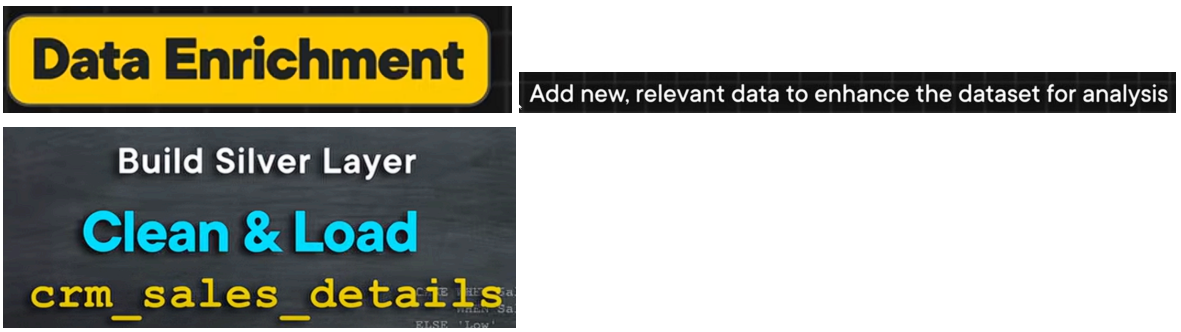


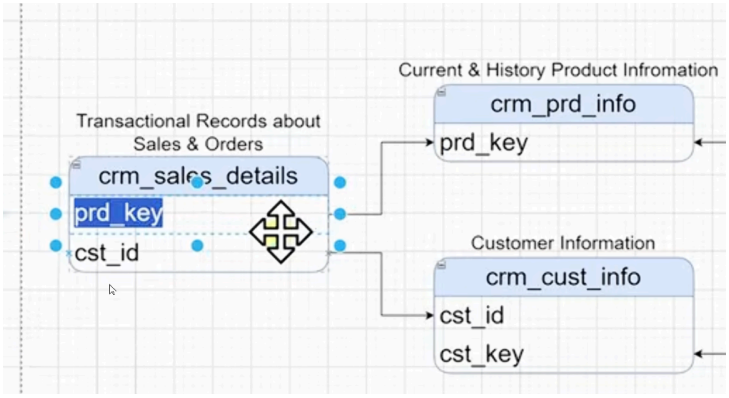

```
SELECT
[sls_ord_num],
[sls_prd_key],
[sls_cust_id],
[sls_order_dt],
[sls_ship_dt],
[sls_due_dt],
[sls_sales],
[sls_quantity],
[sls_price]
FROM [bronze].[crm_sales_details]
WHERE [sls_ord_num] != TRIM([sls_ord_num])
```

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.

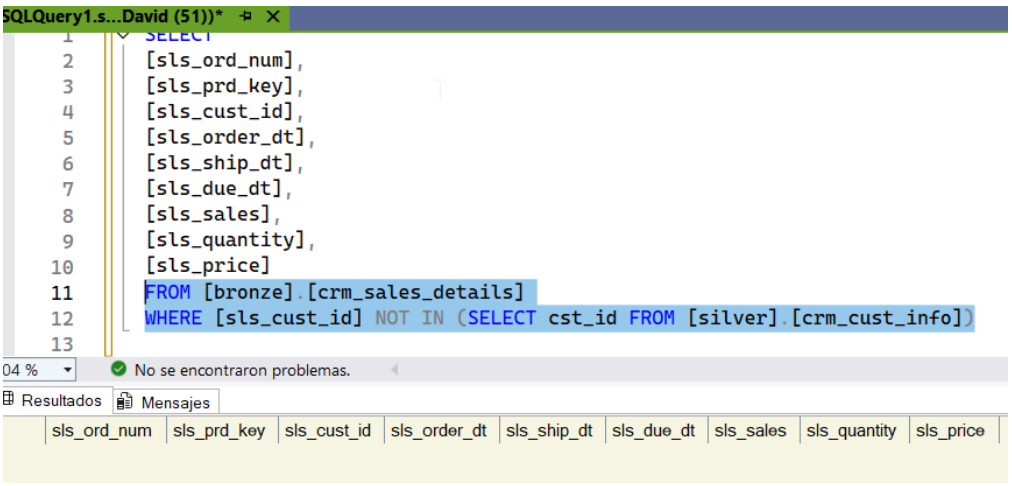
```
SELECT
[sls_ord_num],
[sls_prd_key],
[sls_cust_id],
[sls_order_dt],
[sls_ship_dt],
[sls_due_dt],
[sls_sales],
[sls_quantity],
[sls_price]
FROM [bronze].[crm_sales_details]
WHERE [sls_prd_key] NOT IN (SELECT [sls_prd_key] FROM [silver].[crm_prd_info])
```

104 % No se encontraron problemas.

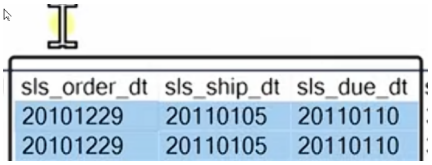
Resultados Mensajes

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.

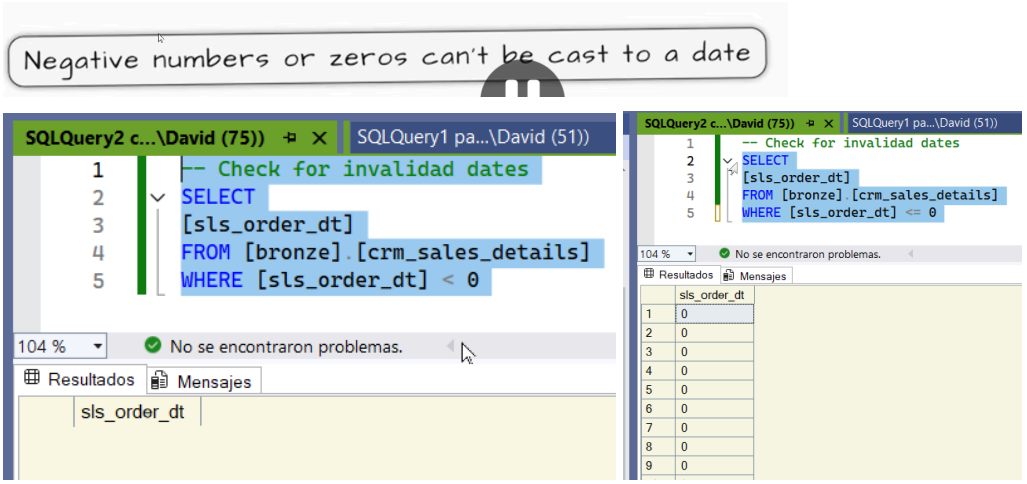


There's no problem, so we can connect to both tables without any problems. No transformations are required.

