2026-01-22 22:52

Status: #done

Tags: project

sql warehouse project

Table of Contents

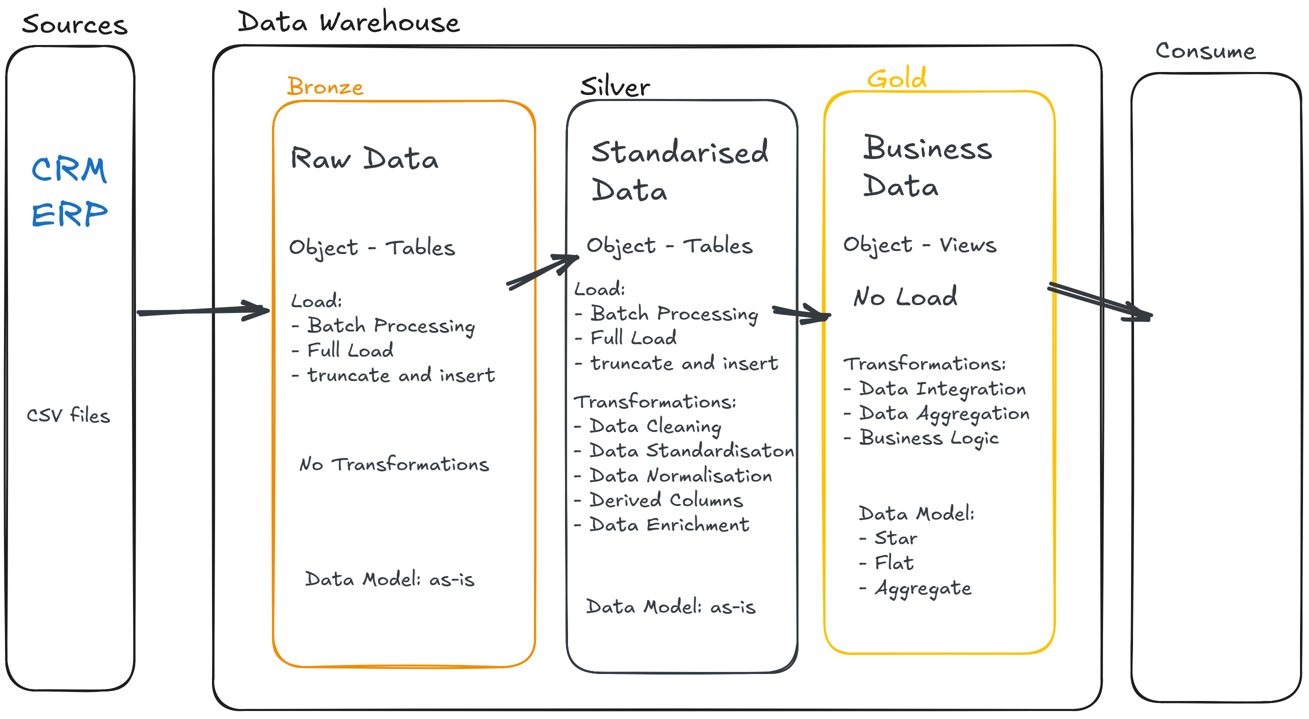
Datasets

- Datasets
  - CRM (Customer Relationship Management)

        - cust_info,csv
        - prd_info.csv
        - sales_details.csv
    - ERP (Enterprise Resource Planning)
        - cust_az12.csv
        - loc_a101.csv
        - px_cat_g1v2.csv

Architecture

- Medallion Architecture
- 3 Layers
    - **Bronze**
    - **Silver**
    - **Gold**
- Each Layer uses Loads and data and Transforms it as per its needs



