

Data Enrichment

Add new, relevant data to enhance the dataset for analysis

Build Silver Layer

Clean & Load

crm_sales_details

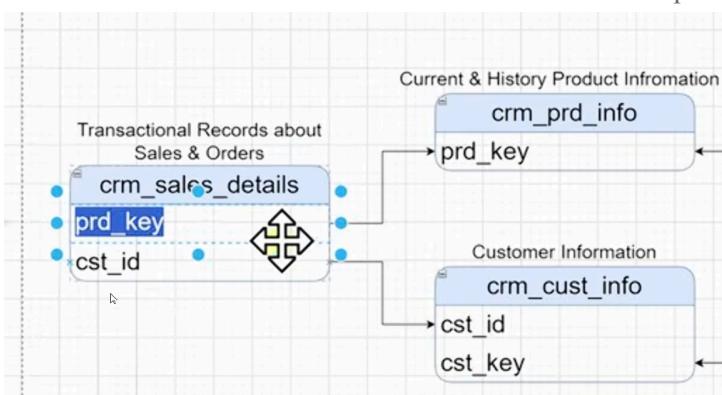
```
1  SELECT
2  [sls_ord_num],
3  [sls_prd_key],
4  [sls_cust_id],
5  [sls_order_dt],
6  [sls_ship_dt],
7  [sls_due_dt],
8  [sls_sales],
9  [sls_quantity],
10 [sls_price]
11 FROM [bronze].[crm_sales_details]
12 WHERE [sls_ord_num] != TRIM([sls_ord_num])
13
```

104 % No se encontraron problemas.

Resultados Mensajes

sls_ord_num	sls_prd_key	sls_cust_id	sls_order_dt	sls_ship_dt	sls_due_dt	sls_sales	sls_quantity	sls_price
-------------	-------------	-------------	--------------	-------------	------------	-----------	--------------	-----------

With this formula we can see that we have no unwanted spaces.



These are the connections we're going to make. That's why we need to have everything perfect.

We use the product key to connect to product information.

We're connecting the customer ID to the customer ID in customer info.

```
1  SELECT
2  [sls_ord_num],
3  [sls_prd_key],
4  [sls_cust_id],
5  [sls_order_dt],
6  [sls_ship_dt],
7  [sls_due_dt],
8  [sls_sales],
9  [sls_quantity],
10 [sls_price]
11 FROM [bronze].[crm_sales_details]
12 WHERE [sls_prd_key] NOT IN (SELECT [sls_prd_key] FROM [silver].[crm_prd_info])
13
```

104 % No se encontraron problemas.

Resultados Mensajes

sls_ord_num	sls_prd_key	sls_cust_id	sls_order_dt	sls_ship_dt	sls_due_dt	sls_sales	sls_quantity	sls_price
-------------	-------------	-------------	--------------	-------------	------------	-----------	--------------	-----------

there is no problem this means that we can use the pk all the prod key from the sales details can be used and connected with the product info.

```

1   SELECT
2     [sls_ord_num],
3     [sls_prd_key],
4     [sls_cust_id],
5     [sls_order_dt],
6     [sls_ship_dt],
7     [sls_due_dt],
8     [sls_sales],
9     [sls_quantity],
10    [sls_price]
11   FROM [bronze].[crm_sales_details]
12  WHERE [sls_cust_id] NOT IN (SELECT cst_id FROM [silver].[crm_cust_info])
13

```

No se encontraron problemas.

Resultados Mensajes

sls_ord_num	sls_prd_key	sls_cust_id	sls_order_dt	sls_ship_dt	sls_due_dt	sls_sales	sls_quantity	sls_price

There's no problem, so we can connect to both tables without any problems. No transformations are required.

sls_order_dt	sls_ship_dt	sls_due_dt
20101229	20110105	20110110
20101229	20110105	20110110
20101229	20110105	20110110

We continue with these columns. They're not dates, they're INTEGER (INT), they're numbers, and we like them that way. We need to change them from INT to DATE.

Negative numbers or zeros can't be cast to a date

```

SQLQuery2 c...\\David (75)) + X | SQLQuery1 pa...\\David (51))
1   -- Check for invalid dates
2
3   SELECT
4     [sls_order_dt]
5   FROM [bronze].[crm_sales_details]
6   WHERE [sls_order_dt] <= 0

```

No se encontraron problemas.

Resultados Mensajes

sls_order_dt
0
0
0
0
0
0
0
0
0

This is a problem because we have dates that say 0.

We can replace this information with null-.

NULLIF ()

Returns NULL if two given values are equal; otherwise, it returns the first expression.

SQLQuery2 c...\David (75) | SQLQuery1 pa...\David (51)

```

1 -- Check for invalid date
2
3 SELECT
4     NULLIF([sls_order_dt],0 ) [sls_order_dt]
5     FROM [bronze].[crm_sales_details]
6     WHERE [sls_order_dt] <= 0

```

104 % No se encontraron problemas.

	sls_order_dt
1	NULL
2	NULL
3	NULL
4	NULL
5	NULL
6	NULL
7	NULL

In this scenario, the length of the date must be 8

sls_order_dt

20101229
20101229
20101229
20101229
20101230
20101230
20101230
20101230

1 2 3 4 5 6 7 8

20101229

Year Month Day

SQLQuery2 c...\David (75) | SQLQuery1 pa...\David (51)

```

1 -- Check for invalid date
2
3 SELECT
4     NULLIF([sls_order_dt],0 ) [sls_order_dt]
5     FROM [bronze].[crm_sales_details]
6     WHERE [sls_order_dt] <= 0 OR LEN([sls_order_dt]) != 8

```

104 % No se encontraron problemas.

	sls_order_dt
1	NULL
2	NULL
3	NULL
4	NULL
5	NULL
6	NULL
7	NULL
8	NULL
9	NULL
10	NULL
11	NULL
12	NULL
13	NULL
14	NULL
15	NULL
16	NULL
17	NULL
18	32154
19	5489

SQLQuery2 c...\David (75) | SQLQuery1 pa...\David (51)

```

1 -- Check for invalid date
2
3 SELECT
4     NULLIF([sls_order_dt],0 ) [sls_order_dt]
5     FROM [bronze].[crm_sales_details]
6     WHERE [sls_order_dt] <= 0
7         OR LEN([sls_order_dt]) != 8
8         OR [sls_order_dt] > 20500101
9         OR [sls_order_dt] < 19000101

```

104 % No se encontraron problemas.

	sls_order_dt
1	NULL
2	NULL
3	NULL
4	NULL
5	NULL
6	NULL
7	NULL
8	NULL
9	NULL
10	NULL
11	NULL
12	NULL
13	NULL
14	NULL
15	NULL
16	NULL
17	NULL
18	32154
19	5489

With this filter, we can see that SLAS order dts have fewer than 8 characters, so they're incorrect.

