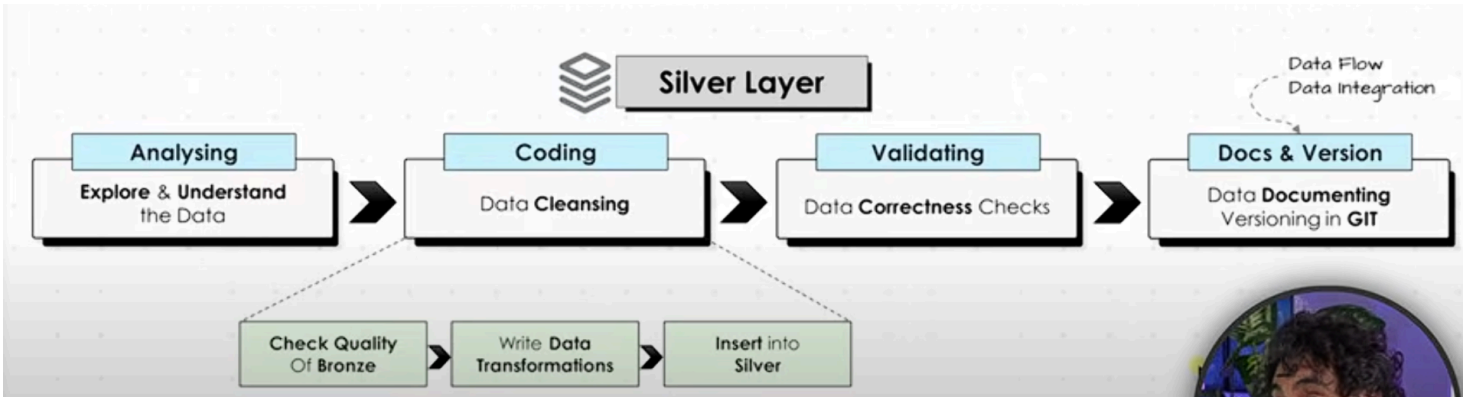
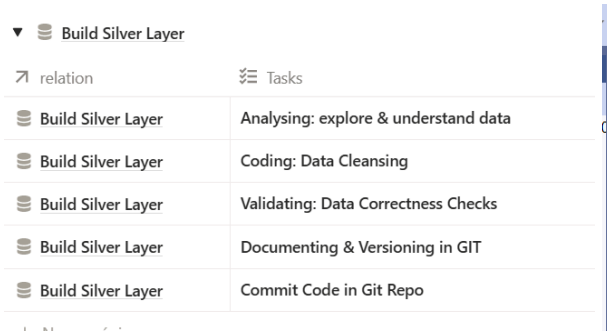
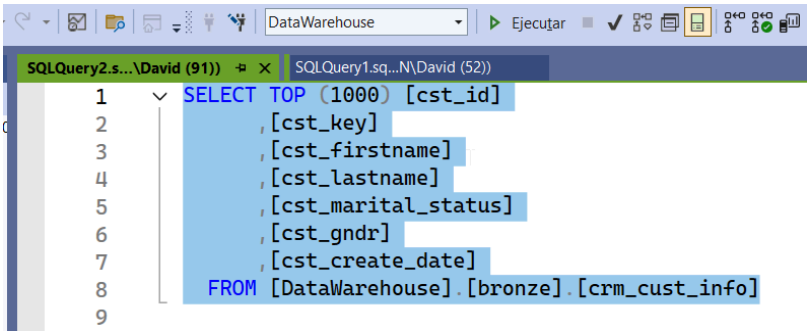
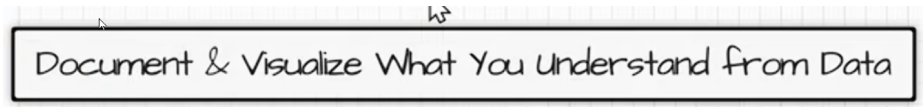
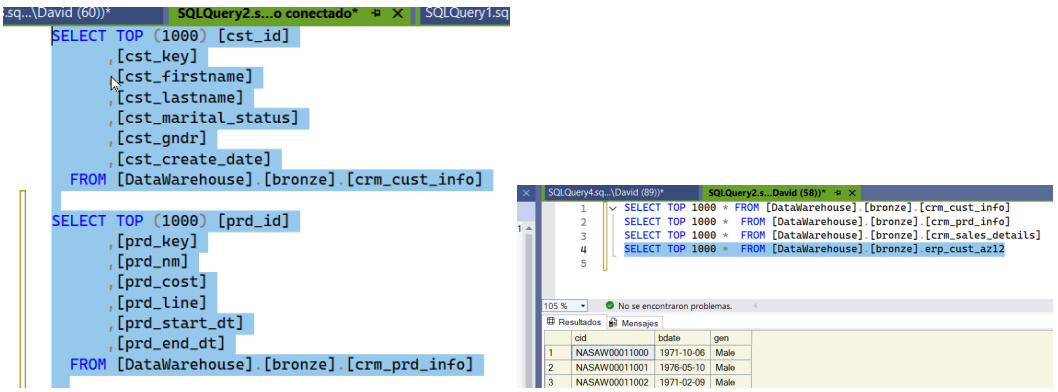




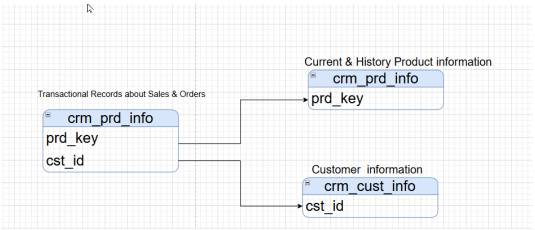
relation	Tasks
Build Silver Layer	Analysing: explore & understand data
Build Silver Layer	Coding: Data Cleansing
Build Silver Layer	Validating: Data Correctness Checks
Build Silver Layer	Documenting & Versioning in GIT
Build Silver Layer	Commit Code in Git Repo





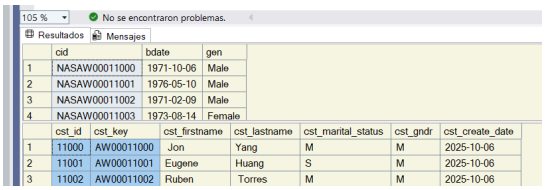


We continue analyzing table by table and put them all in the same query.
We look at each table individually and see how we can join them.




```
SELECT TOP 1000 *  FROM [DataWarehouse].[bronze].[crm_prd_info]
SELECT TOP 1000 *  FROM [DataWarehouse].[bronze].[crm_sales_details]
SELECT TOP 1000 *  FROM [DataWarehouse].[bronze].erp_cust_az12
SELECT TOP 1000 * FROM [DataWarehouse].[bronze].[crm_cust_info]
```

Little by little, we are joining the tables.



105 % No se encontraron problemas.

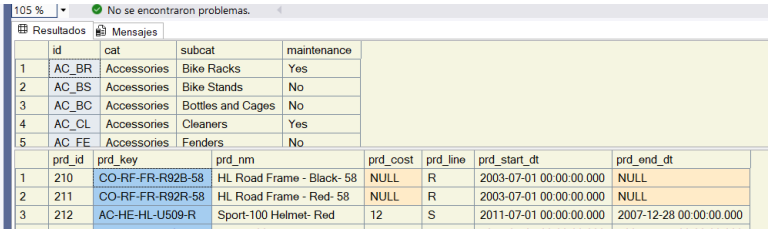
Resultados Mensajes

	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	cst_gndr	cst_create_date
1	11000	AW00011000	Jon	Yang	M	M	2025-10-06
2	11001	AW00011001	Eugene	Huang	S	M	2025-10-06
3	11002	AW00011002	Ruben	Torres	M	M	2025-10-06

We can join these two tables using the `customer_key` ID.
CTRL+Enter adds a new row in Vaio.
ERP PX catalog contains product categories and subcategories. We have special identifiers for that information.

```
SELECT TOP 1000 *  FROM [DataWarehouse].[bronze].erp_loc_a101
SELECT TOP 1000 *  FROM [DataWarehouse].[bronze].erp_px_cat_g1v2
SELECT TOP 1000 *  FROM [DataWarehouse].[bronze].[crm_prd_info]
```



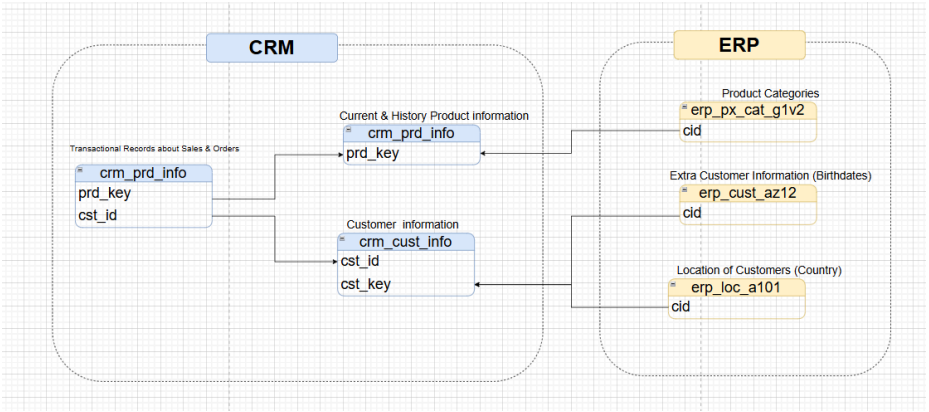
105 % No se encontraron problemas.

