INCREMENTAL DATA LOADING AND AUDIT PROJECT WITH SSIS

ENG. SILVA PARRAGUEZ MAXIMO

I. UNDERSTANDING THE DATA AND THE PROJECT

1. Data source:

A dataset containing information about sales, profits, discounts, and other data related to a fictional retail store was used. It includes data such as order date, product category, region, etc.

The data was obtained from the Kaggle repository : https://www.kaggle.com/datasets/vivek468/superstore-dataset-final

The purpose of this project, using SSIS, is to **integrate data from a source** (in this case, a CSV file) into the **SQL Server Ventas_Destino table without duplicating records** . Only new data will be inserted and **existing records will be updated in the event of changes in the source** . In addition, **a change audit will be implemented that will record, in the Ventas_Auditoria table** , essential details such as the update date, the previous value of the data, the modified column and, for example, the user along with their IP or the name of the computer that made the change.

2. Columns:

The datased contains 21 columns which are detailed below:

No.	Column	Description
1	Row ID	Unique identification for each row.
2	Order ID	Unique order identification for each Customer.
3	Order Date	Product order date.
4	Ship Date	Product shipping date.
5	Ship Mode	Shipping method specified by the Customer.
6	Customer ID	Unique ID to identify each Client.
7	Customer Name	Client Name.
8	Segment	The segment to which the Customer belongs.
9	Country	Country of residence of the Client.
10	City	Client's city of residence.
11	State	Customer's state of residence.
12	Postal Code	Postal code of each Client.
13	Region	Region to which the Client belongs.
14	Product ID	Unique Product Identification.
15	Category	Category of the requested product.
16	Sub- Category	Subcategory of the requested product.
17	Product Name	Product Name
18	Sales	Product Sales.
19	Quantity	Product Quantity.
20	Discount	Discount granted.
21	Profit	Gain/Loss incurred.

II. PROJECT DEVELOPMENT

Step 1: Prepare the Database and Tables:

Three tables were created in a database:

- **Ventas_Origen** (simulates the table where the new data arrives).
- **Ventas_Destino** (stores consolidated data, without duplicates).
- **Ventas_Auditoria** (stores records that were modified for some reason and the details of that update).

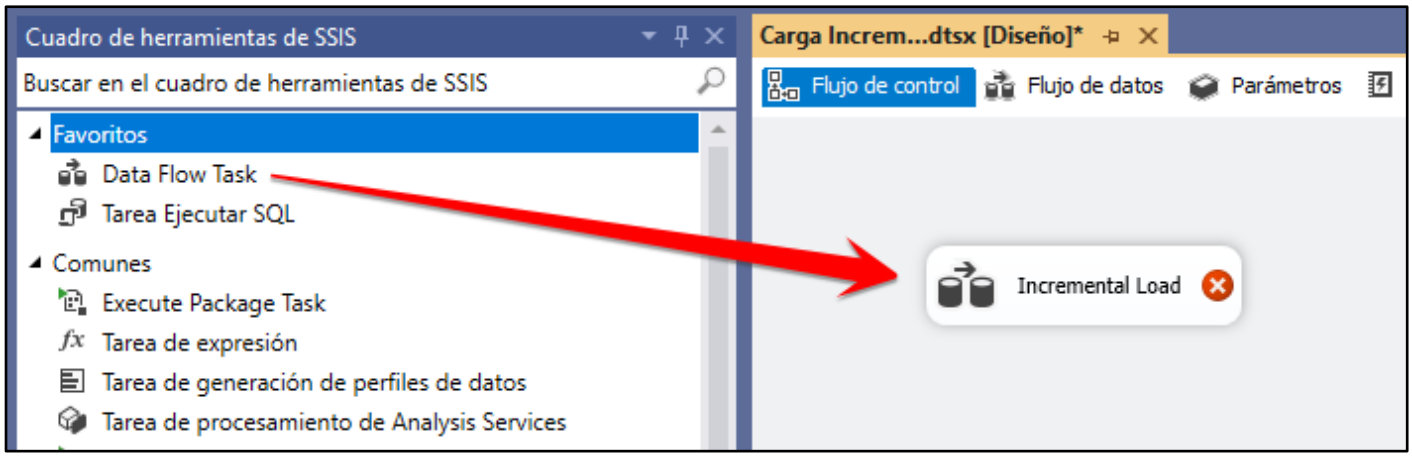
```sql
CREATE DATABASE CARGA_INCREMENTAL;
USE CARGA_INCREMENTAL;

-- Tabla de origen (datos nuevos llegan aquí)
CREATE TABLE Ventas_Origen (
    Row_ID INT,
    Order_ID VARCHAR(50),
    Order_Date DATE,
    Ship_Date DATE,
    Ship_Mode VARCHAR(50),
    Customer_ID VARCHAR(50),
    Customer_Name VARCHAR(100),
    Segment VARCHAR(50),
    Country VARCHAR(50),
    City VARCHAR(100),
    State VARCHAR(100),
    Postal_Code VARCHAR(50),
    Region VARCHAR(50),
    Product_ID VARCHAR(50),
    Category VARCHAR(50),
    Sub_Category VARCHAR(50),
    Product_Name VARCHAR(255),
    Sales DECIMAL(18,2),
    Quantity INT,
    Discount DECIMAL(5,2),
    Profit DECIMAL(18,2),
    Date_Insert DATETIME DEFAULT GETDATE(),
);
```

```sql
-- Tabla de destino (solo almacena nuevos o actualizados)
CREATE TABLE Ventas_Destino (
    Row_ID INT PRIMARY KEY,
    Order_ID VARCHAR(50),
    Order_Date DATE,
    Ship_Date DATE,
    Ship_Mode VARCHAR(50),
    Customer_ID VARCHAR(50),
    Customer_Name VARCHAR(255),
    Segment VARCHAR(50),
    Country VARCHAR(50),
    City VARCHAR(255),
    State VARCHAR(255),
    Postal_Code VARCHAR(50),
    Region VARCHAR(50),
    Product_ID VARCHAR(50),
    Category VARCHAR(50),
    Sub_Category VARCHAR(50),
    Product_Name VARCHAR(255),
    Sales DECIMAL(18,2),
    Quantity INT,
    Discount DECIMAL(5,2),
    Profit DECIMAL(18,2),
    Date_Insert Datetime
);
```

```sql
-- Tabla de Auditoría (almacena detalles de modificaciones de algún registros)
CREATE TABLE Ventas_Auditoria (
    ID_Auditoria INT IDENTITY(1,1) PRIMARY KEY,
    Row_ID INT,
    Order_ID VARCHAR(50),
    Fecha_Actualizacion DATETIME DEFAULT GETDATE(),
    Columnas_Modificadas VARCHAR(255),
    Valores_Anteriores NVARCHAR(MAX),
    Usuario NVARCHAR(255) DEFAULT SUSER_NAME(), -- Usuario que hizo la modificación
    IP_Equipo NVARCHAR(50) DEFAULT HOST_NAME()  -- IP o nombre del equipo que realizó la modificación
);
```

