Tables:** Review all Silver tables and tag them with a business entity. Example: Product, Sales, Customer
2. **Visual Grouping:** Color-code these groups to visualize the target model. This defines your "Logical Data Model."


CRM
ERP
Customers information
crm_cust_info
cst_id
cst_key
CUSTOMER
All informations about events
crm_sales_details
sls_cust_id
sls_prd_key
SALES
Product information with history of prices
crm_prd_info
prd_key
PRODUCT
Additional info about costumers (bdate, gender)
erp_cust_az12
cid
CUSTOMER
Additional info about costumers (country)
erp_loc_a101
cid
CUSTOMER
Additional info about products (category,subcategory)
erp_px_cat_g1v2
id
PRODUCT

---

3. Build `dim_customers` (Implementation)

Goal: Create a single View `gold.dim_customers` integrating data from CRM and ERP.

3.1 Select & Join Strategy

1. **Identify Master Source:**
   - CRM is chosen as the "Master" because it likely holds the most complete customer list.
2. **Join Logic:**
   - Start with `silver.crm_cust_info` (Master).
   - **LEFT JOIN** `silver.erp_cust_az12` (ERP Info) on Integration Keys.
   - **LEFT JOIN** `silver.erp_loc_a101` (ERP Location) on Integration Keys.
   - *Rule:* Avoid INNER JOINs to prevent data loss if secondary sources are incomplete.

3.2 Data Integration (Conflict Resolution)

Scenario: Both CRM and ERP have "Gender" information. Which one do we keep?

First ask owner of the data then create **Business Rule:**

CRM is Master. If CRM has data, use it. If CRM is null/n/a, fallback to ERP.

```sql
-- Derive gender with fallback logic:
-- 1. Use ci.cst_gndr if available and not 'n/a'
-- 2. Otherwise, fallback to ca.gen (or 'n/a' if null)
CASE
WHEN ci.cst_gndr IN ('n/a', NULL) THEN COALESCE(ca.gen,'n/a')
ELSE ci.cst_gndr
END AS gender
```

3.3 Standardization & Renaming

Rules:

- Gold Layer columns must be "business friendly" (no technical abbreviations).
- Reorder columns logically (e.g., group demographics together).

3.4 Surrogate Key Generation

Concept: The Data Warehouse should manage its own primary keys, independent of source systems. These are called **Surrogate Keys**.

Implementation: Use `ROW_NUMBER()` to generate a unique sequence.

```sql
ROW_NUMBER() OVER (ORDER BY ci.cst_id ASC) as customer_key,
```

_Note: Naming convention uses `_key` suffix for Surrogate Keys (e.g., `customer_key`) and `_id` or `_number` for business keys.

3.5 Create View Statement

Wrap the logic in a `CREATE VIEW` statement.

3.6 Quality Checks

Run specific checks immediately after creating the view:

1. **Check Integration Logic:**
2.

```sql
-- Check Integration Logic
-- Expectation: No nulls, only standardized values (Male, Female, n/a)
```

```
    SELECT DISTINCT gender FROM gold.dim_customers`
```

3. **Check Uniqueness:**

```
-- Check Uniqueness
-- Expectation: Empty result (Surrogate key must be unique)
SELECT
customer_key
FROM gold.dim_customers
GROUP BY customer_key
HAVING COUNT(*) > 1
```

4. Build `gold.fact_sales` view

1. By definition, this is a **Fact** table (answers “How much/how many?”).

4.1 Replace source keys with surrogate keys (lookups)

1. Sales table contains:
   - Product key (business key).
   - Customer ID (technical key).
2. Goal: **Use surrogate keys from dimensions** in the fact: `product_key`, `customer_key`.
3. Join tables
4. Drop original source keys from the Fact (use only surrogate keys to connect)