Resultados Mensajes

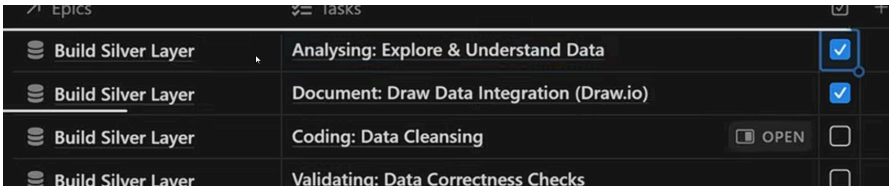
	id	cat	subcat	maintenance
1	AC_BR	Accessories	Bike Racks	Yes
2	AC_BS	Accessories	Bike Stands	No
3	AC_BC	Accessories	Bottles and Cages	No
4	AC_CL	Accessories	Cleaners	Yes
5	AC_FE	Accessories	Fenders	No

	prd_id	prd_key	prd_nm	prd_cost	prd_line	prd_start_dt	prd_end_dt
1	210	CO-RF-FR-R92B-58	HL Road Frame - Black- 58	NULL	R	2003-07-01 00:00:00.000	NULL
2	211	CO-RF-FR-R92R-58	HL Road Frame - Red- 58	NULL	R	2003-07-01 00:00:00.000	NULL
3	212	AC-HE-HL-U509-R	Sport-100 Helmet- Red	12	S	2011-07-01 00:00:00.000	2007-12-28 00:00:00.000

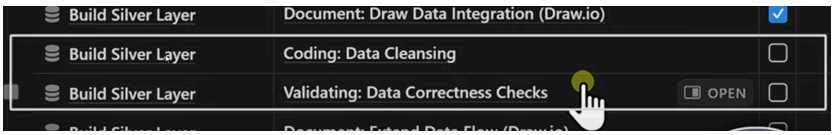
We can join those two tables using the source system `CRM`.

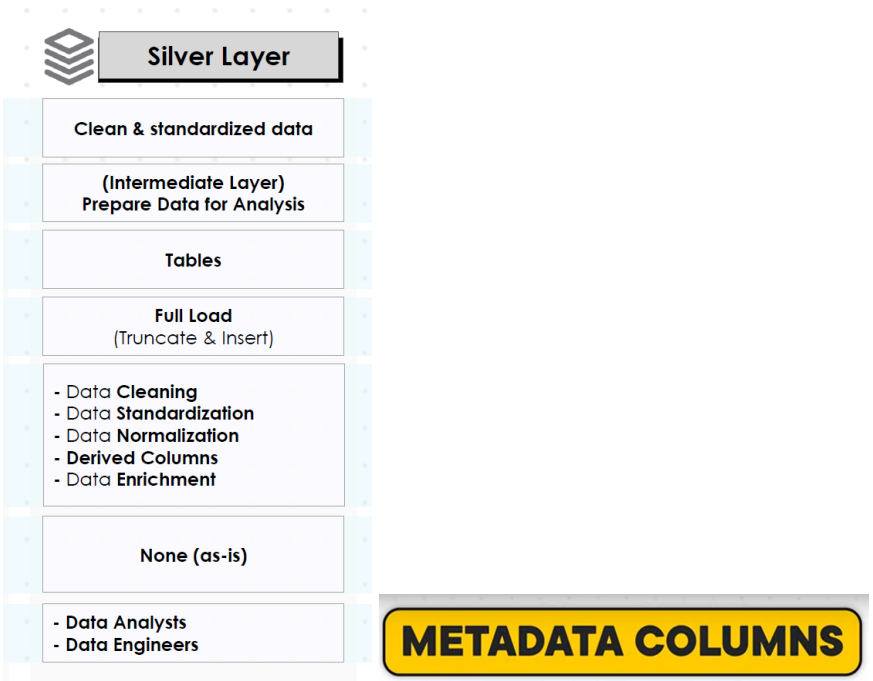


Now it’s much clearer how the tables are connected to each other.



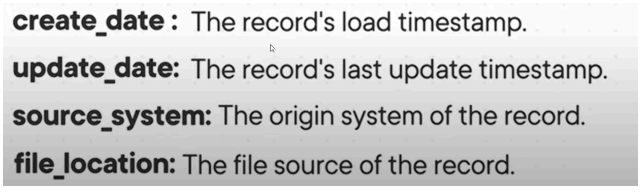
DATA JANITOR





In this part, we have a lot of data transformation.

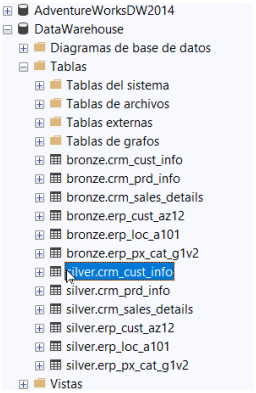
These are extra columns or fills that the data engineers add to each table and don't come directly from the source system.



its good to see if I have gap in my data. Specially if I have increments. Its like put leaves on everything.

We will add a default value for the date. I want the database to generate this information automatically.
the naming convention is very important. and then we repeat the same thing for all other tables.

```sql
CREATE TABLE silver.crm_cust_info (
    cst_id              INT,
    cst_key             NVARCHAR(50),
    cst_firstname       NVARCHAR(50),
    cst_lastname        NVARCHAR(50),
    cst_marital_status  NVARCHAR(50),
    cst_gndr            NVARCHAR(50),
    cst_create_date     DATE,
    dwh_create_date     DATETIME2 DEFAULT GETDATE()
);
GO
```



The commands have completed successfully. End time: 2025-10-13T12:46:42.4945016-03:00. There we created 6 tables.

Before writing any query, we need to assess the quality of the Bronze layer. First, we look at the quality issues, and then we start writing our code.
We'll review all the tables in the Bronze layer, clean them, and then insert them into the Silver layer.

-- Check for nulls or duplicates in Primary Key
-- Expectation: no result

To detect duplicates in the PK, we aggregate the primary key. If we find any value that appears more than once, it means it's not unique and we have duplicates in the primary table.
We're interested in the values that are greater than 1.

```sql
SELECT
[cst_id],
COUNT(*)
FROM [bronze].[crm_cust_info]
GROUP BY cst_id
HAVING COUNT (*) > 1
```

115 % No se encontraron problemas.

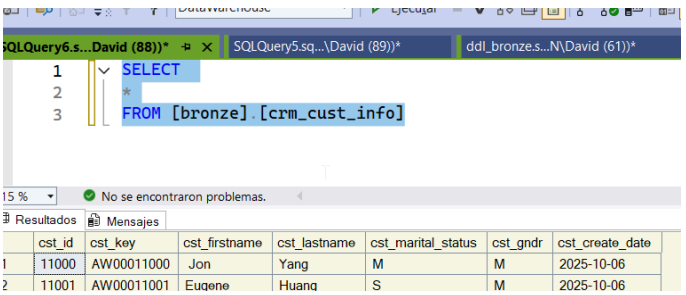
Resultados | Mensajes

	cst_id	(Sin nombre de columna)
1	29449	2
2	29473	2
3	29433	2
4	NULL	3
5	29483	2
6	29466	3

SQLQuery7.sq...no conectado | SQLQuery6.s...no conectado* | SQLQuery5.s...o cor

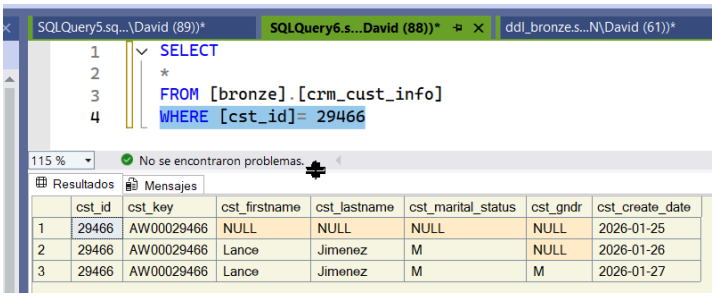
```sql
-- Check for nulls or duplicates in Primary Key
--Expectation: no result

SELECT
[cst_id],
COUNT(*)
FROM [bronze].[crm_cust_info]
GROUP BY cst_id
HAVING COUNT (*) > 1 OR cst_id IS NULL
```

We have the following issues in this table because this indicates that we have duplicates. Those IDs appear more than once in the table.
We'll start by writing the query that will perform the data transformation and data cleaning.



```
SELECT
*
FROM [bronze].[crm_cust_info]
```

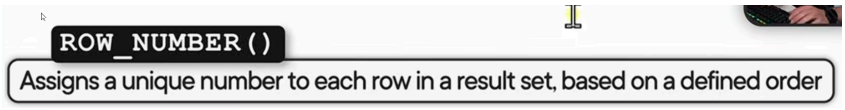
cst_id	cst_key	cst_firstname	cst_lastname	cst_marital_status	cst_gndr	cst_create_date
11000	AW00011000	Jon	Yang	M	M	2025-10-06
11001	AW00011001	Eugene	Huang	S	M	2025-10-06



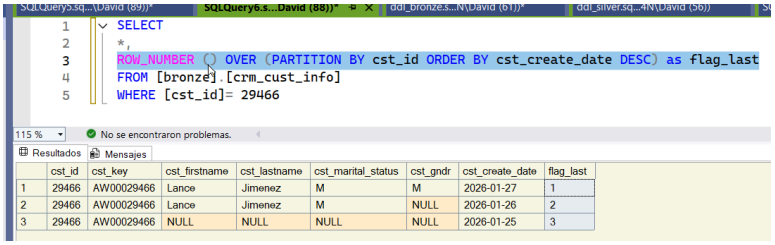
```
SELECT
*
FROM [bronze].[crm_cust_info]
WHERE [cst_id]= 29466
```

cst_id	cst_key	cst_firstname	cst_lastname	cst_marital_status	cst_gndr	cst_create_date
29466	AW00029466	NULL	NULL	NULL	NULL	2026-01-25
29466	AW00029466	Lance	Jimenez	M	NULL	2026-01-26
29466	AW00029466	Lance	Jimenez	M	M	2026-01-27

I focus on the issues. The ID is repeated 3 times.
Which one do we choose? It's better to choose the newest one because it has the most up-to-date information.



