

# **PROYECTO DE LA CREACIÓN DE UN DATAMART DE ACCIDENTES DE TRÁNSITO EN EE.UU. (2016 - 2023)**

**ING. SILVA PARRAGUEZ MAXIMO**

## I. Preparando y entendiendo la Data:

### 1. Fuente:

Los datos se obtuvieron de registros de accidentes automovilísticos a nivel de todo EE.UU. que abarcan 49 estados. La data está comprendida entre febrero del 2016 hasta marzo del 2023. Es un .csv que contiene aproximadamente 7,7 millones de registros de accidentes.

Este proyecto consistió en que a partir de los datos del .CSV, se hizo todo el proceso de ETL, extracción de los Datos, la Transformación de los mismos, y la Carga a un destino final que es un Datamart, el cual es un esquema de ESTRELLA.

La fuente de datos se tomó de: <https://www.kaggle.com/datasets/sobhanmoosavi/us-accidents>

### 2. Columnas:

El .csv en mención contiene 46 columnas las cuales, cada una se detallan a continuación:

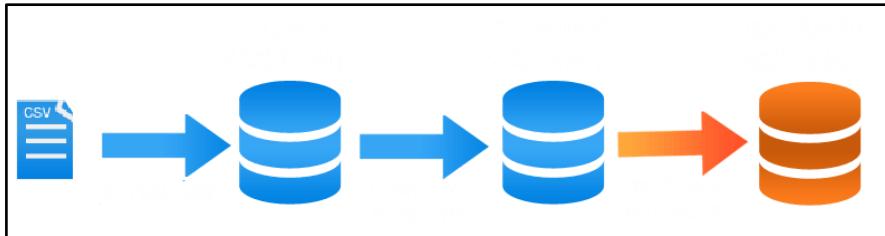
Columna Original	Descripción
ID	Este es un identificador único del registro del accidente.
Source	Fuente de datos brutos sobre accidentes
Severity	Muestra la gravedad del accidente, un número entre 1 y 4, donde 1 indica el menor impacto en el tráfico (es decir, breve retraso como consecuencia del accidente) y 4 indica un impacto significativo en el tráfico (es decir, larga demora).
Start_Time	Muestra la hora de inicio del accidente en la zona horaria local.
End_Time	Muestra la hora de finalización del accidente en la zona horaria local. La hora de finalización aquí se refiere al momento en que se descartó el impacto del accidente en el flujo de tráfico.
Start_Lat	Muestra la latitud en coordenadas GPS del punto de inicio.
Start_Lng	Muestra la longitud en coordenadas GPS del punto de inicio.
End_Lat	Muestra la latitud en coordenadas GPS del punto final.
End_Lng	Muestra la longitud en coordenadas GPS del punto final.
Distance(mi)	La longitud de la carretera afectada por el accidente en millas.
Description	Muestra una descripción del accidente proporcionada por una persona.
Street	Muestra el nombre de la calle en el campo de dirección.
City	Muestra la ciudad en el campo de dirección.
County	Muestra el condado en el campo de dirección.
State	Muestra el estado en el campo de dirección.
Zipcode	Muestra el código postal en el campo de dirección.
Country	Muestra el país en el campo de dirección.
Timezone	Muestra la zona horaria según la ubicación del accidente (este, centro, etc.).
Airport_Code	Indica una estación meteorológica ubicada en el aeropuerto que es la más cercana al lugar del accidente.
Weather_Timestamp	Muestra la marca de tiempo del registro de observación meteorológica (en hora local).
Temperature(F)	Muestra la temperatura (en Fahrenheit).
Wind_Chill(F)	Muestra la sensación térmica (en grados Fahrenheit).
Humidity(%)	Muestra la humedad (en porcentaje).
Pressure(in)	Muestra la presión del aire (en pulgadas).
Visibility(mi)	Muestra la visibilidad (en millas).

Wind_Direction	Muestra la dirección del viento.
Wind_Speed(mph)	Muestra la velocidad del viento (en millas por hora).
Precipitation(in)	Muestra la cantidad de precipitación en pulgadas, si hay alguna.
Weather_Condition	Muestra las condiciones meteorológicas (lluvia, nieve, tormenta, niebla, etc.)
Amenity	Una anotación de POI que indica la presencia de un servicio en una ubicación cercana.
Bump	Una anotación de POI que indica la presencia de un resalte o un tope de velocidad en una ubicación cercana.
Crossing	Una anotación de POI que indica la presencia de un cruce en una ubicación cercana.
Give_Way	Una anotación de POI que indica la presencia de ceda el paso en una ubicación cercana.
Junction	Una anotación de POI que indica la presencia de una intersección en una ubicación cercana.
No_Exit	Una anotación de POI que indica la presencia de sin salida en una ubicación cercana.
Railway	Una anotación de POI que indica la presencia de un ferrocarril en una ubicación cercana.
Roundabout	Una anotación de POI que indica la presencia de una rotonda en una ubicación cercana.
Station	Una anotación de POI que indica la presencia de una estación en una ubicación cercana.
Stop	Una anotación de POI que indica la presencia de una parada en una ubicación cercana.
Traffic_Calming	Una anotación de POI que indica la presencia de medidas de calmado del tráfico en una ubicación cercana.
Traffic_Signal	Una anotación de POI que indica la presencia de una señal de tráfico en una ubicación cercana.
Turning_Loop	Una anotación de POI que indica la presencia de Bucle giratorio en una ubicación cercana.
Sunrise_Sunset	Muestra el período del día (es decir, día o noche) según el amanecer/atardecer.
Civil_Twilight	Muestra el período del día (es decir, día o noche) según el crepúsculo civil.
Nautical_Twilight	Muestra el período del día (es decir, día o noche) en función del crepúsculo náutico.
Astronomical_Twilight	Muestra el período del día (es decir, día o noche) basado en el crepúsculo astronómico.

## II. PASO I: DE CSV a LOAD\_ACCIDENTS.

Debido a que los datos los tenemos en el .CSV tienen un tamaño de 3GB, y con un aproximado de 7,7 millones de registros, lo ideal es pasarlo a una tabla de Carga en SQL Server, esta tabla actuará como un primer almacén donde se guardarán los datos tal cual del .csv, lo cual permitirá manejar los datos de forma más eficiente, controlada y segura antes de ser cargados en una base de datos Staging (esta BD será el segundo almacén intermedio entre los datos originales y el DataMart) antes de ser pasadas al DataMart. Si bien es cierto, se hará dos pasos adicional, pero esto asegurará que el destino final reciba datos limpios, consistentes y listos para ser usados en análisis.

Proceso que se realizó fue:

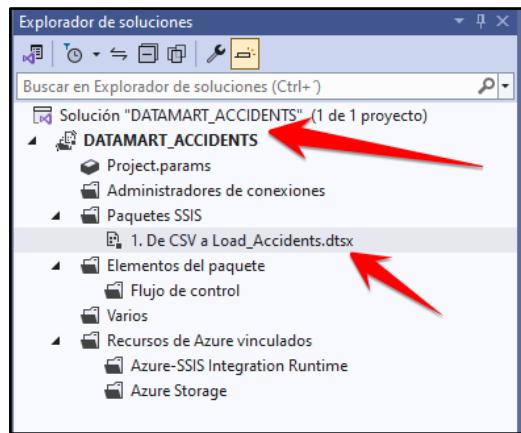


### 1. Creando una Base de datos en SQL Server.

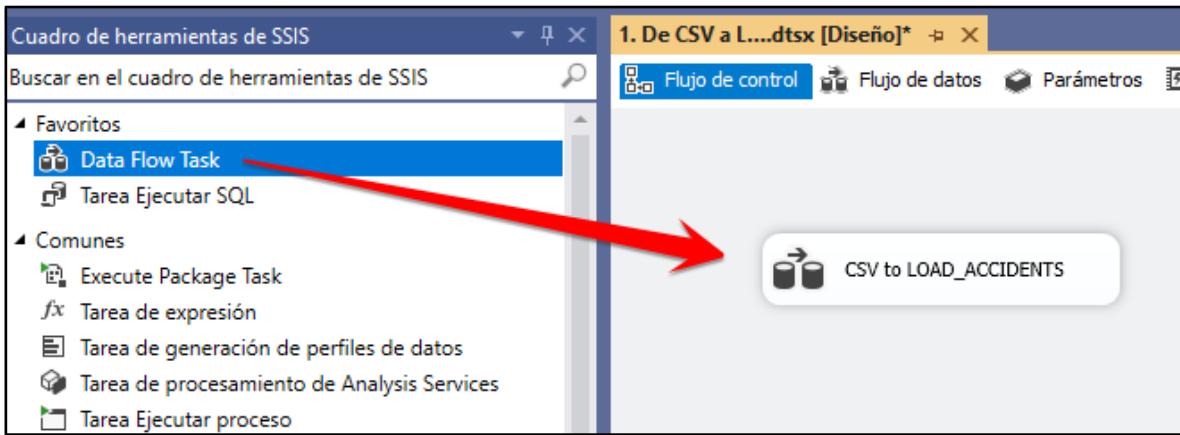
La base de datos LOAD\_ACCIDENTS, será la encargada de recibir los datos del archivo CSV en la tabla L\_ACCIDENTS, para ello, creamos la BD y la tabla:

```
25  CREATE DATABASE LOAD_ACCIDENTS
26  /*CREANDO LA TABLA L_ACCIDENTS*/
27
28  CREATE TABLE L_ACCIDENTS (
29      Id_Staging int identity primary key not null,
30      ID varchar(50),
31      Source VARCHAR(50),
32      Severity int,
33      Start_Time DATETIME2,
34      End_Time DATETIME2,
35      Start_Lat FLOAT,
36      Start_Lng FLOAT,
37      End_Lat FLOAT,
38      End_Lng FLOAT,
39      Distance_mi FLOAT,
40      Description VARCHAR(800),
41      Street VARCHAR(500),
42      City VARCHAR(100),
43      County VARCHAR(100),
44      State VARCHAR(50),
45      Zipcode VARCHAR(50),
46      Country VARCHAR(50),
47      Timezone VARCHAR(50),
48      Airport_Code VARCHAR(50),
49      Weather_Timestamp DATETIME2,
50      Temperature_F FLOAT,
51      Wind_Chill_F FLOAT,
52      Humidity FLOAT,
53      Pressure_in FLOAT,
54      Visibility_mi FLOAT,
55      Wind_Direction VARCHAR(50),
56      Wind_Speed_mph FLOAT,
57      Precipitation_in FLOAT,
58      Weather_Condition VARCHAR(100),
59      Amenity VARCHAR(10),
60      Bump VARCHAR(10),
61      Crossing VARCHAR(10),
62      Give_Way VARCHAR(10),
63      Junction VARCHAR(10),
64      No_Exit VARCHAR(10),
65      Railway VARCHAR(10),
66      Roundabout VARCHAR(10),
67      Station VARCHAR(10),
68      Stop VARCHAR(10),
69      Traffic_Calming VARCHAR(10),
70      Traffic_Signal VARCHAR(10),
71      Turning_Loop VARCHAR(10),
72      Sunrise_Sunset VARCHAR(10),
73      Civil_Twilight VARCHAR(10),
74      Nautical_Twilight VARCHAR(10),
75      Astronomical_Twilight VARCHAR(10)
76 );
```

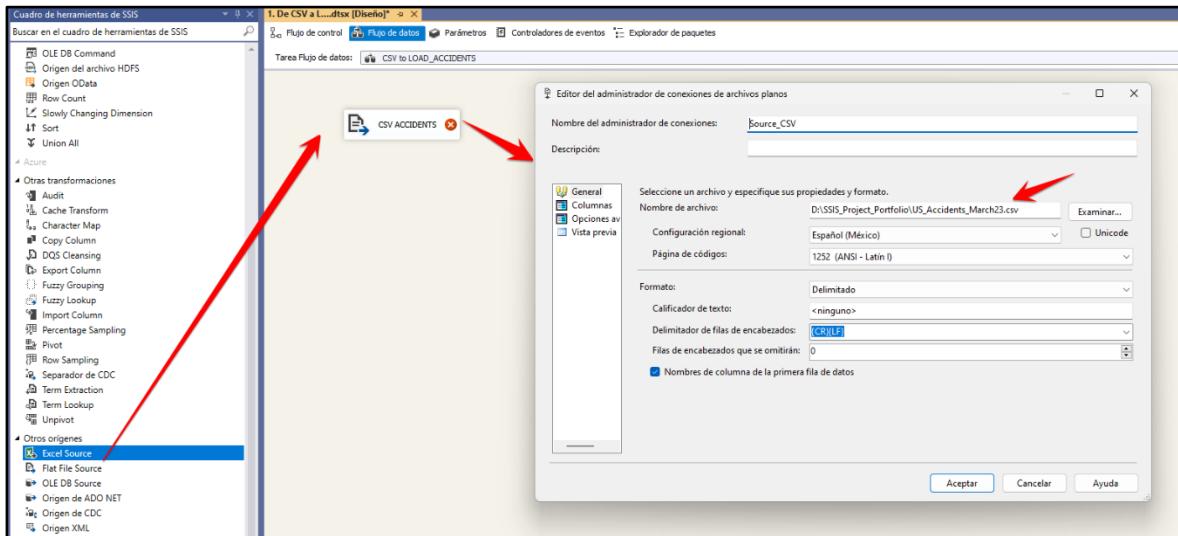
## 2. Creé un Proyecto en SSIS.



## 3. Utilicé una Tarea de Flujo de Datos.



## 4. Configuré el origen un archivo CSV.

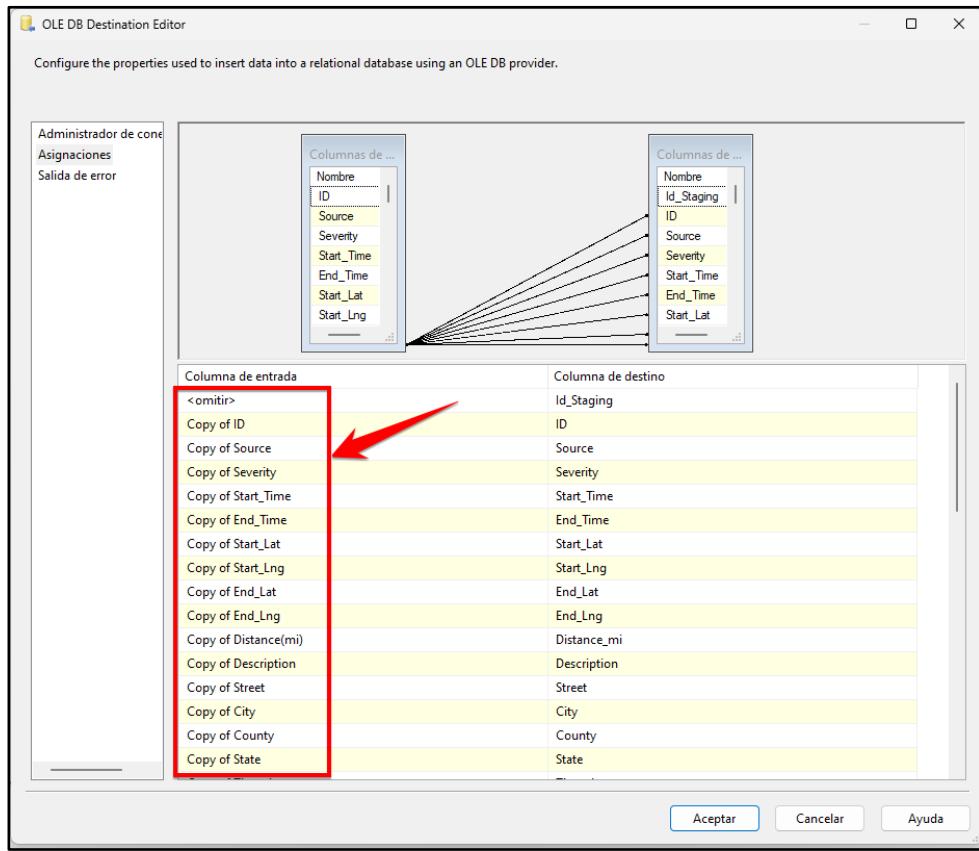


## 5. Convertí los tipos de datos para que no haya errores entre el origen y el destino.

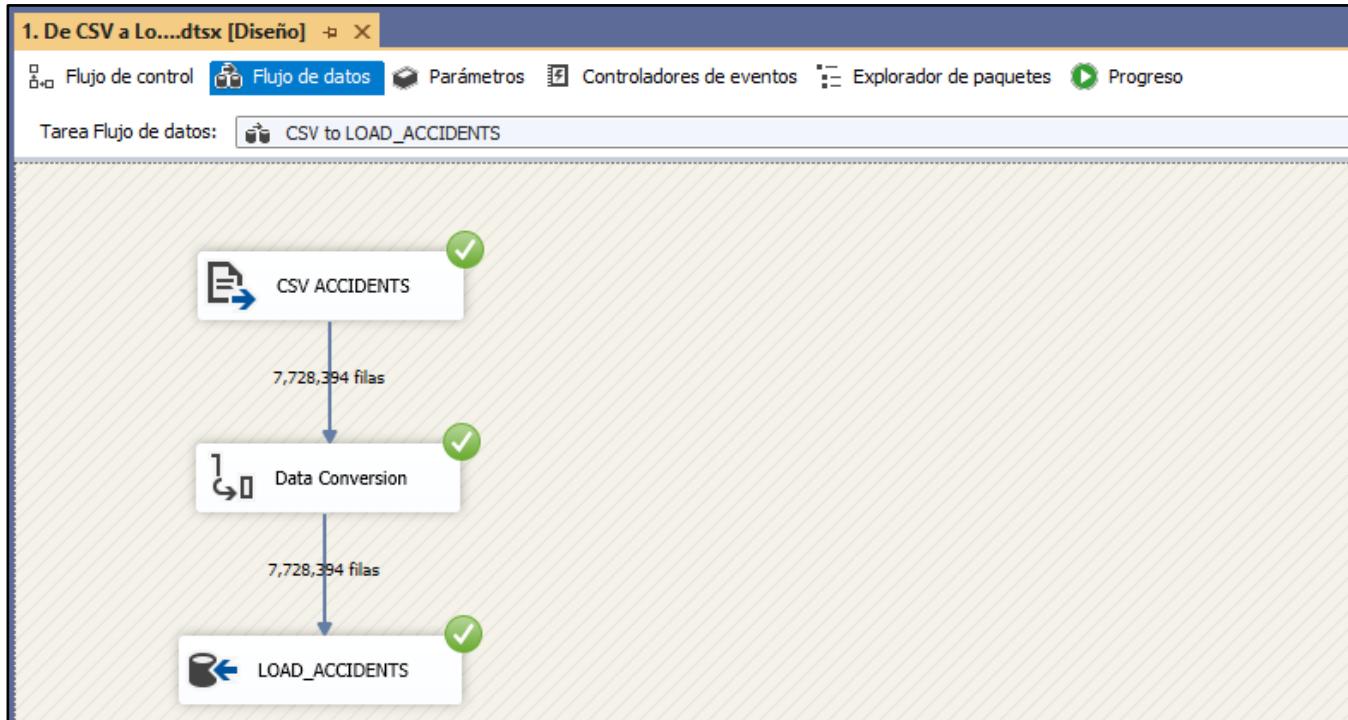
The screenshot shows the SSIS Data Conversion Transformation Editor. It displays a list of input columns (e.g., ID, Source, Severity, Start\_Time, End\_Time, Start\_Lat, Start\_Lng, End\_Lat, End\_Lng, Distance(mi), Description, Street, City, County, State, Zipcode, Country, Timezone, Airport\_Code, Weather\_Timestamp, Temperature(F), Wind\_Chill(F)) and their corresponding output types (e.g., string [DT\_STR], string [DT\_STR], four-byte signed integer [DT\_I4], database timestamp with precision [DT\_DBTIMESTAMP2], double-precision float [DT\_R8], string [DT\_STR], double-precision float [DT\_R8], double-precision float [DT\_R8]). A red box highlights the 'Data Type' column. To the right, the 'LOAD\_ACCIDENTS' connection manager script is shown, also with a red box highlighting the 'Data Type' section. The script creates a database and table, defining various columns with their respective data types and constraints.