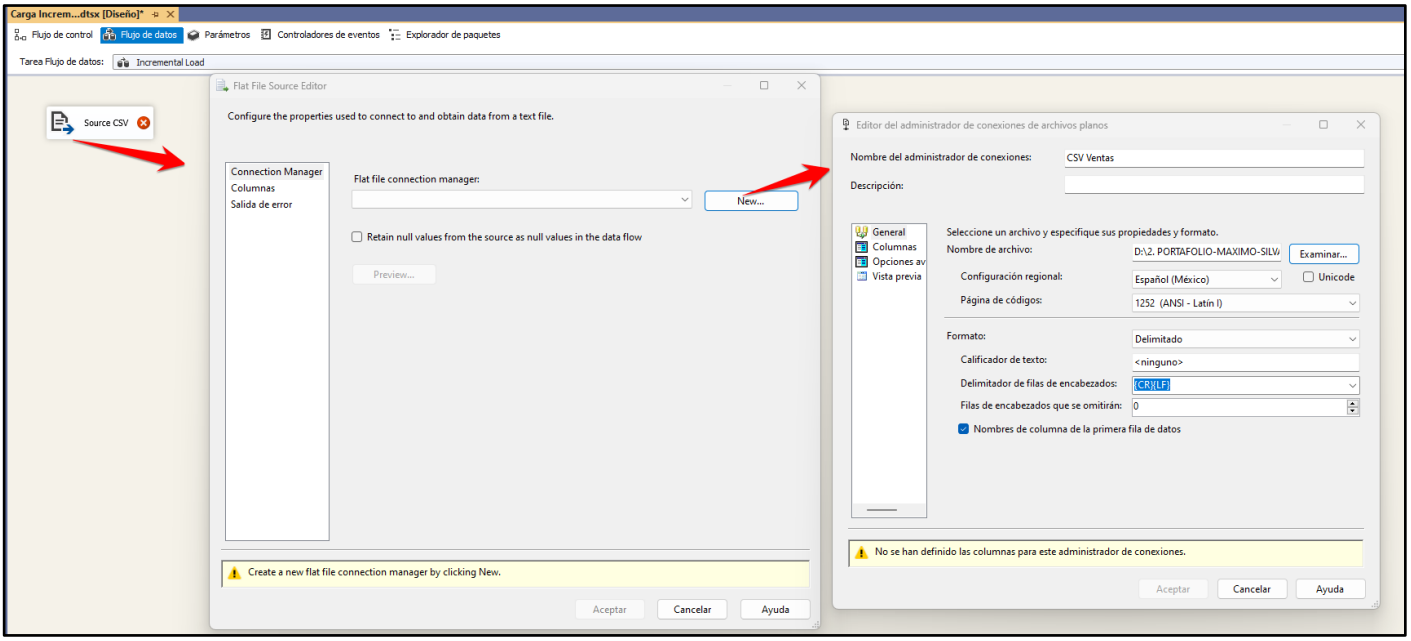
Step 2: Creating a project in SSIS.

Task component .



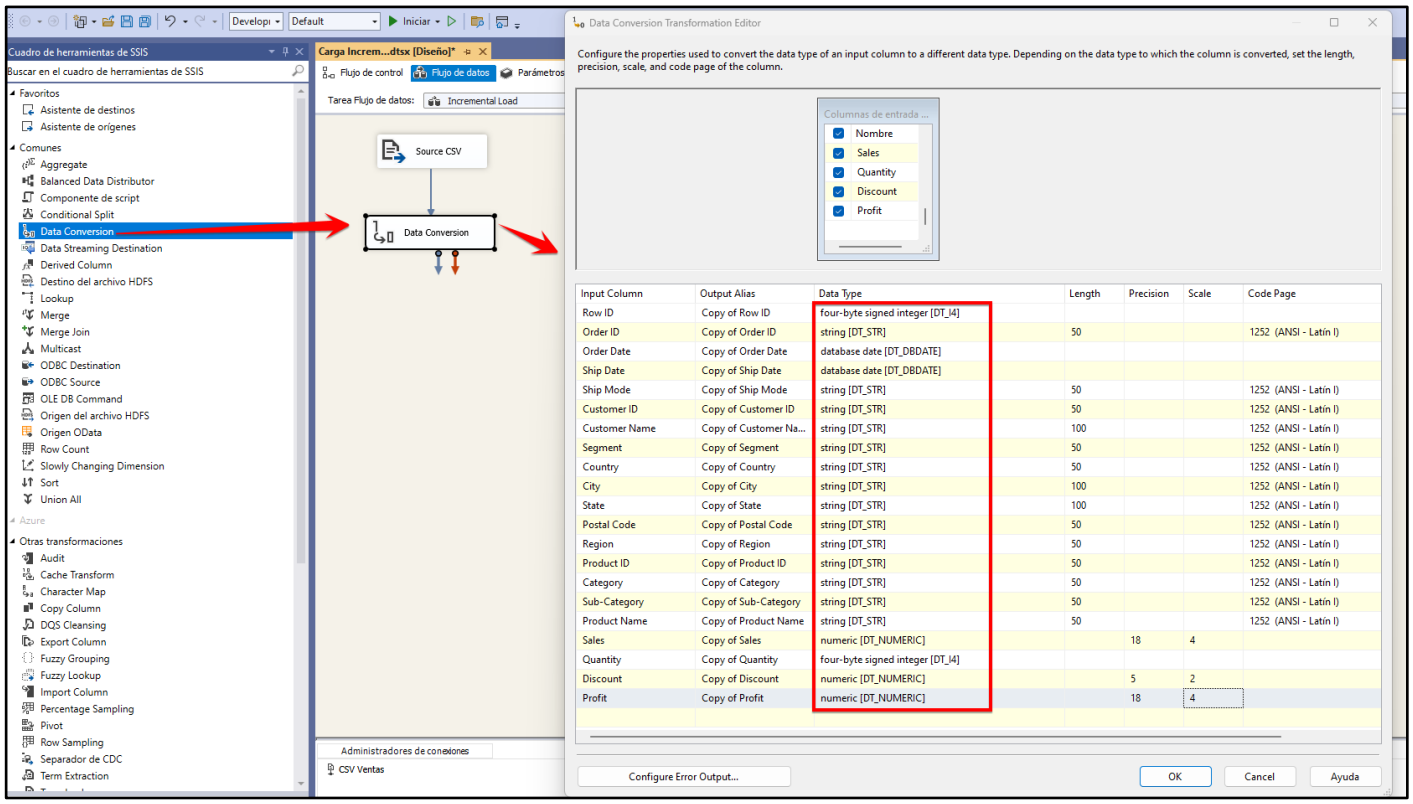
Step 3: Source CSV.

I set the Data Source from the dataset in CSV format:



