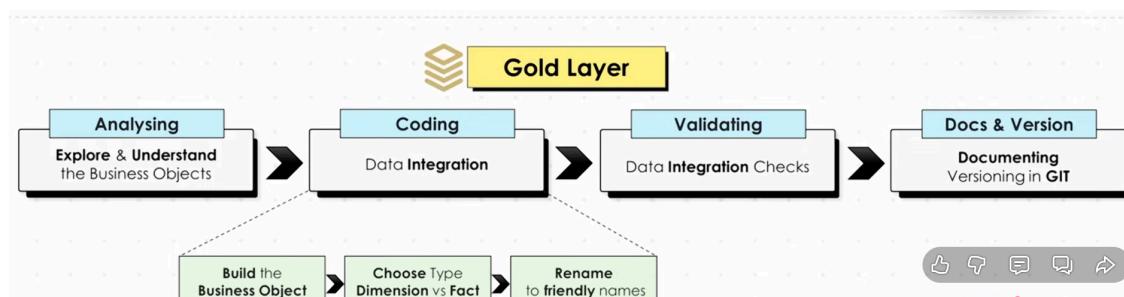
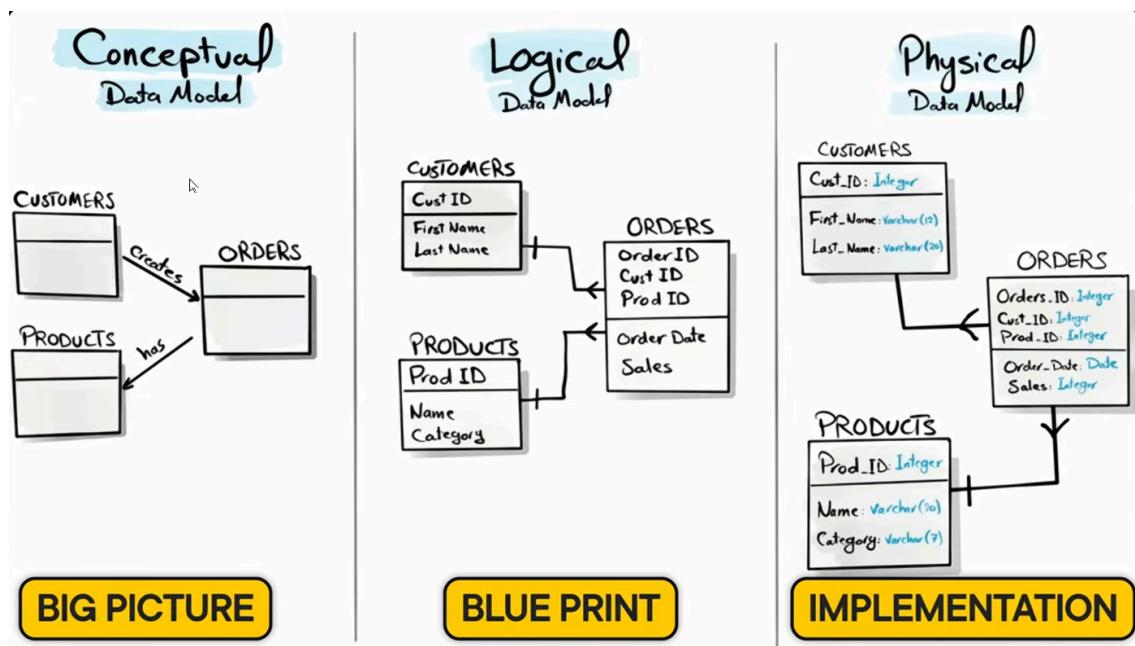


Building GOLD LAYER



Theory

What is Data Modeling?



nosotros vamos a hacer el logical data model

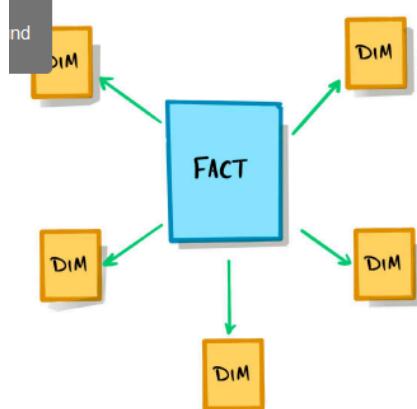
```
M(Sales) Sales  
Sales.Orders  
ProductID
```

Theory

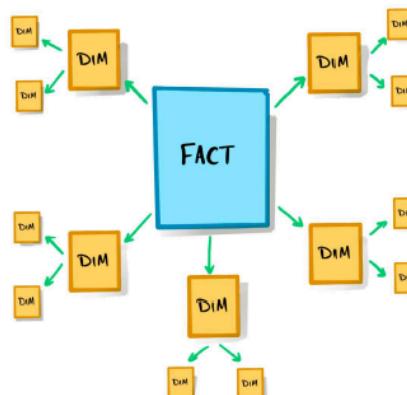
```
ProductId,  
Pro  
Cat  
e.Sales
```

Star Schema vs. Snowflake Schema

STAR SCHEMA



Snowflake SCHEMA



We are going to use star schema

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

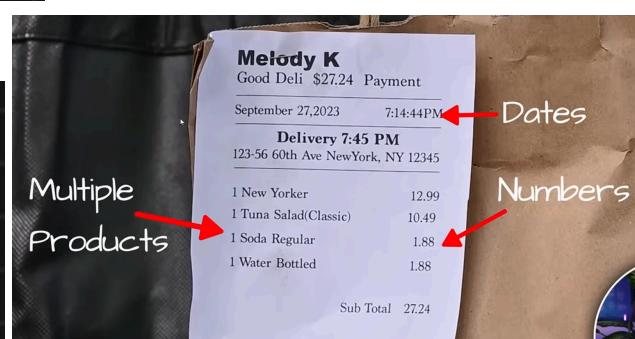
Descriptive information
that give context to your data.



Product Details = Dimension

Who? What? Where?