## 6. Configuré un destino OLE DB para nuestra BD en SQL Server LOAD\_ACCIDENTS.

The screenshot shows the SSIS OLE DB Destination Editor. It displays the connection manager (MAXMSSQLSERVER2022\LOAD\_ACCIDENTS) and the table (dbo.L\_ACCIDENTS) where data will be inserted. A red arrow points from the 'OLE DB Destination' node in the main SSIS design view to this editor.



## 7. Procedí a Ejecutar el Proyecto:



8. Creé una nueva tabla con los nombres de cada ESTADO, ya que en el origen de la data esta en acrónimo, y se desea tener el nombre del estado que se hará más adelante en el proceso de limpieza.

```

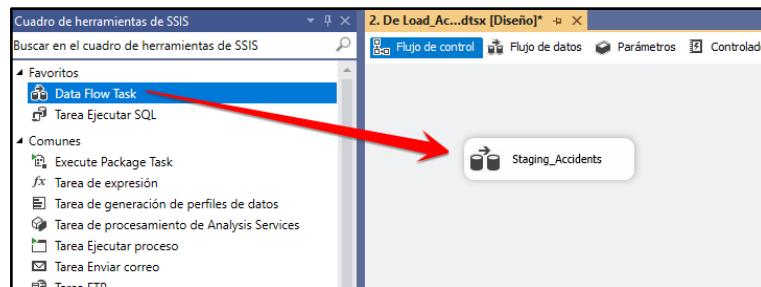
CREATE TABLE STATE_REFERENCE (
    State_Abbreviation CHAR(2) PRIMARY KEY,
    State_Name VARCHAR(50)
);

INSERT INTO STATE_REFERENCE (State_Abbreviation, State_Name)
VALUES
    ('AL', 'Alabama'), ('AK', 'Alaska'), ('AZ', 'Arizona'), ('AR', 'Arkansas'),
    ('CA', 'California'), ('CO', 'Colorado'), ('CT', 'Connecticut'),
    ('DE', 'Delaware'), ('FL', 'Florida'), ('GA', 'Georgia'),
    ('HI', 'Hawaii'), ('ID', 'Idaho'), ('IL', 'Illinois'),
    ('IN', 'Indiana'), ('IA', 'Iowa'), ('KS', 'Kansas'),
    ('KY', 'Kentucky'), ('LA', 'Louisiana'), ('ME', 'Maine'),
    ('MD', 'Maryland'), ('MA', 'Massachusetts'), ('MI', 'Michigan'),
    ('MN', 'Minnesota'), ('MS', 'Mississippi'), ('MO', 'Missouri'),
    ('MT', 'Montana'), ('NE', 'Nebraska'), ('NV', 'Nevada'),
    ('NH', 'New Hampshire'), ('NJ', 'New Jersey'), ('NM', 'New Mexico'),
    ('NY', 'New York'), ('NC', 'North Carolina'), ('ND', 'North Dakota'),
    ('OH', 'Ohio'), ('OK', 'Oklahoma'), ('OR', 'Oregon'),
    ('PA', 'Pennsylvania'), ('RI', 'Rhode Island'), ('SC', 'South Carolina'),
    ('SD', 'South Dakota'), ('TN', 'Tennessee'), ('TX', 'Texas'),
    ('UT', 'Utah'), ('VA', 'Virginia'), ('VT', 'Vermont'),
    ('WA', 'Washington'), ('WI', 'Wisconsin'), ('WV', 'West Virginia'),
    ('WY', 'Wyoming');

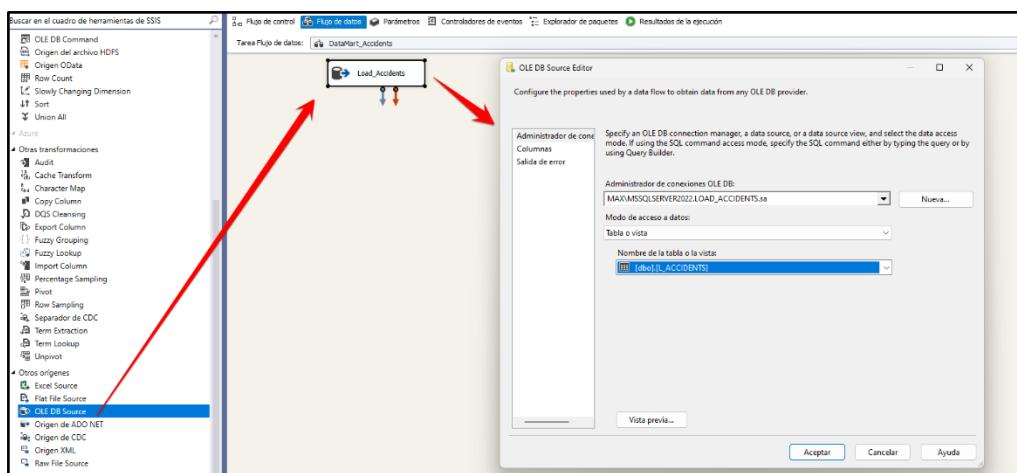
```

### III. PASO II: DE LOAD\_ACCIDENTS a STAGING\_ACCIDENTS (LIMPIEZA DE DATOS)

Creamos un nuevo proyecto donde se limpiará y se cargarán los datos y arrastramos una Tarea de Flujo de Datos:

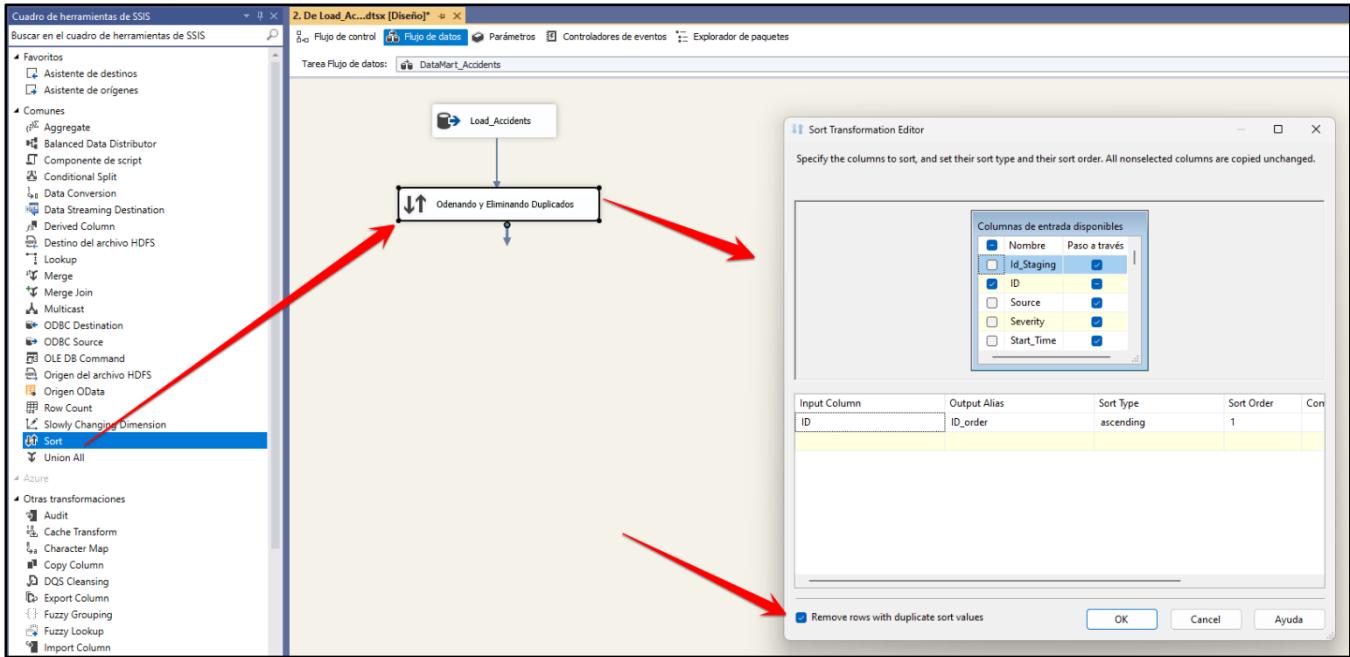


Creamos un origen de datos OLE DB referenciando a la base de datos LOAD\_ACCIDENTS:



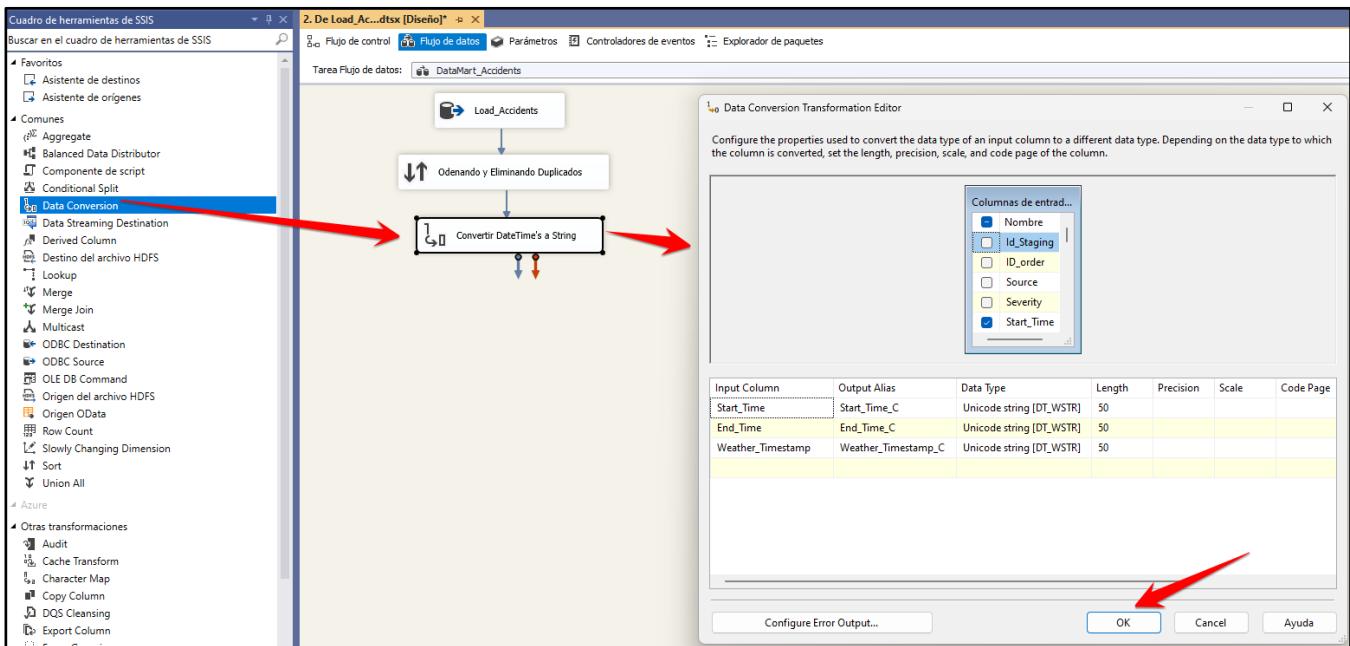
## 1. Eliminar Duplicados.

Para asegurarnos que hay datos únicos, procedemos a eliminar registros que se puedan duplicar mediante el componente **Sort**:



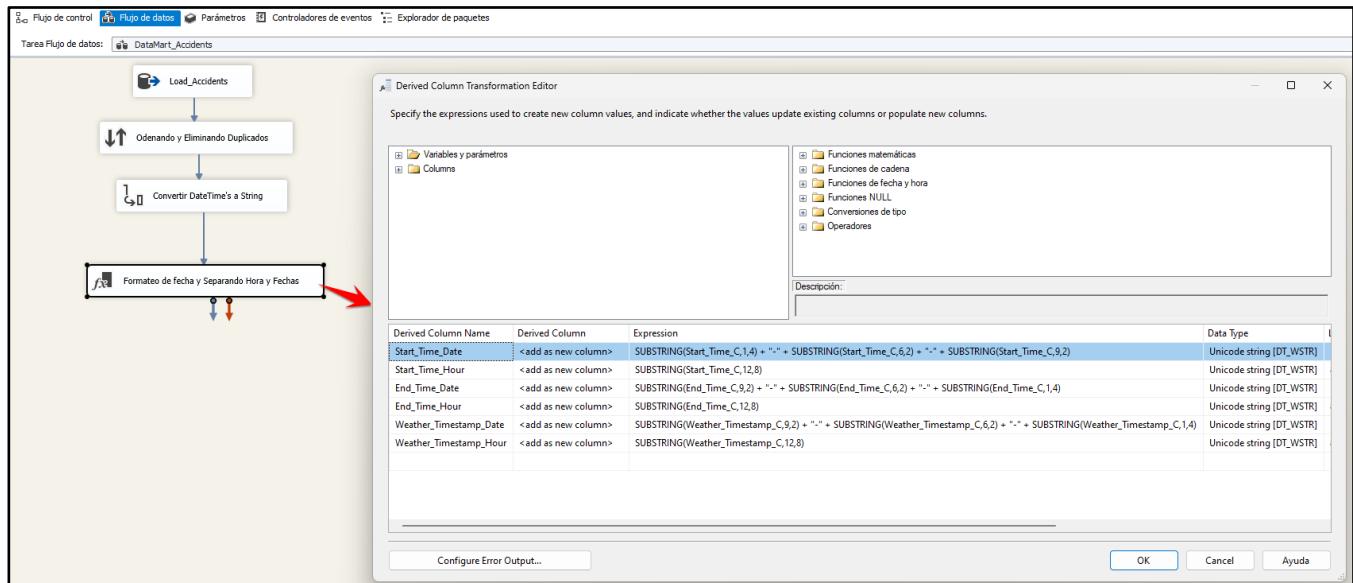
## 2. Conversión de datos.

Como tenemos fechas y horas en formato DateTime, queremos dividir en el siguiente paso las fechas y las horas en diferentes columnas, por lo que convertiremos a String y luego esa cadena extraeremos parte del texto para pasarlo a diferentes columnas, usará el componente de **Data Conversion**.



### 3. Formateo de las fechas y horas.

Para asegurarnos que las fechas estén bien estructuradas y no tengan errores, es necesario asegurar de que todos los registros tengan el mismo formato y las fechas sean correctas, también dividiremos las horas de las fechas, todo esto lo haremos mediante el componente **Derive Column**:



### 4. Tratando Registros sin data / Nulos.

En este paso **verificamos los registros NULOS o datos faltantes** presentes en nuestra Data, para ello exploramos en SQL Server en qué columnas de la Tabla LOAD\_ACCIDENTS no poseen data, para ello aplíqué la siguiente consulta:

```
----- Creando Variables para usar una consulta dinámica
DECLARE @TableName NVARCHAR(MAX) = 'L_ACCIDENTS';
DECLARE @SQL NVARCHAR(MAX);

-- Construcción dinámica de la consulta
SET @SQL =
    'SELECT ColumnName, SUM(NullCount) AS NullCount ' +
    'FROM (' +
    STUFF((SELECT
        ' UNION ALL SELECT ''' + c.name + ''' AS ColumnName, COUNT(1) AS NullCount ' +
        'FROM ' + @TableName + ' WHERE [' + c.name + '] IS NULL ' +
        CASE
            -- Validación adicional si es de tipo DATETIME
            WHEN c.system_type_id IN (61, 58) THEN
                'OR TRY_CAST([' + c.name + '] AS DATETIME) IS NULL '
            -- Para columnas de texto
            WHEN c.system_type_id IN (167, 175, 231, 239) THEN
                'OR [' + c.name + '] = '''' '
            ELSE
                '' -- No se agrega validación extra para otros tipos de datos
        END
        FROM sys.columns c
        INNER JOIN sys.tables t ON c.object_id = t.object_id
        WHERE t.name = @TableName
        FOR XML PATH(''), TYPE).value('.','NVARCHAR(MAX)', 1, 11, '') +
    ') AS Results ' +
    'GROUP BY ColumnName ' +
    'ORDER BY NullCount DESC';

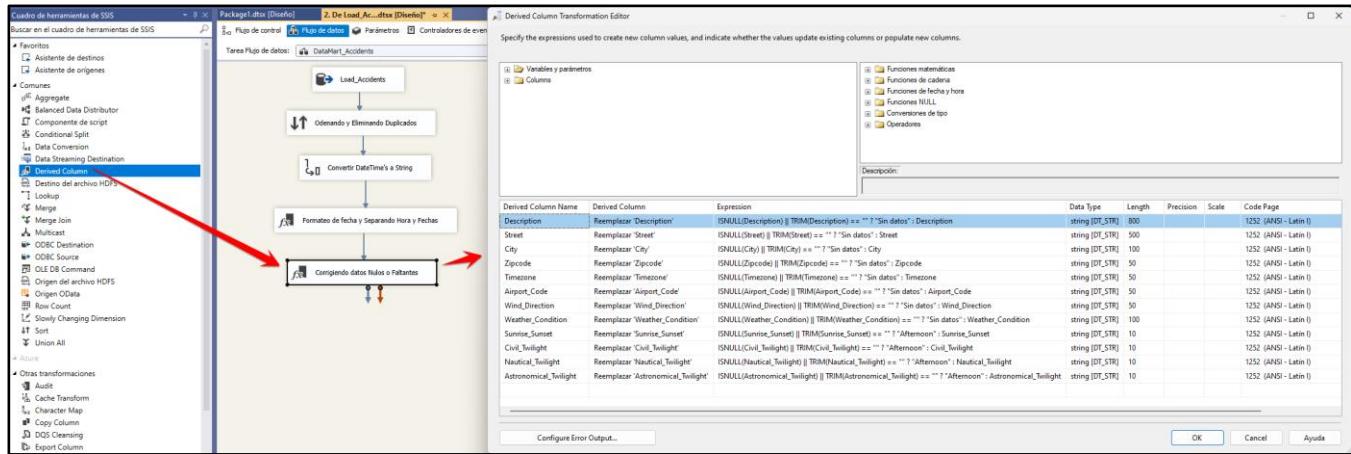
-- Ejecución de la consulta resultante
EXEC sp_executesql @SQL;
```

Al ejecutar validamos que, sí existen columnas con registros nulos o faltantes y al ordenarse de mayor a menor, se ve que hay **12 columnas** que debemos tratar los datos faltantes:

	ColumnName	NullCount
1	Wind_Direction	175206
2	Weather_Condition	173459
3	Sunrise_Sunset	23246
4	Civil_Twilight	23246
5	Nautical_Twilight	23246
6	Astronomical_Twilight	23246
7	Airport_Code	22635
8	Street	10869
9	Timezone	7808
10	Zipcode	1915
11	City	253
12	Description	5
13	Country	0
14	County	0
15	State	0
16	Id_Staging	0
17	ID	0
18	Source	0
19	Severity	0
20	Start_Time	0
21	End_Time	0
22	Start_Lat	0
23	Start_Lng	0
24	End_Lat	0
25	End_Lng	0
26	Distance_mi	0
27	Weather_Timestamp	0
28	Temperature_F	0
29	Wind_Chill_F	0
30	Humidity	0
31	Pressure_in	0
32	Visibility_mi	0
33	Amenity	0
34	Bump	0
35	Crossing	0
36	Give_Way	0
37	Junction	0
38	No_Exit	0
39	Railway	0
40	Roundabout	0
41	Station	0
42	Stop	0
43	Traffic_Calming	0
44	Traffic_Signal	0
45	Turning_Loop	0
46	Wind_Speed_mph	0
47	Precipitation_in	0

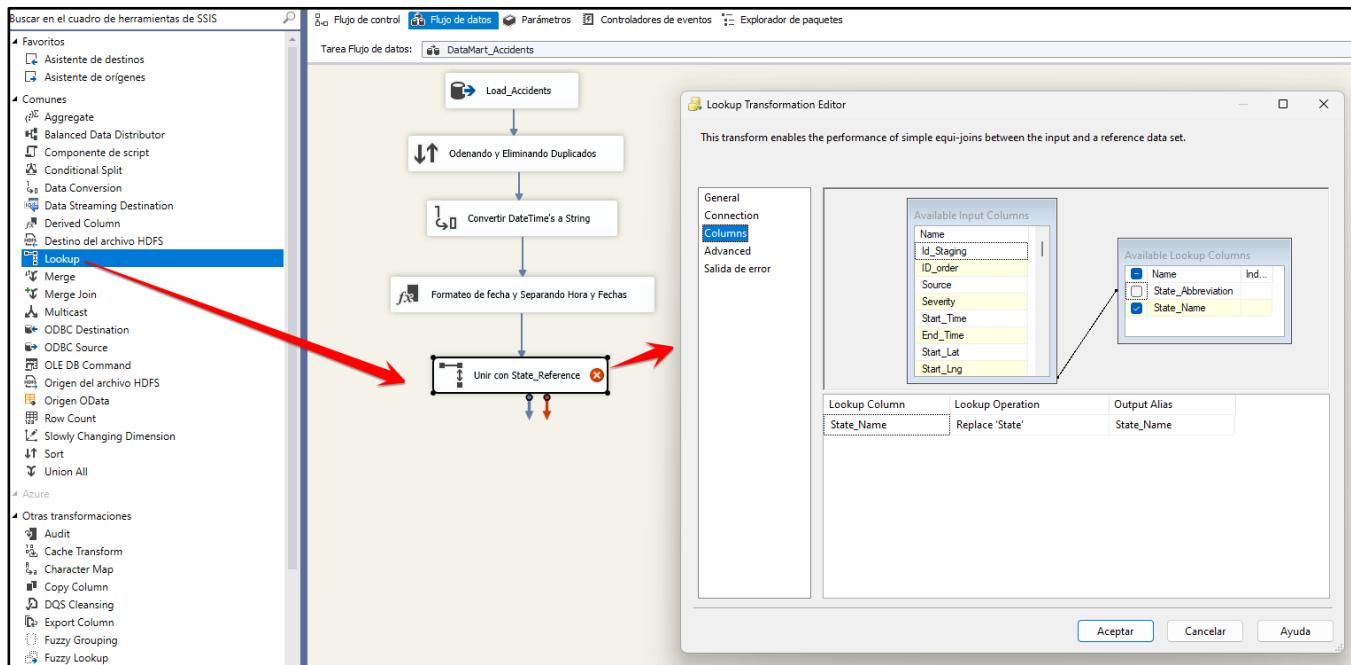


Luego de verificar, utilicé el componente de **DerivedColumn** para reemplazar los valores nulos o faltantes por datos entendibles:



## 5. Normalizar / Reemplazar Valores (Abreviaturas de Estados):

Como mencioné anteriormente, la columna de “**State**” contiene datos de los estados en forma abreviada, se requiere que estén los nombres completos de los estados, para ello se creó la tabla “**STATE\_REFERENCE**” que usaremos junto al componente de **Lookup** para dicho proceso.



## 6. Verificar la validez de las coordenadas.

El campo de Latitud y Longitud debe estar comprendido por valores válidos, la longitud debe estar entre -90º y +90º, en cuanto a la Longitud debe comprender entre -180º y +180º. Para ello verificamos en SQL Server si existen valores fuera de ese campo:

```

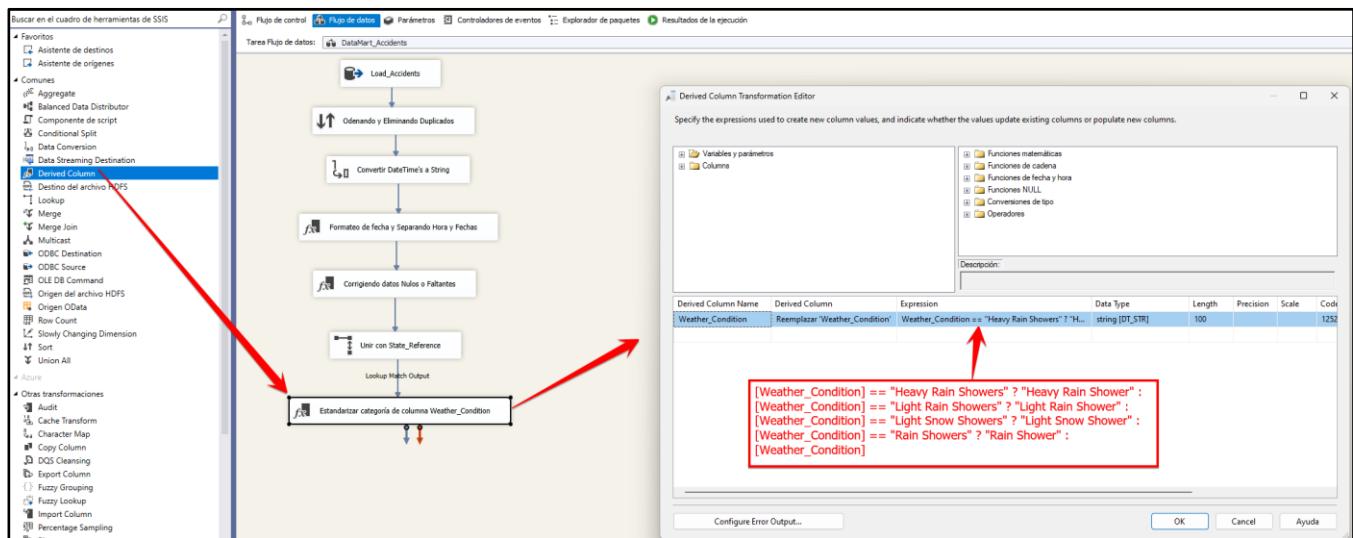
143 select * from STAGING_ACCIDENTS
144 WHERE (Start_Lat < -90 OR Start_Lat > 90) or
145     (Start_Lng < -180 OR Start_Lng > 180)

```

En este caso observamos que ningún valor está fuera de ese rango, por lo que continuaré al siguiente paso.

## 7. Estandarizar Variables categóricas de columnas.

En este paso verifiqué si los datos de las columnas se podían estandarizar, viendo que en la columna **Weather\_Condition** existían las mismas categorías solo que algunas en singular y otras en plural, utilizando el componente de **Derived Column** se reemplazaron los datos en plural, por singular para tener un estándar en los datos.



## 8. Conversión de datos de Cadena a Date y Time.

Las columnas que separamos de fecha y hora, las convertimos a tipo de datos: date y time, para luego ser cargados en la Base de datos Destino.

## 9. Creamos una Base de Datos Destino.

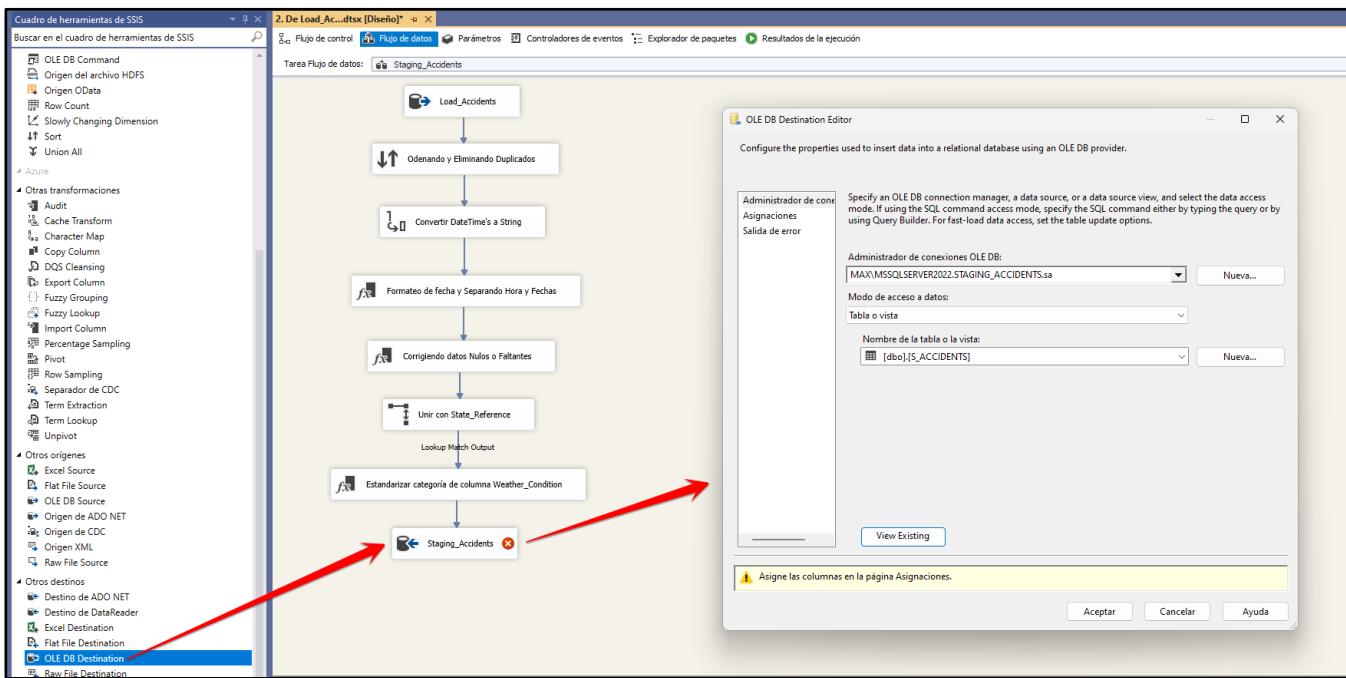
- Creé una Base de datos llamada **STAGING\_ACCIDENTS**, con una tabla **S\_ACCIDENTS**, que será una copia de la tabla L\_Accidents de la Base de datos de Load\_Accidents, solo que añadido las 6 columnas donde sepáramos las fechas y horas de 3 columnas.

```

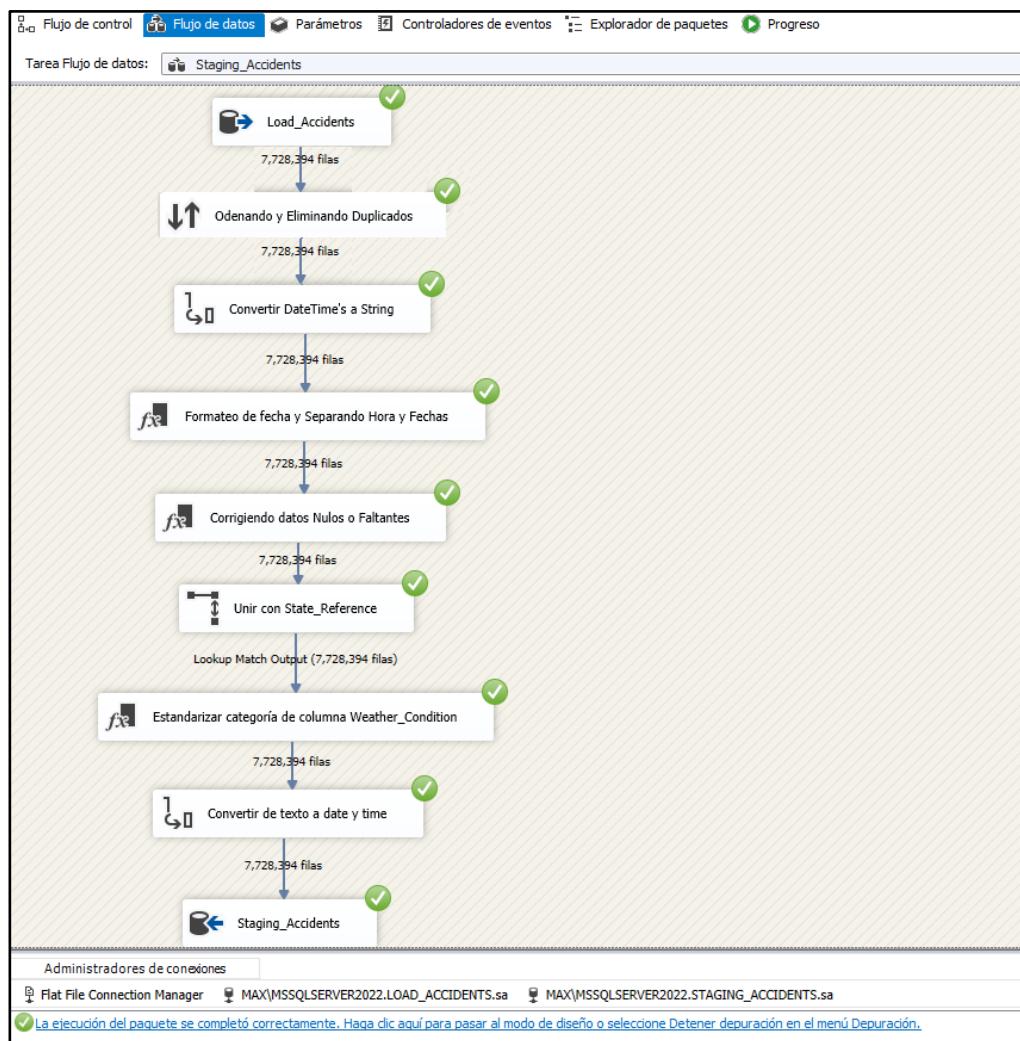
CREATE DATABASE STAGING_ACCIDENTS
/*CREANDO LA TABLA L_ACCIDENTS*/
CREATE TABLE S_ACCIDENTS (
    Id_Staging int identity primary key not null,
    ID varchar(50),
    Source VARCHAR(50),
    Severity int,
    Start_Time DATETIME2,
    End_Time DATETIME2,
    Start_Lat FLOAT,
    Start_Lng FLOAT,
    End_Lat FLOAT,
    End_Lng FLOAT,
    Distance_mi FLOAT,
    Description VARCHAR(800),
    Street VARCHAR(500),
    City VARCHAR(100),
    County VARCHAR(100),
    State VARCHAR(50),
    Zipcode VARCHAR(50),
    Country VARCHAR(50),
    Timezone VARCHAR(50),
    Airport_Code VARCHAR(50),
    Weather_Timestamp DATETIME2,
    Temperature_F FLOAT,
    Wind_Chill_F FLOAT,
    Humidity FLOAT,
    Pressure_in FLOAT,
    Visibility_mi FLOAT,
    Wind_Direction VARCHAR(50),
    Wind_Speed_mph FLOAT,
    Precipitation_in FLOAT,
    Weather_Condition VARCHAR(100),
    Amenity VARCHAR(10),
    Bump VARCHAR(10),
    Crossing VARCHAR(10),
    Give_Way VARCHAR(10),
    Junction VARCHAR(10),
    No_Exit VARCHAR(10),
    Railway VARCHAR(10),
    Roundabout VARCHAR(10),
    Station VARCHAR(10),
    Stop VARCHAR(10),
    Traffic_Calming VARCHAR(10),
    Traffic_Signal VARCHAR(10),
    Turning_Loop VARCHAR(10),
    Sunrise_Sunset VARCHAR(10),
    Civil_Twilight VARCHAR(10),
    Nautical_Twilight VARCHAR(10),
    Astronomical_Twilight VARCHAR(10),
    Start_Time_Date date,
    Start_Time_Hour time,
    End_Time_Date date,
    End_Time_hour time,
    Weather_Timestamp_Date date,
    Weather_Timestamp_Hour time
);

```

- ii) Se arrastra un componente de OLE DB Destination y elegimos la tabla S\_ACCIDENTS como Destino.

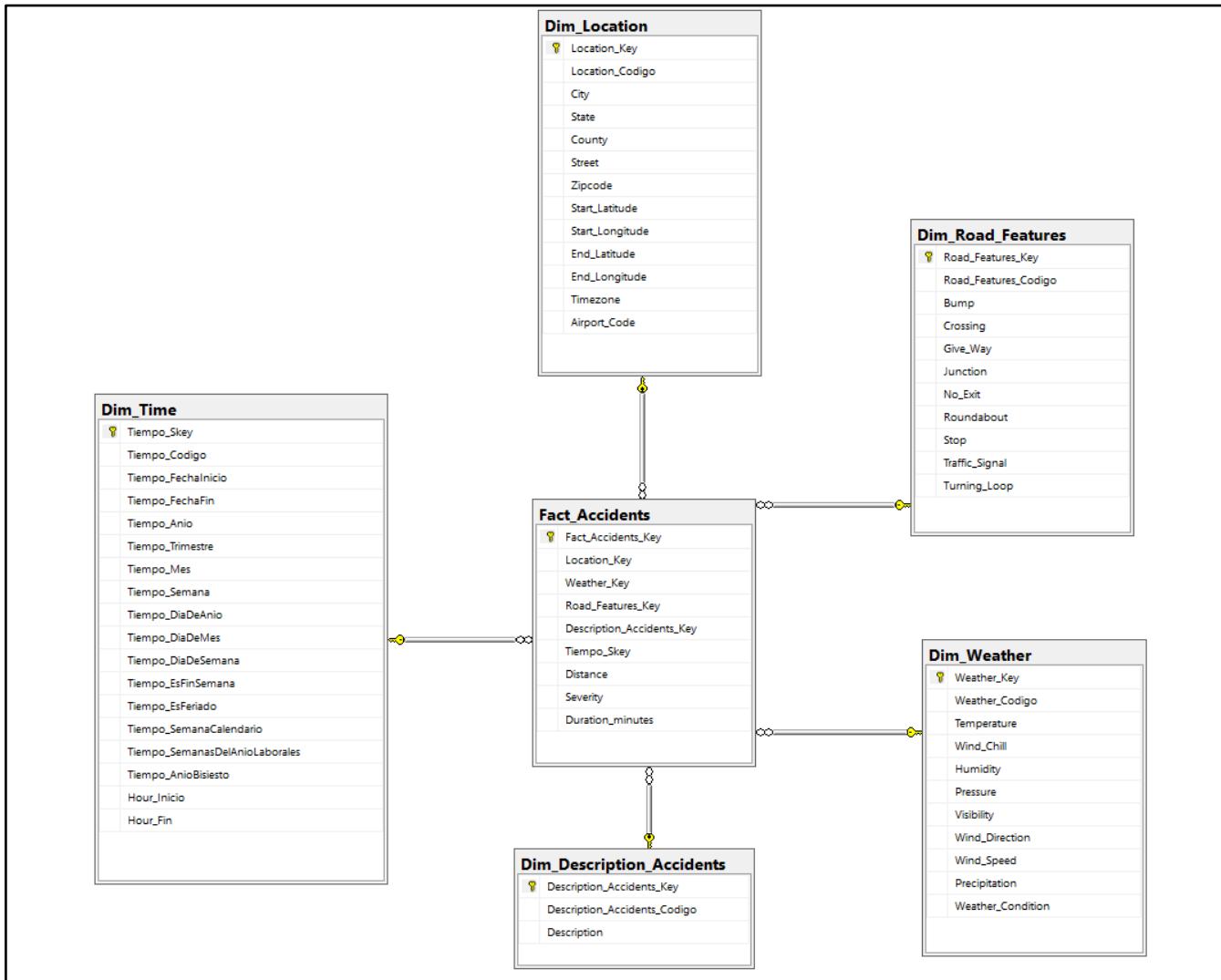


### iii) Ejecutamos el proyecto.



#### IV. PASO III: DE STAGING\_ACCIDENTS A DATAMART\_ACCIDENTS.

El esquema del Datamart será el de **Estrella** (**Se adjunta el modelo en la imagen**), en primer lugar, crearé la base de datos **DATAMART\_ACCIDENTS**, y continuaré creando las dimensiones y la tabla hechos. Luego **crearé también una tabla para cada dimensión dentro de BD STAGING\_ACCIDENTS** la cual guardará los **registros modificados de cada dimensión**, por ello, en el poblamiento de cada dimensión se tendrá 2 orígenes de datos, el primero para sacar los datos de la tabla **S\_ACCIDENTS** y el segundo de la Dimensión correspondiente que poblaremos, esto con la finalidad de comparar registros y guardar los registros modificados en las tablas mencionadas anteriormente.