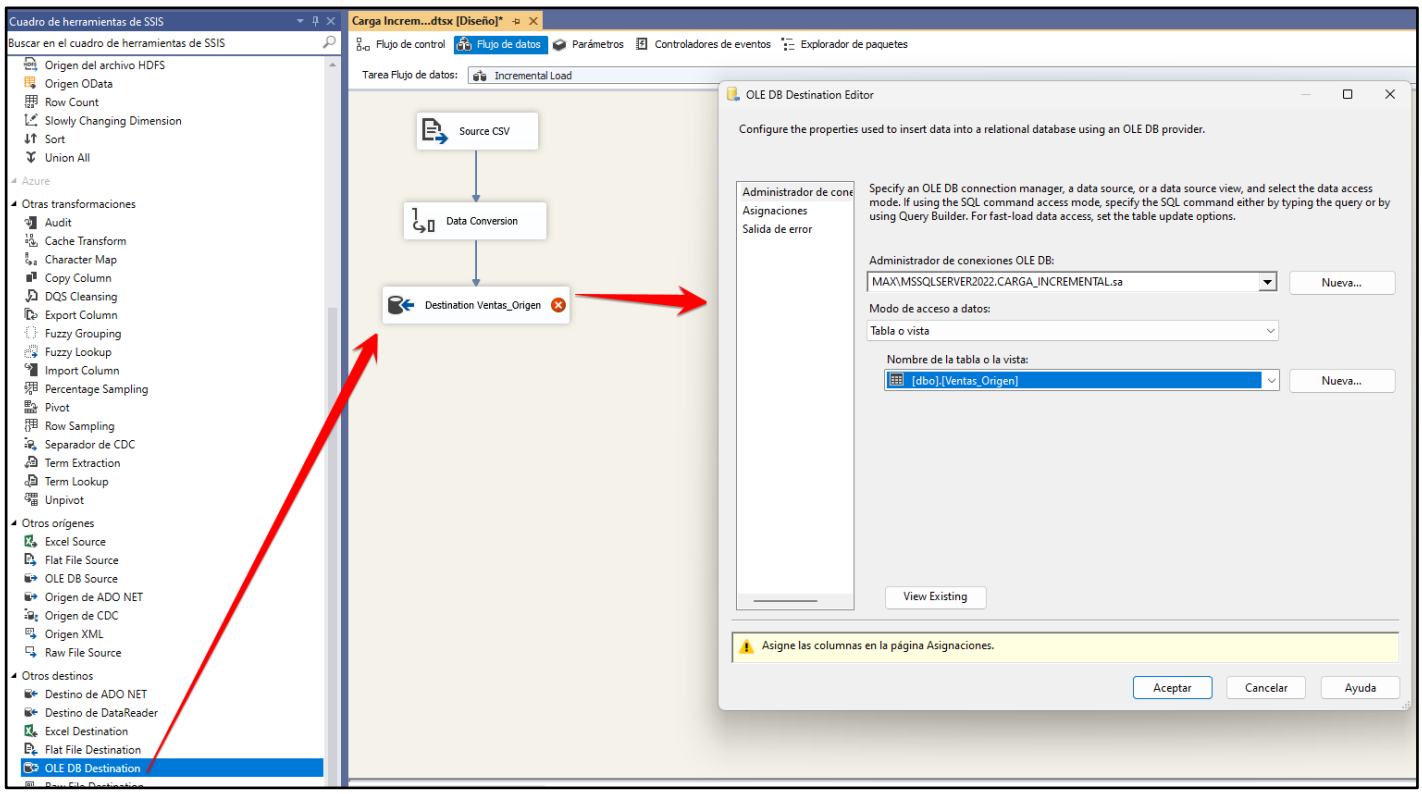
Step 4: Data Conversion Component .

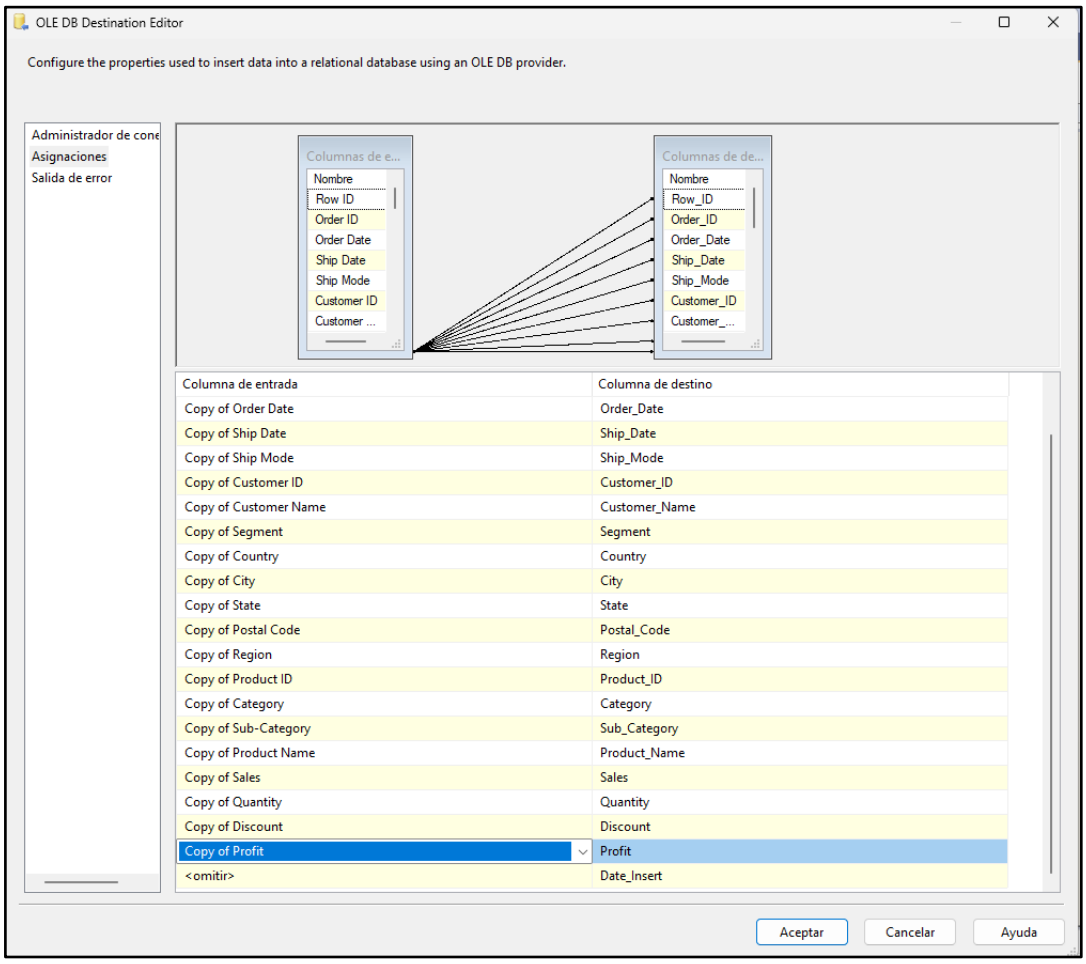
I will use the data conversion component depending on the data type in the DB.



Step 5: From Data Conversion to Sales_Source .

The Destination where the data will be stored is configured Initially , in the Ventas_Origen table .





Step 6: From Ventas_Origen to Ventas_Destino and Ventas_Auditoria .

I used a SQL query that consists of the following:

1. Create a Temporary Audit Table where audit records will be temporarily stored to be used for comparison later.
2. The records that have changed are recorded in the Temporary Table, that is, a comparison is made between the records in the Ventas_Destino table and the records in Ventas_Origen and if data is found that has changed, it is recorded in the Audit temporary table.
3. Ventas_Auditoria table are inserted from the Temporary table.
4. The data is updated, if any, in the Ventas_Destino table .
5. New data will be inserted if it does not exist in the Ventas_Destino table .