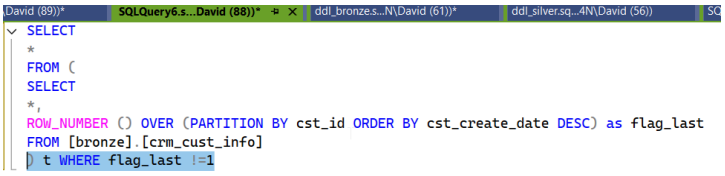
dividimos por customer id



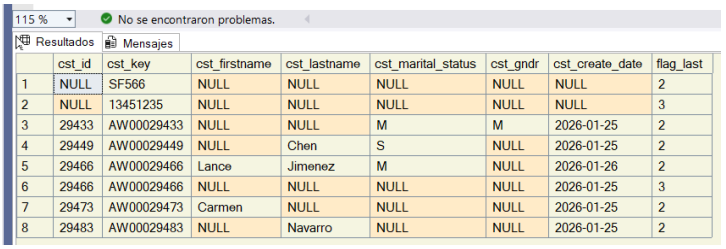
```
SELECT
*,
ROW_NUMBER () OVER (PARTITION BY cst_id ORDER BY cst_create_date DESC) as flag_last
FROM [bronze].[crm_cust_info]
WHERE [cst_id]= 29466
```

cst_id	cst_key	cst_firstname	cst_lastname	cst_marital_status	cst_gndr	cst_create_date	flag_last
29466	AW00029466	Lance	Jimenez	M	M	2026-01-27	1
29466	AW00029466	Lance	Jimenez	M	NULL	2026-01-26	2
29466	AW00029466	NULL	NULL	NULL	NULL	2026-01-25	3

The data is sorted by the creation date.



```
SELECT
*
FROM (
SELECT
*,
ROW_NUMBER () OVER (PARTITION BY cst_id ORDER BY cst_create_date DESC) as flag_last
FROM [bronze].[crm_cust_info]
) t WHERE flag_last !=1
```



cst_id	cst_key	cst_firstname	cst_lastname	cst_marital_status	cst_gndr	cst_create_date	flag_last
NULL	SF566	NULL	NULL	NULL	NULL	NULL	2
NULL	13451235	NULL	NULL	NULL	NULL	NULL	3
29433	AW00029433	NULL	NULL	M	M	2026-01-25	2
29449	AW00029449	NULL	Chen	S	NULL	2026-01-25	2
29466	AW00029466	Lance	Jimenez	M	NULL	2026-01-26	2
29466	AW00029466	NULL	NULL	NULL	NULL	2026-01-25	3
29473	AW00029473	Carmen	NULL	NULL	NULL	2026-01-25	2
29483	AW00029483	NULL	Navarro	NULL	NULL	2026-01-25	2

Here we have all the data that we don't need.

```sql
SELECT
*
FROM (
SELECT
*,
ROW_NUMBER () OVER (PARTITION BY cst_id ORDER BY cst_create_date DESC) as flag_last
FROM [bronze].[crm_cust_info]
WHERE [cst_id] IS NOT NULL
) t WHERE flag_last  = 1
```

cst_id	cst_key	cst_firstname	cst_lastname	cst_marital_status	cst_gndr	cst_create_date	flag_last
11000	AW00011000	Jon	Yang	M	M	2025-10-06	1
11001	AW00011001	Eugene	Huang	S	M	2025-10-06	1

```sql
ROW_NUMBER () OVER (PARTITION BY cst_id ORDER BY cst_create_dat
FROM [bronze].[crm_cust_info]
) t WHERE flag_last  = 1 AND cst_id = 29466
```

cst_id	cst_key	cst_firstname	cst_lastname	cst_marital_status	cst_gndr	cst_create_date	flag_last
29466	AW00029466	Lance	Jimenez	M	M	2026-01-27	1

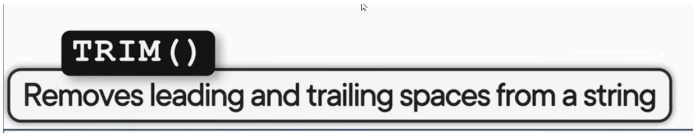
```sql
SELECT
*
FROM (
SELECT
*,
ROW_NUMBER () OVER (PARTITION BY cst_id ORDER BY cst_create_date DESC) as flag_last
FROM [bronze].[crm_cust_info]
WHERE [cst_id] IS NOT NULL
) t WHERE flag_last  = 1
```

If we do it this way, we ensure that our PK doesn't repeat and that each value appears only once. This query removes any duplicates.

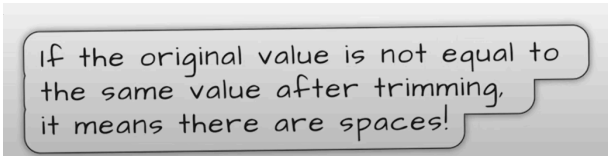

```sql
-- Check for unwanted Spaces
SELECT cst_firstname
FROM bronze.crm_cust_info
```

	cst_firstname
1	Jon
2	Eugene
3	Ruben

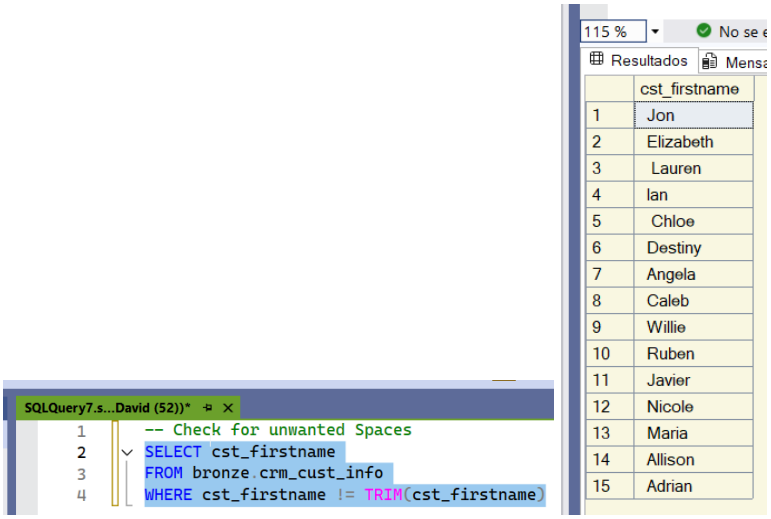
Let's check for empty spaces. We apply a filter.



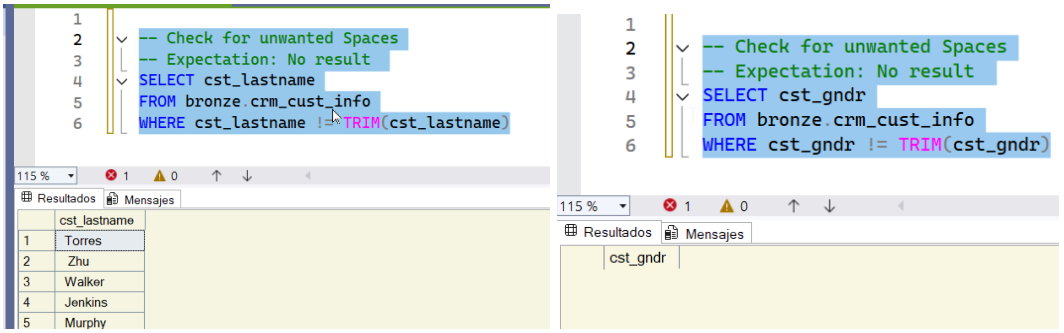
The `TRIM` function removes all empty spaces.



If the first value is not equal to the second value after applying `TRIM`, then we have a problem.
The first name is not equal to the first name after trimming the values.
We can also check the last name.



This is the list of all names where we have extra spaces.
We perform the same check for the last names. We can check all string values in the table.



We didn't get any results, which means the quality is better.
Now we're going to perform a transformation to clean these two columns.

```sql
--Limpiamos espacios innecesarios

SELECT
    [cst_id],
    [cst_key],
    [cst_firstname],
    [cst_lastname],
    [cst_marital_status],
    [cst_gndr],
    [cst_create_date]

FROM (
    SELECT
    *,
    ROW_NUMBER () OVER (PARTITION BY cst_id ORDER BY cst_create_date DESC) as flag_last
    FROM [bronze].[crm_cust_info]
    WHERE [cst_id] IS NOT NULL
) t WHERE flag_last  = 1
```

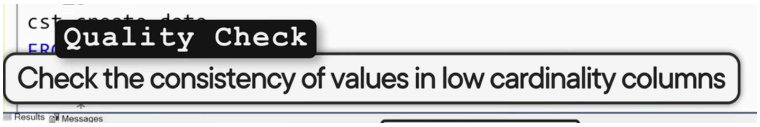
```sql
--Limpiamos espacios innecesarios

SELECT
    [cst_id],
    [cst_key],
    TRIM ([cst_firstname]) AS cst_firstname,
    TRIM( [cst_lastname]) AS cst_lastname,
    [cst_marital_status],
    [cst_gndr],
    [cst_create_date]
FROM (
    SELECT
    *,
    ROW_NUMBER () OVER (PARTITION BY cst_id ORDER BY cst_cr
    FROM [bronze].[crm_cust_info]
    WHERE [cst_id] IS NOT NULL
) t WHERE flag_last  = 1
```

115 % — No se encontraron problemas.