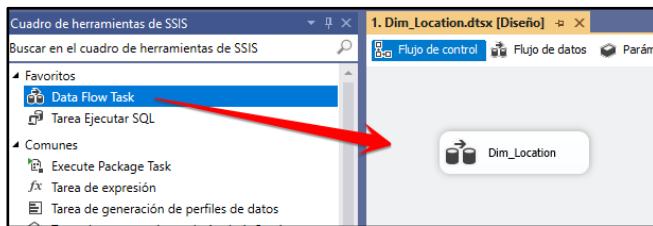
##### 1. Poblando la Dimensión Ubicación:

###### a) Creación de la Tabla **Dim\_Location**.

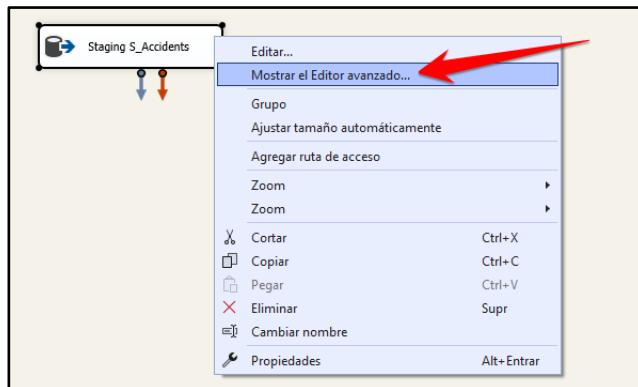
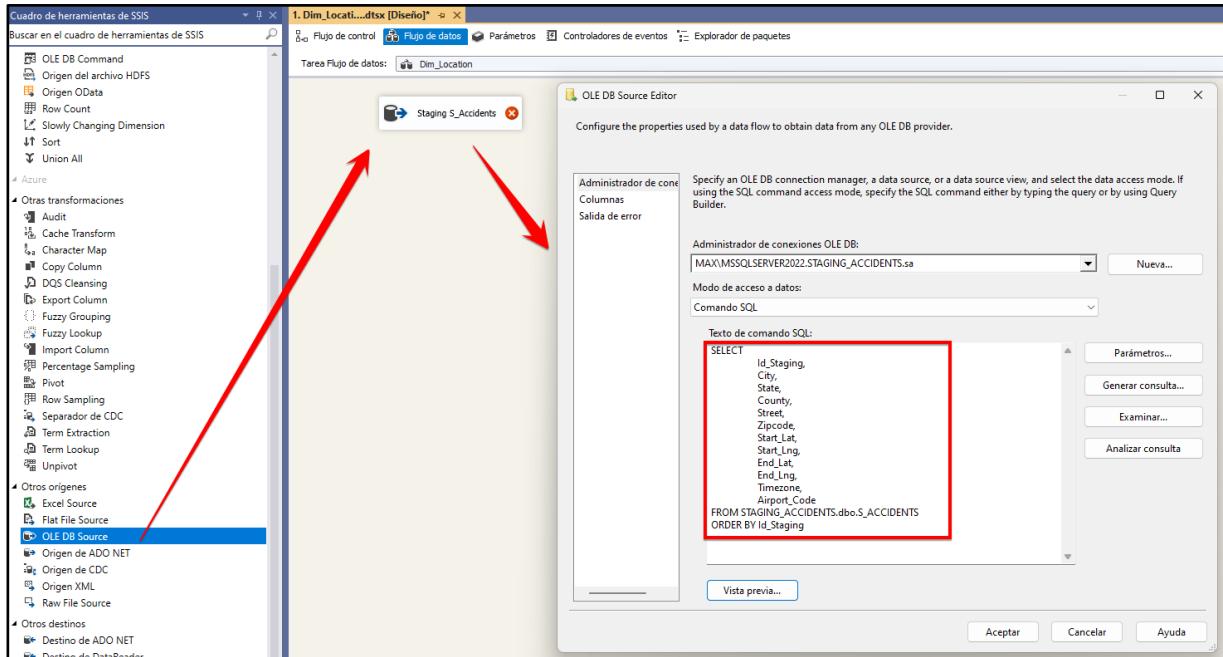
```

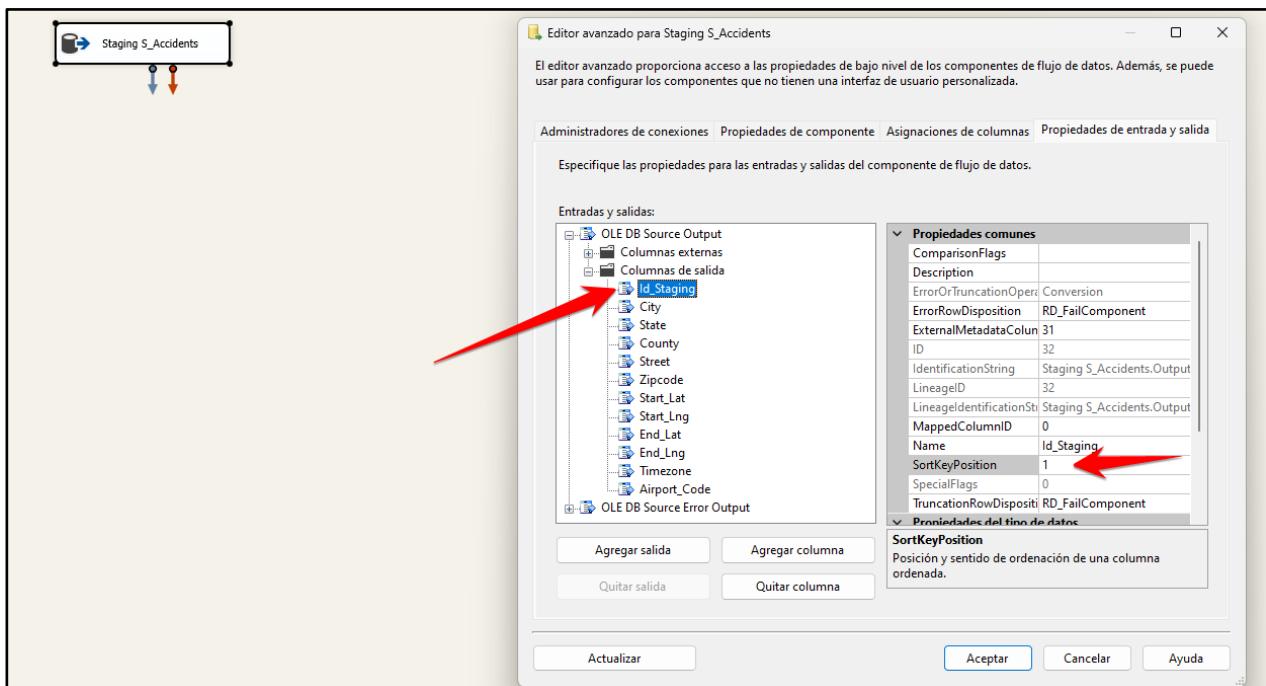
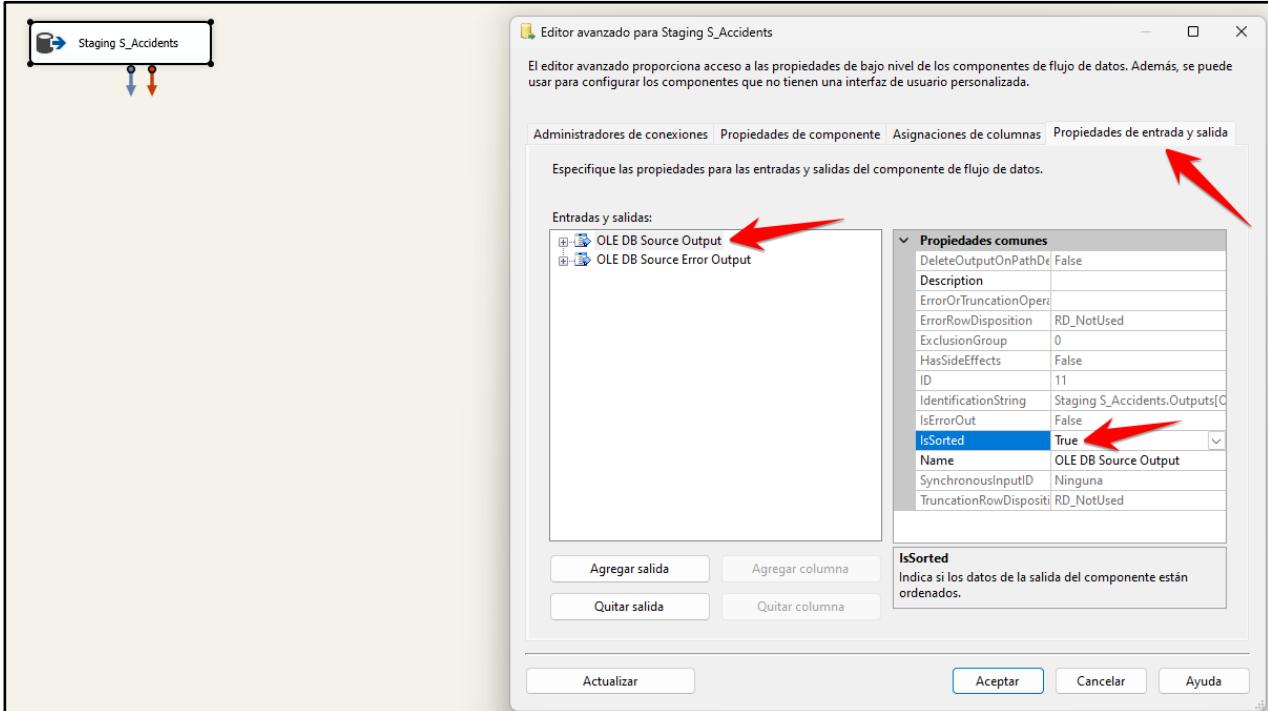
-- Dimensión: Ubicación
CREATE TABLE Dim_Location (
    Location_Key INT PRIMARY KEY IDENTITY(1,1),
    Location_Codigo INT,
    City VARCHAR(100),
    State VARCHAR(50),
    County VARCHAR(100),
    Street VARCHAR(500),
    Zipcode VARCHAR(50),
    Start_Latitude FLOAT,
    Start_Longitude FLOAT,
    End_Latitude FLOAT,
    End_Longitude FLOAT,
    Timezone VARCHAR(50),
    Airport_Code VARCHAR(50)
);
    
```

- b) En el proyecto de Dim\_Location utilizaré una tarea de flujo de datos.

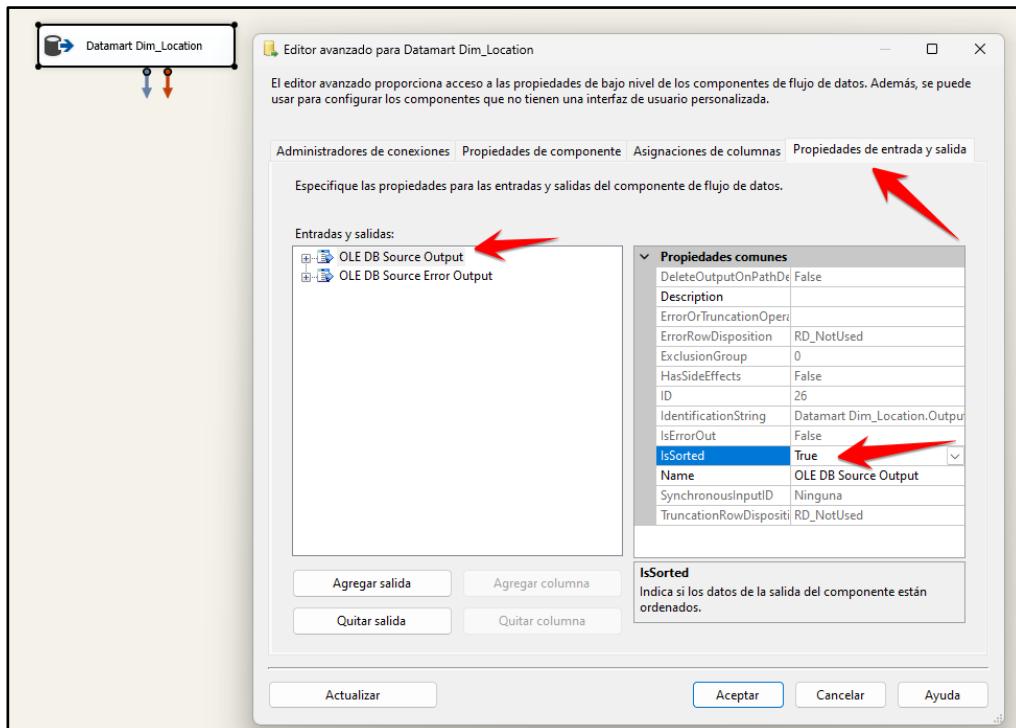
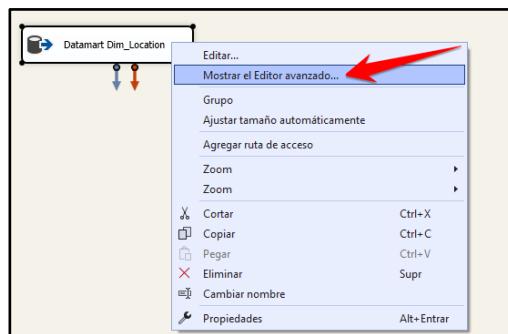
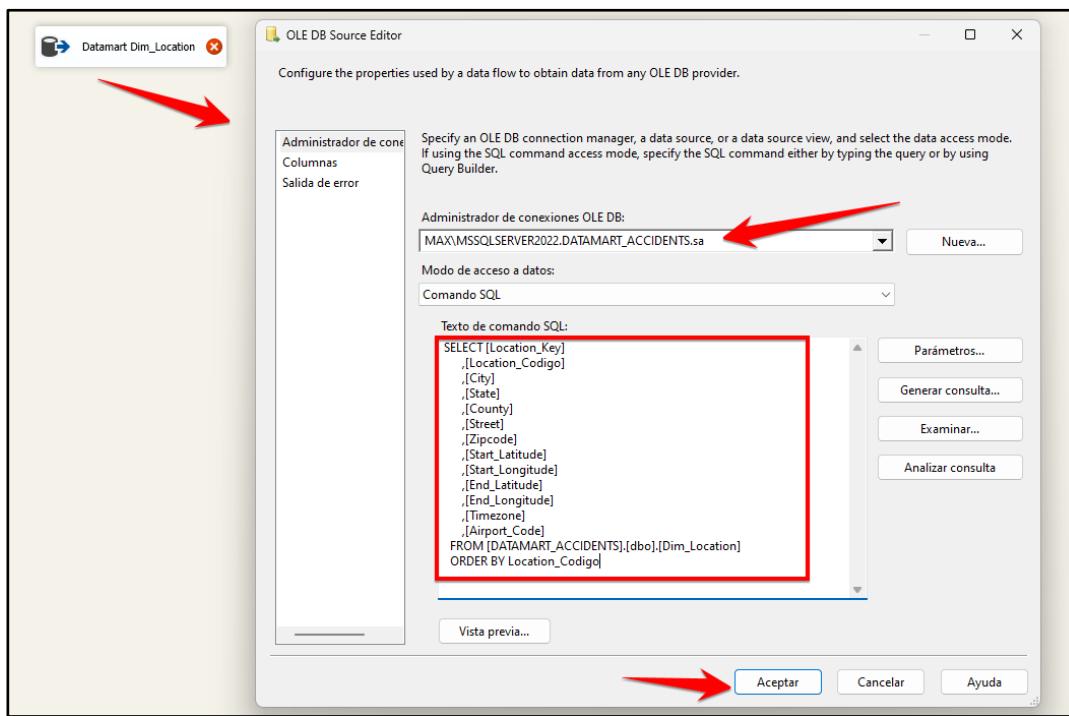


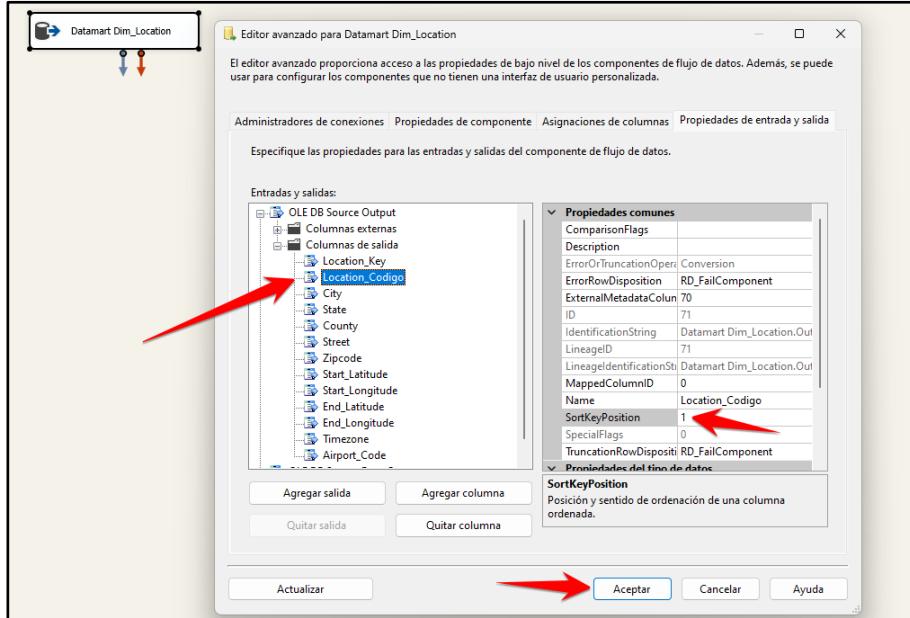
- c) Configurando primer origen de Datos S\_ACCIDENTS.