With this, we check that none are outside your business limits.

In SQL Server, we can't cast from an int to a date. You first have to convert it to a varchar, and from a varchar to a date.

```

SQLQuery2 ch...N\David (75) | SQLQuery1 p...David (51)* ✎ X
1  SELECT
2    [sls_ord_num],
3    [sls_prd_key],
4    [sls_cust_id],
5    [sls_order_dt],
6    CASE WHEN [sls_order_dt] = 0 or
7      LEN ([sls_order_dt]) != 8 THEN NULL
8      ELSE CAST (CAST ([sls_order_dt] AS varchar) AS DATE)
9      END AS [sls_order_dt],
10   [sls_ship_dt],
11   [sls_due_dt],
12   [sls_sales],
13   [sls_quantity],
14   [sls_price]
15   FROM [bronze].[crm_sales_details]
16

```

104 % ▾ No se encontraron problemas.

Resultados Mensajes

	sls_ord_num	sls_prd_key	sls_cust_id	sls_order_dt	sls_order_dt	sls_ship_dt	sls_due_dt	sls_sales
1	SO43697	BK-R93R-62	21768	20101229	2010-12-29	20110105	20110110	3578
2	SO43698	BK-M82S-44	28389	20101229	2010-12-29	20110105	20110110	3400
3	SO43699	BK-M82S-44	25863	20101229	2010-12-29	20110105	20110110	3400
4	SO43700	BK-R50B-62	14501	20101229	2010-12-29	20110105	20110110	699

This is how we transform an INT date to a DATE. We eliminate the zeros and, if they are not 8 characters, we say NULL. We transform from INT to VARCHAR and from VARCHAR to DATE. Now we do the same for the ship date.

```

SQLQuery2 ch...N\David (75) ✎ X SQLQuery1 pa...David (51)*
1  -- Check for invalid dates
2  SELECT
3    NULLIF([sls_ship_dt], 0 ) [sls_ship_dt]
4    FROM [bronze].[crm_sales_details]
5    WHERE [sls_ship_dt] <= 0
6      OR LEN([sls_ship_dt]) != 8
7      OR [sls_ship_dt] > 20500101
8      OR [sls_ship_dt] < 19000101
9
10
11
12
13
14
15
16
17
18

```

104 % ▾ 4 4 0 ↑ ↓

Resultados Mensajes

	sls_ord_num	sls_prd_key	sls_cust_id	sls_order_dt	sls_order_dt	sls_ship_dt	sls_due_dt	sls_sales	sls_quantity	sls_price
1	SO43697	BK-R93R-62	21768	2010-12-29	2011-01-05	20110110	3578	1	3578	
2	SO43698	BK-M82S-44	28389	2010-12-29	2011-01-05	20110110	3400	1	3400	
3	SO43699	BK-M82S-44	25863	2010-12-29	2011-01-05	20110110	3400	1	3400	

The ship date is perfect, so we don't have to make any transformations. While the ship date is fine, we changed the original formula in case the same problems arise in the future.

```

SQLQuery2 ch...N\David (75)* ✎ X SQLQuery1 pa...David (51)*
1  -- Check for invalid dates
2  SELECT
3    NULLIF([sls_due_dt], 0 ) [sls_due_dt]
4    FROM [bronze].[crm_sales_details]
5    WHERE [sls_due_dt] <= 0
6      OR LEN([sls_due_dt]) != 8
7      OR [sls_due_dt] > 20500101
8      OR [sls_due_dt] < 19000101
9
10
11
12
13
14
15
16
17
18
19
20

```

104 % ▾ 4 4 0 ↑ ↓

Resultados Mensajes

	sls_ord_num	sls_prd_key	sls_cust_id	sls_order_dt	sls_order_dt	sls_ship_dt	sls_due_dt	sls_sales	sls_quantity	sls_price
1	SO43697	BK-R93R-62	21768	2010-12-29	2011-01-05	20110110	3578	1	3578	
2	SO43698	BK-M82S-44	28389	2010-12-29	2011-01-05	20110110	3400	1	3400	

DATE)
Order Date must always be earlier
than the Shipping Date or Due Date

The due date is fine. We do the same for the DUE DATE. The order date must always be smaller than the ship date.

```

8      OR [sls_due_dt] < 19000101
9
10     -- Check for invalid Date Orders
11     SELECT
12     *
13     FROM [bronze].[crm_sales_details]
14     WHERE [sls_order_dt] > [sls_ship_dt]
15     OR [sls_order_dt] > [sls_due_dt]

```

!% No se encontraron problemas.

t	sls_sales	sls_quantity	sls_price
0	3578	1	3578
0	3400	1	3400
0	3400	1	3400
0	000	1	000

This is fine because we don't have any orders. We continue with the last three columns.

Business Rules

$$\sum \text{Sales} = \text{Quantity} * \text{Price}$$

✗ Negative, Zeros, Nulls are Not Allowed!