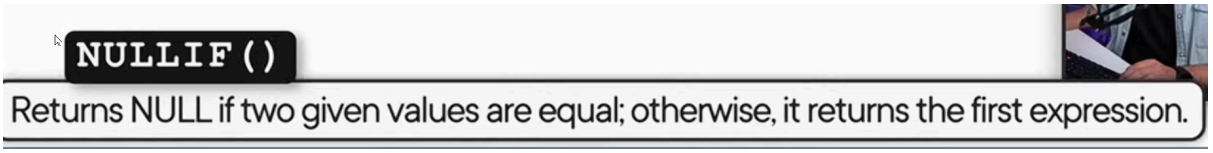


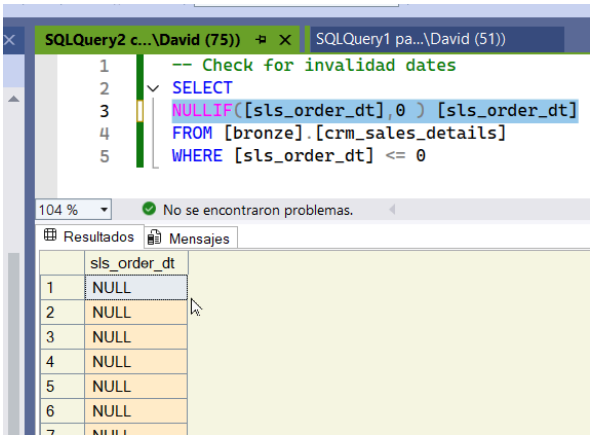
sls_order_dt	sls_ship_dt	sls_due_dt
20101229	20110105	20110110
20101229	20110105	20110110

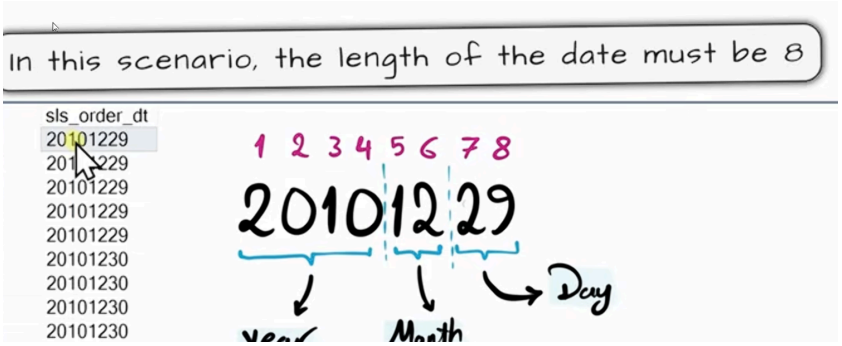
We continue with these columns. They're not dates, they're INTEGER (INT), they're numbers, and we like them that way. We need to change them from INT to DATE.

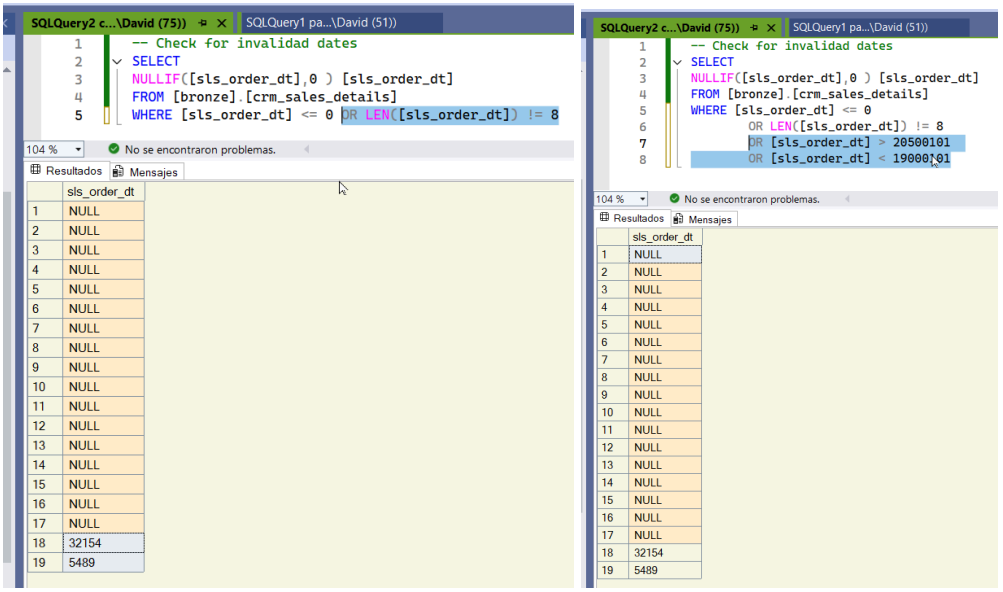


This is a problem because we have dates that say 0.
We can replace this information with null-.









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.

```
SELECT
[sls_ord_num],
[sls_prd_key],
[sls_cust_id],
[sls_order_dt],
    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],
[sls_ship_dt],
[sls_due_dt],
[sls_sales],
[sls_quantity],
[sls_price]
FROM [bronze].[crm_sales_details]
```

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.

```
-- Check for invalidad dates
SELECT
NULLIF([sls_ship_dt],0 ) [sls_ship_dt]
FROM [bronze].[crm_sales_details]
WHERE [sls_ship_dt] <= 0
        OR LEN([sls_ship_dt]) != 8
        OR [sls_ship_dt] > 20500101
        OR [sls_ship_dt] < 19000101
```

sls_ship_dt

```
[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],
[sls_due_dt],
[sls_sales],
[sls_quantity],
[sls_price]
FROM [bronze].[crm_sales_details]
```

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
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.

```
-- Check for invalidad dates
SELECT
NULLIF([sls_due_dt],0 ) [sls_due_dt]
FROM [bronze].[crm_sales_details]
WHERE [sls_due_dt] <= 0
        OR LEN([sls_due_dt]) != 8
        OR [sls_due_dt] > 20500101
        OR [sls_due_dt] < 19000101
```

sls_due_dt

```
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],
[sls_sales],
[sls_quantity],
[sls_price]
FROM [bronze].[crm_sales_details]
```

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_quantit
1	SO43697	BK-R93R-62	21768	2010-12-29	2011-01-05	2011-01-10	3578	1
2	SO43698	BK-M82S-44	28389	2010-12-29	2011-01-05	2011-01-10	3400	1

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

```
OR [sls_due_dt] < 19000101

-- Check for invalidad Date Orders
SELECT
*
FROM [bronze].[crm_sales_details]
WHERE sls_order_dt > [sls_ship_dt]
    OR sls_order_dt >  [sls_due_dt]
```

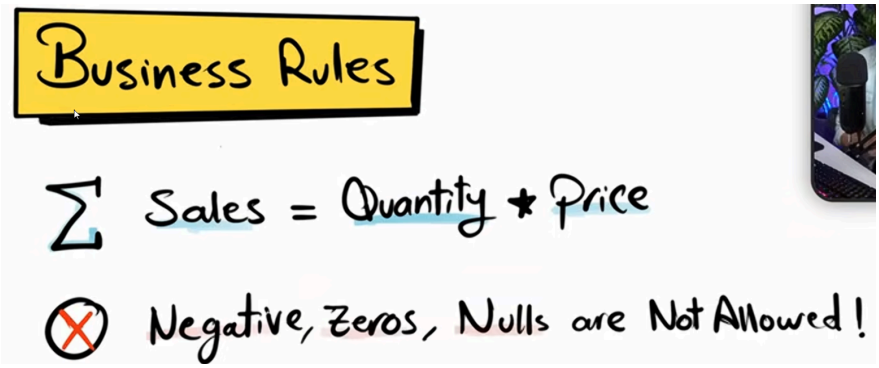
No se encontraron problemas.