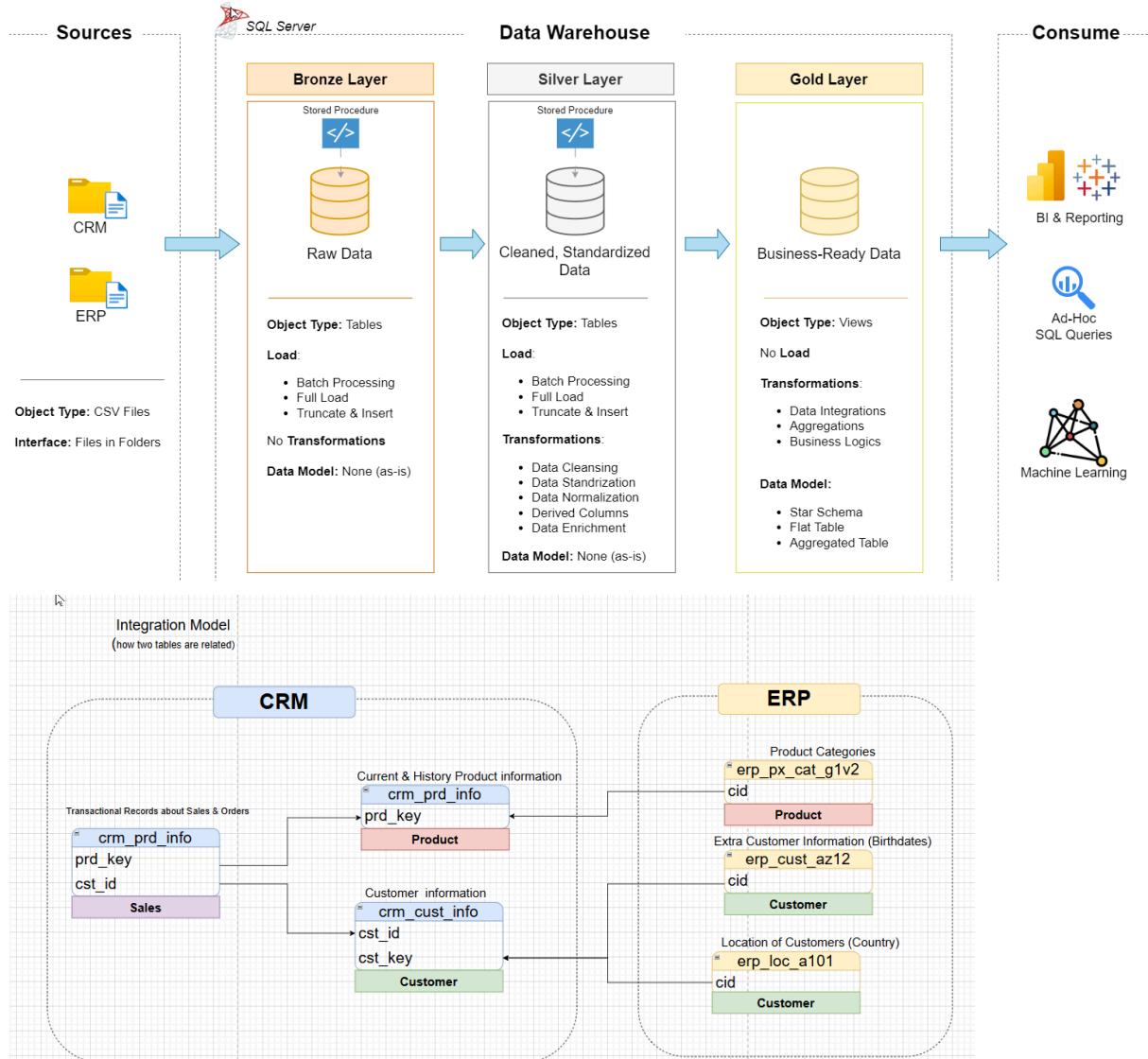
FACT
Quantitative information
that represents events



How much? How many?

Build Gold Layer

Explore the Business Objects



Let's start with the customer table

| | | |
|------------------|-------------------------------------|--|
| Build Gold Layer | Coding: Data Integration | <input type="checkbox"/> |
| Build Gold Layer | Validating: Data Integration Checks | <input type="checkbox"/> OPEN <input type="checkbox"/> |



Now we're going to collect data from the Silver Layer. The silver is already clean.

A screenshot of SQL Server Management Studio (SSMS) showing a query window titled "SQLQuery5.s...David (67)*". The query is:

```

1  SELECT *
2  FROM [silver].[crm_cust_info]

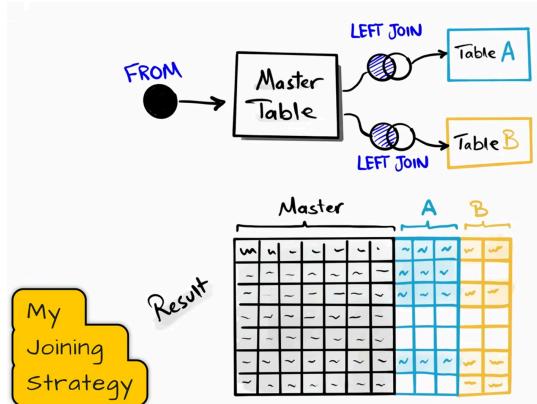
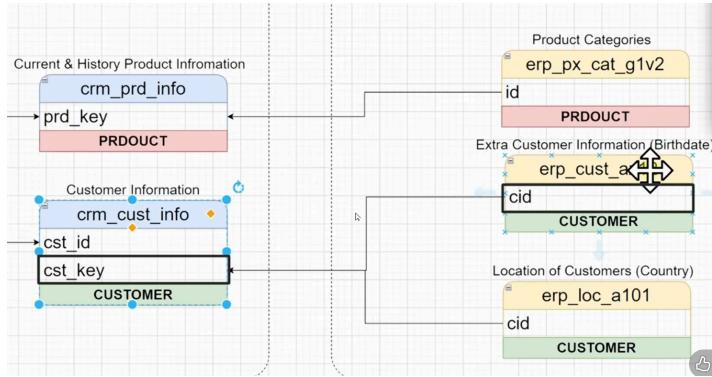
```

Below the query window is a results grid titled "Resultados". The data is as follows:

| | cst_id | cst_key | cst_firstname | cst_lastname | cst_marital_status | cst_gndr | cst_create_date | dw |
|---|--------|------------|---------------|--------------|--------------------|----------|-----------------|----|
| 1 | 11000 | AW00011000 | Jon | Yang | Married | Male | 2025-10-06 | 20 |
| 2 | 11001 | AW00011001 | Eugene | Huang | Single | Male | 2025-10-06 | 20 |
| 3 | 11002 | AW00011002 | Ruben | Torres | Married | Male | 2025-10-06 | 20 |
| 4 | 11003 | AW00011003 | Christy | Zhu | Single | Female | 2025-10-06 | 20 |
| 5 | 11004 | AW00011004 | Frank | Wong | Single | Male | 2025-10-06 | 20 |

Let's choose the columns that will be displayed in the golden layer.

I'll give the table an alias.



We need to join the data with the CID and the customer key. Avoid the inner join.

SQLQuery2.sql...David (97)* + X

```

1  SELECT
2      ci.[cst_id],
3      ci.[cst_key],
4      ci.[cst_firstname],
5      ci.[cst_lastname],
6      ci.[cst_marital_status],
7      ci.[cst_gndr],
8      [cst_create_date],
9      ca.bdate,
10     ca.gen
11  FROM [DataWarehouse].[silver].[crm_cust_info] ci
12    LEFT JOIN silver.erp_cust_az12 ca
13    ON ci.cst_key = ca.cid

```

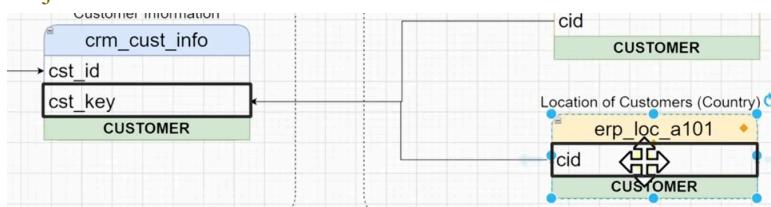
104 % No se encontraron problemas.