We apply a filter to see all those sales that do not meet this function.

```

3     -- Values must not be NULL, Zero or Negative.
4
5     SELECT
6     [sls_sales],
7     [sls_quantity],
8     [sls_price]
9     FROM [bronze].[crm_sales_details]
10    WHERE [sls_sales] != [sls_quantity]*[sls_price]

```

104 % No se encontraron problemas.

	sls_sales	sls_quantity	sls_price
1	769	1	-769
2	30	1	-30
3	22	1	-22
4	35	2	35
5	1701	1	-1701

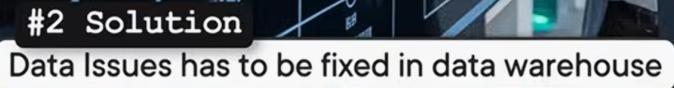
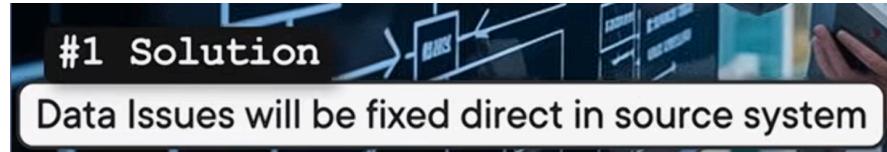

```

5     SELECT DISTINCT
6     [sls_sales],
7     [sls_quantity],
8     [sls_price]
9     FROM [bronze].[crm_sales_details]
10    WHERE [sls_sales] != [sls_quantity]*[sls_price]
11    OR [sls_sales] IS NULL OR [sls_quantity] IS NULL OR [sls_price] IS NULL
12    OR [sls_sales] <= 0 OR [sls_quantity] <= 0 OR [sls_price] <= 0
13    ORDER BY [sls_sales], [sls_quantity], [sls_price]

```

104 % No se encontraron problemas.

	sls_sales	sls_quantity	sls_price
1	NULL	1	2
2	NULL	1	8



Rules

If Sales is negative, zero, or null, derive it using Quantity and Price.

If Price is zero or null, calculate it using Sales and Quantity.

If Price is negative, convert it to a positive value

```
ORDER BY sls_sales, sls_quantity, sls_price
```

ABS()

Returns absolute value of a number

```
1  SELECT
2      [sls_ord_num],
3      [sls_prd_key],
4      [sls_cust_id],
5          CASE WHEN [sls_order_dt] = 0 or
6              LEN ([sls_order_dt]) != 8 THEN NULL
7              ELSE CAST (CAST ([sls_order_dt] AS varchar) AS DATE)
8          END AS [sls_order_dt],
9          CASE WHEN [sls_ship_dt] = 0 or
10             LEN ([sls_ship_dt]) != 8 THEN NULL
11             ELSE CAST (CAST ([sls_ship_dt] AS varchar) AS DATE)
12         END AS [sls_ship_dt],
13         CASE WHEN [sls_due_dt] = 0 or
14             LEN ([sls_due_dt]) != 8 THEN NULL
15             ELSE CAST (CAST ([sls_due_dt] AS varchar) AS DATE)
16         END AS [sls_due_dt],
17         CASE WHEN [sls_sales] IS NULL OR [sls_sales] <=0 OR [sls_sales] != [sls_quantity]* ABS([sls_price])
18             THEN [sls_quantity]* ABS([sls_price])
19             ELSE [sls_sales]
20         END AS [sls_sales],
21         [sls_quantity],
22         CASE WHEN [sls_price] IS NULL OR [sls_price] <=0
23             THEN [sls_sales]/ NULLIF ([sls_quantity], 0)
24             ELSE [sls_price]
25         END AS [sls_price]
26     FROM [bronze].[crm_sales_details]
```

104 % No se encontraron problemas. Linea: 1

	sls_ord_num	sls_prd_key	sls_cust_id	sls_order_dt	sls_ship_dt	sls_due_dt	sls_sales	sls_quantity	sls_price
1	SO43697	BK-R93R-62	21768	2010-12-29	2011-01-05	2011-01-10	3578	1	3578
2	SO43698	BK-M82S-44	28389	2010-12-29	2011-01-05	2011-01-10	3400	1	3400
3	SO43699	BK-M82S-44	25863	2010-12-29	2011-01-05	2011-01-10	3400	1	3400
4	SO43700	BK-R50B-62	14501	2010-12-29	2011-01-05	2011-01-10	699	1	699

We have already cleaned the last 3 columns. Now we must do a check.

```

SQLQuery2 Q...\\David (64)) * SQLQuery2 Qu...\\David (51)* SQLQuery2 ch...\\David (53)*
1  IF OBJECT_ID ('[silver].[crm_sales_details]', 'U') IS NOT NULL
2    DROP TABLE [silver].[crm_sales_details]
3  CREATE TABLE [silver] .[crm_sales_details] (
4    [sls_ord_num] NVARCHAR(50),
5    [sls_prd_key] NVARCHAR(50),
6    [sls_cust_id] INT,
7    [sls_order_dt] DATE,
8    [sls_ship_dt] DATE,
9    [sls_due_dt] DATE,
10   [sls_sales] INT,
11   [sls_quantity] INT,
12   [sls_price] INT,
13   [dwh_create_date] DATETIME2 DEFAULT GETDATE()
14 );
15
104 % ✓ No se encontraron problemas.
Mensajes
Los comandos se han completado correctamente.
Hora de finalización: 2025-10-15T12:54:13.5890366-03:00

```

```

SQLQuery2 Q...\\David (51)* * SQLQuery2 ch...\\David (53)* SQLQuery2 Qu...\\David (64))
1  INSERT INTO [silver].[crm_sales_details] (
2    [sls_ord_num],
3    [sls_prd_key],
4    [sls_cust_id],
5    [sls_order_dt],
6    [sls_ship_dt],
7    [sls_due_dt],
8    [sls_sales],
9    [sls_quantity],
10   [sls_price]
11 )
12
13 SELECT
14   [sls_ord_num],
15   [sls_prd_key],
16   [sls_cust_id],
17   CASE WHEN [sls_order_dt] = 0 or
18     LEN ([sls_order_dt]) != 8 THEN NULL
19     ELSE CAST (CAST ([sls_order_dt] AS varchar) AS DATE)
20   END AS [sls_order_dt],
21   CASE WHEN [sls_ship_dt] = 0 or
22     LEN ([sls_ship_dt]) != 8 THEN NULL
23     ELSE CAST (CAST ([sls_ship_dt] AS varchar) AS DATE)
24   END AS [sls_ship_dt],
25   CASE WHEN [sls_due_dt] = 0 or
26     LEN ([sls_due_dt]) != 8 THEN NULL
27     ELSE CAST (CAST ([sls_due_dt] AS varchar) AS DATE)
28   END AS [sls_due_dt],
29   CASE WHEN [sls_sales] IS NULL OR [sls_sales] <=0 OR [sls_sales] !
30   END AS [sls_sales]
31
32 (60398 filas afectadas)
33
34 Hora de finalización: 2025-10-15T12:57:31.8297125-03:00

```

We create the table, now we insert it. Ready, we insert it into the main query.



```

9
10    -- Check for invalid Date Orders
11    SELECT *
12    FROM silver.[crm_sales_details]
13    WHERE sls_order_dt > [sls_ship_dt]
14        OR sls_order_dt > [sls_due_dt]
15
16
17    -- Check Data Consistency: Between Sales, Quantity, and Price
18    -- Sales = Quantity * Price
19        ...