Resultados | Mensajes

sls_ord_num	sls_prd_key	sls_cust_id	sls_order_dt	sls_ship_dt

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

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



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

```
-- Values must not be NULL, Zero or Negative.

SELECT
[sls_sales],
[sls_quantity],
[sls_price]
FROM [bronze].[crm_sales_details]
WHERE [sls_sales] != [sls_quantity]*[sls_price]
```

104 % No se encontraron problemas.

Resultados | Mensajes

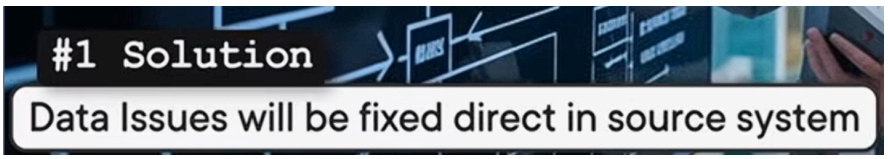
	sls_sales	sls_quantity	sls_price
1	769	1	-769
2	30	1	-30
3	22	1	-22
4	35	2	35
5	1701	1	-1701

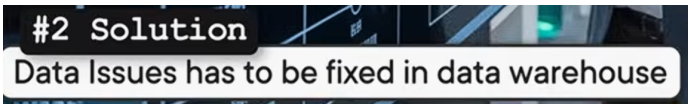
```
SELECT DISTINCT
[sls_sales],
[sls_quantity],
[sls_price]
FROM [bronze].[crm_sales_details]
WHERE [sls_sales] != [sls_quantity]*[sls_price]
    OR [sls_sales] IS NULL OR [sls_quantity] IS NULL OR [sls_price] IS NULL
    OR [sls_sales] <= 0 OR [sls_quantity] <= 0 OR [sls_price] <= 0
ORDER BY [sls_sales], [sls_quantity], [sls_price]
```

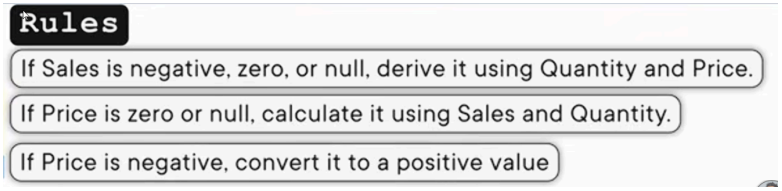
104 % No se encontraron problemas.

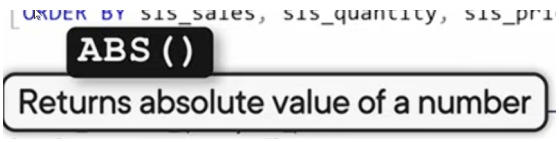
Resultados | Mensajes

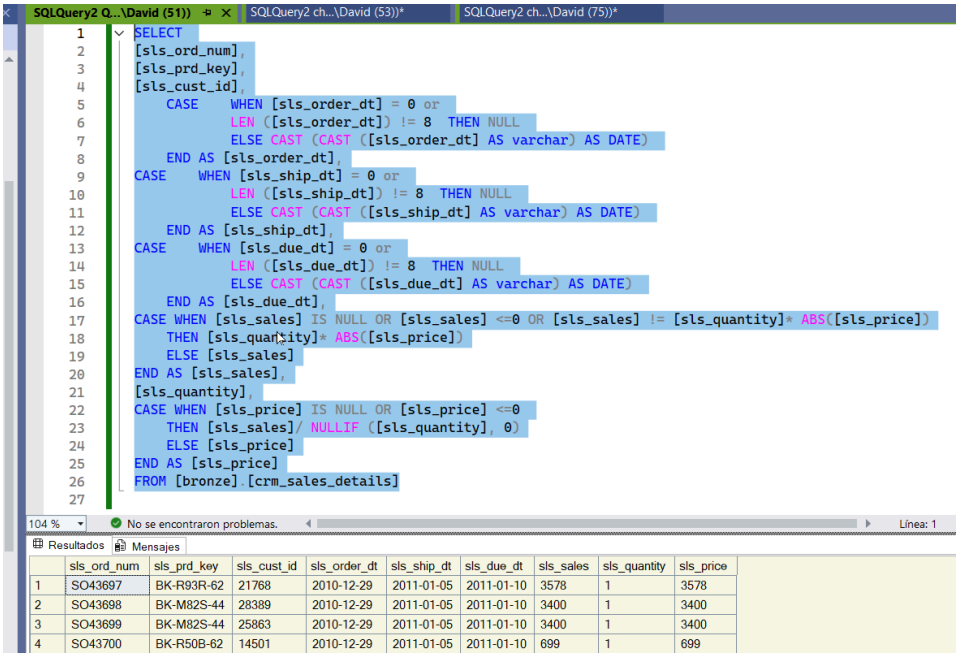
	sls_sales	sls_quantity	sls_price
1	NULL	1	2
2	NULL	1	8











```
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],
[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]
```

104 % — No se encontraron problemas. — Línea: 1

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
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.

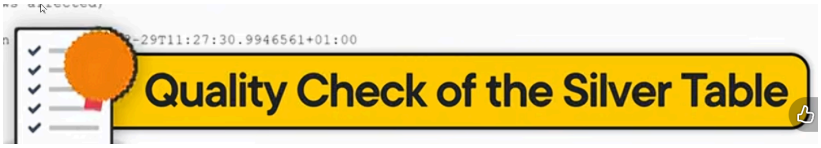
```sql
IF OBJECT_ID ('[silver].[crm_sales_details]', 'U') IS NOT NULL
    DROP TABLE [silver].[crm_sales_details]
CREATE TABLE [silver] [crm_sales_details] (
    [sls_ord_num] NVARCHAR(50),
    [sls_prd_key] NVARCHAR(50),
    [sls_cust_id] INT,
    [sls_order_dt] DATE,
    [sls_ship_dt] DATE,
    [sls_due_dt] DATE,
    [sls_sales] INT,
    [sls_quantity] INT,
    [sls_price] INT,
    [dwh_create_date] DATETIME2 DEFAULT GETDATE()
);
```

```
Los comandos se han completado correctamente.

Hora de finalización: 2025-10-15T12:54:13.5890366-03:00
```

```sql
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] !
```

```
(60398 filas afectadas)

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.



```sql
-- Check for invalidad Date Orders
SELECT
*
FROM silver.[crm_sales_details]
WHERE sls_order_dt > [sls_ship_dt]
    OR sls_order_dt >  [sls_due_dt]

-- Check Data Consistency: Between Sales, Quantity, and Pric
-- Sales = Quantity * Price
```

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_

```sql
-- Check Data Consistency: Between Sales, Quantity, and Price
-- Sales = Quantity * Price
-- Values must not be NULL, Zero or Negative.