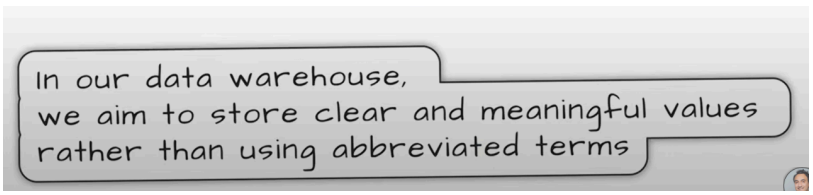
Resultados | Mensajes

	cst_id	cst_key	cst_firstname	cst_lastname	cst_marital_status	cst_gndr	cst_create_date
1	11000	AW00011000	Jon	Yang	M	M	2025-10-06
2	11001	AW00011001	Eugene	Huang	S	M	2025-10-06
3	11002	AW00011002	Ruben	Torres	M	M	2025-10-06

With this, we've already cleaned unnecessary spaces.



Now we check the quality of the `marital_status` and `gndr` columns.

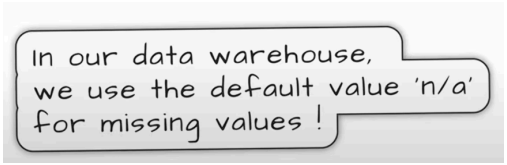



```sql
-- Data Standarization & Consistency
SELECT DISTINCT cst_gndr
FROM bronze.crm_cust_info
```

Resultados | Mensajes

	cst_gndr
1	NULL
2	F
3	M

We're going to change the values `F` and `M` to `Female` and `Male`.
We also need to check for `NULLs`.




```sql
    TRIM( [cst_lastname]) AS cst_lastname,
    [cst_marital_status],
    CASE WHEN UPPER (cst_gndr) = 'F' THEN 'Female'
         WHEN UPPER (cst_gndr) = 'M' THEN 'Male'
         ELSE 'n/a'
    END [cst_gndr],
    [cst_create_date]
```

We'll convert them to UPPER just in case. We use the UPPER function to handle any values that might appear in lowercase.

```sql
  [cst_marital_status],
  CASE WHEN UPPER (cst_gndr) = 'F' THEN 'Female'
       WHEN UPPER (cst_gndr) = 'M' THEN 'Male'
       ELSE 'n/a'
  END [cst_gndr],
```

Something else you can add—if you don't trust the data and want to avoid extra spaces—is to include the `TRIM` function.

```
7   TRIM ([cst_firstname]) AS cst_firstname,
8   TRIM( [cst_lastname]) AS cst_lastname,
9   [cst_marital_status],
10  CASE WHEN UPPER ( TRIM(cst_gndr)) = 'F' THEN 'Female'
11       WHEN UPPER (TRIM(cst_gndr)) = 'M' THEN 'Male'
12       ELSE 'n/a'
13  END [cst_gndr],
14  [cst_create_date]
```

115 % — No se encontraron problemas.

Resultados | Mensajes

	cst_id	cst_key	cst_firstname	cst_lastname	cst_marital_status	cst_gndr	cst_create_date
1	11000	AW00011000	Jon	Yang	M	Male	2025-10-06
2	11001	AW00011001	Eugene	Huang	S	Male	2025-10-06
3	11002	AW00011002	Ruben	Torres	M	Male	2025-10-06
4	11003	AW00011003	Christy	Zhu	S	Female	2025-10-06

```
14  [cst_create_date]
```

115 % — No se encontraron problemas.

Resultados | Mensajes

	cst_id	cst_key	cst_firstname	cst_lastname	cst_marital_status	cst_gndr	cst_create_date
157	11156	AW00011156	Maria	Roberts	S	Female	2025-10-07
158	11157	AW00011157	Hannah	Long	M	Female	2025-10-07
159	11158	AW00011158	Jason	Wright	S	Male	2025-10-07
160	11159	AW00011159	Brianna	Hughes	S	Female	2025-10-07
161	11160	AW00011160	Maurice	Tang	S	n/a	2025-10-07
162	11161	AW00011161	Emily	Wood	S	n/a	2025-10-07
163	11162	AW00011162	Chase	Cox	S	n/a	2025-10-07
164	11163	AW00011163	Gabriel	Wang	M	n/a	2025-10-07

If we don't get any value, it will show `N/A`.

Now we'll do the same for `marital_status`.

```
6   WHERE cst_key != TRIM(cst_key)
7
8   -- Data Standarization & Consistency
9   SELECT DISTINCT [cst_marital_status]
10  FROM bronze.crm_cust_info
11
```

115 % — No se encontraron problemas.

Resultados | Mensajes

	cst_marital_status
1	S
2	NULL
3	M

```
TRIM ([cst_firstname]) AS cst_firstname,
TRIM( [cst_lastname]) AS cst_lastname,
CASE WHEN UPPER ( TRIM([cst_marital_status])) = 'S' THEN 'Single'
     WHEN UPPER (TRIM([cst_marital_status])) = 'M' THEN 'Married'
     ELSE 'n/a'
END [cst_marital_status],
```

```
6   [cst_key],
7   TRIM ([cst_firstname]) AS cst_firstname,
8   TRIM( [cst_lastname]) AS cst_lastname,
9   CASE WHEN UPPER ( TRIM([cst_marital_status])) = 'S' THEN 'Single'
10       WHEN UPPER (TRIM([cst_marital_status])) = 'M' THEN 'Married'
11       ELSE 'n/a'
12  END [cst_marital_status],
13  CASE WHEN UPPER ( TRIM(cst_gndr)) = 'F' THEN 'Female'
```

115 % — 1 — 0

Resultados | Mensajes

	cst_id	cst_key	cst_firstname	cst_lastname	cst_marital_status	cst_gndr	cst_create_date
1	11000	AW00011000	Jon	Yang	Married	Male	2025-10-06
2	11001	AW00011001	Eugene	Huang	Single	Male	2025-10-06

Now, instead of showing `S` or `M`, it shows `Single` or `Married`.

Next, we'll work on the `create_date`.

Now we're going to write the `INSERT` statement.

```
3
4
5   INSERT INTO silver.crm_cust_info (
6       [cst_id],
7       [cst_key],
8       [cst_firstname],
9       [cst_lastname],
10      [cst_marital_status],
11      [cst_gndr],
12      [cst_create_date])
13
14  SELECT
15      [cst_id],
16      [cst_key],
```

(18,484 rows affected)

End time: 2025-10-13T22:09:31.0686826-03:00

With this, we have inserted the clean data into the Silver table.


```sql
-- Check for nulls or duplicates in Primary Key
--Expectation: no result

SELECT
[cst_id],
COUNT(*)
FROM silver.[crm_cust_info]
GROUP BY cst_id
HAVING COUNT (*) > 1 OR cst_id IS NULL
```

cst_id	(Sin nombre de columna)

```sql
-- Check for unwanted Spaces
-- Expectation: No result
SELECT [cst_firstname]
FROM silver.crm_cust_info
WHERE [cst_firstname] != TRIM([cst_firstname])
```

cst_firstname

We don’t have any duplicates in the PK.

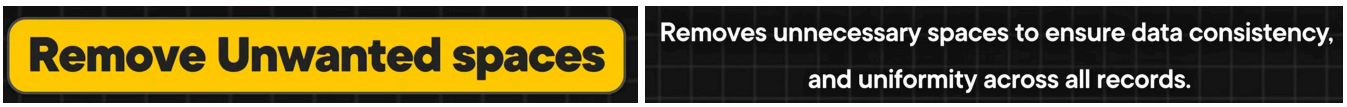
```sql
-- Data Standarization & Consistency
SELECT DISTINCT cst_gndr
FROM silver.crm_cust_info
```

	cst_gndr
1	n/a
2	Male
3	Female

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

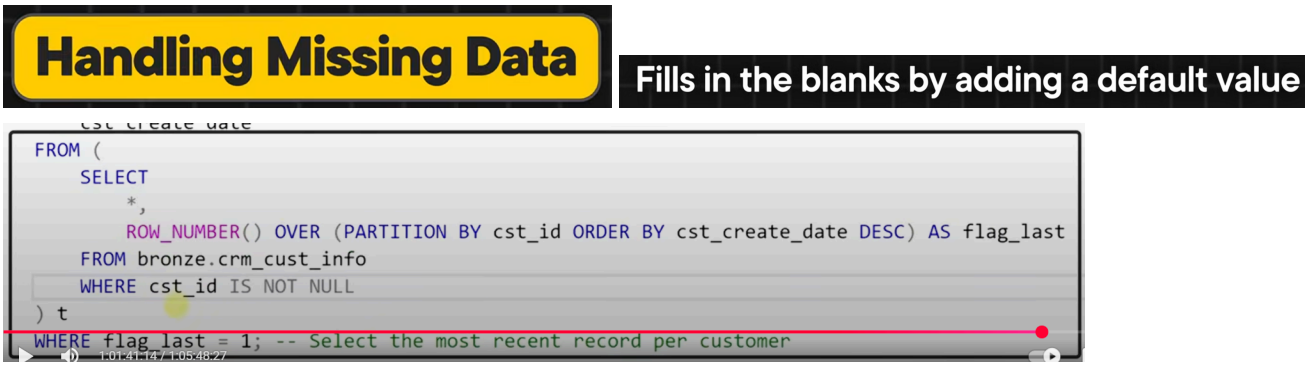
	cst_id	cst_key	cst_firstname	cst_lastname	cst_marital_status	cst_gndr	cst_create_date	dwh_create_date
1	11000	AW00011000	Jon	Yang	Married	Male	2025-10-06	2025-10-13 22:09:30.9466667
2	11001	AW00011001	Eugene	Huang	Single	Male	2025-10-06	2025-10-13 22:09:30.9466667
3	11002	AW00011002	Ruben	Torres	Married	Male	2025-10-06	2025-10-13 22:09:30.9466667
4	11003	AW00011003	Christy	Zhu	Single	Female	2025-10-06	2025-10-13 22:09:30.9466667

Everything if fine.