```

104 % No se encontraron problemas.

Resultados Mensajes

sls_ord_num	sls_prd_key	sls_cust_id	sls_order_dt	sls_ship_dt	sls_due_dt	sls_sales	sls_
-------------	-------------	-------------	--------------	-------------	------------	-----------	------

```

16
17    -- Check Data Consistency: Between Sales, Quantity, and Price
18    -- Sales = Quantity * Price
19    -- Values must not be NULL, Zero or Negative.
20
21    SELECT DISTINCT
22        [sls_sales],
23        [sls_price],
24        [sls_quantity]
25    FROM silver.[crm_sales_details]
26    WHERE [sls_sales] != [sls_quantity]*[sls_price]
27        OR [sls_sales] IS NULL OR [sls_quantity] IS NULL OR [sls_price] IS NULL
28        OR [sls_sales] <= 0 OR [sls_quantity] <= 0 OR [sls_price] <= 0
29    ORDER BY [sls_sales], [sls_quantity], [sls_price]

```

104 % No se encontraron problemas.

Resultados Mensajes

sls_sales	sls_price	sls_quantity
-----------	-----------	--------------

```

30
31    --Final Check
32    SELECT * FROM [silver].[crm_sales_details]

```

104 % No se encontraron problemas.

Resultados Mensajes

	sls_ord_num	sls_prd_key	sls_cust_id	sls_order_dt	sls_ship_dt	sls_due_dt	sls_sales	sls_quantity	sls_price	dwh_create_date
1	SO43697	BK-R93R-62	21768	2010-12-29	2011-01-05	2011-01-10	3578	1	3578	2025-10-15 12:57:31.5500000
2	SO43698	BK-M82S-44	28389	2010-12-29	2011-01-05	2011-01-10	3400	1	3400	2025-10-15 12:57:31.5500000
3	SO43699	BK-M82S-44	25863	2010-12-29	2011-01-05	2011-01-10	3400	1	3400	2025-10-15 12:57:31.5500000
4	SO43700	BK-R50B-62	14501	2010-12-29	2011-01-05	2011-01-10	699	1	699	2025-10-15 12:57:31.5500000
5	SO43701	BK-M82S-44	11003	2010-12-29	2011-01-05	2011-01-10	3400	1	3400	2025-10-15 12:57:31.5500000

Everything is perfect!!!

Now in international format!!!

```

/* =====
   Script:      Load Sales Details - Bronze to Silver
   Author:      Ignacio David Stocco
   Date:       2025-10-15
   Purpose:     Cleans and standardizes sales data for the silver layer
   Notes:       Converts dates, recalculates sales, and validates prices
===== */

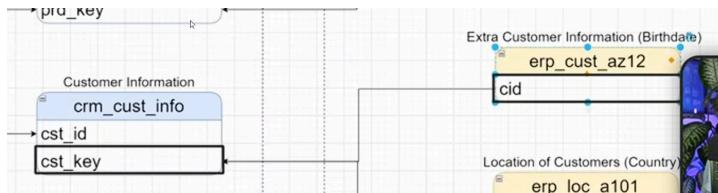
INSERT INTO [silver].[crm_sales_details] (
    [sls_ord_num],
    [sls_prd_key],
    [sls_cust_id],
    [sls_order_dt],
    [sls_ship_dt],
    [sls_due_dt],
    [sls_sales],
    [sls_quantity],
    [sls_price]
)
SELECT
    [sls_ord_num],
    [sls_prd_key],
    [sls_cust_id],
    CASE
        WHEN [sls_order_dt] = 0
            OR LEN([sls_order_dt]) != 8
        THEN NULL
        ELSE TRY_CAST(TRY_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 TRY_CAST(TRY_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 TRY_CAST(TRY_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], -- Recalculate sales if original value is missing or inconsistent
    [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] -- Derive price if original value is invalid
FROM [bronze].[crm_sales_details];

```

Build Silver Layer

Clean & Load

erp_cust_az12



SQLQuery6.s...David (51)*

```

1 SELECT
2   [cid],
3   [bdate],
4   [gen]
5   FROM [bronze].[erp_cust_az12]
  
```

104 % No se encontraron problemas.

Resultados

	cid	bdate	gen
1	NASAW00011000	1971-10-06	Male
2	NASAW00011001	1976-05-10	Male
3	NASAW00011002	1971-02-09	Male
4	NASAW00011003	1973-08-14	Female
5	NASAW00011004	1979-08-05	Female
6	NASAW00011005	1976-08-01	Male
7	NASAW00011006	1976-12-02	Female
8	NASAW00011007	1969-11-06	Male

SQLQuery6.s...David (81)*

```

1 SELECT
2   [cid],
3   [bdate],
4   [gen]
5   FROM [bronze].[erp_cust_az12]
6
7   SELECT *FROM [silver].[crm_cust_info]
  
```

104 % No se encontraron problemas.

Resultados

	cid	bdate	gen
1	NASAW00011000	1971-10-06	Male
2	NASAW00011001	1976-05-10	Male
3	NASAW00011002	1971-02-09	Male
4	NASAW00011003	1973-08-14	Female

	cst_id	cst_key	cst_firstname	cst_lastname	cst_marital_status
1	11000	AW00011000	Jon	Yang	Married
2	11001	AW00011001	Eugene	Huang	Single
3	11002	AW00011002	Ruben	Torres	Married
4	11003	AW00011003	Christy	Zhu	Single

We do a layer check. There are characters we don't like.