SELECT DISTINCT
[sls_sales],
[sls_price],
[sls_quantity]
FROM silver.[crm_sales_details]
WHERE [sls_sales] != [sls_quantity]*[sls_price]
    OR [sls_sales] IS NULL OR [sls_quantity] IS NULL OR [sls_price] IS NULL
    OR [sls_sales] <= 0 OR [sls_quantity] <= 0 OR [sls_price] <= 0
ORDER BY [sls_sales], [sls_quantity], [sls_price]
```

104 % No se encontraron problemas.

Resultados Mensajes

sls_sales	sls_price	sls_quantity

```sql
--Final Check
SELECT * FROM [silver].[crm_sales_details]
```

104 % No se encontraron problemas. Línea: 3

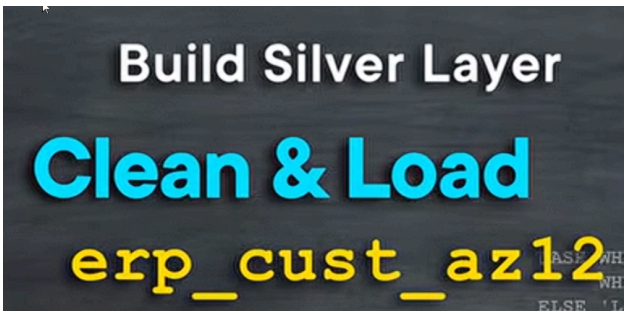
Resultados Mensajes

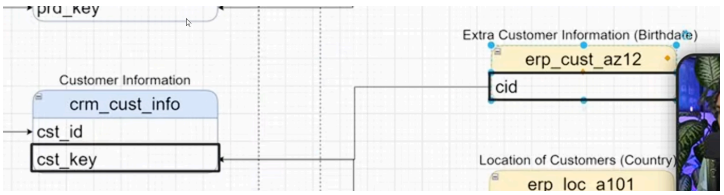
	sls_ord_num	sls_prd_key	sls_cust_id	sls_order_dt	sls_ship_dt	sls_due_dt	sls_sales	sls_quantity	sls_price	dwh_create_date
1	SO43697	BK-R93R-62	21768	2010-12-29	2011-01-05	2011-01-10	3578	1	3578	2025-10-15 12:57:31.5500000
2	SO43698	BK-M82S-44	28389	2010-12-29	2011-01-05	2011-01-10	3400	1	3400	2025-10-15 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

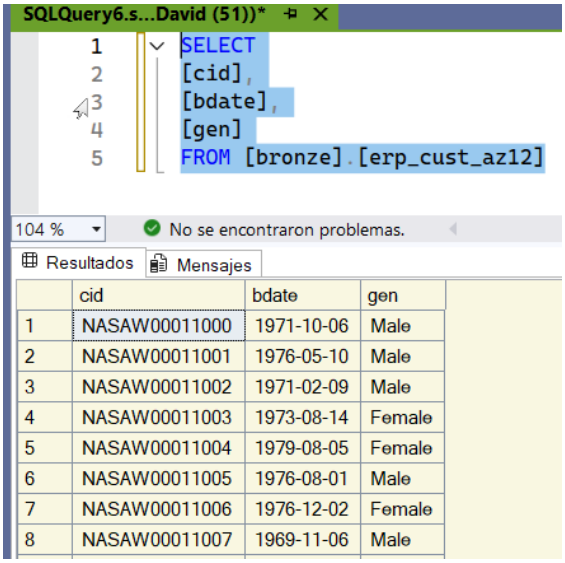
Everything is perfect!!!

Now in international format!!!

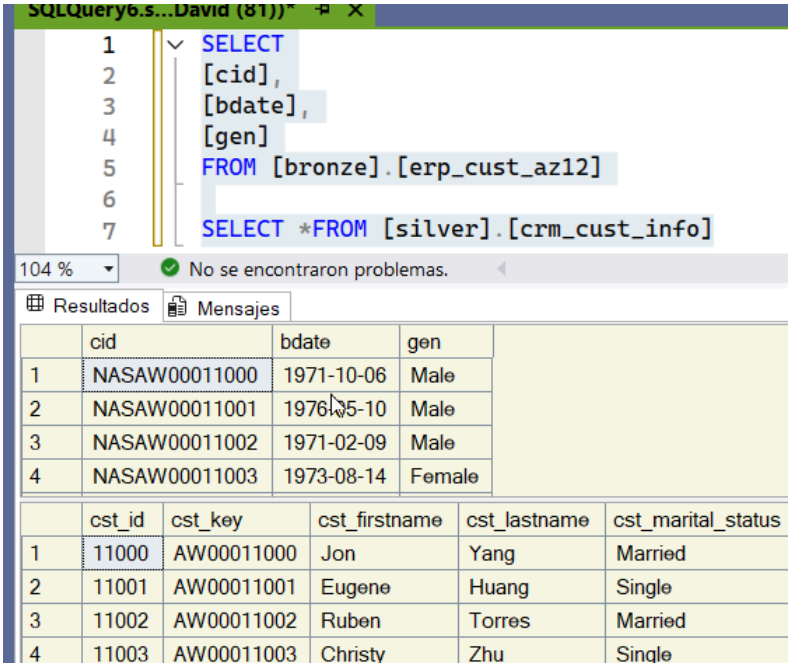
```sql
/* ===============================================================================
   Script:      Load Sales Details - Bronze to Silver
   Author:      Ignacio David Stocco
   Date:        2025-10-15
   Purpose:     Cleans and standardizes sales data for the silver layer
   Notes:       Converts 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(CAST([sls_order_dt] AS VARCHAR(8)) AS DATE)
    END AS [sls_order_dt],
    END AS [sls_order_dt],
    CASE
        WHEN [sls_ship_dt] = 0
             OR LEN([sls_ship_dt]) != 8
        THEN NULL
        ELSE TRY_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 TRY_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],  -- 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];
```







	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



	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.

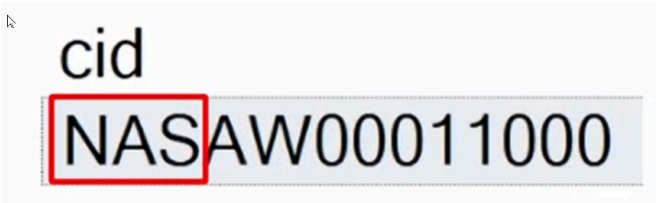
SQLQuery6.s...David (81))*