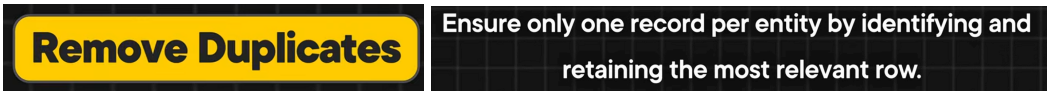


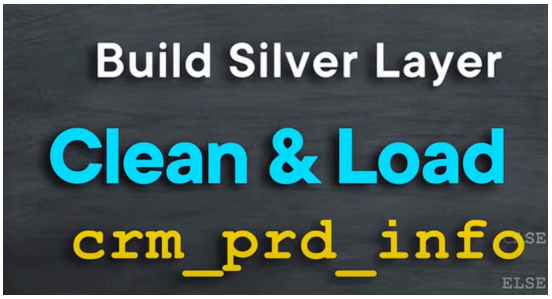

```sql
cst_key,
TRIM(cst_firstname) AS cst_firstname,
TRIM(cst_lastname) AS cst_lastname,
CASE
```

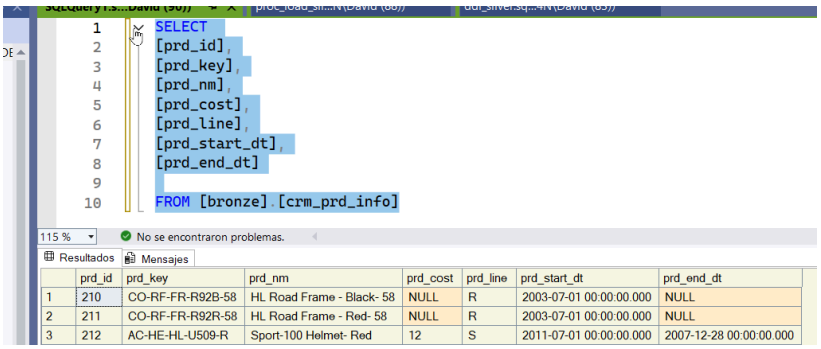

```sql
TRIM(cst_lastname) AS cst_lastname,
CASE
    WHEN UPPER(TRIM(cst_marital_status)) = 'S' THEN 'Single'
    WHEN UPPER(TRIM(cst_marital_status)) = 'M' THEN 'Married'
    ELSE 'n/a'
END AS cst_marital_status, -- Normalize marital status values to
CASE
    WHEN UPPER(TRIM(cst_gndr)) = 'F' THEN 'Female'
```



remove the duplicates







```
SELECT
[prd_id],
[prd_key],
[prd_nm],
[prd_cost],
[prd_line],
[prd_start_dt],
[prd_end_dt]

FROM [bronze].[crm_prd_info]
```

115 % — No se encontraron problemas.

Resultados | Mensajes

	prd_id	prd_key	prd_nm	prd_cost	prd_line	prd_start_dt	prd_end_dt
1	210	CO-RF-FR-R92B-58	HL Road Frame - Black- 58	NULL	R	2003-07-01 00:00:00.000	NULL
2	211	CO-RF-FR-R92R-58	HL Road Frame - Red- 58	NULL	R	2003-07-01 00:00:00.000	NULL
3	212	AC-HE-HL-U509-R	Sport-100 Helmet- Red	12	S	2011-07-01 00:00:00.000	2007-12-28 00:00:00.000

We start with the second table. The first thing we need to check is the PK.

```sql
--Check For Nulls or Duplicates in Primary KEy
--Expectatio: No result
SELECT
[prd_id],
COUNT(*)
FROM [bronze].[crm_prd_info]
GROUP BY [prd_id]
HAVING COUNT(*) > 1 OR [prd_id] IS NULL
```

No se encontraron problemas.

Resultados | Mensajes

prd_id	(Sin nombre de columna)

Perfect, the PK doesn't repeat.

Now we continue with the `prd_key`; we need to split the string into two pieces of information.

We use the SUBSTRING function to extract part of the string.

First, we specify the column from which we want to extract information. Then we define the starting position. Finally, we specify the length.

```sql
SELECT
[prd_id],
[prd_key],
SUBSTRING (prd_key, 1, 5 ) AS cat_id,
[prd_nm],
[prd_cost],
[prd_line],
[prd_start_dt],
[prd_end_dt]
FROM [bronze].[crm_prd_info]
```

No se encontraron problemas.

Resultados | Mensajes

	prd_id	prd_key	cat_id	prd_nm	prd_
1	210	CO-RF-FR-R92B-58	CO-RF	HL Road Frame - Black- 58	NUL
2	211	CO-RF-FR-R92R-58	CO-RF	HL Road Frame - Red- 58	NUL
3	212	AC-HE-HL-U509-R	AC-HE	Sport-100 Helmet- Red	12
4	213	AC-HE-HL-U509-R	AC-HE	Sport-100 Helmet- Red	14
5	214	AC-HE-HL-U509-R	AC-HE	Sport-100 Helmet- Red	13

```sql
FROM [bronze].[crm_prd_info]

SELECT distinct id FROM bronze.[erp_px_cat_g1v2]
```

No se encontraron problemas.

Resultados | Mensajes

	id
1	AC_BC
2	AC_BR
3	AC_BS
4	AC_CL
5	AC_FE
6	AC_HE

We have a new column called `cat_id` that contains the first part of the string.

Let's do a double check to see if we can join the data. We'll check if we can join both tables.

We have a problem: one uses an underscore and the other a hyphen.

We're going to use the REPLACE function.

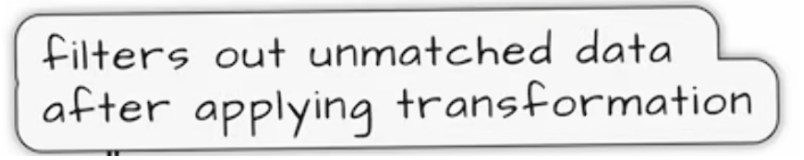
```sql
SELECT
[prd_id],
[prd_key],
REPLACE (SUBSTRING (prd_key, 1, 5 ), '-', '_') AS cat_id,
[prd_nm],
[prd_cost],
[prd_line],
[prd_start_dt],
[prd_end_dt]
FROM [bronze].[crm_prd_info]

SELECT distinct id FROM bronze.[erp_px_cat_g1v2]
```

No se encontraron problemas.

Resultados | Mensajes

	prd_id	prd_key	cat_id	prd_nm	prd_cost	prd_line	prd_start_dt
1	210	CO-RF-FR-R92B-58	CO_RF	HL Road Frame - Black- 58	NULL	R	2003-07-01 00
2	211	CO-RF-FR-R92R-58	CO_RF	HL Road Frame - Red- 58	NULL	R	2003-07-01 00
3	212	AC-HE-HL-U509-R	AC_HE	Sport-100 Helmet- Red	12	S	2011-07-01 00
4	213	AC-HE-HL-U509-R	AC_HE	Sport-100 Helmet- Red	14	S	2012-07-01 00

Perfect, we replaced the hyphen with an underscore.



Now we want to check that there isn't a matching value in the second table.

```sql
[prd_start_dt],
[prd_end_dt]
FROM [bronze].[crm_prd_info]
WHERE REPLACE (SUBSTRING (prd_key, 1, 5 ), '-', '_') NOT IN
    (SELECT distinct id FROM bronze.[erp_px_cat_g1v2])
```

No se encontraron problemas.

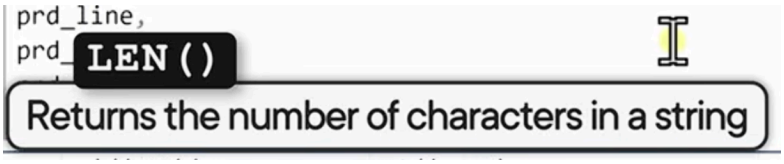
Resultados | Mensajes

	prd_id	prd_key	cat_id	prd_nm	prd_cost	prd_line	prd_start_dt	prd_end_dt
1	542	CO-PE-PD-M282	CO_PE	LL Mountain Pedal	18	M	2013-07-01 00:00:00.000	NULL
2	543	CO-PE-PD-M340	CO_PE	ML Mountain Pedal	28	M	2013-07-01 00:00:00.000	NULL
3	544	CO-PE-PD-M562	CO_PE	HL Mountain Pedal	36	M	2013-07-01 00:00:00.000	NULL
4	545	CO-PE-PD-R347	CO_PE	LL Road Pedal	18	R	2013-07-01 00:00:00.000	NULL
5	546	CO-PE-PD-R563	CO_PE	ML Road Pedal	28	R	2013-07-01 00:00:00.000	NULL

We only found one category that isn't in the second table. Our check is fine.

We already have the first part; now we'll do the same for the second part.

This time, we don't start at position 1 but at position 7. To make it dynamic and capture the full length, we use the LEN function to get all the characters each item has without losing information.



```sql
[prd_id],
[prd_key],
REPLACE (SUBSTRING (prd_key, 1, 5 ), '-', '_') AS cat_id,
SUBSTRING (prd_key, 7, LEN(prd_key) ) AS prd_key,
[prd_nm],
[prd_cost],
[prd_line],
[prd_start_dt],
[prd_end_dt]
FROM [bronze].[crm_prd_info]
```

115 % — No se encontraron problemas.