SQLQuery6.s...David (81)* X

```

1  SELECT
2      [cid],
3      CASE
4          WHEN cid LIKE 'NAS%'
5              THEN SUBSTRING (cid, 4, LEN(cid))
6          ELSE cid
7      END cid,
8      [bdate],
9      [gen]
10     FROM [bronze].[erp_cust_az12]

```

104 % ✓ No se encontraron problemas.

	cid	cid	bdate	gen	
1	NASAW00011000	AW00011000	1971-10-06	Male	
2	NASAW00011001	AW00011001	1976-05-10	Male	
3	NASAW00011002	AW00011002	1971-02-09	Male	
4	NASAW00011003	AW00011003	1973-08-14	Female	
5	NASAW00011004	AW00011004	1979-08-05	Female	

cid
NASAW00011000

We removed NAS at the beginning. If the cid is like 3 characters. If the Id start with those 3 characters then apply transformation otherwise stay as default cid. Transformation is Substring then we define the string is going to be the cid and then we have to define the position where we start cutting or extract. Then we need to define how many characters do we need to be extracted and we make it dynamic so we go with the LEN we have cid. If it is a NAS then extract from cid the position number 4 the rest of the characters.

We don't have any NAS at the start

Now we test if we can join it to the other silver table

The screenshot shows a SQL query window titled "SQLQuery6.s...David (81)*". The query itself is:

```
1  SELECT
2      [cid],
3      CASE
4          WHEN cid LIKE 'NAS%'
5              THEN SUBSTRING (cid, 4, LEN(cid))
6          ELSE cid
7      END cid,
8      [bdate],
9      [gen]
10     FROM [bronze].[erp_cust_az12]
11     WHERE CASE
12         WHEN cid LIKE 'NAS%'
13             THEN SUBSTRING (cid, 4, LEN(cid))
14         ELSE cid
15     END NOT IN (SELECT DISTINCT [cst_key] FROM [silver].[crm_cust_info])
```

Below the code, a status bar indicates "104 % No se encontraron problemas." (No errors found). The results tab is selected, showing a header row with columns: cid, cid, bdate, gen.

This way we check if there is any value that is not repeated in the silver table. We are unable to find any unmatching data between those 2 tables. This means our transformation is working perfectly.

Check for very old customers

Check for birthdays in the future

Now we check the second column, bdate. We check if we have an old date.

```

1  -- Identify out-of-range dates
2
3  SELECT DISTINCT
4    [bdate]
5  FROM [bronze].[erp_cust_az12]
6  WHERE bdate < '1924'

```

104 % ▾ × 1 ▲ 0 ↑ ↓

Resultados Mensajes

	bdate
1	1916-02-10
2	1917-02-09
3	1917-06-05
4	1917-09-20
5	1918-02-11
6	1918-11-08
7	1919-02-14
8	1919-03-10

```

3  -- SELECT DISTINCT
4    [bdate]
5  FROM [bronze].[erp_cust_az12]
6  WHERE bdate < '1924' OR [bdate] > GETDATE()

```

104 % ▾ ● No se encontraron problemas.

Resultados Mensajes

	bdate
4	1917-09-20
5	1918-02-11
6	1918-11-08
7	1919-02-14
8	1919-03-10
9	1920-11-14
10	1922-01-02
11	1922-04-10
12	1922-06-06
13	1923-04-14
14	1923-08-16
15	1923-11-16
16	2038-10-17
17	2042-02-22
18	2045-03-03
19	2050-05-21
20	2050-07-06
21	2050-09-07
22	2050-11-22

We have clients who are over 100 years old. We also have clients whose birth dates are older than the current date. This is unacceptable.

Poor data quality.

```

1  SELECT
2   CASE
3    WHEN cid LIKE 'NAS%' THEN SUBSTRING (cid, 4, LEN(cid))
4    ELSE cid
5   END cid,
6   CASE
7    WHEN [bdate] > GETDATE() THEN NULL
8    ELSE [bdate]
9   END AS [bdate],
10  [gen]
11 FROM [bronze].[erp_cust_az12]
12
13
14 -- Data Standardization & Consistency
15 SELECT DISTINCT gen
16 FROM [bronze].[erp_cust_az12]

```

Results:

gen
NULL
F
Male
Female
M

With this, we'll no longer have any clients with a date of birth in the future. We check all the possible data options. It doesn't look good. We have NULL F and an empty space. Let's clean up this information to have only three values.

```

7  -- Data Standardization & Consistency
8
9  SELECT DISTINCT
10  gen,
11  CASE
12   WHEN UPPER(TRIM ([gen])) IN ('F', 'FEMALE') THEN 'Female'
13   WHEN UPPER(TRIM ([gen])) IN ('M', 'MALE') THEN 'Male'
14   ELSE 'N/A'
15  END AS gen
16 FROM [bronze].[erp_cust_az12]

```

Results:

gen	gen
M	Male
NULL	N/A
F	Female
	N/A
Female	Female
Male	Male


```

2  SELECT
3   CASE
4    WHEN cid LIKE 'NAS%' THEN SUBSTRING (cid, 4, LEN(cid))
5    ELSE cid
6   END cid,
7   CASE
8    WHEN [bdate] > GETDATE() THEN NULL
9    ELSE [bdate]
10   END AS [bdate],
11   CASE
12    WHEN UPPER(TRIM ([gen])) IN ('F', 'FEMALE') THEN 'Female'
13    WHEN UPPER(TRIM ([gen])) IN ('M', 'MALE') THEN 'Male'
14    ELSE 'N/A'
15   END AS gen
16 FROM [bronze].[erp_cust_az12]

```

Results:

cid	bdate	gen
AW00011000	1971-10-06	Male
AW00011001	1976-05-10	Male
AW00011002	1971-02-09	Male
AW00011003	1973-08-14	Female
AW00011004	1979-08-05	Female
AW00011005	1976-08-01	Male

With this, we've covered all the scenarios. Everything's fine, so now let's insert it into the silver layer.

```

1  v INSERT INTO [silver].[erp_cust_az12] ([cid], [bdate], [gen])
2   SELECT
3     CASE
4       WHEN cid LIKE 'NAS%'
5           THEN SUBSTRING(cid, 4, LEN(cid))
6       ELSE cid
7     END cid,
8     CASE
9       WHEN [bdate] > GETDATE()
10      THEN NULL
11      ELSE [bdate]
12    END AS [bdate],
13    CASE
14      WHEN UPPER(TRIM([gen])) IN ('F', 'FEMALE') THEN 'Female'
15      WHEN UPPER(TRIM([gen])) IN ('M', 'MALE') THEN 'Male'
16      ELSE 'N/A'
17    END AS gen
18  FROM [bronze].[erp_cust_az12]

```

104 % No se encontraron problemas.