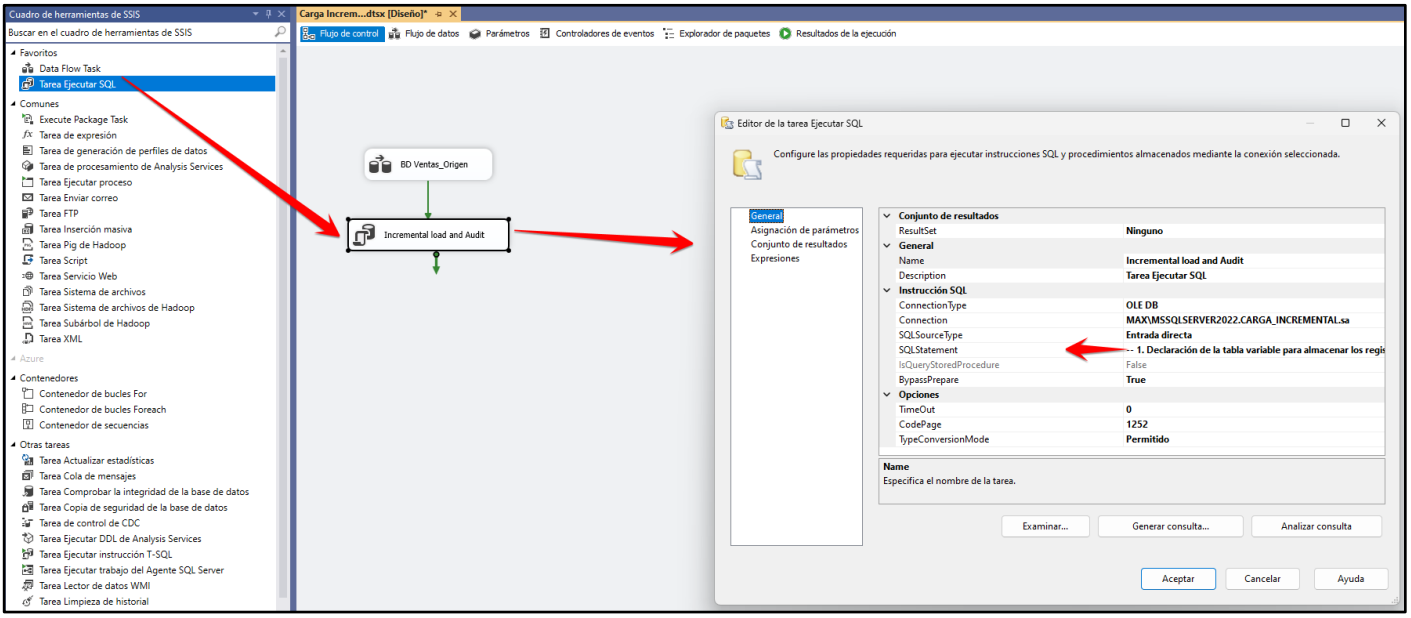
```sql
/*-----------------------------------------CONSULTA PARA LA LOGICA DEL PROYECTO-----------------------------------------*/


-- 1. Declaración de la tabla variable para almacenar los registros de auditoría
DECLARE @TempAuditoria TABLE (
    Row_ID INT,                         -- Identificador de la fila
    Order_ID VARCHAR(50),               -- Identificador de la orden
    Fecha_Actualizacion DATETIME,       -- Fecha y hora en que se actualiza el registro
    Columnas_Modificadas VARCHAR(255),  -- Columnas que han sido modificadas
    Valores_Anteriores NVARCHAR(MAX)    -- Valores anteriores de las columnas modificadas
);

-- 2. Guardar en la variable de tabla los registros que han cambiado
INSERT INTO @TempAuditoria (Row_ID, Order_ID, Fecha_Actualizacion, Columnas_Modificadas, Valores_Anteriores)
SELECT
    target.Row_ID,
    target.Order_ID,
    GETDATE() AS Fecha_Actualizacion,
    STRING_AGG(CASE
        WHEN target.Order_Date <> source.Order_Date THEN 'Order_Date'
        WHEN target.Ship_Date <> source.Ship_Date THEN 'Ship_Date'
        WHEN target.Ship_Mode <> source.Ship_Mode THEN 'Ship_Mode'
        WHEN target.Customer_ID <> source.Customer_ID THEN 'Customer_ID'
        WHEN target.Customer_Name <> source.Customer_Name THEN 'Customer_Name'
        WHEN target.Segment <> source.Segment THEN 'Segment'
        WHEN target.Country <> source.Country THEN 'Country'
        WHEN target.City <> source.City THEN 'City'
        WHEN target.State <> source.State THEN 'State'
        WHEN target.Postal_Code <> source.Postal_Code THEN 'Postal_Code'
        WHEN target.Region <> source.Region THEN 'Region'
        WHEN target.Product_ID <> source.Product_ID THEN 'Product_ID'
        WHEN target.Category <> source.Category THEN 'Category'
        WHEN target.Sub_Category <> source.Sub_Category THEN 'Sub_Category'
        WHEN target.Product_Name <> source.Product_Name THEN 'Product_Name'
        WHEN target.Sales <> source.Sales THEN 'Sales'
        WHEN target.Quantity <> source.Quantity THEN 'Quantity'
        WHEN target.Discount <> source.Discount THEN 'Discount'
        WHEN target.Profit <> source.Profit THEN 'Profit'
    END, ', ') AS Columnas_Modificadas,  -- Se concatena (separado por comas) los nombres de las columnas que han cambiado.
    STRING_AGG(CASE
        WHEN target.Order_Date <> source.Order_Date THEN 'Order_Date: ' + CAST(target.Order_Date AS NVARCHAR)
        WHEN target.Ship_Date <> source.Ship_Date THEN 'Ship_Date: ' + CAST(target.Ship_Date AS NVARCHAR)
        WHEN target.Ship_Mode <> source.Ship_Mode THEN 'Ship_Mode: ' + target.Ship_Mode
        WHEN target.Customer_ID <> source.Customer_ID THEN 'Customer_ID: ' + target.Customer_ID
        WHEN target.Customer_Name <> source.Customer_Name THEN 'Customer_Name: ' + target.Customer_Name
        WHEN target.Segment <> source.Segment THEN 'Segment: ' + target.Segment
        WHEN target.Country <> source.Country THEN 'Country: ' + target.Country
        WHEN target.City <> source.City THEN 'City: ' + target.City
        WHEN target.State <> source.State THEN 'State: ' + target.State
        WHEN target.Postal_Code <> source.Postal_Code THEN 'Postal_Code: ' + target.Postal_Code
```

```
        WHEN target.Region <> source.Region THEN 'Region: ' + target.Region
        WHEN target.Product_ID <> source.Product_ID THEN 'Product_ID: ' + target.Product_ID
        WHEN target.Category <> source.Category THEN 'Category: ' + target.Category
        WHEN target.Sub_Category <> source.Sub_Category THEN 'Sub_Category: ' + target.Sub_Category
        WHEN target.Product_Name <> source.Product_Name THEN 'Product_Name: ' + target.Product_Name
        WHEN target.Sales <> source.Sales THEN 'Sales: ' + CAST(target.Sales AS NVARCHAR)
        WHEN target.Quantity <> source.Quantity THEN 'Quantity: ' + CAST(target.Quantity AS NVARCHAR)
        WHEN target.Discount <> source.Discount THEN 'Discount: ' + CAST(target.Discount AS NVARCHAR)
        WHEN target.Profit <> source.Profit THEN 'Profit: ' + CAST(target.Profit AS NVARCHAR)
    END, '; ') AS Valores_Anteriores  -- Se concatena (separado por punto y coma) el nombre de la columna y su valor anterior
FROM Ventas_Destino AS target
JOIN Ventas_Origen AS source
    ON target.Row_ID = source.Row_ID
WHERE
        target.Order_Date <> source.Order_Date OR
        target.Ship_Date <> source.Ship_Date OR
        target.Ship_Mode <> source.Ship_Mode OR
        target.Customer_ID <> source.Customer_ID OR
        target.Customer_Name <> source.Customer_Name OR
        target.Segment <> source.Segment OR
        target.Country <> source.Country OR
        target.City <> source.City OR
        target.State <> source.State OR
        target.Postal_Code <> source.Postal_Code OR
        target.Region <> source.Region OR
        target.Product_ID <> source.Product_ID OR
        target.Category <> source.Category OR
        target.Sub_Category <> source.Sub_Category OR
        target.Product_Name <> source.Product_Name OR
        target.Sales <> source.Sales OR
        target.Quantity <> source.Quantity OR
        target.Discount <> source.Discount OR
        target.Profit <> source.Profit
       -- Se filtran los registros donde al menos una columna esta diferente entre origen y destino
GROUP BY target.Row_ID, target.Order_ID;

-- 3. Se inserta los cambios en la tabla de auditoría usando la variable de tabla
INSERT INTO Ventas_Auditoria (Row_ID, Order_ID, Fecha_Actualizacion, Columnas_Modificadas, Valores_Anteriores)
SELECT * FROM @TempAuditoria;  -- Se transfiere los registros auditados a la tabla física Ventas_Auditoria

-- 4. Se realiza la actualización en Ventas_Destino usando MERGE
MERGE INTO Ventas_Destino AS target
USING Ventas_Origen AS source
ON target.Row_ID = source.Row_ID

--Cuando se encuentra un registro en ambas tablas (MATCHED) y existe al menos una diferencia en alguno de los campos especificados,
-- se ejecuta la cláusula UPDATE.
WHEN MATCHED AND (
    target.Order_Date <> source.Order_Date OR
    target.Ship_Date <> source.Ship_Date OR
```

```sql
    target.Ship_Mode <> source.Ship_Mode OR
    target.Customer_ID <> source.Customer_ID OR
    target.Customer_Name <> source.Customer_Name OR
    target.Segment <> source.Segment OR
    target.Country <> source.Country OR
    target.City <> source.City OR
    target.State <> source.State OR
    target.Postal_Code <> source.Postal_Code OR
    target.Region <> source.Region OR
    target.Product_ID <> source.Product_ID OR
    target.Category <> source.Category OR
    target.Sub_Category <> source.Sub_Category OR
    target.Product_Name <> source.Product_Name OR
    target.Sales <> source.Sales OR
    target.Quantity <> source.Quantity OR
    target.Discount <> source.Discount OR
    target.Profit <> source.Profit
)
THEN
UPDATE SET  --Se actualizan los registros
    target.Order_Date = source.Order_Date,
    target.Ship_Date = source.Ship_Date,
    target.Ship_Mode = source.Ship_Mode,
    target.Customer_ID = source.Customer_ID,
    target.Customer_Name = source.Customer_Name,
    target.Segment = source.Segment,
    target.Country = source.Country,
    target.City = source.City,
    target.State = source.State,
    target.Postal_Code = source.Postal_Code,
    target.Region = source.Region,
    target.Product_ID = source.Product_ID,
    target.Category = source.Category,
    target.Sub_Category = source.Sub_Category,
    target.Product_Name = source.Product_Name,
    target.Sales = source.Sales,
    target.Quantity = source.Quantity,
    target.Discount = source.Discount,
    target.Profit = source.Profit

-- 5. Se inserta los datos nuevos si no existen en la tabla destino

WHEN NOT MATCHED THEN
INSERT (Row_ID, Order_ID, Order_Date, Ship_Date, Ship_Mode, Customer_ID, Customer_Name,
        Segment, Country, City, State, Postal_Code, Region, Product_ID,
        Category, Sub_Category, Product_Name, Sales, Quantity, Discount, Profit, Date_Insert)
VALUES (source.Row_ID, source.Order_ID, source.Order_Date, source.Ship_Date, source.Ship_Mode, source.Customer_ID, source.Customer_Name,
        source.Segment, source.Country, source.City, source.State, source.Postal_Code, source.Region, source.Product_ID,
        source.Category, source.Sub_Category, source.Product_Name, source.Sales, source.Quantity, source.Discount, source.Profit, source.Date_Insert);
```