Resultados | Mensajes

	prd_id	prd_key	cat_id	prd_key	prd_nm	prd_cost	prd_line
1	210	CO-RF-FR-R92B-58	CO_RF	FR-R92B-58	HL Road Frame - Black- 58	NULL	R
2	211	CO-RF-FR-R92R-58	CO_RF	FR-R92R-58	HL Road Frame - Red- 58	NULL	R
3	212	AC-HE-HL-U509-R	AC_HE	HL-U509-R	Sport-100 Helmet- Red	12	S
4	213	AC-HE-HL-U509-R	AC_HE	HL-U509-R	Sport-100 Helmet- Red	14	S
5	214	AC-HE-HL-U509-R	AC_HE	HL-U509-R	Sport-100 Helmet- Red	13	S
6	215	AC-HE-HL-U509	AC_HE	HL-U509	Sport-100 Helmet- Black	12	S
7	216	AC-HE-HL-U509	AC_HE	HL-U509	Sport-100 Helmet- Black	14	S

Now we extract the second part from the `prd_key` string.
We do this to join it with the `sales_details` table.

```sql
FROM [bronze].[crm_prd_info]

SELECT [sls_prd_key] FROM [bronze].[crm_sales_details]
```

115 % — No se encontraron problemas.

Resultados | Mensajes

	sls_prd_key
1	BK-R93R-62
2	BK-M82S-44
3	BK-M82S-44
4	BK-R50B-62
5	BK-M82S-44
6	BK-R93R-44
7	BK-R93R-62
8	BK-M82B-48
9	BK-M82S-38

We check the data in the table we want to join.
We have many products that don't have any orders.

```sql
[prd_cost],
[prd_line],
[prd_start_dt],
[prd_end_dt]
FROM [bronze].[crm_prd_info]
WHERE SUBSTRING (prd_key, 7, LEN(prd_key) ) NOT IN (
    SELECT [sls_prd_key] FROM [bronze].[crm_sales_details])
```

115 % — No se encontraron problemas. — Línea: 12

Resultados | Mensajes

	prd_id	prd_key	cat_id	prd_key	prd_nm	prd_cost	prd_line	prd_start_dt	prd_end_dt
1	210	CO-RF-FR-R92B-58	CO_RF	FR-R92B-58	HL Road Frame - Black- 58	NULL	R	2003-07-01 00:00:00.000	NULL
2	211	CO-RF-FR-R92R-58	CO_RF	FR-R92R-58	HL Road Frame - Red- 58	NULL	R	2003-07-01 00:00:00.000	NULL
3	218	CL-SO-SO-B909-M	CL_SO	SO-B909-M	Mountain Bike Socks- M	3	M	2011-07-01 00:00:00.000	2007-12-28 00:00:00.000
4	219	CL-SO-SO-B909-L	CL_SO	SO-B909-L	Mountain Bike Socks- L	3	M	2011-07-01 00:00:00.000	2007-12-28 00:00:00.000
5	238	CO-RF-FR-R92R-62	CO_RF	FR-R92R-62	HL Road Frame - Red- 62	748	R	2011-07-01 00:00:00.000	2007-12-28 00:00:00.000
6	239	CO-RF-FR-R92R-62	CO_RF	FR-R92R-62	HL Road Frame - Red- 62	722	R	2012-07-01 00:00:00.000	2008-12-27 00:00:00.000
7	240	CO-RF-FR-R92R-62	CO_RF	FR-R92R-62	HL Road Frame - Red- 62	869	R	2013-07-01 00:00:00.000	NULL

We don't have any orders for the product that starts with FK.

```
[prd_end_dt]
FROM [bronze].[crm_prd_info]
WHERE SUBSTRING (prd_key, 7, LEN(prd_key) ) NOT IN (
    SELECT [sls_prd_key] FROM [bronze].[crm_sales_details]
    WHERE [sls_prd_key] LIKE 'FK-1%')
```

115 % — No se encontraron problemas.

Resultados | Mensajes

sls_prd_key

```
[prd_end_dt]
FROM [bronze].[crm_prd_info]
WHERE SUBSTRING (prd_key, 7, LEN(prd_key) ) IN (
    SELECT [sls_prd_key] FROM [bronze].[crm_sales_details])
```

115 % — No se encontraron problemas. — Línea: 12

Resultados | Mensajes

	prd_id	prd_key	cat_id	prd_key	prd_nm	prd_cost	prd_line	prd_start_dt	prd_end_dt
1	212	AC-HE-HL-U509-R	AC_HE	HL-U509-R	Sport-100 Helmet- Red	12	S	2011-07-01 00:00:00.000	2007-12-28 00:00:00.000
2	213	AC-HE-HL-U509-R	AC_HE	HL-U509-R	Sport-100 Helmet- Red	14	S	2012-07-01 00:00:00.000	2008-12-27 00:00:00.000
3	214	AC-HE-HL-U509-R	AC_HE	HL-U509-R	Sport-100 Helmet- Red	13	S	2013-07-01 00:00:00.000	NULL

We need to remove the products that don't have any orders. Everything is fine. There are only products without orders, so we're good.

Next, we move on to `product_name`, which has unwanted spaces.

```
--Check for unwanted spaces
-- Expectation: No Results
SELECT [prd_nm]
FROM [bronze].[crm_prd_info]
WHERE prd_nm != TRIM(prd_nm)
```

115 % — No se encontraron problemas.

Resultados | Mensajes

prd_nm

```
--Check for NULLS or Negative Spaces
-- Expectation: No Results
SELECT [prd_cost]
FROM [bronze].[crm_prd_info]
WHERE [prd_cost] < 0 OR [prd_cost] IS NULL
```

115 % — No se encontraron problemas.

Resultados | Mensajes

	prd_cost
1	NULL
2	NULL

We'll check that there are no `NULLs` or negative numbers. There are no negative values, but we do have `NULLs`.

We can replace the `NULLs` with zeros.

`ISNULL` is used to replace `NULL` values with zeros.

If it's `NULL`, then replace that value with zero.

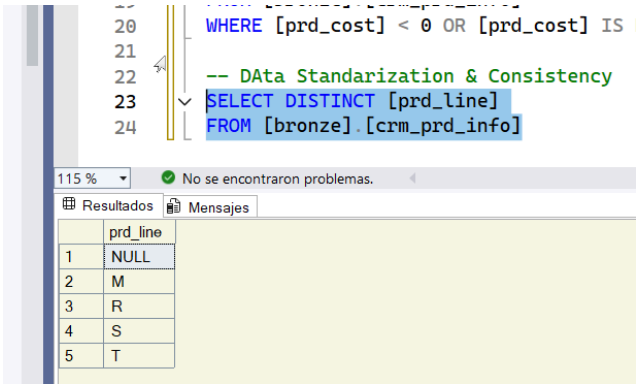
```sql
SELECT
[prd_id],
[prd_key],
REPLACE (SUBSTRING (prd_key, 1, 5 ), '-', '_') AS cat_id,
SUBSTRING (prd_key, 7, LEN(prd_key) ) AS prd_key,
[prd_nm],
ISNULL([prd_cost], 0) AS prd_cost,
[prd_line],
[prd_start_dt],
[prd_end_dt]
FROM [bronze].[crm_prd_info]
```

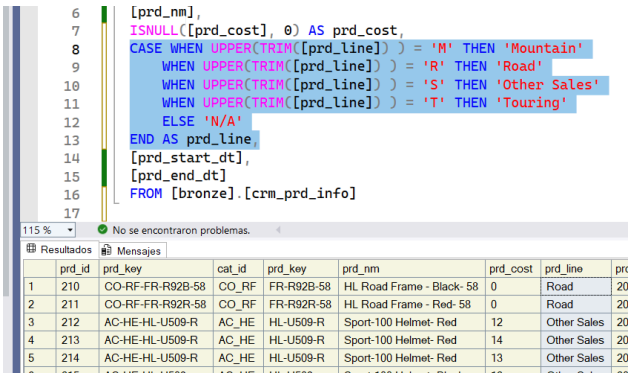
No se encontraron problemas.

Resultados | Mensajes

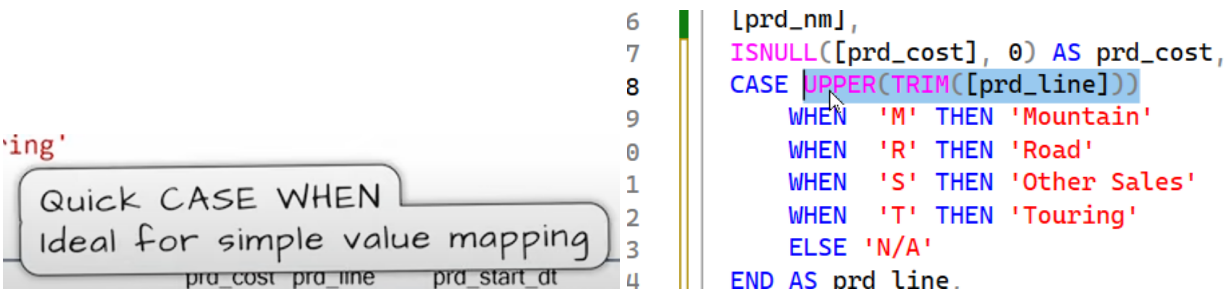
	prd_id	prd_key	cat_id	prd_key	prd_nm	prd_cost	prd_line
1	210	CO-RF-FR-R92B-58	CO_RF	FR-R92B-58	HL Road Frame - Black- 58	0	R
2	211	CO-RF-FR-R92R-58	CO_RF	FR-R92R-58	HL Road Frame - Red- 58	0	R
3	212	AC-HE-HL-U509-R	AC_HE	HL-U509-R	Sport-100 Helmet- Red	12	S
4	213	AC-HE-HL-U509-R	AC_HE	HL-U509-R	Sport-100 Helmet- Red	14	S

We no longer have `NULLs`; we have zeros. This will help us in the future when performing aggregates like `AVG`.