Mensajes

(18483 filas afectadas)

Hora de finalización: 2025-10-15T17:25:34.5227275-03:00



SQLQuery4 co...N\David (54) SQLQuery2 qu...\\David (8)

```

1  -- Identify Out-Of_range Dates
2
3  v SELECT DISTINCT
4    [bdate]
5  FROM silver.[erp_cust_az12]
6  WHERE bdate < '1924'
7    OR [bdate] > GETDATE();
8
```

104 % No se encontraron problemas.

Resultados Mensajes

	bdate
1	1916-02-10
2	1917-02-09
3	1917-06-05
4	1917-09-20
5	1918-02-11
6	1918-11-08
7	1919-02-14
8	1919-03-10
9	1920-11-14
10	1922-01-02
11	1922-04-10


```

9  -- Data Standardization & Consistency
10 v SELECT DISTINCT
11   gen,
12   CASE
13     WHEN UPPER(TRIM([gen])) IN ('F', 'FEMALE') THEN 'Female'
14     WHEN UPPER(TRIM([gen])) IN ('M', 'MALE') THEN 'Male'
15     ELSE 'N/A'
16   END AS gen
17  FROM silver.[erp_cust_az12];

```

104 % No se encontraron problemas.

Resultados Mensajes

gen	gen
N/A	N/A
Female	Female
Male	Male

We only send those old clients. We only change those birthdays that are in the future. It's okay, we only have two values.

Final look

```

18
19    ||-- Final Check
20    SELECT * FROM silver.[erp_cust_az12]

```

104 % ▾ No se encontraron problemas.

Resultados Mensajes

	cid	bdate	gen	dwh_create_date
1	AW00011000	1971-10-06	Male	2025-10-15 17:25:34.4466667
2	AW00011001	1976-05-10	Male	2025-10-15 17:25:34.4466667
3	AW00011002	1971-02-09	Male	2025-10-15 17:25:34.4466667
4	AW00011003	1973-08-14	Female	2025-10-15 17:25:34.4466667

SQL Full Course for Beginners (30 Hours) - From Zero to Hero

```

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

```

} we have handled invalid values.

END AS cid,

```

CASE
    WHEN bdate > GETDATE() THEN NULL
    ELSE bdate
    END AS bdate, -- Set future birthdates to NULL
CASE
    END AS bdate, -- Set future birthdates to NULL
CASE

```

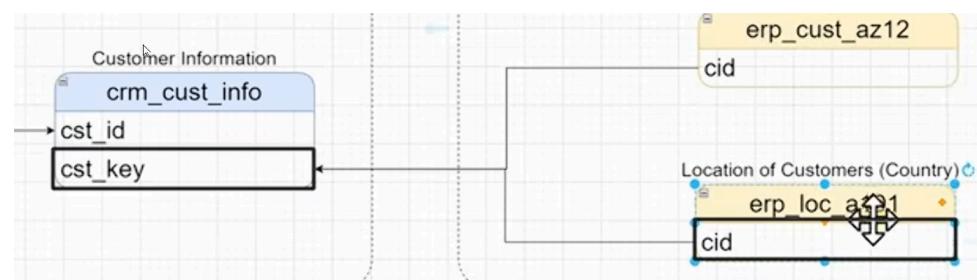
```

WHEN UPPER(TRIM(gen)) IN ('F', 'FEMALE') THEN 'Female'
WHEN UPPER(TRIM(gen)) IN ('M', 'MALE') THEN 'Male'
ELSE 'n/a'
END AS gen -- Normalize gender values and handle unknown cases

```

FROM bronze.erp_cust_az12

we have done data normalization.



```

SQLQuery9.s...David (51)*  SQLQuery9.s...David (51)*
1  SELECT [cid], [cntry]
2  FROM [bronze].[erp_loc_a101]
3
4
5
6  SELECT [cst_key]
7  FROM [silver].[crm_cust_info]
8

```

104 % No se encontraron problemas.

	cid	cntry
1	AW-00011000	Australia
2	AW-00011001	Australia
3	AW-00011002	Australia
4	AW-00011003	Australia
5	AW-00011004	Australia
6	AW-00011005	Australia

	cst_key
1	AW00011000
2	AW00011001
3	AW00011002
4	AW00011003
5	AW00011004

In El Cid there is a minus between the numbers and the characters. We must get rid of that minus.

```

SQLQuery9.s...David (51)*  SQLQuery10....David (65)* | SQLQuery9.s...David (51)*
1  SELECT [cid], [cntry]
2  FROM [bronze].[erp_loc_a101]
3
4
5  SELECT [cst_key]
6  FROM [silver].[crm_cust_info]
7
8
9
10

```

104 % No se encontraron problemas.

	cid	cntry
1	AW00011000	Australia
2	AW00011001	Australia
3	AW00011002	Australia
4	AW00011003	Australia

	cst_key
1	AW00011000
2	AW00011001
3	AW00011002

```

-- Comprobacion de unmatching data
1
2
3
4  SELECT [cid], [cntry]
5  FROM [bronze].[erp_loc_a101]
6  WHERE REPLACE ([cid], '-', '') NOT IN
7    (SELECT [cst_key]
8     FROM [silver].[crm_cust_info])
9
10

```

104 % No se encontraron problemas.

	cid	cntry
--	-----	-------

We check that all the values are equal to join the tables. Our transformation is working. Now we can connect those two tables.

```

11  ||-- Data Standardization & Consistency
12  |> SELECT DISTINCT [cntry]
13  |>   FROM [bronze].[erp_loc_a101]
14  |>   ORDER BY [cntry]
15
16
104 % No se encontraron problemas.