Initial Databases and Schema

- create Database called **warehouse**
- Create 5 tables - one for each CSV
- 3 schemas - Bronze, Silver and Gold.

```
use master;
GO
```

```sql
IF EXISTS (SELECT 1
FROM sys.databases
WHERE name = 'warehouse')
BEGIN
    drop DATABASE warehouse;
END
GO

create DATABASE warehouse;
GO

use warehouse;
GO

IF NOT EXISTS (SELECT 1
FROM sys.schemas
WHERE name = 'bronze')
BEGIN
    EXEC('CREATE SCHEMA bronze;')
END
GO

IF NOT EXISTS (SELECT 1
FROM sys.schemas
WHERE name = 'silver')
BEGIN
    EXEC('CREATE SCHEMA silver;')
END
GO

IF NOT EXISTS (SELECT 1
FROM sys.schemas
WHERE name = 'gold')
BEGIN
    EXEC('CREATE SCHEMA gold;')
END
GO
```

why EXEC('create schema x;')?

schema must be the first command in the batch of commands.
That how SQL Server is setup but since we using a `IF` at the beginning `CREATE`

`SCHEMA` cannot be first in the batch.
`EXEC` creates a child batch where it the first in the batch.

Bronze Layer

- No Insights, Just load the data.
- Data types can be modified in Silver Layer, use **Generic Datatypes**.
- **No Transformations** and **No Data model** or more like No data modelling.
- Naming Convention for tables: **layer.source_tablename**

DDL script

- `bronze.crm_cust_info`

```
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_gender NVARCHAR(50),
     cst_create_date DATE
)
```

- `bronze.crm_prd_info`

```
IF OBJECT_ID('bronze.crm_prd_info','u') is NOT NULL DROP table
bronze.crm_prd_info;

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 DATE,
    prd_end_dt DATE
)
```

- `bronze.crm_sales_details`
    - `*_dt` are not formatted for `DATE`, so `INT` for now.

```
IF OBJECT_ID('bronze.crm_sales_details','u') is NOT NULL DROP table
bronze.crm_sales_details;

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
);
```

- `bronze.erp_loc_a101`

```
IF OBJECT_ID('bronze.erp_loc_a101','u') is NOT NULL DROP table
bronze.erp_loc_a101;

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

- `bronze.erp_cust_az12`

```
IF OBJECT_ID('bronze.erp_cust_az12','u') is NOT NULL DROP table
bronze.erp_cust_az12;

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

- `bronze.erp_px_cat_g1v2`

```
IF OBJECT_ID('bronze.erp_px_cat_g1v2','u') is NOT NULL DROP table
bronze.erp_px_cat_g1v2;

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

making scripts re runabble

`IF OBJECT_ID('table','u') is NOT NULL DROP table;`

- if the `table` is found then `DROP` the table
- This ensures we can recreate and run the script as much as possible.

`GO`

We can write all the scipts in a giant script with `GO` separator as it batches the commands to be run.
This is done in the project itself.

Load The Layer

Podman Dataset access

- Since SQL Server is ran on Podman, we need to give the pod access to datasets folder
- This is done thorough `-v` option in `podman` CLI

```
podman run -e "ACCEPT_EULA=Y" \
            -e "MSSQL_SA_PASSWORD=YourStrongPassword123!" \
            -p 1433:1433 \
            --name sqlserver \
            -v /home/kzcodes/code/datasets:/var/opt/mssql/datasets:ro \
            -d mcr.microsoft.com/mssql/server:2022-latest
```

  - `-e` Environment Variables
  - `-p` port mapping.
  - `--name` give the container a name, by default a random name is generated
  - `-v` Volume Mount
  - links a local folder to internal folder
  - The datasets are now present in `/var/opt/mssql/datasets` folder inside the pod.
  - `ro` Read Only
- `d` Detached Mode - It is ran in background keeping the terminal free.

Loading Procedure Script

- Stored Procedure is written
- the first row of CSV is headers
  - `FIRSTROW = 2`
- Define the Delimiter
  - `FIELDTERMINATOR = ','`
- `TABLOCK`
  - Performance optimization
  - Locks the table instead of each row

```
CREATE OR ALTER PROCEDURE bronze.load_bronze
AS
BEGIN
    DECLARE @start_time DATETIME, @end_time DATETIME, @batch_start_time
DATETIME;
    BEGIN TRY
        SET @batch_start_time = GETDATE();
        PRINT 'Loading Bronze Layer ... ';
        PRINT '─────────────────────────────';
        PRINT 'Loading crm/cust_info....';
        TRUNCATE TABLE bronze.crm_cust_info;
        PRINT '....';
        SET @start_time = GETDATE()
        BULK INSERT bronze.crm_cust_info
        FROM '/var/opt/mssql/datasets/source_crm/cust_info.csv'
        WITH (
            firstrow = 2,
            FIELDTERMINATOR = ',',
```