```
SELECT
[cid],
CASE
    WHEN cid LIKE 'NAS%'
    THEN SUBSTRING (cid, 4, LEN(cid))
    ELSE cid
END cid,
[bdate],
[gen]
FROM [bronze].[erp_cust_az12]
```

104 % No se encontraron problemas.

Resultados Mensajes

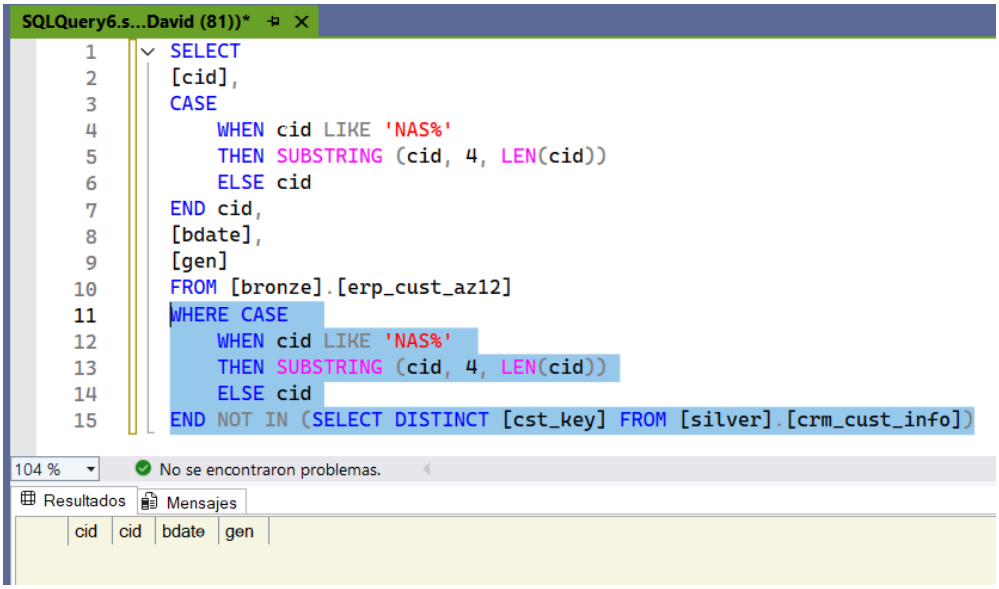
	cid	cid	bdate	gen
1	NASAW00011000	AW00011000	1971-10-06	Male
2	NASAW00011001	AW00011001	1976-05-10	Male
3	NASAW00011002	AW00011002	1971-02-09	Male
4	NASAW00011003	AW00011003	1973-08-14	Female
5	NASAW00011004	AW00011004	1979-08-05	Female



We removed NAS at the beginning. If the cid is like 3 characters. If the Id start with those 3 cgharacters 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 toi define how many chaqatders 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 rst of the characters.

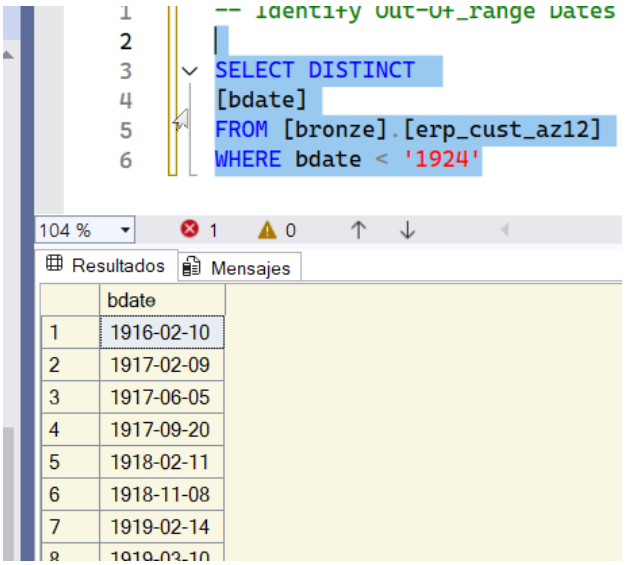
We don't have any NAS at the start

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



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.

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

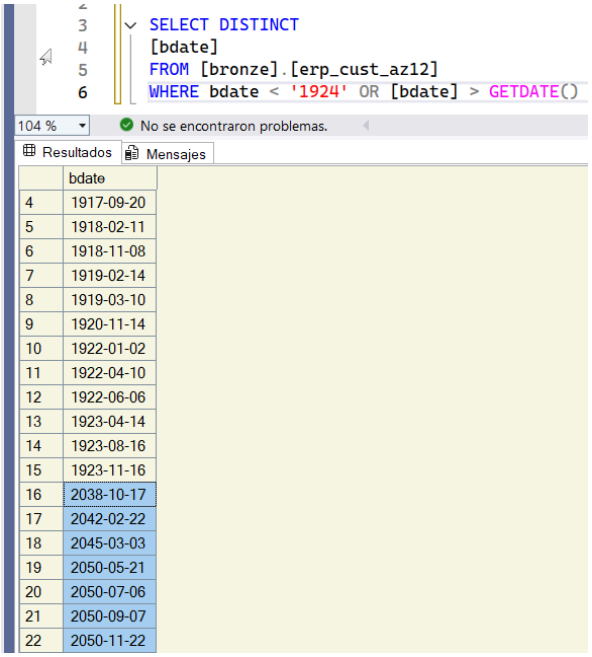


```
-- Identify Out-Of_range Dates

SELECT DISTINCT
[bdate]
FROM [bronze].[erp_cust_az12]
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



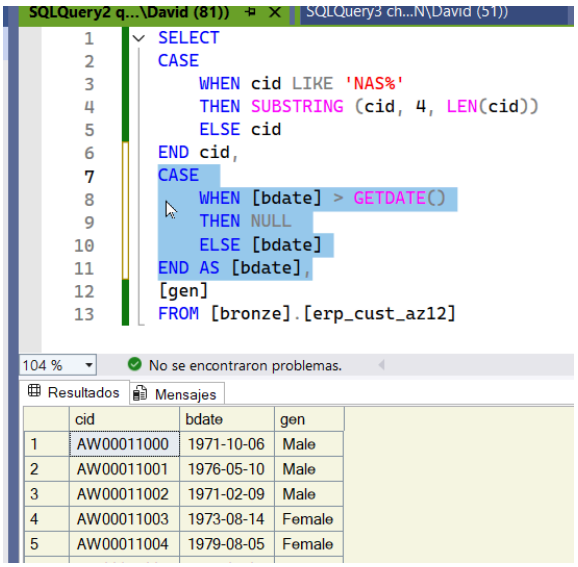
```
SELECT DISTINCT
[bdate]
FROM [bronze].[erp_cust_az12]
WHERE bdate < '1924' OR [bdate] > GETDATE()
```

104 % No se encontraron problemas.

Resultados Mensajes

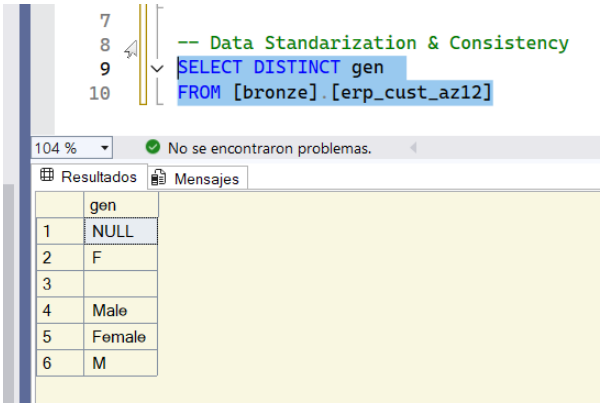
	bdate
4	1917-09-20
5	1918-02-11
6	1918-11-08
7	1919-02-14
8	1919-03-10
9	1920-11-14
10	1922-01-02
11	1922-04-10
12	1922-06-06
13	1923-04-14
14	1923-08-16
15	1923-11-16
16	2038-10-17
17	2042-02-22
18	2045-03-03
19	2050-05-21
20	2050-07-06
21	2050-09-07
22	2050-11-22

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.