Resultados Mensajes

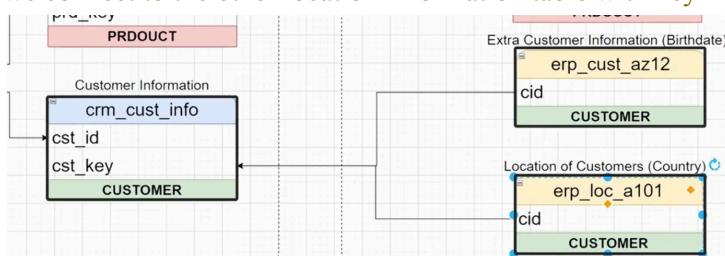
| cst_id | cst_key | cst_firstname | cst_lastname | cst_marital_status | cst_gndr | cst_create_date | bdate | gen |
|--------|------------|---------------|--------------|--------------------|----------|-----------------|------------|--------|
| 11000 | AW00011000 | Jon | Yang | Married | Male | 2025-10-06 | 1971-10-06 | Male |
| 11001 | AW00011001 | Eugene | Huang | Single | Male | 2025-10-06 | 1976-05-10 | Male |
| 11002 | AW00011002 | Ruben | Torres | Married | Male | 2025-10-06 | 1971-02-09 | Male |
| 11003 | AW00011003 | Christy | Zhu | Single | Female | 2025-10-06 | 1972-08-14 | Female |

bronze.erp_px_cat_g1v2
silver.crm_cust_info
silver.crm_prd_info
silver.crm_sales_details
silver.erp_cust_az12

we join these two tables.



we connect to the other location information table with key



The screenshot shows the SQL Server Management Studio interface. On the left, the Object Explorer displays several database objects under the 'silver' schema, including tables like 'crm_cust_info', 'erp_cust_az12', 'sales', and 'erp_loc_a101'. In the center, a query window titled 'SQLQuery2.s...David (97)*' contains the following T-SQL code:

```

1  SELECT
2      ci.[cst_id],
3      ci.[cst_key],
4      ci.[cst_firstname],
5      ci.[cst_lastname],
6      ci.[cst_marital_status],
7      ci.[cst_gndr],
8      [cst_create_date],
9      ca.bdate,
10     ca.gen,
11     la.[cntry]
12  FROM [DataWarehouse].[silver].[crm_cust_info] ci
13  LEFT JOIN silver.erp_cust_az12 ca
14  ON ci.cst_key = ca.cid
15  LEFT JOIN silver.erp_loc_a101 la
16  ON ci.cst_key = la.cid

```

The results grid shows three rows of data:

| cst_id | cst_key | cst_firstname | cst_lastname | cst_marital_status | cst_gndr | cst_create_date | bdate | gen | cntry | |
|--------|---------|---------------|--------------|--------------------|----------|-----------------|------------|------------|-------|-----------|
| 1 | 11000 | AW00011000 | Jon | Yang | Married | Male | 2025-10-06 | 1971-10-06 | Male | Australia |
| 2 | 11001 | AW00011001 | Eugene | Huang | Single | Male | 2025-10-06 | 1976-05-10 | Male | Australia |
| 3 | 11002 | AW00011002 | Ruben | Torres | Married | Male | 2025-10-06 | 1971-02-09 | Male | Australia |

TIP *Key*

After Joining table, check if any duplicates were introduced by the join logic

Let's check if there are duplicates in the query using group by

The screenshot shows the SQL Server Management Studio interface. A new 'Results' tab is visible at the bottom. The query window now includes a 'GROUP BY' clause:

```

1  SELECT [cst_id], COUNT(*) FROM (
2
3      SELECT
4          ci.[cst_id],
5          ci.[cst_key],
6          ci.[cst_firstname],
7          ci.[cst_lastname],
8          ci.[cst_marital_status],
9          ci.[cst_gndr],
10         [cst_create_date],
11         ca.bdate,
12         ca.gen,
13         la.[cntry]
14     FROM [DataWarehouse].[silver].[crm_cust_info] ci
15     LEFT JOIN silver.erp_cust_az12 ca
16     ON ci.cst_key = ca.cid
17     LEFT JOIN silver.erp_loc_a101 la
18     ON ci.cst_key = la.cid
19 )t GROUP BY [cst_id]
20 HAVING COUNT(*) > 1
21

```

The results grid shows one row:

| cst_id | (Sin nombre de columna) |
|--------|-------------------------|
| | |

This query is used to see if we have duplicates in the PK.

This means that after joining all these tables, there is no problem.

We have a problem because we have two sources of gender information. One comes from `ci.cst_gndr` and the other from `ca.gen`

The screenshot shows a table with columns: cst_gndr, cst_create_date, bdate, gen, and cn. A cursor is over the 'gen' column. To the right is a query results window titled 'SQLQuery1 u...David (97)*'. The query is:

```

SELECT DISTINCT
    ci.[cst_gndr],
    ca.gen
FROM [DataWarehouse].[silver].[crm_cust_info] ci
LEFT JOIN silver.erp_cust_az12 ca
ON ci.cst_key = ca.cid
LEFT JOIN silver.erp_loc_az01 la
ON ci.cst_key = la.cid
ORDER BY 1,2

```

The results show 10 rows of gender pairs, with row 7 highlighted in yellow.

| cst_gndr | gen |
|----------|--------|
| Male | Female |
| Female | Male |
| Female | n/a |
| Male | Female |
| Female | Male |
| Male | n/a |
| Female | Female |
| Male | Male |
| Male | n/a |
| Female | n/a |

We need to do data integration. We'll remove all the information and leave those two columns. We'll leave only DISTINCT.

We have all the scenarios.

NULLs often come from joined tables!
NULL will appear if SQL finds no match

| | cst_gndr | gen |
|----|----------|--------|
| 1 | Female | Female |
| 2 | Female | Male |
| 3 | Female | n/a |
| 4 | Male | Female |
| 5 | Male | Male |
| 6 | Male | n/a |
| 7 | n/a | NULL |
| 8 | n/a | Female |
| 9 | n/a | Male |
| 10 | n/a | n/a |

Which source is the master for these values?

The NULL comes from joining tables.

That means there's a customer table and a CRM table that aren't available in the other tables.