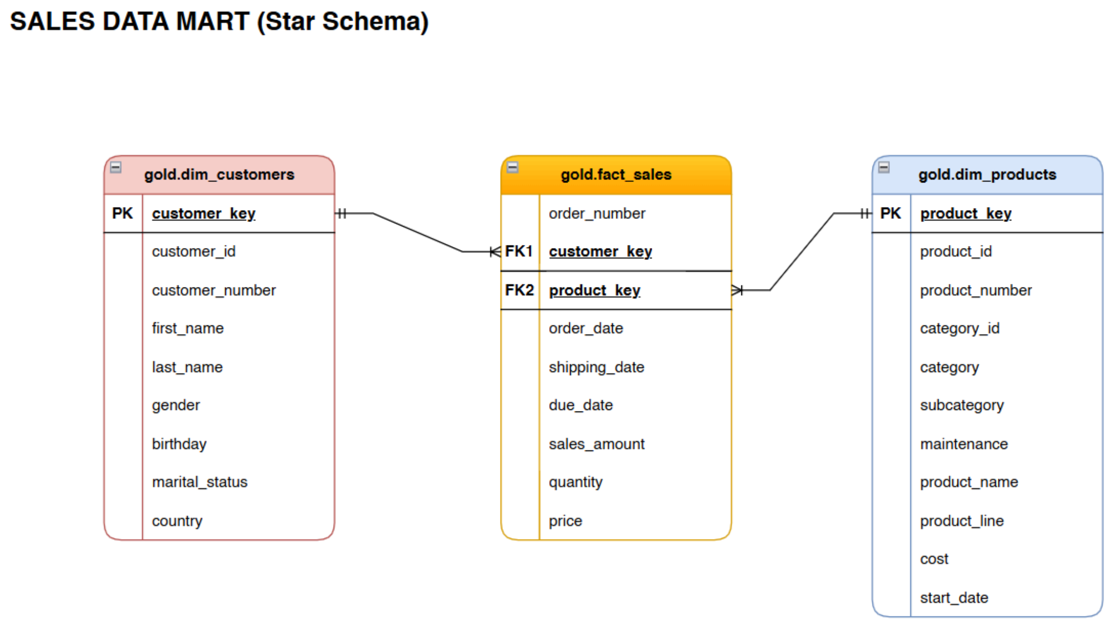
4.2 Connectivity quality checks

```
-- Check fact and customer dimension connectivity
-- Expect no rows (every fact has a valid customer)

SELECT
*
FROM gold.fact_sales fs
LEFT JOIN gold.dim_customers dc
  ON fs.customer_key = dc.customer_key
WHERE dc.customer_key IS NULL;

-- Check fact and product dimension connectivity
-- Expect no rows (every fact has a valid product)

SELECT *
```

```
FROM gold.fact_sales fs
LEFT JOIN gold.dim_products dp
  ON fs.product_key = dp.product_key
WHERE dp.product_key IS NULL;
```

5.

> #### Document- Draw Data Model of Star Schema (Draw io)
>
> ### Draw the Star Schema Model
>
> ### 1. Create the diagram
>
> In your diagram tool:
>
> 1. Add three tables:
>   - `gold.dim_customers` (Dimension, with PK = `customer_key`).
>   - `gold.dim_products` (Dimension, with PK = `product_key`).
>   - `gold.fact_sales` (Fact, no PK field in drawing, but includes FKs).
> 2. For each dimension table, list all columns as in the view definitions.
>
> ### 2. Add relationships
>
> 1. Use “1-to-many” relationship notation:
>   - One `dim_customers` record ↔ many `fact_sales` records.
>   - One `dim_products` record ↔ many `fact_sales` records.
> 2. Draw:



3. Optionally add annotation for business rule:

6.

Document- Create Data Catalog

Data Catalog for Gold Layer

Overview

The Gold layer contains dimensional modeling tables optimized for analytics and reporting, following a star schema pattern with two dimension tables and one fact table.

Usage

Supports sales analytics, customer segmentation, product performance reporting, and cohort analysis across CRM/ERP sources

1. Table: `gold.dim_customers`

Description: Stores customer details with demographic (gender, birthdate) and geographic (country) data.

Column	Data Type	Description	Example
customer_key	INT	Surrogate key for the customer	101

Column	Data Type	Description	Example
customer_id	INT	Source CRM technical ID	4578
customer_number	NVARCHAR(50)	Business customer number	CUST_000123
first_name	NVARCHAR(50)	Customer first name	Anna
last_name	NVARCHAR(50)	Customer last name	Smith
gender	NVARCHAR(50)	Customer gender	Female
birthdate	DATE	Customer date of birth	1985-04-12
marital_status	NVARCHAR(50)	Customer marital status	Single
country	NVARCHAR(50)	Country of the customer	Germany
create_date	DATE	Customer creation date in CRM	2011-01-03

2. Table: `gold.dim_products`

Description: Stores the current list of products with their category structure, maintenance flag, pricing and descriptive attributes used for analysis and reporting.