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

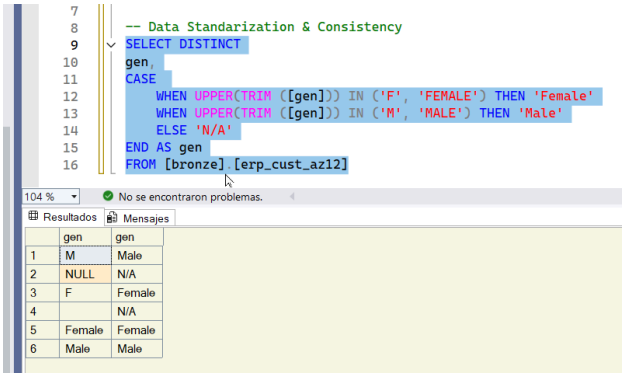
	cid	bdate	gen
1	AW00011000	1971-10-06	Male
2	AW00011001	1976-05-10	Male
3	AW00011002	1971-02-09	Male
4	AW00011003	1973-08-14	Female
5	AW00011004	1979-08-05	Female



```
-- Data Standarization & Consistency
SELECT DISTINCT gen
FROM [bronze].[erp_cust_az12]
```

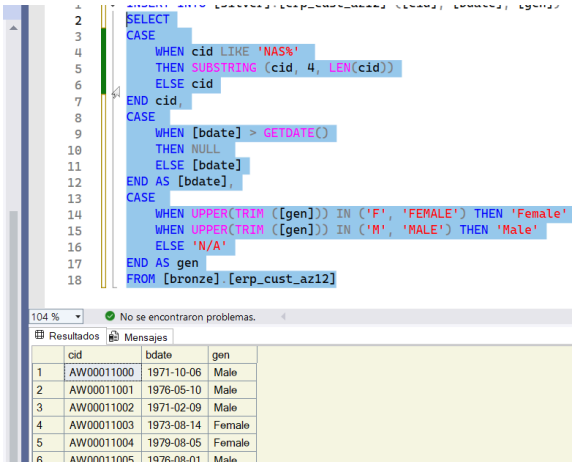
	gen
1	NULL
2	F
3	
4	Male
5	Female
6	M

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.



```
-- Data Standarization & Consistency
SELECT DISTINCT
gen,
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]
```

	gen	gen
1	M	Male
2	NULL	N/A
3	F	Female
4		N/A
5	Female	Female
6	Male	Male



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

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

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

```sql
INSERT INTO [silver].[erp_cust_az12] ([cid], [bdate], [gen])
SELECT
CASE
    WHEN cid LIKE 'NAS%'
    THEN SUBSTRING (cid, 4, LEN(cid))
    ELSE cid
END 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]
```

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 (

```sql
-- Identify Out-Of_range Dates

SELECT DISTINCT
    [bdate]
FROM silver.[erp_cust_az12]
WHERE bdate < '1924'
    OR [bdate] > GETDATE();
```

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

```sql
-- Data Standarization & Consistency
SELECT DISTINCT
gen,
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 silver.[erp_cust_az12];
```

104 % — No se encontraron problemas.

Resultados | Mensajes

	gen	gen
1	N/A	N/A
2	Female	Female
3	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

```sql
-- Final Check
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

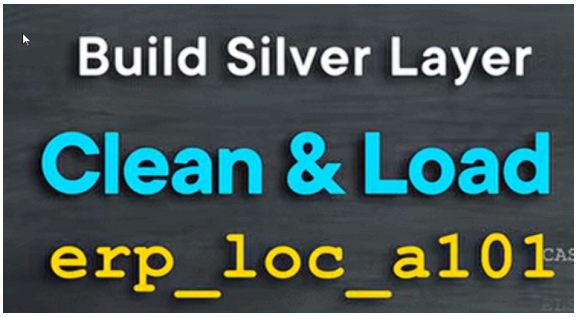

```sql
SELECT
    CASE
        WHEN cid LIKE 'NAS%' THEN SUBSTRING(cid, 4, LEN(cid)) -
        ELSE cid
    END AS cid,
    CASE
```

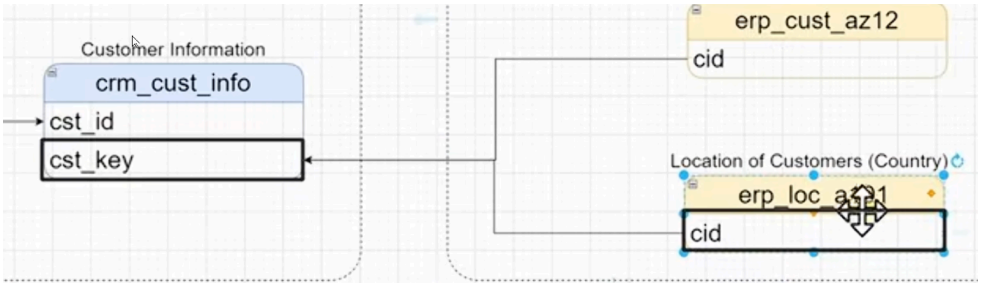
}we have hanlded invalidid values.

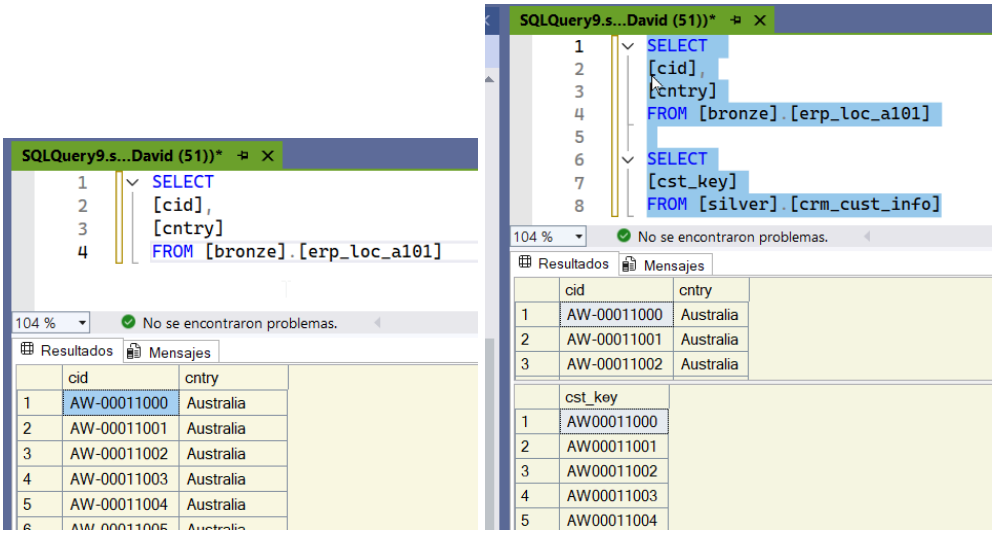
```sql
    END AS cid,
    CASE
        WHEN bdate > GETDATE() THEN NULL
        ELSE bdate
    END AS bdate, -- Set future birthdates to NULL
    CASE
```