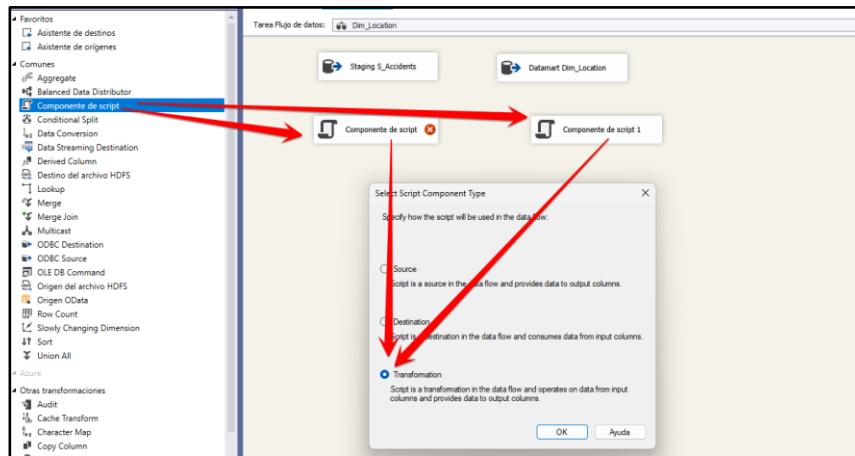


d) Configurando segundo origen de Datos **DIM\_LOCATION**.

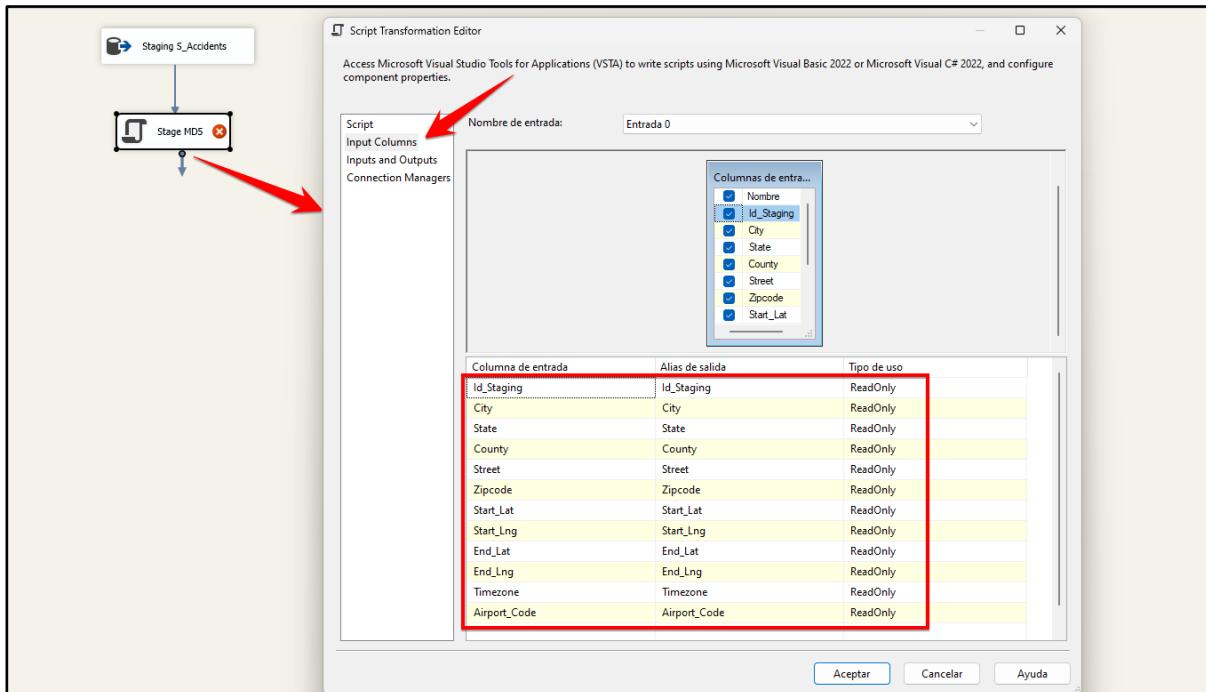


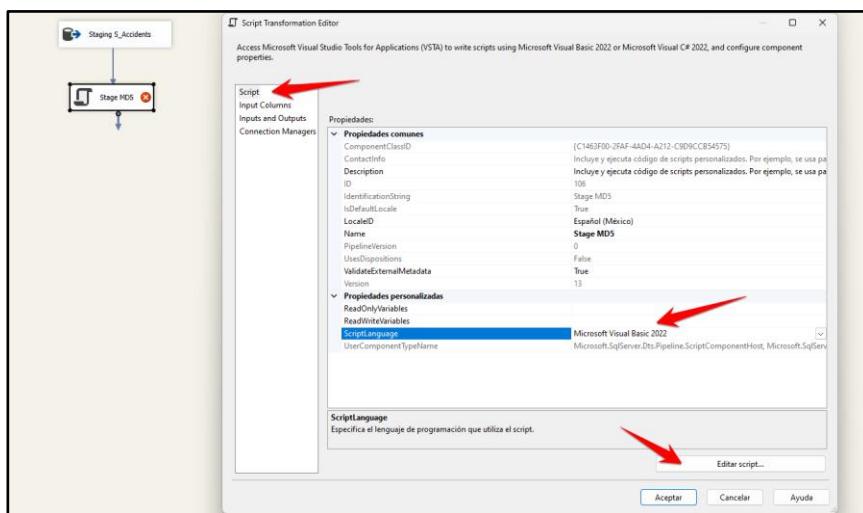
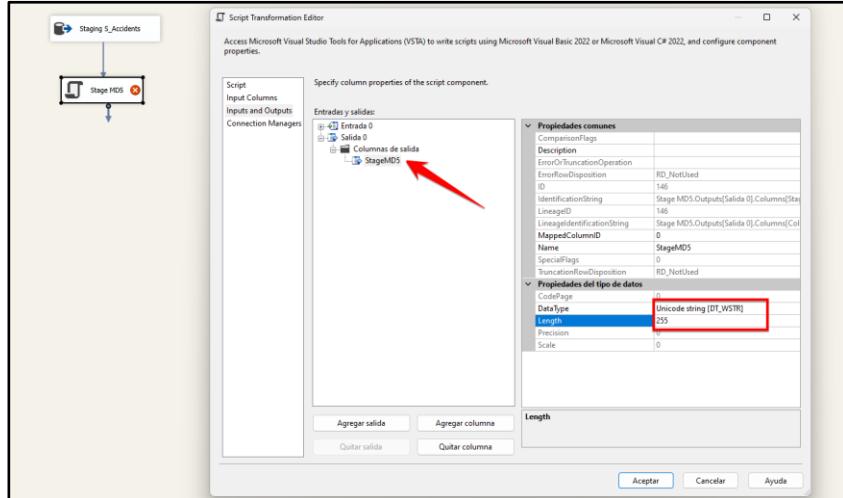


e) Emplearé dos Componentes Scripts con el tipo Transformación.



f) Configurando el Primer Componente:

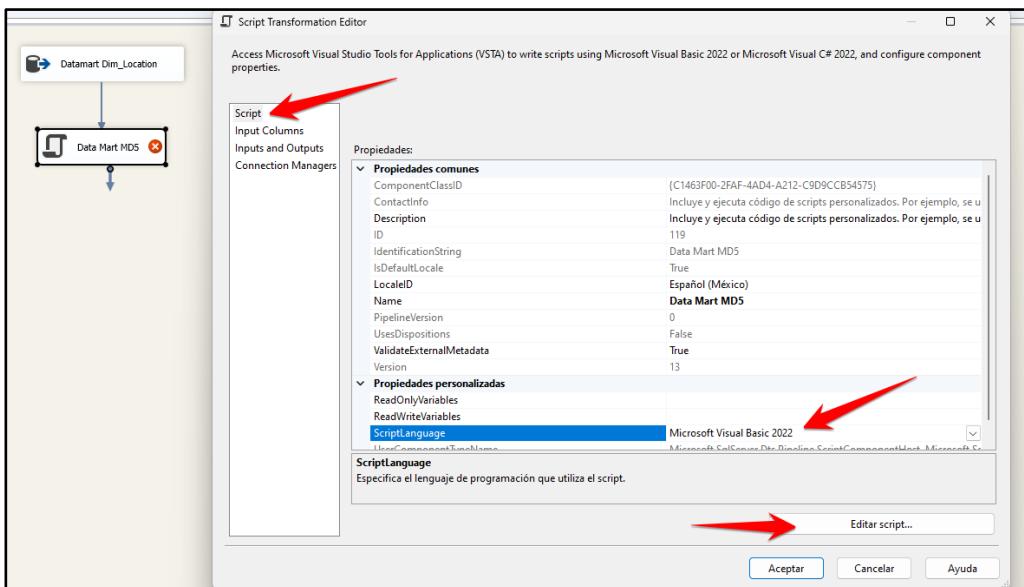
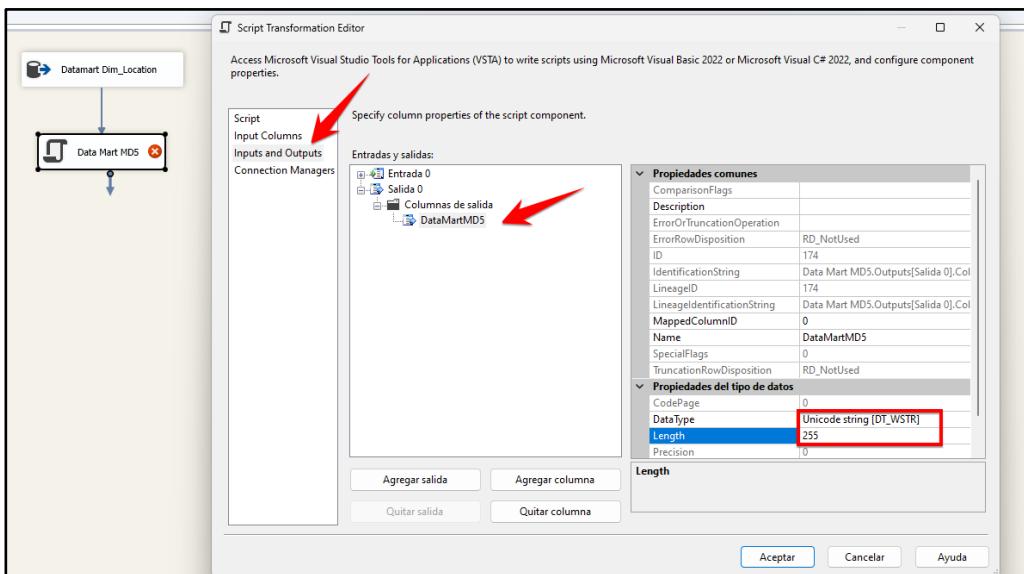
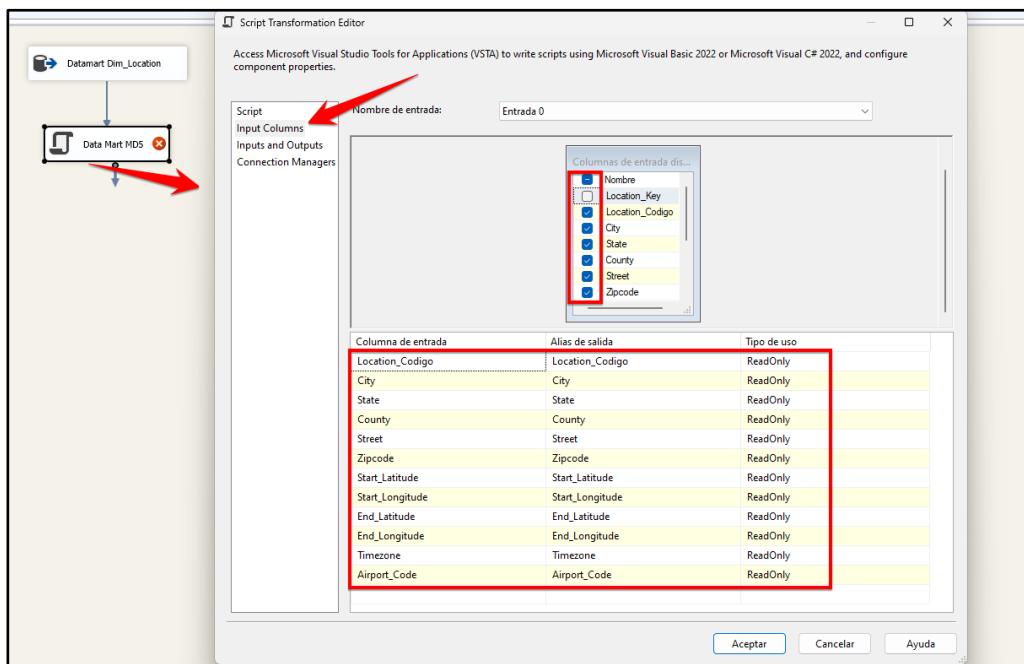




```
main.vb  o  X
SC_e7a23d5f4ed4de2918754d9d898023
1  > Help: Introduction to the Script Component
2
3
4  #Region "Imports"
5  Imports System
6  Imports System.Data
7  Imports System.Math
8  Imports Microsoft.SqlServer.Dts.Pipeline.Wrapper
9  Imports Microsoft.SqlServer.Dts.Runtime.Wrapper
10 Imports Microsoft.SqlServer.Dts.Pipeline
11 Imports System.Text
12 Imports System.Security.Cryptography
13
14 #End Region
```

```
109 Private inputBuffer As PipelineBuffer
110
111 0 referencias
112 Public Overrides Sub ProcessInput(ByVal InputID As Integer, ByVal Buffer As Microsoft.SqlServer.Dts.Pipeline.PipelineBuffer)
113
114     inputBuffer = Buffer
115
116     MyBase.ProcessInput(InputID, Buffer)
117
118 End Sub
119
120 1 referencia
121 Public Shared Function CreateHash(ByVal data As String) As String
122     Dim dataToHash As Byte() = (New UnicodeEncoding()).GetBytes(data)
123     Dim md5 As MD5 = New MD5CryptoServiceProvider()
124     Dim hashedData As Byte() = md5.ComputeHash(dataToHash)
125     RNGCryptoServiceProvider.Create().GetBytes(hashedData)
126     Dim s As String = Convert.ToBase64String(hashedData, Base64FormattingOptions.None)
127     Return s
128 End Function
129
130 2 referencias
131 Public Overrides Sub Entrada0_ProcessInputRow(ByVal Row As Entrada0Buffer)
132     Dim counter As Integer = 0
133     Dim values As New StringBuilder
134     For counter = 0 To inputBuffer.ColumnCount - 1
135         Dim value As Object
136         value = inputBuffer.Item(counter)
137         values.Append(value)
138     Next
139
140     'CAMBIAR EL VALOR VariableSalida A SU COLUMNA DE SALIDA
141     Row.StageMD5 = CreateHash(values.ToString())
142 End Sub
143
End Class
```

g) Configurando el Segundo Componente.