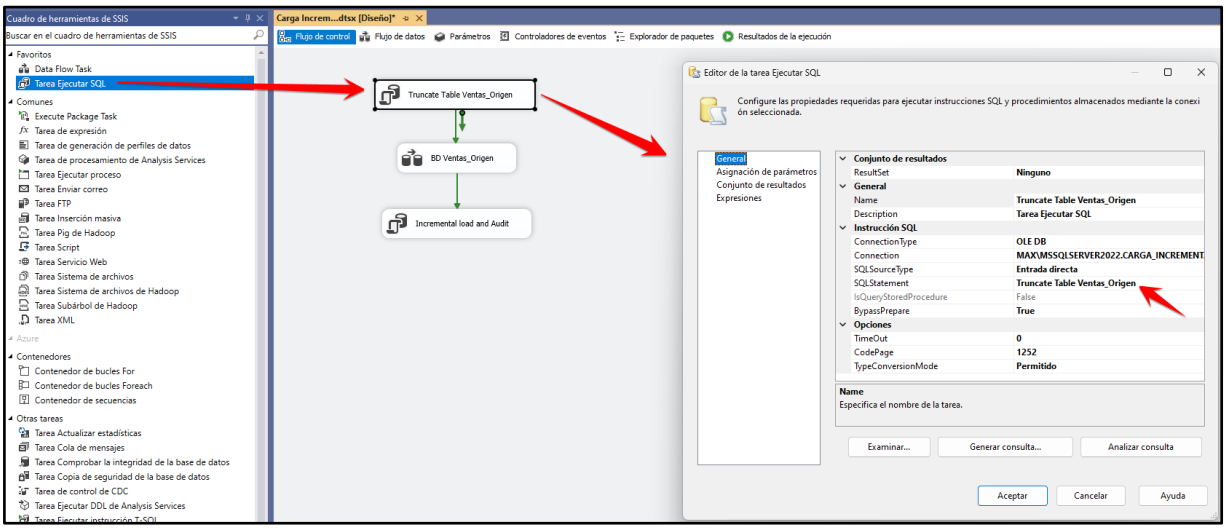
Step 7: Execute SQL Task Component.

I configured the Execute SQL Task component and placed the SQL code from the previous step.



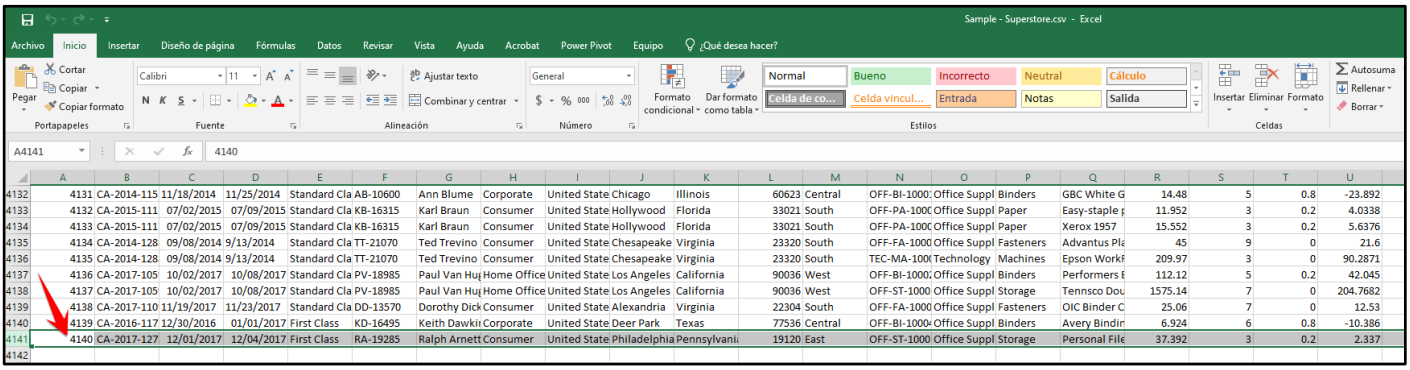
Step 8: Cleaning Ventas_Origen Table .

Using an Execute SQL Task component I will clean up the Ventas_Origen table , since it is a table that is only used to do the incremental load query.