The Master Source of Customer Data is CRM!

| | cst_gndr | gen |
|----|----------|--------|
| 1 | Female | Female |
| 2 | Female | Male |
| 3 | Female | n/a |
| 4 | Male | Female |
| 5 | Male | Male |
| 6 | Male | n/a |
| 7 | n/a | NULL |
| 8 | n/a | Female |
| 9 | n/a | Male |
| 10 | n/a | n/a |

this source system is the master. -- CRM is the Master for gender info

```

SQLQuery1 U...David (97)* SQLQuery2 U...David (88)* SQLQuery3 U...David (76)* SQLQuery4 U...David (97)*
1  SELECT DISTINCT
2      ci.[cst_gndr],
3      ca.gen,
4      CASE WHEN ci.cst_gndr != 'N/A' THEN ci.cst_gndr
5          ELSE COALESCE(ca.gen, 'N/A')
6      END AS new_gen
7  FROM [DataWarehouse].[silver].[crm_cust_info] ci
8  LEFT JOIN silver.erp_cust_az12 ca
9  ON ci.cst_key = ca.cid
10 LEFT JOIN silver.erp_loc_a101 la
11 ON ci.cst_key = la.cid
12 ORDER BY 1,2

```

No se encontraron problemas.

| | cst_gndr | gen | new_gen |
|----|----------|--------|---------|
| 1 | Female | Female | Female |
| 2 | Female | Male | Female |
| 3 | n/a | Female | Female |
| 4 | Male | Female | Male |
| 5 | Male | Male | Male |
| 6 | Male | n/a | Male |
| 7 | n/a | NULL | N/A |
| 8 | n/a | Female | Female |
| 9 | n/a | Male | Male |
| 10 | n/a | n/a | n/a |

| | cst_gndr | gen | new_gen |
|----|----------|--------|---------|
| 1 | Female | Female | Female |
| 2 | Female | Male | Female |
| 3 | n/a | n/a | Female |
| 4 | Male | Female | Male |
| 5 | Male | Male | Male |
| 6 | Male | n/a | Male |
| 7 | n/a | NULL | N/A |
| 8 | n/a | Female | Female |
| 9 | n/a | Male | Male |
| 10 | n/a | n/a | n/a |

If we have data from the CRM system we have to use it is not equal this means its not available (this means we dont have a value female or male) then go and used it. Otherwise that means if its not available then go and use the information from the second table. We have to deal with the NULL we use the COALESCE if this is a null then N/A. So when we have a null it will say NA.

Since we do not have information in the first table, we use the second table. We have a beautiful new column where we integrate 2 source systems in one.

```

SQLQuery2 U...David (88)* SQLQuery3 U...David (76)* SQLQuery4 U...David (97)*
1  SELECT
2      ci.[cst_id],
3      ci.[cst_key],
4      ci.[cst_firstname],
5      ci.[cst_lastname],
6      ci.[cst_marital_status],
7      ci.[cst_gndr],
8      CASE WHEN ci.cst_gndr != 'N/A' THEN ci.cst_gndr -- CRM is the Master for gender info
9          ELSE COALESCE(ca.gen, 'N/A')
10     END AS new_gen,
11     ci.[cst_create_date],
12     ca.bdate,
13     la.[cntry]
14  FROM [DataWarehouse].[silver].[crm_cust_info] ci
15  LEFT JOIN silver.erp_cust_az12 ca
16  ON ci.cst_key = ca.cid
17  LEFT JOIN silver.erp_loc_a101 la
18  ON ci.cst_key = la.cid

```

No se encontraron problemas.

| cst_id | cst_key | cst_firstname | cst_lastname | cst_marital_status | new_gen | cst_create_date | bdate | cntry | |
|--------|---------|---------------|--------------|--------------------|---------|-----------------|------------|------------|-----------|
| 1 | 11000 | AW000011000 | Jon | Yang | Married | Male | 2025-10-06 | 1971-10-06 | Australia |
| | 11001 | AW000011001 | Eduardo | Urgos | Single | Male | 2025-10-06 | 1978-05-10 | Australia |

I joined those two tables with the when statement and then replaced it in the main query. We have no duplicates. We took three tables and put them in one object.

ABC Rename columns to friendly, meaningful names

Now we are going to give it friendly names.

General Principles

- Naming Conventions:** Use snake_case, with lowercase letters and underscores (_) to separate words.
- Language:** Use English for all names.
- Avoid Reserved Words:** Do not use SQL reserved words as object names.

Customer - customer

No se encontraron problemas.

| | customer_id | customer_number | first_name | last_name | marital_status | gender | create_date | birth_date | country |
|---|-------------|-----------------|------------|-----------|----------------|--------|-------------|------------|-----------|
| 1 | 11000 | AW00011000 | Jon | Yang | Married | Male | 2025-10-06 | 1971-10-06 | Australia |
| 2 | 11001 | AW00011001 | Eugene | Huang | Single | Male | 2025-10-06 | 1978-05-10 | Australia |

We changed the names to more friendly ones

Customer - country

Customer - cust_info

sort the columns into logical groups to improve readability

Dimension vs Fact ???

Surrogate Key

System-generated unique identifier
assigned to each record in a table.

we are going to use a windows function ROW

Customer - customer

No se encontraron problemas.

| | customer_key | customer_id | customer_number | first_name | last_name | country | marital_status |
|---|--------------|-------------|-----------------|------------|-----------|-----------|----------------|
| 1 | 1 | 11000 | AW00011000 | Jon | Yang | Australia | Married |
| 2 | 2 | 11001 | AW00011001 | Eugene | Huang | Australia | Single |
| 3 | 3 | 11002 | AW00011002 | Ruben | Torres | Australia | Married |
| 4 | 4 | 11003 | AW00011003 | Christy | Zhu | Australia | Single |
| 5 | 5 | 11004 | AW00011004 | Elizabeth | Johnson | Australia | Single |
| 6 | 6 | 11005 | AW00011005 | Julio | Ruiz | Australia | Single |

now we can use this key to connect the data model.

Now we need to create the object

SQLQuery3 cr...N\David (88) | SQLQuery3.sq...N\David (76) | SQLQuery1 un...N\David (97)

```
CREATE VIEW gold.dim_customers AS
SELECT
    ROW_NUMBER() OVER (ORDER BY cst_id) AS customer_key,
    ci.[cst_id] AS customer_id,
    ci.[cst_key] AS customer_number,
    ci.[cst_firstname] AS first_name,
    ci.[cst_lastname] AS last_name,
```

No se encontraron problemas.

Los comandos se han completado correctamente.

Hora de finalización: 2025-10-20T13:20:13.1064168-03:00

silver.erp_px_cat_g1v2

Vistas

Vistas del sistema

gold.dim_customers

Recursos externos

Síntesis

our first object

Quality Check of the Gold Table

key = ca.cid

SQLQuery4 cr...N\David (88) | SQLQuery1 un...N\David (97)* | SQLQuery3.s...N\David (76)*

```
1 | SELECT * FROM [gold].[dim_customers]
```

104 % 0 1 0 104 %

| | customer_key | customer_id | customer_number | first_name | last_name | country | marital_status | gender | birthdate | create_date |
|---|--------------|-------------|-----------------|------------|-----------|-----------|----------------|--------|------------|-------------|
| 1 | 1 | 11000 | AW00011000 | Jon | Yang | Australia | Married | Male | 1971-10-06 | 2025-10-06 |

SQLQuery1 un...N\David (51) Onwkiwhi.s...\\David

```

1  SELECT DISTINCT gender
2   FROM [gold].[dim_customers]