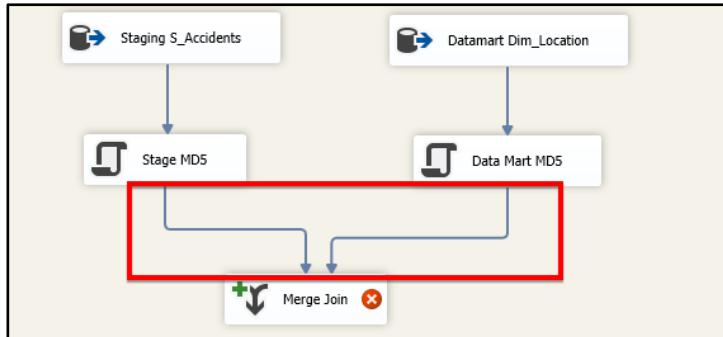
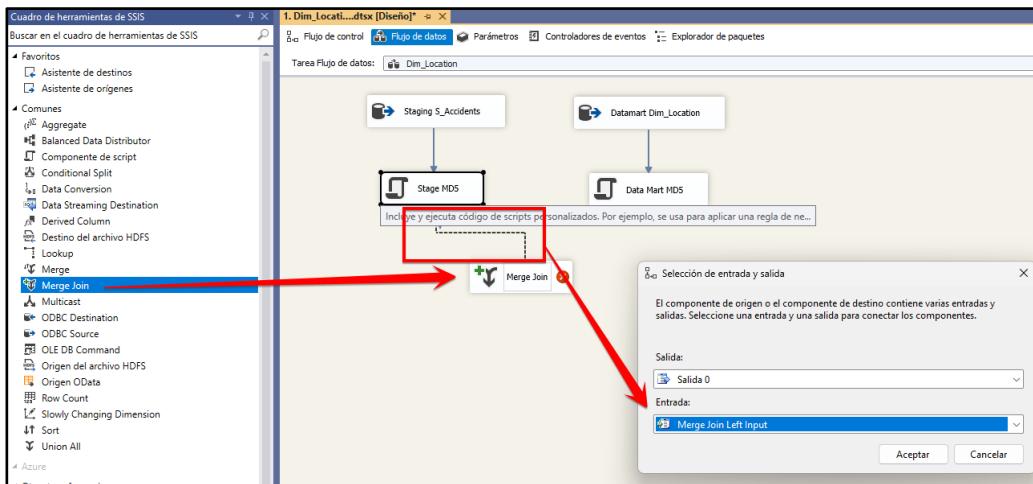
```

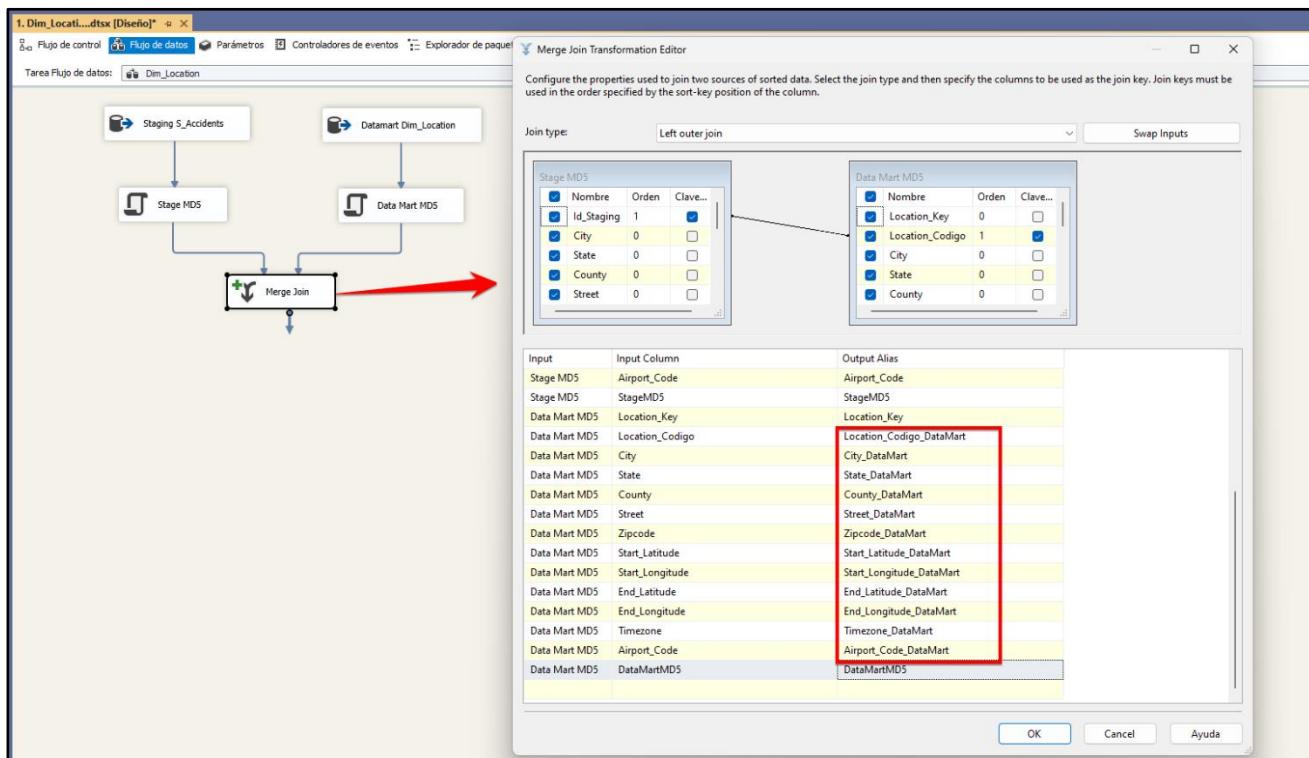
8
9     #Region "Imports"
10    Imports System
11    Imports System.Data
12    Imports System.Math
13    Imports Microsoft.SqlServer.Dts.Pipeline.Wrapper
14    Imports Microsoft.SqlServer.Dts.Runtime.Wrapper
15    Imports Microsoft.SqlServer.Dts.Pipeline
16    Imports System.Text
17    Imports System.Security.Cryptography
18  #End Region

106   ' Row.ZipCode = zipCode
107
108  Private inputBuffer As PipelineBuffer
109
110  Public Overrides Sub ProcessInput(ByVal InputID As Integer, ByVal Buffer As Microsoft.SqlServer.Dts.Pipeline.PipelineBuffer)
111
112      inputBuffer = Buffer
113
114      MyBase.ProcessInput(InputID, Buffer)
115
116  End Sub
117
118  1 referencia
119  Public Shared Function CreateHash(ByVal data As String) As String
120      Dim dataToHash As Byte() = (New UnicodeEncoding()).GetBytes(data)
121      Dim md5 As MD5 = New MD5CryptoServiceProvider()
122      Dim hashedData As Byte() = md5.ComputeHash(dataToHash)
123      RNGCryptoServiceProvider.Create().GetBytes(hashedData)
124      Dim s As String = Convert.ToString(hashedData, Base64FormattingOptions.None)
125      Return s
126  End Function
127
128  2 referencias
129  Public Overrides Sub Entrada0_ProcessInputRow(ByVal Row As Entrada0Buffer)
130      Dim counter As Integer = 0
131      Dim values As New StringBuilder
132      For counter = 0 To inputBuffer.ColumnCount - 1
133          Dim value As Object
134          value = inputBuffer.Item(counter)
135          values.Append(value)
136
137      'CAMBIAR EL VALOR VariableSalida A SU COLUMNA DE SALIDA
138      Row.DataMartMD5 = CreateHash(values.ToString())
139
140  End Sub
141
End Class

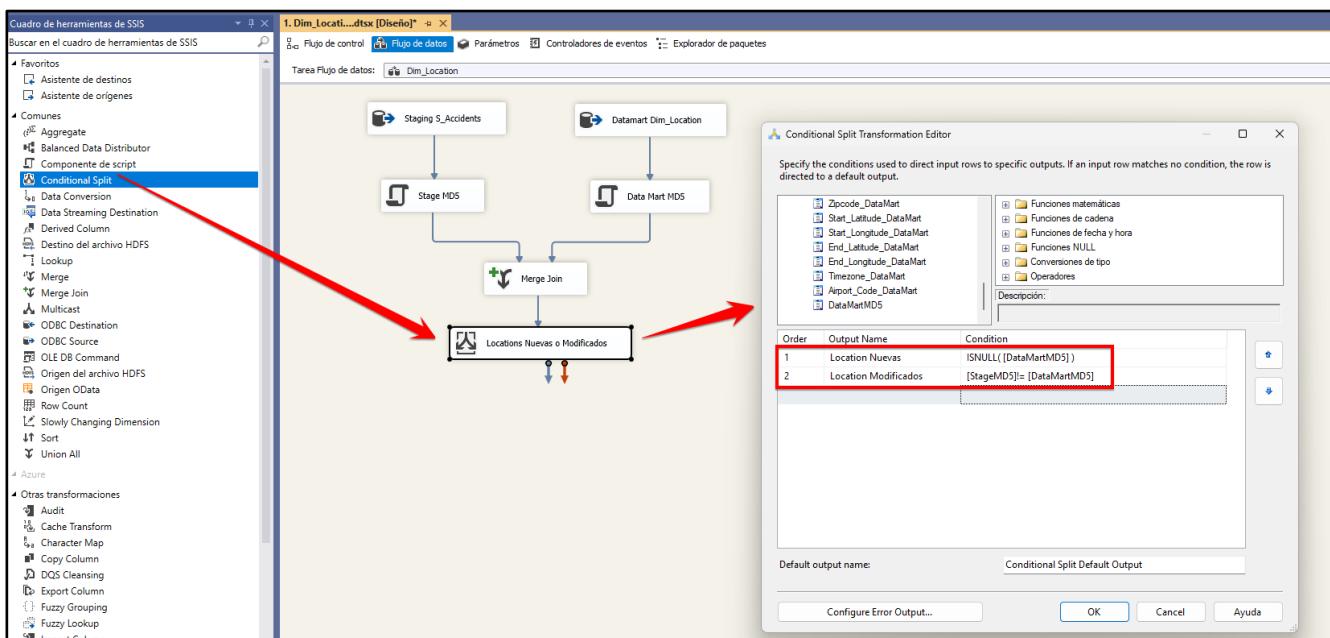
```

#### h) Utilizamos el componente de Combinación de Mezcla o Merge Join.





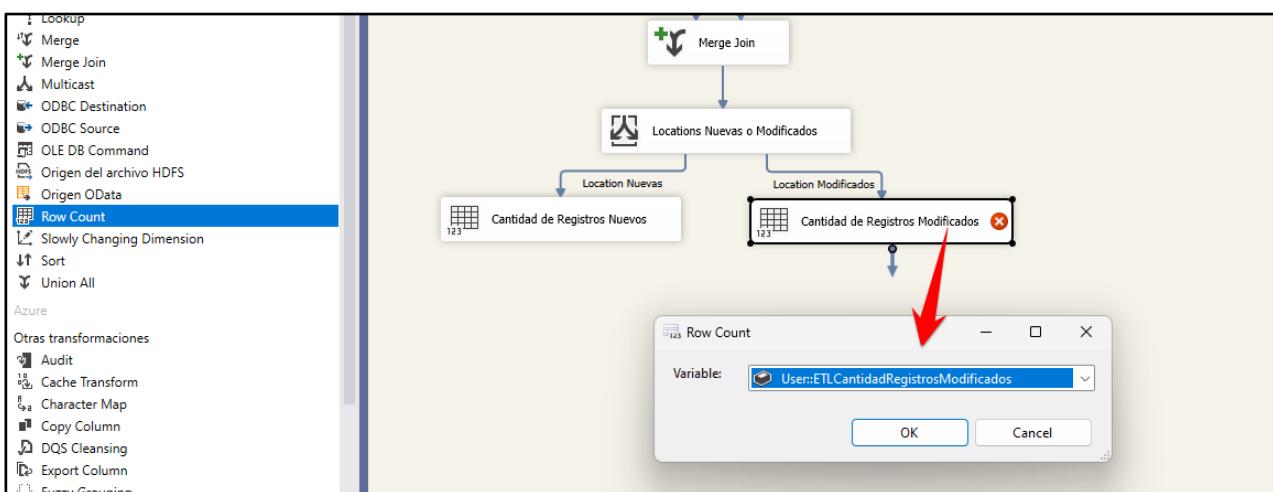
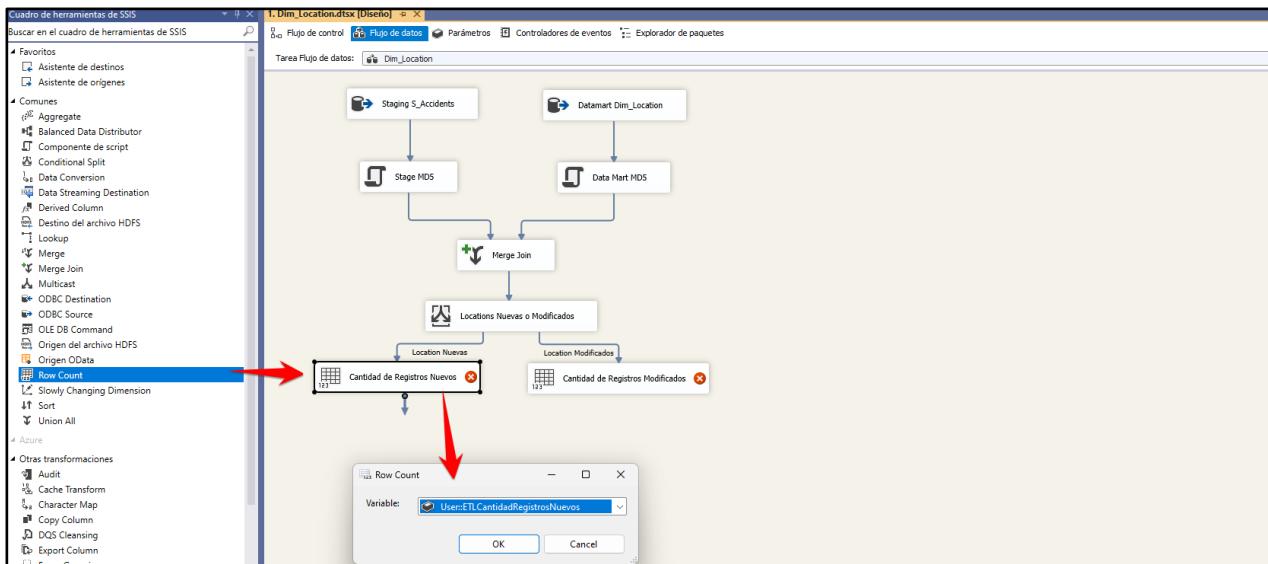
- i) Utilizaré el componente de **División Condicional**, esto para ir comparando los datos y agregándolos a Registros nuevos o Registros Modificados según corresponda.



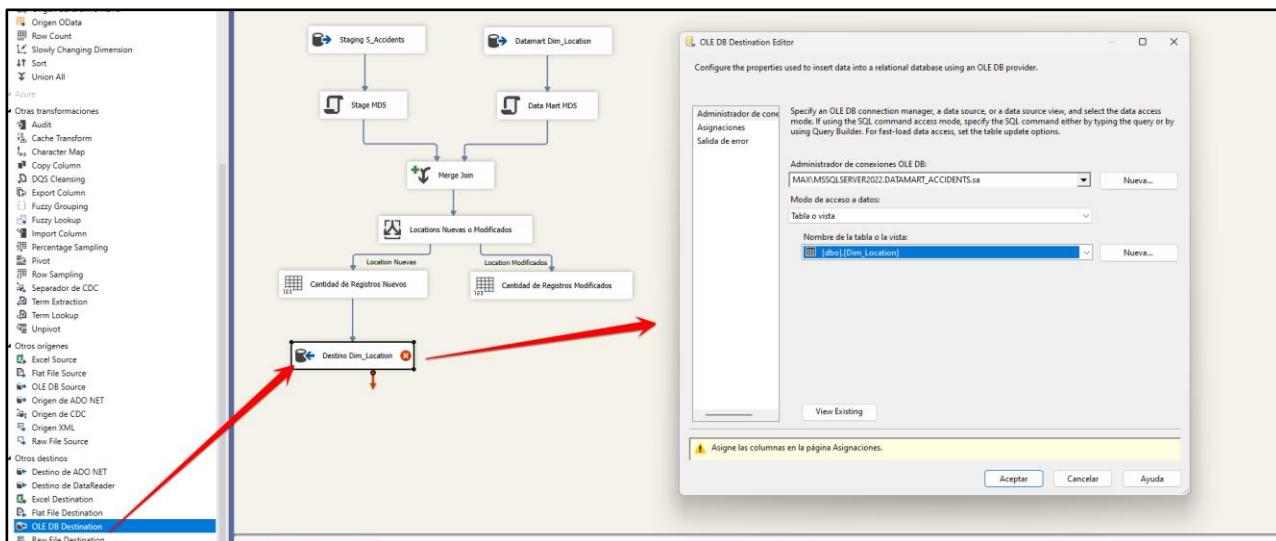
- j) Creé dos variables para usarlas en el siguiente paso:

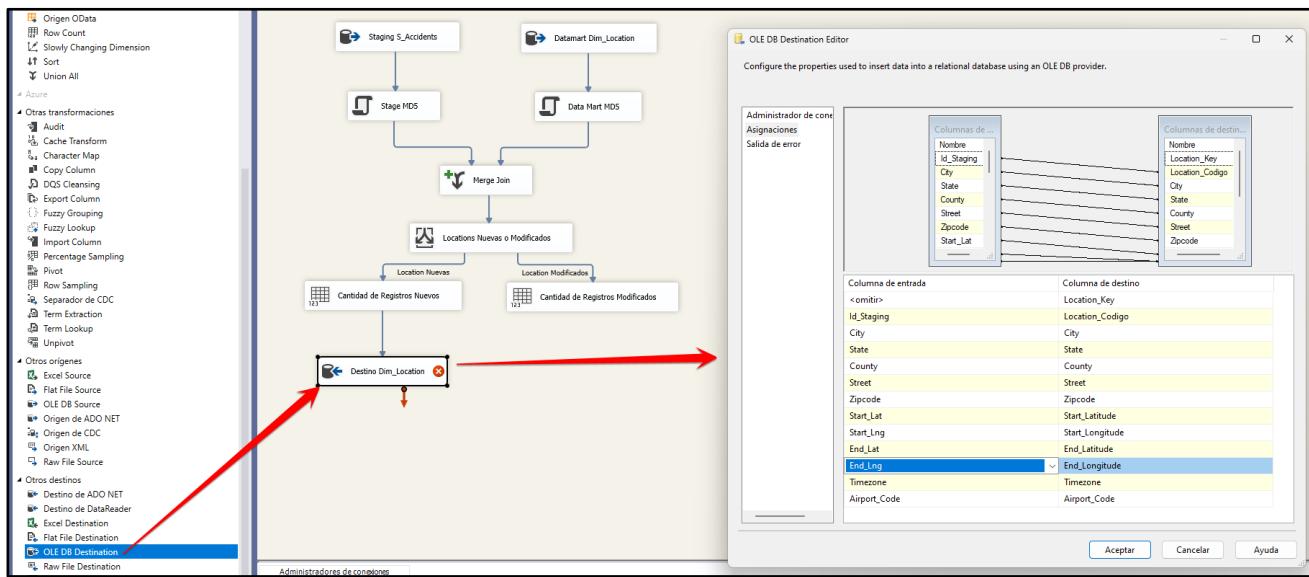
Variables				
Nombre	Ámbito	Tipo de datos	Valor	
ETLCantidadRegistrosNuevos	1_Dim_Location	Int32	0	
ETLCantidadRegistrosModificados	1_Dim_Location	Int32	0	

- k) Usé el componente de **RowCount** para poder contar la cantidad de registros nuevos o modificados que se agregarán.



- l) Ahora configuraremos el destino OLE DB a la tabla **Dim\_Location** donde se almacenarán los datos que no son nuevos.