```sql
                tablock
            );
            SET @end_time = GETDATE()
            PRINT 'DONE in ' + cast(DATEDIFF(second, @start_time, @end_time)
as nVARCHAR)+'s';
            PRINT '';

            PRINT 'Loading crm/prd_info ... ';
            TRUNCATE TABLE bronze.crm_prd_info;
            PRINT '....';
            SET @start_time = GETDATE();
            BULK INSERT bronze.crm_prd_info
            FROM '/var/opt/mssql/datasets/source_crm/prd_info.csv'
            WITH (
                firstrow = 2,
                FIELDTERMINATOR = ',',
                tablock
            );
            SET @end_time = GETDATE();
            PRINT 'DONE in ' + cast(DATEDIFF(second, @start_time, @end_time)
as nVARCHAR)+'s';
            PRINT '';

            PRINT 'Loading crm/sales_details ... ';
            TRUNCATE TABLE bronze.crm_sales_details;
            PRINT '....';
            SET @start_time = GETDATE();
            BULK INSERT bronze.crm_sales_details
            FROM '/var/opt/mssql/datasets/source_crm/sales_details.csv'
            WITH (
                firstrow = 2,
                FIELDTERMINATOR = ',',
                tablock
            );
            SET @end_time = GETDATE();
            PRINT 'DONE in ' + cast(DATEDIFF(second, @start_time, @end_time)
as nVARCHAR)+'s';
            PRINT '';

            PRINT 'Loading erp/cust_az12 ... ';
            TRUNCATE TABLE bronze.erp_cust_az12;
            PRINT '....';
            SET @start_time = GETDATE();
```

```
          BULK INSERT bronze.erp_cust_az12
          FROM '/var/opt/mssql/datasets/source_erp/cust_az12.csv'
          WITH (
               firstrow = 2,
               FIELDTERMINATOR = ',',
               tablock
          );
          SET @end_time = GETDATE();
          PRINT 'DONE in ' + cast(DATEDIFF(second, @start_time, @end_time)
as nVARCHAR)+'s';
          PRINT '';

          PRINT 'Loading erp/loc_a101 ... ';
          TRUNCATE TABLE bronze.erp_loc_a101;
          PRINT '....';
          SET @start_time = GETDATE();
          BULK INSERT bronze.erp_loc_a101
          FROM '/var/opt/mssql/datasets/source_erp/loc_a101.csv'
          WITH (
               firstrow = 2,
               FIELDTERMINATOR = ',',
               tablock
          );
          SET @end_time = GETDATE();
          PRINT 'DONE in ' + cast(DATEDIFF(second, @start_time, @end_time)
as nVARCHAR)+'s';
          PRINT '';

          PRINT 'Loading erp/px_cat_g1v2 ... ';
          TRUNCATE TABLE bronze.erp_px_cat_g1v2;
          PRINT '....';
          SET @start_time = GETDATE();
          BULK INSERT bronze.erp_px_cat_g1v2
          FROM '/var/opt/mssql/datasets/source_erp/PX_CAT_G1V2.csv'
          WITH (
               firstrow = 2,
               FIELDTERMINATOR = ',',
               tablock
          );
          SET @end_time = GETDATE();
          PRINT 'DONE in ' + cast(DATEDIFF(second, @start_time, @end_time)
as NVARCHAR)+'s';
```