```

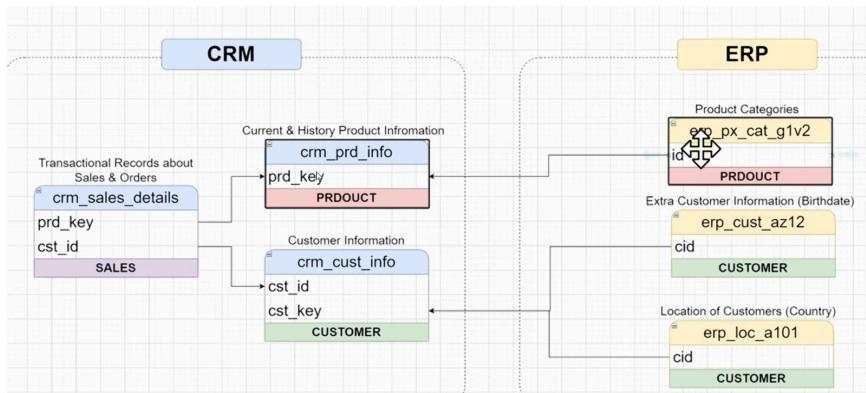
104 % No se encontraron problemas.

| | gender |
|---|--------|
| 1 | n/a |
| 2 | Male |
| 3 | Female |

Build Gold Layer

Create Dimension Products

Now we have to build the 2nd object



SQLQuery1.s...David (76)* SQLQuery1 un...N\David (51)

```

1  SELECT
2   pn.[prd_id],
3   pn.[cat_id],
4   pn.[prd_key],
5   pn.[prd_nm],
6   pn.[prd_cost],
7   pn.[prd_line],
8   pn.[prd_start_dt],
9   pn.[prd_end_dt]
10  FROM [silver].[crm_prd_info] pn

```

104 % No se encontraron problemas.

| prd_id | cat_id | prd_key | prd_nm | prd_cost | prd_line | prd_start_dt | prd_end_dt |
|--------|--------|---------|-----------------------|----------|-------------|--------------|------------|
| 1 478 | AC_BC | BC-M005 | Mountain Bottle Cage | 4 | Mountain | 2013-07-01 | NULL |
| 2 479 | AC_BC | BC-R205 | Road Bottle Cage | 3 | Road | 2013-07-01 | NULL |
| 3 477 | AC_BC | WB-H098 | Water Bottle - 30 oz. | 2 | Other Sales | 2013-07-01 | NULL |
| 4 482 | AC_BP | BL-U102 | Black Road & Bike | 45 | Other Sales | 2013-07-01 | NULL |

this obj contains historical info as well as current information. We are not using the primary key we are using the product key. We need to filter the historical data and stay with only current data.

If End Date is NULL then
It is Current Info of the Product !

if the end date is null then that means is the current data

| / | 400 | AC_HE | HL-U509 | Sport-100 Helmet- Black | 12 | Other Sales | 2011-07-01 | 2012-06-30 |
|----|-----|-------|---------|-------------------------|----|-------------|------------|------------|
| 8 | 215 | AC_HE | HL-U509 | Sport-100 Helmet- Black | 12 | Other Sales | 2011-07-01 | 2012-06-30 |
| 9 | 216 | AC_HE | HL-U509 | Sport-100 Helmet- Black | 14 | Other Sales | 2012-07-01 | 2013-06-30 |
| 10 | 217 | AC_HE | HL-U509 | Sport-100 Helmet- Black | 13 | Other Sales | 2013-07-01 | NULL |

we have the same pd but the first two we have a data but in the last we have null bcs is the current info. This means that it is open and its not closed yet. If we want only the current information its simple.

```

SQLQuery1.s...David (76)* SQLQuery1 un...N(David (51))
1  SELECT
2   pn.[prd_id],
3   pn.[cat_id],
4   pn.[prd_key],
5   pn.[prd_nm],
6   pn.[prd_cost],
7   pn.[prd_line],
8   pn.[prd_start_dt],
9   pn.[prd_end_dt]
10  FROM [silver].[crm_prd_info] pn
11  WHERE [prd_end_dt] IS NULL

```

104 % No se encontraron problemas.

| | prd_id | cat_id | prd_key | prd_nm | prd_cost | prd_line | prd_start_dt | prd_end_dt |
|---|--------|--------|---------|-----------------------|----------|-------------|--------------|------------|
| 1 | 478 | AC_BC | BC-M005 | Mountain Bottle Cage | 4 | Mountain | 2013-07-01 | NULL |
| 2 | 479 | AC_BC | BC-R205 | Road Bottle Cage | 3 | Road | 2013-07-01 | NULL |
| 3 | 477 | AC_BC | WB-H098 | Water Bottle - 30 oz. | 2 | Other Sales | 2013-07-01 | NULL |
| 4 | 483 | AC_BR | RA-H123 | Hitch Rack - 4-Bike | 45 | Other Sales | 2013-07-01 | NULL |

```

SQLQuery1.s...David (76)* SQLQuery1 un...N(David (51))
1  SELECT
2   pn.[prd_id],
3   pn.[cat_id],
4   pn.[prd_key],
5   pn.[prd_nm],
6   pn.[prd_cost],
7   pn.[prd_line],
8   pn.[prd_start_dt],
9   pn.[prd_end_dt]
10  FROM [silver].[crm_prd_info] pn
11  WHERE [prd_end_dt] IS NULL -- Filter out all historical data

```

104 % No se encontraron problemas.

| | prd_id | cat_id | prd_key | prd_nm | prd_cost | prd_line | prd_start_dt | prd_end_dt |
|---|--------|--------|---------|----------------------|----------|----------|--------------|------------|
| 1 | 478 | AC_BC | BC-M005 | Mountain Bottle Cage | 4 | Mountain | 2013-07-01 | NULL |
| 2 | 479 | AC_BC | BC-R205 | Road Bottle Cage | 3 | Road | 2013-07-01 | NULL |

you will only have the current product. Now we dont need the end date bcs its all null
now the next step is joined the the product categories. from the erp. We are going to use the id
you will only have the current product. Now we dont need the end date bcs its all null
now the next step is joined the the product categories. from the erp. We are going to use the id

```

SQLQuery1.s...David (76)* SQLQuery1 un...N(David (51))
1  SELECT
2   pn.[prd_id],
3   pn.[cat_id],
4   pn.[prd_key],
5   pn.[prd_nm],
6   pn.[prd_cost],
7   pn.[prd_line],
8   pn.[prd_start_dt],
9   pc.cat,
10  pc.[subcat],
11  pc.[maintenance]
12  FROM [silver].[crm_prd_info] pn
13  LEFT JOIN [silver].[erp_px_cat_g1v2] pc
14  ON pn.cat_id = pc.id
15  WHERE [prd_end_dt] IS NULL -- Filter out all historical data

```

104 % No se encontraron problemas.

| | prd_id | cat_id | prd_key | prd_nm | prd_cost | prd_line | prd_start_dt | cat | subcat | maintenance |
|---|--------|--------|---------|-----------------------|----------|-------------|--------------|-------------|-------------------|-------------|
| 1 | 478 | AC_BC | BC-M005 | Mountain Bottle Cage | 4 | Mountain | 2013-07-01 | Accessories | Bottles and Cages | No |
| 2 | 479 | AC_BC | BC-R205 | Road Bottle Cage | 3 | Road | 2013-07-01 | Accessories | Bottles and Cages | No |
| 3 | 477 | AC_BC | WB-H098 | Water Bottle - 30 oz. | 2 | Other Sales | 2013-07-01 | Accessories | Bottles and Cages | No |
| 4 | 483 | AC_BR | RA-H123 | Hitch Rack - 4-Bike | 45 | Other Sales | 2013-07-01 | Accessories | Bike Racks | Yes |

now we joined the tables

```

SQLQuery1.s...David (76)* SQLQuery1 un...N(David (51))
1  SELECT
2   pn.[prd_id],
3   pn.[cat_id],
4   pn.[prd_key],
5   pn.[prd_nm],
6   pn.[prd_cost],
7   pn.[prd_line],
8   pn.[prd_start_dt]

```

104 % No se encontraron problemas.