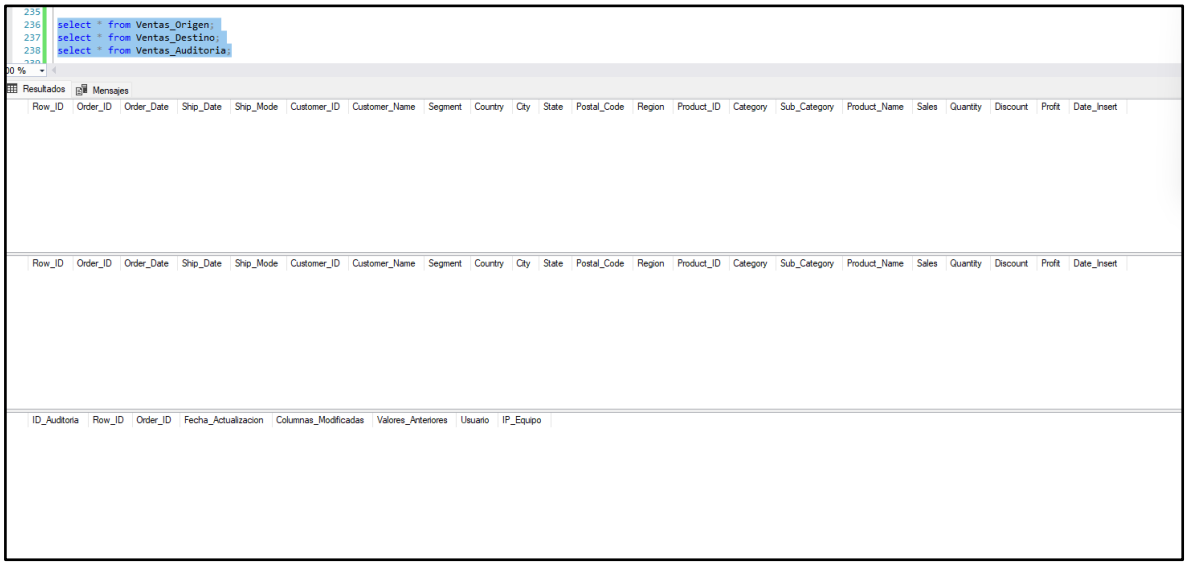


Step 9: Running the project and verifying the data.

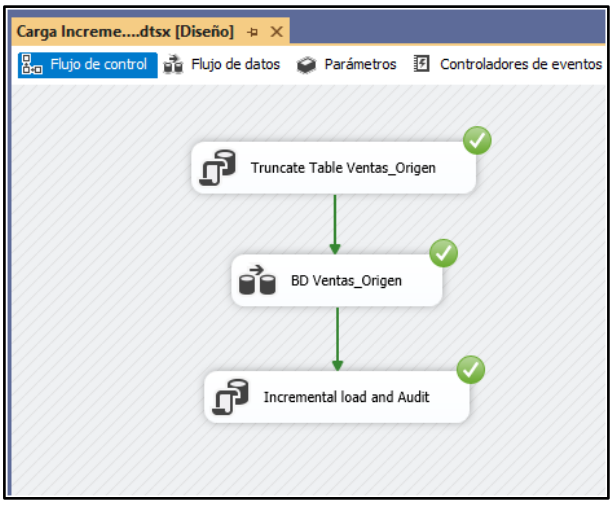
First I extracted 4140 records from the dataset, and we will pass that to the SSIS project.



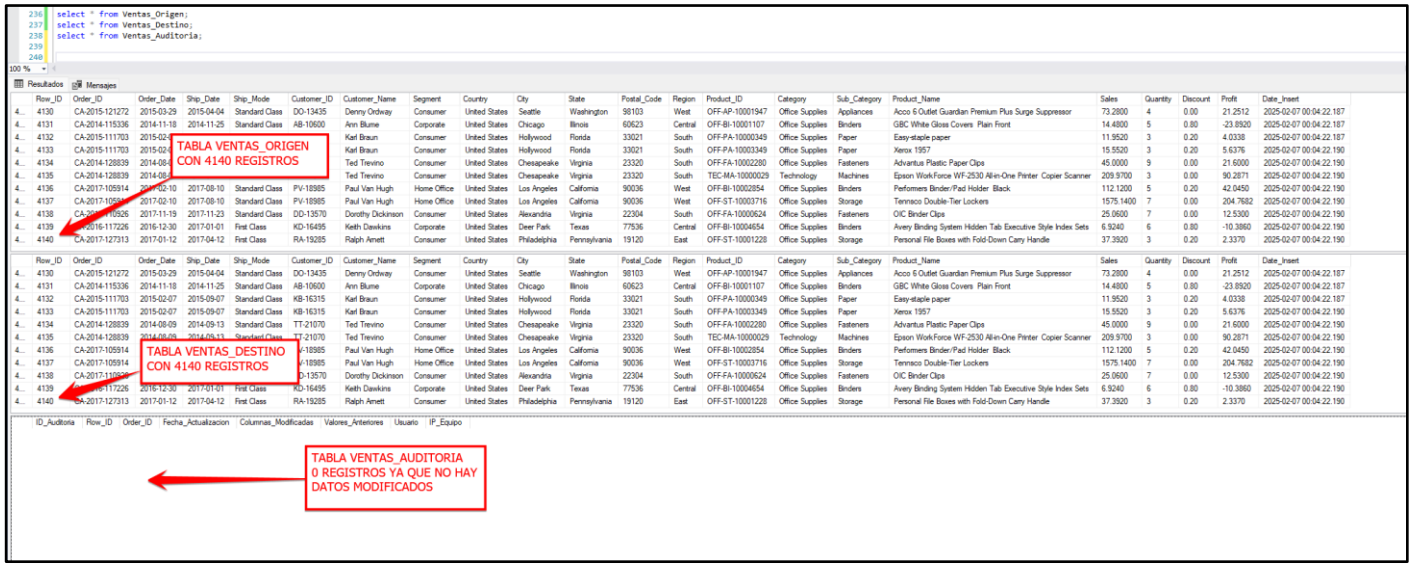
Secondly, we verify that there are no records in the tables in SQL:



Third, I execute the project.

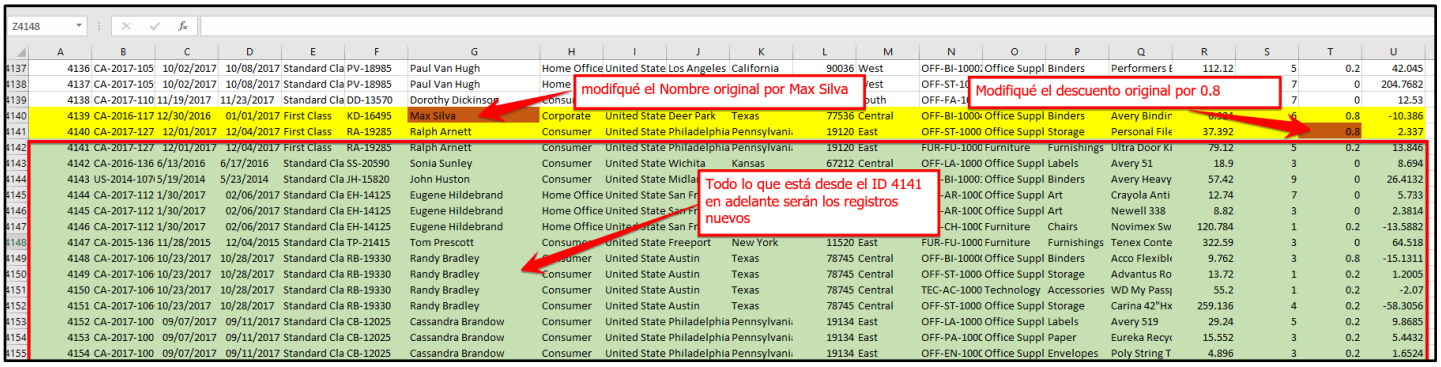


Fourth, we check the records in the tables in SQL Server.