```sql
    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
```

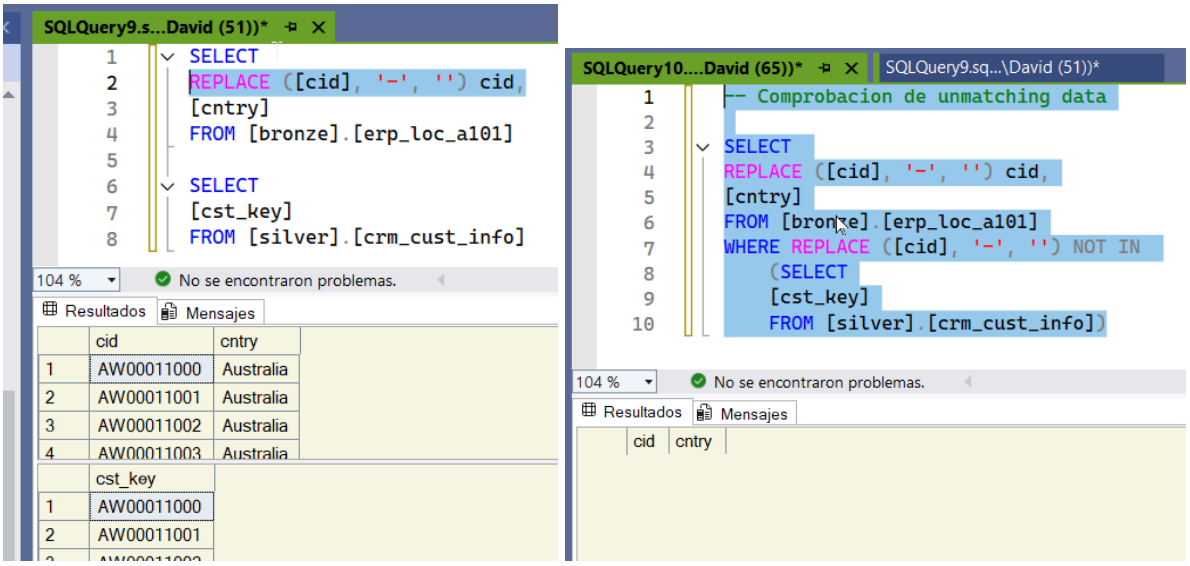
we have done data normalization.







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



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

```sql
-- Data Standarizatiuon & Consistency
SELECT DISTINCT [cntry]
FROM [bronze].[erp_loc_a101]
ORDER BY [cntry]
```

104 % — No se encontraron problemas.

Resultados | Mensajes

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

```sql
SELECT
    REPLACE ([cid], '-', '') 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]
```

104 % — No se encontraron problemas.

Resultados | Mensajes

	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.

```sql
INSERT INTO [silver].[erp_loc_a101]
([cid],[cntry] )
SELECT
    REPLACE ([cid], '-', '') 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]
```

104 % — No se encontraron problemas.

Mensajes

```
(18484 filas afectadas)

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

```sql
--Check final
SELECT DISTINCT [cntry]
FROM silver.[erp_loc_a101]
ORDER BY [cntry]

--Final Look at the table
SELECT * FROM [silver].[erp_loc_a101]
```

No se encontraron problemas.

Resultados | Mensajes

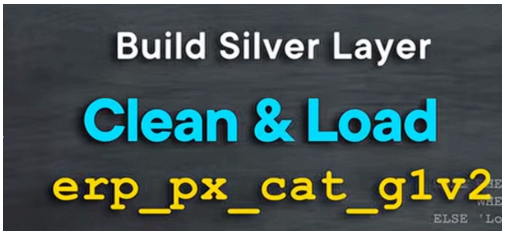
	cntry
1	Australia
2	Canada
3	France
4	Germany
5	N/A
6	United Kingdom
7	United States

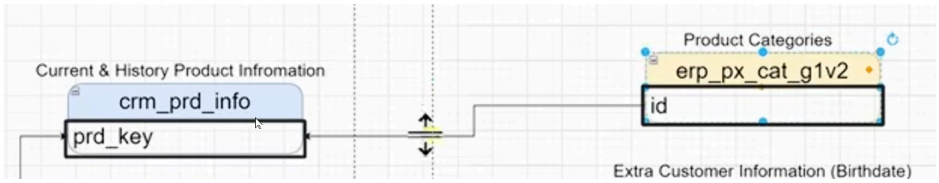
```sql
ORDER BY [cntry]

--Final Look at the table
SELECT * FROM [silver].[erp_loc_a101]
```

No se encontraron problemas.

Resultados | Mensajes

	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





SQLQuery11....David (51))*

```sql
SELECT
[id],
[cat],
[subcat],
[maintenance]
FROM [bronze].[erp_px_cat_g1v2]
```

104 % No se encontraron problemas.

Resultados | Mensajes

	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_HE	Accessories	Helmets	Yes

```sql
SELECT TOP (1000) [prd_id]
      ,[cat_id]
      ,[prd_key]
      ,[prd_nm]
      ,[prd_cost]
      ,[prd_line]
      ,[prd_start_dt]
      ,[prd_end_dt]
      ,[dwh_create_date]
  FROM [DataWarehouse].[silver].[crm_prd_info]
```

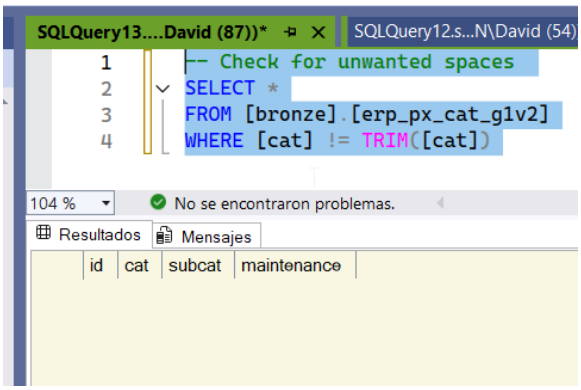
104 % No se encontraron problemas.

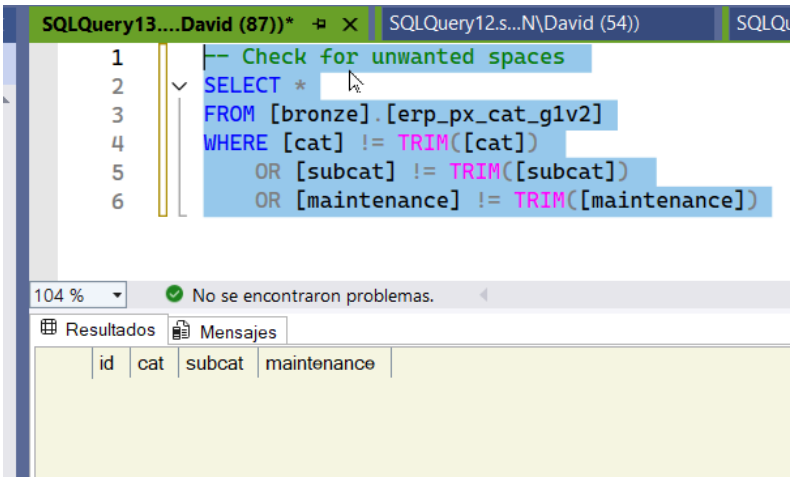
Resultados | Mensajes

	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

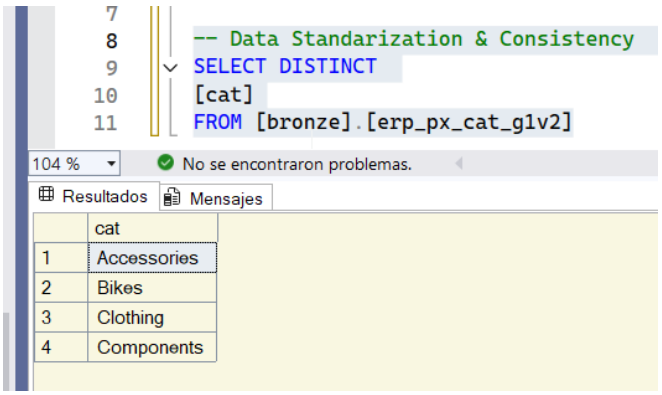
Clean up the last table.

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.