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

| cat | subcat | maintenance |
|-------------|-------------------|-------------|
| Accessories | Bottles and Cages | No |
| Accessories | Bottles and Cages | No |
| Accessories | Bottles and Cages | No |
| Accessories | Bike Racks | Yes |

first table and second table. Now we have collected all the product information from the 2 systems.
Now we need to check the quality of this results.

we want to make sure the product key is unique bcs we are going to use it later in order to join it with the sales.

```

1  SELECT [prd_key], COUNT(*) FROM (
2    SELECT
3      pn.[prd_id],
4      pn.[cat_id],
5      pn.[prd_key],
6      pn.[prd_nm],
7      pn.[prd_cost],
8      pn.[prd_line],
9      pn.[prd_start_dt],
10     pc.cat,
11     pc.[subcat],
12     pc.[maintenance]
13   FROM [silver].[crm_prd_info] pn
14   LEFT JOIN [silver].[erp_px_cat_g1v2] pc
15   ON pn.cat_id = pc.id
16   WHERE [prd_end_dt] IS NULL -- Filter out
17   old products
18   GROUP BY [prd_key]
19   HAVING COUNT(*)>1

```

104 % No se encontraron problemas.

Resultados Mensajes

prd_key (Sin nombre de columna)

We check that the PRD key is not duplicated. We don't have any duplicates.

 sort the columns into logical groups to improve readability

| prd_id | prd_key | prd_nm | cat_id | cat | subcat | maintenance | prd_cost | prd_line | prd_start_dt |
|--------|---------|-----------------------|--------|-------------|-------------------|-------------|----------|-------------|--------------|
| 478 | BC-M005 | Mountain Bottle Cage | AC_BC | Accessories | Bottles and Cages | No | 4 | Mountain | 2013-07-01 |
| 479 | BC-R205 | Road Bottle Cage | AC_BC | Accessories | Bottles and Cages | No | 3 | Road | 2013-07-01 |
| 477 | WB-H098 | Water Bottle - 30 oz. | AC_BC | Accessories | Bottles and Cages | No | 2 | Other Sales | 2013-07-01 |
| 483 | RA-H123 | Hitch Rack - 4-Rail | AC_BR | Accessories | Ride Racks | Yes | 15 | Other Sales | 2013-07-01 |

ABC Rename columns to friendly, meaningful names

because this is a dimension we need to create a PK we are going to use the w x fx row nm

```

1  SELECT
2    ROW_NUMBER () OVER (ORDER BY pn.[prd_start_dt], pn.[prd_key]) AS product_key,
3    pn.[prd_id] AS product_id,
4    pn.[prd_key] AS product_number,
5    pn.[prd_nm] AS product_name,
6    pn.[cat_id] AS category_id,
7    pc.cat AS category,
8    pc.[subcat] AS subcategory,
9    pc.[maintenance],
10   pn.[prd_cost] AS cost,
11   pn.[prd_line] AS product_line,
12   pn.[prd_start_dt] AS start_date
13
14
15   FROM [silver].[crm_prd_info] pn
16   LEFT JOIN [silver].[erp_px_cat_g1v2] pc
17   ON pn.cat_id = pc.id
18   WHERE [prd_end_dt] IS NULL -- Filter out all historical data

```

104 % No se encontraron problemas.

Resultados Mensajes

| product_key | product_id | product_number | product_name | category_id | category | subcategory | maintenance |
|-------------|------------|----------------|---------------------------|-------------|------------|----------------|-------------|
| 1 | 210 | FR-R92B-58 | HL Road Frame - Black- 58 | CO_RF | Components | Road Frames | Yes |
| 2 | 211 | FR-R92R-58 | HL Road Frame - Red- 58 | CO_RF | Components | Road Frames | Yes |
| 3 | 348 | BK-M92B-38 | Mountain-100 Black- 38 | BI_MB | Bikes | Mountain Bikes | Yes |
| 4 | 349 | BK-M92B-42 | Mountain-100 Black- 42 | BI_MB | Bikes | Mountain Bikes | Yes |

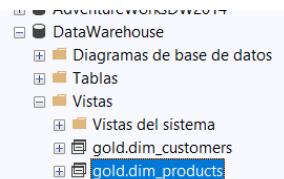
now we have generated it a pk for each product. And we are going to use it in order to connect to our data model,

Now latest build the view

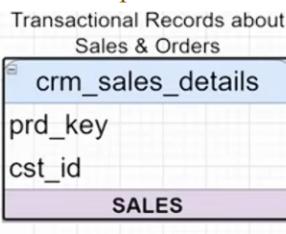
```

SQLQuery7 c...David (76)*  SQLQuery1 un...N(David (51))
1 |> CREATE VIEW gold.dim_products AS
2   SELECT
3     ROW_NUMBER() OVER (ORDER BY pn.[prd_start_dt],
4       pn.[prd_id] AS product_id,
5       pn.[columna_prd_id(int null)] number,
6       pn.[columna_prd_name] name,
7       pn.[cat_id] AS category_id,
8       pc.cat AS category,
9       pc.[subcategory],
10      pc.[maintenance],
11      pn.[prd_cost] AS cost,
12      pn.[prd_line] AS product_line,
13      pn.[prd_start_dt] AS start_date
14
15    FROM [silver].[crm_prd_info] pn
16    LEFT JOIN [silver].[erp_px_cat_glv2] pc
17      ON pn.cat_id = pc.id
18    WHERE [prd_end_dt] IS NULL -- Filter out all histor
19
104 %  No se encontraron problemas.
Mensajes  Los comandos se han completado correctamente.
Hora de finalización: 2025-10-20T14:04:30.9445452-03:00

```



We have created our object. Gold layer 2nd dimension products



```

SQLQuery3.s...David (65)*  SQL
1 |> SELECT
2   sd.[sls_ord_num],
3   sd.[sls_prd_key],
4   sd.[sls_cust_id],
5   sd.[sls_order_dt],
6   sd.[sls_ship_dt],
7   sd.[sls_due_dt],
8   sd.[sls_sales],
9   sd.[sls_quantity],
10  sd.[sls_price]
11  FROM [silver].[crm_sales_details] sd
104 %  No se encontraron problemas.