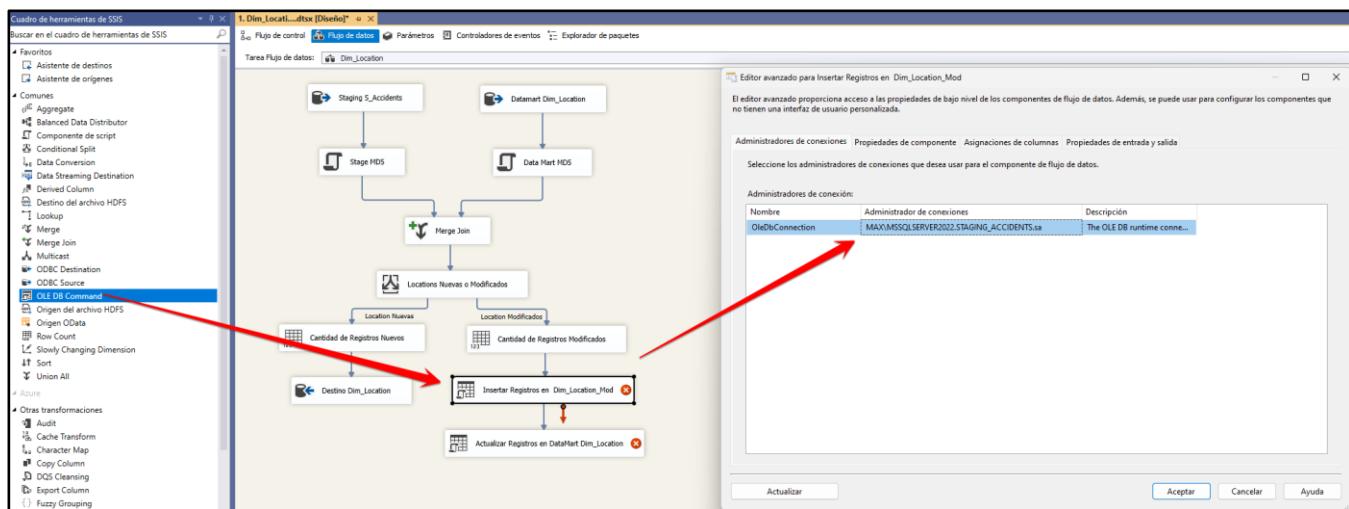


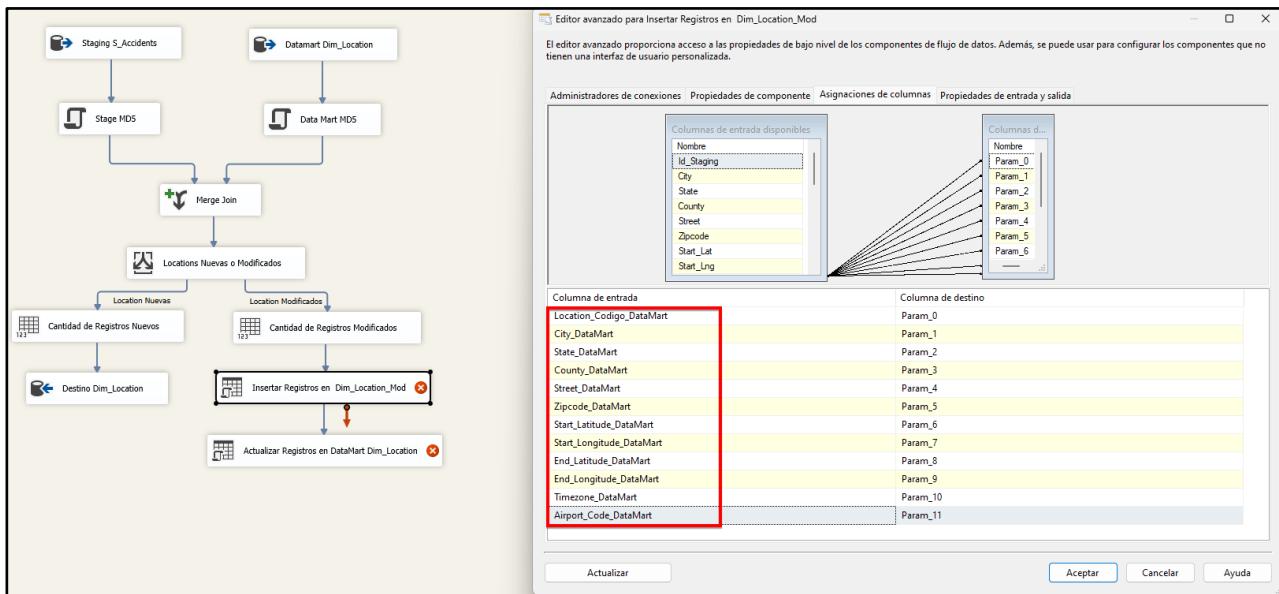
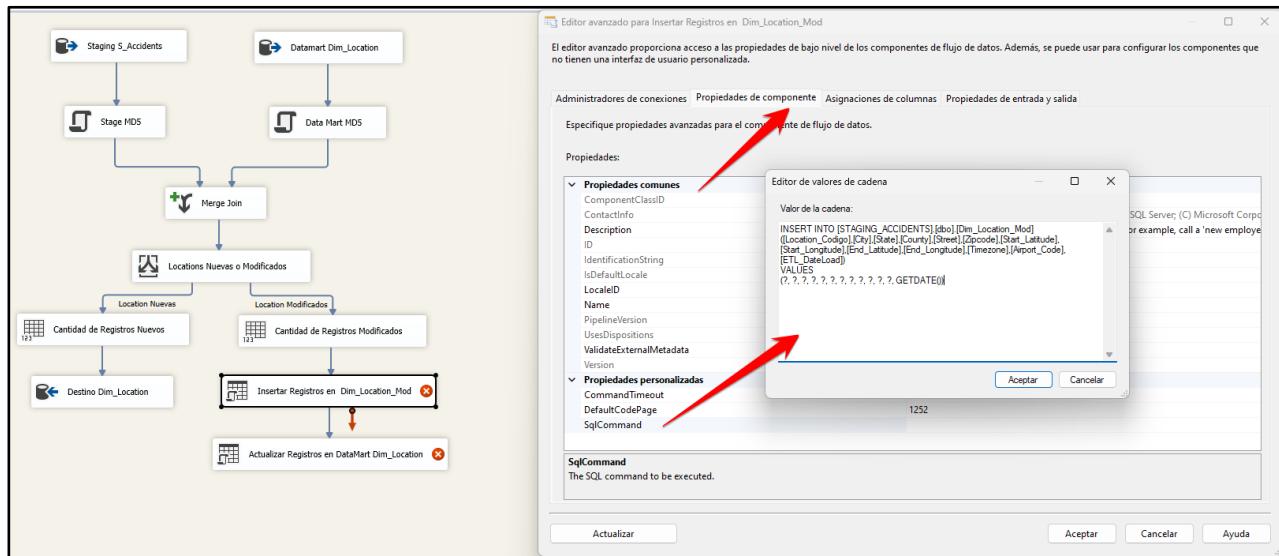


- m) Creé una tabla en la Base de Datos STAGING\_ACCIDENTS, la tabla de Dim\_Location\_Mod, donde se registrarán los registros que han sufrido modificación.

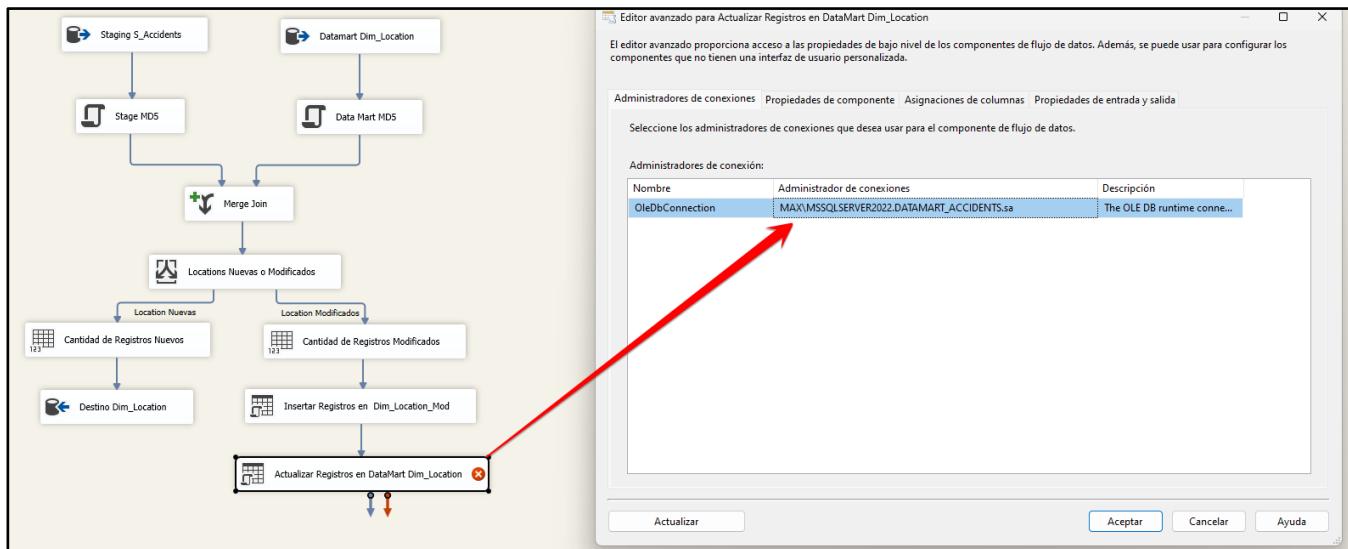
```
CREATE TABLE [STAGING_ACCIDENTS].[dbo].[Dim_Location_Mod] (
    Location_Key INT PRIMARY KEY IDENTITY(1,1),
    Location_Codigo INT,
    City VARCHAR(100),
    State VARCHAR(50),
    County VARCHAR(100),
    Street VARCHAR(500),
    Zipcode VARCHAR(50),
    Start_Latitude FLOAT,
    Start_Longitude FLOAT,
    End_Latitude FLOAT,
    End_Longitude FLOAT,
    Timezone VARCHAR(50),
    Airport_Code VARCHAR(50),
    ETL_DateLoad DATETIME
);
```

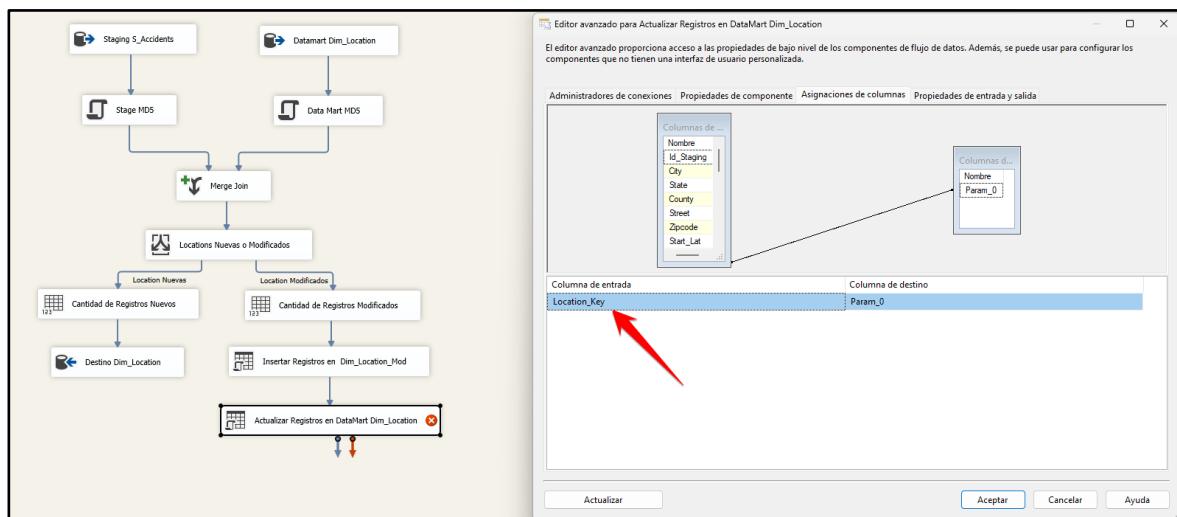
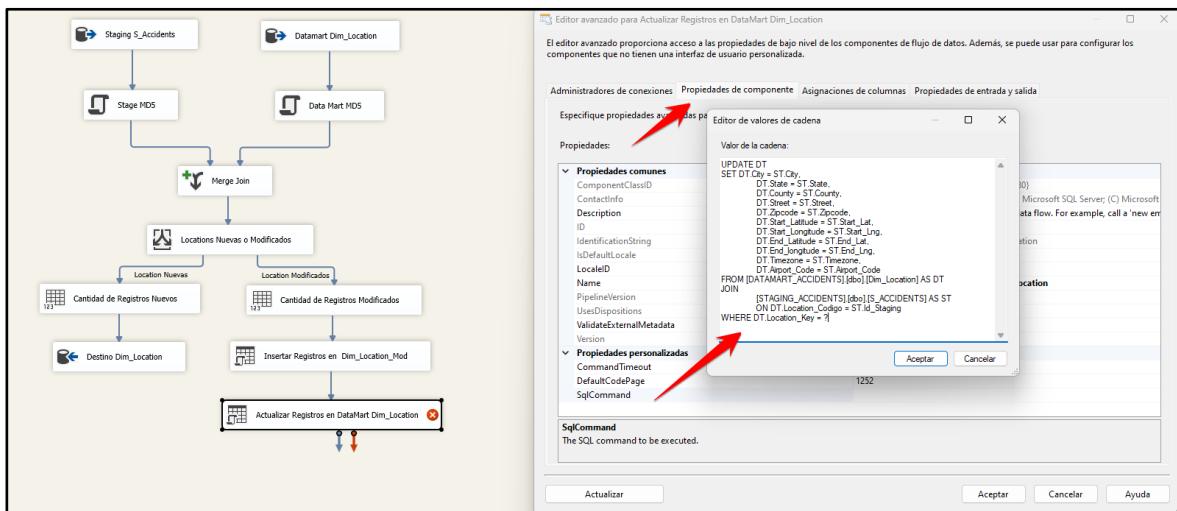
- n) Inserté dos componente de OLE DB Command, el primero agregará los registros modificados en la tabla Dim\_Location\_Mod.



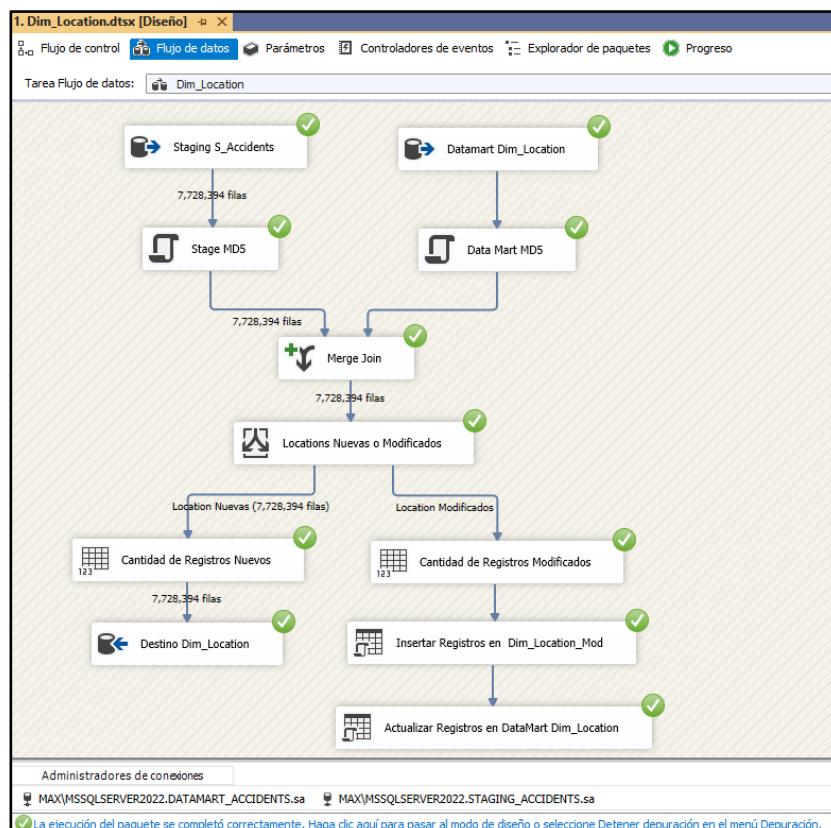


o) El segundo Componente OLE DB actualizará los registros en Dim\_Location.





p) Ejecutamos el proyecto de Dim\_Location, y repetimos el mismo proceso con las demás dimensiones.



## 2. Poblando la Dimensión Clima:

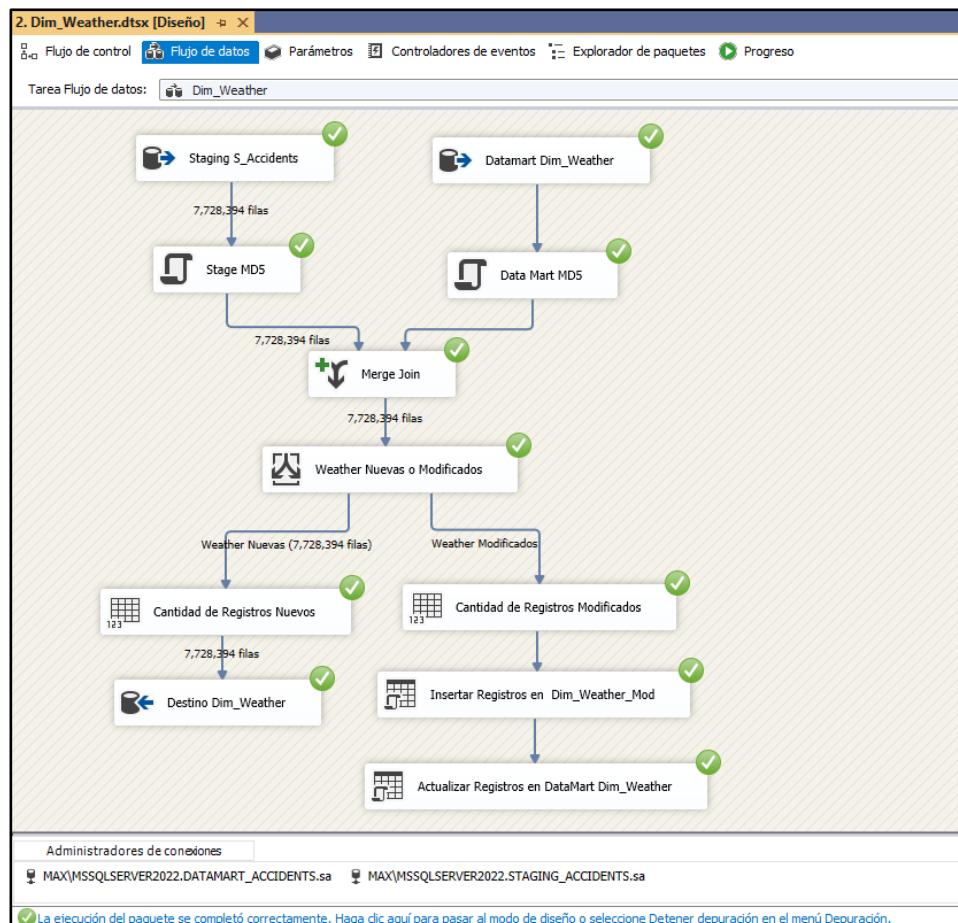
### a) Creación de la Tabla Dim\_Weather.

```
-- Dimensión: Clima
CREATE TABLE Dim_Weather (
    Weather_Key INT PRIMARY KEY IDENTITY(1,1),
    Weather_Codigo INT,
    Temperature FLOAT,
    Wind_Chill FLOAT,
    Humidity FLOAT,
    Pressure FLOAT,
    Visibility FLOAT,
    Wind_Direction VARCHAR(50),
    Wind_Speed FLOAT,
    Precipitation FLOAT,
    Weather_Condition VARCHAR(100)
);
```

### b) Creando Tabla Dim\_Weather\_Mod.

```
CREATE TABLE [STAGING_ACCIDENTS].[dbo].[Dim_Weather_Mod] (
    Weather_Key INT PRIMARY KEY IDENTITY(1,1),
    Weather_Codigo INT,
    Temperature FLOAT,
    Wind_Chill FLOAT,
    Humidity FLOAT,
    Pressure FLOAT,
    Visibility FLOAT,
    Wind_Direction VARCHAR(50),
    Wind_Speed FLOAT,
    Precipitation FLOAT,
    Weather_Condition VARCHAR(100),
    ETL_DateLoad DATETIME
);
```

### c) Ejecutando el proyecto.



## 3. Poblando la Dimensión Características de la Carretera:

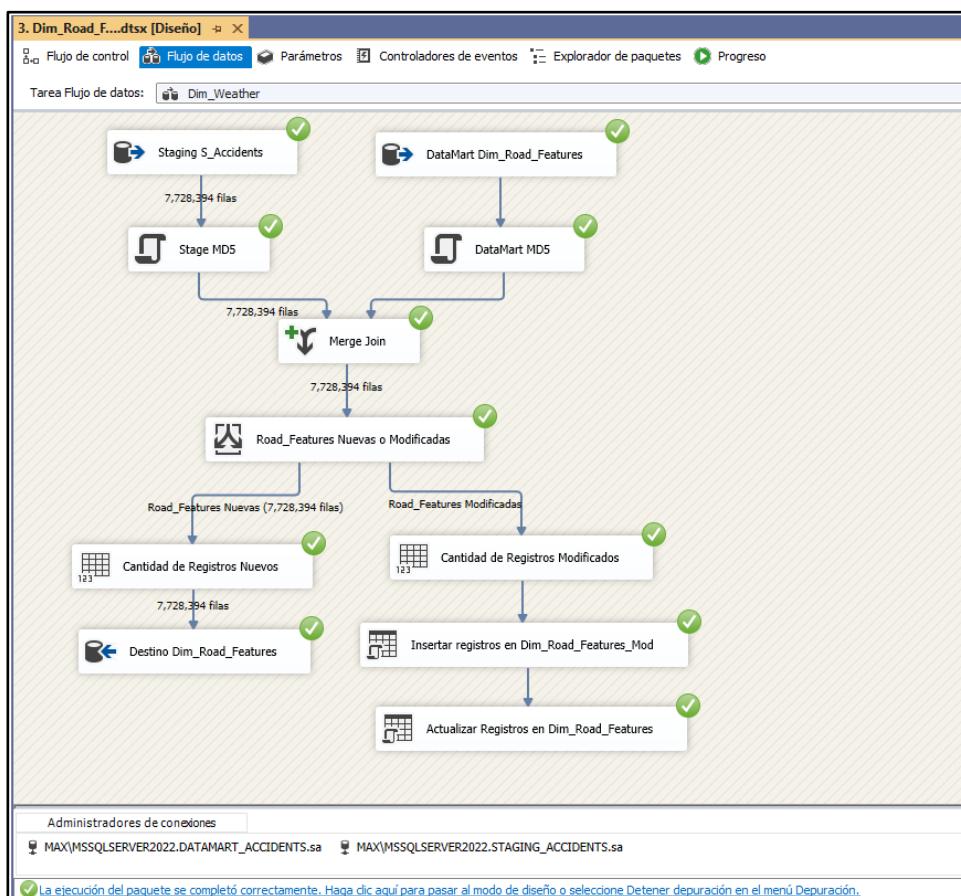
a) Creación de la Tabla **Dim\_Road\_Features**.

```
CREATE TABLE Dim_Road_Features (
    Road_Features_Key INT PRIMARY KEY IDENTITY(1,1),
    Road_Features_Codigo INT,
    Bump VARCHAR(10),
    Crossing VARCHAR(10),
    Give_Way VARCHAR(10),
    Junction VARCHAR(10),
    No_Exit VARCHAR(10),
    Roundabout VARCHAR(10),
    Stop VARCHAR(10),
    Traffic_Signal VARCHAR(10),
    Turning_Loop VARCHAR(10)
);
```

b) Creando Tabla **Dim\_Road\_Features\_Mod**.

```
CREATE TABLE [STAGING_ACCIDENTS].[dbo].[Dim_Road_Features_Mod] (
    Road_Features_Key INT PRIMARY KEY IDENTITY(1,1),
    Road_Features_Codigo INT,
    Bump VARCHAR(10),
    Crossing VARCHAR(10),
    Give_Way VARCHAR(10),
    Junction VARCHAR(10),
    No_Exit VARCHAR(10),
    Roundabout VARCHAR(10),
    Stop VARCHAR(10),
    Traffic_Signal VARCHAR(10),
    Turning_Loop VARCHAR(10),
    ETL_DateLoad DATETIME
);
```

c) Ejecutando el proyecto.