Column	Data Type	Description	Example
product_key	INT	Surrogate key for the product	201
product_id	INT	Technical ID of the product for internal tracking	845
product_number	NVARCHAR(50)	Business product number used for categorization or inventory	PRD_000234
product_name	NVARCHAR(50)	Descriptive product name	Mountain-200 Bike
category_id	NVARCHAR(50)	Category ID	CAT_01
category	NVARCHAR(50)	Product category name	Bikes
subcategory	NVARCHAR(50)	Product subcategory name	Mountain Bikes
maintenance	NVARCHAR(50)	Indicates if product/category requires maintenance	Yes/No
cost	INT	Product cost	450.00
product_line	NVARCHAR(50)	Product line grouping	Mountain, Road
start_date	DATE	Date from which is product available for sale	2013-01-01

3. Table: `gold.fact_sales`

> **Description:** Transaction-level sales fact table that links customer and product dimensions using surrogate keys, with dates and numeric measures for reporting (e.g., revenue, volume).
>
> | Column | Data Type | Description | Example |
> |---|---|---|---|
> | order_number | NVARCHAR(50) | Business order identifier from CRM | SO-20131215-0012 |
> | product_key | INT | Surrogate key linking order to product dimension table | 201 |
> | customer_key | INT | Surrogate key linking order to customer dimension table | 101 |
> | order_date | DATE | Date when the order was placed | 2013-12-15 |
> | shipping_date | DATE | Date when the order was shipped | 2013-12-17 |
> | due_date | DATE | Date when the order payment was due | 2013-12-20 |
> | sales_amount | INT | Revenue for the line: `quantity * price` after cleansing | 900.00 |
> | quantity | INT | Number of units ordered | 2 |
> | price | INT | Unit price of the product | 450.00 |

6.1 Structure of the catalog

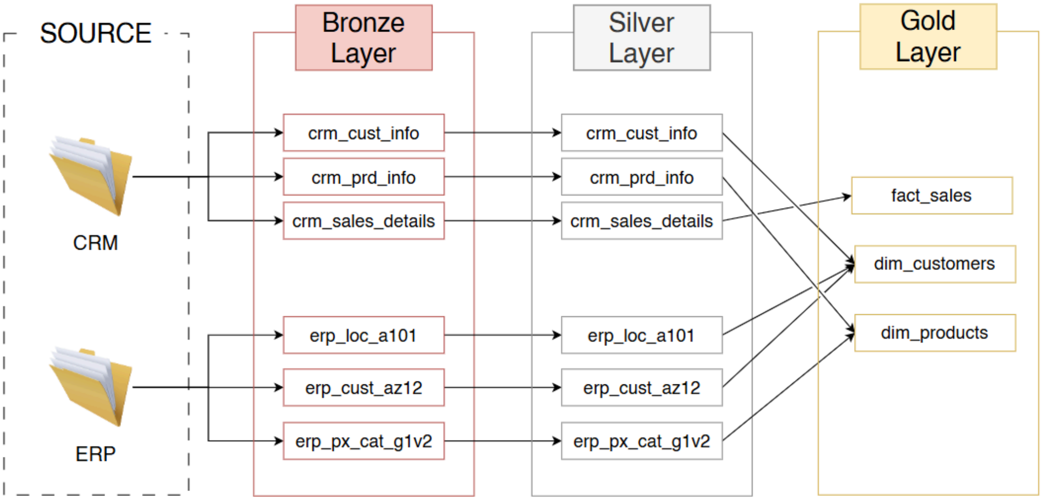
In your repo and in Obsidian:

1. File: `docs/data_catalog_gold.md` or similar.
2. For each Gold object (`dim_customers`, `dim_products`, `fact_sales`):
   - Table name.
   - Short description of purpose.
   - Column list with:
     - Column name.
     - Data type (optional but recommended).
     - Friendly description.
     - Example values.

7.

> ### Document- Extend Data Flow (Draw.io)

DATA FLOW CHART


SOURCE
CRM
ERP
Bronze Layer
crm_cust_info
crm_prd_info
crm_sales_details
erp_loc_a101
erp_cust_az12
erp_px_cat_g1v2
Silver Layer
crm_cust_info
crm_prd_info
crm_sales_details
erp_loc_a101
erp_cust_az12
erp_px_cat_g1v2
Gold Layer
fact_sales
dim_customers
dim_products