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

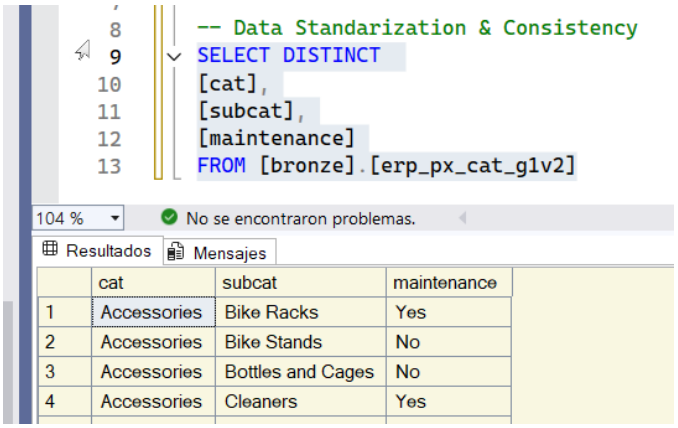


```
-- Data Standarization & Consistency
SELECT DISTINCT
[cat]
FROM [bronze].[erp_px_cat_g1v2]
```

104 % — No se encontraron problemas.

Resultados | Mensajes

	cat
1	Accessories
2	Bikes
3	Clothing
4	Components



```
-- Data Standarization & Consistency
SELECT DISTINCT
[cat],
[subcat],
[maintenance]
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.

```
INSERT INTO [silver].[erp_px_cat_g1v2]
([id], [cat], [subcat], [maintenance])
    SELECT
    [id],
    [cat],
    [subcat],
    [maintenance]
    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
```

```
-- Final Check
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

```
        -- Loading silver.crm_cust_info
PRINT '>> Truncating Table: [silver].[crm_cust_info]';
TRUNCATE TABLE [silver].[crm_cust_info];
PRINT '>> Inserting Data Into: [silver].[crm_cust_info]';
INSERT INTO silver.crm_cust_info (
            cst_id,
            cst_key,
            cst_firstname,
            cst_lastname,
            cst_marital_status,
            cst_gndr,
            cst_create_date
        )
        SELECT
```

104 % — No se encontraron problemas.

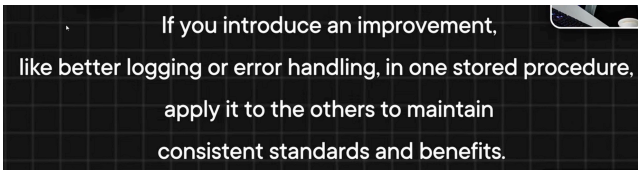
Mensajes

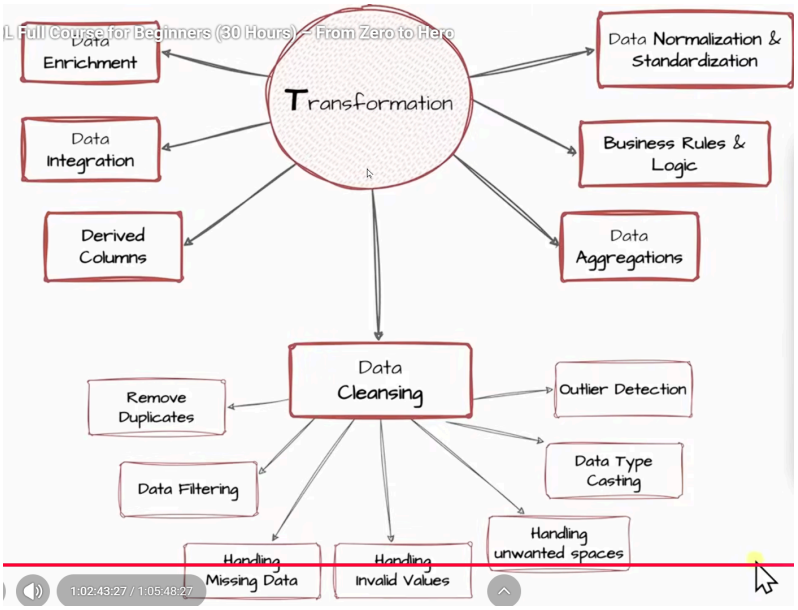
```
(18484 filas afectadas)

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

```
>> Truncating Table: [silver].[crm_cust_info]
>> Inserting Data Into: [silver].[crm_cust_info]

(18484 filas afectadas)
>> Truncating Table: silver.crm_prd_info
>> Inserting Data Into: silver.crm_prd_info

(397 filas afectadas)
>> Truncating Table: silver.crm_sales_details
>> Inserting Data Into: silver.crm_sales_details

(60398 filas afectadas)
>> Truncating Table: silver.erp_cust_az12
>> Inserting Data Into: silver.erp_cust_az12

(18483 filas afectadas)
>> Truncating Table: silver.erp_loc_a101
>> Inserting Data Into: silver.erp_loc_a101

(18484 filas afectadas)
>> Truncating Table: silver.erp_px_cat_g1v2
>> Inserting Data Into: silver.erp_px_cat_g1v2

(37 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.

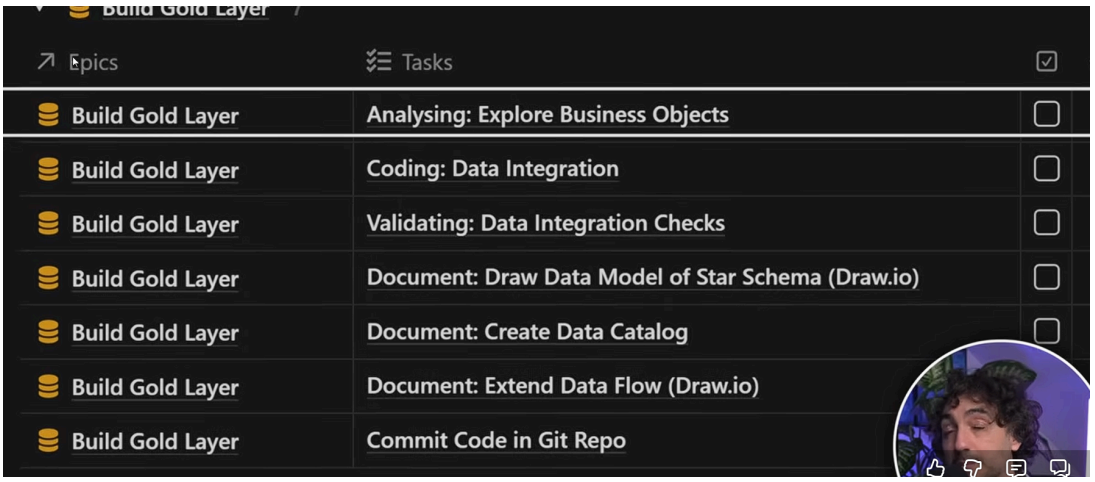

CONSISTENCY


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.


L Full Course for Beginners (30 Hours) – From Zero to Hero
Transformation
Data Enrichment
Data Integration
Derived Columns
Data Normalization & Standardization
Business Rules & Logic
Data Aggregations
Data Cleansing
Remove Duplicates
Data Filtering
Handling Missing Data
Handling Invalid Values
Outlier Detection
Data Type Casting
Handling unwanted spaces
1:02:43:27 / 1:05:48:27


Build Silver Layer
Document: Data Flow


Build Silver Layer
Commit Code in Git Repo


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

the end