We continue with the next column, `prd_line`, and review all the options.
We transform all the values into more user-friendly labels.



We can write it this way to avoid repeating the same code all the time.
Next, we move on to the last two columns.

We have a problem because some end dates occur before the start dates.

```
-- Check for Invalidad Date Orders
SELECT *
FROM [bronze].[crm_prd_info]
WHERE [prd_end_dt]<[prd_start_dt]
```

115 % No se encontraron problemas.

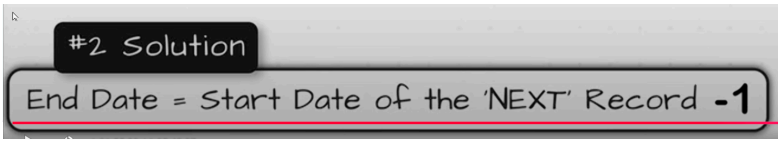
Resultados Mensajes

	prd_id	prd_key	prd_nm	prd_cost	prd_line	prd_start_dt	prd_end_dt
1	212	AC-HE-HL-U509-R	Sport-100 Helmet- Red	12	S	2011-07-01 00:00:00.000	2007-12-28 00:00:00.000
2	213	AC-HE-HL-U509-R	Sport-100 Helmet- Red	14	S	2012-07-01 00:00:00.000	2008-12-27 00:00:00.000
3	215	AC-HE-HL-U509	Sport-100 Helmet- Black	12	S	2011-07-01 00:00:00.000	2007-12-28 00:00:00.000

It should have returned a NULL value. We have problems.

	A	B	C	D	E	F	G
1	212	AC-HE-HL-U509-R	Sport-100 Helmet- Red	12	S	2011-07-01 00:00:00.000	2007-12-28 00:00:00.000
2	213	AC-HE-HL-U509-R	Sport-100 Helmet- Red	14	S	2012-07-01 00:00:00.000	2008-12-27 00:00:00.000
3	214	AC-HE-HL-U509-R	Sport-100 Helmet- Red	13	S	2013-07-01 00:00:00.000	NULL
4							
5	215	AC-HE-HL-U509	Sport-100 Helmet- Black	12	S	2011-07-01 00:00:00.000	2007-12-28 00:00:00.000
6	216	AC-HE-HL-U509	Sport-100 Helmet- Black	14	S	2012-07-01 00:00:00.000	2008-12-27 00:00:00.000
7	217	AC-HE-HL-U509	Sport-100 Helmet- Black	13	S	2013-07-01 00:00:00.000	NULL
8							

We take these values and do a brainstorming session.



212	AC-HE-HL-U509-R	Sport-100 Helmet- Red	12	S	2011-07-01 00:00:00.000	2012-06-30 00:00:00.000
213	AC-HE-HL-U509-R	Sport-100 Helmet- Red	14	S	2012-07-01 00:00:00.000	2013-06-30 00:00:00.000
214	AC-HE-HL-U509-R	Sport-100 Helmet- Red	13	S	2013-07-01 00:00:00.000	NULL

We set the end date equal to the start date of the other order. Then we subtract one day to avoid overlap.

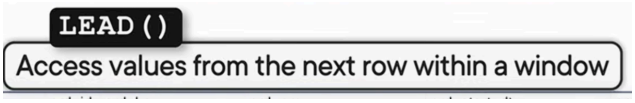
```sql
SELECT
[prd_id],
[prd_key],
[prd_nm],
[prd_cost],
[prd_line],
[prd_start_dt],
[prd_end_dt]
FROM [bronze].[crm_prd_info]
WHERE [prd_key] IN ('AC-HE-HL-U509-R', 'AC-HE-HL-U509')
```

115 % — No se encontraron problemas.

Resultados | Mensajes

	prd_id	prd_key	prd_nm	prd_cost	prd_line	prd_start_dt	prd_end_dt
1	212	AC-HE-HL-U509-R	Sport-100 Helmet- Red	12	S	2011-07-01 00:00:00.000	2007-12-28 00:00:00.000
2	213	AC-HE-HL-U509-R	Sport-100 Helmet- Red	14	S	2012-07-01 00:00:00.000	2008-12-27 00:00:00.000
3	214	AC-HE-HL-U509-R	Sport-100 Helmet- Red	13	S	2013-07-01 00:00:00.000	NULL
4	215	AC-HE-HL-U509	Sport-100 Helmet- Black	12	S	2011-07-01 00:00:00.000	2007-12-28 00:00:00.000
5	216	AC-HE-HL-U509	Sport-100 Helmet- Black	14	S	2012-07-01 00:00:00.000	2008-12-27 00:00:00.000
6	217	AC-HE-HL-U509	Sport-100 Helmet- Black	13	S	2013-07-01 00:00:00.000	NULL

We're going to work with those two values. Now we build our logic.



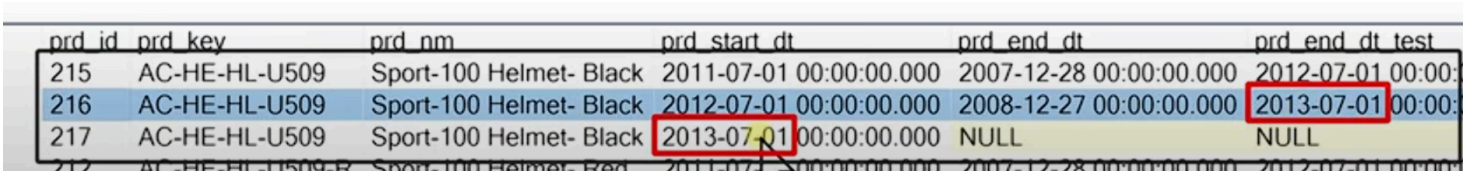
We're going to use the LEAD and LAG functions.
We'll apply the window function per product. We partition the data by `prd_key` and then need to order the data.

```sql
[prd_start_dt],
[prd_end_dt],
LEAD([prd_start_dt]) OVER ( PARTITION BY [prd_key] ORDER BY [prd_start_dt]) AS [prd_end_dt_test]
FROM [bronze].[crm_prd_info]
WHERE [prd_key] IN ('AC-HE-HL-U509-R', 'AC-HE-HL-U509')
```

105 % — No se encontraron problemas. — Línea: 9

Resultados | Mensajes

	prd_id	prd_key	prd_nm	prd_cost	prd_line	prd_start_dt	prd_end_dt	prd_end_dt_test
1	215	AC-HE-HL-U509	Sport-100 Helmet- Black	12	S	2011-07-01 00:00:00.000	2007-12-28 00:00:00.000	2012-07-01 00:00:00.000
2	216	AC-HE-HL-U509	Sport-100 Helmet- Black	14	S	2012-07-01 00:00:00.000	2008-12-27 00:00:00.000	2013-07-01 00:00:00.000
3	217	AC-HE-HL-U509	Sport-100 Helmet- Black	13	S	2013-07-01 00:00:00.000	NULL	NULL
4	212	AC-HE-HL-U509-R	Sport-100 Helmet- Red	12	S	2011-07-01 00:00:00.000	2007-12-28 00:00:00.000	2012-07-01 00:00:00.000
5	213	AC-HE-HL-U509-R	Sport-100 Helmet- Red	14	S	2012-07-01 00:00:00.000	2008-12-27 00:00:00.000	2013-07-01 00:00:00.000
6	214	AC-HE-HL-U509-R	Sport-100 Helmet- Red	13	S	2013-07-01 00:00:00.000	NULL	NULL



It's more organized now.
We made the end date come from the start date of the next item.

```
LEAD([prd_start_dt]) OVER ( PARTITION BY [prd_key] ORDER BY [prd_start_dt]) -1 AS [prd_end_dt_test]
FROM [bronze].[crm_prd_info]
WHERE [prd_key] IN ('AC-HE-HL-U509-R', 'AC-HE-HL-U509')
```

105 % — No se encontraron problemas. — Línea: 9

Resultados | Mensajes

	prd_id	prd_key	prd_nm	prd_cost	prd_line	prd_start_dt	prd_end_dt	prd_end_dt_test
1	215	AC-HE-HL-U509	Sport-100 Helmet- Black	12	S	2011-07-01 00:00:00.000	2007-12-28 00:00:00.000	2012-06-30 00:00:00.000
2	216	AC-HE-HL-U509	Sport-100 Helmet- Black	14	S	2012-07-01 00:00:00.000	2008-12-27 00:00:00.000	2013-06-30 00:00:00.000
3	217	AC-HE-HL-U509	Sport-100 Helmet- Black	13	S	2013-07-01 00:00:00.000	NULL	NULL
4	212	AC-HE-HL-U509-R	Sport-100 Helmet- Red	12	S	2011-07-01 00:00:00.000	2007-12-28 00:00:00.000	2012-06-30 00:00:00.000
5	213	AC-HE-HL-U509-R	Sport-100 Helmet- Red	14	S	2012-07-01 00:00:00.000	2008-12-27 00:00:00.000	2013-06-30 00:00:00.000
6	214	AC-HE-HL-U509-R	Sport-100 Helmet- Red	13	S	2013-07-01 00:00:00.000	NULL	NULL

Then we subtract 1, and now it works.
Now we replace what we wrote in our main query.

```
SELECT
[prd_id],
[prd_key],
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 UPPER(TRIM([prd_line]))
    WHEN  'M' THEN 'Mountain'
    WHEN  'R' THEN 'Road'
    WHEN  'S' THEN 'Other Sales'
    WHEN  'T' THEN 'Touring'
    ELSE 'N/A'
END AS prd_line,
[prd_start_dt],
LEAD([prd_start_dt]) OVER ( PARTITION BY [prd_key] ORDER BY [prd_start_dt]) -1 AS [prd_end_dt]
FROM [bronze].[crm_prd_info]
```

105 % — No se encontraron problemas. — Línea: 16

Resultados | Mensajes

	prd_id	prd_key	cat_id	prd_key	prd_nm	prd_cost	prd_line	prd_start_dt	prd_end_dt
1	478	AC-BC-BC-M005	AC_BC	BC-M005	Mountain Bottle Cage	4	Mountain	2013-07-01 00:00:00.000	NULL
2	479	AC-BC-BC-R205	AC_BC	BC-R205	Road Bottle Cage	3	Road	2013-07-01 00:00:00.000	NULL
3	477	AC-BC-WB-H098	AC_BC	WB-H098	Water Bottle - 30 oz.	2	Other Sales	2013-07-01 00:00:00.000	NULL

We don’t like the zeros in the `start_date`.

```
    ELSE 'N/A'
END AS prd_line,
CAST ([prd_start_dt] AS DATE) AS [prd_start_dt] ,
CAST (LEAD([prd_start_dt]) OVER ( PARTITION BY [prd_key]
      ORDER BY [prd_start_dt]) -1 AS DATE) AS [prd_end_dt]
FROM [bronze].[crm_prd_info]
```

105 % — No se encontraron problemas.