```
        PRINT '─────────────────────────────';
        PRINT 'Bronze Layer loaded in ' + CAST(DATEDIFF(second,
@batch_start_time, GETDATE()) as NVARCHAR) + 's'
        PRINT '─────────────────────────────';
    END TRY
    BEGIN CATCH
        PRINT '══════════════════════';
        PRINT 'Error: BRONZE layer';
        PRINT 'error msg: ' + ERROR_NUMBER();
        PRINT 'error no: ' + CAST(error_number() as NVARCHAR)
        PRINT 'error state: ' + CAST(error_state() as NVARCHAR)
        PRINT '══════════════════════';
    END CATCH
END
```

Silver Layer

- Takes input from Bronze Layer
- **Load** the data from Bronze
- **Data Transformation:**
    - Data Cleaning
    - Data Standardisation
    - Data Normalisation
    - Derived Columns
    - Data Enrichment

Analysis and Transformation of Data

- Analysis drive the Transformations of data
- Every table we add `dwh_create_date DATETIME2 DEFAULT GETDATE()` column
    - `dwh_` means it is created by data engineer column
    - **Metadata** for D.Es

`crm_cust_info`

- `cust_id` have multiple records because it uses a SCD Type 2
    - We can get the most recent records only using a subquery

```
    SELECT
            *,
            ROW_NUMBER() OVER(
                    PARTITION BY cst_id
                    ORDER BY
                         cst_create_date DESC
               ) rank
          FROM
               bronze.crm_cust_info
      WHERE rank = 1
```

- this returns The latest created record

- String Data ⇒ Use `TRIM` to remove accidental leading and /or trailing spaces
- No category inconsistency detected.
- **Categorical Mapping** for `cst_material_status` and `cst_gender`

  -
    ```
    CASE
                WHEN LOWER(TRIM(cst_material_status)) = 's' THEN
    'Single'
                WHEN LOWER(TRIM(cst_material_status)) = 'm' THEN
    'Married'
                ELSE 'n/a'
            END AS cst_material_status,
        CASE
                WHEN LOWER(TRIM(cst_gender)) = 'f' THEN 'Female'
                WHEN lower(TRIM(cst_gender)) = 'm' THEN 'Male'
                ELSE 'n/a'
            END AS cst_gender,
    ```

`crm_prd_info`

- multiple `prd_key` but `prd_id` is unique ⇒ Same Product but different other attributes e.g.
  - It is a SCD Type 2
- `prd_key`
  - First 5 chars of `prd_key` is same as `erp_px_cat_g1v2.id`
    - create a new column `cat_id`
  - The rest of characters are same as `crm_sales_details.sls_product_key`
    - this is the new `prd_key`

```
      REPLACE(SUBSTRING(prd_key, 1, 5), '-', '_') AS cat_id,
       SUBSTRING(prd_key, 7, LEN(prd_key)) AS prd_key
```

- **Invalid Dates**
  - `prd_start_dt` > `prd_end_dt`
  - **Sol:** Use next record `prd_start_dt` -1 as `prd_end_dt`
  - LEAD function is perfect for it

-
  ```
  prd_start_dt,
      DATEADD(day,-1,
      LEAD(prd_start_dt) OVER(PARTITION BY prd_key ORDER BY
  prd_start_dt)
  ) AS prd_end_dt
  ```

- `prd_cost` can be NULL ⇒ use COALESCE
- `prd_line` is Abbreviated

```
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,
```

`crm_sales_details`

- `*_dt` columns have invalid date
    - **Sol:** Convert the valid dates and place `NULL` if invalid

  ```
  WHEN *_dt = 0 OR len(*_dt) ≠ 8 THEN NULL
        ELSE CAST(CAST(*_dt AS varchar) AS date)
  ```

- `sales` has NULLS and invalid columns as per `quantity` and `price` columns
- `price` has NULLS and invalid prices such as Negative prices
- **Sol:** this is usually to be discussed with a team BUT
    - If `price` is negative, turn it positive
    - if `sales` is Invalid as per `quantity * price`, change it to `quantity * price`
    - If `price` is invalid then `sales / quantity`
        - Make sure that `quantity is not 0` ⇒ use NULLIF

```
     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,
        sls_quantity,
    CASE
        WHEN sls_price IS NULL
```

```
          OR sls_price ≤ 0
        THEN sls_sales / NULLIF(sls_quantity, 0)
        ELSE sls_price
    END AS sls_price
```