```

Resultados

	cntry
1	NULL
2	
3	Australia
4	Canada
5	DE
6	France
7	Germany
8	United Kingdom
9	United States
10	US
11	USA


```

SQLQuery9.s...David (51))" X SQLQuery10.s... (David (65))*
1 > SELECT
2     REPLACE ([cid], '-', '') cid,
3     CASE
4         WHEN TRIM([cntry]) = 'DE' THEN 'Germany'
5         WHEN TRIM([cntry]) IN ('US', 'USA') THEN 'United States'
6         WHEN TRIM([cntry]) = '' OR [cntry] IS NULL THEN 'N/A'
7         ELSE TRIM([cntry])
8     END AS [cntry]
9
10    FROM [bronze].[erp_loc_a101]
11
104 % No se encontraron problemas.

```

Resultados

	cid	cntry
1	AW00011000	Australia
2	AW00011001	Australia
3	AW00011002	Australia

We do a quick check and see that we have null and empty values. We also have observations.
We've already made all the transformations to country.

We didn't change anything in the DDL; both are Varchar.

Now we can insert them into our table.

```

1 > INSERT INTO [silver].[erp_loc_a101]
2     ([cid], [cntry])
3     SELECT
4         REPLACE ([cid], '-', '') cid,
5         CASE
6             WHEN TRIM([cntry]) = 'DE' THEN 'Germany'
7             WHEN TRIM([cntry]) IN ('US', 'USA') THEN 'United States'
8             WHEN TRIM([cntry]) = '' OR [cntry] IS NULL THEN 'N/A'
9             ELSE TRIM([cntry])
10        END AS [cntry]
11
12    FROM [bronze].[erp_loc_a101]
13
104 % No se encontraron problemas.

```

Mensajes

(18484 filas afectadas)

Hora de finalización: 2025-10-15T18:10:44.6621041-03:00

Execution time: 2024-12-29T16:44:33.2315991+01:00



```

16
17   --Check final
18   SELECT DISTINCT [cntry]
19   FROM silver.[erp_loc_a101]
20   ORDER BY [cntry]
21
22   --Final Look at the table
23   SELECT * FROM [silver].[erp_loc_a101]

```

No se encontraron problemas.

	cntry
1	Australia
2	Canada
3	France
4	Germany
5	N/A
6	United Kingdom
7	United States

```

20
21
22
23
24   ORDER BY [cntry]
--Final Look at the table
SELECT * FROM [silver].[erp_loc_a101]

```

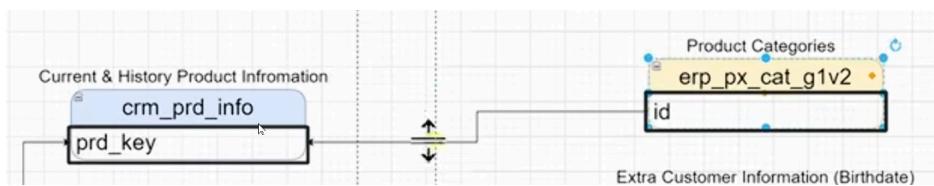
No se encontraron problemas.

	cid	cntry	dwh_create_date
1	AW00011000	Australia	2025-10-15 18:10:44.5600000
2	AW00011001	Australia	2025-10-15 18:10:44.5600000
3	AW00011002	Australia	2025-10-15 18:10:44.5600000
4	AW00011003	Australia	2025-10-15 18:10:44.5600000
5	AW00011004	Australia	2025-10-15 18:10:44.5600000
6	AW00011005	Australia	2025-10-15 18:10:44.5600000
7	AW00011006	Australia	2025-10-15 18:10:44.5600000
8	AW00011007	Australia	2025-10-15 18:10:44.5600000

Build Silver Layer

Clean & Load

erp_px_cat_g1v2



```

SQLQuery11....David (51)* + X
1 SELECT
2   [id],
3   [cat],
4   [subcat],
5   [maintenance]
6   FROM [bronze].[erp_px_cat_g1v2]

```

No se encontraron problemas.

	id	cat	subcat	maintenance
1	AC_BR	Accessories	Bike Racks	Yes
2	AC_BS	Accessories	Bike Stands	No
3	AC_BC	Accessories	Bottles and Cages	No
4	AC_CL	Accessories	Cleaners	Yes
5	AC_FE	Accessories	Fenders	No
6	AC_HF	Accessories	Helmeto	Yes

```

1  SELECT TOP (1000) [prd_id]
2    , [cat_id]
3    , [prd_key]
4    , [prd_nm]
5    , [prd_cost]
6    , [prd_line]
7    , [prd_start_dt]
8    , [prd_end_dt]
9    , [dwh_create_date]
10   FROM [DataWarehouse].[silver].[crm_prd_info]
11

```

No se encontraron problemas.

	prd_id	cat_id	prd_key	prd_nm	prd_cost	prd_line
1	478	AC_BC	BC-M005	Mountain Bottle Cage	4	Mountain
2	479	AC_BC	BC-R205	Road Bottle Cage	3	Road
3	477	AC BC	WB-H098	Water Bottle - 30 oz.	2	Other Sales

Clean up the last table.

We've created an additional table in product information called category id (cat_id), and it completely matches the id in the ERP px cat table g1v2. So there's nothing to do in the PK.

The image shows two separate SSMS windows side-by-side. Both windows have tabs at the top labeled "SQLQuery13....David (87)*" and "SQLQuery12.s...N\David (54)".

The top window contains the following query:

```
1 -- Check for unwanted spaces
2 SELECT *
3 FROM [bronze].[erp_px_cat_g1v2]
4 WHERE [cat] != TRIM([cat])
```

The bottom window contains the following query:

```
1 -- Check for unwanted spaces
2 SELECT *
3 FROM [bronze].[erp_px_cat_g1v2]
4 WHERE [cat] != TRIM([cat])
5 OR [subcat] != TRIM([subcat])
6 OR [maintenance] != TRIM([maintenance])
```

Both windows show a status bar at the bottom indicating "No se encontraron problemas." (No problems found).