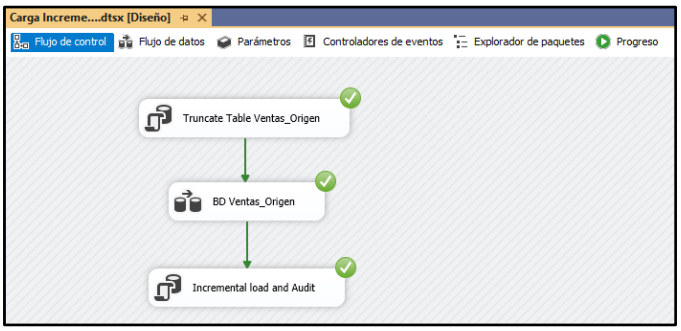


Fifth, I will add new data to the dataset, **I will add 60 records** to complete the 4200 records, and I will also modify the records with Row_ID :

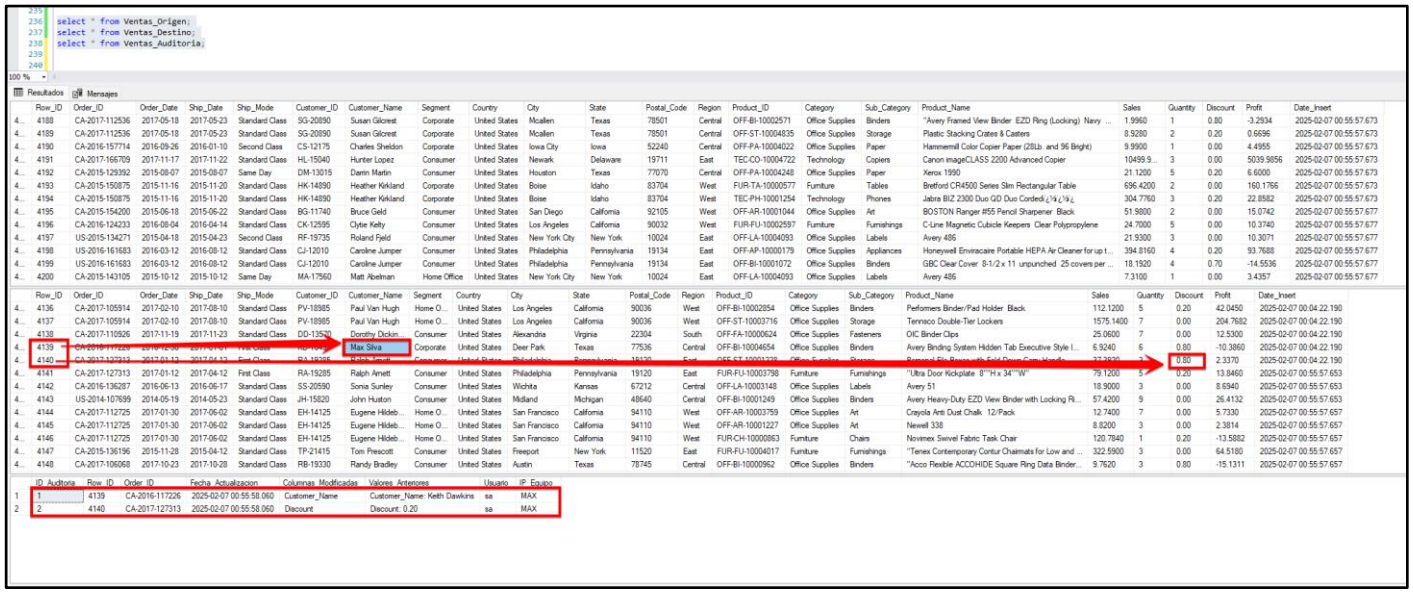
- **Customer Column Name** , I will change “Keith Dawkins” to “Max Silva”.
- **Discount column** , I will change “0.20” to “0.80”.



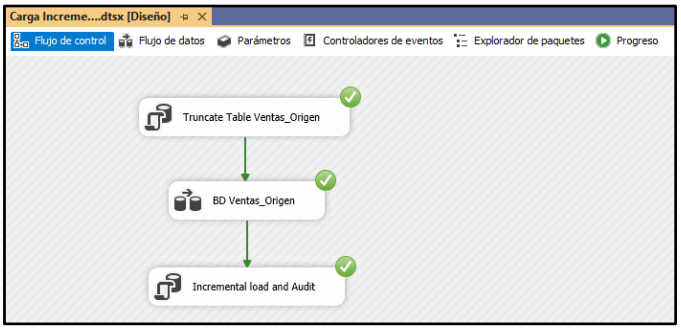
Sixth, we execute the project, where there should be a total of 4200 records in both the Ventas_Origen table and the Ventas_Destino table, and a total of 2 records in the Ventas_Auditoria table, since I made two modifications, and those changes should also be reflected in the Ventas_Destino table.

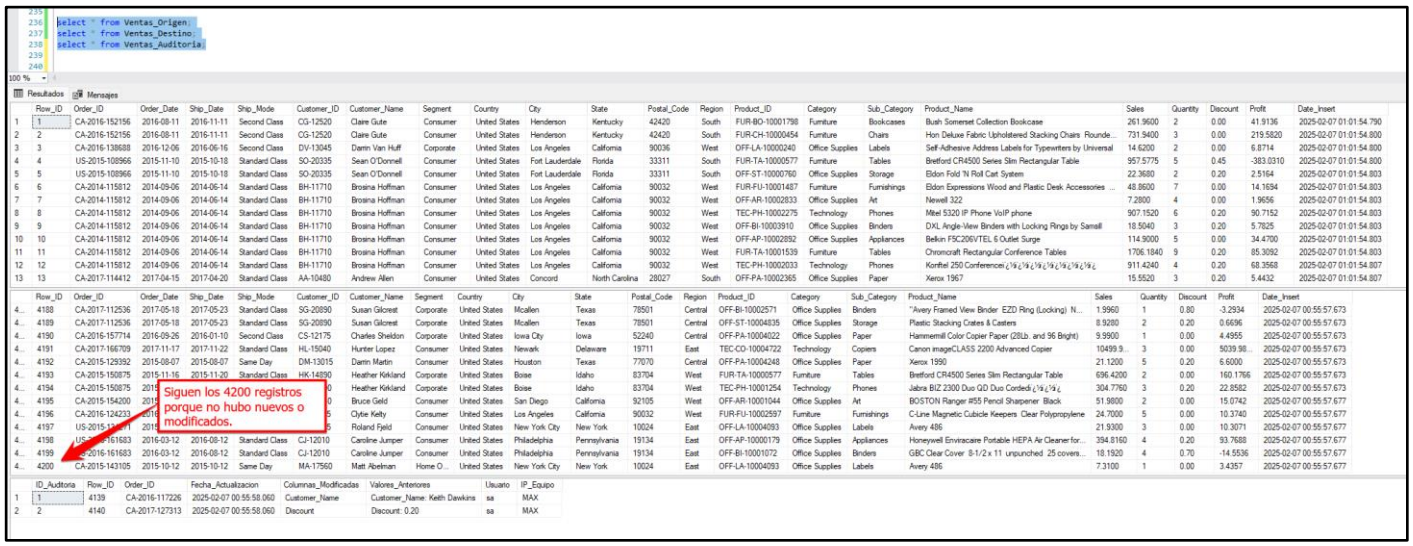


Seventh, we verify that indeed In the Ventas_Destino table there are a total of 4200 records where there are two modifications, those of ID 4139 and 4140, and in the Ventas_Audit table , the details of what was modified were inserted.