`erp_cust_az12`

- `cid` has inconsistent prefix `NAS`
    - **sol:** remove them to be consistent with `crm_cust_info.cst_id`

```
CASE
      WHEN cid LIKE 'NAS%' THEN SUBSTRING(cid, 4, LEN(cid))
      ELSE cid
  END AS cid
```

- If `bdate > GETDATE()` set it to NULL

```
CASE
    WHEN bdate > GETDATE() THEN NULL
    ELSE bdate
END AS bdate,
```

- `gen` has a Categorical Inconsistency
    - **Sol:** Map each Categorical data into Buckets and then use the buckets

```
CASE
      WHEN UPPER(TRIM(gen)) LIKE 'F%' THEN 'Female'
      WHEN UPPER(TRIM(gen)) LIKE 'M%' THEN 'Male'
      ELSE 'n/a'
  END AS gen
```

`erp_loc_a101`

- String columns use TRIM to remove trailing and leading spaces
- `cid` has an '-' inside the string
    - **Sol:** remove it using REPLACE

```
    trim(REPLACE(cid, '-', '')) AS cid,
```

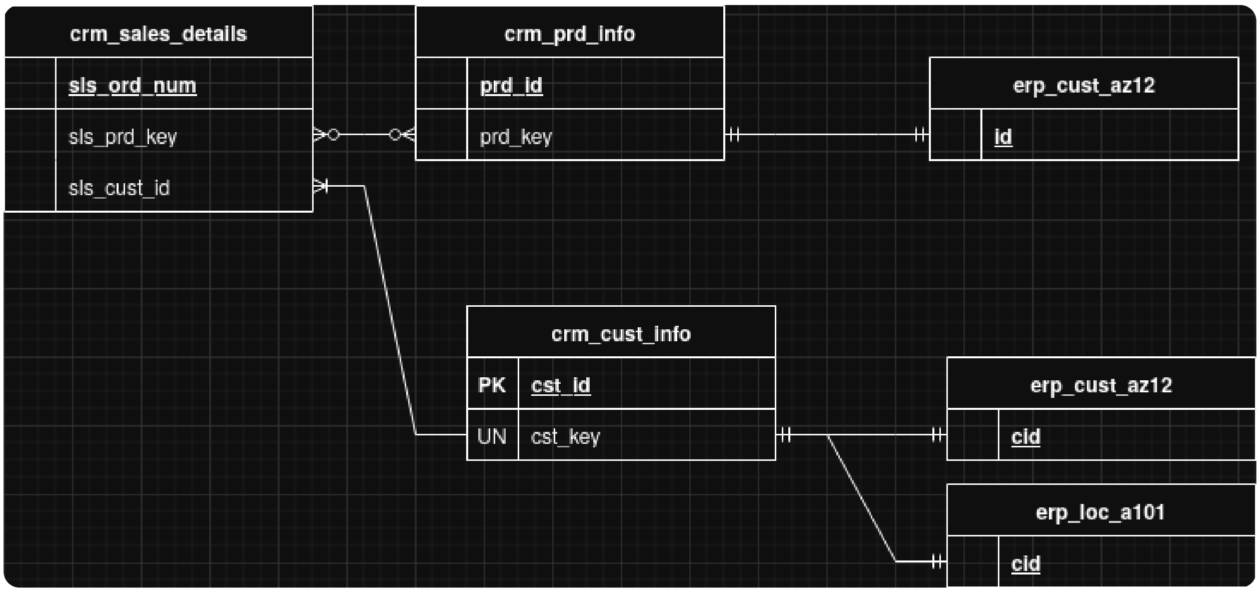
- `cntry` column values end with `\r` which the `TRIM` cannot remove, use REPLACE

`erp_px_cat_g1v2`

- **PERFECT!**

Relations

- Given the analysis the Relation graph looks like:



DDL script

- Using the analysis, **proper DATATYPE** are given to columns
- **New Columns** are derived for more data integrity
- Each table now has `dwh_create_date` column that tells D.E. when was the data load onto silver layer

```
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_gender NVARCHAR(50),
    cst_create_date DATE,
    dwh_create_date DATETIME2 DEFAULT GETDATE()
)

GO
IF OBJECT_ID('silver.crm_prd_info','u') is NOT NULL DROP table
silver.crm_prd_info;

CREATE TABLE silver.crm_prd_info
```

```
(
    prd_id INT,
    prd_key NVARCHAR(50),
    cat_id 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()
)

GO
IF OBJECT_ID('silver.crm_sales_details','u') is NOT NULL DROP table
silver.crm_sales_details;

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()

);
GO
IF OBJECT_ID('silver.erp_loc_a101','u') is NOT NULL DROP table
silver.erp_loc_a101;

CREATE TABLE silver.erp_loc_a101
(
    cid NVARCHAR(50),
    cntry NVARCHAR(50),
    dwh_create_date DATETIME2 DEFAULT GETDATE()

);

GO
```

```
IF OBJECT_ID('silver.erp_cust_az12','u') is NOT NULL DROP table
silver.erp_cust_az12;

CREATE TABLE silver.erp_cust_az12
(
    cid NVARCHAR(50),
    bdate DATE,
    gen NVARCHAR(50),
    dwh_create_date DATETIME2 DEFAULT GETDATE()
);
GO

IF OBJECT_ID('silver.erp_px_cat_g1v2','u') is NOT NULL DROP table
silver.erp_px_cat_g1v2;

CREATE TABLE silver.erp_px_cat_g1v2
(
    id NVARCHAR(50),
    cat NVARCHAR(50),
    subcat NVARCHAR(50),
    maintenance NVARCHAR(50),
    dwh_create_date DATETIME2 DEFAULT GETDATE()

);
```