Resultados | Mensajes

	prd_id	prd_key	cat_id	prd_key	prd_nm	prd_cost	prd_line	prd_start_dt	prd_end_dt
1	478	AC-BC-BC-M005	AC_BC	BC-M005	Mountain Bottle Cage	4	Mountain	2013-07-01	NULL
2	479	AC-BC-BC-R205	AC_BC	BC-R205	Road Bottle Cage	3	Road	2013-07-01	NULL
3	477	AC-BC-WB-H098	AC_BC	WB-H098	Water Bottle - 30 oz.	2	Other Sales	2013-07-01	NULL
4	483	AC-BR-RA-H123	AC_BR	RA-H123	Hitch Rack - 4-Bike	45	Other Sales	2013-07-01	NULL
5	486	AC-BS-ST-1401	AC_BS	ST-1401	All-Purpose Bike Stand	59	Mountain	2013-07-01	NULL
6	484	AC-CL-CL-9009	AC_CL	CL-9009	Bike Wash - Dissolver	3	Other Sales	2013-07-01	NULL
7	485	AC-FE-FE-6654	AC_FE	FE-6654	Fender Set - Mountain	8	Mountain	2013-07-01	NULL
8	215	AC-HE-HL-U509	AC_HE	HL-U509	Sport-100 Helmet- Black	12	Other Sales	2011-07-01	2012-06-30
9	216	AC-HE-HL-U509	AC_HE	HL-U509	Sport-100 Helmet- Black	14	Other Sales	2012-07-01	2013-06-30

Now it looks much better. We now have clean product information.

```sql
CREATE TABLE [silver].[crm_prd_info](
    [prd_id]     INT,
    [prd_key]    NVARCHAR(50),
    [prd_nm]     NVARCHAR(50),
    [prd_cost]   INT,
    [prd_line]   NVARCHAR(50),
    [prd_start_dt]   DATETIME,
    [prd_end_dt]     DATETIME,
    [dwh_create_date]   DATETIME2 DEFAULT GETDATE()
    );

SELECT
```

Everything is fine now, but before creating the tables, we need to do one more step.

```sql
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,
    cat_id              NVARCHAR(50),
    [prd_key]           NVARCHAR(50),
    [prd_nm]            NVARCHAR(50),
    [prd_cost]          INT,
    [prd_line]          NVARCHAR(50),
    [prd_start_dt]      DATE,
    [prd_end_dt]        DATE,
    [dwh_create_date]   DATETIME2 DEFAULT GETDATE()
);
```

105 % No se encontraron problemas.

Mensajes

```
Los comandos se han completado correctamente.

Hora de finalización: 2025-10-14T13:09:20.8569775-03:00
```

We create our table with the correct formats. Sometimes we need to adjust the metadata if the quality isn't good. It's similar to the Bronze layer but with some modifications.
Now we insert the clean data into the Silver table.

```sql
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
```

We inserted the data into the Silver layer.

(397 rows affected)

End time: 2025-10-14T13:19:23.8391306-03:00


```sql
--Check For Nulls or Duplicates in Primary KEy
--Expectatio: No result
SELECT
[prd_id],
COUNT(*)
FROM [silver].[crm_prd_info]
GROUP BY [prd_id]
HAVING COUNT(*) > 1 OR [prd_id] IS NULL

--Check for unwanted spaces
-- Expectation: No Results
```

105 % — No se encontraron problemas.

Resultados | Mensajes

prd_id	(Sin nombre de columna)

```sql
--Check for unwanted spaces
-- Expectation: No Results
SELECT [prd_nm]
FROM [silver].[crm_prd_info]
WHERE prd_nm != TRIM(prd_nm)
```

105 % — No se encontraron problemas.

Resultados | Mensajes

prd_nm

1. We check the PK.

```sql
WHERE prd_nm != TRIM(prd_nm)

--Check for NULLS or Negative Spaces
-- Expectation: No Results
SELECT [prd_cost]
FROM [silver].[crm_prd_info]
WHERE [prd_cost] < 0 OR [prd_cost] IS NULL
```

105 % — No se encontraron problemas.

Resultados | Mensajes

prd_cost

```sql
-- DAta Standarization & Consistency
SELECT DISTINCT [prd_line]
FROM [silver].[crm_prd_info]
```

105 % — No se encontraron problemas.

Resultados | Mensajes

	prd_line
1	Mountain
2	N/A
3	Other Sales
4	Road
5	Touring

```sql
-- Check for Invalidad Date Orders
SELECT *
FROM [silver].[crm_prd_info]
WHERE [prd_end_dt] < [prd_start_dt]
```

105 % — No se encontraron problemas.

Resultados | Mensajes

prd_id	cat_id	prd_key	prd_nm	prd_cost	prd_line	prd_start_dt	prd_end_dt

```sql
WHERE [prd_end_dt] < [prd_start_dt]

-- Final Look
SELECT *
FROM [silver].[crm_prd_info]
```

105 % — No se encontraron problemas.

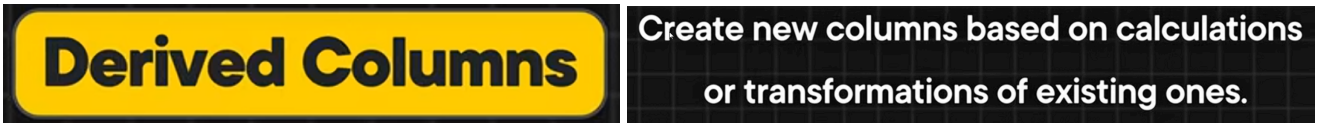
Resultados | Mensajes

	prd_id	cat_id	prd_key	prd_nm	prd_cost	prd_line	prd_start_dt	prd_end_dt	dwh_create_date
1	478	AC_BC	BC-M005	Mountain Bottle Cage	4	Mountain	2013-07-01	NULL	2025-10-14 13:28:49.2833333
2	479	AC_BC	BC-R205	Road Bottle Cage	3	Road	2013-07-01	NULL	2025-10-14 13:28:49.2833333
3	477	AC_BC	WB-H098	Water Bottle - 30 oz.	2	Other Sales	2013-07-01	NULL	2025-10-14 13:28:49.2833333
4	483	AC_BR	RA-H123	Hitch Rack - 4-Bike	45	Other Sales	2013-07-01	NULL	2025-10-14 13:28:49.2833333
5	486	AC_BS	ST-1401	All-Purpose Bike Stand	59	Mountain	2013-07-01	NULL	2025-10-14 13:28:49.2833333
6	484	AC_CL	CL-9009	Bike Wash - Dissolver	3	Other Sales	2013-07-01	NULL	2025-10-14 13:28:49.2833333
7	485	AC_FE	FE-6654	Fender Set - Mountain	8	Mountain	2013-07-01	NULL	2025-10-14 13:28:49.2833333

The last column is automatically generated by the DDL, indicating when we loaded this table.

```sql
SELECT
    [prd_id],
    REPLACE (SUBSTRING (prd_key, 1, 5 ), '-', '_') AS cat_id, -- Extract category ID
    SUBSTRING (prd_key, 7, LEN(prd_key) ) AS prd_key,         -- Extract product key
    [prd_nm],
    ISNULL([prd_cost], 0) AS prd_cost,
    CASE UPPER(TRIM([prd_line]))
        WHEN  'M' THEN 'Mountain'
        WHEN  'R' THEN 'Road'
        WHEN  'S' THEN 'Other Sales'
        WHEN  'T' THEN 'Touring'
        ELSE 'N/A'
    END AS prd_line,                                          -- Map product line codes to descriptive values
    CAST ([prd_start_dt] AS DATE) AS [prd_start_dt] ,
    CAST (
            LEAD([prd_start_dt]) OVER ( PARTITION BY [prd_key]
            ORDER BY [prd_start_dt]) -1
            AS DATE) AS [prd_end_dt]                          -- Calculate end date as one day before the next start date.
FROM [bronze].[crm_prd_info]
```

We performed the following transformations.




```sql
prd_nm,
ISNULL(prd_cost, 0) AS prd_cost,
CASE
```

We handled missing information.

```sql
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, -- Map product line codes to descriptive values
```

data normalization

```sql
END AS prd_line, -- Map product line codes to
CAST(prd_start_dt AS DATE) AS prd_start_dt,
CAST(
```

data type casting. We convert one data tye to another

```sql
CAST(
    LEAD(prd_start_dt) OVER (PARTITION BY prd_key ORDER BY prd_start_dt) - 1
    AS DATE
) AS prd_end_dt -- Calculate end date as one day before the next start date
```