It doesn't give us any results, which means we don't have any unnecessary space.
We don't have any unnecessary space.

```
7  ||| -- Data Standardization & Consistency
8  ||| SELECT DISTINCT
9  ||| [cat]
10 ||| FROM [bronze].[erp_px_cat_g1v2]
```

104 % ✓ No se encontraron problemas.

Resultados Mensajes

	cat
1	Accessories
2	Bikes
3	Clothing
4	Components

```
8  ||| -- Data Standardization & Consistency
9  ||| SELECT DISTINCT
10 ||| [cat],
11 ||| [subcat],
12 ||| [maintenance]
13 ||| FROM [bronze].[erp_px_cat_g1v2]
```

104 % ✓ No se encontraron problemas.

Resultados Mensajes

	cat	subcat	maintenance
1	Accessories	Bike Racks	Yes
2	Accessories	Bike Stands	No
3	Accessories	Bottles and Cages	No
4	Accessories	Cleaners	Yes

The data appears to be consistent. Everything looks perfect. The final table is pretty good, with NO issues.

No need to clean anything up.

We're loading from bronze to silver.

```

1 1.~ INSERT INTO [silver].[erp_px_cat_g1v2]
2 ([id], [cat], [subcat], [maintenance])
3     SELECT
4     [id],
5     [cat],
6     [subcat],
7     [maintenance]
8     FROM [bronze].[erp_px_cat_g1v2]

```

104 % No se encontraron problemas.

Mensajes

(37 filas afectadas)

Hora de finalización: 2025-10-15T18:32:28.4417512-03:00

```

14
15 -- Final Check
16   SELECT * FROM [silver].[erp_px_cat_g1v2]

```

104 % No se encontraron problemas.

Resultados Mensajes

	id	cat	subcat	maintenance	dwh_create_date
1	AC_BR	Accessories	Bike Racks	Yes	2025-10-15 18:32:28.4400000
2	AC_BS	Accessories	Bike Stands	No	2025-10-15 18:32:28.4400000
3	AC_BC	Accessories	Bottles and Cages	No	2025-10-15 18:32:28.4400000
4	AC_CL	Accessories	Cleaners	Yes	2025-10-15 18:32:28.4400000
5	AC_FE	Accessories	Fenders	No	2025-10-15 18:32:28.4400000
6	AC_HE	Accessories	Helmets	Yes	2025-10-15 18:32:28.4400000

final check.

First you have to truncate the data and then do a full load.

For inserting the data we need to make sure to truncate the table. First truncate the data and then make a fully loaded

```

2
3     -- Loading silver.crm_cust_info
4 PRINT '>> Truncating Table: [silver].[crm_cust_info]';
5 TRUNCATE TABLE [silver].[crm_cust_info];
6 PRINT '>> Inserting Data Into: [silver].[crm_cust_info]';
7 INSERT INTO silver.crm_cust_info (
8     cst_id,
9     cst_key,
10    cst_firstname,
11    cst_lastname,
12    cst_marital_status,
13    cst_gndr,
14    cst_create_date
15 )
16     SELECT
17     cst_id

```

104 % No se encontraron problemas.

Mensajes

(18484 filas afectadas)

Hora de finalización: 2025-10-16T06:52:31.1368178-03:00

(18484 filas afectadas)

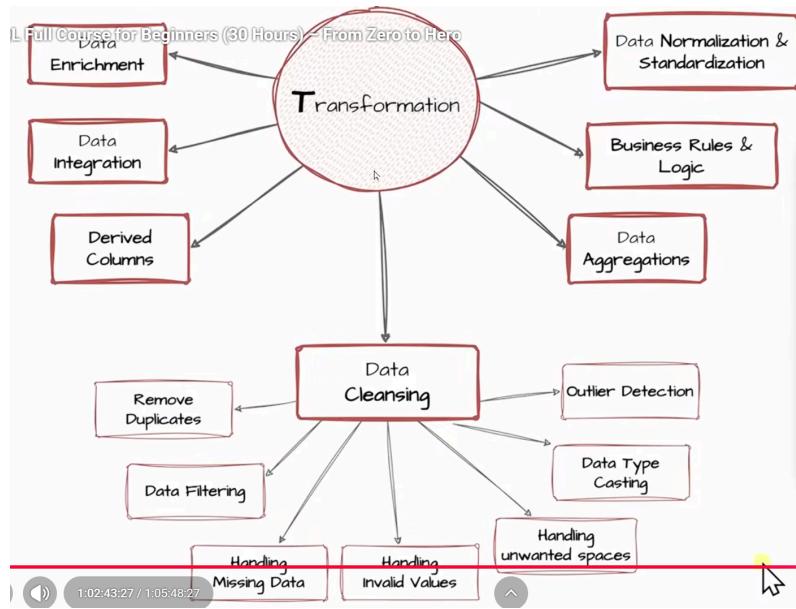
Hora de finalización: 2025-10-16T07:02:15.8339930-03:00

I run them all at the same time. We empty all the tables and insert the data.



CONSISTENCY

If you introduce an improvement, like better logging or error handling, in one stored procedure, apply it to the others to maintain consistent standards and benefits.



Build Silver Layer

Document: Data Flow

Build Silver Layer

Commit Code in Git Repo

Build Gold Layer

Epics	Tasks	
Build Gold Layer	Analysing: Explore Business Objects	<input type="checkbox"/>
Build Gold Layer	Coding: Data Integration	<input type="checkbox"/>
Build Gold Layer	Validating: Data Integration Checks	<input type="checkbox"/>
Build Gold Layer	Document: Draw Data Model of Star Schema (Draw.io)	<input type="checkbox"/>
Build Gold Layer	Document: Create Data Catalog	<input type="checkbox"/>
Build Gold Layer	Document: Extend Data Flow (Draw.io)	<input type="checkbox"/>
Build Gold Layer	Commit Code in Git Repo	    

the end