Load The Layer

- **Data is Cleansed** before loading
- **SCD** column are flattened ⇒ **Data Normalisation**
- **Inconsistent** Data is made consistent ⇒ **Data Standarisation**

```
CREATE
OR ALTER PROCEDURE silver.load_silver
AS
BEGIN
    BEGIN TRY
        DECLARE @start_time DATETIME, @end_time DATETIME
        PRINT 'Truncating: silver.crm_cust_info'
        TRUNCATE TABLE silver.crm_cust_info
        PRINT 'Inserting data into silver.crm_cust_info'
        SET @start_time = GETDATE()
        INSERT INTO silver.crm_cust_info
        (
        cst_id,
        cst_key,
        cst_firstname,
        cst_lastname,
        cst_material_status,
        cst_gender,
        cst_create_date
        )
    SELECT
        cst_id,
        cst_key,
        TRIM(cst_firstname) AS cst_firstname,
        TRIM(cst_lastname) AS cst_lastname,
        CASE
                WHEN LOWER(TRIM(cst_material_status)) = 's' THEN 'Single'
                WHEN LOWER(TRIM(cst_material_status)) = 'm' THEN 'Married'
                ELSE 'n/a'
            END AS cst_material_status,
        CASE
                WHEN LOWER(TRIM(cst_gender)) = 'f' THEN 'Female'
                WHEN lower(TRIM(cst_gender)) = 'm' THEN 'Male'
                ELSE 'n/a'
            END AS cst_gender,
        cst_create_date
    FROM
        (
        SELECT
            *,
            ROW_NUMBER() OVER(
                PARTITION BY cst_id
                ORDER BY
                    cst_create_date DESC
            ) rank
        FROM
            bronze.crm_cust_info
    ) t
    WHERE rank = 1
    SET @end_time = GETDATE()
    PRINT 'Inserted in ' + CAST(DATEDIFF(second, @start_time, @end_time) AS
NVARCHAR) + 's'

    PRINT 'Truncating: silver.crm_prd_info'
    TRUNCATE TABLE silver.crm_prd_info
```

```sql
    PRINT 'Inserting data into silver.crm_prd_info'
    SET @start_time = GETDATE()
    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,
        prd_start_dt,
        DATEADD(
        day,
        -1,
        LEAD(prd_start_dt) OVER(
            PARTITION BY prd_key
            ORDER BY
                prd_start_dt
        )
    ) AS prd_end_dt
    FROM
        bronze.crm_prd_info
    SET @end_time = GETDATE()
    PRINT 'Inserted in ' + CAST(DATEDIFF(second, @start_time, @end_time) AS
NVARCHAR) + 's'
```

```
    PRINT 'Truncating: silver.crm_sales_details'
    TRUNCATE TABLE silver.crm_sales_details
    PRINT 'Inserting data into silver.crm_sales_details'
    SET @start_time = GETDATE()
    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) 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) 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) 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,
```

```sql
        sls_quantity,
        CASE
        WHEN sls_price IS NULL
            OR sls_price ≤ 0 THEN sls_sales / NULLIF(sls_quantity, 0)
        ELSE sls_price
    END AS sls_price
    FROM
        bronze.crm_sales_details
    SET @end_time = GETDATE()
    PRINT 'Inserted in ' + CAST(DATEDIFF(second, @start_time, @end_time) AS
NVARCHAR) + 's'

    PRINT 'Truncating: silver.erp_cust_az12'
    TRUNCATE TABLE silver.erp_cust_az12
    PRINT 'Inserting data into silver.erp_cust_az12'
    SET @start_time = GETDATE()
    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)) LIKE 'F%' THEN 'Female'
        WHEN UPPER(TRIM(gen)) LIKE 'M%' THEN 'Male'
        ELSE 'n/a'
    END AS gen
    FROM
        bronze.erp_cust_az12
    SET @end_time = GETDATE()
    PRINT 'Inserted in ' + CAST(DATEDIFF(second, @start_time, @end_time) AS
NVARCHAR) + 's'

    PRINT 'Truncating: silver.erp_loc_a101'
    TRUNCATE TABLE silver.erp_loc_a101
    PRINT 'Inserting data into silver.erp_loc_a101'
    SET @start_time = GETDATE()
    INSERT INTO
    silver.erp_loc_a101
        (cid, cntry)
    SELECT
        trim(REPLACE(cid, '-', '')) AS cid,
        CASE
        WHEN UPPER(REPLACE(TRIM(cntry), CHAR(13), '')) IN ('USA', 'US',
'UNITED STATES') THEN 'USA'
        WHEN UPPER(REPLACE(TRIM(cntry), CHAR(13), '')) IN('UK', 'UNITED
KINGDOM') THEN 'United Kingdom'
        WHEN UPPER(REPLACE(TRIM(cntry), CHAR(13), '')) = 'DE' THEN 'Germany'
        WHEN REPLACE(TRIM(cntry), CHAR(13), '') = '' THEN 'n/a'
        ELSE REPLACE(TRIM(cntry), CHAR(13), '')
    END AS cntry
    FROM
        bronze.erp_loc_a101;

    SET @end_time = GETDATE()
    PRINT 'Inserted in ' + CAST(DATEDIFF(second, @start_time, @end_time) AS
NVARCHAR) + 's' PRINT 'Truncating: silver.erp_px_cat_g1v2'
    TRUNCATE TABLE silver.erp_px_cat_g1v2
    PRINT 'Inserting data into silver.erp_px_cat_g1v2'
    SET @start_time = GETDATE()
    INSERT INTO
    silver.erp_px_cat_g1v2
        (id, cat, subcat, maintenance)
    SELECT
        id,
        cat,
        subcat,
        maintenance
    FROM
        bronze.erp_px_cat_g1v2;

    SET @end_time = GETDATE()
    PRINT 'Inserted in ' + CAST(DATEDIFF(second, @start_time, @end_time) AS
NVARCHAR) + 's'
END TRY
BEGIN CATCH

END CATCH
END
```

Gold Layer

- **Data Modelling**
- Friendlier Table names
- **Identify Business Objects**
- Star Schema vs. Snowflake Schema

Explore Business Objects

- `crm_prd_info` and `erp_px_cat_g1v2` ⇒ **Product**
- `crm_cust_info`, `erp_loc_a101` and `erp_cust_az12` ⇒ **Customer**
- `crm_sales_details` ⇒ **Sales**