```

| | 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 |



| KEYS | | | DATES | | | MEASURES | | | Original IDs | | |
|-------------|-------------|-------------|--------------|-------------|------------|-----------|--------------|-----------|--------------|-------------|-----|
| sls_ord_num | sls_prd_key | sls_cust_id | sls_order_dt | sls_ship_dt | sls_due_dt | sls_sales | sls_quantity | sls_price | sls_prd_key | sls_cust_id | sls |
| SO43697 | BK-R93R-62 | 21768 | 2010-12-29 | 2011-01-05 | 2011-01-10 | 3578 | 1 | 3578 | BK-R93R-62 | 21768 | 20 |
| SO43698 | BK-M82S-44 | 28389 | 2010-12-29 | 2011-01-05 | 2011-01-10 | 3400 | 1 | 3400 | BK-M82S-44 | 28389 | 20 |
| SO43699 | BK-M82S-44 | 25863 | 2010-12-29 | 2011-01-05 | 2011-01-10 | 3400 | 1 | 3400 | BK-M82S-44 | 25863 | 20 |
| SO43700 | BK-R50B-62 | 14501 | 2010-12-29 | 2011-01-05 | 2011-01-10 | 699 | 1 | 699 | BK-R50B-62 | 14501 | 20 |

Building Fact

Use the dimension's surrogate keys instead of IDs
to easily connect facts with dimensions

we are going to join both dimensions in order to
the prd key and the customer id those information comes from the source system we are going to
replace those 2 information with the surrogate key that we have generated it.

DATA LOOK UP

in the silver layer we don't have any subrogue keys we have in the golden layer.
the only information that we need from the dimension is the key

SQLQuery3.s...David (65)*

```

1  SELECT
2    sd.[sls_ord_num],
3    pr.product_key,
4    cu.customer_key,
5    sd.[sls_order_dt],
6    sd.[sls_ship_dt] columnn sls_order_dt(date, null),
7    sd.[sls_due_dt],
8    sd.[sls_sales],
9    sd.[sls_quantity],
10   sd.[sls_price]
11  FROM [silver].[crm_sales_details] sd
12  LEFT JOIN gold.dim_products pr
13  ON sd.sls_prd_key = pr.product_number
14  LEFT JOIN gold.dim_customers cu
15  ON sd.[sls_cust_id] = cu.customer_id

```

Results

| sls_ord_num | product_key | customer_key | sls_order_dt | sls_ship_dt | sls_due_dt | sls_sales | sls_quantity | sls_price |
|-------------|-------------|--------------|--------------|-------------|------------|-----------|--------------|-----------|
| SO43697 | 20 | 10769 | 2010-12-29 | 2011-01-05 | 2011-01-10 | 3578 | 1 | 3578 |
| SO43698 | 9 | 17390 | 2010-12-29 | 2011-01-05 | 2011-01-10 | 3400 | 1 | 3400 |
| SO43699 | 9 | 14864 | 2010-12-29 | 2011-01-05 | 2011-01-10 | 3400 | 1 | 3400 |
| SO43700 | 41 | 3502 | 2010-12-29 | 2011-01-05 | 2011-01-10 | 699 | 1 | 699 |
| | | | | | | | | |

Dimension Keys

| product_key | customer_key | sls |
|-------------|--------------|-----|
| 20 | 10769 | 20 |
| 9 | 17390 | 20 |
| 9 | 14864 | 20 |
| 41 | 3502 | 20 |
| 9 | 4 | 20 |
| 16 | 16646 | 20 |

In this way we join the tables with the surrogate key that we created and delete the ones that come by default.

In our fact table we have the 2 keys from the dimension and this will help us to connect the data model to connect the fact with the dimensions. Now we give them friendly names

SQLQuery3.s...David (65)*

```

1  SELECT
2    sd.[sls_ord_num] AS order_number,
3    pr.product_key,
4    cu.customer_key,
5    sd.[sls_order_dt] AS order_date,
6    sd.[sls_ship_dt] AS shipping_date,
7    sd.[sls_due_dt] AS due_date,
8    sd.[sls_sales] AS sales_amount,
9    sd.[sls_quantity] AS quantity,
10   sd.[sls_price] price
11  FROM [silver].[crm_sales_details] sd
12  LEFT JOIN gold.dim_products pr
13  ON sd.sls_prd_key = pr.product_number
14  LEFT JOIN gold.dim_customers cu
15  ON sd.[sls_cust_id] = cu.customer_id

```

Results

| order_number | product_key | customer_key | order_date | shipping_date | due_date | sales_amount | quantity | price |
|--------------|-------------|--------------|------------|---------------|------------|--------------|----------|-------|
| SO43697 | 20 | 10769 | 2010-12-29 | 2011-01-05 | 2011-01-10 | 3578 | 1 | 3578 |
| SO43698 | 9 | 17390 | 2010-12-29 | 2011-01-05 | 2011-01-10 | 3400 | 1 | 3400 |

DIMENSION KEYS

| order_number | product_key | customer_key |
|--------------|-------------|--------------|
| SO43697 | 20 | 10769 |
| SO43698 | 9 | 17390 |

DATES

| order_date | shipping_date | due_date |
|------------|---------------|------------|
| 2010-12-29 | 2011-01-05 | 2011-01-10 |
| 2010-12-29 | 2011-01-05 | 2011-01-10 |

MEASURES

| sales_amount | quantity | price |
|--------------|----------|-------|
| 3578 | 1 | 3578 |
| 3400 | 1 | 3400 |

they have friendly names.

SQLQuery9 f...\\David (65)

```

1 CREATE VIEW gold.fact_sales AS
2   SELECT
3     sd.[sls_ord_num] AS order_number,
4     pr.product_key,
5     cu.customer_key,
6     sd.[sls_order_dt] AS order_date,
7     ...

```

Object Browser

- silver.erp_px_cat_g1v2
- Vistas
- Vistas del sistema
- gold.dim_customers
- gold.dim_products
- gold.fact_sales
- Recursos externos

now we have 3 objects in the gold layer. 2 dimension and 1 fact



SQLQuery4.s...David (69)*

```

1 SELECT * FROM [gold].[fact_sales]

```

Results

| order_number | product_key | customer_key | order_date | shipping_date | due_date | sales_amount | quantity | price |
|--------------|-------------|--------------|------------|---------------|------------|--------------|----------|-------|
| SO43697 | 20 | 10769 | 2010-12-29 | 2011-01-05 | 2011-01-10 | 3578 | 1 | 3578 |
| SO43698 | 9 | 17390 | 2010-12-29 | 2011-01-05 | 2011-01-10 | 3400 | 1 | 3400 |
| SO43699 | 9 | 14864 | 2010-12-29 | 2011-01-05 | 2011-01-10 | 3400 | 1 | 3400 |

Fact Check

Check if all dimension tables can successfully join to the fact table

Fact Check

Check if all dimension tables can successfully join to the fact table

```

SQLQuery4.s...David (69)*  SQLQuery9 fa...\\David (65)*
1  SELECT * FROM [gold].[fact_sales] f
2  LEFT JOIN [gold].[dim_customers] c
3  ON c.customer_key = f.customer_key
4  WHERE c.customer_key IS NULL
5

SQLQuery4.s...David (69)*  SQLQuery9 fa...\\David (65)*
1  SELECT *
2  FROM [gold].[fact_sales] f
3  LEFT JOIN [gold].[dim_customers] c
4  ON c.customer_key = f.customer_key
5  LEFT JOIN [gold].[dim_products] p
6  ON p.product_key = f.product_key
7  WHERE p.product_key IS NULL
8

```

The left window shows a query joining fact_sales and dim_customers, with a WHERE clause filtering for customer_keys that are null. The right window shows a similar query joining fact_sales and dim_products, also filtering for product_keys that are null. Both queries run successfully.

everything is matching perfectly.

