Sales & Customer Analytics Project Report

Detailed Table Creation, SQL Code & Data Integration Diagram

Data Integration Diagram (how tables are related):

Introduction

This document explains the table creation and ingestion steps used in the Bronze, Silver, and Gold layers of the project. Each table definition taken from your provided SQL scripts is reproduced as a formatted code block and followed by an explanation. The Data Integration diagram above shows relationships between CRM and ERP source tables and how they map to the project domains.

Bronze Layer - Raw Ingestion (DDL & BULK INSERT)

bronze.crm_cust_info - DDL

```
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
);
```

Explanation:

Purpose: Raw copy of CRM customer data as ingested from the CSV. Columns kept as-is for auditability.

BULK INSERT for bronze.crm_cust_info

```
BULK INSERT bronze.crm_cust_info
FROM 'C:\Users\mahes\OneDrive\Desktop\White Scholars\SQL\datasets\source_crm\cust_info.csv'
WITH (
  FIRSTROW = 2,
  FIELDTERMINATOR = ',',
  ROWTERMINATOR = '\n',
```

```
  TABLOCK
);
```

Explanation:

Loads the cust_info.csv into the bronze table. FIRSTROW=2 skips header.

bronze.crm_prd_info - DDL

```
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
);
```

Explanation:

Raw product master table storing original timestamps and codes.

BULK INSERT for bronze.crm_prd_info

```
BULK INSERT bronze.crm_prd_info
FROM 'C:\Users\mahes\OneDrive\Desktop\White Scholars\SQL\datasets\source_crm\prd_info.csv'
WITH (
  FIRSTROW = 2,
  FIELDTERMINATOR = ',',
  TABLOCK
);
```

Explanation:

Loads product CSV into bronze.prd_info.

bronze.crm_sales_details - DDL

```
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
);
```

Explanation:

Raw sales details; date fields stored as INT (YYYYMMDD) and cleaned later in Silver.

BULK INSERT for bronze.crm_sales_details

```
BULK INSERT bronze.crm_sales_details
FROM 'C:\Users\mahes\OneDrive\Desktop\White Scholars\SQL\datasets\source_crm\sales_details.csv'
WITH (
  FIRSTROW = 2,
  FIELDTERMINATOR = ',',
  TABLOCK
);
```

Explanation:

Loads sales details CSV.

ERP Bronze tables - DDL & BULK INSERT examples

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

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

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

Explanation:

ERP lookup tables loaded from LOC_A101.csv, CUST_AZ12.csv, PX_CAT_G1V2.csv respectively.

Silver Layer - Cleaned Tables & Transformations

silver.crm_cust_info - DDL

```
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
);
```

Explanation:

Cleaned customer table: deduplication, TRIM, normalized marital status and gender.

INSERT into silver.crm_cust_info (dedupe logic)

```
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, TRIM(cst_firstname), TRIM(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,
     CASE WHEN UPPER(TRIM(cst_gndr)) = 'F' THEN 'Female' WHEN UPPER(TRIM(cst_gndr)) = 'M' THEN
```

```
'Male' ELSE 'n/a' END,
    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;
```

Explanation:

Deduplicates by cst_id, keeps latest record, normalizes text fields.

silver.crm_prd_info - DDL

```
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()
);
```

Explanation:

Canonical product catalog with derived cat_id, humanized product line, and version end dates.

INSERT into silver.crm_prd_info (transformations)

```
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 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,
    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 bronze.crm_prd_info;
```

Explanation:

Derives cat_id, normalizes prd_key, ensures cost defaults, computes prd_end_dt using LEAD().

silver.crm_sales_details - DDL

```
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()
);
```

Explanation:

Sales table with proper DATE types and lineage timestamp.

INSERT into silver.crm_sales_details (transformations)

```
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(8)) 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(8)) 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(8)) 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,
      CASE WHEN sls_price IS NULL OR sls_price <= 0 THEN sls_sales / NULLIF(sls_quantity,0) ELSE sls_price END AS sls_price,
    sls_quantity
FROM bronze.crm_sales_details;
```

Explanation:

Validates and converts packed integer dates, recalculates sales when inconsistent, computes missing price.

silver.erp_cust_az12 - transformations

```
INSERT INTO silver.erp_cust_az12 (cid, bdate, gen)
SELECT
    CASE WHEN cid LIKE 'NAS%' THEN SUBSTRING(cid,4,LEN(cid)) ELSE cid END AS cid,
    CASE WHEN bdate > GETDATE() THEN NULL ELSE bdate END AS bdate,
      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
FROM bronze.erp_cust_az12;
```

Explanation:

Normalizes cid, nullifies future birthdates, normalizes gender.

silver.erp_loc_a101 - transformations

```
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
FROM bronze.erp_loc_a101;
```

Explanation:

Normalizes customer IDs and maps country codes to full names.