Data Integration

Building Customer Table

- First we need to join tables

```
SELECT
    cci.cst_id,
    cci.cst_key,
    cci.cst_firstname,
    cci.cst_lastname,
    cci.cst_material_status,
    cci.cst_gender,
    cci.cst_create_date,
    eca.bdate,
    eca.gen,
    ela.cntry
FROM silver.crm_cust_info cci
    LEFT JOIN silver.erp_cust_az12 eca ON cci.cst_key = eca.cid
    LEFT JOIN silver.erp_loc_a101 ela ON cci.cst_key = ela.cid
```

- Duplicate check let the above query be `q`
  - `SELECT cst_id, count(*) from q group by cst_id having count(*) > 1`
  - Returns nothing, 👍
- `cci.cst_gender` and `eca.gen` are two sources of same gender
  - `eca.gen` can be NULL
  - `cci,cst_gender` is **Priority** as CRM are customer authority usually

```
CASE
     WHEN cci.cst_gender ≠ 'n/a' THEN cci.cst_gender
     ELSE COALESCE(eca.gen, 'n/a')
 END as gender
```

- Using a surrogate key ⇒ as primary key
- The DDL command

```
CREATE VIEW gold.dim_customers
AS
    (SELECT
        ROW_NUMBER() over(order by cci.cst_create_date) as customer_key,
        cci.cst_id as customer_id,
        cci.cst_key as customer_number,
        cci.cst_firstname as first_name,
        cci.cst_lastname as last_name,
        cci.cst_material_status AS material_status,
        cci.cst_create_date as create_date,
        eca.bdate as birth_date,
        ela.cntry as country,
        CASE
        WHEN cci.cst_gender ≠ 'n/a' THEN cci.cst_gender
        ELSE COALESCE(eca.gen, 'n/a')
    END as gender
    FROM silver.crm_cust_info cci
        LEFT JOIN silver.erp_cust_az12 eca ON cci.cst_key = eca.cid
        LEFT JOIN silver.erp_loc_a101 ela ON cci.cst_key = ela.cid
)
```

Building Product Table

- Lets only get the latest data ⇒ `silver.crm_prd_info.prd_end_dt` IS NULL
- Join the tables

```
SELECT
    cpi.prd_id,
    cpi.cat_id,
    cpi.prd_key,
    cpi.prd_nm,
    cpi.prd_cost,
    cpi.prd_line,
    cpi.prd_start_dt
FROM silver.crm_prd_info cpi
```

```sql
    LEFT JOIN silver.erp_px_cat_g1v2 epc ON epc.id = cpi.cat_id
WHERE cpi.prd_end_dt IS NULL
```

- Check Duplicate rows
  - `SELECT prd_id, count(*) from q group by prd_id having count(*) > 1`
  - Returns Nothing. 👍
- The DDL Command

```sql
CREATE VIEW gold.dim_products
AS
    (SELECT
        ROW_NUMBER() over(order by cpi.prd_start_dt, cpi.prd_key) as
product_key,
        cpi.prd_id as product_id,
        cpi.prd_key as product_number,
        cpi.prd_nm as product_name,
        cpi.prd_line as product_line,
        cpi.cat_id as category_id,
        epc.cat as category,
        epc.subcat as sub_category,
        epc.maintenance as maintenance,
        cpi.prd_cost as cost,
        cpi.prd_start_dt as start_date
    FROM silver.crm_prd_info cpi
        LEFT JOIN silver.erp_px_cat_g1v2 epc ON epc.id = cpi.cat_id
    WHERE cpi.prd_end_dt IS NULL
)
```

Building Sales Table

- Select everything
- Join Everything but replace the columns with Primary Keys of the Tables joined
- DDL

```sql
CREATE VIEW gold.fact_sales
AS
    (SELECT
        sd.sls_ord_num as order_number,
        -- sls_prd_key,
        pr.product_key,
        -- sls_cust_id,
        cust.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,
        sd.sls_quantity as quantity,
        sd.sls_price as price
    from silver.crm_sales_details sd
        LEFT JOIN gold.dim_products pr ON pr.product_number = sd.sls_prd_key
        LEFT JOIN gold.dim_customers cust on cust.customer_id =
sd.sls_cust_id )
```