#### 4. Poblando la Dimensión Descripción de los Accidentes:

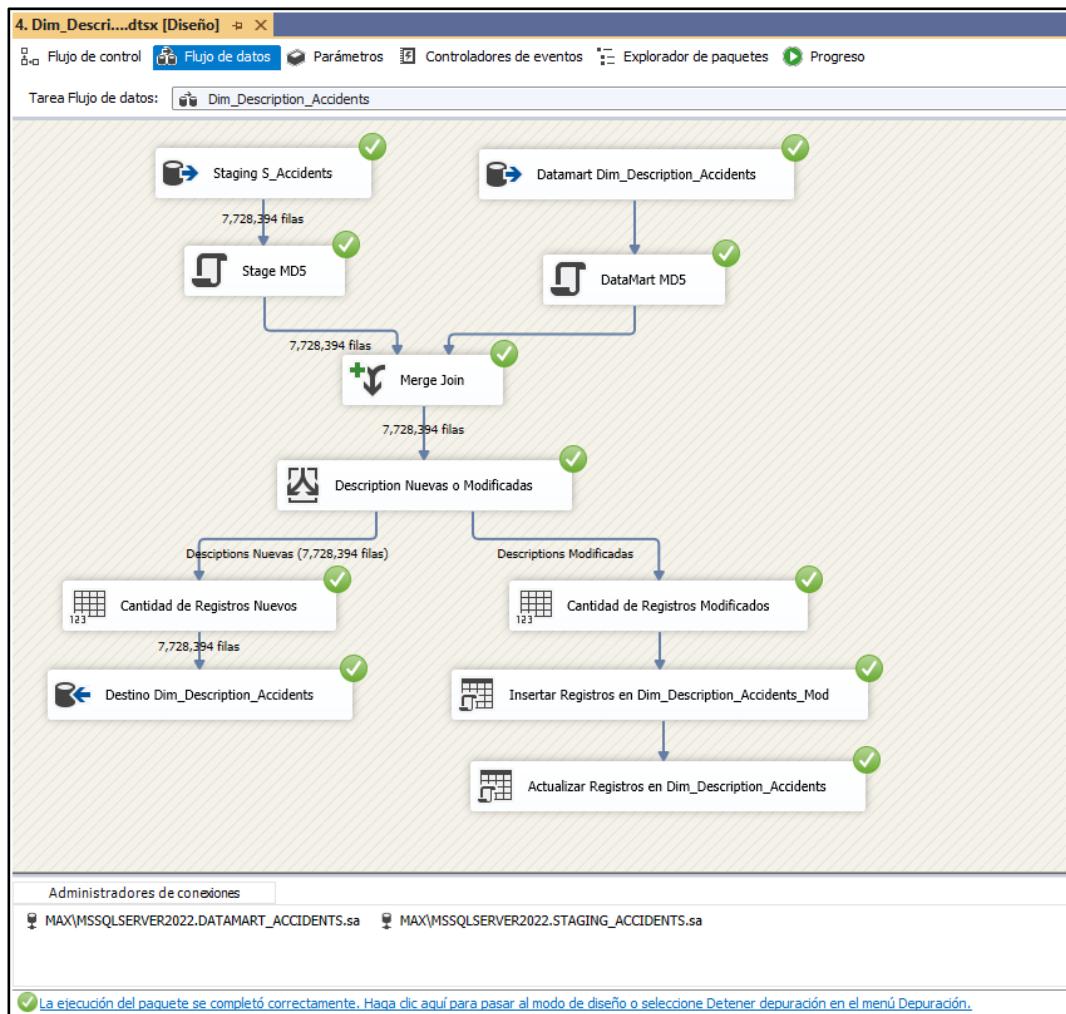
##### a) Creación de la Tabla **Dim\_Description\_Accidents**.

```
CREATE TABLE Dim_Description_Accidents (
    Description_Accidents_Key INT PRIMARY KEY IDENTITY(1,1),
    Description_Accidents_Codigo INT,
    Description VARCHAR(800)
);
```

##### b) Creando Tabla **Dim\_Description\_Accidents\_Mod**.

```
CREATE TABLE [STAGING_ACCIDENTS].[dbo].[Dim_Description_Accidents_Mod] (
    Description_Accidents_Key INT PRIMARY KEY IDENTITY(1,1),
    Description_Accidents_Codigo INT,
    Description VARCHAR(800),
    ETL_DateLoad DATETIME
);
```

##### c) Ejecutando el proyecto.



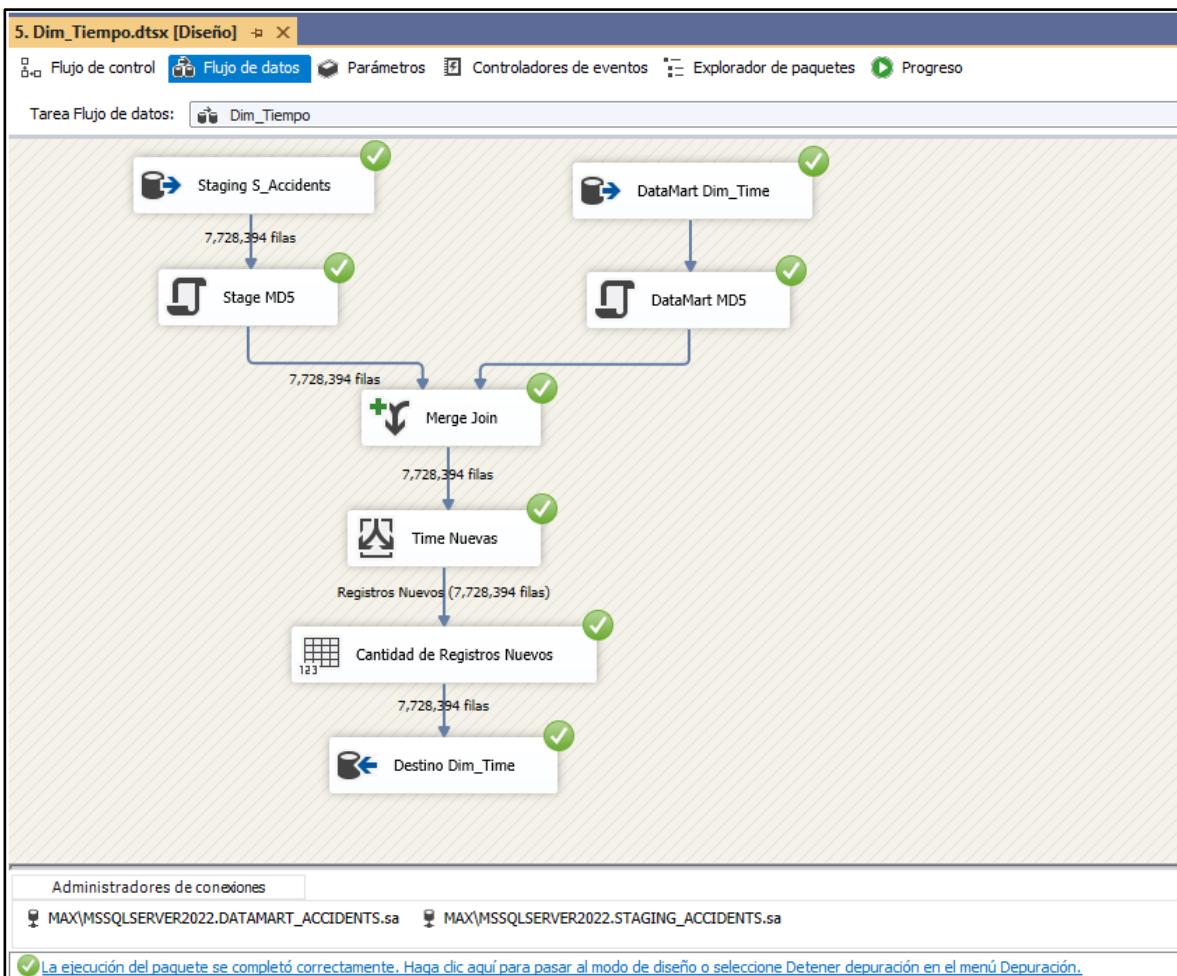
## 5. Poblando la Dimensión Tiempo:

### a) Creación de la Tabla Dim\_Tiempo.

```
CREATE TABLE [DATAMART_NORTHWND].[dbo].[Dim_Time]
(
    [Tiempo_SKey] INT IDENTITY(1,1) NOT NULL,
    [Tiempo_Codigo] INT,
    [Tiempo_FechaInicio] DATETIME,
    [Tiempo_FechaFin] DATETIME,
    [Tiempo_Anio] INT,
    [Tiempo_Trimestre] INT,
    [Tiempo_Mes] INT,
    [Tiempo_Semana] INT,
    [Tiempo_DiaDeAnio] INT,
    [Tiempo_DiaDeMes] INT,
    [Tiempo_DiaDeSemana] INT,
    [Tiempo_EsFinSemana] INT,
    [Tiempo_EsFeriado] INT,
    [Tiempo_Comentarios] VARCHAR(20),
    [Tiempo_SemanaCalendario] INT,
    [Tiempo_SemanasDelAnioLaborales] INT,
    [Tiempo_AnioBisiesto] INT,
    [Hour_Inicio] TIME,
    [Hour_Fin] TIME
);
```

- b) En la dimensión Tiempo se encontrarán solo Registros Nuevos y NO Registros Modificados por lo que no crearé una tabla para los modificados.

### c) Ejecutando el proyecto.



## 6. Poblando la Tabla Hechos:

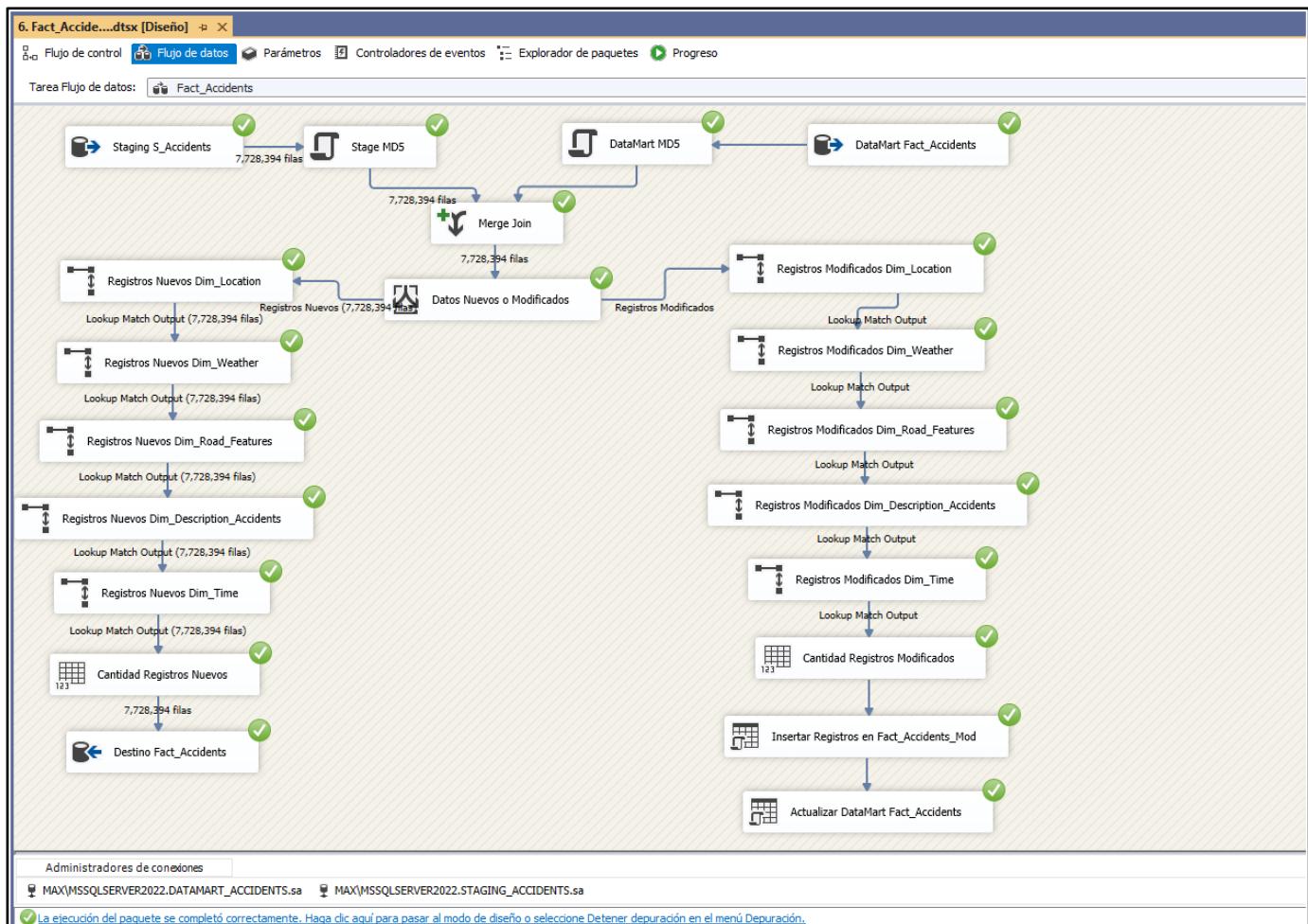
### a) Creación de la Tabla Fact\_Accidents.

```
CREATE TABLE Fact_Accidents (
    Fact_Accidents_Key INT PRIMARY KEY IDENTITY(1,1),
    Location_Key INT NOT NULL,
    Weather_Key INT NOT NULL,
    Road_Features_Key INT NOT NULL,
    Description_Accidents_Key INT NOT NULL,
    Tiempo_Skey INT NOT NULL,
    Distance FLOAT,
    Severity INT,
    Duration_minutes INT
    FOREIGN KEY (Location_Key) REFERENCES Dim_Location(Location_Key),
    FOREIGN KEY (Weather_Key) REFERENCES Dim_Weather(Weather_Key),
    FOREIGN KEY (Road_Features_Key) REFERENCES Dim_Road_Features(Road_Features_Key),
    FOREIGN KEY (Description_Accidents_Key) REFERENCES Dim_Description_Accidents(Description_Accidents_Key),
    FOREIGN KEY (Tiempo_Skey) REFERENCES Dim_Time(Tiempo_Skey)
);
```

### b) Creando Tabla Fac\_Accidents\_Mod.

```
CREATE TABLE [STAGING_ACCIDENTS].[dbo].[Fact_Accidents_Mod] (
    Fact_Accidents_Key INT PRIMARY KEY IDENTITY(1,1),
    Location_Key INT NOT NULL,
    Weather_Key INT NOT NULL,
    Road_Features_Key INT NOT NULL,
    Description_Accidents_Key INT NOT NULL,
    Tiempo_Skey INT NOT NULL,
    Distance FLOAT,
    Severity INT,
    Duration_minutes INT,
    ETL_DateLoad DATETIME
);
```

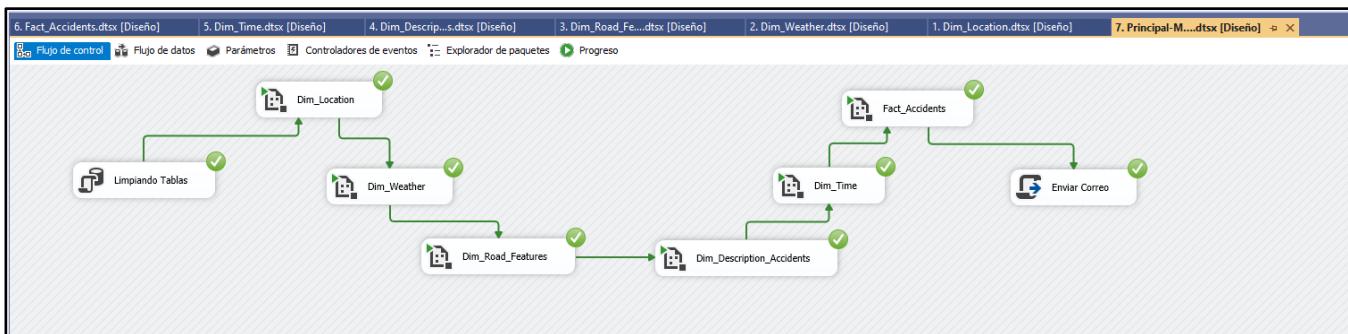
### c) Ejecutando el proyecto.



## 7. Proyecto Principal y Envío de Correo.

En este último paso, crearé un **Proyecto llamado PRINCIPAL-MASTER**, el cuál será el encargado de **Limpiar las Tablas** (Considerándolo que será la primera vez que se pasarán los datos y como tuve data de prueba, lo prefiero borrar y ejecutarlo todo desde el inicio), **luego ejecutará cada Poblado de Dimensión y de la tabla Hechos**, y al final me enviará un correo indicando si todo se realizó correctamente. Es importante mencionar que el envío de correo lo realicé mediante el componente de **Tarea Script**, debido a que se hará de manera local, en un ambiente de producción si se utilizaría el Componente de Tarea Enviar Correo utilizando un servidor de correo y su respectiva configuración.

- Ejecutamos todo el proyecto.



- Se observa que todo se ejecutó correctamente, por lo tanto, me llegó el correo indicando que todo el proceso terminó exitosamente.