Gold Layer - Views, Fact & Dim Modeling and Analytics

gold.dim_customers - VIEW (DDL)

```
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,
    CASE WHEN ci.cst_gndr != 'n/a' THEN ci.cst_gndr ELSE COALESCE(ca.gen,'n/a') END AS gender,
    ci.cst_create_date AS create_date,
    ci.cst_marital_status AS marital_status,
    ca.bdate AS birthdate,
    la.cntry AS country
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;
```

Explanation:

Creates a unified customer dimension view with surrogate key and ERP fallbacks.

gold.dim_products - VIEW (DDL)

```
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 AS 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 pn.prd_end_dt IS NULL;
```

Explanation:

Product dimension view exposing active SKUs and category labels.

gold.fact_sales - VIEW (DDL)

```
CREATE VIEW gold.fact_sales AS
SELECT
    sd.sls_ord_num AS order_number,
    pr.product_key AS product_key,
    cu.customer_key AS 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;
```

Explanation:

Sales fact view that maps cleaned sales to product and customer surrogate keys.

Performance Analysis - Query (example)

```
WITH yearly_product_sales AS (
    SELECT YEAR(f.order_date) AS order_year, p.product_name, SUM(f.sales_amount) AS current_sales
    FROM gold.fact_sales f
    LEFT JOIN gold.dim_products p ON f.product_key = p.product_key
    WHERE f.order_date IS NOT NULL
    GROUP BY YEAR(f.order_date), p.product_name
)
SELECT order_year, product_name, current_sales,
       AVG(current_sales) OVER (PARTITION BY product_name) AS avg_sales,
       current_sales - AVG(current_sales) OVER (PARTITION BY product_name) AS diff_avg,
       LAG(current_sales) OVER (PARTITION BY product_name ORDER BY order_year) AS py_sales,
        current_sales - LAG(current_sales) OVER (PARTITION BY product_name ORDER BY order_year) AS
diff_py
FROM yearly_product_sales;
```

Explanation:

Yearly performance analysis with average and YoY comparisons.

gold.report_products - CREATE VIEW snippet

```
IF OBJECT_ID('gold.report_products','V') IS NOT NULL DROP VIEW gold.report_products;
GO
CREATE VIEW gold.report_products AS
WITH base_query AS (
        SELECT  f.order_number,  f.order_date,  f.customer_key,  f.sales_amount,  f.quantity,  p.product_key,
p.product_name, p.category, p.subcategory, p.cost
    FROM gold.fact_sales f
    LEFT JOIN gold.dim_products p ON f.product_key = p.product_key
    WHERE f.order_date IS NOT NULL
), product_aggregations AS (
    SELECT product_key, product_name, category, subcategory, cost,
           DATEDIFF(MONTH, MIN(order_date), MAX(order_date)) AS lifespan,
           MAX(order_date) AS last_sale_date,
           COUNT(DISTINCT order_number) AS total_orders,
```