Eighth, I will run the same project again to validate that no more values will be entered into the Ventas_Destino table , since they are the same ones that are already in it because this time I will not make any modifications.

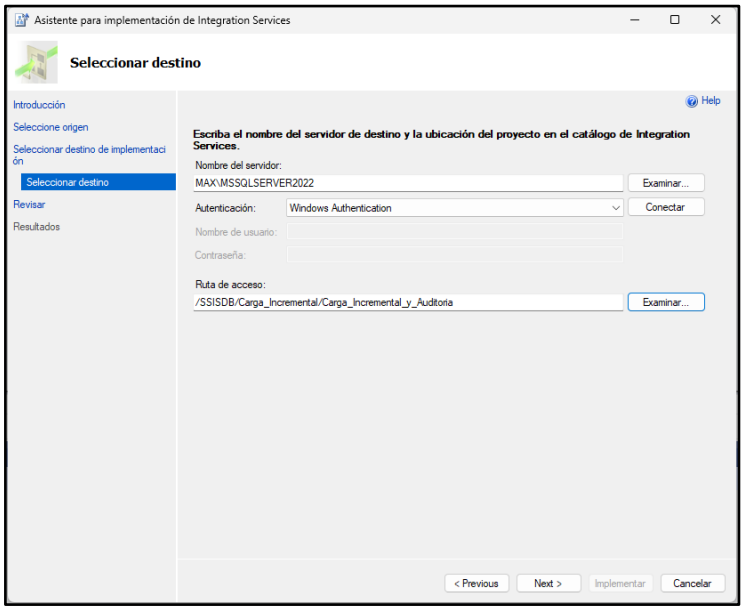


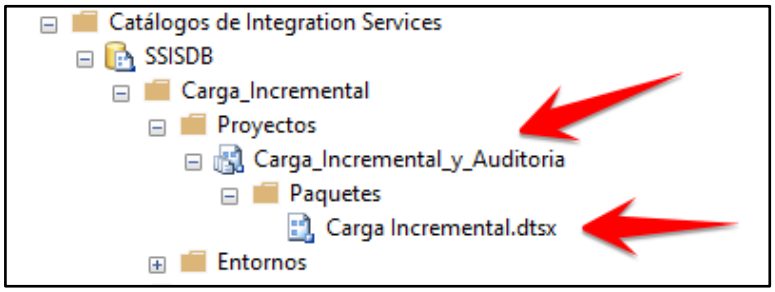


Step 10: Creating a scheduled task to run for example every 2 hours.

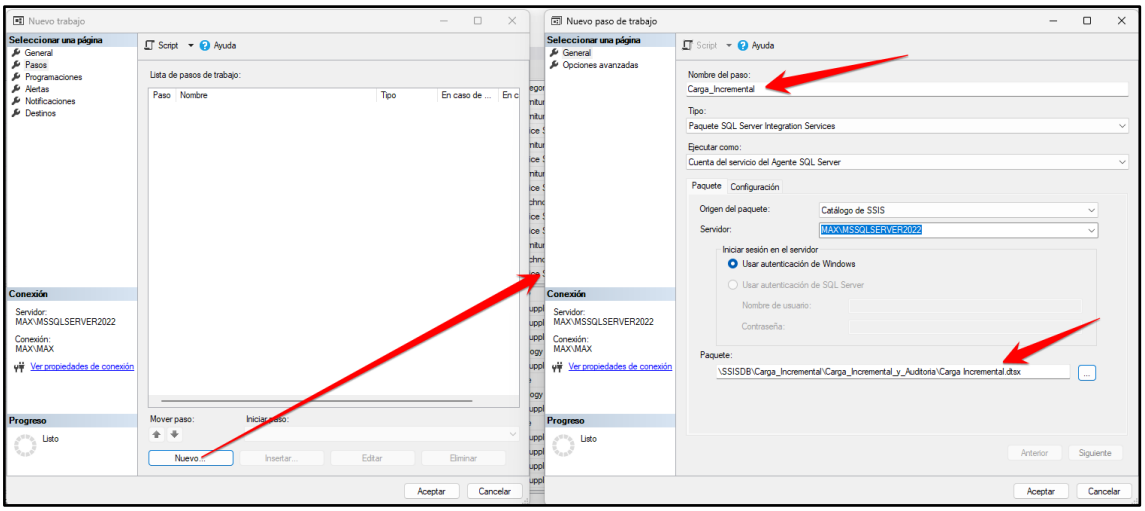
For this project, a scheduled task was configured to run the project automatically every 2 hours. To do this, we first implemented the SSIS project so that it appears in the catalog section in SQL Server, and then created a scheduled task.

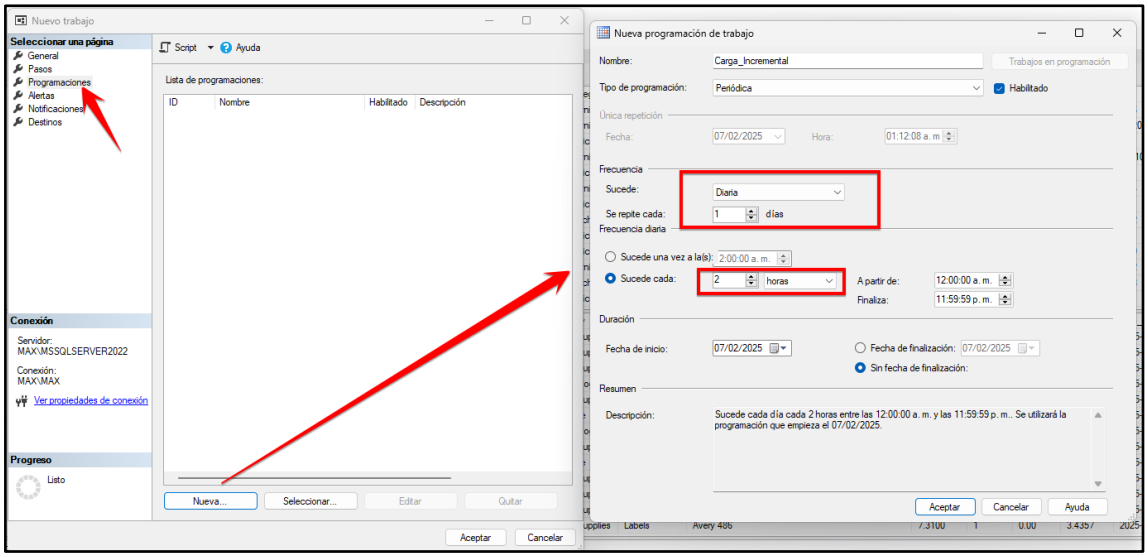
INTEGRATION SERVICE IMPLEMENTATION:

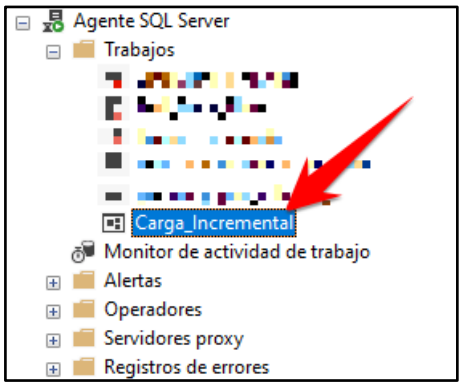




CREATING NEW WORK:







Finally, with this, this process will be done automatically every day every 2 hours, having as a source the CSV file that contains the dataset and final destination Ventas_Destino , but it can be done with any data source that is inserting data from time to time and thus have an audit of the records and their modifications as well as consistent, solid data that is not repeated in Ventas_Destino.