The Jira board has the following tasks:

- Build Gold Layer - Coding: Data Integration (Status: OPEN)
- Build Gold Layer - Validating: Data Integration Checks (Status: OPEN)
- 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

A detailed view of the "Validating: Data Integration Checks" task is shown, containing sub-tasks:

- Document: Draw Data Model of Star Schema (Draw.io)
- Document: Create Data Catalog
- Document: Extend Data Flow (Draw.io)
- Commit Code in Git Repo

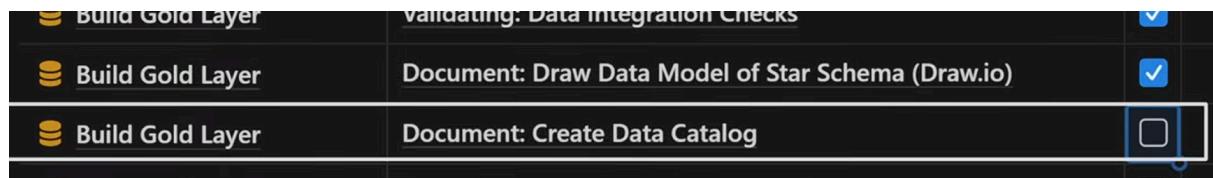
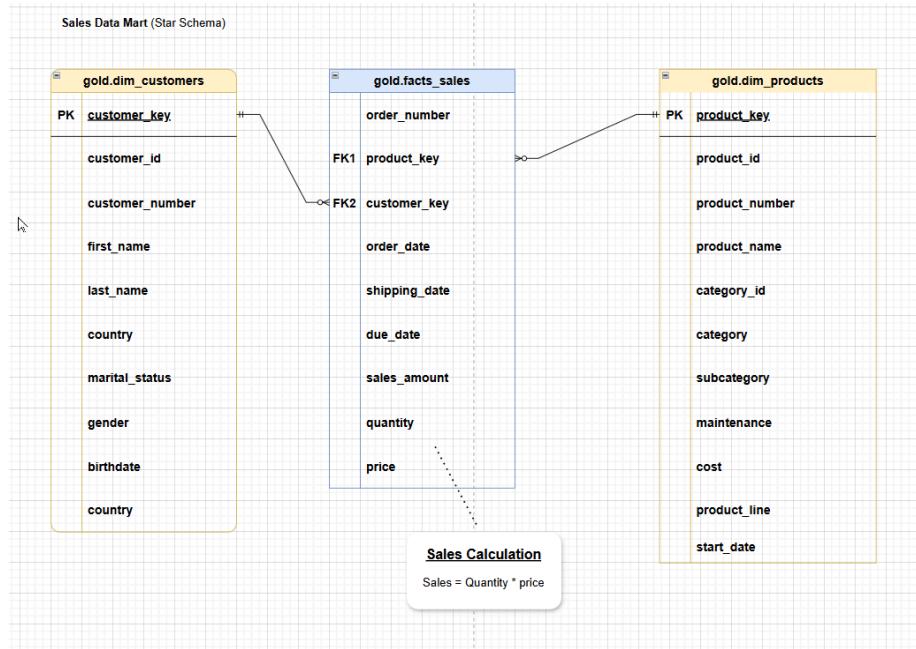
Relationship

in a star schema, the relationship between fact and dimensions is 1-to-many (1:N)

The diagram shows a relationship named "Many (Optional)" between two entities. A callout bubble lists three points:

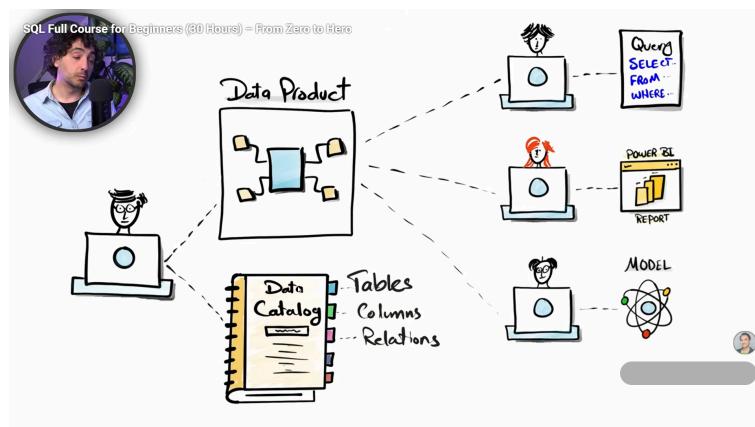
1. Customers who haven't placed any orders yet.
2. Customers who have placed only one order.
3. Customers who have placed multiple orders.

- Many (Optional)**
1. Customers who haven't placed any orders yet.
 2. Customers who have placed only one order.
 3. Customers who have placed multiple orders.



Build Gold Layer

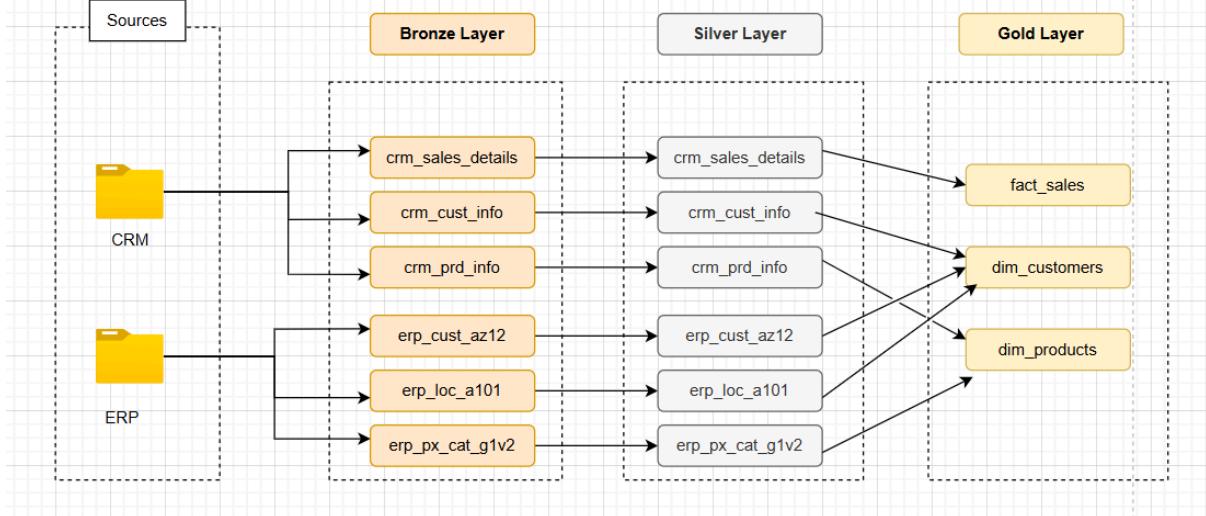
Data Catalog



Build Gold Layer

Data Flow

Data flow Diagram



The end.