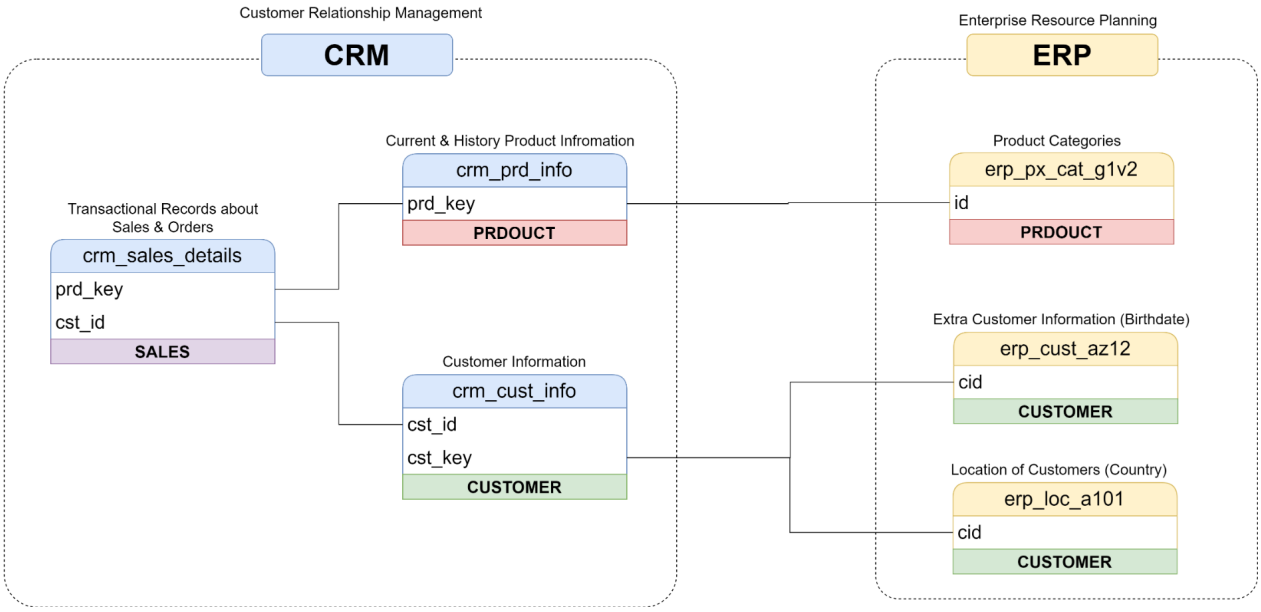
```
        COUNT(DISTINCT customer_key) AS total_customers,
        SUM(sales_amount) AS total_sales,
        SUM(quantity) AS total_quantity,
        ROUND(AVG(CAST(sales_amount AS FLOAT) / NULLIF(quantity,0)),1) AS avg_selling_price
    FROM base_query
    GROUP BY product_key, product_name, category, subcategory, cost
)
SELECT product_key, product_name, category, subcategory, cost, last_sale_date,
      DATEDIFF(MONTH, last_sale_date, GETDATE()) AS recency_in_months,
            CASE WHEN total_sales > 50000 THEN 'High-Performer' WHEN total_sales >= 10000 THEN
'Mid-Range' ELSE 'Low-Performer' END AS product_segment,
      lifespan, total_orders, total_sales, total_quantity, total_customers, avg_selling_price
FROM product_aggregations;
```

Explanation:

Aggregated product report view used by BI dashboards.

Table of Contents (Manual)

Data Integration (how to tables are related)



Project Presentation Explanation

1. Project Overview

This project demonstrates a complete end-to-end data engineering pipeline using the Medallion Architecture (Bronze → Silver → Gold) to build a validated data warehouse and generate business insights. We integrated CRM and ERP data sources, cleaned and standardized them, built dimensional models, and generated analytical reporting across products, customers, sales, and KPIs.

2. Medallion Architecture Overview

Bronze Layer – Raw ingestion of CRM/ERP files.
Silver Layer – Cleaning, standardization, validation, enrichment.
Gold Layer – Final dimensional modeling (DimCustomers, DimProducts, FactSales).

3. Data Integration

The CRM data provided customer info, product details, and sales transactions. The ERP system supplied customer birthdates, gender corrections, location details, and product category mapping. These were merged to form unified dimensions.

4. Bronze Layer

All raw data was ingested using BULK INSERT into bronze tables, preserving original structure for traceability.

5. Silver Layer Transformations

Customer cleanup (gender normalization, marital status mapping, removing duplicates). Product cleanup (category extraction, product line normalization, lifecycle date corrections). Sales cleanup (date corrections, fixing sales/price mismatches).

6. Gold Layer Modeling

DimCustomers and DimProducts were created with surrogate keys. FactSales was built with clean revenue, quantity, and order details. These serve as the analytical foundation.

7. Analytical Queries

Performance Analysis – Yearly product sales, historical average comparison, YoY analysis.
Part-to-Whole – Category-level contribution to total revenue.
Product Segmentation – Grouping SKUs by cost tiers.
Customer Segmentation – Classifying customers into VIP/Regular/New.
Customer Report – Complete profile including recency, frequency, and monetary KPIs.
Product Report – Lifecycle performance, average selling price, customer penetration.

8. Key Highlights

Fully end-to-end pipeline, proper medallion architecture, clean SQL transformations, strong dimensional modeling, advanced analytics, and business insights.

9. Business Value Delivered

Improved data quality, unified data sources, enhanced analytics, informed decision-making for pricing, retention, and promotions.

10. Final Takeaway

This project demonstrates strong data engineering, SQL, modeling, and analytical skills suitable for real-world data pipelines.